Towards a better interaction between design and dependability analysis: FMEA derived from UML/SysML models

P. David, V. Idasiak & F. Kratz
Institut PRISME/LVR, ENSI de Bourges, Bourges, France

ABSTRACT: It is commonly admitted that one of the crux, during the design process of new complex systems, is the efficient communication between the experts. The OMG developed UML to tackle this problem in software design. For several years, many researchers have worked on adapting and reusing UML for whole kind of systems. These efforts led to the creation of SysML, a new modeling language for specifying, analyzing, designing and verifying complex multi-disciplinary systems. In this paper, we present how those modeling languages can be integrated to combine reliability study with the design process in order to compose a conception process close to dependability priorities. We have set up a method to conduct reliability studies without perturbing the classic procedures of designers. Our objectives were on the one hand to facilitate communication between functional and dysfunctional analyzers and to simplify the execution of reliability studies on the other hand. In this article, we raise the utility of FMEA for a non-intrusive modeling of dysfunctional aspects. We thus give our solution for an automatic synthesis of FMEA from UML/SysML models, which only describe the nominal behavior. We insist on the keystone of this automatic synthesis, namely the use and management of a database constructed from the information of the return on experience. The purpose of this article is thus to describe a new approach to enhance the realization of dependability studies by automating steps and using easily understandable models.

1 INTRODUCTION

The complexity of new systems has dramatically grown and continues to follow this trend. So, the intricacy of the development phase is clearly related to the number of different technologies embedded in the system. For instance, the necessity to integrate software raised many difficulties that have been tackled by numerous research teams. One origin of these difficulties is the difficult communication between the experts of each discipline. Ambiguities exist with discipline-specific models and languages. The synchronization and aggregation of technology-specific models is a complicated task, error-prone and time-consuming. The dependability studies suffer of the same hindrance. Indeed, dependability evaluation or qualification need dedicated models not easily understandable by designers. Therefore, reliability studies and conception processes are too often considered as separated workshops in development projects. This leads to a lack of communication between the experts of both part of the system conception. To sum up, we can act that system designers and analysts must face two main issues: Inter-discipline communication and multi-technology integration.

Thus, the challenge for efficient reliability studies in development phase is to combine and synchronize the design models and reliability analysis. In this paper, we will present a method and tools that we developed in order to link as fast as possible the design choices and their impacts on the global system reliability. In the remaining of this article, we will highlight the benefit of the use of UML/SysML for complex systems design. We will mention how those languages can be helpful for reliability studies. Then, we will introduce a method for reliability studies using UML/SysML diagrams. Namely, we will explain how to deduct the dysfunctional behavior of a system from its functional description, using an automatically generated FMEA. Finally, we will introduce future developments and underline the improvements brought by SysML.

2 UML AND SYSML FOR RELIABILITY ANALYSIS

2.1 *Languages for multi-technologic system design*

Communication problems were serious in software engineering, because of the very precise and specific language utilized by computer engineers. The end-

users, non-expert in computing method, had often difficulties to express their needs to the programmers. To tackle this issue, the computer science community developed several languages synthesized in a common language: UML. UML has been specified in order to describe, in an easy understandable way, software projects and complex systems embodying software. This language well fitted its objectives and is currently widely used in software industry. Since UML 2.0, this language is not restricted for software design; thanks to its object-oriented approach it is exploitable to describe any type of system. Many published works proposed the use of UML for system engineering and demonstrate all the advantages of object-oriented method in complex system design like (Lykins et al. 2000). Nevertheless, because of a semantic too software-oriented, it has not been much used for complex systems design. Therefore, system engineers always needed a modeling language allowing sharing and unifying discipline-specific parts.

That is why OMG created SysML (System Modeling Language) (OMG 2007) for specifying, analyzing, designing and verifying complex systems. SysML has been constructed as an extension of UML adapted for system engineering. It brings new means for system modeling and requirements. In (Willard, 2006) the author highlights the heritage from UML 2.0 and presents the new possibilities brought by SysML, he claims that the main benefit of SysML is "to provide system engineers with a standard and comprehensive system specification paradigm". In (Hause 2006) the reader can find a good introduction to SysML and its new diagrams namely requirements diagrams, parametric diagrams. SysML uses UML 2.0 diagrams as class or object diagrams but adapts the semantic to avoid software vocabulary (e.g. class and object are replaced by blocks). Moreover, new diagrams are added to simplify requirements declaration and to build a bridge towards simulation-based design. Parametric diagrams describe the equations linking the multiple parameters of the model. In (Peak et al. 2007a,b), the authors introduce the background of the SysML parametric diagrams and their relation with Composable OBject (COB) technology; they explain how parametric diagrams will allow SysML to support simulation-based design. In fact, they demonstrate how the models can be exploited with analysis tools and equation solver as XaiTools (X-Analysis Integration Toolkit is a trademark of Georgia Institute of Technology).

