Modeling framework based on SysML and AltaRica data flow languages for developing models to support complex maintenance program quantification

Thomas Ruin, Eric Levrat, Benoît Iung

Modeling Framework based on SysML and AltaRica Data Flow languages for developing models to support complex maintenance program quantification

T. Ruin*, E. Levrat*, B. Iung*

* *Lorraine University, CRAN (Nancy Research Center for Automatic Control), CNRS UMR 7039,Campus sciences, B.P. 70239 54506 Vandoeuvre lès Nancy, France (e-mail: {thomas.ruin; eric.levrat; benoit.iung}@univ-lorraine.fr).*

Abstract: With the financial crisis attacking every industry and the new sustainability requirements such as the extension of a system operation time subject to ageing life (i.e. nuclear power plant), the importance of maintenance being effective and efficient is one of the top priorities for any industrial company. This challenge cannot be achieved only through conventional maintenance optimization models focusing mainly on few components but through maintenance programs based on "system thinking" considerations. In that way, managers need to have at their disposal new decision-making tools well adapted to support these considerations and allowing comparing off-line the impact of maintenance programs on complex system performances like costs and availability (Complex Maintenance Program Quantification – CMPQ). Thus, this paper proposes a model driven framework based both on the use of SysML to model a system-of-interest subject to ageing and maintenance and on the use of formal language AltaRica Data Flow to support model simulation.

Keywords: Maintenance engineering, Discrete-Event simulation, SysML, AltaRica Data Flow

1. INTRODUCTION

Production systems (or system-of-interest; SoI) are planned and controlled with the objective to supply products with the expected performances. In that way, the system features and capabilities are designed *a priori* knowing that initially, the production system performs as designed. As time passes, the components age and un-planned failures occur, causing the system performance to drift away from its initial state (Wiendahl and al., 2007). Thus maintenance is needed to restore or maintain the system in operational conditions. It means, for keeping performance optimality, to offer maintenance managers models allowing to take decisions about the maintenance strategies and programs (selecting new ones; re-designing existing ones) to be implemented (Takata and al., 2005). So, the importance of maintenance being effective and efficient is one of the top priorities for any industrial company. In that way, models used to support optimal maintenance strategies generally cover four main aspects (Dekker, 1996):

- a description of the SoI being maintained;
- knowledge on the SoI deteriorates and deterioration consequences;
- a description of the available information on the SoI and the available response options
- an objective function according to which the optimal maintenance strategy has to be derived.

Therefore there is a proliferation of "optimal maintenance models" (Wang and Pham, 2006) not well adapted to the industrial reality, in terms of mastering the SoI complexity, the interactions between the SoI and its enabling systems such as the support (maintenance and resources) one, the new sustainability considerations, the reuse of model based on COTS (Components Off The Shelf) principle (INCOSE, 2010).

For example, the extension of SoI operation time subject to ageing life (i.e. in the case of French nuclear power plants) implies to model new constraints. Indeed at their initial planned End-Of-Life, these systems must continue to operate during an additional period rather than to be dismantled or destroyed (i.e. economical reasons). These new constraints are the management of the additional inspections, the ignorance of some degradation processes (e.g. emergence of new degradation laws) related to duration extension and its evolving under fluctuating environmental and operational conditions. To face most of the lacks on current optimization models previously mentioned, the contribution developed in this paper consists in proposing a generic modeling framework for Complex Maintenance Program Quantification (CMPQ) built in two parts:

- A "static and interactional" part based on SysML semi formal language to model all the knowledge related to the SoI, its missions, the support system (i.e. maintenance strategies and resources) and their common interactions. This set of knowledge is structured with generic concepts (i.e. components, functions, maintenance etc.) in consistence with standards such as MIMOSA (*www.mimosa.org*) and (EN13306, 2001) in order to obtain a reference (ideal) model for CMPQ. Reference means "able to address all the CMPQ considerations whatever SoI application domains are".
- A "concept behaviour" part resulting from the transformation of the previous concepts formalized with SysML into dynamic behaviour based on AltaRica Data Flow (ADF) formal language.

