Interoperability between a dynamic reliability modeling and a Systems Engineering process – Principles and Case Study

Gilles Deleuze, Aurélie Léger, Pierre Yves Piriou, Sylvain Chabroux

▶ **To cite this version:**

Gilles Deleuze, Aurélie Léger, Pierre Yves Piriou, Sylvain Chabroux. Interoperability between a dynamic reliability modeling and a Systems Engineering process – Principles and Case Study. Embedded Real Time Software and Systems (ERTS2014), Feb 2014, Toulouse, France. hal-02271335

HAL Id: hal-02271335
https://hal.science/hal-02271335

Submitted on 26 Aug 2019

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.

Interoperability between a dynamic reliability modeling and a Systems Engineering process – Principles and Case Study

Gilles Deleuze[1], Aurélie Léger[1], Pierre Yves Piriou[1], Sylvain Chabroux[2]

[1]EDF R&D, Clamart, France, {gilles.deleuze, aurelie.leger}@edf.fr
[2] Knowledge Inside, Versailles, France, sylvain.chabroux@k-inside.com

Abstract

Industrial systems are often large, and complex, in terms of structure, dynamic interactions between subsystems and components, dynamic operational environment, ageing, etc. The dynamic reliability approach is a convenient framework to model the behavior of such systems. However, there is a price to pay, e.g. in terms of amount of data, size of state graphs, volume of reliability calculations, and combination of various engineering activities. A sound Systems Engineering process, benefiting from the improvement of most recent tools, may be a fruitful approach to decrease these difficulties. Although feasibility demonstrations have been done for conventional, static, approaches of dependability, interoperability between dynamic reliability modeling and Systems Engineering has not the same maturity level. The article explains how, on the basis of Systems Engineering (SE) process definitions, a Meta-model defines a framework for integrating the safety into SE processes. It supports a “hub automaton”, that is the key element for interoperability with the tools and activities required for a dynamic reliability assessment. The case study is the dynamic assessment of availability of a feed-water control system in a power plant steam generator, presented in previous articles.

Keywords: Systems engineering, systems modeling, RAMS, dynamic dependability assessment, dynamic reliability, interoperability.

Introduction

Feasibility demonstrations and significant studies have been done to demonstrate the feasibility and benefits of an interoperability between System Engineering frameworks and Reliability, Availability, Maintainability and Safety (RAMS) activities ([David, 2010], [Aboutaleb, 2012]). However, these studies considered only so called “static” dependability approaches, assuming an invariant structure of the system. It focuses on “events combinations” with models like FMEA, static Fault Trees and Event Trees. Considering the system dynamics requires “dynamic” dependability approaches, focusing on event sequences with models like Dynamic Fault Trees, Boolean Driven Markov Processes…, and in term of Meta-model, a semantic adapted to the system behavior.

Improvement of the completeness of assessment of large and complex industrial systems, requires these “dynamic” dependability approaches, since they cover a wider range of phenomena. Given this context, the problematic is to develop meta-models integrated in a proven Systems Engineering (SE) process ([SE 2010], [IEEE, 2005], [ISO/CEI 26702, 2007], [ISO/CEI 15288, 2008]), properly combined with RAMS engineering activities, including dynamic modeling, and benefiting from recent support tools.

A SE process implemented in a consistent platform helps the engineer to manage the processes (description of functional and physical architectures, allocation of functions, preliminary assessment of requirements...) which are necessary, upstream of the dependability study. Then, it is possible to describe the failure features of the components, to model realistic failure/repairing scenarios and to define redundancy policies and dynamical allocation of functions caused by failure events. To do that, an idea is to develop a “hub automaton”, that supports the translation of the dynamic model into specific dynamic dependability tools.

This article is organized into four main sections:

1. Definitions and theoretical framework. We define the fundamental concepts for the study, e.g. System Engineering (SE), RAMS Activities and Hybrid System.
2. Principles. We present the key elements on which this demonstration is based: choice of interoperability vs. integration, System Engineering process supported by a Meta-model, Dynamic Reliability approach.

3. Method description. This section describes the processes and data used in the method and the Meta-model developed to support them.
4. Case Study. The case study is focused on the availability of a feed-water control system in a power plant steam generator. The meta-model is instantiated for specifying a sub-system, a dynamic model is generated for describing its behavior and an availability assessment is performed on it.

1. Definitions and theoretical framework

1.1. Systems Engineering

SE consists in concurrent interdisciplinary approaches for the design and validation of complex technical systems. The process proposed by INCOSE Systems Engineering Handbook [SE, 2010], integrates all the disciplines and specialty groups, e.g. RAMS, into a team effort that focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation.

1.2. RAMS activities

RAMS acronym refers to a set of engineering activities aiming to perform Reliability, Availability, Maintainability and Safety assessments, based on deterministic and probabilistic approaches. INCOSE Handbook [SE 2010] refers to Availability, Reliability, Maintainability and Supportability (RAMS), also known as dependability. The dependability of a system describes its ability to fulfill the required functions under given conditions, considering degradation of performance due to failure and maintenance. RAMS studies are considered in this case study, as validations, as defined by [ISO 9000: 2005], based on modeling and focused on dependability requirements.

