Flexibility of Analysis Through Knowledge Bases

Ola Bäckström

Lloyd's Register, Sweden. E-mail: Ola.Backstrom@lr.org

Marc Bouissou

EDF Research, France. E-mail: Marc.Bouissou@edf.fr

Pavel Krcal

Lloyd's Register, Sweden. E-mail: Pavel.Krcal@lr.org

Pengbo Wang

Lloyd's Register, Sweden. E-mail: Pengbo.Wang@lr.org

Model-based safety analysis (MBSA) offers multiple advantages for the stakeholders. The close link between a system description and the model that supports safety or dependability analysis allows automated model building and facilitates model reviews. Maintenance of the model benefits from a high-level system description with assumptions presented explicitly as an integral part of the model. Finally, the analysis process can utilize various properties of the system that are encapsulated in the components and their interactions. This paper focuses on the flexibility of analysis that does not require large remodeling or even building a new model when the question to be answered calls for additional or different system aspects to be considered. We show how knowledge bases built in the Figaro modeling language enable this flexibility on several examples from various application domains and for multiple specific system features. The modeling effort becomes to a large part decoupled from the analysis. An analyst building a model does not have to keep a specific analysis method in mind and might not even be aware of methods that will be required in the future.

Keywords: *Knowledge base, High-level modeling, Figaro, Automated model generation, MBSA.*

1. Introduction

The way in which we model a system affects the way in which we can perform its dependability analyses. Even though the system under study is the same one, the degree of conservatism in results differs between a fault tree model, a Markov chain, Petri nets or differential equations. The other way around, the purpose of an analysis and perhaps the focus on particular aspects of system behavior point towards a specific modeling formalism. If we have a description of components, their failures, behavior and interactions with other components (a *knowledge base*) then building a model and defining analyses become two orthogonal tasks. Modeling includes selecting system components and connecting them according to the actual dependencies. Defining an analysis means describing the properties of interest and the set of behaviors and/or interactions that shall be included in the analysis. As long as these behaviors are specified in the knowledge base, the model does not need to change.

This structure of encapsulating possibly complex dependability relevant functioning of components and their interactions into knowledge bases and using these pre-defined components in models of specific systems is used in some tools, such as RiskSpectrum ModelBuilder (KB3), with a graphical interface for modeling, complemented by import possibilities, and a support for various analysis tools. We describe the exact mechanism allowing us to decouple modeling from analysis and exemplify its power on several cases related to scenarios occurring in the industrial praxis. We also illustrate on some of the examples how different aspects of the same system can be captured in a single model. An analyst can afterwards decide which features or aspects of the real-life system shall be considered in the analysis and which ones can be screened away or abstracted by a simplifying mathematical description that enables using a more efficient analysis algorithm.

We include the following cases:

- A production system where productivity is as important as reliability. We show how stand-by components and different repair strategies can be modeled and how can they be taken into account in dependability calculations.

- A spent fuel pool system of a nuclear power plant with a special role of repairs under a long mission time. We use this system also to

Proceedings of the 31th European Safety and Reliability Conference
Edited by Bruno Castanier, Marko Cepin, David Bigaud and Christophe Berenguer
 Published by Research Publishing, Singapore
ISBN: 981-973-0000-00-0 :: doi: 10.3850/981-973-0000-00-0 esrel2021-paper

illustrate modeling of deterministic failure and grace times that arise from physical aspects of the system.

- A thermohydraulic safety system enabling analyses beyond standard fault trees.

- A system that cannot be repaired during its mission-time, where cold spares are essential for safety.

This work relies on the MBSA modeling language Figaro (Bouissou et al., 1991) and the methodology based on knowledge bases. There are several other established MBSA frameworks which offer a high-level modeling approach for dependability studies. AltaRica (Point and Rauzy (1999), Arnold et al. (2000)) supports hierarchical modeling of components and their interaction. Models ultimately translate to guarded transition systems with a precise mathematical semantics and a number of analysis tools. Lipaczewski et al. (2015) gives an overview of the main features by a comparison to another MBSA framework, Safety Analysis Modeling Language (SAML) proposed by Güdemann and Ortmeier (2010). Hierarchically Performed Hazard Origin and Propagation Studies (Hip-HOPS) (Papadopoulos and McDermid (1999)) equips individual components with failure modes and mechanisms of their propagation. This allows for standard Fault Tree Analysis (Papadopoulos and Maruhn (2001)) and Failure Modes and Effects Analysis. But this formalism offers also dynamic analyses, e.g., based on Petri Nets (Kabir et al. (2018)). Formal Safety Analysis Platform (FSAP) (Bozzano and Villafiorita (2006)) utilizes symbolic model checking to perform safety analysis of a system model. This platform has been succeeded by xSAP (Bittner et al. (2016)) which extends it by providing general libraries for modeling and additional tools for safety analysis.

