AUTAS: a tool for supporting FMECA generation in aeronautic systems

Claudia Picardi[1], **Luca Console**[1], **Frederic Berger**[2], **Jan Breeman**[3], **Tony Kanakis**[3], **Jeroen Moelands**[3]
Stephan Collas[4], **Emmanuel Arbaretier**[4], **Nino De Domenico**[5], **Ermanno Girardelli**[5], **Oskar Dressler**[6]
Peter Struss[7], **Benjamin Zilbermann**[8]

Abstract. The goal of the FMECA (Failure Mode, Effects and Criticality Analysis) process is to determine the consequences that failures may have on the function of a complex system. In aeronautic industries, this process is very important and must comply with international standards. Today, most of the process is performed manually. This can be problematic, since, although the basic process is not difficult, taking into account all behaviors and all the interactions between the behaviors of several components of a system can be very complex, error prone and costly. In the paper, we discuss how MBR (Model Based Reasoning) can support the process and we present a software environment which implements this concept.

1 INTRODUCTION

FMECA (Failure Modes Effects and Criticality Analysis) is a technical process which is performed on engineered artifacts in order to analyse the consequences that failures may have on a system function and behavior. It is performed after each phase in the design process to verify that the system does not manifest critical behavior and loss of function whenever one of the components fails to operate properly. In case reliability requirements are not met, this should lead to a re-design of the system. A table reporting the FMECA should also be provided as a final documentation of the system behavior; this is important in case the system is used as a component of a more complex system (to perform the FMECA of the complex system) and as a documentation to be provided to people that will perform maintenance and diagnosis/repair of a system.

An example of a FMECA table is reported in figure 1. Each row of the table corresponds to a fault occurrence in a system component and it reports (among other things):

- the local effect of the fault (i.e. the effect at the level of the component itself);
- the next-higher level effect (i.e. the effect at the level of the subsystem the component belongs to);
- the end effect (i.e. the effect on the overall system);
- an indication of the severity of the fault (on a scale from I to IV);
- the method that can be used to detect the presence of the fault.

Notice that the effects are given in qualitative terms: either something that was supposed to happen does not, or something happens unexpectedly.

The FMECA process is very critical in aerospace companies, where legislation imposes standards for reliability and also for the FMECA process itself and for presentation formats (see as an example the military standards [1]). Today this activity is performed manually by expert engineers, based on their experience. They reason exploiting their knowledge of components and subsystems behavior (possibly in the form of the FMECA of a subsystem), to determine the behavior of the system in case a component fails. This activity is very time-consuming, because the analysis has to be performed for each different scenario in which a system is used (e.g., take-off, landing). Moreover, it has also to be performed for every configuration or smaller variant of each individual system.

The activity, in a sense, is very routinely and it requires reasoning patterns which per-se are not very complicated since they involve a qualitative analysis of the system behavior, which is a very usual form of reasoning for an expert engineer. However, the activity is at the same time very complex. First of all, all effects, even very remote ones, have to be taken into account (a critical effect may manifest in a component which is physically and logically very distant from the fault). Second, it requires taking into account intrinsically redundant systems with many components, and thus many paths of interaction between components and subsystems[9]. Finally, although FMECA is usually based on a single-fault assumption, it is important to check that, for very critical systems, the occurrence of a second fault does not lead to safety critical situations. This means that several aspects and interactions have to be taken into account simultaneously.

FMECA requires very expert specialists and this aspect may be critical to maintain and manage corporate knowledge; all the experience and work should not get lost in case of changes in the staff.

The above mentioned considerations, namely the fact that the activity is error-prone and that corporate knowledge should be stored and managed, call for the introduction of software tools that automate some of the phases and support the whole activity. Indeed tools for supporting FMECA have been produced, marketed and used in the last few years, see, e.g., SIMFIA [6] and AutoSteve [3, 4]. None of them, however, has been designed to meet the demanding needs of the aerospace industry; all of them have limitations; e.g., AutoSteve is limited to electric and electronic circuits, while SIMFIA only works on systems described in terms of Boolean equations.

