

Tool Support for Incremental Failure Mode and Effects Analysis of Component-Based Systems

Jonas Elmqvist and Simin Nadjm-Tehrani^{*†}
Department of Computer and Information Science
Linköping University
Linköping, Sweden
{jonel, simin}@ida.liu.se

Abstract

Failure Mode and Effects Analysis (FMEA) is a well-known technique widely used for safety assessment in the area of safety-critical systems. However, FMEA is traditionally done manually which makes it both time-consuming and costly, specially for large and complex systems. Also, small modifications in the design may result in a complete revision of the initial FMEA.

This paper presents a tool support for automated incremental component-based FMEA of SW and HW. It is based on component safety interfaces and a formal compositional safety analysis method. This tool support enables engineers to focus on more important steps in the safety assessment process. Also, during system upgrades, the tool incrementally registers the changes and identifies possible effects in the FMEA which enables the use of earlier safety analysis results. Finally, this formal approach based on design models of the components and the system always creates FMEAs which are consistent with the system design.

1 Introduction

Many safety-critical domains such as the automotive and aerospace industries are facing an increase in the complexity in hardware and software. This is the effect of the cost-efficient introduction of more advanced features in cars and aircrafts and replacing mechanical functions with software and hardware. A big challenge in these industries is now to lower the costs and decrease time-to-market while coping with the increased complexity and ensuring safety.

One way of dealing with the problems of increased complexity in the design is component-based system develop-

ment [17, 4] (CBSD) which promotes compositional development and component reuse. However, the use of commercial-off-the-shelf components (COTS) in safety-critical system is highly unexplored. There are no obvious methods to incorporate knowledge about functions and resilience of COTS in safety assessments.

Traditional methods such as Failure Mode and Effects Analysis (FMEA) and Fault-Tree Analysis (FTA) [10] are well known in the reliability engineering discipline. However, these techniques become intractable to use since the complexity of the systems makes failure propagation hard to derive and often produces extremely large fault trees. One way of dealing with the increased complexity in safety assessment is to integrate the two separate activities of design and safety analysis through introduction of formal methods [3, 9]. With this approach, the safety assessment is based on the system design model augmented with formal failure models. This model-based approach enables verification tools, such as model checkers, to automatically check if the system design tolerates the modelled failures at design time.

A novel attempt to combine the component-based approach with safety-critical systems while integrating design and safety analysis uses the concept of *safety interfaces* [6, 5]. Briefly, the safety interface can be seen as a formal description of the component in terms of failures directly affecting the component. This approach enables a compositional safety-analysis where the safety interfaces of component are used for overall system safety analysis. The compositional approach to safety analysis becomes efficient upon component reuse and upgrades since earlier safety analysis results can be partially reused during the new safety analysis. However, for practical use, the use of safety interfaces needs to be incorporated in current engineering methods and tools.

In this paper we present a tool support for an automatic synthesis of FMEA from the safety interfaces of the

^{*}This work was supported by the the National Aerospace Research Programme (NFFP), and the Swedish Strategic Research Foundation (SSF) supported project SAVE.

[†]The second author has been partially supported by University of Luxembourg during the preparation of this manuscript.

components in the system while performing a system-level safety analysis. This framework also supports an incremental FMEA upon reuse of components, where the effect of changes in the component assembly are automatically updated in the FMEA table. The process is automated by a prototype tool that generates FMEA matrices and the method is evaluated by application to a hydraulic control system in an aerospace application.

2 Related work

There are a number of approaches addressing the problems related to manual FMEA using different techniques for automatic generation of FMEA tables. However, most of these approaches does not use any formal design models to generate the FMEA. Instead, they require safety engineers to specify impact of component failures on the system. This requires in-depth knowledge about all individual components in the system.

Grunske et. al. [7] proposes a method for automated FMEA. In their approach, FMEA tables are built from engineering diagrams that have been augmented with information about component failures. Safety analysis is performed on the composed system and it requires explicit modelling of failure behaviour and propagation inside a component. Our approach is based on support for automatic generation of safety interfaces and enables compositional reasoning.