All of these formalisms offer a high-level modeling approach for dependability studies. An analyst works only with (a graphical representation of) components on the level of the system description that Computer Aided Design (CAD) or Model Based System Engineering (MBSE) tools use. These tools typically lack a direct support for dependability analyses. For example, work of Flori et al. (2018) links models specified in the MBSE tool Capella to a description of system's failure behavior in Figaro to allow for safety analysis. However, failures of components have to be modeled ad hoc for each component. There is no generic knowledge about failure behavior of components to rely on.

2. Preliminaries

This section presents basics of knowledge bases and the modeling language Figaro. We also briefly describe how knowledge bases can be used in a graphical modeling environment such as RiskSpectrum ModelBuilder (KB3).

Figaro is a text-based, object-oriented general-purpose modeling language for dependability analyses. Its semantics is a timed state-transition system with both deterministic and stochastic transitions. The language offers modelers great support to abstract away from low-level transition systems and think in terms of components and their interactions.

Components can be defined as classes in a standard class-hierarchy used in object-oriented languages. Figaro uses so called occurrence and interaction rules to describe the behavior of a class. Occurrence rules describe stochastic transitions with their guards and the associated probability distributions. The purpose of interaction rules is to propagate the effects that are the immediate and certain consequences of a stochastic transition in the system. A detailed description of all features can be found in Bouissou et al. (2019).

Figaro can serve as a general language to describe standard formalisms used in dependability studies, such as Fault Trees, Markov Processes, Generalized Stochastic Petri Nets, Digraphs and Reliability Block Diagrams. It can also be used to build customized modeling libraries for various types of systems, such as thermohydraulic, electric, or production systems. In Figaro libraries, classes may have many specific features that go beyond a combinatorial characterization of the failure behavior. For example:

- Repairs, including complex repair strategies with shared resources and imperfect repairs;

- Stand-by dependencies and multistage reconfiguration strategies;

- Deterministic behavior: clocks, grace periods, deterministic timing of failures;

- Production / processing / throughput

Knowledge bases consist of component descriptions (class definitions in the Figaro language) and a graphical part that turns components into graphical icons that enable graphical building of system models, their linking through graphical links or by setting instance specific properties and visualization of the component state. Defining a model then consists in selecting correct components, placing them on a canvas, linking them, and setting instance specific data. This includes also definitions of system configurations – initial states, relevant failure modes, and success criteria. Information entered for a specific model is sometimes called a *fact base*. Each fact base requires a knowledge base to be able to interpret behavior of components and their interaction.

Finally, we sketch the mechanism that separates modeling and analysis. The actual process of building a model does not require detailed description of system behavior. This is encapsulated in the knowledge base. Therefore, the Figaro description of components can include many types of behaviors. Figaro allows to separate

them into different groups (called *rule groups*) which give rise to different aspects of the behavior. For example, rules for a component failing can be in one group. Rules for repairs can be in another group. The analyst builds the same system model irrespective of whether an analysis should take repairs into account or not. At the moment of defining an analysis, one needs to specify which rule groups shall be taken into consideration.

The following sections will illustrate these concepts and especially the flexibility of analysis on several examples, showing (highly simplified) industrial systems. Knowledge bases for all examples are publicly available as a part of Visual Figaro (2021).

3. Production System

This section gives examples on features that can be utilized when modeling a system that processes or produces something based on the status, capabilities and relations between individual blocks in a plant. We exemplify it by a fault tolerant production system where production units corresponding to various stages of the elaboration of a product are in series, but each unit may contain several production blocks in parallel, in order to cope with possible failures without having to stop the whole system. Blocks can differ in their processing/production capacity and their reliability data.

Let us consider the following plant structure. There are three units connected in series through a single production pipe. The first unit contains a single block that simply processes the input with the capacity of 100%. The next unit consists of four blocks in parallel. Two of them have capacity of 30%, while the other two ones have capacity 40%, but the fourth block is a cold spare for the third one. Therefore, the maximal production capacity of this unit also adds up to 100%. The last unit consists of eight identical blocks with 15% capacity each. This exceeds the 100% capacity, which means that the blocks do not need to run on their maximal nominal capacity to cover the demand, unless some of them fail.