Those two languages, and especially SysML, are then suitable to system engineering. The oriented object approach and the hierarchy and composition possibilities permit to model multi-technological systems. Moreover, it is widely noticed that their graphical description enhances understanding and reduces miscommunication. Finally, we have to mention that a crucial advantage of those languages is that many software tools support them.

2.2 *UML/SysML for reliability study and risk analysis*

Because UML caught the interest of system engineers, dependability analysts decided to explore how dependability study could combine with this kind of models. This leads to several works that are currently extended to SysML.

Integrating reliability requirements or conducting reliability or risk studies on UML models raise many challenges of model construction and information representation. The problem of modeling reliability using UML is presented for software systems in (Leangsukun et al. 2003). The authors notice that techniques utilized for dependability estimation are not well mastered by engineers. Consequently they have imagined a method to reduce the gap between conception practices and reliability analysis. They propose to define new UML stereotypes to describe the dysfunctional behavior of the system. This dysfunctional behavior is thus added to the model so that the two views of system activities are mixed. Thanks to the interchange file format (XMI) used by UML tools, the authors automatically extract the information needed to "feed" the SHARPE tool (Symbolic Hierarchical Automated Reliability and Performance Evaluator developed at Duke University) as failure and repair rate. Fault trees and Markov chains are then constructed and analyzed. The analysis of a two-tier client/server architecture illustrates their paper. Zarras and Issarny have already used a similar approach. In (Zarras & Issarny 2001) they present their stereotypes for reliability declaration. Moreover, they decide to specify the requirements with OCL (Object Constraint Language), which is another OMG specification for expressing special relationships in the model. In the two previous articles, the additional information concerning dysfunctional behavior is clearly constructed in order to be used by the analysis tools employed. In (Zarras et al. 2004) the approach is slightly different. The model is constructed in a similar way but the exploitation of the model is not a simple data transformation. In fact, algorithms are employed to automatically generate some artifacts usable for reliability analysis. Block diagrams, fault trees and Markov chains are thus created and analyzed to assess the functioning of a composite web service. Other works exploit UML models for risk analysis. In (Guiochet et al. 2004), the approach is fundamentally different. They analyze a medical robot for tele-echography. The model is not constructed to characterize the possible failure, but to conduct a risk analysis on the represented system. The authors define error models for the messages of UML sequence diagrams. Firstly, they model the task and mission of the system. Then, they

use their message error model to product the FMEA of the tasks.

These multiple approaches showed that it is possible to express or deduct information about risks or dysfunctional behavior of systems modeled in UML. Nevertheless, those works are guided by specific goal. The modeling principles and the information given in models are conditioned by the analysis that users want to apply. Moreover, the information must respect a very strength semantic, which is difficult to follow, as the stereotypes defined in Zarras and Leangsukun works (Zarras & Issarny 2001, Leangsukun et al. 2003). In those cases reliability analysis add complexity to the model. Furthermore, to construct them the modeler must already know the dysfunctional behavior and be able to quantify the failures of each component. On the contrary, in his work Guiochet only uses the model to deduct risks from an only functional point of view. But, the model is only built to perform the risks analysis and not really used for the system design. In each case, we can say that the model is built for reliability study purpose, so that the models built by system engineers are not really analyzed.

The use of SysML mitigates some of those drawbacks by the mean of new diagrams. Actually, requirements diagrams will be precious to express reliability requirements without using complex OCL rules as in (Zarras & Issarny 2001). Besides, modeling reliability aspects have been clearly an interest for SysML (OMG 2007), parametric diagrams represent constraints on system property values such as reliability. Furthermore, in works as (Peak et al. 2007a,b), we see that it will be possible to connect parametric diagrams with function solver, allowing us to compute new reliability indicators. In the next section, we will present our method for reliability analysis, which aims at bringing new solutions in order to reduce the gap between design practices and reliability analysis methods.