Papadopolous et. al. [11] extend a functional model of a design with Interface-Focused FMEA. In [12, 13], the authors propose an automated FMEA. However, their approach requires a special characterization of failure logic (failure annotations) for each component while our approach is based on the actual design model. Their approach also lacks formal verification support and suffers from combinatorial explosion in large fault trees.

The ESACS project [3] applied a model-based safety analysis approach using Statechart models. Within this project, a tool (FSAP/NuSMV [2]) for automating fault injection, automatic fault tree construction and failure ordering analysis has been developed. However, it only generates flat fault tree structures (and-or) which may become inconvenient for large systems, and the approach does not support compositional safety assessment.

Raheja [16] presents a method called Software System FMEA (SSFMEA) which is an extended form of FMEA for software. However, the approach is not component-based and the analysis is not incremental.

A number of approaches for automated FMEA has been proposed for use in analyzing electrical systems [15, 14, 18]. Some of these, for example [15, 18], have an incremental approach, but are based on simulation to generate the FMEA. This makes the method impractical for complex systems with large state spaces since it will become impossible to simulate every state in the system. Our approach of

using model checking promises adherence to a given safety property and computes efficiently.

3 Synthesizing FMEA

In previous work, we have presented a method for generating safety interfaces for components [6]. The safety interface generation tool serves as a front-end to Esterel Studio and uses the built in model checker to verify the design and to analyse the failure effects [5]. This section outlines the procedure for analysing the effect of failure modes at system level and deriving the cFMEA table. It also describes how system FMEA are generated and how incremental analysis is supported.

3.1 Component FMEA (cFMEA)

Given a set of fault modes and a system level safety property, the safety interface of a component captures the behaviour of the component in presence of faults in its environment w.r.t to the safety property. The safety interface makes explicit which single and double faults the component can tolerate. For every fault mode in the safety interface, there is a corresponding assumption that needs to be fulfilled by the environment in order for the component to be resilient to the fault mode. The main purpose of the assumptions is to enable formal assume guarantee reasoning on the system assembly in order to analyse overall safety.

However, since safety interfaces only specify which fault modes it might tolerate, there are fault modes that we still need to consider in the overall safety analysis that are omitted in the safety interfaces. Thus, the procedure of generating cFMEA tables must consider *every fault mode* affecting the component, both those in the safety interface and those not in the safety interfaces.

For every fault mode in the safety interface, we generate a row in the cFMEA matrix. By traversing the elements in the safety interface, we can extract the necessary and useful information to be placed in the different columns in the cFMEA row. First, we add the name identifier and the type of the fault mode. At this stage, we also create a description field in the FMEA row which enables safety engineers to add a complete description of the fault mode. Then, we need to consider what possible effect the fault mode will have. Since by definition of the safety interface, each failure mode is analysed w.r.t to a safety property φ , the violation of φ is also the possible effect. Thus, we need to consider the assumptions that are needed for the component to be resilient to the failure mode. If the component is resilient to the failure mode in any environment, then the failure mode will not effect φ .

However, the assumptions can contain a very large number of variable constraints. In the current version of the front-end to Esterel [5] these assumptions are generated as

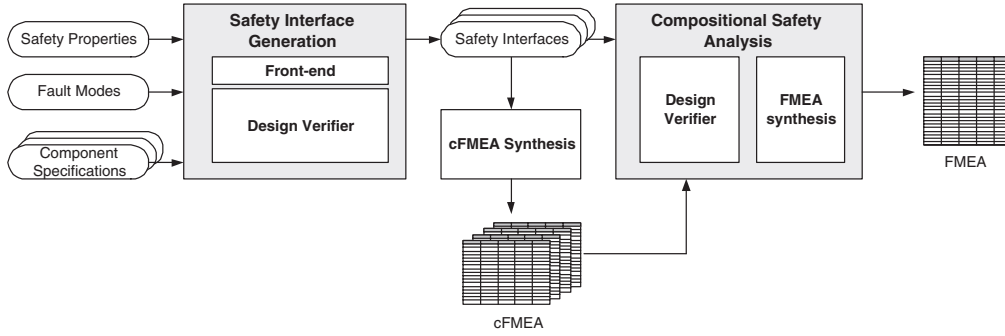


Figure 1. The tool chain.