1.3. System complexity

A technical system may be considered at the highest abstraction level, as a structured set of variable elements, acting in a technical environment for an explicit purpose. In the RAMS engineering domain, it is interesting to characterize its complexity by the combination of two attributes: “interactive complexity” and “tight coupling” [Perrow, 85]. Interactive complexity results from presence of dynamic phenomena, occurrence of rare event sequences and non-linear effects. The consequence is a risk of incomplete knowledge of the system. Tight coupling results from strong interdependence between phenomena. The consequence is a risk of dependent failures, e.g. common-cause and cascade failures.

1.4. Hybrid System

System that combines continuous physical processes, deterministic event sequences (defined by specifications), random events (that may be intrinsic, like failures, or extrinsic, like unexpected context modification).

1.5. Dynamic Dependability, Dynamic Reliability

Dynamic Dependability is a branch of dependability engineering that deals with *"the influence of time, process dynamics, and human actions, on system operations and failures, and accidental scenarios*." [Brissaud, 2011]. Discrete Dynamic dependability approaches rely on models like Dynamic Fault Trees, Boolean Driven Markov Processes…

Hybrid dependability approaches permit to improve the modeling of hybrid systems as they consider also the multiple interactions between deterministic and random discrete events, and continuous processes. The mathematical framework derives from Kolmogorov-Chapman equations [Labeau, Smidts, 2000]. Its applicationconsists basically in the modeling or the simulation of Piecewise Deterministic Markov Processes (PDMP) [Dufour, 2002]. Exact analytical solutions of stochastic differential equations are complex and require, beyond a small system size, simplifying assumptions, whereas approximation (using numerical methods or by Monte Carlo simulations) allow realistic modeling and provide data for comprehensive statistical analysis. It is very useful to apply this framework in the case of software intensive hybrid systems, as it explicitly considers the interactions between system states and context evolutions that may represent the most contributing risk factor to system failures [Leveson, 2011].

In addition to consideration of hybrid dynamic dependability, the so called dynamic reliability assume that some continuous phenomena (like for example, ageing), are influenced by stochastic events or drifts. It means that some reliability characteristics are influenced by the process.

2. Principles

2.1. Interoperability vs. Integration

Designing a complex system require a set of activities in interaction (Figure 1). This set is practically an organizational system and its management has to consider in particular the exchanges of data between the activities. At this point, a decision has to be done between ‘knowledge integration’ and ‘knowledge interoperation’ [Léger, 2009].

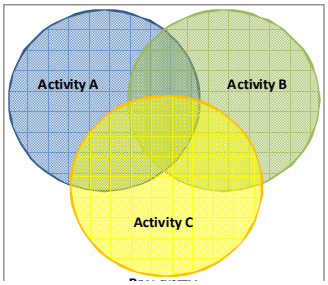

Figure 1: Activities in interaction for a real system

Knowledge integration consists in modeling the different activities of a system through a shared semantics (Figure 2). As such, the modeling focuses on knowledge extracted from each of the studied activities. The difficulty is to limit the loss of knowledge representation due to a common semantic.

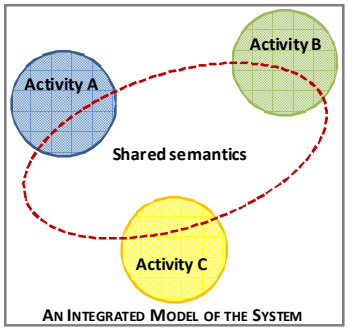

Figure 2 : An "integrated" model

Knowledge interoperation consists in using neutral exchange formalism as a “pivot” or “hub” for the interaction between the different activities (Figure 3). As such, a large amount of specific knowledge is kept in each activity.

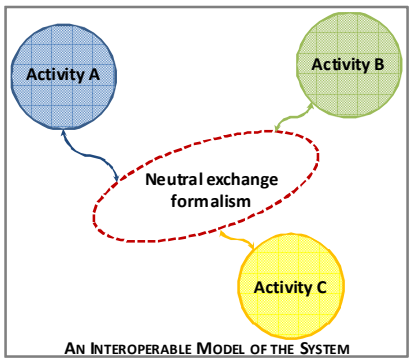

Figure 3: An "interoperable" model

For the case study presented here, the choice is to move towards an interoperation of the models through a dedicated Systems Engineering platform.