3 CONNECTING DESIGN PROCESS AND DEPENDABILITY STUDY

As we exposed in the preceding section, the current works, on bringing together design process and reliability studies, are certainly perfectible but demonstrate huge possibilities and really encouraging results. Since UML is the main trend in design activities, we are persuaded that developing method exploiting UML models is the right solution. Moreover, an adaptation to SysML is necessary for these techniques, in order to deal with system engineering.

The aim of our research is not to find how to model reliability in UML or SysML, but to be able to analyze real conception models expressed in those languages and to obtain new information concerning the dependability of the system. A major drawback of reliability study is that it delays and complicates the design process. Consequently, we develop a method that does not interfere with the model created by the designer. We set up a software tool that helps an engineer to perform this task as easily as possible. We thus propose a deductive and iterative method in three steps:

- Deduction of the dysfunctional behavior with a FMEA
- Construction of the dysfunctional model
- Analysis and quantification

Each step of this approach requires dissimilar models and methods. Our role is thus to enhance as much as possible the link between each activity. That is why we are currently creating tools to automatically conduct risk and reliability studies from the designer model written in UML/SysML. The goal of each step is to generate new knowledge or to automatically create new models for various analyses. A diagram representing the approach is shown on Figure 1 and details the successive models.

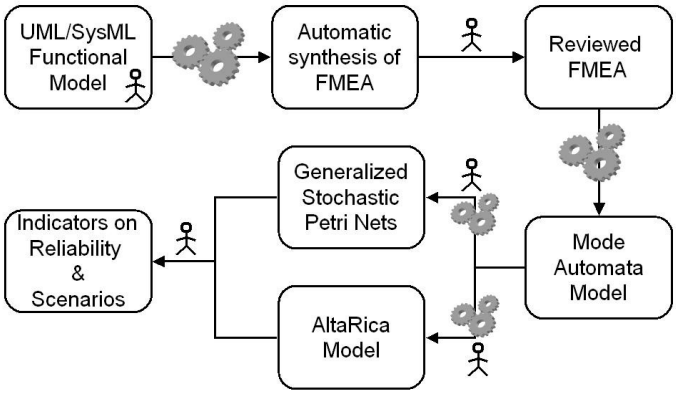

Figure 1. An approach from functional model to reliability indicators

The first step of the instrumented method is the establishment of a FMEA. This FMEA is generated from the study of UML/SysML models. We use a FMEA to find the dysfunctional behavior of the system. In fact, in our approach we do not consider that the dysfunctional behavior is already known as (Leangsukun et al. 2003) and (Zarras et al. 2001, 2004) do. Therefore, this step is very critical in our approach because it has to highlight the failure modes that will be qualified and quantified in the rest of the study. In order to generate the FMEA from UML/SysML models, we use techniques of data translation using files exploiting XML (eXtensible Markup Language). The creation of the FMEA is detailed in the fourth section of this article.

We do not want to add the dysfunctional behavior to the first model. We thus build a model in mode automata (Rauzy 2002) to depict the whole behavior of the system. We have chosen the mode automata formalism because of its structure. With mode automaton, it is easy to represent for each component of the system their reachable states and the

treatment of the flow that they receive and transmit. The construction of the model is done component by component, so that the decomposition established in the FMEA can be naturally reused. Moreover, this modeling formalism highlights the error propagation between components by describing the flow ports and the state of the flow. Then, with a mode automata model it is possible to derive models in AltaRica (programming language implementing mode automata) or Generalized Stochastic Petri Nets (GSPN). The main concern of those models is to show the influence of data and energy transmission throughout the architecture of the model. The rules of creation of those models will be given in future communications

Both AltaRica and GSPN models are used in software tools (respectively the OCAS module of Cecilia workshop and Jagrif edited by Dassault Data Services). Those software tools permit to quantify reliability indicators, such as global failure rate or mean time to failure. Solutions to compute fault trees also exist (e.g Aralia Sim Tree) and give a mean to find failure scenarios and also to compute failure rates. The last step of our approach is thus to use those kind of tools on the previously obtained models in order to compute the necessary results for the design evaluation.

In order to render this approach efficient, we develop tools for the automatic creation of each model and for the construction of the files needed by the reused tools. We try to construct a bridge as short as possible between a functional design and the evaluation of the reliability characteristics of the proposed architecture. A software tool is developed in order to integrate the different phases in a single application. In the next section, we will discuss the realization of the first step. We will present the problems raise by the automatic synthesis of FMEA and mention the existing works. Then, we will describe our policy for an automatic synthesis of FMEA.