The main result of the generic framework is a library of generic "concept behaviour" modeled with ADF. Then this library can be used, for a specific SoI, to develop the specific executable model needed to support, by simulation, the

CMPQ. This particular model results (a) from the instantiation of the generic "concept behaviour" with regards to the specific application knowledge (i.e. degradation laws specification for the "component" concept, the maintenance period specification for the "maintenance" concept) and (b) from the assembling of these instantiated behaviours.
In relation to this framework considerations, section 2 highlights more precisely the problem statement on CMPQ, and then section 3 justifies the different items of the proposed framework. Section 4 and 5 detail the framework steps, and an application on a case study is made on section 6. Finally conclusions and perspectives are given in section 7.

2. PROBLEM STATEMENT ON CMPQ

Monnin and al. (2011) underline the comparison between the different main scientific approaches related to maintenance decision, and describes CMPQ as a medium time decision applying to a wide number of components. In comparison, contributions related to maintenance optimization are more linked to a long term decision (Wang and Pham, 2006; Barros and al., 2009) and focused on one or few components. In that way, CMPQ can be considered as a dependability study with a wider modeling scope because it has to take into account not only the maintenance strategies but also the whole support system. It leads that CMPQ relies on the framework proposed by the System Engineering (INCOSE, 2010) to assess Key Performance Indicators (KPI) (Crespo-Marquez, 2008), and described by the following steps:

- Choice of the indicators to be assessed (KPIs),
- Model building according to expected KPIs,
- Model execution to assess KPIs.

The translation of these steps for CMPQ context implies:

- To define required KPIs both related at least to maintenance costs and dependability,
- To create reference model of the concepts (SoI and Maintenance considerations) in order to reduce the modeling effort for each study,
- To build any SoI according this reference model in order to assess required KPIs.

In relation to CMPQ modeling phase, different modeling techniques can be used as shown in Table 1 mainly in relation to the applications domains of the SoI. Nevertheless, only few of them (Medina Oliva and al., 2011; Monnin and al., 2011) focused on support system aspects in order to assess costs related to maintenance.

Table 1. Classification of some CMPQ models

	Model building	Model execution
(Betous Almeida and al, 2004)	Stochastic Petri Nets	Stochastic Petri Nets
(Medina and al., 2011)	Bayesian Networks	Bayesian Networks
(Clavereau and Labeau, 2009)	Stochastic Petri Nets	Stochastic Petri Nets
(Monnin and al., 2011)	UML	Stochastic Activity Network
(Boiteau and al., 2006)	ADF	ADF
(Zille and al., 2009)	Un-formal	Stochastic Petri Nets

Table 1 puts in evidence two kinds of approaches:
(a) Approaches driven by the simulation tool (Stochastic Petri Nets, Bayesian Networks). Although these tools present the advantage to support both modeling and simulation aspects (and to allow a gain of time), they cannot model some complex processes and interactions. It induces crucial problems for the model reusability.
(b) Model-driven approaches where the simulation tool modeling is preceded by a high level (or natural) language modeling step. They present the advantage to model knowledge from a generic informal or semi formal (and so capitalizable) model, but imply a transformation language step between the model building language and the model execution language. Semi formal languages (UML2 (2010), SysML (2008)...) provide, by means of different views (static, interaction and behavioural), a semantic frame needed to create a static (reference) model but also helping the transformation language step to create simulation model. Thus in relation to CMPQ considerations, the main challenge for such approaches consists in selecting a formal language (supporting model execution) well adapted to represent the "concept behaviour" knowing that the concepts have been previously formalized. Most of the time, a state-transition formal tool/ language is chosen and has to perform the following constraints:

- to be able to model accurately CMPQ related knowledge in formal concepts,
- to manage SoI complexity (i.e. concepts reusability).

A lot of specific tools exist to be in phase with previous considerations. (Trivedi and al., 1993) studied classical ones (Petri Nets, Markov Chains...) and highlighted lacks notably in model reusability. New high level formal languages for FIGARO (Bouissou and al., 2002), SDM (Ramesh and al. dependability analysis (AltaRicaDF (Rauzy, 2002),, 1999)...) appeared some years ago. They own interesting genericity abilities and allow representing easily a complex system according to COTS libraries and remain able to model complex interactions and phenomenon (Boiteau and al., 2006).

On the basis of the semi-formal and formal languages previously identified for approaches (b), it is necessary now to select the more suitable ones in relation to CMPQ framework for supporting both the "static-interactional" part and the "concept behaviour" one.