formulas constraining the input variables. This format is not optimal for reading and understanding the context. Thus, we transform the assumptions to something more useful for engineers and store them in the cFMEA table. One transformation is to extract those variables that the assumption is constraining. This will at least show the safety engineer on what components and signals the component under study is directly relying.

For those faults that do not exist in the safety interface, we create a FMEA row but add “Unknown” to the effect column since we do not know the effect. This marks this failure mode as a failure that needs to be reconsidered by other means.

Also, we add cells for the traditional properties used in FMEA such as severity, probability, criticality, likelihood for detection. However, no knowledge about these concepts is stored in the safety interface, but can be added later in the safety assessment process by safety engineers.

3.2 System-Level FMEA

The system FMEA is constructed using information in each cFMEA and the partial results from the compositional safety analysis. Every cFMEA is parsed for information, and while the compositional safety analysis produces results, the system FMEA is generated.

System level safety analysis investigates the effect of every single fault in the system and every double fault. The local effect of failure modes are found in the safety interfaces, together with the assumptions that the component makes on the environment. By using compositional proof rules[6], these known assumptions of every component can be checked considering other components as part of the environment.

Failures modes that can lead to an unsafe state are marked with a red icon. Failure modes that cannot directly contribute to a hazardous event are marked green.

3.3 Incremental FMEA

The automated component FMEA helps system developers to study the effect of propagation of a fault mode from a component input towards a change in the outputs that violates a given safety property φ . However, the effect of changing or upgrading a component in a safety-critical system often results in a complete revision of the initial FMEA. This is due to the fact that the traditional FMEA is not supported by formal methods and the previous results must be proven to hold in the new system.

In our approach, a new component comes with its (automatically generated) safety interface. When a component is replaced, the new safety interface is first used to generate a new cFMEA, as described in Section 4.2. By inspecting the cFMEA of the new component, and comparing with the old cFMEA, we can get an initial intuition if the new component will affect overall safety in the system. Our prototype tool at least identifies differences in the assumptions on a syntactical level and marks these rows with a yellow warning sign.

Also, since our system-level safety analysis is compositional, a change in one component essentially creates a need for repeating part of the analysis at the system level. What needs to be rechecked due to the ripple effects is identified by the compositional rules used in the analysis. Partial results from the initial analysis can be reused, which is not the case for traditional FTA or FMEA.

In our approach, since FMEA tables are constructed from the component design model, the FMEA remains consistent. Hand written FMEA tables are both error prone and could create inconsistencies between the real design and the analysed failure behaviour.

4 Tool support

The tool chain supporting our framework is shown in Fig 1. The generation of safety interfaces is automatically done using a front-end to Esterel studio [6]. The output of this tool is safety interfaces for every component. In this paper we present a prototype developed in Java that reads

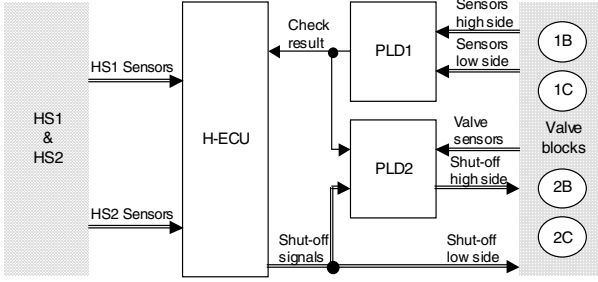


Figure 2. The hydraulic leakage detection system.

component safety interfaces, and produces cFMEA tables. As of now, the output is parsed and displayed in a table in the application, but can also be exported into XML-format which enables a wide range of options for further usage. The XML-file can be nicely visualised using XSL (the eX-tensible Stylesheet Language) as a table in a web browser, or can be imported into any FMEA tool that supports XML import.

