Automatic Fault Tree Generation from SysML System Models

Conference Paper · July 2014

DOI: 10.1109/AIM.2014.6878163

3 authors:

Faïda Mhenni
Supméca - Institut supérieur de mécanique de Paris
55 PUBLICATIONS **360** CITATIONS
SEE PROFILE

Nga Nguyen
Ecole Internationale des Sciences du Traitement de lInformation
51 PUBLICATIONS **295** CITATIONS
SEE PROFILE

Jean-Yves Choley
Supméca - Institut supérieur de mécanique de Paris
158 PUBLICATIONS **764** CITATIONS
SEE PROFILE

Automatic Fault Tree Generation From SysML System Models

Faida Mhenni[1] and Nga Nguyen [2] and Jean-Yves Choley[1]

***Abstract*— Safety critical systems must satisfy increasingly rigorous safety requirements with shortening time to market. In order to validate these requirements, different safety analysis techniques can be carried out. However, traditionally, these safety analyses are performed manually with separate tools. Consequently, they are time consuming and error prone and are not automatically updated when the system models are.**

In this paper, a methodology is proposed to integrate safety analysis within a systems engineering approach. This methodology is based on SysML models and aims at generating (semi-) automatically safety analysis artifacts, mainly FMEA and FTA, from system models. Preliminary functional and component FMEA are automatically generated from the functional and structural models respectively, then completed by safety experts. By representing SysML structural diagram as a directed multi-graph, through a graph traversal algorithm and some identified patterns, generic fault trees are automatically generated with corresponding logic gates and events.

The proposed methodology provides the safety expert with assistance during safety analysis. It helps reducing time and error proneness of the safety analysis process. It also helps ensuring consistency since the safety analysis artifacts are automatically generated from the latest system model version. The methodology is applied to a real case study, the electromechanical actuator EMA.

I. INTRODUCTION

Today's technical systems are integrating more functionalities to offer more assistance and comfort to users. As a result, engineers are facing challenges to successfully design systems with high complexity while industrial competitiveness requires them to shorten time to market and reduce costs. The increasing complexity of manufactured systems makes their development more difficult since huge efforts are required to manage the complexity, maintain consistency through the development, and deal with numerous requirements relevant to multiple domains.

Technical systems may potentially be harmful to humans or facilities. Several safety analysis techniques have been developed in order to assess the potential risks of industrial systems. These techniques can be split into two categories : qualitative and quantitative approaches. Qualitative methods try to find the causal dependencies between a hazard on system level and failures of individual components, while quantitative methods aim at providing estimations about probabilities, rates and severity of consequences. To perform safety analyses, the two most traditionally used fault modeling techniques are Failure Modes and Effects Analysis (FMEA)and Fault Tree Analysis (FTA) [1], [2]. Other techniques such as FFA, HAZOP, etc. are also used [3].

The safety analyses mentioned above are usually performed manually and separately with independent tools, based on informal design documents [4]. Consequently, they occur late in the design process when the design is already finalized and thus, miss the opportunity to influence design choices and decisions. Furthermore, various safety studies are performed with different tools and at different stages of the design process and the consistency between these studies is not ensured [3]. Safety analysis results may also be inconsistent with the system models since there is usually a gap between the design process and safety analyses that are not automatically updated according to design changes.

Model-Based Systems Engineering (MBSE) approach is required to manage the complexity, enhance consistency and allow modeling and simulation of the whole system. In this approach, SysML [5], [6], an OMG standard is becoming more and more supported by industry because it provides a consistent, well-defined, and well-understood language to communicate the requirements and corresponding designs among engineers. That 's why SysML is choosen as the support modeling language in our work.

Joining the safety analysis together with the design process through SysML models will enhance consistency in the whole system, enable earlier error detection, avoid expensive redesign and reduce time to market delays. Some works about the integration of safety analysis in the early design stage, based on the functional models have been carried out both in industrial and academic domains [7], [8], [9], [10].