3. LANGUAGES PROPOSED FOR THE FRAMEWORK

3.1 The use of a SysML-based framework for « static and interactional » modeling.

Although some works deals with the transformation from UML language to simulation tools (SAN (Monnin and al., 2011), Stochastic Petri Nets (Bernardi and al., 2008)), this language remains more suitable for information system modeling. SysML language, which is an extension of UML, appears more adapted not only for industrial system modeling but also for the model reusability during the design phase. Indeed, Hoffman (2008) proposes the use of different SysML

diagrams (providing static, interactional and behavioural views) during the whole life cycle of a system. However, some lacks remain for the quantification of the support system impact on the SoI leading to mistakes in design. The use of SysML for CMPQ allows to fulfill this lack, and to allow quantifying the Support System organization on the SoI design. However, although blocks, sequences, and parametric diagrams are particularly suitable to model interactions between different concepts needed for CMPQ, the SysML "concept behavior" view (State Machine diagrams) does not provide a well defined semantic frame (Borger and al., 2000) inducing difficulties for simulation. It leads to keep the principle of selecting another language to support simulation (with a step of language transformation): ADF language.

3.2. The use of AltaRicaDF language for « concept behaviour » modeling.

Among formal languages identified previously in section 2, ADF presents a system engineering construction philosophy, in order to be compatible with other languages like Lustre or Modelica dedicated to other modeling views**.** The ADF language presents a well defined semantics relying on the mode automaton formalism. A mode automaton, formally defined in Rauzy (2002), is an input/output automaton. It has a finite number of states that are called modes. At each instant, it is in one (and only one) mode. It may change of mode when an event occurs. In each mode, a transfer function determines the values of output flows from the values of input flows. In addition, this formalism allows both to model and store reusable objects as libraries elements and to minimize the SoI modeling effort. Some simulation environments (Safety Designer (*www.3ds.com*), SIMFIA(*www.apsys.eads.net)*) allowing to manage these libraries without generate complementary ADF code.
(David and al., 2010) already deal with the creation of a ADF code from a SysML model. Moreover, this language is more and more used in industry and some dependability dedicated software tools are able to execute it. They allow both to store the "concept behaviour" models within libraries and to perform stochastic simulation by assembling and instantiation of the suitable "concept behaviour". Although this language is used in the proposed framework, it can be noted that it presents some restrictions like the impossibility to model looped systems (i.e. the state of a node is function of the state of another and vice versa). However, recent works on ADF new versions tend to address this issue (Rauzy, 2008).

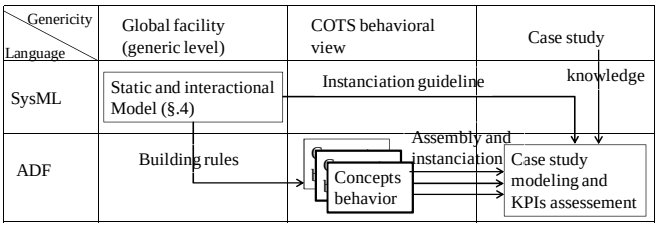

Fig. 1. Modeling framework

In summary, Fig. 1 illustrates the proposed framework, through its different steps: from the initial SysML based reference model until ADF-based simulation. From now on, SysML related items (extracted from SysML (2008)), will be marked in *italic* characters and ADF elements in **thick** ones.

4. "STATIC AND INTERACTIONAL" MODELING

This part modeled with SysML diagrams has been supported by Rational Rhapsody tool (*www.ibm.com*).

4.1 The static model: the use of SysML block diagrams

On the static model, a first scientific investigation has been published in (Ruin and al., 2012). This model is constructed in terms of block diagrams formalizing in concepts the CMPQ related knowledge. Thus the blocks *attributes* are extracted from maintenance related standards and works with regards to the required KPIs. Diagrams are structured with three main items:

- the SoI composed of its components/asset including related concepts (e.g. failure modes, degradation mechanisms, symptoms…) and their links (e.g. parallel),
- the SoI missions (e.g. environmental and operational conditions…),
- the support system composed with the maintenance system (e.g. maintenance strategies…) and the resources (operators, spare parts…).

These block diagrams do not contain whole CMPQ information. Indeed, interactions between objects are just specified by relations between *blocks*. Thus static additional knowledge regarding relations between *attributes* (modeled by means of SysML internal block diagrams) is required. Moreover, knowledge on dynamic aspects (modeled by means of SysML sequence diagrams) is needed to go towards the simulation.

