Reliability study of complex physical systems using SysML

Pierre David*, Vincent Idasiak, Frédéric Kratz

Institut PRISME – ENSIB, 88 Boulevard Lahitolle, 18020 Bourges Cedex, France

A R T I C L E   I N F O

Article history:
Received 1 December 2008
Received in revised form 26 November 2009
Accepted 28 November 2009
Available online 6 December 2009

Keywords:
Design process
FMEA
Reliability studies
SysML

A B S T R A C T

The development of safety critical systems becomes even harder since the complexity of these systems grows continuously. Moreover, this kind of process involves the use of powerful design methods and precise reliability techniques that utilize dissimilar models and construction policy. In this article we propose a method to unify and enhance this process by linking functional design phase using SysML with commonly used reliability techniques such as FMEA and dysfunctional models construction in AltaRica Data Flow. We present how SysML models can be analyzed automatically in order to produce an FMEA and expose a parallel between SysML models and AltaRica Data Flow ones. The given approach is structured around a database of dysfunctional behaviors that supports the studies and is updated by the obtained results. We exemplify the approach to analyze a system of level controlling of a tank.

1. Introduction

Over the past decade, the complexity of safety critical systems has considerably grown. Systems become larger and involve a huge diversity of technology. For instance, software components are widely embedded in systems regrouping electronic devices, sensors, actuators and mechanical structures. Developers impose on these new systems precise requirements concerning dependability, performance, correctness and safety. Well-specified modeling methods and languages are needed to manage their development, but also for the validation of their design. The study of these new systems brings up the issue of performing reliability studies during the conception phase.

This problem has been tackled in numerous works [1–5] that permitted to identify the challenges of such analysis. The objectives of these techniques are to furnish rapidly quantitative and qualitative indicators on the safety, reliability and performances of the studied system. These results must be obtained only with the knowledge of the functional behavior. In the best case analysts possess a database of the lesson learnt on the behavior of similar components used in previous solutions. The difficulty of this activity is to be able to construct and analyze the dysfunctional models rapidly enough to respect the time allocated to the design phase in order to cope with evolving normative context or to avoid delaying system delivering and allocated to the design phase in order to cope with evolving dysfunctional models rapidly enough to respect the time allocated to the design phase in order to cope with evolving normative context or to avoid delaying system delivering and keeping a competitive advantage [6].

In order to steer the conception process by a study of reliability enhancement, it is capital to combine efficiently the design process and the reliability evaluation. Safety and reliability analyses need domain-specific formal methods. Since a technical expertise is needed for their use, reliability studies are too often treated as isolated activities [4]. The main difficulties for the development of safety critical systems are to master, during design phase, the complexity of the combination of various technologies and to obtain reliability and safety indicators fast enough to influence design decisions as well as to avoid too late risk identification leading to a costly reengineering [1,6]. Moreover, as complexity grows, reliability expert will face more difficulties to handle system functioning. Therefore, supports for system analysis have to be provided. Standards as the IEC 61508 [7] propose a succession of tasks to develop safety systems. Nevertheless, these standards precisely define the analysis to produce but do not provide methods to efficiently conduct them [8]. Therefore, there is a real need in today’s industry to obtain tools and methods to support the design of safety critical systems, by linking classical methods of each kind of specialist call to work on those projects. The objective of this paper is precisely to propose a way to bridge the gap between design process and the already well-mastered techniques of reliability studies.

Significant works addressed these challenges, they were performed mainly in aeronautics and software industry. One of the most significant studies was performed by the HIDE project [2]. This project focused on the validation of design models written in UML [9] for software validation purposes. Also for safety critical software design, [3] proposed a framework integrating each step of software construction, including dependability processes. The dependability analysis is not restricted to the design phase but is a connecting thread throughout the

* Corresponding author. Tel.: +33248484066; fax: +33248484050.
E-mail addresses: pierre.david@ensi-bourges.fr (P. David), vincent.idasiak@ensi-bourges.fr (V. Idasiak), frederic.kratz@ensi-bourges.fr (F. Kratz).

0951-8320/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ress.2009.11.015
2. SysML for safety critical system design

As we mentioned in the introduction, our approach is to be applied on SysML projects. SysML is having the same success in system engineering as UML in the software industry. SysML is presented in [19] as a general-purpose modeling language for systems engineering applications. This specification has been set up on the precious experience gathered on UML 2.0 and on successful engineering methods as Composable Object (COB) technology and Enhanced Functional Flow Block Diagram (EFFBD) [17].

2.1. SysML for complex systems engineering

2.1.1. The inheritance from UML 2.0

SysML is constructed as a subset of UML 2.0 complemented by additional modeling possibilities. SysML shows the same ability than UML 2.0 to model large systems [19]. As UML 2.0 did for software modeling languages, SysML proposes to be a common practice for system engineers replacing the wide range of modeling languages and techniques currently used for complex system development.

Since SysML is built as a UML profile many lessons learnt on the use of UML for complex system development must be considered. UML has reached its objectives and is currently widely used in the software industry. Since UML 2.0, this language has not been restricted to software design; thanks to its object-oriented approach, it is exploitable to describe any type of system. Many published works propose the use of UML for system engineering and demonstrate all the advantages of object-oriented methods in complex system design like [18].

Nevertheless, UML suffers of several drawbacks that justified the creation of a modeling language specified for system engineering. First, UML has not been much used for complex systems design because it is overly software-oriented semantics. Secondly, UML lacks modeling possibilities for concrete physical systems. The modeling of continuous flows, either of material, energy or information is badly treated. Finally, no modeling constructs exist in UML 2.0 to realize a real management of requirements all along a design project. Therefore, system engineers always needed a modeling language allowing sharing and unifying discipline-specific parts.

That is why OMG created SysML for specifying, analyzing, designing and verifying complex systems [17]. SysML has been constructed as an extension of UML adapted for system engineering. It brings new means for system modeling and requirements. In [19] 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”. 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 (PD) describe the equations linking the multiple parameters of the model.

The objective of SysML creators was to provide a simple but powerful modeling language for system engineers. Therefore, the software-oriented vocabulary of UML is abandoned and the specification insists on requirements management and continuous system modeling. As mentioned in SysML specification, the fundamental design principles of model written in SysML are:

- Requirements-driven: SysML provides modeling constructs to allocate and manage requirements.
- Partitioning: systems are modeled as association of components and regrouped in packages. The packages show up logical group and reduce cross dependencies.
- Layering: The various packages can be layered and organized between several levels of granularity in the system model.
- Interoperability: SysML reuses the XML interchange format from UML. The models are exchangeable between software tools. Rules of coding ensure these operations. The development of new tools exploiting SysML for new application, as reliability analysis, is facilitated.

SysML creators decided to simplify SysML and reduced the number of diagrams in the specification. The diagram presented in Fig. 1 shows the partitioning of the different kind of diagrams. The organization of diagrams reuses the well-accepted decomposition for system modeling: the behavioral part on the one hand and the structure on the other hand. This decomposition is well suited for the analysis of multi-technological complex systems. We will examine in the remainder of this section the utility of different diagrams for our purpose and specially mention the novelties brought by the new diagrams.

2.1.2. Managing Requirements in SysML

The management of Requirement is a central task for current development projects. For new product design, it is crucial to formally express requirements, to allocate them to parts of the systems and to verify them at various steps of the development. As a proof of the importance of the task we note the large offer (more than 40 tools) existing on the market of requirement management tools referenced by the INCOSE (International Council on Systems Engineering) requirement management tools survey [20]. For safety critical systems the requirements are not only used to fix the expected functional behaviors, but also to indicate the safety and reliability needs of the system. As a language for system engineering SysML focuses on the management of requirements, namely a new type of diagram dedicated for their declaration has been created.

2.1.2.1. Requirement Diagram. SysML provides modeling constructs to capture requirements and relate them to other modeling elements. A requirement specifies a function that the modeled system must perform or a performance condition the system must achieve. Requirements structured in this language can be allocated to precise component of systems. Moreover, they can be organized through the use of various kinds of dependencies. The syntax of Requirements is given in Fig. 2.

