New Challenges and Opportunities in Reliability Engineering of Complex Technical Systems

Antoine Rauzy

Norwegian University of Science and Technology, Department of Mechanical and Industrial Engineering, S. P. Andersens veg 3, 7491 Trondheim, Norway, Antoine.Rauzy@ntnu.no

Abstract: In this article, we discuss the impacts of technological transformations currently at work on reliability engineering of complex technical systems. We consider transformations both in systems and in means to study them. We review challenges to meet in order to manage the current technological paradigm shift. We advocate the potential benefits of the so-called model-based approach in probabilistic risk assessment. We exemplified this approach by presenting the S2ML+X modeling technology.

Introduction

This article aims at discussing new challenges and opportunities brought to reliability engineering of complex technical systems by technological transformations currently at work. It echoes some reflections initiated a few years ago by Aven and Zio [1][2][3][4], and aims at contributing to the on-going debate about the future of our discipline.

All complex technical systems (aircrafts, nuclear power plants, offshore platforms, civil and military drones…) present risks to themselves, their operators and the environment. Therefore, one must ensure that these risks are economically, ecologically, and socially acceptable. This is the role of reliability engineering. Reliability engineering encompasses processes as diverse as safety analyses, optimizations of maintenance policies, assessments of the expected production level of a plant over a given period, assessments of the resilience of a socio-technical infrastructure and so on. In a word, reliability engineering aims at assessing the operational performance of systems subject to random events such as mechanical failures, operator errors, sudden changes in environmental conditions… With that respect, it relies on models and more specifically on stochastic models as its objective is to deal with aleatory uncertainties. Probabilistic risk analysis or equivalently probabilistic risk assessment is the process by which these models are designed and used to calculate performance indicators. The WASH1400 report [5], which followed the Three Miles Island nuclear accident, is usually considered as the historical starting point of the worldwide, cross industry adoption of probabilistic risk analyses. As of today, they rely mostly on modeling technologies such as fault trees, reliability block diagrams and event trees [6][7]. These technologies are well suited for mechanical systems and well mastered by practitioners.

In this article, we address two questions:

1. Are these modeling technologies still suitable to assess risks in new generations of systems, which are software intensive and rely on ubiquitous control mechanisms?
2. Can the new capacities provided by artificial intelligence and information technologies change the probabilistic risk analysis process?

The first question is indeed of importance because most of the systems currently designed by industry fall into this category. Our answer to this question is essentially negative. More powerful modeling frameworks are needed.

Our answer to the second question is positive, even though many challenges remain to meet. We advocate here that part of the answer relies on the so-called model-based approach in probabilistic risk assessment and more generally in systems engineering. This approach relies itself on a new generation of modeling techniques and tools that make possible to represent more accurately the behavior of complex technical systems, to maintain more easily models through the life-cycle of systems, to integrate seamlessly probabilistic risk analyses with other model-based systems engineering processes, and last but not least, to take advantages of the fantastic opportunities provided by artificial intelligence and information technologies.

We present here the main underlying ideas of this approach and exemplified them with the description of the S2ML+X family of domain specific modeling languages [8][9].

Of course, if the model-based approach can contribute to solve problems at stake, it does not solve all of them. As software engineers use to say, “there is no silver bullet” [10]. In particular, risk analysts must face the combinatorial explosion of the number of scenarios to analyze, with inherently limited calculation resources, whatever technology is used to support the analysis [11]. We shall discuss here this issue and review some other of the challenges to meet.

The contribution of this article is thus threefold. First, it discusses the two above questions, with the point of view of a computer scientist. Second, it provides a brief introduction to the model-based approach in probabilistic risk assessment. Third, it discusses challenges to meet to manage the current paradigm shift in technologies.

The remainder of this article is organized as follows.

In the next section, we shall recall the basic principles of probabilistic risk analyses, explain why the current process will probably change dramatically soon, and sketch the forthcoming process. In the third section, we shall present the model-based approach in probabilistic risk assessment and exemplified it with the S2ML+X modeling technology. In the fourth section, we shall discuss challenges to meet. Finally, the fifth section concludes the article.

The probabilistic risk assessment process

Current process

The current probabilistic risk assessment process is described in Figure 1.

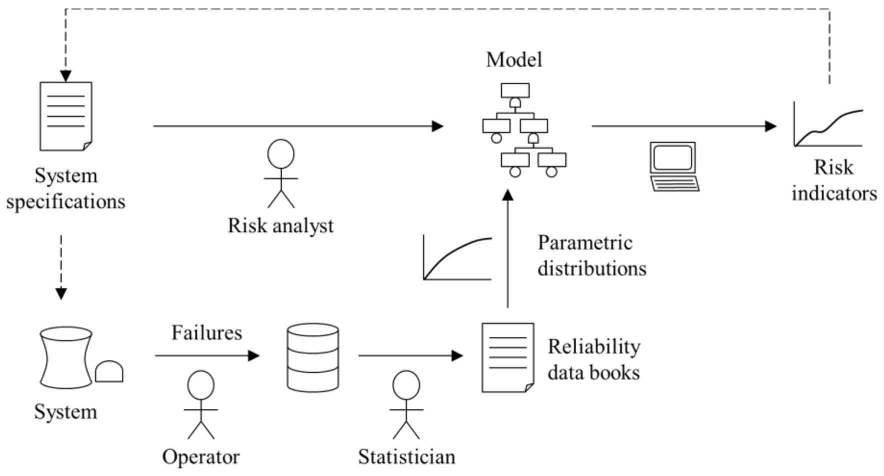

Figure 1. The (current) probabilistic risk assessment process

The risk analyst uses two kinds of prior knowledge to perform probabilistic risk assessment: the specifications of the system under study, so to understand how the system works and how it may fail, and reliability data for the components of the system, typically those recorded in the OREDA book [11] for the oil and gas industry. From this knowledge, the analyst designs a model, e.g. a fault tree. Then, he calculates indicators of operational performance such as the availability of the system, its reliability, its mean down time, its average production and so on.

These indicators are eventually used to make decisions about the design of the system.

Game changers

The above process will necessarily change, probably sooner than most of us expect, for at least three reasons.

First, systems designed by industry are increasingly complex, due notably to the massive introduction of software and ubiquitous control mechanisms. We are gradually moving from mechanical systems to mechatronic systems, cyber-physical systems and even systems of systems. As we shall show in the fourth section, fault trees and related modeling formalisms are not suitable to represent accurately the dynamic aspects of the behavior of these systems.

Second, we are quickly moving from a situation where reliability data are scarce and difficult to access to a situation where data are over numerous and easy to access. This will induce considerable changes in the probabilistic risk assessment process, although it is admittedly still hard to see the premises of this (r)evolution.

We ask here the reader to consider again the current situation, as it is represented Figure 1: reliability data are manually collected by operators, then aggregated by experts (with high skills in statistics) who try to fit them into parametric distributions such as the negative exponential distribution or the Weibull distribution. The parameters of these distributions are then recorded into books like the already mentioned OREDA [11]. Risk analysts pick up eventually data in these books to feed their models.

This way of doing things was fine but looks now completely outdated.

First, manual processing of data will no longer be possible when these data will come from sensors continuously monitoring systems.