One can model this plant by the tree in Figure 1 where “gates” encode whether their children are connected in series or in parallel. The total plant production, at the top of the tree, is the minimum of the productions of the three units. Each unit produces the sum of the capacity of its children. One of the blocks in the second unit, denoted D2, is failed in the represented configuration in Figure 1. The red dashed line between the blocks C1 and C2 denotes the stand-by redundancy.

There are different types of analyses that one could perform on such a system. One could specify minimal performance criteria such as the minimal number of blocks in each unit that are required so that the whole plant can fulfill its task. From this, one can let the tool generate a fault tree and obtain the unavailability of the plant by a standard fault tree analysis, given unavailabilities of individual blocks.

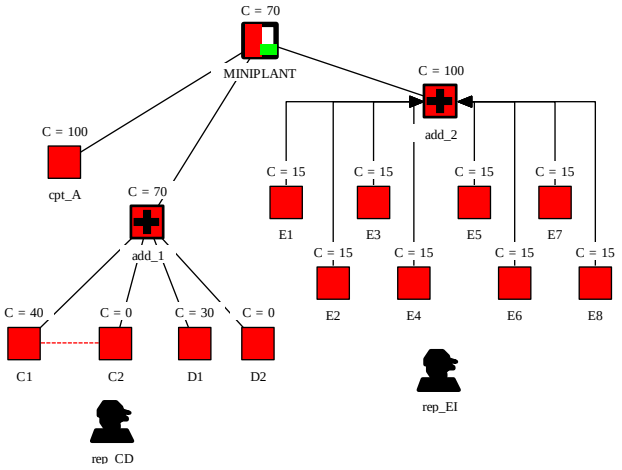

Figure 1. A model for the production analysis of a sample plant built with the Miniplant knowledge base in RiskSpectrum ModelBuilder (KB3).

This might give a very coarse indication about the effect of possible failures on the plant production, which might be in its turn also of a very limited value in risk-based decision taking. One might want to analyze the average plant production capacity or a fraction of time in which the plant production drops below a certain critical level. For this, we need to take repairs of failed blocks into account.

We could model repairs as independent from each other, assuming sufficient resources to perform a repair whenever a block fails. This can be a reasonable approximation in highly reliable systems. If we want to refine the analysis, we might want to model the fact that there are limited resources for repairs and they are used according to a pre-defined repair strategy. For instance, we assume that each unit has its own repair crew, which repairs failed blocks in the first-come first-served order. This is depicted by two repair crew icons in Figure 1.

The block repairs, repair strategy and a class for a repair crew are specified in the knowledge base. The only work that is left for the model (the fact base) consists of specifying repair parameters, e.g. mean times to repair for the case of exponentially distributed repair times or parameters for differently distributed repair times, defining the repair crews and their areas of responsibility.

Once we want to analyze the production level of a plant then we need to reach above the fault trees, above the definition of failures and their propagation through the system. We need to capture the state and its changes in time.

In the Miniplant knowledge base, occurrence rules model the failures, repairs and the distributions of their times, while interaction rules model the bottom up propagation of production capacity, the switching on and off of C2 and the management of the first-come first-served repair policy. The analysis of such a model can resort to a Monte Carlo simulation, which offers full flexibility also when it comes to the choice of probabilistic distributions.

The analysis is again steered by the parameters that we select when we start it. The underlying model stays the same. We can select rule groups or attributes in the knowledge base that determine if RiskSpectrum ModelBuilder (KB3) generates a fault tree for pre-defined static success criteria or if it generates a stochastic transition system whose simulation yields the mean production of each subsystem and the whole system on a given time period (Bouissou et al. 2004). For all repair strategies implemented in the knowledge base, it suffices again to select the appropriate rule groups or set the value of an attribute to select the repair strategy to be considered in the analysis.

4. Thermohydraulic Systems

Modeling thermohydraulic systems is a large part of Probabilistic Safety Assessment work for, e.g., Nuclear Power Plants. The scope, resolution and the regulatory requirements on regular updates of such studies pose a severe challenge for structuring the model. First, the relation between the actual system and the resulting fault trees must not be obscured. Secondly, assumptions made while building the model have to remain transparent for reviewing and future updates.

A well-established approach to cope with all these challenges is to convert Piping and Instrumentation Diagrams (P&IDs) automatically to a model based on a corresponding knowledge base, for instance from a CAD tool. The visual part of the system description in RSMB naturally matches the system topology in the graphical part of a P&ID. By this, one can preserve the relation between the system structure and a (high level) system safety model. Safety analysts still need to input all reliability information which is not present in the P&IDs. A knowledge base tailored to a specific plant can simplify this work as a large portion of reliability data can be pre-defined dependent on the component types. The rest needs to be input separately either by manual updates or by an import function. Figure 2 contains a model of a sample cooling system.