2.2. Choice of a Meta-model as a framework

In software engineering and systems engineering, a Data Model, or Meta-model, is a structured collection of "concepts" (things, terms, etc.) within a given knowledge domain. A model complies with a Meta-model in the way that a computer program complies with the grammar of the programming language in which it is written. The modeling language is used to manage diversity of objects types and their relationship, such as hierarchical, production/consumption... The properties of each type of object are defined by attributes. In this demonstration, the Meta-model allows to manage a large quantity of heterogeneous data.
In order to implement and support an innovative Meta-model for dynamic reliability, we used arKItect Designer, a Commercial Off-The-Shelf (COTS) tool having relevant capabilities for this demonstration:

1. Ready to use. As the SE framework is predefined, it was easy to design a Meta-model from scratch, estimated in the range of 10 to 30 times faster that with classical Object-Oriented (OO) approaches.
2. Interpretation. The Meta-model and the model can be modified dynamically, both are displayed in parallel in real time. This enables a fast prototyping of the Meta-model that seems more questionable, with other platforms where recompilation is needed after each modification of the Meta-model.
3. Generation of customizable building block diagrams. It allows getting immediately graphical editing views for all the artifacts needed in the design.
4. Easy to use. This tool does not require a high level of competence in Object Oriented approaches and programming to design a specific Meta-model. The rules for using the tool are rather simple and an engineer can build his data model and concentrate fairly on his own domain of activity.
5. Completeness. It is possible to represent in the meta-model all the artifacts needed to describe the system, as for example, stochastic automata.

2.3. Choice of a dynamic model

In a previous study, modeling the availability of a feed-water control system (considered as hybrid) in a power plant steam generator, the feasibility of two complementary approaches has been demonstrated [Aubry et al., 2012].

One approach models the system with Stochastic Hybrid Automaton (SHA) presented in [Babykina, 2011] and [Castaneda, 2011]. Analysis of the model may be quantitative, with Monte Carlo Simulations, to make dependability assessments; or qualitative, with technique of sequence exploration, to find sequences having a length inferior to a given criteria. The other approach, presented in [Zhang, 2012], models the system with State Charts and a dedicated COTS (Matlab/Simulink). The analysis of the model is quantitative, with Monte Carlo simulations

Both approaches require a combination of various engineering activities, the use of computational power, a large volume of data, component level models, and exchange between various tools (Matlab, Scilab…).
This demonstration focuses on the interoperability between SE process and a dynamic modeling based on SHA. SHA allow the exploration of event sequences and model checking, which is an interesting feature for dependability studies. Furthermore, EDF R&D has the disposition of an internally developed open source tool, Pycatshoo, also based on SHA [Chraïbi, 2013].

3. Method description

3.1. Specification of RAMS/SE interoperability

Dependability assessment requires a coupling between SE and the various RAMS activities. However, there is in practice sometimes a discontinuity between the system design and the dependability studies because they are considered lately in the project development. Functional analysis has to be performed during the system design phase, but it has also sometimes to be performed again during the dependability analysis, especially if the engineering framework does not consider RAMS/SE interoperability. Moreover, SE and RAMS teams often use tools that are not interoperating. This may lead to incoherence between the functional analysis and the failure analysis.

In a context of hybrid systems, interoperability between SE and RAMS engineering processes, is critical and must start early in the system life cycle. Their interoperability deserves particular attention all over the development process. Thus, it is needed to clearly specify the RAMS/SE interoperability on following points:

- Stages of the SE process,
- Stages of the RAMS process,
- Relations between the processes,
- Documentation to be generated. A tool such as ArKItect facilitates the production of systems and dependability studies. It can easily be used to provide diverse customizable views on the system and its dependability aspects.

3.2. Detailed Description of RAMS/SE interoperability

For this case study, [SE 2010] interpretation in the arKItect environment has led to the definition of a process divided into two sub-processes (see Figure 10, in appendix): (1) System Specifications (SS) and (2) System Design (SD).

The SS sub-process contains:

- Analysis of customer needs, detailed in terms of system lifecycle and mission phases, requirements statement, functions identification and context contributions definition tasks.
- Functional Architecture stage. It consists of requirements and functions refinement, definition of sub-functional flow exchanges. During this stage, the specification activity is iterative; allocating requirements upon functions allows the emergence of system sub-functions and requires the refining of requirements into sub-requirements.

The system Design (SD) sub-process contains:

- System Architecture (SA) stage. It consists in defining system components, functions allocation to components, definition of physical interfaces allocation of flows, and allocation of requirements to components.
  Refinement Feedback. Based on the requirements allocation upon systems, it insures with reasonable certainty, that the selected system architecture is adequate, given a “reasonable certainty” , with the requirements produced from the Customer Needs Analysis stage. The allocation of design requirements to the system components can lead to the generation of further functional requirements. The assessment of the “reasonable certainty” is the purpose of the risk assessment processes done in parallel.

The arKItect environment also covers Dependability studies and is divided into two sub-processes: (3) Preliminary Risk Analysis (PRA) and (4) System Risk Analysis (SRA). PRA is linked to the functions definition in the System Specifications stage. It includes:

- System state definition
- System risk event identification
- Undesired Customer Event (UCE) identification the preliminary risk analysis, where risks are linked to functions and priorized.