Second, relying on books sounds weird at a time where billions and billions of digital data circulate on internet every second. Therefore, modeling environments should be soon directly connected to data bases.

Third, the main reason to use parametric distributions was they provide a compact way to store the information. However, the smallest image posted on social networks contains more information than required to describe any empirical probability distribution. Therefore, why going on using parametric distributions and not directly source data? Using directly source data would produce more accurate results. It could also make it possible to update data if not in real time, at least much more often than currently. Moreover, different treatments could be performed on data depending on the needs of the analysis.

In a word, reliability data that are used in probabilistic risk assessment are currently obtained via intermediations that will probably disappear in a near future.

The third game changer is also linked to the digital transformation of industrial processes: any complex system comes now with hundred, if not thousands, of models and data sets. These models and data sets constitute what is sometimes called the “digital twin” of the system [13].

We entered definitely in the model-based systems engineering era. Models are used not only to design systems, but also to operate and even to decommission them. This has two consequences: first, one needs to update models much more frequently than before. Second, one needs to ensure the coherence of the various models designed by the different engineering disciplines. With both respect, modeling formalisms like fault trees, block diagrams or event trees are not well suited. Validating and updating these models is an extremely hard task because of their cognitive distance to systems specifications. Concretely, it is nearly impossible to understand how the system works from the fault tree describing how it may fail. This calls for a new generation of modeling formalisms making it possible to reduce the gap between (model-based) systems specifications and risk assessment models.

Envisioned process

The emerging risk assessment process, as we envision it, is pictured in Figure 2. As the reader can see, it presents significant difference with the process described in Figure 1.

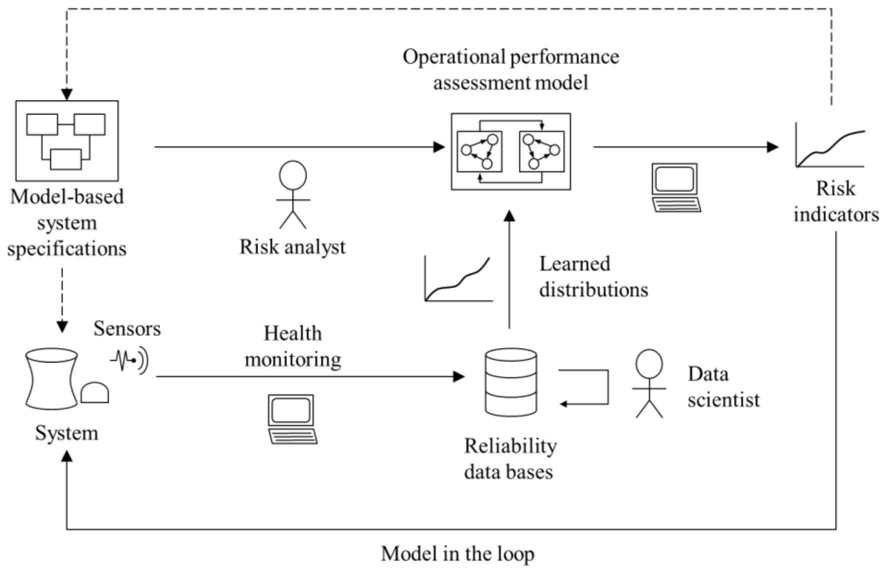

Figure 2. Envisioned probabilistic risk assessment process

Systems specifications from which the analyst derives the risk assessment model will rely more and more on models, as opposed to documents. Means should thus be put in place to synchronize system architecture models with risk assessment models. We shall come back to that point in the fourth section.

Manual recording of failures will be progressively replaced by automated health monitoring of systems, by means of sensors. Monitoring data will be stored in data bases. Data analysts will use artificial intelligence and machine learning techniques to extract from these data some learned degradation indicators and probability distributions of failure of components. These indicators will be integrated directly into models via digital communications.

Risk assessment models will also evolve so to be able to represent faithfully the behaviors of systems, which will much more dynamic that those of purely mechanical systems.

Finally, models will be used on-line to make decision about systems operations and not off-line in the design phase. In a word, models will be "in the loop".

Bets on evolution of technologies are usually lost. The future probabilistic risk assessment process will thus probably not look to what we described above. Nevertheless, we are convinced that there are there elements of the future process and in any case issues that are worth to study.

Discussion

The process described in the previous section relies heavily on models designed by the analyst. In an even more futuristic vision, one could imagine performing risk analyses straight from systems specifications and health monitoring data, by means of artificial intelligence techniques. The author believes that such a dream (for managers at least) has little chance to become reality, if any.

There are at least two major reasons to support our disbelief. First, artificial intelligence techniques require large training sets. As Yann LeCun, chief scientist at Facebook and one of the gurus of deep learning, keeps repeating the progresses made in artificial intelligence in the recent past years come for a good part of the availability of very large training sets [14]. But incidents and accidents are hopefully rare. Therefore, even though there are enough data to feed handmade models, there are not enough to get rid of these models. Second, artificial intelligence techniques, as any other computer tool, are efficient on well-defined problems. But precisely, the process of designing a model is the process by which the analyst makes the problem well-defined.

We shall now clarify what we mean by model-based approach in probabilistic risk assessment as it is an essential ingredient of the envisioned process.

Model-Based Safety Assessment

The promise of model-based risk assessment

In systems engineering, the model-based approach is defined as opposed to the document-based approach [15]. The situation is indeed different for probabilistic risk analysis that relies in essence on models. Rather, the model-based approach in reliability engineering is characterized the type of models that are used.

The most widely used modeling formalisms for safety analyses lack either expressiveness, e.g., fault trees and event trees, or structure, e.g., Markov chains and stochastic Petri nets. Consequently, they are far from systems specifications. These deficiencies make the models hard to design, hard to share with stakeholders, and even more importantly, hard to maintain through the entire lifecycle of systems.

As an illustration, consider for instance the small system pictured in **Error! Reference source not found.** that we shall use throughout this section.

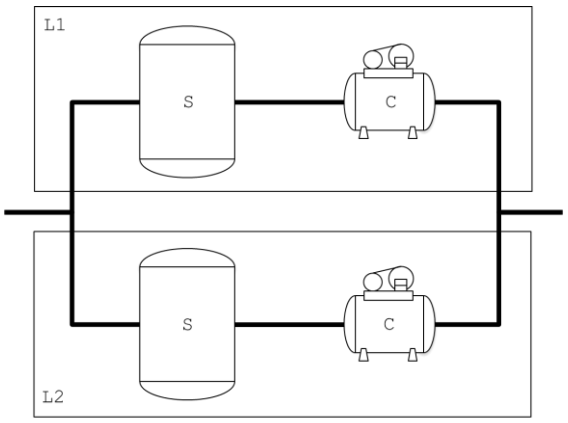

Figure 3. A two-line separation system

This system is made of two lines (`L1` and `L2`). Each line consists itself of a separator `S` and a compressor `C`. The system is working if at least one of the two lines is working. A line is working if both its separator and its compressor are working. **Error! Reference source not found.** shows a minimal fault tree describing the possible failures of this system.

