

Mastering Safety and Reliability in a Model Based Process

Robin Cressent, Vincent Idasiak, Frédéric Kratz, Pierre David

▶ **To cite this version:**

Robin Cressent, Vincent Idasiak, Frédéric Kratz, Pierre David. Mastering Safety and Reliability in a Model Based Process. 2011 Proceedings - Annual Reliability and Maintainability Symposium, Jan 2011, Lake Buena Vista, FL, United States. 6 p., 10.1109/RAMS.2011.5754506 . hal-00630827

HAL Id: hal-00630827
https://hal.science/hal-00630827

Submitted on 11 Oct 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Mastering Safety and Reliability in a Model Based Process

Robin Cressent, Vincent Idasiak & Frederic Kratz
Institut PRISME, UPRES 4229, ENSI de Bourges, Bourges, France.

Pierre David
Heudiasyc, UMR CNRS 6599, Université de Technologie de Compiègne, Compiègne, France

Key Words: Reliable system engineering, SysML, AADL, Simulink

SUMMARY

This article follows a line of papers focused on defining a method to improve the realization of reliability analysis during the System Engineering process. As MBSE becomes a fundamental concept for specifying and designing systems, our method takes full advantages of this approach and try to provide tools to ease the specification stage and the integration of RAMS early in the conception process. Our method called MeDISIS is related to the use of SysML to support MBSE and RAMS activities.
Currently, MeDISIS is used within an industrial project to design a hypersonic aircraft which is a relevant complex and critical system. During this project, MeDISIS has been adapted to take into account technologies devoted to embedded systems. Furthermore, MeDISIS had to comply with the tools, used by our industrial partners during the design stage. In this work, we present the new architecture of MeDISIS, and the process added recently.

1 INTRODUCTION

Nowadays, the Model Based System Engineering (MBSE) paradigm is becoming the predominant concept used for System Engineering (SE) (1). The main idea brought by this practice, is to enhance the design process of complex systems by making them more reliable and by organizing development process activities through formalized system representations. The Model Based representations enable to obtain more consistent, traceable, coherent, reusable and expressive views of the system to be developed, thus helping the management and realization of its design process. However, RAMS activities are generally forgotten in this engineering field.

Our contributions to MBSE focus on defining a method to improve the realization of reliability analysis during the SE process and its early design phases. This method introduced in several publications (2)(3)(4), is called MeDISIS and is related to the use of SysML (5). We assume that input models are expressed in SysML and we intend to build a repository that registers and manages the knowledge raised by the performed activities, in a structure modeled in this language.

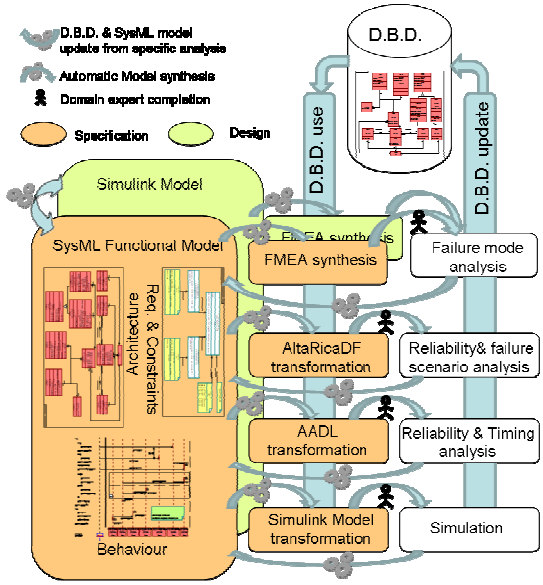


Figure 1. MeDISIS processes overview

MeDISIS proposes a deductive and iterative approach that aims at facilitating crucial reliability analysis and enhancing the use of the diverse tools and languages used for dysfunctional behavior validation. MeDISIS includes the following process:

- Deduction of the dysfunctional behaviour with an FMEA, identification of the impacted requirements.
- Construction of a model integrating functional and dysfunctional behaviours with a formal language such as Altarica DataFlow.
- Analysis and quantification of dysfunctional behaviour and the impact on requirements and timing constraints with a semi formal language such as AADL.

Those first two steps are described more precisely in (4).