[1] Dip. Informatica, Univ. Torino, Italy, email: picardi@di.unito.it
[2] Eurocopter, Marseille, France
[3] National Aerospace Laboratory NLR, AVCE, Amsterdam, The Netherlands
[4] Sofreten, Paris, France
[5] Alenia Areonautica S.p.A., Torino, Italy
[6] Occ'm Software, Munich, Germany
[7] Tech. Univ. Munich, Munich, Germany
[8] Israel Aircraft Industries, Tel Aviv, Israel

[9] A "sneak circuit" [8], is an example of this problem; in this case we may have unexpected behavior due to unexpected current flows in a circuit. Determining these situations may be difficult even for an expert engineer.

Failure Mode Effects Analysis

System Description: Landing Gear

Operation Mode: Flight - Level 2

Item Number	Item Description	Function	FM. Id.	Failure Mode	Local Effect	Next Higher Effects	End Effects	Sev.	Detection Method	Compensating Provisions	Remarks
1.1.1	Main Pump	Provides pressure when requested by Pilot Command	1	Fails to operate	No effect during this phase	No effect during this phase	No effect	IV	Indication to pilot	None	
			2	Untimely operation	Untimely hydraulic pressure in Main Hydraulic Generation Assembly	Untimely hydraulic pressure from Main Hydraulic Generation Assembly to Actuator Assembly	Untimely extension of Landing Gear	I	Indication to pilot	None	
1.1.2	Check Valve (Main)	Prevents reverse flow	1	Stucked closed	Loss of fluid flow through the Main Generation Assembly check valve	No effect during this phase	No effect	IV	Indication to pilot	None	
			2	Stucked open	Permits fluid flow through the main assy check valve when not required	No effect during this phase	No effect	IV	Undetected	None	

Figure 1. An example of FMEA table

The development of a software environment supporting FMECA in aerospace industry is the goal of the AUTAS European project[10], which aims at implementing the aeronautic standards to support the generation of FMECA. The solution we propose is based on *(i)* the adoption of a model-based approach to represent the models of the components and systems under examination and *(ii)* qualitative reasoning as the basic inference mechanism for performing FMECA. These choices are motivated by the qualitative nature of the FMECA process as it is currently carried out in the partner companies; they will be presented and discussed in the subsequent sections.

2 GOALS AND CONCEPTS

At a very abstract level, the goals of the project can be summarized as follows: developing a software environment that can manage corporate knowledge about systems and their FMECAs and automate some of the inference steps of the FMECA. A FMECA supporting software can therefore be seen as an integrated environment offering tools that support different steps of a FMECA:

- A tool for mantaining libraries of components and subsystems models. Such models should be generic and reusable, following a compositional principle: a new system can be created by composing existing components and subsystems; the new system is added to the library so that it is possible both to analyse it and to instantiate it as a part of a more complex system.
  Together with each component or system the user must be able to define the *effects* she wants to study on it. For example, for a pump, the user might want to consider the effect “pump produces no pressure”, while for a landing gear extraction system the user might be interested in the effect “landing gear does not extract”.
  This tool must include a Graphical User Interface that makes it possible for specialists to define models in a simple way.
- A tool for editing mission scenarios: for a given system FMECA should be performed in different operating conditions (missions).
- A tool for performing the basic inference steps underlying FMECA, that is determining the consequence of a component failure, at all the levels (local, next higher, end). Basically, the tool should be able to reason on component and system models to generate predictions.
- A tool for integrating such predictions with further information, e.g., that concerning criticality, obtained from the library and/or directly provided by the user, and for producing the final FMECA table with different presentation formats[11].