4 AUTOMATIC SYNTHESIS OF FMEA DERIVED FROM UML/SYSML FUNCTIONAL MODELS

FMEA is the standard method for risk analysis in the design phase (Tumer et al. 2003). It is a well-known inductive and qualitative method that proposes to explore the system component by component. For each one the analyst searches their failure modes and their effects on the system, detailing their severity and occurrence rate, in order to underline their weak points.

4.1 *Works on FMEA automation*

Over the past few years, many authors have mentioned the obstacles against FMEA execution. In fact, many organizational and operational constraints reduce its efficiency. FMEA is often seen as a time consuming and error prone analysis. For industrial user, it is frequently difficult to link the results of FMEA with the execution of corrective procedures. The main interest of FMEA, which brings up information to direct design efforts, is thus lost. The heaviness of the method seems to have many origins, as the difficulty to call together the participants or the huge amount of information to produce, represent and understand. Nevertheless, this method, created to satisfy precise needs in system analysis, has sufficient advantages to justify works on it. The method provides a systematic detection of risk and failure at an early stage of the design process. It guarantees the exhaustive identification and the classification of risks. Finally, it allows to identify the weak points of the system only from a functional view.

The benefits of this analysis are important enough to justify the employment of new techniques to enhance its utilization. There are two options to improve FMEA, which are: to ameliorate the organization or to support the study with software. Bassetto, in (Bassetto 2005), thanks to an experiment on a whole plant, gives organizational advises to involve the engineers and enhance their perception of FMEA:

- Define the analysis limits and objectives.
- Constitute team with experts and user of the analyzed system.
- Teach the method to the participants.
- Impose regular meetings.
- Obtain the interest of the management and operational teams by underlying their benefits in using the method.

To improve those dispositions, it seems very important to support FMEA with a software tool. Many authors and normative documents have pointed the lack of adapted software for FMEA management. By the way, many researchers have explored how software could be useful for FMEA development. Therefore, it exists various helps brought by software tools:

- Maintaining a return on experience database (Bassetto 2005).
- Help to fill the FMEA table.
- Automatic synthesis of parts of the FMEA table (Papadopoulos et al. 2004a), (Bull et al. 1996).
- Organizing and highlighting the relevant elements (Price, 2000).
- Help for the use and creation of failure taxonomy (Tumer et al. 2003), (Bassetto, 2005).
- Model generation from FMEA table (Papadopoulos 2004b).
- Managing simultaneous failures (Price and Taylor, 2002), (Papadopoulos, 2004b).

The automatic synthesis of FMEA is treated in multiple works. Some of those are centered on the exploitation of a database and on making it sustainable. Others deal with the automatic generation of information by causal reasoning. In the two main works (Price and Taylor 2002), (Papadopoulos et al. 2004b), the requisites for automatic synthesis of the FMEA are the modeling of each failure for each components. The real benefit of these methods is the computation of the effect on the whole system. The profit of the FMEA is no more to identify the unitary failures that could occur, but to find multiple failures that could be significant. Finally, we can mention works on the semantic used in FMEA. To solve the ambiguity of coupling natural vocabulary with quoted values, (Bowles and Pelaez, 1995) propose to use fuzzy logic rules. This technique is still experimented in recent works (Xu et al. 2002), (Yeh and Hsieh 2007) on various kinds of system (engine, sewage plant). Those different works helped us to identify the crucial points for the automation of FMEA creation.

4.2 *Influential points for Automatic synthesis of FMEA*

We have identified two major points for the success of an automatic synthesis of FMEA. These are the model from which the analysis is built on the one hand, and the database of dysfunctional behavior on the other hand.

FMEA specification indicates that this analysis is developed on the results of a functional analysis. In fact, the analyst needs to know each function of each component to conduct his FMEA. In order to fill each line of the FMEA, the engineer plays a component life scenario, so he must be familiar with the functioning of the relationships between system components. To express this functional information, many methods and languages are utilizable; the majority of them are technologically dedicated solutions. Currently, the methods utilized for the automatic synthesis of FMEA are functional models enriched by data on the dysfunctional behavior of the components (Price and Taylor 2002, Papadopoulos et al. 2004a). This is quite paradoxical with the first aim of FMEA, which is to underline and detect risks and failures only from the knowledge of the expected functional behavior. For us, a model is exploitable for FMEA, if it allows to isolate the architecture of the system (physical distribution of the functions), as well as the energy and data transmission between components. In fact, in a deductive study it is important to easily recognize the propagations between the system elements. To sum up, the elements that the modeling language, used for FMEA, should be able to model are the following:

- Architecture of the system and its functionalities
- Hierarchy between blocks
- Data and flow transmission

Those aspects can be modeled with UML or SysML by the use of class diagrams, composite diagrams and deployment diagrams. A data and flow transmission view also exists in sequences diagrams and interaction diagrams, moreover the UML 2.0 and SysML specifications introduce very useful flow port diagrams. Finally, the possibility to classify the objects gives to those languages a great capacity to furnish easily reusable uniform information.