The aim of this paper is to propose a new methodology that integrates systems engineering with safety assessment. It allows the automatic generation of safety analysis artifacts, namely FMEA and FTA. FMEAs are generated firstly, in the functional then the structural design phases. Fault trees are generated next based on the structural model and the component FMEA results. Block patterns and graph traversal algorithms are used to generate automatically fault trees, based on the topology of the system. Then specific fault trees regarding to specific undesired top events are derived from autogenerated one.

[1] ISMEP, Saint-Ouen, France, `faida.mhenni@supmeca.fr`, `jean-yves.choley@supmeca.fr`

[2] EISTI, Cergy, France, `nn@eisti.eu`

This paper is organized as follows. In section II, a brief overview of the related work is given. Then the integrated methodology is presented in section III. A focus is given in section III-B to the automatic fault tree generation which is the main contribution of this paper. A case study is given in section IV. Finally, the paper is concluded in section V.

II. State of the Art

A. Integration of Safety Analysis and SysML

The integration of safety analysis and MBSE has been carried out in different researches [9], [7], [11], [12].

Dubois [9] proposed to directly include safety requirements in the design process with SysML. For this purpose, a SysML profile respecting safety standards called RPM (Requirement Profile for MeMVaTEX) was developed. The requirement stereotype of SysML is replaced by the MeMVaTEX requirement, by adding various properties such as *verifiable, verification type, derived from, satisfied by, refined by, traced to*, etc. In this work, only the integration of safety requirements is considered and these requirements are traced to the other elements of the model but safety analyses techniques are performed separately.

P. David attempted to integrate reliability analysis in the design process based on an MBSE approach with SysML [7], [13]. In this work, a preliminary FMEA report is generated by extracting the needed information on the dysfunctional behavior from functional SysML models. Dysfunctional models are then constructed using the AltaRica language in order to compute reliability indicators. Then the final FMEA report is obtained with help from experts in the safety domain. To facilitate such a deductive and iterative method, a database of dysfunctional behaviors is kept updated in order to rapidly identify failure modes in different analysis phases.

R. Guillerm in [11] and [14] proposes a method for safety management in complex systems engineering. The proposed method is based on the elicitation and declination of safety requirements. It combines three of the well knows safety analysis techniques, FMECA, fault trees and event trees to define all safety requirements at the system level and then, based on the system decomposition, declines these requirements at the component level. The information model that supports this approach is built in SysML.

B. Automatic Fault Tree Generation

Fault trees are widely used for safety assessment and reliability of systems for over 40 years [15]. Manual construction of fault trees is time consuming and error prone, especially for complex systems. To cope with this complexity, automatic generation of fault tree has been subject of many research works. The main difference concerns the type of the starting model based on which the generation is performed. In the following, a brief overview of some recent works is given.

In [15], fault trees are constructed from MATLAB Simulink models. The nominal model is built in Simulink and then is manually extended with failure behavioral information of the system. Based on this extended model and the classification of components, fault tree for a specific top event is automatically constructed.

An automatic generation of fault trees from AADL (Architecture Analysis and Design Language) models is proposed in [4]. In this work, the system architectural model is built with the AADL language and then is annotated with fault and failure information using the Error Annex, a sub-language of AADL. Based on the annotated model, fault trees are automatically generated in a commercial tool : CAFTA. However, as far as we know, there is no related work concerning the fault tree generation from SysML models as in our study.

III. Integrated SE/SA methodology

The proposed methodology aims at integrating safety analysis within the systems engineering approach. It consists in performing the appropriate safety analysis at the appropriate design stage based on the available system models. It allows starting safety analysis very early in the design process, since only an abstract functional representation of the system is available. It is then kept updated as long as the design evolves. A safety profile with different safety stereotypes is associated to SysML models in order to store these updates in the database. Consistency is maintained throughout the whole process.

Two well known safety techniques, FMEA and FTA, are used in this methodology. The FMEA generation will be described in section III-A. Then the fault tree generation is detailed in section III-B.