From a more technical point of view, the project is based on the following principles:

- *Component-oriented modeling.* The library is composed of models of generic component types; multiple instances of a type can be generated to create a subsystem model, which, in turn, will be added to the library as a new type (i.e., multiple instances can be generated to create the model of a more complex system).
  This means that the library must support modeling at multiple hierarchical levels. In this way, specialists (engineers) focus on modeling elementary components (e.g. pipe, check valve, pumps, ...).
  The principle of compositionality allows users to quickly create more and more complex models starting from elementary components (e.g. the model of the landing gear of an aircraft can be obtained by composing electronic and hydraulic subsystems and components). Furthermore, the possibility of re-using generic component models allows the user to easily produce multiple variants and configurations of a system.
- *Qualitative modeling*. Models describe the behavior of a component by in terms of qualitative constraints that relate variables of a component for each behavior mode. This representation captures

[10] AUTAS, GRD1-2001-40133: AUTomating FMECA for Aircraft Systems; partners: Alenia, Eurocopter, Israel Aircraft Industries, NLR, OCC’M, Sofreten, Technical University of Munich, University of Torino.

[11] The ability to configure the representation format of the FMECA table is a major issue, since different companies must comply with different and rigid standards.

the distinctions which are needed to support FMECA. For example, in order to produce the FMECA of an hydraulic system, it is sufficient to reason in terms of presence or absence of flow in a circuit, or in terms of the fact that a flow is lower than it should be, without taking into account the exact numeric value of the flow or of its deviation. Moreover, a qualitative approach to modeling and reasoning was successfully adopted for diagnosis and diagnosis related activities, to other application areas, such as the automotive field (see e.g., [2, 3, 4, 5, 7]).

- *Hierarchical reasoning.* FMECA is performed according to a hierarchical scheme, following the prescriptions of the standards adopted by aeronautic industries. This means that the process mirrors the hierarchical composition of models: given a component, the consequences (*effects*) of its faults must be evaluated at the level of the component itself (*local effects*), at the level of the subsystem to which it belongs (*next higher level effect*), and at the level of the overall system (*end effects*).

3 TOOLS IMPLEMENTATION

In this section we analyse the three tools constituting AUTAS (Models and Mission Builder (MMB), Qualitative FMEA Engine (QFE), FMECA Support System (FSS); see figure 2) and we discuss some of the choices in the software design and implementation. The overall architecture of the software is sketched in figure 2. We can see that the MMB supports both model editing and mission editing.

As said before, one of the two main goals was to build a persistent archive allowing users to reuse existing knowledge. This goal is achieved through the Model Library, that can be created, edited and maintained with the aid of the MMB, and stored in the Model Repository.

The first basic choice we made was thus the definition of a standard language for the model library, in particular defining:

- The qualitative domains for different physical quantities such as for flow, pressure, current intensity, resistance, etc.
- A standard representation of a component type model, defined through a set of attributes with well specified semantics: interface variables, state variables, parameters, behavior modes (including an OK mode at a set of fault modes) and the sets of constraints defining them.
- The representation of a system as a set of component type instances that are connected via their interface variables.
- The *effects* that one wants to observe on a given entity (component, subsystem or system); in particular it has been decided that an effect is represented by a constraint among (some of) the variables of the entity.
- Mission scenarios, that is the specific operating context and conditions in which the FMECA has to be performed; specifying a scenario corresponds to setting a value for state variables that are part of the components in the system under examination.

These standards have been implemented with an XML schema.

The second goal is to automate the inference steps behind FMECA, in order to generate automatically a FMECA table. The process of producing a FMECA table can be split in three phases, depicted in figure 2:

Phase 1: the MMB exports the qualitative model of a system together with a mission scenario towards the Qualitative FMEA Engine.

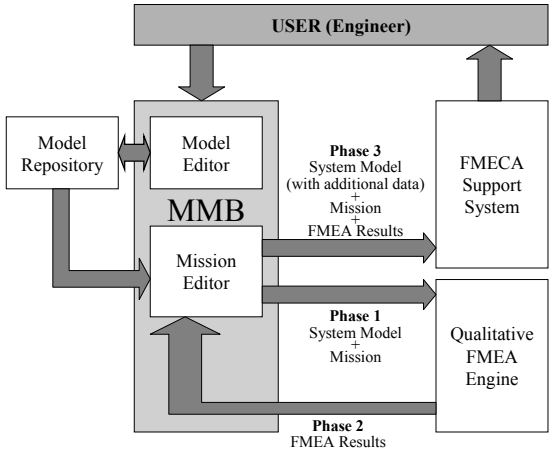

Figure 2. Software architecture

Phase 2: the QFE produces the inferences, by associating with each fault mode the effects that can be observed on the system at the different hierarchical levels when the fault is present.