For a reliability analysis, one needs to specify top failures and relevant plant configurations. The tool then automatically generates fault trees for the analysis, which can be solved by a fault tree/event tree analysis tool. There is a direct import function from RiskSpectrum ModelBuilder (KB3) to RiskSpectrum PSA, where fault trees can be linked to event trees. The event tree/fault tree solver then performs an analysis of relevant sequences or consequences. Results can be evaluated by inspection of minimal cut sets or by built-in importance, sensitivity and uncertainty analysis.

Automated fault tree generation in RiskSpectrum ModelBuilder (KB3) ensures a uniform structure of fault trees independent of individual safety engineers building the model. Also, all updates to the model are performed on the high-level representation in RiskSpectrum ModelBuilder (KB3) and again propagated to fault trees in a uniform way through the automatic fault tree generation.

Figure 3 shows an automatically generated fault tree for one of the trains of the component cooling system from Figure 2.

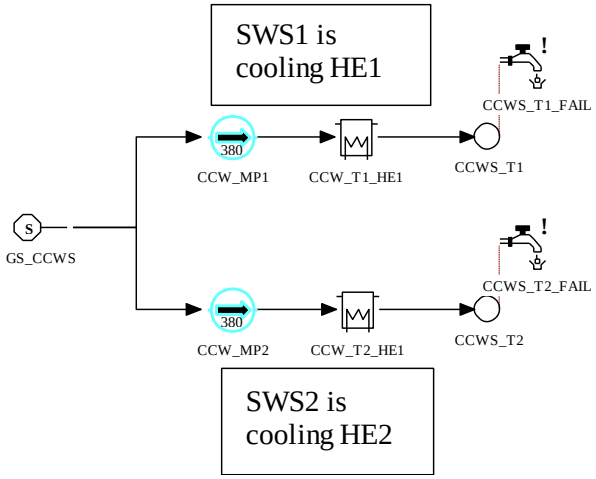

Figure 2. A model of a sample Component Cooling Water system in RiskSpectrum ModelBuilder (KB3) built with a thermohydraulic knowledge base.

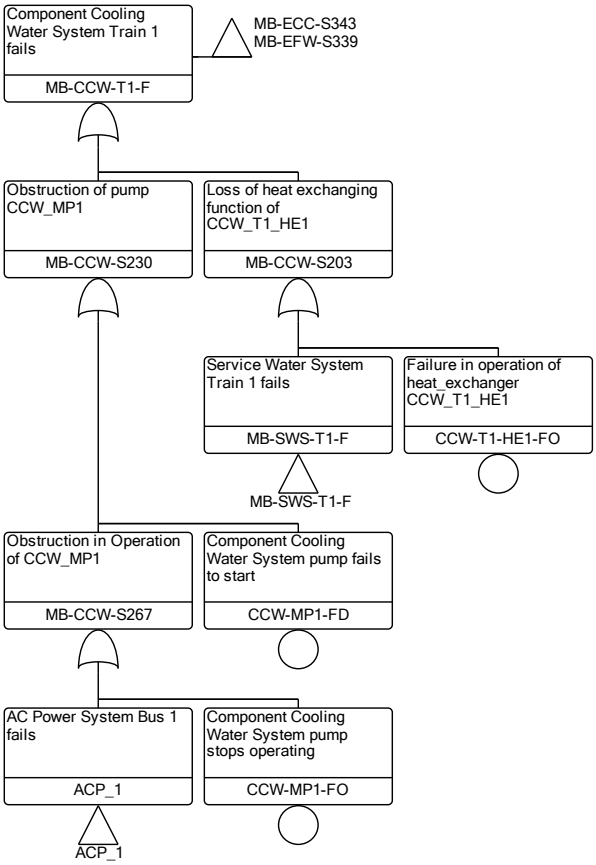

Figure 3. An automatically generated fault tree for Train 1 of the Component Cooling Water system from Figure 2.

Both options have their pros and cons for the model building and mostly for the model quality assurance. The former does not require ad-hoc changes of failure definitions by modelers, but it can lead to increasingly complex knowledge bases. The latter requires manual modeling of special cases by so called Manual Rules. This makes it easier for a modeler or reviewer to understand where and how these special cases are covered. On the other hand, it is easier to forget a manual rule at some place in the model. In both cases, it is an algorithm that performs most of the repetitive work that an engineer would have to do when drawing all fault trees manually. This eliminates a large portion of error-prone activities and leaves more time to focus on quality assurance for special cases. Moreover, there are additional possibilities to validate and trouble-shoot a model.