A. Overview of the integrated methodology -FMEA generation

In the scope of this paper, only the requirements definition and analysis as well as architectural design processes of the systems engineering approach are relevant to our work. The requirement definition and analysis process aims at providing a comprehensive and consistent set of requirements that will be the basis for the following design steps. In the second phase, system functions are identified from functional requirements and one or more functional architectures are defined and compared. In SysML, functions are represented by *activity* elements. Each activity can be detailed in an *activity diagram* to show how its input flows are transformed into output flows through sub-functions. This results in a hierarchical breakdown of the system functions with different levels of abstraction. At this level, safety analysis can already start. A preliminary functional FMEA is automatically generated from the SysML functional models and then completed by the safety expert. Derived safety requirements are integrated in the SysML model and, if necessary, a new design iteration is performed in order to take into account the resulting changes. For each design iteration, the safety analysis is also updated according to the updated system model. Then components are allocated to functions to define the physical structure of the system. In the same way, several physical structures are considered and compared according to different criteria like performance, cost and safety among others. For each physical structure, two complementary safety analysis artifacts (FMEA, FTA) are automatically generated from SysML models. First a preliminary component FMEA is automatically generated then completed by safety expert. To maintain consistency with the functional FMEA, the component FMEA contains the functions allocated to each component and their respective failure modes. These are translated into the corresponding component failure modes by the safety expert. The derived safety requirements from this step are also integrated and accounted for in the system models. The design and safety analysis are iterated as long as necessary. The final step is the generation of the fault trees to assess the fault propagation inside the system and check if the system satisfies the safety requirements, notably the failure rates.

B. *Automatic Fault Tree Generation from SysML IBD*

In this section, we will describe our method to generate fault trees automatically from structural diagrams, i.e SysML Internal Block Diagrams (IBD). IBD gives the internal structure of the system and the interactions among components. The interfaces through which the components interact are represented via standard and flow ports and the interactions are represented via paths between the corresponding ports called ”connectors”. If a failure occurs in one component, it will be propagated throughout the system via these paths. The idea of this work is to automatically generate fault trees based on the system IBD, using two concepts : *directed graph traversal* and *block design patterns*. Each concept is detailed hereafter.

An IBD can be represented as a directed graph $G = (V, E)$ where V is the set of vertices that composes of parts and some special ports called *external interfaces*, and E is the set of directed edges. The external interfaces are ports that are situated on the border of the IBD, which can be input or output port. In order to simplify the graph, we do not need to represent internal ports because they can be abstracted by edges directly connecting parts. It is also noted that the graph G accepts multi-edges that symbolize different kinds of items flowing between the two parts. So to build a fault tree for a given undesired top event, a graph traversal algorithm can be used to find out components relating to each other by using the directed edges. It follows the principle of backtracking from the hazard to the leaf events. The traversal starts at an external output port, traces back to nodes that are his predecessors and continue to visit the other nodes. Since a node can have several predecessors, a branch is finished when we reach an external input port, or when the node has been already visited.

To facilitate the fault tree generation, we also use the ”divide and conquer” principle by identifying some interesting patterns in an IBD. They are *entry, exit* and *feedback* patterns. Another kind of pattern, named *redundant* pattern, related to safety design criteria where a part block can have input ports coming from components assuring redundancy for higher reliability is also studied. Each pattern gives rise to a sub-fault tree and the whole fault tree will be assembled automatically by using the mentioned graph traversal algorithm.

The following subsections describe the recognized patterns as well as their generated fault trees. In order to gain space, all the patterns are grouped in an illustrating IBD in Fig. 1 and the comments written in a note show us the corresponding pattern.