Phase 3: the MMB endows the system model with additional data about fault probability and severity, as well as text descriptions that need to be printed out in the FMECA table. It exports such data together with the results of QFE towards FSS. FSS allows the user to format data in a FMECA table, by using existing data, doing some basic computation (e.g. the criticality of a fault), or entering other information.

Again the exchange of data among the different tools takes place by exchanging XML files. Thus not only the representation of the input data (models and mission scenarios) is standardized with an XML schema, but also the representation of intermediate results.

3.1 Model and Mission Builder

The MMB is a graphical tool for maintaining models and missions that will be stored in the "Models and Missions Library". Figure 3 shows a screenshot of the tool GUI. Each component type is defined independently of the system it will be used in. Subsystems can be created by connecting instances of component types, where the connection is provided by means of interface variables of the same type. The tool supports hierarchical modeling, in the sense that, once a subsystem has been defined, it can be used as a component of a higher level system.

The behavioral model is qualitative (i.e., variables range over qualitative domains) and it can be expressed as a set of constraints, either in an intensional way (i.e., by means of algebraic constraint between a set of variables) or in an extensional way, i.e., by using tables. The tool also allows the user to introduce parameters concerning fault criticality, e.g., the probability of failure of each component (and each fault mode). For each component, subsystem or subsystem it is possible to define the *effects* that one wants to observe. An effect is defined as a constraint relating the variables of the component, subsystem or system it is being associated with. Let us suppose that we want to define the effect "landing gear does not extract" for the landing gear extraction subsystem. The system description will involve a variable, let us say X, representing the qualitative status of the landing gear extraction (+ for extracted, 0 for not extracted), and a second variable, C representing the command sent to the subsystem by the pilot (0 for off, 1 for on). Then the above mentioned effect can be described as $(C = 1) \wedge (X = 0)$.

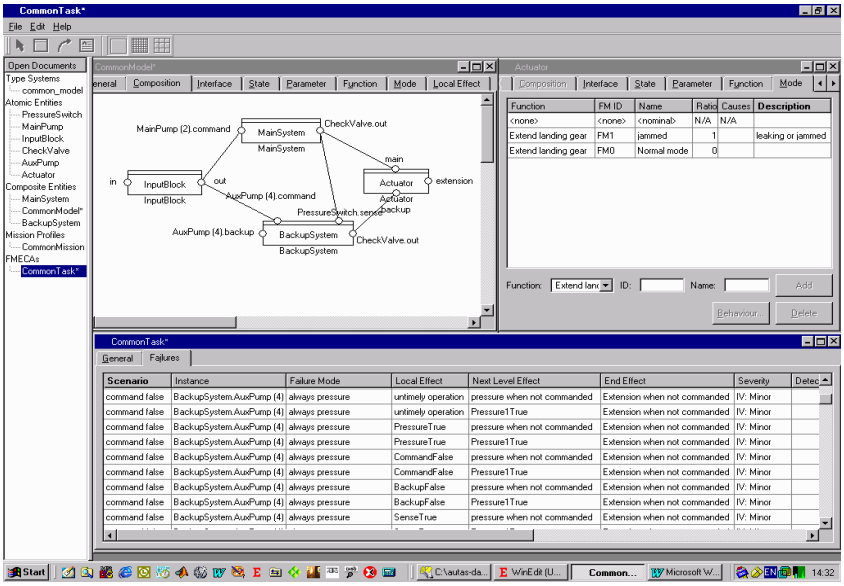

Figure 3. The Graphical User Interface of MMB

A mission scenario is defined as a specific situation in which the behavior of a system has to be analysed. It is characterized by a specific configuration of the system (e.g., only some components or subsystems may be active in that situation; this is obtained by selecting a specific system model) or by a specific operational mode of other components (i.e., to a specific setting of some of the components variables that in this case behave as sorts of parameters). For example, a scenario may correspond to the landing phase, where the landing gear is initially retracted and the landing gear extraction subsystem is switched on in order to extract it. In this scenario $C = 1$, thus if the landing gear does not extract ($X = 0$) we have the above mentioned effect "landing gear does not extract".