4.2 The interaction model: the use of SysML-additional diagrams

Sequence diagrams allow to model CMPQ system scenarios. For example, making part in CMPQ, interactions between different *lifelines* equivalent to different *blocks* defined previously are modeled. In that way, the model shown in Fig.2, formalizes the interactions between concepts describing corrective strategy. This kind of strategy is divided in three states modeled by *condition marks* (plan: modeling the strategy waiting state; "in_prep": modeling the strategy requested state; "in_progress": modeling the maintenance action application). Each *condition mark* occurrence is preceded by *looped back messages* modeling the corrective maintenance internal event. These *messages* can be (a) temporized (e.g. the *message* "end_mc" is related to maintenance duration parameters), or (b) immediate and triggered by the occurrence of a receipted *message* (e.g. the *message* "prep_mc" is triggered by the *message* "fail").

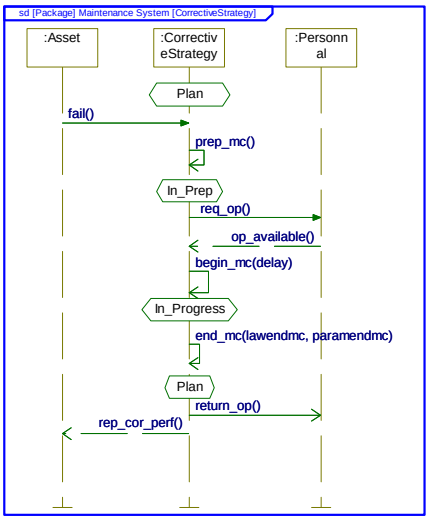

Fig 2. Sequence diagram for corrective maintenance scenario

By keeping the same modeling way for all possible scenarios related to each *lifeline* (corrective maintenance is only one possible scenario), the generic activity diagram related to each *lifeline* and gathering semantic of all SysML sequence diagrams can be automatically generated. This kind of diagram provides some dynamic information on interactions between *blocks* identified on the static model suitable for a future transformation into a discrete event formalism.
However, some quantitative modeling is needed to assess KPIs. SysML parametric diagrams are particularly suitable for supporting this issue because their goal is to model the *block* plus *attributes* and their relationships. For example, *constraintparameters* are blocks *attributes* impacting or impacted by others, as shown in Fig. 3. Their relations are modeled by an equation represented in the *constraint*. It should be emphasized that, to be suitable with ADF formalism, this step must avoid:

- to model looped systems,
- to express the *constraintparameter* equations in another form than the logic one.

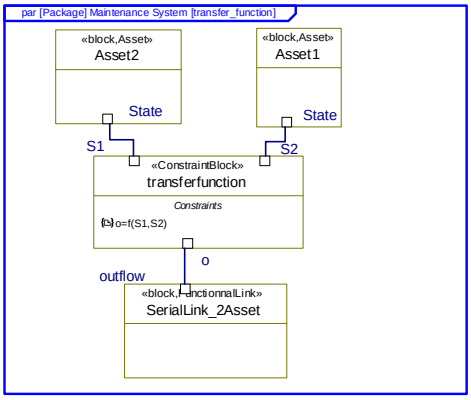

Fig. 3. Parametric Diagram for a two serial components

The *constraint* expressed in Fig. 3, models the impact of two serialized components on the system availability (*constraintparameter* "outflow"). This generic equation will be instanciated according knowledge feedback for any case study. Parametric diagrams can be used also to model the impact of an operator skill on maintenance action efficiency.

In summary, these two additional diagrams provide a global view of a set of *blocks* in interactions. At this step, SysML state machine diagrams can be generated according previous sequence diagrams in order to perform model checking on the model. However, this paper focus on stochastic simulation, and next session addresses the creation of a "concept behaviour" library by transforming the concepts into ADF formalism.

5. "CONCEPT BEHAVIOUR" MODELING

Now as the previous diagrams, it is necessary to translate previously built SysML diagrams into the ADF language, more suitable for execution and simulation requirements. ADF is based on the mode automaton formalism are not suitable

5.1 The ADF language

An ADF code is structured in 7 main items: The part **node** where the object is defined; the part **state** where the set reachable in a node is defined; the part **flow** where the different flows able to transit in a node, their direction and their type (real, bool…) are defined; the part **event**, where the different events related to a node are defined; the part **trans** where the impact of different events on states of a node are modeled; the part **assert** where the transfer function of a node is model, according to the node states, flows, and transition; and finally the part **init** where initial state is given.
As it is a high level language, mode automaton allows the systems description like hierarchies of reusable components to master system complexity. It is done by means of three operations: the **parallel compositions**, the **connections** and the **synchronizations**.