This model makes a number of implicit assumptions: the two lines are assumed to be in hot redundancy, the capacities of unit are assumed to be either 100% or 0% and so on. Perhaps more importantly, it is nearly impossible from such a fault tree to retrieve the actual architecture of the system. If failures of separators and compressors are further decomposed, the analyst must duplicate by hand the descriptions of these failure conditions, which is both tedious and error prone, not to speak about maintenance (of the mode) issues.

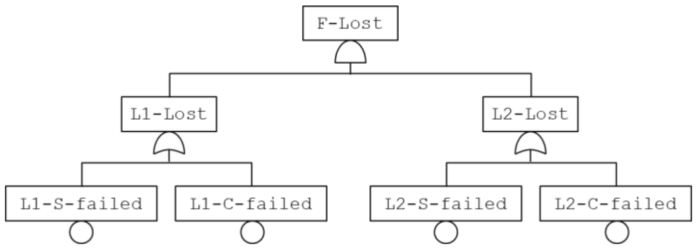

Figure 4. A minimal fault tree describing failures of the system pictured ***Error! Reference source not found.****.*

Modeling systems in a more structured way and with suitable mathematical frameworks can reduce the distance between systems specifications and models, without increasing the complexity of calculations. This is the promise of the so-called model-based risk assessment. This approach affords the ability to animate/simulate models, to ease their validation, and to share them with stakeholders. Moreover, it presents the following important benefits for risk analyses *stricto sensu*:

– A single model can address several safety goals, which eases versioning, configuration and change management;
– It can be assessed by several assessment tools, which increases versatility of assessments and quality-assurance of results (even if at a certain cost);
– It allows fine grain analyses, which limits over-pessimism resulting from coarse grain analyses as performed for instance with fault trees.
– Its maintenance is alleviated significantly, as it is closer to systems specifications.
– Similar formalisms can be used to design simple static models as well as dynamic models, hence facilitating the acquisition of competences and the industrial deployment of tools.
– The graphical animation of models makes it possible to share them with non-specialists.
– The same technology can be used not only for risk analyses but more generally to assess the operational performance of systems (in terms of costs, delays, production levels…).

Modeling formalisms that support this approach can be classified into three categories. The first category consists of specialized profiles of model-based systems engineering formalisms such as SysML, see e.g. [16]. The objective here is however more to introduce a safety facet into models of system architecture than to design actual safety models. The second category consists of extensions of fault trees or reliability block diagrams so to enrich their expressive power. This category includes dynamic fault trees [17][18], multistate systems [19][20][21], and some other proposals [22]. The third category, which aims at taking full advantage of the model-based approach, consists of modeling languages such as Figaro [23] or AltaRica [24].

We shall focus on the latter category, as we consider it as the most promising. More exactly we shall now present the S2ML+X family of modeling languages.

The S2ML+X Paradigm

Modeling languages of the S2ML+X family consists of two parts: a specific part, the X, which is a particular mathematical framework, e.g. guarded transition systems (GTS) [25][26] in the case of AltaRica 3.0, and a general part, S2ML. S2ML stands for S2ML stands for "system structure modeling language" [9]. S2ML gathers in a coherent way structuring constructs stemmed from object-oriented programming [27], and prototype-oriented programming [28]. In other words, languages of the S2ML+X family obeys the following equation, which echoes the title of the famous book of Wirth on the Pascal programming language ("Algorithms + Data-Structures = Programs") [29].

Behaviors + Architectures = Models

Our thesis is that it is possible to obtain full-fledged object-oriented modeling languages by putting S2ML on top of a core mathematical framework aiming at describing behaviors in a certain way.

This applies not only to guarded transition systems (X=GTS), which gives to AltaRica 3.0, but also for systems of stochastic Boolean equations (X=SBE), which are the underlying mathematical framework of fault trees and reliability block diagrams. This is actually what we have implemented in the new version of our tool XFTA, which probably the most powerful and efficient calculation engine in its class [30]. Beyond, it would be possible to apply the same principle to finite degradation structures [31] and even mathematical frameworks used outside of reliability engineering such as ordinary differential equations, obtaining in this way modeling languages similar to Matlab/Simulink [32] or Modelica [33].

S2ML in a Nutshell

At this point, it is probably time for us to provide the reader with more insights about S2ML.

Surprisingly enough, S2ML relies on only ten concepts: those of ports, connections, prototypes, classes, composition, cloning, instantiation, inheritance, reference and aggregation.

Ports are basic objects of a model. For instance, in S2ML+SBE, parameters of probability distribution, basic, intermediate and, house events, as well as common cause failure groups are ports.

Connections are relations, taken in a broad sense, that link ports. Connections capture the behavior of the system. For instance, in S2ML+SBE, equations defining parameters, basic, intermediate and house variables, as well as definitions of common cause group failures are connections.

Ports and connections suffice to create a model. For instance, the fault tree pictured Figure 4 is just a graphical representation of the following system of Boolean equations.

```
F-Loss = L1-Loss and L2-Loss
L1-Loss = L1-S-Failed or L1-C-Failed
L1-Loss = L1-S-Failed or L1-C-Failed
```

For the sake of simplicity, we let here aside the description of probability distributions associated with the four basic events `L1-S-Failed`, `L1-C-Failed`, `L2-S-Failed` and `L2-C-Failed`, but it can be encoded in a similar way. Writing such a set of equations, or equivalently drawing the fault tree, is easy when the system under study is small. However, a model made only of ports and connections reflect only very indirectly the architecture of the system under study, as discussed above.

To structure models, one needs *containers* for declarations of ports and connections (and other elements). The fundamental container is the *prototype*, i.e. a container with a unique occurrence in the model. In languages of the S2ML+X family, prototypes are called blocks.

When a container, a block, or any other type of container, contains an element, one says that the container *composes* this element. Composition is a fundamental relation between model elements, sometimes referred to as the *is-part-of* relation.

With ports, connections, and prototypes, it is already possible to design hierarchical models, i.e. to decompose the system under study into functional or physical subsystems, then these subsystems into sub-subsystems, and so one until the wanted degree of granularity is reached.

Such models would lack however of two fundamental ingredients. First, a way to represent that two elements of the model describe similar parts of the system. This is especially of interest in reliability engineering where redundancy is a key element to ensure the required level of performance (like in our example). Second, a way to connect elements located in different places in the hierarchy.

When two parts of the system under study are alike, e.g. the system pictured in Figure 3 is made of two identical lines, the description of the second line is the same as the description of the first one, up to the naming of elements. Once the first line described, the description of the second one can be obtained by a kind of copy-paste operation. This is however error prone and hides a fundamental information: the fact, precisely, that line 1 and line 2 are identical.

S2ML provides the concept of *cloning* to deal with such situations. Rather to copy-paste the prototype describing the first line to get the prototype describing the second one, one says that the second prototype is a clone of the first one. The assessment tool, XFTA in our case, is then in charge of performing the duplication.

For instance, the description of the system pictured in Figure 3 could have the following structure.

```
block System
   block Line1
      block Separator
         // description of the behavior of the separator
      end
      block Compressor
         // description of the behavior of the compressor
      end
   end
   clones Line1 as Line2;
end
```

Note that the above structure is independent of the mathematical framework chosen to describe behaviors.