Using the safety interfaces for compositional analysis is supported by a previously developed safety analysis tool that also uses the built in design verifier in Esterel Studio. Extending our FMEA generator and combining with the current safety analysis tool, the FMEA table can be incrementally generated using the cFMEA tables of the components and the partial results from the safety analysis tool.

5 Case study: Hydraulic System

As a proof of concept we have applied our method to the leakage detection subsystem of the hydraulic system of the JAS 39 Gripen multi-role aircraft, obtained from Saab Aerosystems. The system has been decomposed into components (see Figure 2). Both the original system model presented in [9] and our component-based version are described in the language of Esterel [1], a synchronous language whose compiler ensures the nonblocking property upon composition. For a short description of the functionality and the safety requirements, see [6].

5.1 Fault modes

A total of eighteen fault modes were identified (presented in Table 1). In an earlier study fifteen fault modes were studied which included both physical faults in the four shut-off valves but also random faults in the three components [8]. However, since we omit the valves in our analysis (consider them to be a part of the environment) some of the faults identified in the original analysis do not exist, while we introduce some new faults in our system.

For each signal in the system, a corresponding fault

<i>Fault</i>	<i>Type</i>	<i>Comp.</i>	<i>Input</i>
F_1	Random	H-ECU	CheckOK
F_2	Random	PLD1	SensorHigh_1B
F_3	Random	PLD1	SensorHigh_1C
F_4	Random	PLD1	SensorHigh_2B
F_5	Random	PLD1	SensorHigh_2C
F_6	Random	PLD1	SensorLow_1B
F_7	Random	PLD1	SensorLow_1C
F_8	Random	PLD1	SensorLow_2B
F_9	Random	PLD1	SensorLow_2C
F_{10}	Random	PLD2	CheckOK
F_{11}	Random	PLD2	ShutOffRequest_1B
F_{12}	Random	PLD2	ShutOffRequest_1C
F_{13}	Random	PLD2	ShutOffRequest_2B
F_{14}	Random	PLD2	ShutOffRequest_2C
F_{15}	Random	PLD2	ValveSensor_1B
F_{16}	Random	PLD2	ValveSensor_1C
F_{17}	Random	PLD2	ValveSensor_2B
F_{18}	Random	PLD2	ValveSensor_2C

Table 1. Identified possible faults in the hydraulic system

mode was identified. Every fault mode in the system was classified as a random fault. Since all signals in the system are pure (i.e. Boolean), a random fault can non-deterministically switch the signal from high to low and vice versa. In practice, this is modelled by setting this signal as a free (un-restricted variable) to the affected component.

5.2 Automatic cFMEA generation

The production of the cFMEA tables for each component generates FMEA tables for every single fault and combination of double faults. Fig. 3 shows the result of the single fault cFMEA of component PLD2 as presented in the tool. The green icons in the left cell indicate that the component is resilient to that failure mode, i.e. that the failure mode will not have any effect on overall safety in terms of safety property φ' .

Note that one cFMEA is created for every component, and every system typically has many safety properties. In this case, since we only consider one safety property φ , one failure mode is only analysed in terms of one safety property. When analysing many safety properties, one failure mode can have different impacts on withholding different safety properties, thus appearing in many rows in the cFMEA.