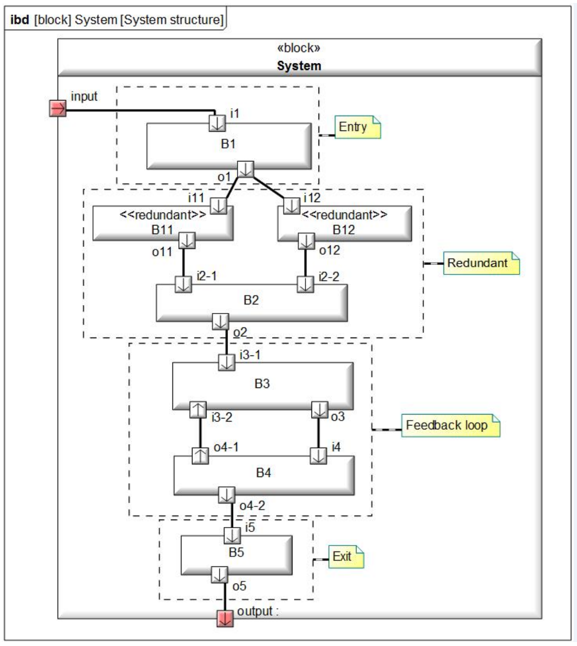

Fig. 1. IBD Block Design Patterns

1) Entry pattern: An entry part in an IBD is a block part that has at least one input port receiving item flow from outside the actual system/subsystem. In the generated sub fault tree (Fig. 2), this special input port will be transformed into a basic event representing a failure or error of a system component that is outside the actual block. We will have an OR logic gate whose operands are : the basic event representing the external failure, the internal failure of the part and all other eventual input ports.

2) Exit pattern: An exit part in an IBD is a block part that has at least one output port sending item flow out of the actual system/subsystem. In the corresponding fault tree (Fig. 3), this special output port gives rise to a top event undesired state of the actual block. We will have an OR gate whose operands are : the internal failure of the exit part and all other eventual intermediate events

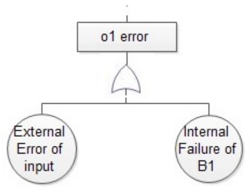

Fig. 2. Fault Tree for Entry Pattern in Fig. 1

that characterize failures coming from other input ports.

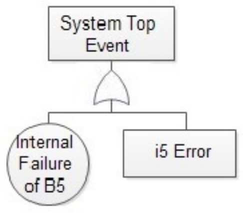

Fig. 3. Fault Tree for Exit Pattern in Fig. 1

3) Feedback pattern: By traversing the directed graph representing an IBD, if we encounter a node that has already been visited, we say that we have a loop or a feedback in the current graph. In Fig. 1, when generating the logic diagram for the output port o4-2 of the part B4, we need to take into account the input port i4 which comes from the output port o3 of the part B3. In its turn, the logic diagram of o3 must consider errors that may come from i3-2, propagated from B4. A cut can be realized here in order to not take into consideration the input ports such as i4 as an operand of the OR gate. The corresponding fault tree of the feedback pattern of Fig. 1 is illustrated in Fig. 4.

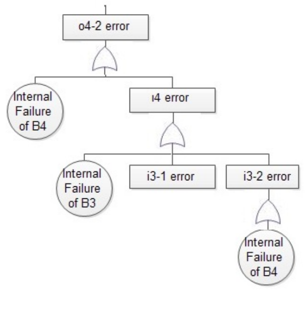

Fig. 4. Fault Tree for Feedback Pattern in Fig. 1

4) Redundant pattern: When a part in an IBD receives item flows coming from redundant blocks that carry out the same system function, we say that we have a redundant pattern (B2, B11 and B12 in Fig. 1). In this case, an AND gate is used for different faults coming from different inputs to model the fact that if there is no internal failure in the component B2, the component will not work only if all the redundant item flows fail. The fault tree for our example of redundant pattern is given in Fig. 5.

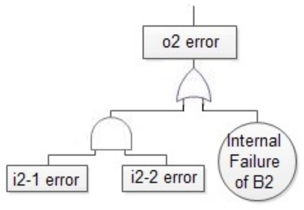

Fig. 5. Fault Tree for Redundant Pattern in Fig. 1

When the whole fault tree is generated automatically from an IBD by using the identified patterns and a graph traversal algorithm, we will have a generic fault tree for the corresponding system. In order to have a specific fault tree for an undesired top event failure, information from previous FMEA analysis can be used to refine this generic fault tree. Knowledge of safety experts is also very important in order to detail some branches with different failure modes or to cut out some unreachable branches, regarding the undesired top event failure. This proposal will be explained via the case study given in Section IV.