Requirements are traced thanks to their unique identifier. User can allocate them diverse properties, such as verification status. In current project, requirements capturing and allocation are crucial and intricate. SysML furnishes many design constructs to cope with the complexity of these processes. Several requirements relationships are specified, that enable the modeler to represent hierarchy or composition between requirements, as well as verification mechanism.

These elements make it possible to obtain reusable requirements and to specify their validity in different contexts. This is useful to manage requirements reused in product families (versions/variants). This guarantees a concordance of requirements expression between similar projects and insures an efficient repercussion of requirements among subprojects.

A requirement can be derived to more precise ones. This allows the modeler to apply a general requirement at the next granularity level of system hierarchy, and to keep the traceability to the top-level requirement. A requirement can also contain sub-requirements built to specify the multiple goals induced by the top-level requirement. In both cases, the relationships between requirements are explicitly expressed by the properties Derived, DerivedFrom, SubRequirement and ParentRequirement of requirements. The satisfy relationships perform the requirement allocation to system elements. They describe how system parts satisfy one or more requirements. This relationship will be very valuable in the safety or reliability study of systems.

Finally the modeler can specify the methods used to verify the requirements, by associating a test case element to requirements. These test case elements are accessible with the property VerifiedBy. Studying test cases can give indications on how to detect a failure during the use of the system. These test cases can be organized hierarchically to reflect general practices or combination of tests. Test cases own a verdict property that illustrates the verification results. Identifying these test cases is very valuable in reliability and safety analysis. Starting from a

![Fig. 1. The SysML diagrams taxonomy as presented in [17].](image-url)
component failure, the impacted requirements are previously obtained. Then test cases employed to verify these requirements are identified. There exists two kinds of test cases: those performed in the validation phase (off line tests: tests on prototypes, tests on samples in mass production) and those deployed during product utilization (test in line: diagnosis, error recovery detection). This second kind refers to diagnosis policies embedded in the system. This information on tests deployed to achieve system survey is also crucial in order to conduct accurate and efficient testability studies. The problematic of testability raises many challenges for safety-critical system design; nevertheless, we decide not to deal with this problematic.

It is important to note that previous techniques proposed for UML models can be simplified and maintained using SysML. For instance, [4] proposed an efficient manner to formalize the expression of requirements using PFS in a UML environment. In this work requirements are depicted thanks to StateChart diagrams. StateCharts still exist in SysML and design constructs as refine or realization relationships are available for all graphical entities. Therefore, reusing the works of [4], we can imagine declaring a requirement in a Requirement Diagram and link it to a StateChart Diagram. This link will have the advantage to be explicit and easily identifiable, so that the traceability of the requirements and its complete definition will be much clearer.

In order to support our analysis, it would be profitable that designers follow some kind of practice. It is recommended that designers focus on allocating the requirements to model elements. A first step of design should be the requirements declaration by decomposing the functional objectives in more and more precise requirements. Then the architecture of the system should be described and the modeler should notify which requirements the modeled parts are satisfying. Safety requirements can then be indicated and allocated to the system components.

2.1.3. Modeling structural and physical constraints

New construction possibilities address the issue of physical behavior modeling and more generally mathematical relations expression. This provides a way to model a large panel of physical system.

2.2. Constraint Blocks and Parametric Diagrams

SysML provides modeling elements that permit to set up networks of constraints on physical properties of systems. These are typed as Constraint blocks. They own a constraint description (Constraints) and properties (Parameters), which are the combined parameters. Constraints are mathematical or logical expressions that bind the parameters of blocks defined in the model. Constraint blocks can also be used to declare the computation of performance indicators describing systems features. These criteria can be dedicated to compare alternative design solution.

Constraint blocks are generic, so they define reusable constraints that can be grouped in domain-specific libraries and reused in different projects. To apply a constraint in a specific context, modelers have to utilize parametric diagrams. In these diagrams, constraints are nested by the link created between their parameters. Therefore complex constraints can be defined in terms of more basic ones. Moreover, in parametric diagrams, basic parameters as mass, size or resistance are defined by the attributes of components described in the model. To invoke such parameters, SysML provides a naming policy that permits to refer to attributes deeply declared in model, by giving the path among blocks and packages to access these parameters with dot notations.

Constraint expression can include time as a property or time dependent properties. Additionally, constraints can be conditioned on functional configuration. In this case, the constraint will be applicable only in a precise functional value subset of its parameters. The determination of parameter state can also be a
constraint on diverse system properties. For instance, the computation of the flow of a pump could depend on the state of the material that is pumped (steam or liquid). Moreover, this state depends on the value of other parameters (pressure, temperature).

SysML is well suited for constraint expression. Nevertheless, it does not provide or specify a method to interpret them. The user will have to connect its own mathematical tools to solve the defined constraint systems. Some works already exist that provide effective tools to reuse SysML constraint specification and solve them. In [19] the authors introduce the background of the SysML parametric diagrams and their relation with COB technology; they explain how parametric diagrams support simulation-based design. In fact, they demonstrate how the models can be exploited with analysis tools and equation solver as XaiTools.1 They show how SysML parametric diagrams helped them to formalize and analyze spring systems and how they connected their models to the solver. They conclude that parametric diagrams provide several advantages over traditional engineering analysis representations. These are a richer semantics, a greater expressivity and from a view of knowledge in a modular and reusable manner. The reuse of this technology for reliability indicators computation is not treated but is to our point of view clearly adaptable to this field.

2.2.1. Blocks, Connectors, Ports and Flows

Blocks are the basic concepts of SysML. Blocks are modeling elements that describe each modular unit of a system. A block defines a set of features and operations that represents the structure and behavior of a component. The blocks are declared and organized hierarchically with relationships within the Block Definition Diagrams (BDD). In BDDs, blocks own properties, which define their attributes and sub elements as parts and ports, moreover, they possess the set of operations that they are able to perform. Their internal structure described by connectors between their properties is shown in Internal Block Diagrams (IBD). In IBDs, components are represented by parts. Parts model the role of a component, whose type is defined by a block, in the system or a subsystem. The parts that compose a given component appear in the block that types the component as properties. Visually, they can be shown as a compartment of the block definition in BDDs.

In SysML models, many construction elements allow user to model relationships and dependencies between components. The required information on links between components appears in two types of SysML diagrams: Sequence Diagrams (SD) and IBDs. As explained before, IBDs declare a structural view of the system, whereas SDs define interactions between parts in given scenarios and are centered on exchanged commands. First, we will focus on information that can be extracted from IBDs. In IBDs connectors that link ports attached to the parts model connections between components. The connectors can carry specified item flows, which express what flows through the connection on a precise context. The ports are divided into two types: standard ports (inherited from UML 2.0) and flow ports ( SysML extension). Standard ports are well suited for service-oriented architecture description mostly used in software design. Flow ports defined what might flow between a block and its environment. Flow ports are typed so that data, material or energy that pass through the port are well defined. The ports are linked in IBDs by connectors. This allows to show the functional connections and to model how energy, material and data flow throughout the systems. A connector expresses that the elements connected on both ends have the same value and share the same type.

In SDs the blocks communicate with messages. Messages can be seen as call for action. They model the command requests between actors and the system or between parts of the system. Messages carry information about their sender and receiver in addition to the called operation.

2.3. SysML for reliability studies

We have seen in the preceding section that many constructs available in SysML promise to be utilizable for reliability analysis. Since SysML is a new language no works currently explore this issue in detail. Nevertheless, we can note that SysML possesses the same abilities as UML for this type of study and provides new constructs even more adapted. In order to discuss the appropriateness of using this type of language for our purpose, we can first observe what has been done with UML on this subject.