SRA is linked to the internal functional analysis from the SE process. It includes:

- ”Static” analysis such as Failure Modes and Effects Analysis (FMEA)
- Fault Tree Analysis (FTA). FTA risk events and basic risk events are identified in the System Architecture resulting from the SE process.
- For this case study, dynamic assessments capabilities have been added, through interoperability.

Results are capitalized into the arKItect project database and they are compared with safety and reliability requirements elicited previously in System Specification phase. ArKItect includes Python scripting language that ensures interfacing between

System Engineering platform and analysis tools, like for example, Fault trees analyzing tools and availability estimation tools. Another scripting application is the report generation for project deliveries.

3.3. Data and data management for dependability study

The Approdyn case study [Deleuze, 2011], deals with data relevant for Dependability studies, like system states, failure modes, availability, reparability, reliability …

The various data required for the dependability studies may be structured as follows:

- Functional description of the system (elements, context, operating conditions, states…).
- Qualitative and quantitative reliability data at element level, including failure modes, wear out and dynamic phenomena, state graphs (figure 4)…
- Models of the physical phenomena of interest for the RAMS studies of the system (including normal and possible abnormal conditions) and relevant control logic.
- Fault Tolerance Features (redundancy policy, defense in depth…).
- Qualitative and quantitative RAMS data at system level, including Operation and Maintenance procedures, Periodic Testing.

The challenge is to manage object relationships (like allocation, production, consumption, etc.) in interoperable SE and RAMS processes.

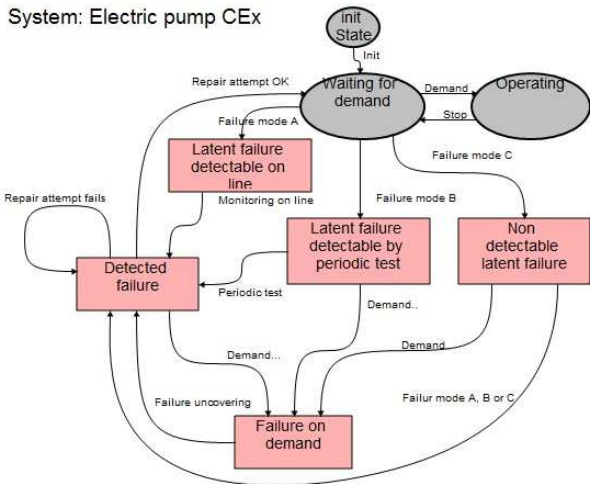

Figure 4: Example of state graph for one element of the steam generator

The SE process implementation requires managing project data relating to requirements, functional and physical architecture.

A graphical representation tool like arKItect proposes a solution for both efficient and seamless SE and RAMS processes. Indeed, the model is based on the system architecture, that means all activities related to the system design and to the dependability analysis are gathered in the same representation space.

3.4. The Meta-model for interoperability

Existing meta-models for SE, like [Pfister, 2012], organize the data and support the SE processes defined by INCOSE standard. However, they consider only the normal, faultless, operation of the system, whereas for critical systems safety and dependability matter too. Thus, the Meta-model defined by [Pfister, 2012], has been extended to be able to represent Dependability into SE processes. [Piriou, 2013] and [Piriou, 2014] describe in detail the design of this extension.
The meta-model proposed in this study adds the semantics required to perform dependability analyses on phased mission systems with repairable multistate components. The meta-model is represented by an UML class diagram [SysML, 2007].
It supports the development of a “hub automaton” or ”pivotal automaton”, comparable to a Guarded Transition System [Rauzy, 2008], which is the key element for interoperability between the tools and activities needed for dynamic reliability assessment. Figure 5 presents the meta-model.

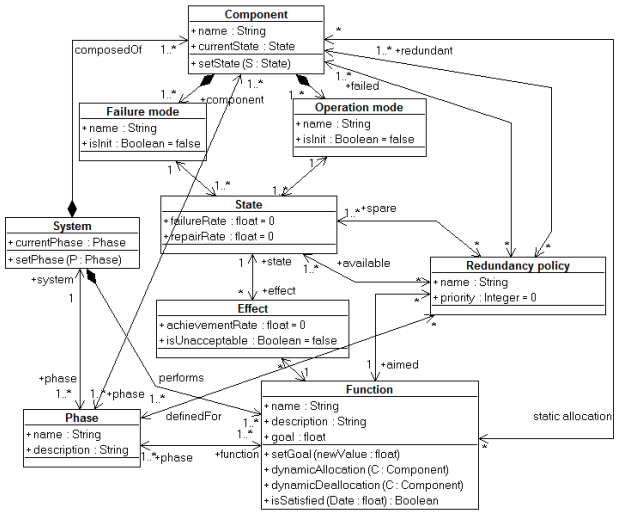

Figure 5: Complete meta-model for the integration of dependability analyzes into SE processes [Piriou, 2013]

It includes the following aspects:

Phased missions. A phased mission system is characterized by a structure, failure and recovery

processes, and success criteria that can change from one phase to another.