Cloning makes it possible to duplicate modeling elements within a model, but not to reuse them from models to models. Moreover, considering the description of basic components, e.g. pumps or valves, the choice of the initial model element (from other similar model elements are obtained by cloning) is very arbitrary.

The idea is therefore to create libraries of on-the-shelf modeling elements, outside any particular model, and to clone these modeling elements into the model, when needed. In S2ML (and more generally in object-oriented programming), this is achieved by the concepts of classes and instances.

A *class* is just a prototype declared outside the model. *Instantiation* is the operation by which a class is cloned into a model. The resulting prototype is called an *instance* of the class.

For instance, we could define classes to describe the behavior of separators and compressors, then instantiate them into our model:

```
class Separator
  // description of the behavior of the separator
End

class Compressor
   // description of the behavior of the compressor
end

block System
   block Line1
      Separator S;
      Compressor C;
   end
   clones Line1 as Line2;
end
```

Note again that the above structure is independent of the mathematical framework chosen to describe behaviors.

Now it is sometimes the case that a component is a particular type of a more general category of components, e.g. a solenoid valve is a particular type of valve. Most of the properties of the particular component are actually common to all components of the category, while some are specific. To represent that, it would be indeed possible to create a class for generic components then to instantiate this class into the class describing specific ones. This would lead however to awkward modelse: a solenoid valve is not part of a generic one. Rather, a solenoid valve *is-a* valve.

In S2ML (and more generally in object-oriented programming), capturing is-a relations is achieved by means of *inheritance*. When a prototype or class inherits from another prototype or a class, it means that all elements composed by the latter are composed by the former. It is then possible to modify the definitions of these elements or to add new ones to reflect the particular properties of the specific component. E.g.

```
class Valve
  // description of the behavior of a generic valve
end

class SolenoidValve
   extends Valve;
   // description of the specific features of solenoid valves
end
```

The last ingredient we need to deploy fully object-oriented modeling is the possibility to refer to an element located somewhere in the hierarchy of prototypes from anywhere else in this hierarchy. The notion of *reference* is thus key. In S2ML, referring to ports is achieved by means of *paths*. Within a block, each element is uniquely identified with a name, called its *identifier*. Two elements cannot have the same name, even though they are of different types. To refer to an element located in other blocks, one uses paths built with the dot notation and the two primitives `main` and `owner`:

– `B.E` refers to the element `E` composed by the block `B` itself composed by the current block. Applying this principle recursively makes it possible to refer to any element located in the hierarchy rooted by the current block.
– `owner` refers to the parent block of the current block. Therefore, owner.`owner.B.E` refers to the element `E` composed by the block `B` itself composed by the grand-parent block of the current block. The primitive `owner` makes it possible to create relative paths referring to any element in the current hierarchy.
– `main` refers to the outermost block of the current hierarchy, i.e. the model itself. Therefore, `main.B.E` refers to the element `E` composed by the block `B` declared at the top-level. The primitive `main` makes it possible to create absolute paths referring to any element in the model.

For instance, assuming that the class `Separator` declares a variable out, that is true if and only if the separator works properly and that the class `Compressor` declares a variable `in` to reflect the flow upstream the compressor. Then, at line level we have to connect these two variables by means of an equation. This can be done as follows, using the dot notation.

```
block Line1
   Separator S;
   Compressor C;
   flow C.in = S.out;
end
```

There are cases where one needs to refer to not only an individual element, like a parameter or a variable, but a whole container, possibly itself composing subcontainers. In that case, using paths would be tedious, and error prone.

The solution consists in the last concepts provided by S2ML, namely the *aggregation* of containers.

Let `A` and `B` be two containers located at different places in the same hierarchy. Let `π.B` the path (relative or absolute) to go from `A` to `B` in that hierarchy. To access an element `E` composed by `B` from `A`, one must normally use the path `π.B.E`. By aggregating in `A` the container `B` (actually the container `π.B`) under the name `C`, one makes possible to `E` in `A` by means of the path `C.E`. In some sense, this creates the *alias* `C` for the path `π.B` in `A`.

Aggregation should not be seen however only as a technical solution to create references. More fundamentally, it represents a *uses* relation. `A` uses `B` although `B` is not declared in the vicinity of `A`.

Aggregation is a key tool to describe so-called functional chains [34] as well as to glue together, within the same model, descriptions of functional and physical architectures [35].

AltaRica 3.0

So far, we used on systems of stochastic Boolean equations, which have the expressive power of fault trees or reliability block diagrams. With AltaRica 3.0 [24], we leave the category of combinatorial modeling formalisms, to enter the category of state automata, see reference [11] for a discussion on these categories.

Due to space limitations, it is not possible to present here all the features of the language. In the previous section, we gave a flavor of S2ML. We shall thus illustrate here the expressive power of guarded transition systems [25][26] by means of an example.

Assume that, in our case study, the second line is a backup for the first one, i.e. that its separator and its compressor are put in operation on demand. Systems of stochastic Boolean equations are not powerful enough to represent faithfully this behavior (and more generally to take into account time dependencies).

Figure 5 shows the graphical representation of a guarded transition system representing a standby unit.

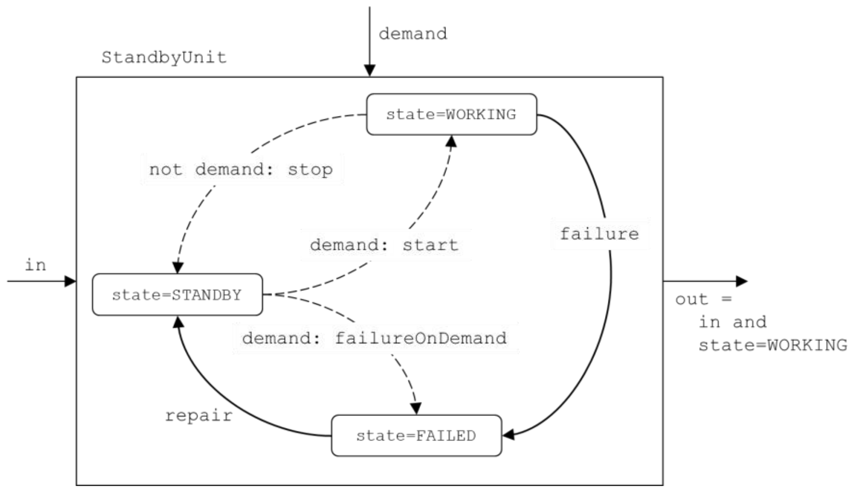

Figure 5. The guarded transition system representing a standby unit

Figure 6 shows the AltaRica code for this guarded transition system.

```
domain StandbyUnitDomain {STANDBY, WORKING, FAILED}

block MotorPump
   StandbyUnitDomain state (init = STANDBY);
   Boolean demand, in, out (reset = false);
   event start (delay = Dirac(0), expectation=gamma);
   event failureOnDemand (delay=Dirac(0),expectation=1-gamma);
   event stop (delay = Dirac(0));
   event failure (delay = exponential(lambda));
   event repair (delay = exponential(1/tau));
   parameter Real lambda = 1.0e-4;
   parameter Real tau = 8;
   parameter Real gamma = 0.02;
   transition
      start: demand and state==STANDBY -> state := WORKING;
      failureOnDemand: demand and state==STANDBY
         -> state := FAILED;
      stop: not demand and state==WORKING -> state := STANDBY;
      failure: state==WORKING -> state := FAILED;
      repair: state==FAILED -> state := STANDBY;
   assertion
      out := in and state==WORKING;
end
```

Figure 6. AltaRica code for the guarded transition system pictured Figure 5

Just as in systems of stochastic Boolean equations, guarded transition systems use two types of variables to represent the current state of the system under study: state variables that represent actually the state of the system and flow variables that represent flows of matters, energy of information circulating in the network of components.