Our current work consists in the application of these methods to the specification and the conception of the embedded controller of an aircraft system (LEA project). We add to the specification stage of an embedded system a specific tool based on AADL to achieve the temporal study of the specified system and highlight possible impacts on dependability at the same time. In addition we will highlight several best practices that should be merged with the design activities. By following some rules and using some SysML artifacts (parametric diagram, BDD, item flow), it is possible to collect and classify the parameters which influence reliability, such as the system life cycle, the real mission profile of the system, the use conditions and overstresses. For each identified class of parameters, we define the flow sets that will be used to create test vectors and perform reliability studies.

After a first functional analysis, we use the FMEA generator of MeDISIS. At this stage, each failure mode of each component is capitalized in a dysfunctional model repository and will be reused later to build a formal or semi-formal representation of the system and its dysfunctional behaviour.

At the design stage, we focus on introducing failure modes in the system's model thanks to SysML parametric diagrams and IBDs translated in Simulink blocks.

After introducing the synchronization process between SysML and Matlab, we present how performing failure mode propagation studies, using Simulink blocks generated from selected parametric diagrams, or IBDs identified in the FMEA.

The proposed procedure is to select each impacted couple (requirement and flow set) from our FMEA tool, based on SysML. The SysML model analysis allows us to catch all the impacted components or functions of the model. After their translation into Simulink, we obtain computable models to carry out reliability or safety studies.

In this paper, we present the processes used for the first design step of our LEA project.

To perform the processes of MeDISIS several tools and analysis routines have been defined to support each phase and optimize the speed and quality of the reliability studies. These developments will permit to construct a complete System Development Environment (SDE) supporting system design and MeDISIS. The results of this study produce the dysfunctional behavior of each component that will be stored in the DBD (Dysfunctional Behaviour Database) to be used later to build the Altarica DF model for example. They are capitalized in a dysfunctional models repository and reused to construct a formal representation of the system using the AltaRica Data Flow (6) language. The construction of this formal model, mandatory for system validation, is also helped by analysis techniques systematizing the creation of this reliability-oriented view. A service to support embedded systems analysis has also been defined recently. It proposes to generate AADL (7) models exploitable for real time application studies using a scheduling tool named Cheddar (8). The FMEA automatic synthesis has been explained in details in (3) and (4). The first part of this paper will present how to perform reliability analysis using Altarica DF, the second part will precise how to make a timing analysis AADL, and finally we will describe a new way to extend MeDISIS to be used through the design stage where safety and reliability issues will be take into account.

2 CONNECTION TO FORMAL DESCRIPTIONS WITH ALTARICA DF

The second support needed in MeDISIS, is the integration of formal means of validation and quantification of the dysfunctional behavior. Many solutions to perform this task are available on the market. Therefore we concentrated on creating bridges between the tools used by functional engineers and those dedicated to reliability studies. We focused on using the AltaRica DF language, which is widely used among reliability engineers and which efficiently equips with solutions such as BPA-DAS (Dassault Systems product) (2)(3)(4).

The service is performed in two major steps, which are the translation of the SysML model to obtain the AltaRica DF description of the functional view of the system, and the modeling of the dysfunctional view using the Dysfunctional Behavior Database (DBD) built with FMEA results and previous studies. The first translation is important in order to construct a reliability study dedicated model consistent with the description of the system that is common to the whole development lifecycle. As SysML and AltaRica DF share an Object-Oriented approach, many elements are easy to translate. Nevertheless, some divergent declaration philosophies, such as the treatment of state and flow variables, impose to use more complicated translation rules. Moreover, the complete automation of the translation is possible only if the semi formal nature of the SysML description is constrained by the construction rules of the SysML model like the utilization of expressive allocations between the modeling elements.

The completion of the functional view by the description of the dysfunctional behavior of the components permit to point out the benefit of the MeDISIS framework and its DBD that centralizes the relevant information for reliability studies. In fact, the data raised by FMEA are added to the AltaRica DF model, thanks to its expression in the DBD. The complete model for formal reliability analysis is thus obtained and then exploited with the market software tools. The meta-model of the DBD has been developed in order to be coherent with the SysML description and to store the needed elements for the construction of the AltaRica DF final model. Therefore the DBD is built in SysML and integrates state machines diagram to prepare dysfunctional models creation.

MeDISIS has been designed as an evolutionary framework aiming at connecting all the needed specialized analysis tools, to assess all system behavior dimensions. It has been augmented with a service for real time constraints considerations exposed in the next paragraphs.