IV. Case study

The case study considered in this paper is an electro-mechanical actuator (EMA) used to actuate ailerons. The use of EMAs in flight control is increasing since they have many advantages [16] :

– Better environmental respect with suppression of hydraulic power and oil leak risks ;
– Weight saving on aircraft ;
– Maintenance cost reduction ;
– Performance increase and speed accuracy due to electric actuators.

Functional breakdown as well as the functional FMEA of the EMA are already presented in [12]. A logical architecture was also proposed in [12] and an improved version is given in Fig. 6. The three parts constituting the EMA are named respectively **Geared Motor with Encoder** for the electric motor, **Mechanical Transmission** for the mechanical transmission and **Embedded MCU with Power Bridge** for the electronic and the code that control the system.

A. Introduction

A preliminary FMEA is automatically generated from the IBD in Fig. 6 containing the list of components. To ensure the consistency of the component FMEA with respect to the functional FMEA, the functions allocated to each component and their failure modes are also added in the preliminary component FMEA. The safety expert then associates the functional failure modes into the corresponding component failure to obtain the final FMEA. Table I shows an extract of the final component FMEA for the EMA containing the failure modes leading to the “Aileron locked” undesired event, which will be used for the fault tree generation.

TABLE I

Extract of the component FMEA of the EMA

Component	**Function**	**Failure Mode**	**...**	**Local Effects**	**System Effects**
Embedded MCU with Power Bridge	Adapt Supply Energy	Electronic hardware defect	...	Motor stops or Motor rotates continuously	Alileron locked or Aileron continuously moving
	Translate Pilot Instructions	Software error	...		Aileron locked
		Hardware defect	...		Aileron locked
		Synchronization error	...		Aileron locked
		Memory defect or loss	...		Aileron locked
Geared Motor with Encoder	Adapt Mechanical Energy	Jam	...	Locked output	Aileron locked
	Transform Energy	Jam	...		Aileron locked
		Loss of structural integrity	...		Aileron locked or Free movement of the aileron
		Short-circuit between two windings	...		Aileron locked
	Measure Current	Hardware defect	...	Erroneous current measure	Aileron locked
Mechanical Transmission	Transmit Energy	Jam	...	Motor overloaded and output locked	Aileron locked