5.3 System-level FMEA generation

System-level safety analysis is performed with the safety interfaces of PLD1, PLD2, and H-ECU as input. While

cFMEA - Component name: PLD2

Failure M...	Type	Description	Cause	Effect	Relying Inputs	Criticality	Probability	
F_10	Random	Random failure in CheckOK	-	no	-	-	-	✓
F_11	Random	Random failure in ShutOffRequest_1B	-	no	-	-	-	✓
F_12	Random	Random failure in ShutOffRequest_1C	-	no	-	-	-	✓
F_13	Random	Random failure in ShutOffRequest_2B	-	no	-	-	-	✓
F_14	Random	Random failure in ShutOffRequest_2C	-	no	-	-	-	✓
F_15	Random	Random failure in ValveSensor_1B	-	no	-	-	-	✓
F_16	Random	Random failure in ValveSensor_1C	-	no	-	-	-	✓
F_17	Random	Random failure in ValveSensor_2B	-	no	-	-	-	✓
F_18	Random	Random failure in ValveSensor_2C	-	no	-	-	-	✓
F_10, F_11	Random	Combination of F10 and F11	-	no	-	-	-	✓
F_10, F_12	Random	Combination of F10 and F12	-	no	-	-	-	✓

Figure 3. The cFMEA for PLD2 (only subset of all faults).

System FMEA - System name: Hydraulic System | Components: H-ECU, PLD1, and PLD2

Component	Failure Mode	Type	Description	Cause	Effect	Relying Inputs	Criticality	Probability	
PLD1	F_7	Random	Random failure in CheckOK	-	no	-	-	-	✓
PLD1	F_8	Random	Random failure in ShutOffRequest_1B	-	no	-	-	-	✓
PLD1	F_9	Random	Random failure in ShutOffRequest_1C	-	no	-	-	-	✓
PLD2	F_10	Random	Random failure in CheckOK	-	no	-	-	-	✓
PLD2	F_11	Random	Random failure in ShutOffRequest_1B	-	no	-	-	-	✓
PLD2	F_12	Random	Random failure in ShutOffRequest_1C	-	no	-	-	-	✓
PLD2	F_13	Random	Random failure in ShutOffRequest_2B	-	no	-	-	-	✓
PLD2	F_14	Random	Random failure in ShutOffRequest_2C	-	no	-	-	-	✓
PLD2	F_15	Random	Random failure in ValveSensor_1B	-	no	-	-	-	✓
PLD2	F_16	Random	Random failure in ValveSensor_1C	-	no	-	-	-	✓
PLD2	F_17	Random	Random failure in ValveSensor_2B	-	no	-	-	-	✓
PLD2	F_18	Random	Random failure in ValveSensor_2C	-	no	-	-	-	✓
PLD1, PLD2	F_1, F_10	Random	Combination of F1 and F10	-	no	-	-	-	✓
PLD1, PLD2	F_1, F_11	Random	Combination of F1 and F11	-	no	-	-	-	✓

Figure 4. The FMEA for the whole system (only subset of all faults).

cFMEA - Component name: PLD2

Failure Mode	Type	Description	Cause	Effect	Relying Inputs	Criticality	Probability	
F_10	Random	Random failure in CheckOK	-	Invalid safety property	ShutOffLow_1B, Shu...	-	-	?
F_11	Random	Random failure in ShutOffRequest_1B	-	Invalid safety property	ShutOffLow_1B, Shu...	-	-	?
F_12	Random	Random failure in ShutOffRequest_1C	-	Invalid safety property	ShutOffLow_1B, Shu...	-	-	?
F_13	Random	Random failure in ShutOffRequest_2B	-	Invalid safety property	ShutOffLow_1B, Shu...	-	-	?
F_14	Random	Random failure in ShutOffRequest_2C	-	Invalid safety property	ShutOffLow_1B, Shu...	-	-	?
F_15	Random	Random failure in ValveSensor_1B	-	Invalid safety property	ShutOffLow_1B, Shu...	-	-	?
F_16	Random	Random failure in ValveSensor_1C	-	Invalid safety property	ShutOffLow_1B, Shu...	-	-	?
F_17	Random	Random failure in ValveSensor_2B	-	Invalid safety property	ShutOffLow_1B, Shu...	-	-	?
F_18	Random	Random failure in ValveSensor_2C	-	Invalid safety property	ShutOffLow_1B, Shu...	-	-	?
F_10, F_11	Random	Combination of F10 and F11	-	Invalid safety property	ShutOffLow_1B, Shu...	-	-	?
F_10, F_12	Random	Combination of F10 and F12	-	Invalid safety property	ShutOffLow_1B, Shu...	-	-	?