It is essential to construct a database containing the return on experience on the failure modes of utilized components. (Basetto 2005) exposes that: “if each risk component possesses its own typology, the automatic generation of risk can be envisaged”. For Basetto a “risk” is the term that refers to a FMEA line. A risk is thus composed of an element name, its failure modes, their effects and causes and optionally its severity and occurrence. According to him the automatic generation of FMEA can be performed only if the set of words utilized to qualify each risk element is finite and previously known. In the FMEA process, this kind of database is used in two ways: firstly the database is a source that helps proposing for each element the right and precise failure modes, secondly after the user has reviewed and completed the FMEA the new information must update the database for the specific use of the component in the studied system. In those conditions, FMEA becomes a precious resource for the establishment of a management process for return on experience. Nevertheless, the use of a database causes an unavoidable rigidity in the employed vocabulary. In fact, taxonomy must be fixed, researcher teams as (Tumer et al. 2003) work on that topic. For instance this team defined taxonomy on failure modes for plastics. The taxonomies aim at describing in precise words the elements that compose FMEA lines. Those elements depend on the studied technology. Each technological domain possesses its own terminology. The risk elements that should have their proper taxonomy are the components, functions, failure modes, effects, causes, detection means and the corrective actions. This exhaustive list can be reduces regarding the goals of the FMEA. A typology for the components is essential for the automatic synthesis, moreover it provides means for the reuse of known components in new systems. The component typology constitutes an entry in a database that indicates each failure modes for the designated element. This simple mechanism is a main step in the automation of FMEA synthesis, which helps to guarantee the exhaustive enumeration of failure modes. Then a typology of failure modes is necessary and will authorizes to reason on effects and characteristics as severity and occurrence. Using typologies for effects and causes makes possible to underline consequence relationships between risks.

For example Basetto analyses the bijections between the typology set of effects and causes and so highlights the "risk core" of his application. The use of typologies for the database exploitation in the case of automatic FMEA synthesis is essential. Therefore, to be efficient we must impose on the designer the respect of a typology for the definition of his model. The best way is to reuse as much as possible his own vocabulary and to let him modify and enrich the database with his own experience.

4.3 *Using UML/SysML for automatic synthesis of FMEA*

We will describe in this section our solution for the automatic synthesis of FMEA. Our goal, in this step, is to render operational the execution of a FMEA. The role of this FMEA is to help us to find the dysfunctional behavior of the studied system with its functional model. We wish to use the model designed by the conception engineer without adapting it to our purpose. We want to make this generation as complete as feasible, but also as fast as possible.

4.3.1 *Exploited model and first solution*

As explained in the second section, using models in UML/SysML seems to be the best solution in order to define a general approach that is to be integrated in a modern design process. The models in UML/SysML are well suited to be exploited for FMEA generation. Traditionally, FMEAs are built from functional analysis. Our first algorithm only exploited the sequence diagrams of the designer models. The use of sequence diagrams in order to create an FMEA is justified because we observe that it is possible to find the same information about the studied system in both formats. The functional analysis makes it possible to identify all the system functions. These are the functions that are described by each use case represented by the sequence diagrams. An algorithm that we developed, whose instructions are described in the following lines, realizes the treatment that creates the FMEA from sequences diagrams.

Input of the algorithm:

- Set of the Actors noted A
- Set of the Objects noted O ($A \cap O = \varnothing$)
- Set of the messages noted M

The elements of M are couples (x,y) where $x, y \in A \cup O$, x is the sender of the message and y its receiver.

- C and E are two empty sets (Causes, Effects)
- The set of the failure modes is noted FM

FM = {partial function, no function, intermittent function, unintended function}

Output of the algorithm:

- Table of the FMEA

Operators:

- + is the operator of concatenation on the right
- for $m \in M$, m = (m[1], m[2])

Start of the algorithm:

Creation of a table whose first line is composed of 9 boxes that indicate, the name of the object or actor, the failure mode, the cause(s), the effect(s), the severity, the probability, the risk, the means of detection, the technical solutions.