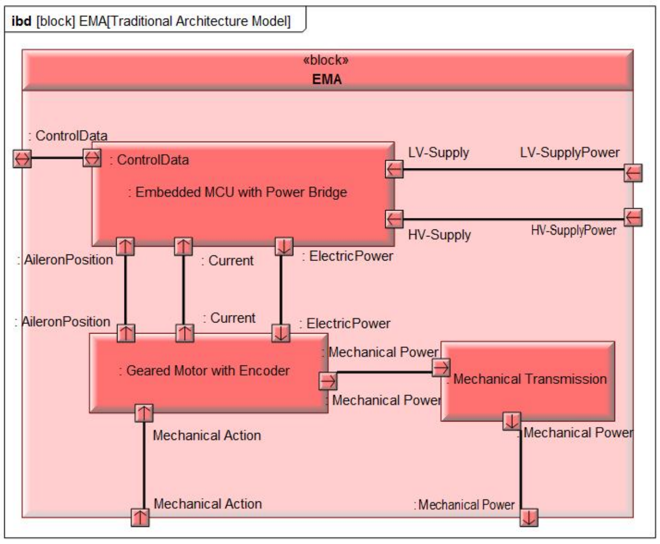

Fig. 6. IBD : EMA architecture

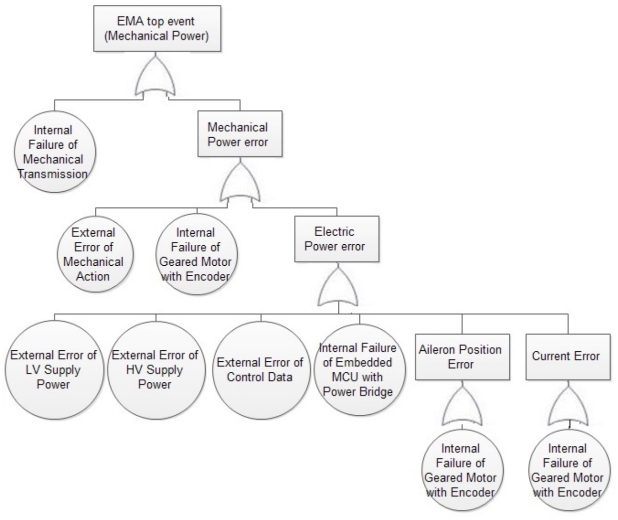
EMA top event (Mechanical Power)
Internal Failure of Mechanical Transmission
Mechanical Power error
External Error of Mechanical Action
Internal Failure of Geared Motor with Encoder
Electric Power error
External Error of LV Supply Power
External Error of HV Supply Power
External Error of Control Data
Internal Failure of Embedded MCU with Power Bridge
Aileron Position Error
Current Error
Internal Failure of Geared Motor with Encoder
Internal Failure of Geared Motor with Encoder

Fig. 7. EMA Generic Fault Tree

B. Fault Tree Generation

Based on the system model given in Fig. 6, a generic fault tree for a top event at a specific output of the system is automatically generated. This fault tree is built from partial fault trees generated according to the patterns and the graph traversal algorithm given in section III-B. The generic fault tree for the "Mechanical Power" output is given in Fig. 7.

Several undesired events can occur at each output. For each specific undesired top event, we can extract from the FMEA results the corresponding failure modes of each component that lead to the top event in question. The specific fault tree for the "Aileron locked" top event is given in Fig.8. In this fault tree, a branch is eliminated because no failure mode of the **Geared Motor with Encoder** leads to the "Aileron locked" event. Internal failure of some components such as **Geared Motor with Encoder** and **Embedded MCU with Power Bridge** is completed by their specific failure modes.

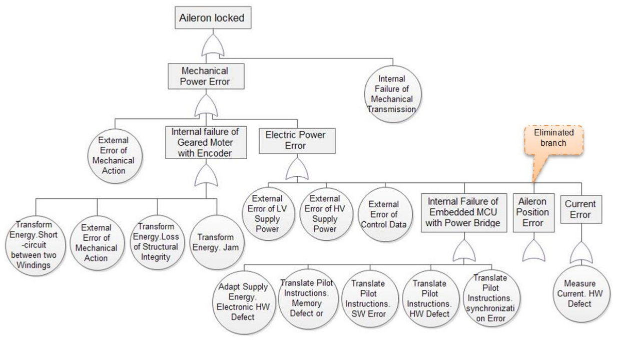
Aileron locked
Mechanical Power Error
Internal Failure of Mechanical Transmission
External Error of Mechanical Action
Internal failure of Geared Moter with Encoder
Electric Power Error
Eliminated branch
Transform Energy.Short -circuit between two Windings
External Error of Mechanical Action
Transform Energy.Loss of Structural Integrity
Transform Energy. Jam
External Error of LV Supply Power
External Error of HV Supply Power
External Error of Control Data
Internal Failure of Embedded MCU with Power Bridge
Aileron Position Error
Current Error
Adapt Supply Energy. Electronic HW Defect
Translate Pilot Instructions. Memory Defect or
Translate Pilot Instructions. SW Error
Translate Pilot Instructions. HW Defect
Translate Pilot Instructions. synchronization Error
Measure Current. HW Defect

Fig. 8. EMA Specific Fault Tree for "Aileron locked" top event

V. Conclusions

In this article, a methodology to integrate safety analysis within a SysML-based systems engineering approach is presented. This methodology concerns some well known safety analysis artifacts, i.e. FMEA and FTA. The automatic generation of these artifacts will help reducing time and errors. Consistency between safety analyses and design is maintained throughout this integrated process since FMEAs and fault trees are directly generated from the latest system model versions. Coherence between different safety analyses is also maintained since each step is based on the previous analysis results.