Even though the size and complexity of large-scale PSA models poses tremendous challenges to alternative analysis methods, one can use them on a smaller scale to validate parts of the model. The simplest method suitable for thermohydraulic systems is an interactive simulation (Figure 4) which allows users to explore combinations of events and inspect their effect on the system. Provided that a visualization of the system state is pre-defined in the knowledge base, starting the analysis via interactive simulation requires only to select the corresponding rule group and possibly to define an initial state.

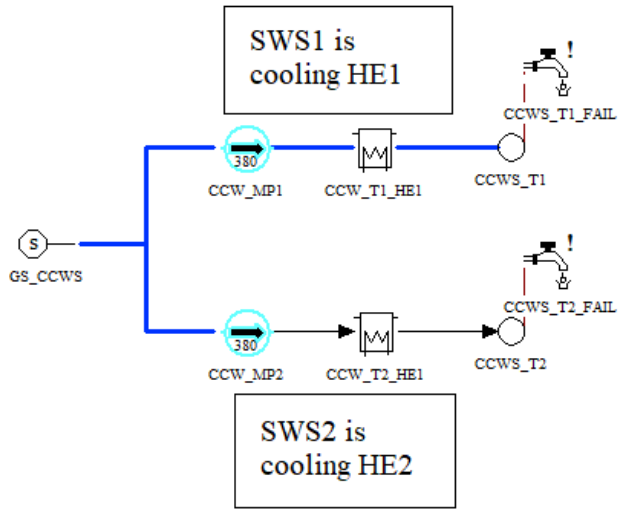

Figure 4. A snapshot of an interactive simulation of the Component Cooling Water system from Figure 2 that shows the effect of a failure of the pump CCW_MP2 on the flow.

An example knowledge base for thermohydraulic systems that contains both fault tree generation and interactive simulation is publicly available as a part of Visual Figaro (2021).

5. Spent Fuel Pool

We describe a simplified cooling system of a Spent Fuel Pool of a nuclear power plant Olsson (2018) in this section. Even though the model itself utilizes standard thermohydraulic components, it is not only another illustration of the features described in the previous section. A definition of safe end states for accident scenarios of a Spent Fuel Pool requires significantly longer mission times compared to a standard Probabilistic Safety Assessment (PSA). Therefore, the features that we want to highlight here, especially the ability to select between an analysis with repairs considered or a standard non-repairable fault tree analysis, gain on their importance. Moreover, we have a possibility to specify a grace delay due to the thermal inertia of the water in the pool and deterministic failures of equipment like exhaustion of batteries or tanks.

The cooling function consists of a primary cooling system that cools the spent fuel pool water by two redundant heat exchangers. If this system fails, then there is a possibility to feed water from the reactor water storage into the fuel pool (“feed and bleed” mode). A scheme created in RiskSpectrum ModelBuilder (KB3) in Figure 5 and Figure 6 shows the components of these systems.

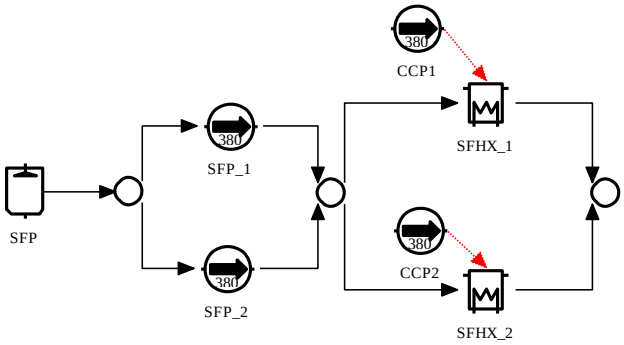

Figure 5. The main cooling system of the Spent Fuel Pool.

The most straightforward way of analyzing reliability of thermohydraulic systems in the nuclear domain is the (static) fault tree analysis. Here, we assume that the safety systems can bring the plant into a safe state within a mission time estimated by an analyst. This model can then be automatically converted to fault trees and analyzed by a fault tree analysis tool.

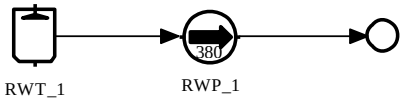

Figure 6. The feedwater system of the Spent Fuel Pool.