Figure 5. The cFMEA for PLD2' (only subset of all faults).

analysing system-level fault resilience of every single and double fault, the system FMEA is automatically generated. Information is received from each cFMEA and from the analysis results in order to generate each FMEA row in the table.

The resulting system FMEA can be seen in Fig. 4. In this table, we can see that no single or double fault of a single component will cause a threat to system-level safety. Also, the system is resilient to any double fault from multiple components.

5.4 Incremental FMEA

Now let's assume that we need to update PLD2. Thus, we create a component PLD2'. However, the updated component PLD2' does not work as intended. PLD2 should check all possible shut-off requests and ensure that no two valves will close at the same time. However, the developer of PLD2' has forgotten to check one of the combinations of shut-off requests from the H-ECU. The automatically generated cFMEA of the new component PLD2' can be seen in Fig 5.

In the cFMEA, we actually see the effect of the difference between PLD2 and PLD2'. The tool identifies which rows have been effected and highlights these with a yellow warning sign to the right. Every failure mode can now have the effect of not satisfying φ' which is also marked with the red warning signs to the right. Using the "Relying input" column, we can see that PLD2' for example is relying on the inputs of ShutOffLow_1B and ShutOffLow_2B in order to satisfy the safety property. Further in-depth analysis on exactly how these signals effect safety is done by analysing the safety interface, in particular the assumptions for every failure mode.

When this change has been done in the system, a new system-level safety analysis can be done. According to the compositional rules in [6], only parts of the analysis is affected by the change which avoids a complete re-analysis of the system. While performing the safety analysis, the tool automatically generates the system FMEA table. The new FMEA table is compared to the initial table, but in this case no change is detected by the tool (i.e. the new FMEA looks like Figure 4). This means that the overall safety is **not** effected by replacing PLD2 with the faulty PLD2'.

6 Conclusions

Failure Mode and Effects Analysis serves as an important part of the safety assessment process for safety-critical systems. However, due to the increased complexity of the systems, the process of FMEA becomes troublesome, both in terms of time but also actually identifying the effect of every failure mode.

In this paper we have presented an automatic approach to incremental FMEA of component-based systems. The

approach is based on the formal framework of safety interfaces and compositional safety analysis. This practical use of safety interfaces for analysing component-based fault tolerant systems has many benefits:

- It eliminates much of the manual work of creating FMEA tables for the engineers, enabling them to focus on more important steps in the safety assessment process, such as failure identification or failure mitigation actions.
- Deriving component and system FMEA tables using their design models, the resulting FMEA is always consistent with the system design.
- This approach automatically generates a FMEA table with qualitative information which may serve as basis for future quantitative analysis and eventual design decisions.
- Critical effects of new components in the system are automatically identified.
- The incremental approach decreases the analysis workload upon component upgrades and reuse since results of partial analysis at both component and system level can be reused.

This approach has been implemented in a prototype which is integrated with previously developed tools, creating a tool-chain from deriving component safety interfaces to the automatic generation of system FMEA tables.

More work is needed to fully utilise the potential of this approach. Future research directions include creating more advanced analysis of the comparison between component assumptions. This could enable an increased knowledge for integrators during component upgrades.

Also, adding quantitative reasoning to this framework would be a suitable extension. The use of probabilistic model checking could be one step towards an integrated qualitative and quantitative analysis framework.

Another step for future work would be to focus on common-cause faults. As of now, this approach does not handle those types of faults but their effect is of course an important part of the safety assessment.