Integrating reliability requirements or conducting reliability or risk studies on UML models raises many challenges for model construction and exploitation. The problem of modeling reliability with UML is presented for software systems in [11]. The authors propose to define new UML stereotypes to describe the dysfunctional behavior of systems. The models using these stereotypes are automatically converted to use the SHARPE tool2 and compute Fault trees and Markov chains. Many UML model translations to Petri Nets or Fault Trees have been defined [12–14]. These translations were achieved for performance or reliability evaluation. Other similar approaches were developed in [22,23]. They have also been used for specification checking on StateChart, where UML models are a starting point to integrate safety and formal analyses [4,15]. Nevertheless, the majority of these works are only used for software analysis. Their application to system engineering remains tricky and does not tackle issues like continuous flow degradation or requirements satisfaction. Moreover, describing reliability aspects of systems with UML compels the modeler to use and define stereotypes that are not fixed in the standard specification. This is a significant drawback, since these adaptations, proper to each user, forbid the use of a common XMI (XML Metadata Interchange) for file exchange, conversion and reuse.

The use of SysML mitigates some of those drawbacks by means of new diagrams. Actually, requirements diagrams will be precious to express requirements. Besides, modeling reliability aspects has clearly been an interest for SysML developers. Parametric diagrams represent constraints on system properties values such as reliability indicators. Furthermore, works such as [21] prove the possibility to connect parametric diagrams with function solvers, allowing the computation of functional indicators (torque, force). By extension, the calculation of reliability indicators will be realizable. This aspect will be presented as a part of the Dysfunctional Behavior Database (DBD) that we propose to set up using SysML. However, since SysML is a recent specification, its use for reliability study has not been widely described. Current works only highlight the potentiality for requirement expression and constraint calculation without giving methods to use them and application examples.

Nevertheless, a recent work [24] explores how SysML may support the overall functional safety approach of IEC 61508. The methodological guide of this standard expresses the tasks to perform in each step of the safety lifecycle. The use of semi formal languages as SysML is not evoked in this guide. Nevertheless, [24]

---

1 X-Analysis Integration Toolkit is a trademark of Georgia Institute of Technology.

2 Symbolic Hierarchical Automated Reliability and Performance Evaluator developed at Duke University.
studied when each type of SysML diagram could be used during this process. The article does not describe how constructing and analyzing the diagrams, but emphasizes that the needed information for each step can be model. The purpose of our work is to show how exploiting the diagram in the reliability analysis phases and to steer the process towards good practices. In the next section we give our approach using SysML to integrate reliability analysis and tools in the design process.

3. Integrating reliability study in design process

The aim of our work 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 studies is that they delay and complicate the design process. Consequently, we are developing a method using techniques that can be mainly realized automatically by software tools. We propose a deductive and iterative method in three steps (Fig. 3):

- Deduction of the dysfunctional behavior with an FMEA, identifying the impacted requirements.
- Construction of a model integrating functional and dysfunctional behaviors with a formal language.
- Analysis and quantification of dysfunctional behavior.

The separated steps of this approach can be directly reused in standardized approach as IEC 61508 and its domain-specific derivations IEC 61513 [25] (nuclear plant), IEC 61511 [26] (process industry) or IEC 62061 [27] (manufacturing sector). In fact, these steps are parts of hazard and risk analysis and overall safety validation phases defined in the IEC 61508.

Each type of analysis requires dissimilar models and methods. Our role is thus to enhance the link between each activity. That is why we create tools to automatically conduct risk and reliability studies from the designer model written in SysML. The goal of each step is to generate new knowledge or to automatically create new models for various analyses. During the whole process, we focus on tracing the impact of failures on the performance and reliability requirements.

At the beginning of our approach we assume that a model expressing the architecture and the functional behavior of the system has been set up by designers. Additionally, as SysML offers great facilities to manage requirements we expect that performance and reliability requirements are declared. It is possible to use the approach without requirements declaration, but valuable information would be lost without them. The search of the initial dependability and reliability requirements can be helped by classic practices like Preliminary Hazard Identification or Functional Hazard Analysis. The second assumption is that developers have access to information on the behavior of the components usually utilized in design. This base of information can be the result of studies on previous exploitation of similar components in system already in use. In Section 4 we will propose a way to formalize this database using SysML. To this point of the development, the first operation to analyze the system reliability is to find what may cause failures in the functioning of the system. The most utilized method to perform this task is the FMEA.

Therefore, the first step of the instrumented method is the establishment of an FMEA. This FMEA is generated from the study of SysML models. We use an FMEA to find the basic dysfunctional behavior of the system. FMEA 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 failures. This step is very critical in our approach because it has to highlight the Failure Modes (FM) that will be qualified and quantified in the remaining of the study. In order to generate the FMEA from SysML models, we apply techniques of data

![Fig. 3. An approach from functional model to reliability indicators.](image-url)
translation using files exploiting XML (eXtensible Markup Language). We employ data analysis and organization techniques, namely using a DBD written in SysML, that serves to automatically exploit document traditionally built in engineering companies. Section 4 details this step and gives examples of algorithms implanted for its realization as well as the management of the DBD. We present an original approach for the automatic synthesis of FMEA. This work provides some novelties for reader searching new techniques for improving FMEA management. After this phase is performed, it is necessary to compute the impacts of the identified FM at the system scale. The next phase uses formal languages and formalism that reliability engineers developed. Those formal languages are mandatory in order to accurately compute the performance and reliability indicators.

The next step in our method is to compute models for reliability evaluation, thanks to the identification of the dysfunctional behavior performed by the FMEA. The model to be constructed must reflect the functional behavior as well as the FM identified during the preceding step. There exists numerous formalisms to perform this task; we can refer the AltaRica [28] or Figaro [29] languages or widely used concepts like Generalized Stochastic Petri Nets (GSPN), Stochastic Activity Networks [30] and Markov Chains. We focus on using languages or way of modeling that process component by component and link them with ports of flow propagation. AltaRica and Figaro use syntaxes very close to this point of view as the models are constructed as lists of nodes representing components or subsystems. This compositional approach is also applicable by using layered Petri Nets. The construction of the model is done component by component, so that the decomposition established in the FMEA can be naturally reused. Moreover, these models highlight the error propagation between components by describing the flow ports and the state of the flow passing through them. Those models clearly deal with the dynamic aspects of system behavior, like the influence of data and energy transmission throughout the architecture of the system. A whole simulation of the system can be processed or a limitation to several parts of it, by using fault injection techniques. Experts of reliability studies generally build these models. What is interesting to do is not to change their method, but to help them building the models by enhancing the reuse of knowledge rose in the previous steps of the process.

AltaRica, Figaro and GSPN models are used in software tools (respectively, BPA DAS – former OCAS module of Cecilia workshop – [28], KB3 [31] and Jagriff edited by Dassault Data Services). These software tools enable to quantify reliability indicators, such as the global failure rate, the mean time to failure or Probability of Failure on Demand (PFD) [5,32]. Solutions to compute fault trees also exist (e.g. Aralia Sim Tree3) and give a means to find failure scenarios and also to compute failure rates. The last step of our approach is thus to use those kinds of tools on the previously obtained models in order to compute the necessary results for design evaluation and orientation. Those results can also be used to refresh the information of the DBD. Using languages already instrumented is a priority for use as our aim was to enhance and join the current efficient practices.

In order to render this approach effective, we are developing tools for the automatic creation of each model and for the construction of the files needed by the reused tools. In the next section, we will discuss the realization of the first step. We will present the problems raised by the automatic synthesis of FMEA and mention the existing works. Then, we will describe our policy for an automatic synthesis of FMEA. As the DBD is crucial for this step, we will define its functioning and interest for this process.

4. Automatic computation of FMEA from SysML functional models

FMEA is the standard method for risk analysis in the design phase [33]. 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. This analysis technique is widely used in all industrial domains. Several norms (IEC 60812, MIL STD 1629A, BS 5760) are presenting its use and adapt its vocabulary for domain-specific needs. FMEAs are deployed in most of industrial projects but suffer from several weak points that we decide to address using automation of its process.