As mentioned above, the case of the Spent Fuel Pool requires prolonged mission times compared to a standard PSA. On the other hand, when we want to analyze accident durations of more days or weeks, the results will become overly conservative for non-repairable systems. This might skew any safety insights obtained from the importance or

sensitivity analysis and might also make it more difficult to achieve regulatory requirements.

In this case, we assume the following model of repairs – each component is repaired independently of other ones and the repair times are distributed exponentially. In other words, each component where we want to consider repairs has a mean time to repair assigned. The possibility to repair components and the actual repair mechanism can be already pre-defined in the knowledge base. This means that the graphical representation of the model is the same for both the standard fault tree analysis and the analysis taking repairs into account. The only part that an analyst needs to update in the model fact base to enable an analysis with repairs are component repair rates if (just like for failure rates) they differ from the default values pre-defined in the knowledge base.

The choice between an analysis with repairs or without taking repairs into account can be done by setting a variable in the system, illustrated by a dialog from RiskSpectrum ModelBuilder (KB3) in Figure 7, or selecting the correct rule group. In this use case, the failure behavior can be captured by fault trees with repairable components. This makes it suitable for an efficient analysis method called *I&AB* developed by Bouissou and Hernu (2016), Bouissou (2018), and whose implementation in RiskSpectrum PSA has been evaluated on full-scale PSA studies in Bäckström et al. (2018).

Generating a fault tree from a fact base (a specific model) and a knowledge base (generic description of component behavior and interactions) relies on the concept of failure models which ultimately define basic events. The selection of the rule group for repairable or non-repairable systems in our example determines whether the failure of individual components is modeled by failure models corresponding to the mission time reliability model or to the model of a component that can be repaired under the accident duration. Having a model with repair rates, it takes no additional effort to run an analysis either with or without repairs.

Object	Nature	Variable	Default values
system	Attribute	repairable	TRUE
system	Constant	system_total_cost	(SUM FOR ALL x A...
system	Attribute	visualization	'unknown'

Figure 7. One possibility to tell whether the system shall be analyzed as repairable can be to set the attribute “repairable” of the object “system”. This is all pre-defined in the knowledge base.

There are two additional features that occur in the spent fuel pool accident scenario and which are difficult to capture in a standard fault tree analysis. The first one stems from the fact that a failure to cool the pool leads to heating the water until it starts to boil. If the cooling system is repaired before this happens then we can avoid the undesired consequence. This gives us an additional fixed time period to repair the cooling system and restore its function – so called *grace time*.

This fact can be modeled, for instance, by a component added to the plant scheme and connected to the output of the cooling system. Again, depending on whether we consider repairs or not in the analysis, this will be used or ignored in the analysis. If we abstract away from the grace time then we risk overly conservative results that decrease usability of the safety analysis.

The second feature is typically a limited capacity of a component used in a safety system that gets depleted withing a fixed, constant time – so called *deterministic failure*. This can occur in many situations, such as batteries powering a safety system or, as in this case, a back-up water reservoir that can supply a limited amount of coolant. The water tank of the back-up cooling system will be depleted after a fixed time. This will cause the back-up cooling system to fail, irrespective of the failure of the pump.

This is again a failure mode that can be pre-programmed in the knowledge base. An analyst needs only to specify the fixed failure time for the water tank. This feature cannot be considered in a standard fault tree analysis, but for instance the I&AB method can use it (in an approximative way). Therefore, once the model is completed, it is up to the analyst to select which features should be considered in the analysis. This choice does not involve any modeling effort. One simply selects which features shall be included in the analyzed model.

This choice naturally influences the computational cost of the analysis. In a general case, including repairs would require transient analysis of the underlying Markov process. Approximations of the I&AB method avoid the state space explosion and yield calculation times comparable to those of a static fault tree analysis (Bouissou et al. (2020)).

6. Non-Repairable Mission System

Previous sections illustrated flexibility of analysis on systems where accounting for repairs increases precision of analysis. Here we explore systems that are non-repairable by design, such as aerospace systems on a mission. One of the techniques to increase reliability of these systems is to add cold stand-by redundancies.

Typical probabilistic analyses of such systems rely on fault trees. However, fault trees can hardly model phased mission systems, and can be overly conservative when it comes to modeling cold redundancies.

Here is a simple example: an airplane with an engine and propeller on each wing (Fig. 8) needs the two engines to work at full power during take-off and beginning of ascent. After, say, five minutes of ascent, it can fly with only one engine. The fuel is stored in the wings. Normally, each engine is supplied by the tank situated on the same wing. But in case of fuel shortage (because of a human error in the filling procedure, or a leak), the pilot can reconfigure the fuel circuits to supply both engines by a single tank. This reconfiguration can fail. The failure rate associated to tanks failures is supposed to be higher in the

ascent phase than in the flight phase: most failures are due to events that happened before or during take-off.