5.2. From SysML language to ADF language

SysML diagrams were constructed with a semantic frame making the transformation language step to ADF easier. ADF "concept behaviour" can be created exhaustively by means of algorithms from both parametric diagrams and sequence diagrams elements, to ADF parts (7 items).
For example, Fig. 4 describes the algorithm allowing the part creation of every ADF (**node**, **state**, **event**, **init**, and **trans**, only if *messages* parameters are not *constraintsparameters*) code formalizing "concept behaviour" from a sequence diagram according to the following notations presented on table 2

Table 2. SysML elements notations

$\mathcal{E}=\{E^i\}$	Set of *lifelines*
n	Number of *lifelines*
$\mathcal{S}^i=\{S^i_j\}$	Set of *condition marks* on the i^{th} *lifeline*
p	Number of *condition marks* on the i^{th} *lifeline*
$\mathcal{T}^i_{j,j+1}=\{T^i_{j,j+1,k}\}$	Set of looped back messages between *condition marks j* and *j+1*
q	Number of looped back messages between *condition marks j* and *j+1*

Sequence diagrams have been designed by making sure that each looped back *message* is triggered by a new condition mark, or by a *message* receiving (from another *lifeline*). Thus only one or zero receipted *message* can occur between two looped back *messages*, and let's consider T'^{i}_{q} the receipted *message* before looped back *message q*.

```
For i from 1 to n
        Create node E^i
        Create init= state S^i_1
                For j from 2 to p
                Create state S^i_j
                For k from 1 to j
                        Create event T^i_k
                        Create transition S^i_j-1 |- T^i_k -> S^i_j
                        if T'^i_q≠{Ø}
                        Create sync <T'^i_q, T^i_k>
                        k=k+1
                End for
        j=j+1
        End for
i=i+1
End for
End
```

Fig. 4. Algorithm to go from SysML sequence diagrams to ADF

Fig. 5a is showing the ADF code obtained by applying these rules to the sequence diagram and its lifeline "CorrectiveStrategy" given Fig. 2. Fig. 5b is illustrating the **synchronizations** also created. **Synchronisations** differ for each system. Indeed, **synchronization** is exhaustive only when all interacting **nodes** are defined. By proceeding in the same way for parametric diagrams, exhaustive ADF code related to generic "concept behaviour" has been built.

```
node CorrectiveStrategy
 state
  Var : {Plan,In_Prep,In_Progress};
 event
  begin_mc,prep_mc,end_mc;
 init
  Var := PLAN;
 trans
  ( (Var = In_Prep)) |- begin_mc -> Var := In_Progress;
  ((Var = PLAN)) |- prep_mc -> Var := In_Prep ;
  ((Var = In_Progress)) |- end_mc -> Var := PLAN ;
edon
```

(a)

```
sync
    <fail, prep_mc>
    <op_available, begin_mc>
```

(b)

Fig. 5. (a) ADF code created from sequence diagram defined from Fig. 2 and (b) its ADF synchronisations

In summary, the results of the entire language transformation step can be materialised by a library of generic "concept behaviour" (COTS) modelled with ADF. These COTS can be then instantiated and assembled (to form an executable model) with regards to a specific application (specific SoI & Support System). The system hierarchy, links between objects, and the completion of synchronisation are defined at this step, according a guideline proposed in future works aiming to help the user of the tool to manage the library of COTS.

6. APPLICATION TO A CASE STUDY: A TWO COMPONENTS SYSTEM SUBJECT TO DIFFRENT MAINTENANCE ORGANIZATIONS

In order to show the interest and feasibility of the proposed framework, instantiation of generic "concept behaviour" is performed with regards to a simple case study in order to develop simulations. Because ADF has been designed to manage system complexity by linking assets functional flows, this case study will focus on a basic system (2 components) considering supply systems objects. Indeed it aims to assess maintenance organization impact on system availability according a set of libraries available. A production system compounded by two serialized identical components is considered. It is modeled according the previous sequence diagram with a component malfunctioning characterized by two two-level degradation mechanisms impacting one failure mode. An "As Good As New - AGAN" corrective maintenance action is applied on the failed component.