The guarded transition system pictured in Figure 5 used one state variable, `state`, and three flow variables `demand`, `in` and `out`. In AltaRica 3.0, variables take their values into sets of constants called *domains*. The domain of the variable `state` is the set of three symbolic constants `{STANDBY, WORKING, FAILED}`. The three flow variables are Boolean.

The value of flow variables is calculated from the value of state variables, which means that the former is recomputed each time the former is modified.

The value of state variables changes under the occurrence of *events*. In AltaRica, these changes are described by means of guarded transitions. A *guarded transition* is a triple (event, guard, action). The *guard* of a transition is a Boolean condition telling when the transition is enabled. The *action* of a transition is the way this transition modifies the value of state variables, when fired.

In our example, there are five transitions labeled respectively by the events `start`, `stop`, `failureOnDemand`, `failure` and `repair` and represented by arrows.

Events are associated with probability distributions.

In our example, transitions labeled with events `start`, `stop`, and `failure-OnDemand` are deterministic and instantaneous (associated with Dirac distributions), while transitions labeled with events `failure` and `repair` are timed and stochastic.

The combination of GTS and S2ML results in a powerful, versatile language which exploits optimally assessment algorithms.

An integrated modeling environment for AltaRica 3.0 (AltaRica Wizard) has been developed as join effort of the Open-AltaRica team at IRT-SystemX (Paris, France) and the author at NTNU. Industrial partners (Airbus, Safran and Thalès) support this project. A versatile set of assessment tools is under development, which includes:

– A step by step simulator making possible to play "what-if" scenarios and to validate models. This simulator implements abstract interpretation techniques so to simulate faithfully stochastic and timed executions [36].
– A compiler of AltaRica models into fault trees. This compiler relies on advanced algorithmic techniques [37]. Fault trees are then assessed with XFTA [30], which is one of the most efficient available calculation engines.
– A compiler of AltaRica models into Markov chains. This compiler produces Markov chains that approximate the original model while staying of reasonable sizes [38]. Markov chains are then assessed with Mark-XPR, as very efficient calculation engine [39].
– A generator of critical sequences.
– A stochastic simulator. Stochastic simulation is itself a versatile tool to assess complex models [40].

These tools make the AltaRica 3.0 technology extremely efficient. They make it possible cross-verification. They prefigure what will be the next generation of modeling environments for the assessment of operational performance of complex technical systems.

Textual versus graphical representations

As all modeling languages of the S2ML+X family, S2ML+SPBE and AltaRica 3.0 are a primarily textual, just as computer programs. Graphical representations can be used, but the ultimate reference is the text. Not only we do not consider that as a drawback, but we claim it is a necessity. At first, this thesis may seem at best extremely provocative, as most of the models designed for both system architecture and risk analyses (as well as in other engineering disciplines) are authored via graphical modeling environments and many practitioners just refuse to write a single line of code. However, graphical modeling is mainly useful to describe structural parts of models and systems, see e.g. reference [41] for an interesting discussion on the pragmatics of graphical modeling. It is hard to conceive how to author a differential equation or the probability distribution of the basic event of a fault tree graphically. Behavioral descriptions, such as Markov chains or Petri nets, can be represented graphically. However, as soon as models become large, which is the case for nearly any industrial-scale system, their graphical representations become more problematic than useful: as they cannot fit into any reasonable space (computer screen or printed out paper), the analyst can only visualize them by parts. This means that she or he must anyway develop a global cognitive model to understand local graphical representations. Moreover, many subtle differences in behaviors are just impossible to represent graphically. In a word, models exist independently of their graphical representations. These graphical representations, even taken together, cannot fully describe the model, except in simple cases. It is often very convenient to have several partial graphical representations for the same information and to extract dynamically graphical representations according to one's needs.

The parallel with software engineering is here fruitful. It is useful to represent the architecture of software by diagrams such those of UML [42]. However, the software exists independently of these representations and the code is the ultimate reference. Moreover, below a certain level of abstraction, the code gives a more compact, more precise, in a word more useful, information than any drawing. At the end of the day, the humanity invented writing to overcome the lack of precision of drawing.

It remains that making textual models adopted is one of the challenges that we must meet. We shall now discuss these challenges.

Challenges

Transforming big data into smart data

Sensors produce already lots of data (big data) and will produce even more in the future. However, most of these data cannot be exploited for probabilistic risk assessment. Therefore, a key question is how to collect data that can be translated into (probabilistic) degradation models, in complement or not to physical models. If we can do so, it will remain to introduce degradation models for components into risk assessment models for systems, i.e. to accommodate them into stochastic discrete events systems. This latter point does not seem a major technical or scientific issue. We can be reasonably optimistic on this question.

Handling the increasing complexity of systems

The behavior of software intensive systems, often called mechatronic systems, or cyber-physical systems if they are connected to the net, is highly dynamic. Control mechanisms change the configuration of these systems depending on the state of their components, of the environment or on the needs in terms of production. Condition-based maintenance policies, which are increasingly adopted for the sake of reducing costs of maintenance interventions and reducing production down-times due to these interventions, enter in this category. The introduction these control mechanisms creates dependencies among components as well as dynamically scheduled phases in the life cycle of systems. Maintenance interventions are not scheduled once for all, on a calendar basis, but decided dynamically based on the monitoring of the condition of the system, which in turn depends on maintenance interventions.

Static models, such as fault trees, event trees or reliability block diagrams, cannot represent faithfully these dependencies and dynamically scheduled phases of life cycles. To be able to do so, one needs at least the expressive power of (stochastic) discrete event systems, like AltaRica.

Moving from static models to discrete event systems has however a triple cost: first, analysts should be trained to these new modeling technologies; second, the computational cost of calculations of risk indicators increases significantly; third, as modeling formalisms are more powerful and problems at stake are more complex, models are more difficult to design and to validate. We shall discuss the two first point latter in this section. The third one, model design and validation of complex technical systems, is one of the major technical challenges we are facing. It is striking how, as of today, the reliability engineering literature is still silent of this issue. It is like modeling was a subsidiary task, requiring no other competences and skills than a good mathematical background and a solid practical knowledge of the systems under study. Nothing is more illusory. Models must be recognized as first-class citizens of scientific research in our domain. We need to develop the science and the engineering of models (of engineering). With that respect, much can be learned from the historical development of computer science and software engineering.