Our FMEA generation has already been implemented by using a program written in Python language. This program parses an XMI file representing the SysML models and generates an Excel file corresponding to the functional or the component FMEA worksheet. This Excel file can be then completed by safety analysists, and parsed again by another Python script in order to insert new information into the safety stereotypes intergrated in the XMI file, via the safety profile extension. The next step of our work is to carry out the automatic fault tree generation by extending the existing software framework.

References

[1] C. A. Ericson, *Hazard Analysis Techniques for System Safety*. John Wiley & sons, 2005.

[2] N. Xiao, H.-Z. Huang, Y. L. L. He, and T. Jin, "Multiple failure modes analysis and weighted risk priority number evaluation in FMEA," *Engineering Failure Analysis*, vol. 18, pp. 1162–1170, 2011.

[3] Y. Papadopoulos, J. A. McDermid, R. Sasse, and G. Heiner, "Analysis and synthesis of the behaviour of complex programmable electronix systems in conditions of failure," *Reliability Engineering & System Safety*, vol. 71, pp. 229–247, 2001.

[4] A. Joshi, P. Binns, and S. Vestal, "Automatic generation of static fault trees from AADL models," in *Proceedings of the IEEE/IFIP Conference on Dependable Systems and NetworksŠ Workshop on Dependable Systems, DSN07-WADS, Edinburgh, Scotland-UK, June*, 2007.

[5] (2013) www.omgsysml.org.

[6] *OMG Systems Modeling Language (OMG SysML)*, OMG Std., June 2012.

[7] P. David, V. Idasiak, and F. Kratz, "Reliability study of complex physical systems using SysML," *Reliability Engineering and System Safety*, vol. 95, no. 4, pp. 431 – 450, 2010.

[8] R. Laleau, F. Semmak, A. Matoussi, D. Petit, A. Hammad, and B. Tatibouet, "A first attempt to combine SysML requirements diagrams and B," *Innovations in Systems and Software Engineering*, vol. 6, pp. 47–54, 2010.

[9] H. Dubois, "Gestion des exigences de sûreté de fonctionnement dans une approche IDM," in *Journées Neptune Nř5, Paris, France*, 08 avril 2008.

[10] F. Thomas and F. Belmonte, "Performing safety analyses and SysML designs conjointly : a viewpoint matter," in *Complex Systems Design & Management*, 2011.

[11] R. Guillerm, H. Demmou, and N. Sadou, "Global safety management method in complex system engineering," in *Complex Systems Design and Management Proceedings of the Fourth Inernational Conference on Complex Systems Design and Management, CSDM 2013*, 2013.

[12] F. Mhenni, J.-Y. Choley, A. Rivière, N. Nguyen, and H. Kadima, "SysML and safety analysis for mechatronic systems," in *Mechatronics (MECATRONICS), 2012 9th France-Japan & 7th Europe-Asia Congress on and Research and Education in Mechatronics (REM), 2012 13th Int'l Workshop on Research and Education in Mechatronics, IEEE MECATRONICS REM, Paris*, 2012, iSBN 978-1-4673-4772-3.

[13] P. David, "Contribution à l'analyse de sûreté de fonctionnement des systèmes complexes en phase de conception : application à l'évaluation des missions d'un réseau de capteurs de présence humaine," Ph.D. dissertation, Université d'Orléans, Novembre 2009.

[14] R. Guillerm, "Intégration de la sûreté de fonctionnement dans les processus de l'ingénierie système," Ph.D. dissertation, Université de Toulouse, 2011.

[15] F. Tajarrod and G. Latif-Shabgahi, "A novel methodology for synthesis of fault trees from MATLAB-Simulink model," *World Academy of Science, Engineering and Technology*, vol. 17, no. 5, pp. 1256–1262, 2008.

[16] D. Mami, "Définition, Conception et Expérimentation de structures d'actionneurs électromécaniques innovants incluant par conception des fonctionnalités de sûreté et de sécurité de fonctionnement," Ph.D. dissertation, Université de Toulouse, Institut National de Politechnique de Toulouse, 2010.