3 SUPPORT TO THE EMBEDDED DESIGN PROCESS USING AADL

AADL is a formal and textual language that appeared for the first time in 2004. Its graphical form and other extensions

were added in 2006. The recent revision (7) shows the interest of the community in keeping the language up-to-date. The use of AADL gives the opportunity to formally analyze real-time and embedded systems. To reach this objective, the use of a transformation of SysML models into AADL ones is an efficient support. Furthermore, some tools dedicated to AADL exist such as Cheddar (8), which permits to study the scheduling, processor usage, and respect of temporal constraints.

The aim of the translation is to automatically reuse the knowledge contained in the SysML model, to perform the real time behavior analysis. However, certain pieces of information such as the temporal properties of the system are often absent from the SysML model. In fact, SysML is usually used for high-level design that does not contain much temporal information. Nevertheless, we can help reuse information contained in the preliminary conception SysML model and ease its completion with the missing pieces of information in order to enhance the analysis in terms of speed and consistency. In this perspective, we have identified the possible links that could be made between the two languages.

The object-oriented approach of both languages allows an efficient translation of architectural concepts. Nevertheless, since AADL is a lower level representation, it uses more specific types of components. To classify the components according to the 10 categories (Memory, processor …) available in AADL; we have to consider another source of information to perform the model translation. The usable techniques are listed below:

- Imposing a methodology to model the system in SysML differentiating the various AADL stereotypes.
- Asking a specialist to classify each component. For example, using a questionnaire can be a way.
- Using a database of correspondences between SysML blocks and their category in AADL, based on the recorded past projects.

A similar problem is found to define all the properties that size the system, representing the quantitative properties needed to use tools properly such as Cheddar (8) or RMA (11). To completely define the *properties* of our system, we suggest the development of one of the three solutions presented before.

To manage those problems, we use a similar method to the one used to create FMEA and AltaRica DF models: using specialist judgment to complete our model, and maintain a database of feedbacks for future projects. The steps used to create the AADL model where described in (12) and are summarized below:

Step n°1. Identifying all the SysML blocks and parts and establishing the hierarchy between all those entities, taking the different levels of design into consideration.

Step n°2. Mapping every component with each other using ports and connections.

Step n°3. Categorizing each component of the system. (e.g.: this « shared memory » block belongs to the *memory* category).

Step n°4. Creating the structural model in AADL (textual and graphical models can be made at this point).

Step n°5. Filling in the properties that are not deducted from the SysML model.

Step n°6. Creating the final AADL model, which includes the structure description and the system properties.

It is visible that steps 1,2,4,6 can be instantaneous with proper software, but even with the database, steps 3 and 5 require a specialist, because some information may not have been recorded in the database yet.

Concepts	**AADL**	**SysML**
Software component /Implementation	Software component /Implementation	Block Part
Hardware component /Implementation	Hardware component /Implementation	Block Part
Bindings	Bindings	Block Bindings
Subcomponents	Subcomponents	Part
Connectors Flow	Port Connections Event, Data, Data-Event In, Out, Inout	Flow ports Value type / Block Flow Port Direction / Interface
States	Modes	State Diagram/state
Properties	Properties	Requirement Diagram, Parametric Diagram

Figure 2. Concept correspondence between AADL and SysML

The table from figure 2 highlights the correspondence between the main concepts of both languages, SysML and AADL. This table is a basic translation table that leads us to steps 1, 2 and 4.

Using those steps with our DBD, we can easily obtain an AADL DBD since dysfunctional model only contain SysML artifact used also in the functional model. The need to model dysfunctional behaviour in AADL is not new, moreover an extension released by the SAE in 2006 (13) was created to fulfill this need: the error model annex. This annex should provide artifacts to model dysfunctional behaviour in AADL and provide help to generate dependability studies. The use of the error model annex and the enhancement it can provide to safety studies are well presented in works such as (14).