Let's consider the three following maintenance organizations:

(A) 2 independent scheduled preventive strategies acting on each component, with one maintenance operator for each component.

(B) Case (A) with only one maintenance operator for the two components.

(C) Case (A) with opportunistic rules on preventive strategies (e.g. if failure occurs on component 1 then preventive strategy on component 2 is triggered).

Preventive maintenance actions are supposed to be "As good As New", corrective maintenance duration law and parameters are *exp(0,05)* while preventive maintenance duration ones are *exp(0,1)*.

Table 3 gives model parameters according to instantiated parametric diagrams and knowledge feedback. For example, the *constraint* "transferfunction":

f=if {S1=failed or S2=failed then failed}

For each organization, an executable model is built from the COTS available in libraries. The library is stored on SIMFIA simulation environment supporting ADF language. Thus for developing the model, COTS are just picked up in a library and linked by the user through the SIMFIA GUI (Fig. 6) according proposed guideline. These links may be done trough **flows** (e.g. from "asset1" to SerialLink_2Asset") or through **synchronizations** (e.g. between operator1, CorrectiveStrategy1 and Asset1). Then, transitions are instantiated according to knowledge feedback and parametric equations.

Table 3. Instantiation parameters for the two two-level degradation mechanisms components

Degradation mechanism 1	Degradation mechanism 2	Failure mode impact
level 0	level 0	exp (1e-4)
level 0	level 1	Exp(1e-3)
level 1	level 0	Exp(1e-3)
level 1	level 1	Exp(1e-2)

Simulations have been performed for 1000 stories and 10000 UT. Only few **synchronization** modifications are needed to go from one model to another (i.e. Model A to Model B) because the reusability is maximal.

The simulation results of the three cases are given table 3, for the following KPIs: operator(s) and SoI availability.

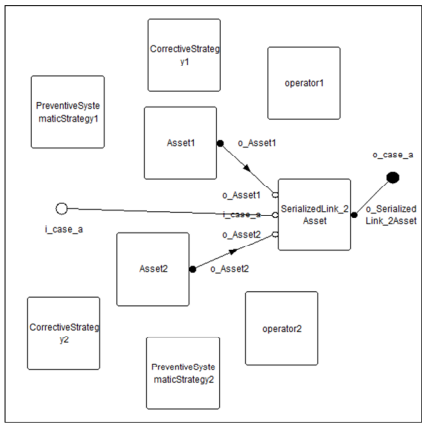

Fig. 6. Case A defined on SIMFIA GUI

Table 3. Simulation results for the three cases

	Preventive maintenance period	Operator(s) availability	SoI availabily
A	100 UT	0.931/0.882	0.882
B	100 UT	0.553	0.553
C	100 UT	0.956/0.968	0.956

The results are not industrially significant for such a SoI but they prove that the framework can provide some good indicators to compare different maintenance organizations. These results have been checked analytically under Markovian assumption for the simple case of one component subject to only one degradation mechanism and one failure mode.

7. CONCLUSIONS

This paper proposes a model driven framework based both on the use of SysML semi formal language to model the "static and interactional" part related to CMPQ and on the use of the formal language ADF both to model the "concept behaviour" part and to perform simulation. This high level language, initially made to model a flow propagation through an set of physical components, give satisfying results addressing objects like maintenance strategies or missions. However, some constraints related to simulation algorithm available in SIMFIA (considering only Boolean flow, impossibility to constraint transition parameter) induce to adapt initial model (e.g. impossibility to consider functional degradation…) and to restrict initial assumption (maintenance actions supposed AGAN). In future works, in addition to the implementation of an exhaustive COTS library, developments will be made on these restrictions in order to apply this framework to a wide scale industrial system.

REFERENCES

Barros A., Bérenguer C., Grall A. (2009). A maintenance policy for two-unit parallel systems based on imperfect monitoring information. *Reliability Engineering and System Safety*, 91, 131–136.

Bernardi, S., Merseguer, J., Petrinu, D.C. (2008). Adding dependability analysis to MARTE profile. *Proceeding: MoDELS'08 proceedings of the 11th international conference of Model Driven Engineering Language and Systems*, 2008.

Betous-Almeida C, Kanoun K. Construction and stepwise refinement of dependability models. *Performance Evaluation*, 56, 277–306.