References

- [1] G. Berry. *The Esterel v5 Language Primer*, v 5.91 edition, 2000.
- [2] M. Bozzano, A. Cavallo, M. Cifaldi, L. Valacca, and A. Villaforita. Improving safety assessment of complex systems: An industrial case study. In *FME 2003: Formal Methods: International Symposium of Formal Methods*, volume 2805 of *Lecture Notes in Computer Science*, pages 208–222. Springer Verlag, September 2003.

- [3] M. Bozzano and et al. ESACS: an integrated methodology for design and safety analysis of complex systems. In *ESREL 2003*, pages 237–245. Balkema, June 2003.
- [4] I. Crnkovic, J. A. Stafford, H. W. Schmidt, and K. C. Wallnau, editors. *Proceedings of 7th International Symposium on Component-Based Software Engineering (CBSE)*, Edinburgh, UK, May 2004. Springer Verlag.
- [5] J. Elmqvist and S. Nadjm-Tehrani. Safety-oriented design of component assemblies using safety interfaces. In *Third International Workshop on Formal Aspects of Component Software (FACS'06)*, pages 1–15, Prague, Czech Republic, September 2006. ENTCS.
- [6] J. Elmqvist, S. Nadjm-Tehrani, and M. Minea. Safety interfaces for component-based systems. In R. Winther, B. A. Gran, and G. Dahll, editors, *SAFECOMP*, volume 3688 of *Lecture Notes in Computer Science*, pages 246–260. Springer, 2005.
- [7] L. Grunske, P. A. Lindsay, N. Yatapanage, and K. Winter. An automated failure mode and effect analysis based on high-level design specification with behavior trees. In J. Romijn, G. Smith, and J. van de Pol, editors, *IFM*, volume 3771 of *Lecture Notes in Computer Science*, pages 129–149. Springer, 2005.
- [8] J. Hammarberg. High-level development and formal verification of reconfigurable hardware. Master's thesis, Linköpings University, November 2002.
- [9] J. Hammarberg and S. Nadjm-Tehrani. Formal verification of fault tolerance in safety-critical reconfigurable modules. *International Journal of Software Tools for Technology Transfer (STTT)*, 7(3), June 2005. Springer Verlag.
- [10] E. Henley and H. Kumamoto. *Reliability Engineering and Risk Assessment*. Prentice Hall, 1981.
- [11] Y. Papadopoulos, J. A. McDermid, R. Sasse, and G. Heiner. Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure. *Reliability Engineering and System Safety*, 71(3):229–247, 2001.
- [12] Y. Papadopoulos, D. Parker, and C. Grante. Automating the failure modes and effects analysis of safety critical systems. In *Proc. of the 8th IEEE International Symposium on High Assurance Systems Engineering (HASE'04)*, March 2004.
- [13] Y. Papadopoulos, D. Parker, and C. Grante. A method and tool support for model-based semi-automated failure modes and effects analysis of engineering designs. In *SCS '04: Proceedings of the 9th Australian workshop on Safety critical systems and software*, pages 89–95. Australian Computer Society, Inc., 2004.
- [14] C. Price and N. Taylor. Automated multiple failure FMEA. In *Reliability Engineering and System Safety*, volume 76, pages 1–10, 2002.
- [15] C. J. Price. Effortless incremental design FMEA. In *Annual Reliability and Maintainability Symposium*, pages 43–47, Las Vegas, NV, 1996. IEEE.
- [16] D. Raheja. Software system failure mode and effects analysis (SSFMEA) – a tool for reliability growth. In *International Symposium on Reliability and Maintainability (ISRM'90)*, pages 271–277, Tokyo, Japan, 1990.
- [17] C. Szyperski. *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley, 2nd edition, 2002.
- [18] D. Throop, J. Malin, and L. Fleming. Automated incremental design FMEA. In *IEEE Proceedings of Aerospace Conference*, volume 7, 2001.