Component States. Each component can be activated and can fail according to several operation and failure modes. These modes represent respectively the functional and dysfunctional properties of the component. A component state is a pair built with one operation mode and one failure mode. Thus, possible states of a component are defined by instantiating its failure and operation modes.

The resulting class diagram enables to model the stochastic evolutions of the component in the form of transitions from a state to another. These transitions are due to failure or repair events that occur with a probability law depending on rates defined for the considered state.

It is assumed here that these rates do not depend on time. Thus the scope of the model to represent dynamic aspects of dependability is not fulfilled and will require further developments.

Effects of component states on function achievement. It is assumed, that the achievement of a function is quantifiable. This means that the achievement percentage can be computed for every function from the knowledge of the current states of its allocated component. For instance, the achievement percentage of a function whose aim is to fill in a tank and for which two identical pumps are allocated is equal to 100% when the two pumps are faultless and 60% when one pump is leaking.

Redundancy policies. The set of components allocated to a function can changes during operation; redundancy policies specify these changes. The “Redundancy policy” class and its relations are introduced in the meta-model. Also, two methods are added to the “Function” class that updates the allocation attribute of the “Function” class. The “Redundancy policy” class permits to specify how the faulty components are replaced by redundant components.

3.5. Systematic generation of a dynamic model

For describing the dysfunctional behavior of the system, we have developed an algorithm for building a dynamic model from an instance of this meta-model, in a generic way. Since the Stochastic Guarded Transition System (SGTS), defined by [Rauzy, 2008], allows a permissive and compact description of the system, we chose it as the formalism for representing this dynamic model. Moreover, the respect of best practices for using this formalism leads to constructions which preserve the structure aspect of the system. This conservation of data complies relevantly with the interoperability purpose.

A SGTS is defined as a 7-uplet <V,E,π,T,i,H,B> where:

- V is a set of variables
- E is a set of symbols called avents
- π is a priority function (a mapping of events to non-negative integers)
- T is a set of transitions, i.e. of triple <G,e,P>, where:
  - G is a guard (a Boolean expression built over V)
  - E is an event
  - P is a post-condition (an instruction built over V)
- i is an assignment called the initial assignment (a mapping from the set of variables to the set of values).
- H and B are two possibly empty instructions called respectively the head and body parts of the assertion.

The instructions can be seen as a mapping from assignments to assignments.

A transition $G \xrightarrow{e} P$ is fireable in a state (an assignment) s, if it validates the guard G and if the fix point $B^{\varpi}(H(P(s)))$ exists, i.e. there is an integer n such that $B^{n+1}(H(P(s))) = B^{n}(H(P(s)))$. The firing of this transition consists in three steps:

- First, the post-condition P of the transition is performed.
- Second, the head part of the assertion H is performed
- Finally, the body part of the assertion B is iterated until a fixpoint is reached.

Hence, the state $B^{\varpi}(H(P(s)))$ is called the successor of s by the transition $G \xrightarrow{e} P$.

4. Case Study

4.1. Description

The case study is focused on the availability of a feed-water control system used in a power plant steam generator. It is described in [Deleuze, 2011]. The case is a classical problem of hybrid dependability with dynamic reliability issue. It has been studied e.g in France [Aubry et al., 2012], [Zhang, 2012], [Deleuze et al., 2011] and USA [NUREG 6942].

4.2. Instantiation of the meta-model

In this article, only the sub-system composed of the two feeding turbo pumps (TPA, in Figure 9), are considered. This sub-system has to perform only one function F: “*Supply enough water to the steam generator*”. The pumps may fail and be repaired. To increase dependability of the function, the operation mode of each pump must be managed dynamically according to redundancy policies.

Defining Mission Phases
A first step is to build object diagrams on the basis of the meta-model. The main mission contains three phases (Table I). The function is mandatory in all phases but only one pump is needed to perform this function during the first and third phase.

id	role	description
$P1$	To increase the power from zero to the nominal value	A single pump is able to perform correctly the function.
$P2$	To produce the nominal power	The two pumps have to run together to perform correctly the function.
$P3$	To decrease the power from the nominal value to zero	A single pump is able to perform correctly the function.

Table I: Phases Description

Defining Component States
Instance diagrams are used to model the combination of operation modes and failure modes (see figure 6).

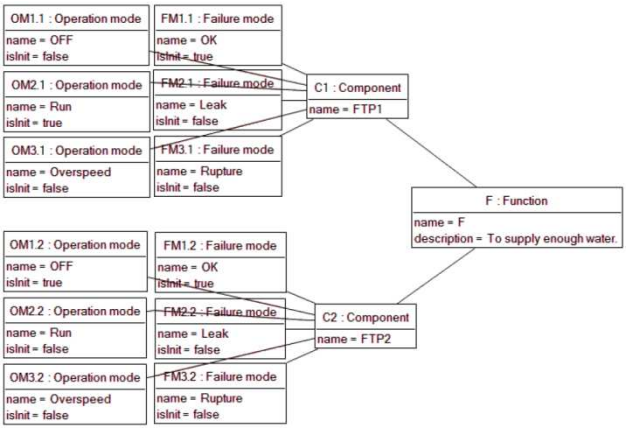

Figure 6. Example of Instance Diagrams

Figure 7 presents the quantification of the attributes of the State class instances. The first value (e.g. 0.01 for the state Run-Leak) is the failure rate of the transition that leads to this state and the second one (0.1 for the same state), is the repair rate to leave this state. As the pumps are identical, their states description are similar.