$\forall i \in A \cup O$ and $\forall fm \in FM$
  $\forall m \in M$, if m[2] = i, C = C + m[1]
  $\forall m \in M$, if m[1] = i, E = E + m[2]
Creating a line = {i, fm, C, E, ∅, ∅, ∅, ∅, ∅}

This algorithm has been tested on diagrams used by (Guiochet et al. 2004). Results and benefits of its utilization are exposed in (Belhadaoui et al. 2007). The main interest of this version is that it automatically lists all systems components and proposes to evaluate their behavior according to the fourth basic failure modes. Moreover, it directly associates the entity linked with the component so that the effects and causes are easier to identify. In (Belhadaoui et al. 2007), the authors used this algorithm on a processor UML model. They remark that our solution allowed them to find faster the same information that they found in a previously manually built FMEA. They add that they saved time, by using this algorithm (implemented in a prototype), in the FMEA creation. Moreover, they indicate that it permitted them to identify new risks forgotten in the previous study, thanks to the links between components and the failure modes proposed by the software tool.

Nevertheless, this first solution has a significant drawback. In fact, too many lines are produced and the identified risks need to be clarified by the user. Consequently, we decided to enhance this algorithm, thanks to a connection with a component database. The database proposes for each component its proper failure modes. Those modes are taken from documents on the return on experience of the components. The created FMEA became more precise and concise. The connection to the database intervenes for the enumeration of the failure modes. The algorithm identifies the type of the studied component and then it searches the right failure modes for the component in the database.

4.3.2 *Classification and database constitution for automatic FMEA synthesis*

As presented in section 4.2.2 the database of functional behavior is crucial because it involves the return on experience for the risk identification. This way, we insure the coherence among various risks studies and we automatically obtain more precise results. The database that we use contains the failure modes of the component that we exploit. Therefore, the new algorithm indicates for each risk the right failure modes as they are described in the database.

Nevertheless, if a component not indicated in the database is met, the algorithm will continue to propose the basic failure modes as in the first version. Later, by integrating the failure modes corrected by the analyzers, the database can be enhanced. Figure 2 illustrates the use of a database for FMEA creation. We have constructed our database using the concept of class depicted in UML and extended in SysML with the Block Definition Diagrams (BDD). This enables us to work at various level of detail in the systems description. In fact, types of components can be depicted with generalization relationships.