As explained in the previous section, AltaRica 3.0 embeds already the most advanced concepts for structuring models. These concepts are stemmed from object-oriented programming and prototype-oriented programming [27][28]. Relying on a power mathematical framework and versatile structuring mechanisms is mandatory to handle the problems at stake. It is however not sufficient. To make the modeling process efficient both in terms of model design and model validation, it is of primary importance to reuse as much as possible modeling components within models and between models. In modeling languages such as Modelica [33], this goal is achieved via the design of libraries of on-the-shelf ready-to-use modeling components. Reusing components is also possible in probabilistic risk analyses, but to a much lesser extent. The reason is that these analyses represent systems at a high level of abstraction. Modeling components, except for very basic ones, tend thus to be specific to each system. Reuse is mostly achieved by the design of modeling patterns, i.e. examples of models representing remarkable features of the system under study. Once identified, patterns can be duplicated and adjusted for specific needs [24]. The notion of patterns is pervasive in systems engineering. For instance, it has been developed in the field of technical system architecture, see e.g. [43], as well as in software engineering [44]. Patterns are also excellent communication mean: in order to document models (or programs), it is often sufficient to refer to the patterns that have been used to design them. The author strongly believes that one of the tasks of the reliability engineering community should be to perform a systematic exploration of the modeling patterns for probabilistic risk assessment of nowadays technical systems. It is probably the only way to tame the complexity of these systems.

Computational complexity of probabilistic risk assessment

The risk analyst must face the combinatorial explosion of the number of scenarios to analyse. Whatever modeling technology is used, the calculation of probabilistic risk indicators is provably computationally hard, namely #P-hard, as demonstrated by Valiant [45] and further completed by Toda [46]. This was already true for mechanical systems; this is indeed even more sensitive for mechatronic systems.

During the last decades fantastic progresses have been made in the development of efficient algorithms and heuristics for probabilistic risk assessment. The power of computers has also dramatically increased. No doubt that more progresses will be made in both directions in the future. Nevertheless, the above mathematical limits will continue to apply. The risk analyst will always have limited calculation capacities at hand. In practice, this means that probabilistic risk assessment models result necessarily of a trade-off between the accuracy of the description of the system under study and the ability to perform calculations on this description. In other words, the risk analyst faces the fundamental epistemic and aleatory uncertainties of risk assessment with a bounded calculation capacity. This bounded capacity overdetermines both the design of models and the decisions that can be made from models, see reference [11] for an in-depth discussion on this topic. With that respect, he or she is like Simon's economical agent who must make decisions with a bounded rationality [47].

The scientific and technological question at stake here is therefore to work on algorithms, heuristics and modeling methodologies that help to use as efficiently as possible the calculation resources at hand.

At this point, we must say few words about probabilistic risk analyses of systems of systems, which are increasingly present in industry and more generally in our lives [48]. These systems are very different from mechanical, mechatronic and even cyber-physical systems. We can characterize them as being:

– Opaque: their states can be observed only by indirect means;
– Reflective: they embody models of their own behavior and environment;
– Deformable: their architecture changes throughout their mission.

Clearly, even modeling technologies like AltaRica 3.0 are not suited to represent systems having these properties as they assume a fixed architecture of the system under study is fixed [48]. To represent the behaviors of these systems of systems, another class of modeling frameworks is probably required that we can called stochastic process algebras in reference [11]. This class includes formalisms as diverse as (stochastic variants of) colored Petri nets (with an unbound number of colors) [50], process algebras such as Milner's pi-calculus [51] and agent-oriented modeling languages [52]. These formalisms are extremely powerful. They have however a major drawback: most of the questions we may ask are undecidable [53]. Consequently, we must forge new concepts to analyze these systems.

Integrating seamlessly models and data sets into the digital twin

To face the complexity of technical systems, the engineering disciplines contributing to the design and operations of these systems are designing models and collecting engineering data: as already said, any technical system comes now with hundred, if not thousands, of models and data sets. These models are designed by different teams in different modeling formalisms, at different levels of abstraction, for different purposes. Models mature also at different rates. The question is thus how to ensure that they describe the same system, i.e. how to synchronize them.

There are at least four distinct aspects in this question: a first one concerns the management of models and data sets in the context of the extended enterprise. This is the realm of collaborative data bases, product life cycle and product data management environments [54]. The concept of "digital twin" is gaining popularity to designate systems in charge of models (and engineering data) management [55]. It impacts all engineering disciplines, including of course probabilistic risk assessment, as collaborative data bases will provide the infrastructure for the system analysis and modeling processes.

A second aspect is related to the seamless cooperation of models of different abstraction levels within a discipline. This is an important and difficult topic [56]. This aspect concerns also probabilistic risk analyses. The question here is how models designed by a client and its suppliers can cooperate. A mere integration cannot be the answer for both intellectual property and computational complexity issues. Mathematical concepts and algorithmic tools must be developed this purpose.

A third aspect regards the co-simulation of heterogeneous but compatible models, such as experiments performed in the framework of the Ptolemy project [57]. For probabilistic risk analyses, it would mean for instance to couple risk assessment models with 3D physical simulation codes. This would be of interest, especially in terms of communication with the stakeholders. Regarding calculation of risk indicators, it is probably quickly limited by computational complexity issues.

The fourth aspect regards the alignment of heterogeneous models representing the system at about the same level of abstraction. The alignment of system architecture models and probabilistic risk assessment models is a paradigmatic example of that. This alignment is an industrial necessity and is required by Safety Standard such as IEC 61508 [58] and IEC 61511 [59]. The heterogeneity of these models makes it impossible to compare them directly. To compare them, we first must abstract them into a common language, and then perform a comparison of their abstractions. Once the comparison has been made, it is possible to go back to original models via a concretization mechanism. This principle is close to Cousot's abstract interpretation of programs [60]. Significant results have been obtained in this direction that show the interest of this approach [61][62][63].

Managing the change

The technological transformations we discussed in this article cannot be achieved without a conscious, organized and systematic management of change.

To start with, it requires to solve numerous intellectual property issues: who is the owner of the data, who can access to them, under which conditions and so on. This problematic goes indeed well beyond probabilistic risk analyses. It concerns actually the whole digital twin in the context of the extended enterprise.

Training risk analysts to new modeling technologies is also a major issue. In now more than twenty years of experience in both academia and industry, the author knows perfectly how hard it is to pull well trained, experienced experts out of their comfort zone. Risk analysts are conservative so to say in essence: you must have very good reasons to change a solution that worked so far. But reasons for a radical change are here. Here again, we learn lessons from the historical development of computer science and software engineering: new programming paradigms have been progressively introduced in industry by new generations of engineers who learned them at university. Nowadays students are not afraid to write computer code. On the contrary: to attract the best students, we should propose them state-of-the-art activities and competences. One of the author's deepest convictions is that much more discrete mathematics – see e.g. reference [64] for an introduction – should be introduced in engineering curricula.

Conclusion

In this article, we discussed the impact of current technological transformations on probabilistic risk analyses. We advocated that two major changes in the probabilistic risk assessment process are foreseeable. First, in-book reliability data collected and organized by statisticians will be replaced by databases of degradation indicators obtained machine learning techniques ran by data scientists. Second, classical modeling formalisms such as fault trees, event trees or reliability blocks diagrams will be replaced by modeling formalisms supporting the model-based approach, as exemplified by XFTA (S2ML+SBE) or AltaRica 3.0.