The contribution of one pump to the function depends on its state. For example, When the pump is disabled (operation mode OFF), this contribution is obviously equal to zero. This is also the case when the Rupture failure mode has occurred. The analysis is not easier for the remaining four states, especially as they suppose leaks, with no complete loss of function. Thus values of the attribute "achievementRate" for every state of a pump have to be defined (see Figure 8).

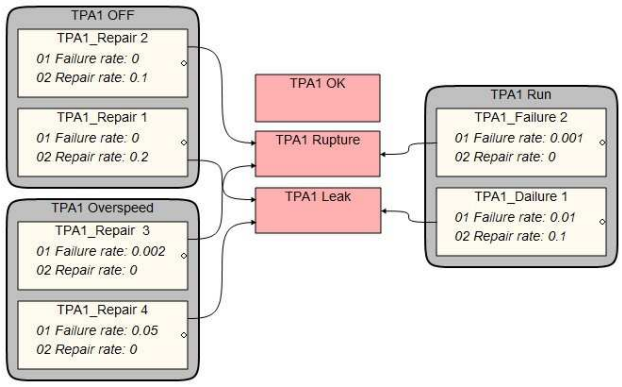

Figure 7. States description

Description of the effects of component states

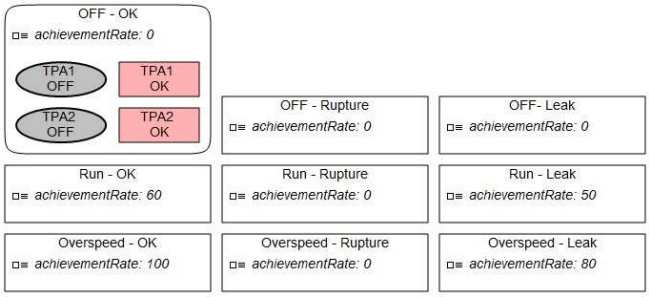

Figure 8: Description of the achievement rates of the pumps

Definition of redundancy policies
Two instance diagrams are also needed to describe the redundancy policy in the multiple phases.

4.3. Building the dynamic model

This subsection presents a large part of the SGTS which models the dysfunctional behavior of the pumps. It is built on the basis of the instance given in the last subsection.

Defining and initializing variables
The variables of the models are listed below:

- Phase = P1;
- F.goal = 60; F.isSatisfied = *True*;
- TPA1.OM = Run ; TPA1.FM = OK
- TPA2.OM = OFF ; TPA2.FM = OK
- R1.called = *False*; R2.called = *False*

Defining the transitions
There are three transitions for changing the current mission phase, whose priority is 0:

$$Phase = P1 \xrightarrow{\varphi_{1\to 2}} Phase = P2 \ \wedge F.goal = 100$$
$$Phase = P2 \xrightarrow{\varphi_{2\to 3}} Phase = P3 \ \wedge F.goal = 60$$
$$Phase = P3 \xrightarrow{\varphi_{3\to 1}} Phase = P1 \ \wedge F.goal = 60$$

There are seven stochastic transitions (not priority) by pump which model failure and repair occurrences. The values of probability rates are given by the Table II. Since the pumps are identical, only the transitions defined for the pump TPA1 are given below:

$$TPA1.OM = Run \wedge TPA1.FM = OK \xrightarrow[0.01]{TPA1.leak} TPA1.FM = Leak$$

$$TPA1.OM = Run \wedge TPA1.FM = OK \xrightarrow[0.001]{TPA1.rupture} TPA1.OM = OFF \wedge TPA1.FM = Rupture$$

$$TPA1.OM = Overspeed \wedge TPA1.FM = OK \xrightarrow[0.05]{TPA1.leak} TPA1.FM = Leak$$

$$TPA1.OM = Oversp \wedge TPA1.FM = OK \xrightarrow[0.002]{TPA1.rupt} TPA1.OM = OFF \wedge TPA1.FM = Rupture$$

$$TPA1.OM = OFF \wedge TPA1.FM = Leak \xrightarrow[0.2]{TPA1.repair} TPA1.FM = OK$$

$$TPA1.OM = Run \wedge TPA1.FM = Leak \xrightarrow[0.1]{TPA1.repair} TPA1.FM = OK$$

$$TPA1.OM = OFF \wedge TPA1.FM = Rupture \xrightarrow[0.1]{TPA1.repair} TPA1.FM = OK$$