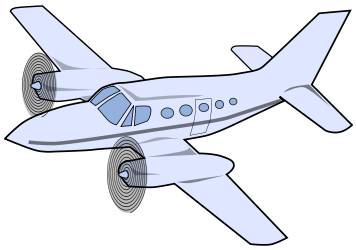

Figure 8. Twin-engine airplane

Even for this simple example, fault tree analysis is not a usable method. The minimal requirements and failure rates differ between the two phases, which means that two fault trees would be needed. But they cannot be treated independently. A correct probability assessment must take into account the dependency between the phases. If a tank is emptied during ascent, this must be remembered to initialize the model of the flight phase.

A natural modeling formalism for this case are Boolean logic Driven Markov Processes (BDMPs) by Bouissou and Bon (2003). It extends the fault tree formalism by carefully selected features that, on one hand, make modeling very close to well-known fault trees, allow the analysis to select the most efficient method given the features used, but cover a large set of dynamic features on the other hand. In fact, examples from previous sections could be also modeled as BDMPs.

The prominent BDMP feature used in this section is a *trigger*. It expresses cold stand-by dependencies in the model. The secondary back-up system is started only when the primary system fails. If failures of both systems are modeled by gates in a BDMP then we say that the gate for the primary system triggers the gate of the secondary system and depict it by a dashed red line.

The BDMP shown in Figure 9 and commented hereafter is a straightforward solution to this modeling problem. If the length of phases is represented as exponentially distributed, the model is Markovian and it can be processed with very quick and precise Markov analysis tools. In order to take into account phases with a fixed time, the only change to do in the model is an option in the clocks representing the phases. But then, a Monte Carlo simulation is needed to solve the model.

The clocks represent phases. Initially the "ascent" phase is active (considered as true in the logical structure), and thanks to the trigger going from this phase to the "flight" phase, when the first phase ends, the second one immediately starts. The two lower triggers are there to trigger on demand failures that can happen when a reconfiguration of the fuel path is needed. The leaves "tank1_empty" and "tank2_empty" are of the type SF (standby failure); they have a higher failure rate as long as the "ascent" phase is active, thanks to the triggers coming from the "ascent" phase. Hence, this BDMP accurately represents all the hypotheses on the airplane mission.

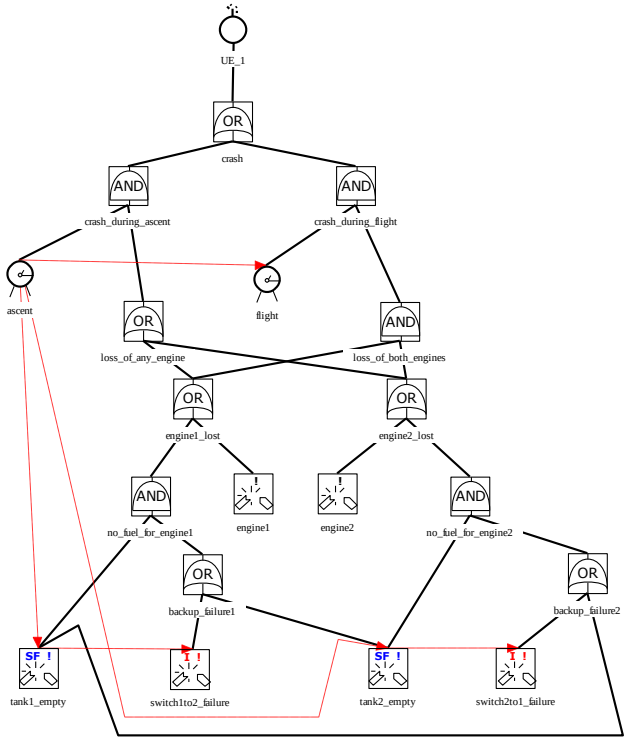

Figure 9. A BDMP corresponding to the safety analysis case of an airplane mission.

The analyst can flexibly select between a standard fault tree analysis of this model (e.g., for a qualitative evaluation), a Markovian analysis which approximates the fixed mission times by an exponential distribution, or a Monte Carlo simulation of the complete system model. The model itself does not have to be updated for these analyses. The analyst chooses corresponding rule groups at the moment of defining or starting the analysis.