4.1. Challenges for Automatic FMEA creation

4.1.1. Previous works on FMEA

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 risks and failure at an early stage of the design process. It guarantees the exhaustive identification and the classification of risks. Finally, it allows identifying 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: ameliorating the organization or supporting the study with software. Bassetto, in [34], thanks to an experiment on a whole plant, gives organizational advises to involve the engineers and enhance their perception of FMEA. 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 softwares for FMEA management. By the way, many researchers have explored how software could be useful for FMEA development. Therefore, there exists various helps brought by software tools:

- Maintaining a return on experience database [34].
- Help to fill the FMEA table.
- Automatic synthesis of parts of the FMEA table [35,36].
- Organizing and highlighting the relevant elements [37].
- Help for the use and creation of failure taxonomy [33,34].
- Model generation from FMEA table [38].
- Managing simultaneous failures [1,38].

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 [1,28], 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

primary failures that could occur, but to find multiple failures that could be significant. Finally, we can mention works on the semantics used in FMEA. To solve the ambiguity of coupling natural vocabulary with quoted values, [39] propose to use fuzzy logic rules. This technique is still experimented in recent works [40,41] on various kinds of system (engine, sewage plant). [42] pushed this reasoning deeper. Namely, they identified that the classic numerical rankings bring ambiguities since dissimilar combination of factors can give same Risk Priority Numbers. Therefore, they utilize fuzzy rules and Grey theory approach that allow to use linguistic terms in the analysis and render the classification of risks much more reliable and precise. Those different works helped us to identify the crucial points for the automation of FMEA creation.

4.1.2. Requisites for automatic FMEA synthesis

We have identified two major points for the success of an automatic synthesis of FMEA. These are the models 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 system to render the analysis and render the classification of risks much more reliable and precise. Those different works helped us to identify the crucial points for the automation of FMEA creation.

Those aspects can be modeled with SysML by the use of BDD and IBD. A data and flow transmission view also exists in SD and IDB. Finally, the possibility to classify the objects gives to this database a great capacity to furnish easily reusable uniform information.

It is essential to construct a database containing the lesson learnt on the failure modes of utilized components. [34] exposes that: “if each risk component possesses its own typology, the automatic generation of risk can be envisaged”. 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 [33] 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 mainly the components and failure modes. This exhaustive list can be reduced 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 authorize to reason on effects and characteristics such as severity and occurrence. The use of taxonomies for the database exploitation in the case of automatic FMEA synthesis is essential. Therefore, we must impose on the designer the respect of a typology for the definition of his model. The best way is to use as much as possible his own vocabulary and to let him modify and enrich the database with his own experience. That is why we propose to build a DBD using SysML, which is the language we suppose the modelers will use. We define an architecture of DBD that future user can decide to use entirely or only partially if they do not possess all the required information. Simple mechanisms exist in our software that manage and maintain the database and its taxonomy. For instance, new terms are added only after its comparison to the already registered terms and the validation of the user.

4.2. Dysfunctional Behavior Database in SysML

This kind of database is primarily needed for rapid FM identification during the FMEA process. To realize a major part of FMEA automatically it is mandatory to be able to catch the information available for the studied component. Several database are currently used to refer this kind of data. For instance, the KB3 tool and the BPA DAS environment embed component databases were the dysfunctional behaviors of certain sort of components is described. Nevertheless, those databases are not available as open source resources and are implemented in application-dedicated languages that are not accessible for the majority of the engineers’ community. We chose to define a general architecture of DBD using SysML in order to ensure coherence with the analyzed models and to furnish a reusable framework.

Since we use FMEA as a step among a more general approach, it is relevant to use this database as a central resource for the reliability analysis. Therefore, we studied the use of SysML to construct a BDD that embeds information from failure rate computation to the failure behavior expression. For a whole reliability study process the significant information on the dysfunctional aspects concerning a component are:

- The List of FMs,
- Their Failure Rate, obtained by a Failure Rate Computation,
- Their Failure Law,
- Their Behavior and impact on nominal properties of the component.

These are the set of information needed to integrate and save the results of very different kinds of reliability analyses. We
worked on a metamodel for the DBD that provides a way to present the data needed for AltaRica, GSPN or Figaro models creation and also for storing FMEA results and more generally all necessary information to characterize the dysfunctional behavior of a component. The metamodel of this DBD is given in Fig. 4.

The DBD is built around the component types. Each component type has its associated model elements. The coherence with the model of the designer is made by the concordance between the name of the component type in the model and their declaration in the DBD. That imposes on the designer to use for a component type the same typography as the one registered in the DBD. During the reliability analysis process the information of the DBD can either be used to progress in the study either be updated or completed by new knowledge.

A Component block describes a type of component in the DBD. FMs can be associated to a Component block authorizing the declaration of each known FM. A FM has a Failure Profile that describes its parameters in its occurrence probability law. For example FMs that follow an exponential distribution have a Failure Profile reduced to one parameter: the Failure Rate generally noted $\lambda$ in the literature. We associate to this property a constraint block stereotyped by FPcomputation (Failure Profile computation) that declare the function linking environmental and structural parameters to the parameter of the occurrence probability law that follows the FM. This block allows adding in the DBD computation functions as those declared in document as the FIDES database for electronic components reliability prediction. We also associate to each FM another constraint block stereotyped by FailureLaw, which defines the occurrence probability of failure applicable to the FM. The failure laws declared are written in the constraint property of the FailureLaw block and concerns the parameters of the same block. A simple instance is to define the exponential law with a constraint property $P=e^{-\lambda t}$ and constraint parameters $P$, $\lambda$, and $t$. Lastly, the behavior of the FM is defined through a statemachine. This statemachine models the transition between the nominal and faulty state of the component. Moreover, it can model the functioning of the components in its faulty state. In exploitation phase, all those elements can be represented graphically in BDDs, Parametric Diagrams and Statemachine Diagrams.

The DBD was firstly constituted to support FMEA automatic synthesis, but is now a repository used to manage the reliability information necessary and created during the analysis. This DBD offers different constructs to express the knowledge about the dysfunctional behavior. These constructs are not necessarily all utilized, for instance if only qualitative information are available, the DBD will only contain a behavioral description with the named FMs. We want to mention that this DBD is flexible and modular to the respect the user specific needs and knowledge.

Concerning the organization of the DBD, the classification is classical and follows what is done in all similar databases. The components and their attached concepts are grouped in packages expressing domain applications. The SysML Package constructs help to perform and manage this classification easily. Moreover, the flexibility of packages permit the user to construct its proper organization or to express different views of the DBD, like classification by components belonging to a same system or by class of component.

### 4.3. Algorithms for automatic FMEA creation

As we presented in Section 4.1, FMEA can only be integrated efficiently in our approach if a tool corrects its heaviness. Nevertheless, FMEA seems unavoidable to initiate the approach. We created several algorithms supporting the FMEA from SysML.
models. This FMEA help us find all the primary failures that can occur in the system architecture. The objective of our tool is to make FMEA more rapid, more relevant, easier to construct and more effective in the whole approach. The algorithms developed focus on the following points:

- Exhaustive identification of the components.
- Automatic exploitation of lesson learnt through the DBD.
- Helping experts’ decisions:
  - Highlighting functional links between components,
  - Identifying the control and command flows on studied components,
  - Underlining impacts on the project requirements,
  - Proposing potential causes on effects.
- Automatic realization of the document which will support the FMEA.

The algorithm processes as a research of elements and links in the model. With the collected information it creates lists of elements that are ordered to construct the FMEA report. The algorithm uses the organization of information in a SysML model like it is defined in the SysML metamodel. This insure the reusability of the techniques on every models built with a modeling tool respecting this specification. The algorithms can be implemented on XML (extensible Markup Language) format, as SysML models can be written in the formalize XMI (XML Metadata Interchange) format that ensures the exchangeability of SysML models between software tools. The following are the notations utilized in the definition of our algorithms:

\[
\begin{align*}
\forall c : c \in \text{ModelIBDs} & \quad \text{ModelIBDs} \ni \text{IBD} \\
\forall c : c \in \text{ModelSDs} & \quad \text{ModelSDs} \ni \text{SD} \\
\forall c : c \in \text{ModelPDs} & \quad \text{ModelPDs} \ni \text{PD} \\
\forall c : c \in \text{ModelIBDs} & \quad \text{ModelIBDs} \ni IBD \\
\end{align*}
\]

Table 1 presents the inputs and outputs of the algorithms.