Finally, there are four priority (immediate) transitions by pump, which is fired for forcing one of its operation modes, due to a redundancy policy request:

$$TPA1.OM = OFF \wedge TPA1.FM = OK \wedge R1.called = True \xrightarrow{TPA1.run} TPA1.OM = Run$$

$$TPA1.OM = Run \wedge TPA1.FM = OK \wedge R2.called = True \xrightarrow{TPA1.over} TPA1.OM = Overspeed$$

$$TPA1.OM = Run \wedge R1.called = False \xrightarrow{TPA1.OFF} TPA1.OM = OFF$$

$$TPA1.OM = Overspeed \wedge R2.called = False \xrightarrow{TPA1.run} TPA1.OM = Run$$

Defining the assertion

The assertion aim is to compute if the function is satisfied and if the redundancy policies must be called. The head part of the assertion is sufficient, and the body part is reduced to the identity.

H:

$$F.isSatisfied \leftarrow Effect(TPA1.OM, TPA1.FM) + Effect(TPA2.OM, TPA2.FM) \geq F.goal$$

$$R1.called \leftarrow (Phase = P1 \vee Phase = P3) \wedge not\ F.isSatisfied$$

$$R2.called \leftarrow Phase = P2 \wedge not\ F.isSatisfied$$

4.4. Contribution to System validation

The last stage we performed on this case study, is the assessment of the availability of the pumps for realistic scenarios. To do this, the SGTS described above, has been fully implemented using the tool PyCATSHOO [Chraïbi, 2013]. At this step, expert knowledge had been integrated to the model for filling the lack of knowledge due to the semantic used for interoperability. Finally, the availability is assessed with a Monte Carlo simulation (figure 9).
The result is obtained for a sequence of twelve identical missions where the first phase lasts 1 day, the second 28 days and the third 1 day. The average unavailability is equal to 0.62%.

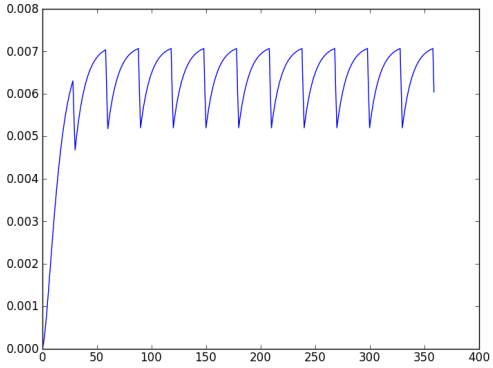

Figure 9: Unavailability of the pumps (x-axis : time in hours, y-axis : unavailability)

Traceability is maintained throughout all levels of system model, since the early Customer Needs Analysis phase, that provides functional and dependability requirements:

1. Allocation of System Requirements to hardware, software, or manual actions.
2. Allocation of all functional and performance requirements or design constraints, either derived from or flowed down directly to components.
3. Traceability of requirements from source documentation through the whole project life cycle.
4. Traceability of the history of each requirement on the system is maintained and is retrievable.

To bridge functional and failure analysis, the SE framework used, as a starting point, the recent ISO 26262 safety standard. It specifies how to ensure traceability between specification activities and those of validation. System validation ensures that requirements and system implementation provide the right solution to the customer needs in terms of functionality and dependability.

Conclusions

With the extension of SE Meta-model presented in [Piriou, 2013], this case study describes how, from outputs of a SE process, a RAMS engineer can model realistic failure/repairing scenarios, define redundancy policies and dynamical allocation of functions caused by failure events.

A sound SE process, supported by a tool like arKItect, can support the RAMS engineer to get data and models from the upstream processes (functional and physical architectures, allocation of functions, requirements...). It is especially helpful to manage the amount of data and the concurrent engineering activities required to assess dependability of hybrid systems, e.g. through traceability and capabilities to reconsider efficiently upstream stages of the design process with the output the RAMS studies.

A "hub automaton" support the translation of the dynamic dependability model into dedicated RAMS tools like Pycatshoo.

This case study is a first step towards interoperability of Systems Engineering and dynamic RAMS studies

Complementary studies have to be done at the hub automaton level to represent more hybrid aspects, and at RAMS models level, to improve the dynamic reliability modeling aspects.

References

[Aboutaleb, 2012] H. Aboutaleb, M. Bouali, M. Adedjouma, and E. Suomalainen, An integrated approach to implement system engineering and safety engineering processes: Sasha project. ERTS 2012, Toulouse, France, February 2012.

[Aubry et al., 2012] J.F. Aubry, G. Babykina, N. Brinzei, S. Medjaher, A. Barros, C. Berenguer, A. Grall, Y. Langeron, D. N. Nguyen, G. Deleuze, B. de Saporta, F. Dufour, H. Zhang, The APPRODYN project: dynamic reliability approaches to modeling critical systems. In: Supervision and Safety of Complex Systems, pp. 181–222, Wiley-ISTE editor, 2012.