The industrial deployment of such radical changes will take time and is by no means certain. However, there are solid scientific and technological arguments to support them. The author hopes that the present article will at least serve to open the discussion on the future of probabilistic risk analyses and will contribute to create fruitful exchanges between academia and industry.

References

[1] Zio E. (2009) Reliability engineering: Old problems and new challenges. Reliability Engineering and System Safety; 94: 125–141.

[2] Zio E and Aven T. (2013) Industrial disasters: Extreme events, extremely rare. some reflections on the treatment of uncertainties in the assessment of the associated risks. Process Safety and Environmental Protection; 91: 31–45. doi:10.1016/j.psep.2012.01.004.

[3] Aven T, Baraldi P, Flage R et al. (2014) Uncertainty in Risk Assessment: The Representation and Treatment of Uncertainties by Probabilistic and Non-Probabilistic Methods. Chichester, West Sussex, United Kingdom: Wiley-Blackwell. ISBN 978-1118489581.

[4] Aven T. (2015) The concept of antifragility and its implications for the practice of risk analysis. Risk Analysis; 35(3): 476–483. doi:10.1111/risa.12279

[5] Rasmussen N. C. (1975) Reactor Safety Study. An Assessment of Accident Risks in U.S. Commercial Nuclear Power Plants. U.S. Nuclear Regulatory Commission. Rockville, MD, USA. WASH 1400, NUREG-75/014. October.

[6] Andrews JD and Moss RT. (2002) Reliability and Risk Assessment (second edition). Materials Park, Ohio 44073-0002, USA: ASM International. ISBN 978-0791801833.

[7] Kumamoto H and Henley EJ. (1996) Probabilistic Risk Assessment and Management for Engineers and Scientists. Piscataway, N.J., USA: IEEE Press. ISBN 978-0780360174.

[8] Rauzy A and Haskins C. (2018) Foundations for Model-Based Systems Engineering and Model-Based Safety Assessment. Journal of Systems Engineering. Wiley Online Library. doi:10.1002/sys.21469.

[9] Batteux M, Prosvirnova T and Rauzy A. (2018) From Models of Structures to Structures of Models. IEEE International Symposium on Systems Engineering (ISSE 2018). IEEE. Roma, Italy. October. doi:10.1109/SysEng.2018.8544424. Best paper award.

[10] Brooks F. (1995 The Mythical Man-Month. New York, NY, USA: Addison-Wesley. ISBN 0-201-83595-9.
[11] Rauzy A. (2018) Notes on Computational Uncertainties in Probabilistic Risk/Safety Assessment. Entropy. MDPI. 20:3. doi:10.3390/e20030162.
[12] Oreda handbook – offshore reliability data, (2015) volume 1 and 2, 6th edition.
[13] Datta S., Emergence of Digital Twins, DSpace@MIT, https://dspace.mit.edu/handle/1721.1/104429, 015
[14] Lecun Y, L'apprentissage profond, Leçons inaugurales au Collège de France (2017) Fayard, ISBN 978-2213701820 (in French)
[15] Holt J. and Perry S. (2013) SysML for Systems Engineering: A Model-Based Approach. Institution of Engineering and Technology. Stevenage Herts, United Kingdom. ISBN 978-1849196512.
[16] Yakymets N., Munoz Julho Y. and Lanusse A. (2014) Sophia framework for model-based safety analysis. Actes du congrès Lambda-Mu 19 (actes électroniques). Institut pour la Maîtrise des Risques. ISBN 978-2-35147-037-4. Dijon, France.
[17] Dugan J.B, Bavuso S. J. and Boyd M. A. (1992) Dynamic fault-tree models for fault-tolerant computer systems. IEEE Transactions on Reliability. IEEE. 41:3. pp. 363–377. September. doi:10.1109/24.159800.
[18] Bouissou M. and Bon J.-L. (2003) A new formalism that combines advantages of Fault-Trees and Markov models: Boolean logic-Driven Markov Processes. Reliability Engineering and System Safety. Elsevier. 82:2. pp. 149–163. doi:10.1016/S0951-8320(03)00143-1.
[19] Lisnianski A. and Levitin G. (2003). Multi-State System Reliability. World Scientific. London, England. ISBN 981-238-306-9.
[20] Papadopoulos Y., Martin M., Parker D., Rüde E., Hamann R., Uhlig A., Grätz U. and Liend R. (2011). An approach to optimization of fault tolerant architectures using HiP-HOPS. Journal of Engineering Failure Analysis. Elsevier Science. 18:2. pp. 590–608. March. doi:10.1016/j.engfailanal.2010.09.025.
[21] Zaitseva E and Levashenko V. (2017) Reliability analysis of multi-state system with application of multiple-valued logic International Journal of Quality and reliability Management, Emerald Publishing 34:6, pp 862-878, doi 10.1108/IJQRM-06-2016-0081.
[22] Signoret J.-P., Dutuit Y., Cacheux J.-P., Folleau C., Collas S. and Thomas P. (2013) Make your Petri nets understandable: Reliability block diagrams driven Petri nets. Reliability Engineering and System Safety. Elsevier. 113. pp. 61–75. doi:10.1016/j.ress.2012.12.008
[23] Bouissou M., Bouhadana H., Bannelier M. and Villatte N. (1991) Knowledge modeling and reliability processing: presentation of the FIGARO language and of associated tools. Proceedings of SAFECOMP'91, IFAC International Conference on Safety of Computer Control Systems. Johan F. Lindeberg Ed.. Pergamon Press. ISBN 0-08-041697-7. pp. 69–75. Trondheim, Norway
[24] Batteux M, Prosvirnova T and Rauzy A. (2019) AltaRica 3.0 in 10 Modeling Patterns. International Journal of Critical Computer-Based Systems. Inderscience Publishers. 9:1-2. pp. 133–165. doi:10.1504/IJCCBS.2019.098809.
[25] Rauzy A. (2008) Guarded Transition Systems: a new States/Events Formalism for Reliability Studies. Journal of Risk and Reliability. Professional Engineering Publishing. 222:4. pp. 495–505. doi:10.1243/1748006XJRR177.
[26] Batteux M., Prosvirnova T. and Rauzy A. (2017) AltaRica 3.0 Assertions: the Why and the Wherefore. Journal of Risk and Reliability. Professional Engineering Publishing. September. doi:10.1177/1748006X17728209.
[27] Abadi M. and Cardelli L. (1998) A Theory of Objects. Springer-Verlag. New-York, USA. ISBN 978-0387947754.

[28] Noble J., Taivalsaari A. and Moore I. (1999) Prototype-Based Programming: Concepts, Languages and Applications. Springer-Verlag. Berlin and Heidelberg, Germany. ISBN 978-9814021258.

[29] Wirth N. (1976) Algorithms + Data Structures = Programs. Prentice-Hall. Upper Saddle River, New Jersey 07458, USA. ISBN 978-0130224187.

[30] Rauzy A. (2020) Probabilistic Safety Analysis with XFTA. AltaRica Association. Les Essarts le Roi, France. ISBN 978-82-692273-0-7.

[31] Rauzy A. and Yang L. (2019) Finite Degradation Structures. Journal of Applied Logics -- IfCoLog Journal of Logics and their Applications. College Publications. 6:7. pp. 1471–1495.