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>Outputs</td>
</tr>
<tr>
<td>ModelIBDs</td>
<td>ConstraintLinkedToPar Structure that contains all the constraints whose</td>
</tr>
<tr>
<td>ModelSDs</td>
<td>properties are linked to the studied component attributes. Each line indicates</td>
</tr>
<tr>
<td>ModelPDs</td>
<td>a constraint name and its linked attribute.</td>
</tr>
<tr>
<td>ModelBlocks</td>
<td>PartConnectedUpstream Structure whose lines contain the parts that transmit</td>
</tr>
<tr>
<td>SystemRequirements</td>
<td>PartConnectedDownstream Structure whose lines contain the parts that receive</td>
</tr>
<tr>
<td>ModelConstraints</td>
<td>PartFailureModes Structure that contains all the known FMs applicable to the</td>
</tr>
<tr>
<td></td>
<td>PartReceiverSD Structure whose lines present the parts that receive messages</td>
</tr>
<tr>
<td></td>
<td>PartSenderSD Structure whose lines present the parts that send messages to</td>
</tr>
<tr>
<td></td>
<td>ReqLinkedToBlock Structure whose lines present the requirements not directly</td>
</tr>
<tr>
<td></td>
<td>ReqLinkToPar Structure whose lines present the requirements satisfied by a</td>
</tr>
<tr>
<td></td>
<td>SystemComponents Structure that contains all component names.</td>
</tr>
</tbody>
</table>

Table 1 presents the inputs and outputs of the algorithms.

### 4.3.1. Identifying system components

Traditionally, FMEA are conducted after a good functional analysis. We claim that the SysML model we study is designed upon the result of such analysis. The first task performed during FMEA is to identify all the components that have to be analyzed. SysML constructs allow the modeler to describe the decomposition of his system at every level of detail. IBDs describe the organization of system components by expressing the role of blocks in the system being modeled. Therefore, all the components can be listed by registering each part met in the IBDs. The selection of IBDs to consider allows to limit the FMEA to a given level of decomposition in the system view. Moreover, it is possible to study only parts of the whole model by selecting the IBDs modeling the target subsystem. In previous works [16] we exploited UML SD to list the component that might be surveyed. This method is still applicable with SysML models since SysML SDs are based on UML SDs. But because SDs focus on commands exchanged, the study suffered of a lack of information concerning the material and energy flowing through the structure. We thus need diagrams that go further in the system modeling. The best solution is to utilize SysML IBDs that are defined in order to model the functional connections we search. Therefore, IBDs are the diagrams on which our study is primarily based. The search of components is performed by exploring all blocks of the model and their parts. The specification indicates that a unique block names each IBD, thus the search can be performed as in Algorithm 1.

The components to study are regrouped in the structure SystemComponents. The algorithms create structure of elements in relation with the studied part. These algorithms are utilized on each element of SystemComponents, in a way that a set of structures is attached to each element of SystemComponents. All those structures are then utilized to produce the preliminary FMEA report.

### 4.3.2. Identifying impacts on requirements

Since the aim of our approach is to steer design to a reliable and safe solution, it is crucial to follow the respect of requirements. That is why we decide to mention in the FMEA table the requirements affected when a component fails. Thanks to the SysML modeling facilities, the requirement satisfied by studied components are known, we can foresee the degradation of a function that has to be compliant with a specified requirement, which is due to precise component failure. Our tool automatically indicates to the user which requirements are affected by the
failure listed in FMEA. This information will be given in the ‘potential effects’ column of the FMEA table. Thus, the attention of the user is focused on how the requirements achievement could be threatened. He will have to evaluate if the failure causes a non-respect of the requirement or not, and determine the severity of the failure mode.

The algorithm explores the requirements linked to parts by using the relationships described in Section 2.1.3, it finds directly linked requirements but also identify upper level requirements. The structure of SysML models simplifies the research of impacted requirements thanks to the routing of elements properties. In fact the representation of requirements includes a property SatisfiedBy that contains the name of each components involved in requirement satisfaction. In the same way, it is possible to follow the chain of impacted requirements by considering the property DerivedFrom of the previous impacted requirements, these two relations are mentioned on Fig. 2. Considering a component that failed, we are thus able to track back the impacted requirements. The degree of derivation of the requirement is taken into account in the construction of the impacted requirements list. This list is built by Algorithm 2. The first line contains the requirements directly attached to the part. The second line contains the requirements derived from the first ones. The requirements of the last line are all top-level requirements. In a second time the algorithm builds a list of requirements not directly satisfied by the part, but by the block that owns it. The search is restricted to unmentioned requirements, but derivations are considered.

The algorithms used to build the other structures follow similar reasoning and will not be detailed.

### 4.3.3. Identifying components relationships

We assume that failure effects dramatically depend on the propagation of the primary impact on the faulty components, through the functional links existing with other components. Therefore in an analysis like FMEA, which proposes to explore the effects of failure at various levels of detail, it is crucial to know the connections that exist between the studied component and the remaining of the system. The metamodel of model element connection is given in Fig. 5. We can read on it that the search of elements’ neighborhood to another can be achieved by noting the properties of the association that link them.

Observing connectors is crucial to identify failure effects, since propagation of errors is conditioned and achieved by them. The information provided by the study of connectors enriches the columns ‘potential causes’ and ‘potential effects’ of FMEA table. The study is realized as follows: for each studied parts (components), ports are analyzed. The ports are directly accessible as they are properties owned by the parts and inherited from the blocks that type them. For each port, the connectors that link them are identified. This is performed by finding the connectors whose end role is the considered port (see Fig. 5). Then, other ports linked by the connector are tracked as well as the parts that own them. The direction property of ports, that expresses the direction of the flow (in, out or inout), permits to decide if the found parts are potential cause or potentially impacted components. Presence of item flows on connectors gives supplementary information on the nature of the passing flow. This allows more complex reasoning as the observation of dysfunctional states in which the considered item flows can be.

The algorithm utilized examines the connectors linked to parts in IBDs in order to find the affected entities in case of failure modes propagation through the structure. The algorithm treats the flow ports and standard ports, since the two types of port exists in SysML.

In order to complete the analysis of connections, it is valuable to examine the available SDs. In those diagrams, we exploit messages exchanged between parts. These messages carry operations that parts perform on each other. In the same manner as for connectors, the message sender and receiver are identified, so that the parts in relation with the studied one and the operations existing between them are added on the columns ‘potential causes’ or ‘potential effects’ of FMEA table.

#### 4.3.4. Identifying impacts on system properties and performance

The use of constraint blocks was presented in Section 2.1.3, it was shown that these blocks indicate important information as performance indicators computation or physical relationships between components properties. Therefore, it is valuable to mention in the FMEA which constraint can be affected by a given failure mode on a component. The algorithm constructs the structure of impacted constraints. By extension, if the analyst knows the variation domain of component parameters in the failure states, then the Parametric Diagrams indicated will allow to compute the resulting performances of failure states. The algorithm considers each parameter of the studied part and searches if it is linked to a constraint block in a Parametric Diagram of the model. If it is the case the names of the constraint and the parameter are registered.

#### 4.3.5. Collecting failure modes in the DBD

The automatic exploitation of the knowledge stored in the DBD is achieved by identifying the type of the studied part. The type is referenced in the DBD, the FM names are then collected since FM are linked to the component type in the DBD. This is a first use of the DBD but not the last. This step helps to construct the first version of the FMEA. Then the user will have to consider the other diagrams associated in the DBD to evaluate precisely the other criterions of the FMEA table.

#### 4.3.5.1. Component types not registered in the DBD

If the component type has not been already encountered it is important to invite the analysts to fully study them. Therefore, the structure PartFailureModes is not left empty but contains the basic declaration of FMs (No function, loss of function, unexpected function, degraded function). This case triggers an update of the DBD. The software invites the user at creating this new type and at filling its section with the result of the current analysis.