3.2 Qualitative FMEA Engine

The Qualitative FMEA[12] Engine (QFE) is the core of our software as it is the inference engine supporting the process.

FMEA requires to determine the consequences of each single fault of each component under each specified scenario; this corresponds to "simulating" the system in the situation where the component under examination is assigned a fault mode and all the others are in the OK mode. This analysis has to determine whether an effect from a set of pre-specified effects occurs. Effects correspond to certain violations of the intended function of the faulted component itself, the subsystem it belongs to ("next level effects") and the entire system ("end effects"). Technically, this is done by constraint solving which provides values for the variables that specify the effects. From those values the engine can extract the pieces of information to be inserted in the FMECA table; in particular:

- The values of specific variables directly provide information on the consequences of the fault (e.g., the fact that a tank is empty or that an actuator does not operate).
- Knowledge about the system structure and hierarchy, with the defined effects, allows the engine to infer the functionalities that are lost (next higher level and end effects).

[12] FMEA (Failure Mode Effects Analysis) is FMECA without Criticality Analysis.

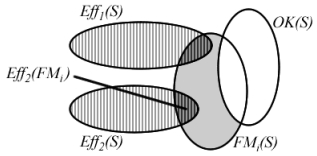

Figure 4. FMECA as a constraint solving problem

Let us briefly illustrate the required inferences as a constraint solving problem. Models of behavior modes are stated as relations (constraints) over a set of system variables. Scenarios specify certain exterior conditions and a particular state of the system and are expressed as a relation over the respective model variables. The joins of the model relations and the scenario relation describe the behavior of the system under this scenario, S. In figure 4, these are displayed as sets $OK(S)$ and $FM_i(S)$ for the correct behavior and some fault FM_i. Also the interesting effects are represented as relations over a different subset of variables, namely those that can be used to characterize the function and, hence, its violation. Because of the nature of effects, they are only well-specified if their associated relation and the correct behavior relation are disjoint, as indicated in figure 4 for two effects Eff_1, Eff_2. The FMEA analysis can then be stated as the problem of computing the join of the fault mode relation under scenario S and the effect relations (see $Eff_2(FM_i)$ in figure 4). In the QFE, this computation is performed as constraint propagation based on Ordered Multiple Decision Diagrams (OMDD).

3.3 FMECA Support System

The FSS is responsible for producing a standard output for the FMECA, including also additional information that can be computed or directly taken from the library, or provided by the user. In order to generate the output, the tool must integrate two types of information: (i) the output of the QFE, which provides a description of the consequences of each fault of each component; (ii) information about fault probability and severity, which is used to compute the criticality of

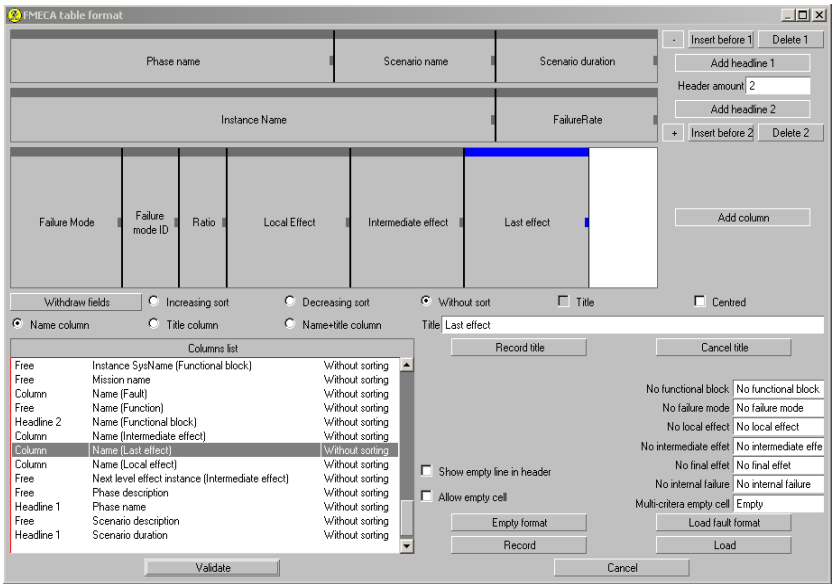

Figure 5. The Graphical User Interface of FSS

each fault. The output can be formatted with different layouts, in order to comply with industrial standars, selecting how to display each type of information and organizing or aggregating the table columns in different ways (figure 5 shows the GUI). Moreover, the output can be output to a file in different languages: XML or HTML for visualization, Excel sheets for further editing, and others.