The main difference between our dysfunctional representation in the DBD using classic AADL artifact and the use of the error model annex is the modeling of failure propagation: because our SysML DBD was made to ease FMEA analysis, the failure propagation is made through the fact that the data transmitted are corrupted and false, but no new signal is emitted (i.e. the error model annex use a signal dedicated to the propagation of an error), then a component must compute a diagnosis of their input data to detect a failure. It's very efficient to simulate the whole system in functional and dysfunctional mode and to study the real impact of a failure on output data. But this method is too heavy to allow fault trees or Markov model generation, used generally to study safety, reliabili-

ty and availability. In addition, the errors models of low-level components need specific information for this type of studies. Dependability analysis requires dependability-related information from the model: fault assumptions, repair assumptions, fault-tolerance mechanisms, stochastic parameters of the system (i.e., the occurrence of fault events and propagations).

Finally, the error model annex will permit to enhance the dysfunctional models of our components from the DBD in AADL. The tools provided by the error model annex are very useful to carry out a dependability analysis of the system originally modeled in SysML that is used as the backbone of our entire method. However, we use our error model to take into account the main dysfunctional behaviour in the design step. This is underlined in the part four.

4 SUPPORT TO DESIGN USING MATLAB/SIMULINK

We want to transpose our method to standard engineering and safety tools. In a recent partnership, we used MeDISIS during the specification of a hypersonic vehicle, and we encountered several problems. The first was the deployment of our tools in our partner's industrial network, and the second was the deployment of our methodology on the tools commonly used by our partner. Simulink appeared to be a tool that could solve our issue since it is widely used in industrial processes and offers artifacts of modeling compatible with SysML. Furthermore Simulink is an important step in a design process since it permits to detail the design and to simulate the system.

We will now outline the help that can be brought by the modeling of our system using Matlab/Simulink. This model would allow us to simulate the system to get information about error propagation, early in the design process. We will highlight how to translate SysML artifact to Simulink and after, we will describe the possibility provided by the Simulink model to study the dysfunctional behaviour of the system during its design.

Concept	**Simulink**	**SysML**
Components	Block	Block / Part
Bindings	Line	Block Association
Subcomponents	Subsystems	Part
Connectors Flow	Inport / Outport Line	Flow ports Flow specification
States	Stateflow Diagram/states	State Diagram /States
Constraints	Block	Parametric diagram /Constraint block
Constraint association	Line	Parametric diagram /Connections
Requirement	Block	Requirement Diagram
Requirement association	Line	Requirement Diagram /Connections

Figure 3. Correspondence table between Simulink and SysML artifacts

First, we can easily find correspondence between SysML artifacts of modeling and the one from Simulink. **Blocks** and **line** are basic entities of a Simulink model. A **block** represents a system that might contain a subsystem. The subsystem is specified using **Inport** and **Outport** relationships. A **line** connects two **blocks** together. We can find equivalent modeling entities in Simulink and in SysML, as both languages are object oriented.

A Simulink **block** will be represented by a SysML *block* and a **subsystem** will be represented by an *internal block diagram* structure. **Lines** between Simulink **blocks** correspond to SysML *connectors* with *ports* attached to it. Control flow and data flow through a Simulink **connector** can also be directly represented as *control* and *data flow* in SysML. SysML provides options for standard port requiring service-based interface that is used in conjunction with *flow ports* to specify the **inport/outport** structure and **line** representing flow/interaction between **blocks** in Simulink. In terms of behaviour mapping, **Stateflow** in Simulink is represented by a *state machine diagram* in SysML. And the constraints imposed to our system that are modeled using *parametric diagrams* in SysML will be represented also using **blocks** and **lines** in Simulink.

As we can see in figure 3, some different artifacts in SysML will be transformed into the same type of artifacts in Simulink, for example: *lines* in Simulink will represent both the association connection and the flowport connections from the SysML model. In fact, the transformation from SysML to Simulink is surjective, which means that there will be a loss of information in the transformation process, or at least a loss of precision in the representation of the system. On the other way, the transformation from Simulink to SysML will produce a model far from being complete since some information needed in SysML cannot be stored in a Simulink model.

For example, tagging a line with "Association" if it was a block association in SysML or with "Flow" if it was a flow port connection.

This process added in MeDISIS (figure 1) will give us a functional model in Simulink in parallel with our functional model in SysML. In fact every process based on the SysML functional model would be conceivable, but the one that draws our attention is the FMEA synthesis. The FMEA synthesis will be easier due to the simulation of the system that will help find the effects on the systems of error propagation.

To make the simulation of error propagation possible, we will use a dysfunctional library associated with our DBD to complete our Simulink model with dysfunctional behaviour. Finally, the system will be simulated for each main failure mode to determine the possible causes and the effects it has on the system.