As in the Spent Fuel Pool case, also here will the choice of modeling features influence calculation costs. Efficiency of the Markovian analysis will substantially depend on selected approximations. A transient analysis of the underlying model might not scale beyond ca 300 basic events. Applicability of approximate methods depends on the model characteristics, such as reliability of the system (low failure rates, high repair rates). For certain classes of systems, one can hope that results sufficiently close to the exact values might be obtained even for larger models. Monte Carlo simulations are most flexible as they do not restrict the distributions of time intervals between events. The drawback might be longer simulation times, especially for reliable systems with rare events.

7. Conclusions

This paper illustrates the flexibility that a system model based on a Figaro knowledge base gives us for its dependability analyses. Knowledge bases can simultaneously capture multiple system features, such as repairs including complex repair strategies, stand-by dependencies, deterministic system development, production levels, and mission phases. An analyst who

enters the model on the level of a system description does not have to keep a certain type of analysis in mind. Only when a system is defined, the analyst selects which of the model features shall be considered by the analysis. This also determines the analysis tools that have the capability to produce required results.

References

Arnold A., Griffault A., Point G. and Rauzy A.. The AltaRica formalism for describing concurrent systems. Fundam. Inf. 40, issue 2-3, pages 109-124, IOS Press, 2000.

Bäckström, O., Bouissou, M., Gamble, R., Krcal, P., Sörman, J. and Wang, W. (2018). Introduction and Demonstration of the I&AB Quantification Method as Implemented with RiskSpectrum PSA, Proc. of PSAM'14.

Bittner B., Bozzano M., Cavada R., Cimatti A., Gario M., Griggio A., Mattarei C., Micheli A., and Zampedri G. (2016). The xSAP Safety Analysis Platform. Proc. of TACAS'16.

Bouissou M., Bon J. L. (2003). A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes. *Reliability Engineering & System Safety 82*, 149-163.

Bouissou M., Bouhadana H., Bannelier M., Villatte N. (1991). Knowledge modeling and reliability processing: presentation of the Figaro language and associated tools. Proc. of SAFECOMP'91.

Bouissou M., Chraibi H., Muffat S. (2004). Utilisation de la Simulation de Monte Carlo pour la résolution d'un benchmark (MINIPLANT). 14ème congrès de fiabilité et maintenabilité, Bourges, (France).

Bouissou, M. and Hernu, O. (2016). Boolean approximation for calculating the reliability of a very large repairable system with dependencies among components, Proc. of ESREL 2016, Glasgow, UK.

Bouissou, M. (2018). Extensions of the I&AB method for the reliability assessment of the spent fuel pool of EPR, Proc. of ESREL 2018, Trondheim, Norway.

Bouissou, M., Humbert, S. and Houdebine, J. (2019), Reference manual for the FIGARO probabilistic modelling language (Version-E), EDF.

Bouissou, M., Khan, S., Katoen J.-P., and Krcal P. (2020) Various Ways to Quantify BDMPs. In Proc. of MARS'20.

Bozzano, M., Villafiorita, A. (2006). The FSAP/NuSMV-SA safety analysis platform. *International Journal on Software Tools for Technology Transfer (STTT) 9*, 5-24.

Flori, A., Flori, S. and Houdebine, J. (2018). Experiment on Exchanges between Models in Systems Engineering and Models in Dependability / Cyber-security for Dynamic Systems. (*in French*) Proc. of Congrès Lambda Mu 21.

Güdemann M., Ortmeier F. (2010). A framework for qualitative and quantitative model-based safety analysis. In Proc of HASE 2010.

Kabir S., Walker M., and Papadopoulos Y. (2018). Dynamic system safety analysis in HiP-HOPS with Petri Nets and Bayesian Networks. *Safety Science* 105:55-70.

Lipaczewski M., Ortmeier F., Prosvirnova T., Rauzy A., and Struck S. (2015). Comparison of modelling formalisms for Safety Analyses: SAML and AltaRica, *Reliability Engineering & System Safety 140*, 191-199.

Olsson, A. (2018). Leaving mission times backstage and taking repair into account in long term scenarios, Proc. of PSAM 14.

Papadopoulos Y., McDermid J. (1999). Hierarchically performed hazard origin and propagation studies. In Proc. of SAFECOMP.

Papadopoulos Y, Maruhn M (2001). Model-based automated synthesis of fault trees from Matlab-Simulink models. Proc. of International Conference on Dependable Systems and Networks (DSN'01).

Point G., Rauzy A. (1999). AltaRica: Constraint Automata as a Description Language. *Journal Européendes Systémes Automatisés* 33(8–9):1033–52.

Visual Figaro (2021). The package contains sample knowledge bases. https://sourceforge.net/projects/visualfigaro/.