4 CONCLUSIONS

In the paper we presented a software environment we designed and implemented to support the FMECA process in aeronautic industries. The FMECA of subsystems and systems of an aircraft (or helicopter) is very important as it is the basis of reliability aspects concerning the aircraft. The activity is currently performed manually and this can be problematic, since taking into account all behaviors and all the interactions between the behaviors of several components of a system can be very complex, error prone and costly.

In the paper we presented a software that supports the process; the core of the tool is a qualitative reasoning engine that can determine in an automatic way the consequences of each fault of each component of a system. The software relies on the use of a library of models of components, thus allowing an engineer to analyse easily several variants of a system or several contexts or configurations or operation modes (missions) on the same system.

The software is currently being used by the three industrial partners (IAI, Alenia and Eurocopter) and by NLR in order to test its behavior on different types of models. In particular, a first experimentation has been carried on a simple hydraulic model of the landing gear actuation (including a pump, a backup pump, valves, controllers and the actuator driving the gear). Given the interesting results obtained on this system, the attention focused on more complex systems: the Electronic Power Generation System of a military airplane, the Transmission Gearbox of an helicopter, the Landing Gear of a business jet aircraft and the Air-Conditioning System of a civil aircraft. The modelled components are essentially hardware parts; we considered open loop models and we abstracted from problems in the control software. Dynamics are taken into account by considering different single-snapshot scenarios; the user is responsible for identifying significant states in the dynamics of the system and selecting them for FMECA generation. This mirrors what is currently done manually.

The first experimentations showed that the model-based support can have a significant impact in improving FMECA in aeronautic applications. Indeed engineers dealt comfortably with qualitative models, although the effort for building models is significant, even for simple components. This means that the effort for building actually reusable model libraries will be a major one, but the availability of such libraries will greatly propel the importance of AUTAS results. FMECA will in fact be easier, faster and more reliable.

ACKNOWLEDGEMENTS

The authors would like to thank all the people who worked and are working in the AUTAS team. The work is supported by the EU, grant GRD1-2001-40133.

REFERENCES

[1] USA Department of Defense, ‘Military standard: Procedures for performing a failure mode, effects and criticality analysis (MIL-STD-1629A)’, Technical report, (1980).

[2] C. Picardi, R. Bray, F. Cascio, L. Console, P. Dague, O. Dressler, D. Millet, B. Rehfus, P. Struss, and C. Vallée, ‘IDD: Integrating Diagnosis in the Design of automotive systems’, in *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI2002)*, pp. 628–632, (2002).

[3] C. J. Price, ‘AutoSteve: Automated electrical design analysis’, in *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI2000)*, pp. 721–725, (2000).

[4] C. J. Price and N. S. Taylor, ‘Automating multiple failure FMEA’, *Reliability Engineering and System Safety*, **76**, 1–10, (2002).

[5] M. Sachenbacher, P. Struss, and C. Carlen, ‘A prototype for model-based on-board diagnosis of automotive systems’, *AI Communications*, **13**(2), (2000).

[6] Sofreten, ‘Simfia, a simulation workshop for system engineering’, Technical report, Downloadable from `http://www.sofreten.fr`, (1990).

[7] P. Struss and C. J. Price, ‘Model based reasoning in automotive industry’, *AI Magazine*, **24**(4), 17–34, (2003).

[8] F. E. Walker, ‘Sneak circuit analysis automation’, in *Boeing Aerospace, Proceedings Annual Reliability and Maintainability Symposium*, ed., IEEE, Seattle, (1989).