#### 4.3.6. Constructing the FMEA table

The preceding algorithms automatically identified various valuable information for FMEA construction. Components names and FM names will directly be written on the final report. The other data as linked components and impacted requirements are indicated on a preliminary FMEA report. They have to be studied by the FMEA redaction committee. Different policies in the creation of the preliminary FMEA report can be applied. This operation is quite simple as it is just a formatting process of the

<table>
<thead>
<tr>
<th>Algorithm 1—Identifying system components</th>
</tr>
</thead>
<tbody>
<tr>
<td>∀ ibd ∈ ModelIBDs</td>
</tr>
<tr>
<td>∀ Par</td>
</tr>
<tr>
<td>SystemComponents[ ] ← SystemComponents[ ] • Par.name</td>
</tr>
</tbody>
</table>

---

P. David et al. / Reliability Engineering and System Safety 95 (2010) 431–450

441
information embedded in the structures. Table 2 shows an example of what can be done.

The aim of those algorithms is to furnish a help to FMEA study. The software tool implementing them provides the automated realization of the preliminary tasks of FMEA and selects the appropriate information to efficiently judge each risk. We observed that our solution saves a lot of time in FMEA process. Namely, listing component and their known FM is a basic task usually time consuming, which is performed in a few seconds. Therefore, the analysts will have more time to concentrate on the critical phases of the study. The second improvement is to automatically highlight the influential point for the risk evaluation. The analyst judgment is focused on the important parts of the model and avoids to get lost on uninteresting parts of the

```
Algorithm 2–Identifying impacted requirements.
∀ Par ∈ SystemComponents

∀ Req1 ∈ SystemRequirements

if (Par = Req1.SatisfiedBy)

Req1.linkedToPart[I, :) ← Req1.linkedToPart[I, :) • Req1.name

i ← 1

While (∃ Req1.1 ∈ Req1.linkedToPart[I, :) | Req1.1.DerivedFrom ≠ NULL)

∀ Req1.2 ∈ Req1.linkedToPart[I, :) | Req1.2.DerivedFrom ≠ NULL

∀ Req2 ∈ SystemRequirements\{Req3 | Req3.name ∈ Req1.linkedToPart\}

if Req2 = Req1.2.DerivedFrom

Req1.linkedToPart[I+1, :) ← Req1.linkedToPart[I+1, :) • Req2.name

i ← i + 1

for bl | Par = bl.part

∀ Req4 ∈ SystemRequirements\{Req6 | Req6.name ∈ Req1.linkedToPart\}

if (bl = Req4.SatisfiedBy)

Req1.linkedToBlock[I, :) ← Req1.linkedToBlock[I, :) • Req4.name

j ← 1

While (∃ Req1.3 ∈ Req1.linkedToBlock[J, :) | Req1.3.DerivedFrom ≠ NULL)

∀ Req1.4 ∈ Req1.linkedToBlock[J, :) | Req1.4.DerivedFrom ≠ NULL

∀ Req5 ∈ SystemRequirements\{Req6 | Req6.name ∈ Req1.linkedToPart ∪ Req1.linkedToBlock\}

if Req5 = Req1.4.DerivedFrom

Req1.linkedToBlock[J+1, :) ← Req1.linkedToBlock[J+1, :) • Req5.name

j ← j + 1
```
model. The extracted and highlighted parts of models that are automatically provided to the analyst allow him to master the complexity of the model, and to examine for each component where negative impacts can appear and how evaluating them. This is a true improvement in FMEA study that can be more quickly and safely conducted. To reflect our experience with these algorithms we will evoke their utilization on a case study in Section 6. Finally, this format of analysis provides a data structure that supports the process of maintaining the experience gathered by the studies.

5. Constructing the dysfunctional model in formal languages from SysML models and FMEA results

As mentioned in the presentation of the global approach for reliability study, the use of formal models is mandatory to compute relevant reliability indicators and qualitative information. Effective tools and languages already exist to write and analyze those models. Therefore, we mainly want to develop techniques to set up those models from others written in SysML. In this section, we will evoke the parallel between SysML and the language AltaRica. We will show how the necessary information to build models of the system in this language can be found in SysML models. Then we will present how the DBD and FMEA study can help the construction of the whole model showing the dysfunctional behavior among the functional activities of the system.

5.1. Towards deriving an AltaRica data flow model from SysML

AltaRica data flow is a restriction of the AltaRica language well adapted to derive fault trees and Boolean formulae. Mode automata are the underlying mathematical concepts of this language [28]. It is supported by the BPA DAS workshop to conduct reliability studies. This software provides a graphical interface to construct model in AltaRica data flow and integrates tools to perform the computation of fault trees for critical events. An AltaRica data flow model is an automaton that generates a graph of reachable states from initial states, called Kripke structure [28]. This section underlines how the element of a Kripke structure can be found in SysML.

As exposed in [31], the elements needed in a model for reliability study are the following:

- The failure and functional states of the components that make them behave in a determined way.
- The events provoking a state change as failures or changes of functioning modes.
- The functions performed in the different states.

The functional model described in SysML is exploitable to identify the functional aspects expected to set up the model for reliability analysis. Reusing the notation of [31] we propose to identify the element of a Kripke structure that are expressed in a classic SysML model. A Kripke structure is defined as follows (W, W0, Var, Dom(Var), Evt, m, t):

- W set of reachable states and W0 subset of W constituted of the possible initial states,
- Var finite state of the variables either denoting the flows or the internal states of components,
- Dom (Var) set of reachable values by the variables,
- Evt finite set of events,
- m: W × Var → Dom(Var) the function that determines the value of variables in a given state,
- t: W × Evt → W transitions between states fired by the events.

We detailed the set Var in two complementary sets Var/O and VarState. Var/O ∪ VarState=Var and Var/O ∩ VarState=0. Var/O is the set of flow variables, VarState the set of component states...
variables. Table 2 gives the SysML modeling constructs or diagrams that can be used to build the Kripke structure elements.

The construction of the formal model for reliability study is performed in two steps. First, the functional part is built from designers’ model, then after the first risk analysis (e.g. FMEA) we complete this model by its dysfunctional parts. The functional part of the reliability study model is derivable from the designer’s model in SysML. Since languages as AltaRica data flow focus on states and flows representation, a parallel can be found with the SysML models. Elements are picked up in various constructs and diagrams of the SysML model. For the states identification, the best structure to consider is the statemachine described in a StateChart diagram. The various statemachines of the model represent the behavior of components as a state history. The statemachine is a formalism very close to the concepts used in AltaRica. [28] mentions that Mode automata are strongly related to visual specification languages as StateCharts. Therefore, the statemachine reveals a major part of the future AltaRica model. W contains the possible combination of states defined in the statemachines, W0 refers to the initial state also defined in state machines. Elements of VarState are the states defined in the state machines and their Dom is expressed in the statemachines. These VarState elements are the modes of the components, referring to the functioning modes of components. Another type of VarState elements is defined in the blocks, those variables are the properties of the component that have an influence on the treatment of the flows received by the component. These variables have generally continuous domains defined by their type. They are accessible in other blocks definition, generally regrouped in a type definition package of the SysML model. The set of events is reflected in the messages of the SysML model, the messages from external components are seen as events by the called components. Moreover, events are also declared in statemachines and the notion is presented in the transitions that form the t function. The m function can be determined by the action performed in a given state of a statemachine and by the constraint defined in Parametric Diagrams that mainly expresses the computation of the second type of VarState elements. Concerning the Varl/O elements, these are clearly defined through ports declaration in the blocks and parts definition. Their domains depend on their type defined by specific blocks, but they can sometimes be indicated in requirements on their accepted value range. Concerning Activity Diagrams, they emphasize the inputs, outputs, sequences and conditions for coordinating other behaviors. Therefore, they are used to identify states, VarState elements, their domains as well as t functions and the set of events. Nevertheless, we can notice that Activity Diagrams are not the best structured design constructs of SysML and that the phenomenon they modeled are already widely covered by Sequence Diagrams and statemachines. Therefore, we prefer to focus on the use of these modeling constructs.