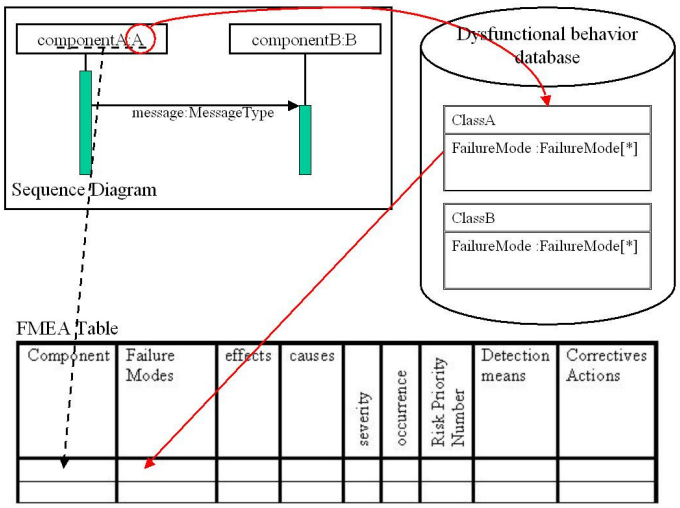

Figure 2. Use of a classified database for FMEA synthesis

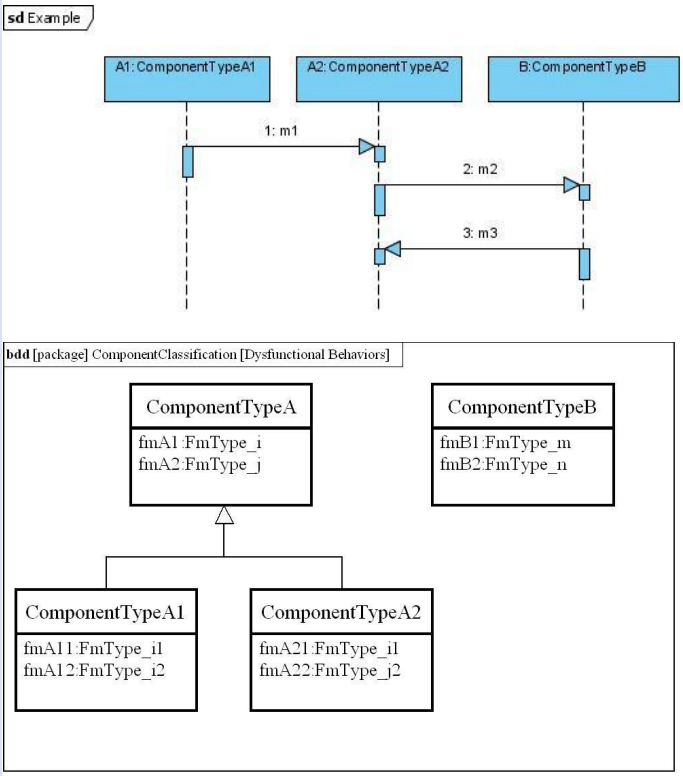

Figure 3. Typical sequence diagram and component database

If SysML is used, the database is constructed with BDDs, an example his given on the bottom part of Figure 3. The database is organized around the components types. Generalization relationships exist between them, for each type we indicate their failure modes as attributes. The names of the failure modes are indicated so that the algorithm will be able to note them in the FMEA table. This technique helps us to save a lot of time in the FMEA synthesis. We are able to create a major part of the FMEA in a precise way and we also guarantee that we exhaustively study the components. Figures 3 and 4 show a typical use of this algorithm on simple sequence diagrams. The part of FMEA that is automatically created is given. We can read in this table the precise failure modes of the components as well as suggestions about the behavior of the failure (components involved in causes and effects).

The component database can be completed by a failure modes database. This second database allows us to conduct the next step of the process presented in section 3. In fact, this database describes the failure mode types defined in the component database. We associate diagrams expressing their behavior with the failure mode types. Those diagrams could be GSPN models, AltaRica files or Statecharts diagrams. This will be a first step in an automatic creation of the dysfunctional model. Moreover those diagrams can help to find the effects of the failure modes and thus to build the FMEA.

Components	Failure Modes	Potentially affected Components	Effects	Components potentially involved in the Failure
A1	fmA11	A2 (m1)	fmA11 behavior	A1 (internal failure)
	fmA12	A2 (m1)	fmA12 behavior	A1 (internal failure)
A2	fmA21	B (m2)	fmA21 behavior	A2 (internal failure) A1 (m1) B (m3)
	fmA22	B (m2)	fmA22 behavior	A2 (internal failure) A1 (m1) B (m3)
B	fmB1	A2 (m3)	fmB1 behavior	B (internal failure) A2 (m2)
	fmB2	A2 (m3)	fmB2 behavior	B (internal failure) A2 (m2)

Figure 4. Deducted FMEA

4.3.3 *Development of the software tool*

We are currently developing a software tool in order to fully integrate the first step of our approach. This tool is designed to help designers in FMEA synthesis. The UML or SysML files written in XML, thanks to the XMI formalism of the OMG, are loaded. Then the software uses a parser in order to interpret the data and execute the algorithm for the creation of the FMEA. This software is also designed to manage the database described before, it authorizes the user to fill and utilize it for the FMEA synthesis. A second parser will analyze the complete FMEA in order to pass the more precise information, on the failure modes, to the component database.

5 CONCLUSION

In this article we discussed the appropriateness of using UML/SysML for reliability studies. We briefly described our approach that aims at making easier

reliability studies during design process. We pointed out the essential issues for automatic synthesis of FMEA as a method for risk and failure identification. Then, we presented our first solution for this synthesis. We presented a more efficient version of the preceding algorithm and focused on databases construction for this purpose. In the future, we will develop and integrate in our tool, a second parser in order to create Statecharts and GSPN from the results of the FMEA and the UML/SysML models. Moreover, the novelties of SysML specifications open new opportunities for reliability studies. For instance, we currently develop a third version of our algorithm exploiting the new SysML diagrams. We hope to realize the complete reliability study, using SysML and existing analysis tools. We wish to link the failure rates computations to their declaration in the component database. To perform that, it is possible to associate a parametric diagram for each failure mode of the components. This diagram models the equation for the failure rate calculus, linking environmental parameters and the component features. Those equations can be found in the normative documents dedicated to the studied technology.