[Babykina, 2011] G. Babykina, N. Brinzei, J.F. Aubry, G. Perez Castaneda, Reliability assessment for complex systems operating in dynamic environment, in European Safety and Reliability Conference ESREL, Troyes, France, 2011.

[Brissaud, 2011] F. Brissaud, C. Smidts, A. Barros, C. Bérenguer, Dynamic reliability of digital-based transmitters, Reliability Engineering & System Safety, Volume 96, Issue 7, July 2011, Pages 793-813.

[Chraïbi, 2013] H. Chraïbi, Dynamic reliability and assessment with PyCATSHOO: Application to a test case. PSAM, Tokyo, Japan, April, 14th-18th 2013.

[Castaneda, 2011] G. Perez-Castaneda, J.-F. Aubry, N. Brinzei, Stochastic hybrid automata model for dynamic reliability assessment, Journal of Risk and Reliability 225 (1) (2011) 28–41.

[David, 2010] P. David, V. Idasiak, and F. Kratz, Reliability study of complex physical systems using SYSML. International Journal in Reliability Engineering and System Safety, vol. 95, no. 4, pp. 431-450, 2010.

[Deleuze et al., 2011] Aubry J.-F., Babykina G., Barros A., Brinzei N., Deleuze G., de Saporta B., Dufour F., Langeron Y., Zhang H. Rapport final du projet APPRODYN: APPROches de la fiabilité DYNamique pour modéliser des systèmes critiques. Rapport de recherche. Ref. hal-00740181.

[Dufour, 2002] Dufour F., Dutuit Y., Dynamic Reliability: a new model, Lambda-Mu-13, ESREL02, vol. 1, p. 350-353, 2002.

[IEEE, 2005] IEEE Std 1220TM, Institute of Electrical and Electronics Engineers (IEEE): IEEE Standard for Application and Management of the Systems Engineering Process, 2005.

[ISO/CEI 15288, 2008] ISO/CEI 15288: Systems and software engineering – System life cycle processes. Second edition, 2008.

[ISO/CEI 26702, 2007] ISO/CEI 26702: Systems Engineering - Application and management of the systems engineering process. First edition, 2007.

[ISO 9000: 2005] Quality management systems. Fundamentals and vocabulary. International Electrotechnical Commission, October 2005.

[NUREG 6942] NUREG/CR-6942, Dynamic Reliability Modeling of Digital Instrumentation and Control Systems for Nuclear Reactor Probabilistic Risk Assessments, US Nuclear Regulatory Commission, Washington DC, 2007.

[Labeau, Smidts, 2000] P. Labeau, C. Smidts, S. Swaminathan, Dynamic reliability: towards an integrated platform for probabilistic risk assessment, Reliability Engineering & System Safety 68 (3) (2000) 219–254.

[Léger, 2009] A. Léger. Contribution à la formalisation unifiée des connaissances fonctionnelles et organisationnelles d'un système industriel en vue d'une évaluation quantitative des risques et de l'impact des barrières envisagées. Thèse de l'université Henri Poincaré – Nancy 1. Réf. tel-00417164.

[Leveson, 2011] N.G. Leveson. Engineering a Safer World. Systems thinking applied to safety. The MIT Press. Cambridge, 2011.

[OMG, 2003] OMG, Uml 2.0 OCL specification, Object Management Group, 2003.

[Perrow 85] Perrow C. Normal accidents: Living with high risk technologies. New York Basic Books, 1985.

[Pfister, 2012] F. Pfister, V. Chapurlat, M. Huchard, C. Nebut, and J.-L. Wippler, A proposed meta-model for formalizing systems engineering knowledge, based on functional architectural patterns, Systems Engineering, vol. 15, pp. 321–332, Autumn 2012.

[Piriou, 2013] P.Y. Piriou, J.M. Faure; G. Deleuze. A Meta-model for Integrating Safety Concerns into Systems Engineering Processes, SysCon 2013.

[Piriou, 2014] P.Y. Piriou, J.M. Faure, G. Deleuze, A Meta-model for Integrating Dependability Concerns into Systems Engineering Processes. Submitted to Systems Journal.

[Rauzy, 2008]. A. Rauzy. Guarded Transition Systems: a new States/Events formalism for Reliability studies, Journal of Risk and Reliability, 222(4), 495–505. 2008.

[SE, 2010] SE Handbook Working Group and International Council on Systems Engineering, Systems Engineering Handbook - A guide for system Life Cycle Processes and Activities. V.3.2. Edited by Cecilia Haskins, 2010.

[SysML 2007] SysML Specification 2007, OMG Systems Modeling Language (OMG SysML), V1.0 Object management group.

[Zhang, 2012] H. Zhang, B. de Saporta, F. Dufour, and G. Deleuze, Dynamic reliability: Towards efficient simulation of the availability of a feedwater control system, in NPIC-HMIT 2012, San Diego, USA, July 22-26 2012.

Figure 10. Interoperable System Engineering and RAMS processes developed for the test case