Based on a functional model of our System in SysML, we saw how we could help through the process of redacting an FMEA, and how the information obtained during this process could help us model the dysfunctional behaviour of the system. At this moment, we obtain the same level of modeling as

in the previous AADL process using the error model annex but we are able to study physical effects of a failure mode. The whole system can be simulated to check error propagation and the effects of such failure mode on the system and its blocks. It is now possible to design some mechanism or control law to avoid the propagation of failure in the system.

In addition with that aspect of modeling functional and dysfunctional behaviour, Simulink provides means to perform detailed design, to enhance the precision of our FMEA synthesis that is still possible from the Simulink Model as we can see on the figure 3.

4.1 D.B.D. update for design in Simulink

After building the FMEA from the functional model of the specification stage, we obtain the list of the failure modes and their severity. So, we can introduce in the design model, dysfunctional behaviour of selected components. Those selected components are the ones on which the effect of a failure is partially known or on which the designer chose to develop a control law or mechanism to decrease the effect of failure.

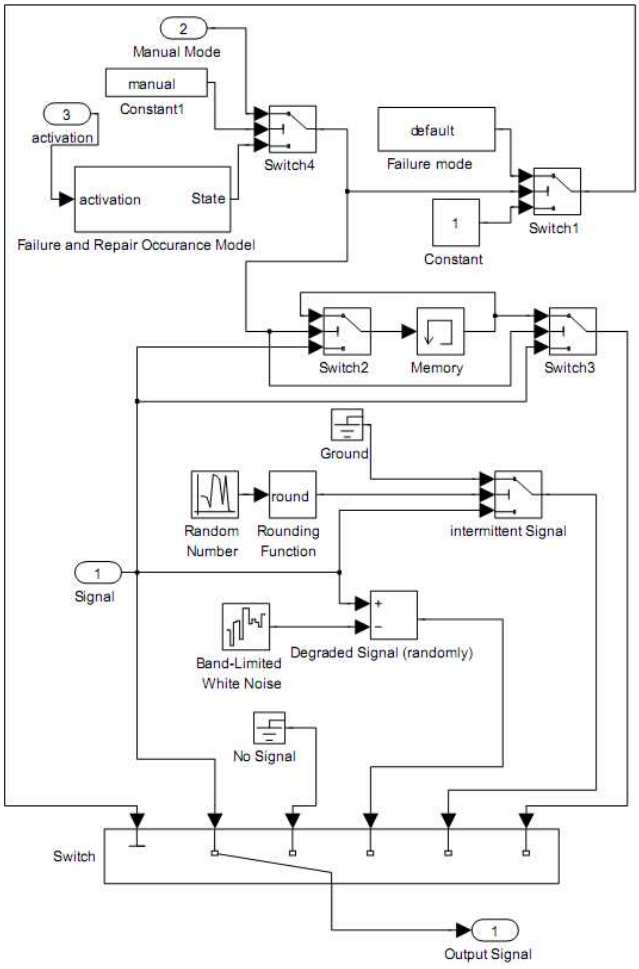


Figure 4 Simulink model of a failure mode.

Formerly, we must update the project DBD. As we said before, the FMEA process establishes a selection in the generic DBD. The achievements of the FMEA are done by the update of each component in the project DBD. The knowledge about dysfunctional behaviour can be obtained from the Altarica or AADL model analysis. For each selected component, the design of Simulink dysfunctional model is realized from a generic dysfunctional model chosen regarding the type of the component.

The dysfunctional generic model (figure 4) introduces some configuration parameters:

- Failure type,
- Activation mode,
- Failure and repair law

Concerning the failure type, there are multiple choices to consider: no service, degraded service, intermittent service, and even the "no failure" value.

The activation mode provides two choices: manual triggered and automatic activation following failure and repair laws. In our studies, we only use manual mode. We activate one or several failure modes following the test scenarios elaborated during previous safety studies.

Considering the possibility to choose automatic activation, we needed to introduce parameters of configuration:

- The failure law (Exponential law or Uniformly distributed Law),
- The failure rate to be used with an exponential law,
- The possibility to repair the system,
- The repair rate to be used following an exponential law.