ACKNOWLEDGEMENTS

We specially want to thank all our partners involved in the CAPTHOM project. This work was realized with the financial help of the French Industry Ministry and local collectivities, within the framework of the CAPTHOM project of the Competitiveness Pole S^2E^2, www.s2e2.fr.

REFERENCES

Basetto, S. 2005. Contribution à la qualification et à l'amélioration des moyens de production ~ Application à une usine de recherché et production de semi conducteurs. *Thesis presented at Ecole Nationale Supérieure des Arts et Métiers, 28 june 2005.*

Belhadaoui, H. Cassier, C. Idasiak, V. Malassé, O. Aubry, J.F. 2007. Outil d'aide a la conception des systèmes mécatroniques sûr de fonctionnement. *Qualita 2007,Tanger, 20-22 march 2007.* 426-433.

Bowles, J.B. & Pelaez, C.E. 1995. Fuzzy logic priorization of failures in a system failure mode, effects and criticality analysis. *Reliability engineering and system safety* 50:203-213.

Bull, D.R. Burrows, C.R. Edge, K.A. Hawkins, P.G. Woolons, D.J. 1996. A computational tool for failure modes and effects analysis of hydraulic systems. *ASME annual winter meeting, Atlanta, 17-22 november 1996.*

Guiochet, J. Motet, G. Baron, C. Boy, G. 2004. Toward a human-centered UML for risk analysis. In Jonhson, C.W. & Palanque, P. (eds), *IFIP world computer congress, Human error, Safety and Systems Development; Proc. Intern. Symp.* 177-191.

Hause, M. 2006. The SysML modeling language. *15th European systems engineering conference, September 2006.*

Leangsukun, C. Song, H. Shen, L. 2003. Reliability modeling using UML. *International conference on software engineering research and practise, Las Vegas, June 2003.*

Lykins, H. Friedenthal, S. Meilich, A. 2000. Adapting UML for an Object-Oriented systems engineering method (OOSEM). *Proceedings of the 10th annual INCOSE symposium, july 2000.*

OMG 2007. OMG systems modeling language (OMG SysML) V1.0. 1st September 2007.

Papadopoulos, Y. Parker, D. Grante, C. 2004a. Automating the failure modes end effects analysis of safety critical systems. *IEEE Hase 2004.*

Papadopoulos, Y. Parker, D. Grante, C. 2004b. A method and tool support for model-based semi-automated FMEA of engineering designs. Cant, T. (ed.). *9th Australian workshop on safety related programmable systems,* 47

Peak, R.S. Burkhart, R.M. Friedenthal, S.A. Wilson, M.W. Bajaj, M. Kim, I. 2007a. Simulation-based design using SysML Part 1: a parametric primer. *INCOSE intern. Symp., San Diego, 4 may 2007.*

Peak, R.S. Burkhart, R.M. Friedenthal, S.A. Wilson, M.W. Bajaj, M. Kim, I. 2007a. Simulation-based design using SysML Part 2: celebrating diversity by example. *INCOSE intern. Symp., San Diego, 4 may 2007.*

Price, C.J. 2000. AutoSteve: automated electrical design analysis. *ECAI proc. Int. symp.,* 721-725.

Price, C.J. & Taylor, N.S. 2002. Automated multiple FMEA. *Reliability engineering and system safety,* 76: 1-10.

Rauzy, A. 2002. Mode Automata and their compilation into Fault tree. *Reliability Engineering and System Safety*, 78: 1-12.

Tumer, I.Y. Stone, R.B. Bell, D.G. 2003. Requirement for a failure mode taxonomy for use in conceptual design. *ICED, Stockholm, 19-21 august 2003.*

Willard, B. 2006. UML for systems engineering. *Computer standard and interfaces,* 29:69-81.

Xu , K. Tang L.C. XIE, M. Ho S.L. Zhu M.L. 2002. Fuzzy assessment for engine systems. *Reliability engineering and system safety,* 75:17-29.

Yeh, R.H. & Hsieh M.-H. 2007. Fuzzy assessment of FMEA for sewage plant. *Journal of the Chinese institute of industrial engineers,* 24:505-512

Zarras, A. & Issarny, V. 2001. A UML-based framework for assessing the reliability of software systems. *Proc. Intern. Symp. ICSE worshop on describing software architecture with UML, Toronto, mat 2001.* 36-46.

Zarras, A. Vassiliadis, P. Issarny, V. 2004. Model-driven dependability analysis of web services. *Proc. Intern. Conference on distributed objects and applications, October 2004*