[32] Klee H. and Allen R. (2011). Simulation of Dynamic Systems with MATLAB and Simulink. CRC Press. Boca Raton, FL 33431, USA. ISBN 978-1439836736.

[33] Fritzson P. (2015) Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach. Wiley-IEEE Press. Hoboken, NJ 07030-5774, USA. ISBN 978-1118859124.

[34] Voirin J.-L.. (2008) Method and Tools for Constrained System Architecting. Proceedings 18th Annual International Symposium of the International Council on Systems Engineering (INCOSE 2008). Curran Associates, Inc.. ISBN 978-1605604473. pp. 775–789. Utrecht, The Netherlands.

[35] Batteux M., Prosvirnova T., Rauzy A. and Yang L. (2018) Reliability Assessment of Phased-Mission Systems with AltaRica 3.0. Proceedings of the 3rd International Conference on System Reliability and Safety (ICSRS). IEEE. pp. 400–407. Barcelona, Spain. November. doi:10.1109/ICSRS.2018.00072.

[36] Batteux M., Prosvirnova T., Rauzy A. (2020) Abstract Executions of Stochastic Discrete Event Systems, submitted to SoSym.

[37] Prosvirnova T. and Rauzy A. (2015) Automated generation of Minimal Cutsets from AltaRica 3.0 models. International Journal of Critical Computer-Based Systems. Inderscience Publishers. 6:1. pp. 50–79. 2015 doi:10.1504/IJCCBS.2015.068852.

[38] Brameret P.-A., Rauzy A. and Roussel J.-M. (2015) Automated generation of partial Markov chain from high level descriptions. Reliability Engineering and System Safety. Elsevier. 139. pp. 179–187. doi:10.1016/j.ress.2015.02.009.

[39] Rauzy A. (2004) An Experimental Study on Six Algorithms to Compute Transient Solutions of Large Markov Systems. Reliability Engineering and System Safety. Elsevier. 86:1. pp. 105–115.

[40] Zio E. (2013) The Monte Carlo Simulation Method for System Reliability and Risk Analysis. Springer London. London, England. ISBN 978-1-4471-4587-5.

[41] Fuhrmann H.A. L. (2011) On the Pragmatics of Graphical Modeling. Book on Demand. Norderstedt, Germany. ISBN 978-384480084.

[42] Rumbaugh J., Jacobson I. and Booch G. (2005) The Unified Modeling Language Reference Manual. Addison Wesley. Boston, MA 02116, USA. ISBN 978-0321267979.

[43] Maier M. W. (2009) The Art of Systems Architecting. CRC Press. Boca Raton, FL 33431, USA.

[44] Gamma E., Helm R., Johnson R. and Vlissides J. (1994) Design Patterns -- Elements of Reusable Object-Oriented Software. Addison-Wesley. Boston, MA 02116, USA. ISBN 978-0201633610.

[45] Valiant L. G. (1979) The Complexity of Enumeration and Reliability Problems. SIAM Journal of Computing. SIAM. 8:3. pp. 410–421.

[46] Toda S. (1991) PP is as Hard as the Polynomial-Time Hierarchy. SIAM Journal on Computing. SIAM. 20:5. pp. 865–877.

[47] Simon H. (1957) Models of Man: Social and Rational. Mathematical Essays on Rational Behavior in a Social Setting. Wiley. New York, New Jersey, U.S.A.

[48] Maier M. W. (1998) Architecting principles for systems-of-systems. Systems Engineering. Wiley Periodicals, Inc. 1:4. pp. 267-284. doi:10.1002/j.2334-5837.1996.tb02054.x.

[49] Kloul L., Prosvirnova T. and Rauzy A. (2013) Modeling systems with mobile components: a comparison between AltaRica and PEPA nets. Journal of Risk and Reliability. Professional Engineering Publishing. 227:6. pp. 599–613. doi:10.1177/1748006X13490497.

[50] Jensen K. (2014) Coloured Petri Nets. Springer-Verlag. Berlin and Heidelberg, Germany. ISBN ISBN-10: 364242581X. ISBN-13: 978-3642425813.

[51] Milner R. (1999) Communicating and Mobile Systems: The pi-calculus. Cambridge University Press. Cambridge, CB2 8BS, United Kingdom. ISBN 978-0521658690.

[52] Railsback S. and Grimm V. (2011) Agent-Based and Individual-Based Modeling - A Practical Introduction. Princeton University Press. Princeton, New Jersey, USA. ISBN 978-0691136745.

[53] Esperza J. (1998) Decidability and Complexity of Petri Nets Problems - An introduction. Lectures on Petri Nets I: Basic Models. W. Reisig and G. Rozenberg Ed.. Springer. ISBN 3-540-65306-6. 1491. pp. 374–428.

[54] Stark J. (2011) Product Lifecycle Management: 21st Century Paradigm for Product Realisation (2nd ed). Springer London Ltd. London, England. ISBN 978-0857295453.

[55] Datta S. (2015). Emergence of Digital Twins. https://dspace.mit.edu/handle/1721.1/104429

[56] Mainini L. and Maggiore P. (2012) Multidisciplinary Integrated Framework for the Optimal Design of a Jet Aircraft Wing. International Journal of Aerospace Engineering. Hindawi Publishing Corporation. doi:10.1155/2012/750642.

[57] Ptolemaeus C. (2014) System Design, Modeling, and Simulation using Ptolemy II. Ptolemy.org. ISBN 978-130442106. http://ptolemy.org/books/Systems.

[58] IEC (2010) International IEC Standard IEC61508 - Functional Safety of Electrical/Electronic/Programmable Safety-related Systems (E/E/PE, or E/E/PES). International Electrotechnical Commission. Geneva, Switzerland. ISBN ISBN 978-2-88910-524-3.

[59] IEC (2016) International IEC Standard IEC61511 - Functional safety - Safety instrumented systems for the process industry sector. International Electrotechnical Commission. Geneva, Switzerland. ISBN 978-2-8322-4752-5.

[60] Cousot P. and Cousot R. (1977) Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. ACM Press. pp. 238–252. New York, NY, USA.

[61] Legendre A., Lanusse A. and Rauzy A. (2016) Directions towards supporting synergies between design and probabilistic Safety assessment activities: illustration on a fire detection system embedded in a helicopter. Proceedings PSAM'13. IPSAM. Seoul, South-Korea.

[62] Batteux M., Prosvirnova T., Rauzy A. (2019) Model Synchronization: A Formal Framework for the Management of Heterogeneous Models. Model-Based Safety and Assessment. Yiannis Papadopoulos, Koorosh Aslansefat, Panagiotis Katsaros and Marco Bozzano Ed.. Springer. ISBN 978-3-030-32871-9. 11842. pp. 157–172. Thessaloniki, Greece.

[63] Batteux M., Choley J.-Y., Mhenni F., Prosvirnova T. and Rauzy A. (2019). Synchronization of system architecture and safety models: a proof of concept. Proceedings of the IEEE 2019 International Symposium on Systems Engineering (ISSE). IEEE. Edinburgh, Scotland.

[64] O'Regan G. (2016) Guide to Discrete Mathematics: An Accessible Introduction to the History, Theory, Logic and Applications. Springer International Publishing AG. Cham, Switzerland. ISBN ISBN 978-3319445601.