Boiteau, M., Dutuit Y., Rauzy A., Signoret J-P. (2006). The altarica dataflow language in use: modeling of production availability of a multistate system. *Reliability Engineering & System Safety*, 91(7).

Borger, E. Cavarra, A, Riccobene, E.. Modeling the Dynamics of UML State Machines. *In Y. Gurevich, P. Kutter, M. Odersky, and L. Thiele, editors, Abstract State Machines: Theory and Applications*, volume 1912 of LNCS, pages 223–241. Springer-Verlag, 2000.

Bouissou, M., Humbert, S., Muffat, S., Villatte, N. (2002) KB3 tool: feedback on knowledge bases *ESREL 2002*, Lyon, France.

Bouissou, M. (2007). Gestion de la complexité dans les études quantitatives de sureté de fonctionnement de systèmes,*Paris : Lavoisier*.

Clavareau J., Labeau P-E. (2009). A Petri net-based modelling of replacement strategies under technological obsolescence. *Reliability Engineering and System Safety*. 94, 357-369.

Crespo-Márquez A. (2008). The maintenance management framework: models and methods for complex systems maintenance. *Springer Series in Reliability Engineering* ISBN-10:1846288207.

David, P. Idasiak, V., Kratz, F. (2010), Reliability study of complex physical systems using SysML, *Reliability Engineering and System Safety*, 95, 431-450.

Dekker, R. (1996), Applications of maintenance optimization models: a review and analysis. *Reliability Engineering and System Safety*, 51, 229-240.

Hoffman, H.P. (2008). "Harmony/SE - Model-Based Systems Engineering Using SysML" Hans-Peter Hoffmann, *Proceedings of the SDR '08 Technical Conference and Product Exposition*.

INCOSE (2010). *Systems Engineering Handbook : a guide for system life cycle processes and activities* (ed.3.2). International Council on Systems Engineering.

Medina-Oliva, G., Weber, P., Levrat, E., & Iung, B. (2010). Use of Probabilistic Relational Model (PRM) for Dependability Analysis of Complex Systems. *12th LSS IFAC symposium Large Scale Systems: theory and applications*, Villeneuve d'Ascq, France.

Monnin M., Iung B., Sénéchal O (2011): Dynamic behavioural model for assessing impact of regeneration actions on system availability: Application to weapon systems. *Reliability Engineering and System Safety*, 96(3), 410-424.

NF EN 13306 (2001). Maintenance terminology.

Ramesh, A.V., Twigg, D.W., Sandadi, U.R., Sharma, T.C., Trivedi, K.S., Somani, A.K. (1999). An integrated reliability modeling environment. *Reliability Engineering and System Safety*; 65:65-75.

Rauzy, A (2002). Mode automata and their compilation into fault trees Reliability Engineering and System Safety 78 1–12

Rauzy A. (2008) Guarded transition systems: a new states/events formalism for reliability studies. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability* ; 505, 222:495.

Ruin, T., Levrat, E., Iung, B., Despujols, A.. Using SySML language for maintenance decision-making model development to support complex maintenance program quantification. *Esrel 2012*, Helsinki, Finland

SysML. (2008). Systems Modeling Language. *Object Management Group*, Version 1.1.

Takata, S., Kimura F., van Houten F.J.A.M., Westkämper E., Shpitalni M., Ceglarek D. and Lee; J. (2005). Maintenance: changing role in life cycle management. *Annals of the CIRP*, 53 (2), 643–655.

Trivedi K, Malhotra M. Reliability and performability techniques and tools: a survey (invited paper). In: Proceedings of the 7th ITG/GI conference on measurement, modelling and evaluation of computer and communication systems. Aachen University of Technology, p. 27–48

UML (2010) 2.2 superstructure specification. *Object Management Group*, Needham, MA.

Wang, H., Pham, H. Reliability and Optimal Maintenance (2006). *Springer Series in Reliability Engineering*. Springer-Verlag, London.

Wiendahl, H.P., and al, (2007), Changeable Manufacturing – Classification, Design and Operation, CIRP Annals – Manufacturing Technology, 56/2: 783–809.

Zille, V., Zio, E., Rossetti, G., Despujols, A. (2009). Monte Carlo simulation for modelling degradation in maintenance programs assessment. *ESREL 2009*, Praha, Czech Republic.