It is primordial to notice that all this modeling constructs are used to determine only the functional parts of the model for reliability evaluation. Other information is necessary to obtain the final model including the dysfunctional behavior. The DBD and final FMEA report bring those last constituents. The missing parts are the failure mode states, the behavior of components in those states and the effects of those states on the variables. The goal of FMEA study is to determine these elements. The FMs identified in the FMEA are the new states reachable by the components. The description of their effects gives the behavior of the component in the FM. The causes and effects column depicts also the events that cause the FM and the events generated by the FM. The mechanisms described also express the t function and new aspects of m function associating failure state with new values of the various variables. The data presented in the FMEA can either be directly used to construct the final reliability study model or/and be injected in the DBD in an updating process. We advise the second utilization since this process will lead to an even better DBD, exploitable for future studies. The DBD written in SysML is structured in order to host this kind of data. Table 2 showed that it was possible to represent those aspects in SysML and the DBD metamodel of Fig. 4 described how this knowledge is organized. Namely, each FM state is described by one named FM that is attached to the component type in the DBD. The behavior, and therefore intermediary state composing the failure state, has to be declared in a statemachine. This proposition is concordant to Table 2, as well as the fact to declare the FM profile with constraint blocks. All that concerns events and transitions of the dysfunctional behavior is indicated through the statemachine of the FM. The computation of the domain of variables into a faulty state is also indicated in the statemachine.

6. Case study: the level control system

This case study introduces the use of SysML for the design of safety critical systems and illustrates the use of our approach for its reliability study. It presents the declaration of safety requirements, the use of SysML to describe the system architecture and behavior, as well as the use of our FMEA creation algorithms.

The Level Control System (LCS) is mounted on a tank that contains an unspecified fluid. The LCS has to insure that the fluid level will never exceed the tank capacity. The LCS is composed of two electrovalves (Ev1, Ev2) connected upstream the tank, two mechanical valves (Mv1, Mv2) one connected upstream and the second downstream for draining and one alarm (Al). Two level sensors (S1, S2) command the electrovalves and the alarm. If S1 detects a too high level of fluid in the tank, it closes Ev1. If S2 detects a too high level, it closes Ev2 and activates Al. When Al rings, an operator closes Mv1. If the fluid level is still too high after 3 min, the operator opens Mv2 that drains the tank. This case study is representative of our problematic since it constitutes a safety critical system that embeds physical parts (tank, valve, fluid dynamic), control flow (command of valve, sensor management) and human intervention (the operator).

6.1. SysML specification of the LCS

In this section, the SysML model of the LCS development project is presented. The various SysML modeling possibilities are used to fix many aspects of the project as requirement declaration and allocation, structure definition and command flow during nominal scenarios.

6.1.1. Requirement specification

Requirements management is one of the main improvements of SysML compared to UML. Their definition and allocation for the LCS is clear and permits to ensure their traceability, which is crucial for the project success. On Fig. 6, reader can discover the organization of the basic requirements and their allocation to the components of the system. The requirements are presented on a Requirement Diagram edited with the commercial tool ARTISAN Studio from ARTISAN Software®. The requirements developed are of two kinds: the Level Control (and derived requirements) covers functional aspects, the Availability Requirement exposes a requirement of reliability on demand. These definitions can be quantified by availability rates or accepted failure rates.
6.1.2. Structure specification

The structure is declared through two types of diagrams: the BDD declares an abstract view of the LCS, whereas the IDB shows the functional links between components. The BDD of Fig. 7 presents the general view of the LCS through the whole process. The different parts that compose the LCS are also mentioned in this diagram.

The physical connections of the LCS components are presented in Fig. 8 through an IDB. The main interest of this representation is to show the components connected in their context. The data and physical channels are represented. Moreover, the port typing allows the identification of what flows into these various channels. This information might be very valuable for reliability analysis. In fact, when flows are known, it is possible to consider their variation domain and the critical state that the flow can reach. Namely, we expect to extend our DBD to the declaration of flow “dysfunctional” behavior in the sense of registering the critical state of flows encountered in the systems designed. The presented IDB shows the circulation of the fluids through the LCS as well as the command channels between sensors and valves. This IDB also represents the interaction devices used by the operator.

6.2. FMEA study of the LCS

The study of the LCS is performed using the approach presented in Section 3. At the beginning of the study a database concerning the exploitation of valves and electrovalves was considered. The algorithms presented in Section 4.3 are employed. The Table 3 contains the structures constructed for the component S1.

These structures concerning S1 are built for all the other parts identified (system components). Then, the information contained in the structures is presented in order to produce the preliminary FMEA report. Table 4 presents the extract of this report concerning the component S1 and the beginning of the section on Ev1. This table is also an example of how can the data of the structures be organized. For instance, in the “Impacted Requirements” column, an indication is given on the degree of derivation and origin of the requirement. The prefixes “p” and “bl” indicate if the requirement is linked to the part or to the overall block. The number indicates the derivation of the requirement. In the “Possible Causes” and “Possible Effects” columns, internal phenomenons are mentioned in order to make the analysts consider them. In fact, the previous algorithms focus on interaction and do not consider internal structure. To perform the FMEA study of a component, users just have to use the algorithms on the IBD describing the target component as it is done here to study the LCS that is a component of a bigger system. This preliminary report is integrally automatically constructed. It constitutes a significant help for the realization of the final report and is a guideline for the group of analysts. This example underlines the fact that the analysis of both SD and IBD is valuable. Each diagram brings complementary information. If we observe the “possible causes” column we observe for component S1 two proposals, respectively, made from the IBD analysis and the SD study. The first one points out a risk from Ps (the power supply) through the power input. This leads to express that S1...
could suffer of a power alimentation rupture. The second proposition is the risk of failure during the execution of the “activation” message. This leads to interpret that the FM can be caused by a non-activation of S1. This can be translated in the final report by “no sensibility of S1” or “fail to interpret signal from sensible cell”. The fact that is important to note is that no unique diagram was able to show those two relations. Nevertheless, the exploitation of both type of diagrams makes it possible to explore a large spectrum of risks, causes and effects. Moreover, the indications given in the automatically constructed report demonstrate their efficiency in helping to build the FMEA and dysfunctional ones expressed in SysML. All these techniques underlined the constitution of this kind of model from functional and dysfunctional behavior. In this kind of node, probability laws can express the occurrence of the events. Each event models a FM and provokes the component to be in a specific state. This AltaRica representation of the FM has its correspondence in the DBD written in SysML. The metamodel of Fig. 4 shows that the failure law is attached to the FM as well as Parametric Diagrams expressing the computation of the failure rates. Moreover, the statemachine associated to the FM indicates the effects of the event occurrence and that this process was automatizable in order to enhance the efficiency of the study. Functional and dysfunctional aspects are added considering the conclusions of the FMEA study. For each component, the AltaRica node is constructed. The nodes are connected and synchronized to obtain the analyzable model. From this model the search of failure scenarios and their quantification is possible using commercial tools exploiting this language (BPA DAS, FIGSEQ from Dassault Systems and eDF R&D).

An AltaRica node consists in declaring the different structures and function of the components and their relations. The flows are deducted and named from the IBD of Fig. 8. In this diagram the name, type and direction of the ports are considered. The states are deducted from the functional state-machine (written in the SysML model) of the component and from its FMs. The initial state is found in the statemachine. The events are also found in the statemachine and events are created for the element of the ‘cause’ column of the FMEA. The transitions and assertions are deducted from the statemachine and the Parametric Diagrams describing S1. Additionally the ‘effect’ column of the FMEA helps building the assertion section for the dysfunctional behavior.

The next step in the FMEA process is to study the various points set out in the preliminary report and to use the available models to evaluate them. The real causes and effects are deducted by analyses on diagrams of the DBD and of the functional model. From Table 4 the part of the final FMEA report presented on Table 5 has been created.

The main improvements noted during this study were that we saved a lot of time and were ensured that we studied precisely the system. Moreover, the bigger the studied system will be the more justified will this technique be. This work is a good breakthrough to render the FMEA more operational and utilizable during design process. Another advantage of this technique is that it is easily integrable in the current commercial tools. The algorithm respects the SysML metamodel and does not need additional stereotyping except for the construction of the DBD (not mandatory to list the component and their relations.