After the update of the DBD, the designer have some new blocks that are the fault model to be selected to perform fault diagnosis, feedback control with fault rejection or fault tolerant control. This activity follows the rules and definitions from the field of control theory. In our actual study, we use the models defined by (15) which are efficient for additive faults and system structural changes for linear systems.

The failure mode block (figure 4)) represents a generic failure mode for an electronic component (i.e. adapted for embedded system). To have a better understanding, we illustrate our reasoning with an example of how to integrate such a block (figure 5) in a Simulink model.

4.2 Generic failure mode and the functional model

We can place the failure mode block for each signal that must be studied. Two points of view are possible for the placement of failure mode block (FMB). In case of fault injection study, we chose to insert the block as an input of a block (functional block or component block) to study the effect of the failure on this block. On the other hand, when we must study the impact of error propagation from FMEA results, we insert the FMB after the output of the faulty block. In this case, the output of the FMB block became the new output of the faulty block.

Figure 5 shows how this failure mode simulation block interacts on the signal. On this example, we decided to use an automatic mode configuring both failure and repair law in the block parameters. This explains why there is constant 0 signal

for the “manual mode” entry. A constant 1 signal connected to the “activation” entry allows activating the generation of the failure law. In fact, the “manual mode” port can receive a signal with pulse that will command the failure occurrence and the “activation” port can inhibit the failure occurrence in automatic mode. The signal on which the block is applied is represented by a Sine wave. The two-scope-representation below represents the signal of failure and repair occurrence and the output signal with its failure mode (i.e. no service).

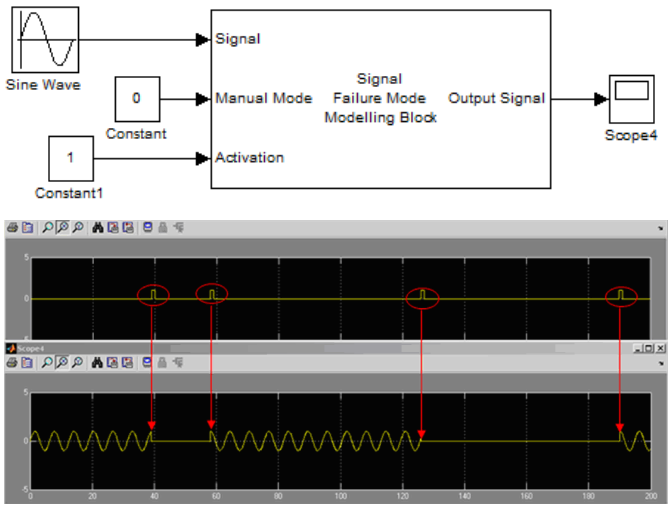


Figure 5 Example of automatic failure generation

CONCLUSION

The complexity of the multi-domain design and optimization at the specific tool level is a major obstacle against a better design process. While considering a model based approach for complex systems, we want to handle the complexity issues at the specific tool level, the exchange of model information and parameters between different domains, the communication from the specific tool level back to the system level (i.e. SysML model) and “include this process in the MeDISIS method. The embedded system specification can be led using the description language AADL. With the bridge to Simulink models, we finalize the specification stage and we begin the design process of our system.

Some work like (16) describe, from a control engineer point of view, a comprehensive method based on fault tolerant control scheme to design a fault tolerant controller of compact disc player. Our study links the FMEA process and the centric SysML model to this kind of work in a coherent and traceable way. The AADL process provides to the designer, the control structure of the embedded system and the Altarica DF process establishes the vectors of test from the most significant failure scenarios.