### 6.3. Synthesis of AltaRica nodes

As mentioned in Section 5, the study is continued with the creation of the AltaRica Model for final analysis. The AltaRica nodes are created in two phases. First the functional part is constructed from the study of SysML models. Then, the dysfunctional aspects are added considering the conclusions of the FMEA study. For each component, the AltaRica node is constructed. The nodes are connected and synchronized to obtain the analyzable model. From this model the search of failure scenarios and their quantification is possible using commercial tools exploiting this language (BPA DAS, FIGSEQ from Dassault Systems and eDF R&D).

An AltaRica node consists in declaring the different structures and function of the components and their relations. The flows are deducted and named from the IBD of Fig. 8. In this diagram the name, type and direction of the ports are considered. The states are deducted from the functional state-machine (written in the SysML model) of the component and from its FMs. The initial state is found in the statemachine. The events are also found in the statemachine and events are created for the element of the ‘cause’ column of the FMEA. The transitions and assertions are deducted from the statemachine and the Parametric Diagrams describing S1. Additionally the ‘effect’ column of the FMEA helps building the assertion section for the dysfunctional behavior.

The next step in the FMEA process is to study the various points set out in the preliminary report and to use the available models to evaluate them. The real causes and effects are deducted by analyses on diagrams of the DBD and of the functional model. From Table 4 the part of the final FMEA report presented on Table 5 has been created.

The main improvements noted during this study were that we saved a lot of time and were ensured that we studied precisely the system. Moreover, the bigger the studied system will be the more justified will this technique be. This work is a good breakthrough to render the FMEA more operational and utilizable during design process. Another advantage of this technique is that it is easily integrable in the current commercial tools. The algorithm respects the SysML metamodel and does not need additional stereotyping except for the construction of the DBD (not mandatory to list the component and their relations.

### 7. Conclusion and future works

In this paper we illustrate that SysML is a good modeling language to support the design process of safety critical systems. SysML can be utilized as a central formalism for each activity needing specialist tools or modeling constructs. We especially raise the compliance of SysML for reliability and risk studies. We underlined how SysML models were analyzable for FMEA studies and that this process was automatizable in order to enhance the efficiency of the study. Functional and dysfunctional behavior can be described with this language allowing the construction of models well suited for classic risk analysis. Therefore SysML is not presented here to create a new way for reliability and risk analysis, but to underline that current efficient method can be supported by this new language. We presented a way to organize a DBD that collects and maintains the experience of previous studies on similar systems. Finally, we detailed a connection to an efficient modeling language for reliability study: AltaRica. We underlined the constitution of this kind of model from functional and dysfunctional ones expressed in SysML. All these techniques have been exemplified on a case study.

The process described here can be seen as a part of a SysML-centered Model-Based System Engineering process, where safety and dependability relevant activities are merged to the whole system development cycle. The various connections, between a SysML model and the dependability analysis, described in this article were built to sketch a method accompanying the system.

---

**Fig. 7. BDD of the LCS environment.**
design from the seeking of its components failure modes, to its overall behavior validation. Moreover, we also proposed a mean to efficiently register the knowledge raised by this process by defining a DBD structuration. Those results were obtained by the willingness to provide a general contribution applicable by any SysML user wanting to initiate safety and dependability validations. This work contributes to underline the benefits of Model-Based approach in system development. Indeed, we have shown that using such techniques provides a way to enhance consistency between the designer proposition, the results of FMEA and the
formal model dedicated to dysfunctional behavior analysis. Moreover, we can also observe that this type of description allows to automate some parts of the reasoning and thus reducing the workload of the dependability experts. This method is on use in our laboratory initiatives for example during the LEA project, aiming at developing a dual-mode ramjet powered vehicle. Complementary works are now needed to set up, maintain and impose FM and component taxonomies, in order to create huge database supporting new systems creation. Connections to other analysis formalisms have also to be developed to tackle specific class system, as hybrid ones, as well as to adapt to a large panel of reliability analysis techniques (e.g. Petri Nets, Markov processes, Dynamic Fault Trees). Finally, we foresee to work on SysML.

Table 3
Structures generated for part S1: Sensor.

<table>
<thead>
<tr>
<th>Structure name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConstraintLinkedToPart</td>
<td>Ø</td>
</tr>
<tr>
<td>PartFailureModes</td>
<td>No Detection, False Detection</td>
</tr>
<tr>
<td>ReqLinkedToPart</td>
<td>DetectingFluidLevel</td>
</tr>
<tr>
<td>ReqLinkedToBlock</td>
<td>LevelControl</td>
</tr>
<tr>
<td>PartConnectedUpstream</td>
<td>Ps, PpPs, P51, PowerInput</td>
</tr>
<tr>
<td>PartConnectedDownstream</td>
<td>Ev1, C51, C5Ev1, CommandInterface</td>
</tr>
<tr>
<td>PartReceiverSD</td>
<td>S1, Activation, Ev1, CommandEv</td>
</tr>
<tr>
<td>PartSenderSD</td>
<td>S1, Activation</td>
</tr>
</tbody>
</table>

Table 4
Excerpt of the preliminary FMEA.

<table>
<thead>
<tr>
<th>Component</th>
<th>Failure mode</th>
<th>Affected constraints</th>
<th>Impacted requirements</th>
<th>Possible causes</th>
<th>Possible effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>No detection</td>
<td>Ø</td>
<td>DetectingFluidLevel (p 1)</td>
<td>Internal cause From Ps through PopS-P51 [PowerInput]</td>
<td>Internal effect On Ev1 through C51-C5Ev1 [CommandInterface]</td>
</tr>
<tr>
<td></td>
<td>False detection</td>
<td>Ø</td>
<td>LevelControl (p 2)</td>
<td></td>
<td>On Ev1 by CommandEv</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AvailabilityRequirement (bl 1)</td>
<td>Internal Cause From S1 by Activation</td>
<td>On S1 by Activation</td>
</tr>
<tr>
<td>Ev1</td>
<td>Stuck opened</td>
<td>FluidFlowEv1</td>
<td>StoppingTankAlimentation (p 1)</td>
<td>Internal cause From S1 through C51-C5Ev1 [CommandInterface]</td>
<td>Internal effect On LCS through InEv1-InLCS [Fluid]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LevelControl (p 2)</td>
<td>From Ps through PopS-P51 [PowerInput]</td>
<td>On Ev2 through OutEv1-InEv2 [Fluid]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AvailabilityRequirement (bl 1)</td>
<td>From LCS through InLCS-InEv1 [Fluid]</td>
<td>On Ev1 by InvertState</td>
</tr>
</tbody>
</table>

Table 5
Excerpt of final FMEA report.

<table>
<thead>
<tr>
<th>Component</th>
<th>Failure Modes</th>
<th>Causes</th>
<th>Effect on Component Behavior</th>
<th>Effect on System</th>
<th>Effect on Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>No detection</td>
<td>Loss of electrical power</td>
<td>Command through C51 stuck to 0</td>
<td>No detection when the tank is full at 90%. Ev1 stay opened. The function of the LCS is still respected by the chain commanded by S2.</td>
<td>The Failure of S1 affects the availability requirement of the LCS.</td>
</tr>
<tr>
<td></td>
<td>False detection</td>
<td>Loss of electrical power</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ev1</td>
<td>Stuck opened</td>
<td>Mechanically stuck</td>
<td>Always let the fluid passing from InLCS to InEv2.</td>
<td>Ev1 closes the alimination of the tank. Possibility to see the Tank empty. No fluid for the Distribution network.</td>
<td>The Failure of Ev1 affects the availability requirement of the LCS.</td>
</tr>
</tbody>
</table>
models construction method that will permit to enhance the efficiency of the tools proposed in this paper.

Acknowledgements

We specially want to thank all our partners involved in the CAPTHOM project. This work was done with the financial help of the French Industry Ministry and local collectivities, within the framework of the CAPTHOM project of the Competitiveness Pole S²E², www.s2e2.fr.

References


Author's personal copy