REFERENCES

1. Friedenthal, S., Moore, A. & Steiner, R., “A Practical Guide to SysML : The Systems Modeling Language”, *The MK/OMG press, Elsevier*, 2008.
2. David, P., Idasiak, V. & Kratz, F., “Automating the synthesis of AltaRica Data-Flow models from SysML”. *Proceedings of ESREL 2009*, Prague, Czech Republic, 7-10 September 2009.
3. David, P., Idasiak, V. & Kratz, F., “Use and improvements os SysML in reliability study”. *Proceedings of the 55th Annual Reliability and Maintainability Symposium, RAMS2009*, Fort Worth, Texas, USA, 26-29 January 2009.
4. David, P, Idasiak, V. & Kratz, F., “Reliability study of complex physical systems using SysML”, *Journal of Reliability Engineering and System Safety*, Volume 95, Issue 4, April 2010, Pages 431-450
5. OMG 2008. “OMG Systems Modeling Language (*OMG SysML) V1.1.*, 1st November 2008.
6. Rauzy, A., “Mode Automata and their compilation into Fault tree”. *Reliability Engineering and System Safety 78: 1-12*, 2002.
7. SAE, Society of Automotive Engineers, “*Architecture Analysis & Design Language. Specification V2”*, January 2009.
8. Singhoff, F., “The Cheddar AADL Property sets (Release 2.x). *LISyC technical report*, February 2007
9. Price, C. & Taylor, N., “Automated multiple failure FMEA”. *Reliability Engineering and System Safety* Vol. 76, pp. 1-10, 2002.
10. Teoh, P. & Case, K., “Failure modes and effects analysis through knowledge modelling. *Journal of Materials Processing Technology* 153-154, pp. 253-260, 2004.
11. Klein, M., Ralya, T., Pollak, B., Obenza, R., Harbour, M. & Harbour G. “A practitioner’s Handbook for Real-Time Analysis”. *Kluwer Academic Publishers*, 1993.
12. Cressent, R., David, P. & Idasiak, V. & Kratz, F., “Increasing Reliability of Embedded Systems in a SysML Centered MBSE Process: Application to the LEA Project”, *1st M-BED workshop, during DATE 2010*, Dresden, Germany, 12 March 2010.
13. SAE, Society of Automotive Engineers, June 2006. “*SAE Standards: AS5506/1, Architecture Analysis & Design.*
14. Feiler, P. & Rugina A. “Dependability Modeling with the Architecture Analysis & Design Language (AADL)”**,** *Carnegie Mellon Institute***,** *CMU/SEI-2007-TN-043*, July 2007.
15. Niemann, H. & Stoustrup, J., “An Architecture for Fault Tolerant Controllers”, *International Journal of Control*, 78(14):1091-1110, 2005.
16. Odgaard, P.F., Stroustrup, J., Andersen, P., Wickerhauser, M.V.& Mikkelsen, “Feature based handling of surface faults in compact disc players”. *Control Engineering Practice, Volume 14, Issue 12*, 2006.

BIOGRAPHY

CRESSENT Robin
ENSIB,
88 boulevard Lahitolle,

18020, Bourges CEDEX, France
e-mail: robin.cressent@ensi-bourges.fr

Robin CRESSENT holds Master's Degree in Engineering and Risk Management from the Ecole Nationale Supérieure d'Ingénieur de Bourges (ENSIB, engineering school). He is currently pursuing his Ph.D. in Control Engineering at Orleans University, working for the team-project Modeling, Control and Diagnosis of Systems (MCDS) of the PRISME laboratory. His research topics are RAMS activities supported by the Model-Based System Engineering approach, applied on industrial projects.

IDASIAK Vincent
ENSIB,
88 boulevard Lahitolle,
18020, Bourges CEDEX, France
e-mail: vincent.idasiak@ensi-bourges.fr

Vincent IDASIAK received the Ph.D. Degree in Software Engineering and Robotics in 1996 from the University of Pierre et Marie Curie (Paris VI). Since 1997, he is assistant professor at the Ecole Nationale Supérieure d'Ingénieurs de Bourges and a member of the team-project Modeling, Control and Diagnosis of Systems of the PRISME laboratory. His interests include Real Time System, System Engineering and Formal Methods, Safety and Reliability Studies of Complexes Systems.

KRATZ Frédéric
ENSIB,
88 boulevard Lahitolle,
18020, Bourges CEDEX, France
e-mail: frederic.kratz@ensi-bourges.fr

Frédéric KRATZ received the Master's Degree in Engineering in 1988 from the Ecole Nationale Supérieure de Physique de Strasbourg, France, the Ph.D. Degree in Electrical Engineering in 1991 from the University of Nancy, France and his Accreditation to supervise researches in Electrical Engineering in 1998 from the Institut National Polytechnique de Lorraine (INPL). He has been employed at the INPL from 1992 to 2000 as an assistant professor and from 2000 to 2005 at the University of Orléans, France as a professor. Since 2005, he is professor at the Ecole Nationale Supérieure d'Ingénieurs de Bourges. His interests include process control, diagnosis for nonlinear hybrid systems. He is then head of the team-project Modeling, Control and Diagnosis of Systems of the PRISME laboratory.