Preliminary System Safety Analysis with Limited Markov Chain Generation

Pierre-Antoine Brameret, Jean-Marc Roussel, Antoine Rauzy

Preliminary System Safety Analysis with Limited Markov Chain Generation

P.-A. Brameret * **J.-M. Roussel** * **A. Rauzy** **

* Lurpa, Ens *de Cachan, 61 avenue du Président Wilson, 94235 Cachan cedex,* France
{*pierre-antoine.brameret,jean-marc.roussel*}*@lurpa.ens-cachan.fr*
** Lix, *École Polytechnique, Route de Saclay, 91128 Palaiseau cedex,* France
rauzy@lix.polytechnique.fr

Abstract: Markov chains are a powerful and versatile tool to calculate reliability indicators. However, their use is limited for two reasons: the exponential blow-up of the size of the model, and the difficulty to design models. To overcome this second difficulty, a solution consists in generating automatically the Markov chain from a higher level description, e.g. a stochastic Petri net or an AltaRica model. These higher level models describe the Markov chain implicitly. In this article, we propose an algorithm to generate partial Markov chains. The idea is to accept a little loss of accuracy in order to limit the size of the generated chain. The cornerstone of this method is a Relevance Factor associated to each state of the chain. This factor enables the selection of the most representative states. We show on an already published test case, that our method provides very accurate results while reducing dramatically the complexity of the assessment. It is worth noticing that the proposed method can be used with different high-level modeling formalisms.

Keywords: Model Based Safety Assessment, Markov chains, State space build

1. INTRODUCTION

Safety and reliability of critical systems are today of a paramount importance. Consequently, system designers have to perform safety and reliability analyses since the earliest phases of their projects. These preliminary studies must be both representative of the designed system and obtained within short delays. Therefore, approximate calculations of reliability indicators are acceptable provided they are accurate enough.

Nowadays, industrial systems embed complex safety mechanisms, e.g. cold redundancies and reconfigurations. Fault Trees and related formalisms are not powerful enough to capture the behavior of theses mechanisms. Markov chains have much greater expressive power. They are a powerful and versatile tool to calculate reliability indicators. However, their use is limited for two reasons: the exponential explosion of the size of the model, and the difficulty to design models. To overcome this second difficulty, a solution consists in generating automatically the Markov chain from a higher level description, such as a stochastic Petri net (Ajmone Marsan et al., 1994), an AltaRica model (Prosvirnova et al., 2013) or a Boolean Driven Markovian Process (Bouissou and Bon, 2003). These higher level models describe the Markov chain in an implicit way.

In this article, we propose an algorithm to generate partial Markov chains. The idea is to accept a little loss of accuracy in order to reduce the size of the generated chain. The cornerstone of this method is a Relevance Factor associated to each state of the chain. This factor enables the selection of the most representative states. We show on an already published test case made with 11 components, that it is possible to obtain the unreliability (probability that the system is down at a given time) with a relative error lower than 0.25% while limiting its state space to only 74 states instead of $1,056$. It is worth noticing that the proposed method can be used with different high-level modeling formalisms.

The remainder of this paper is organized as follows. Section 2 introduces the context of the work, some already existing Markov chain reduction methods, and an overview of our method. Section 3 is the core of the paper and presents the limited build of a state space. Section 4 shows the results of the method applied to an example taken from literature, where numeric values for probabilistic indicators can be compared.

2. PROBLEM STATEMENT

2.1 Context of the work

Fig. 1 presents the context of our work: the preliminary system safety analysis. In this early phase of the project, engineers must compare different possible architectures for the studied system. It is common for them to have a refined view on these architectures without an accurate knowledge of the safety indicators of components. A mean to compare these architectures is to assess the safety indicators of a system built with each architecture.

Calculation of the safety indicators can be made as follows:

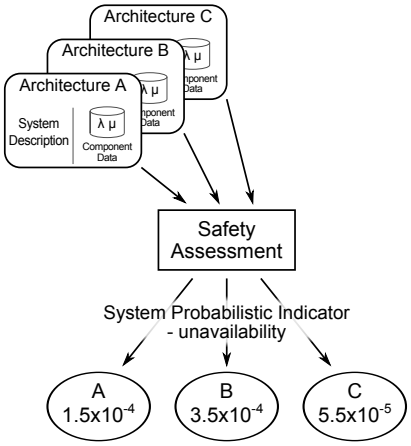


Fig. 1. Context of the work: preliminary system safety analysis

- build (homogeneous) continuous time Markov chain which represents the failure/repair behavior of the fault-tolerant system,
- evaluate the Markov chain for several mission times.

When architectures are well detailed, the system has a large number of states and an even larger number of transitions between those states. As the computation cost is expensive, safety engineers are often forced to limit their experimentation. That is why efficient methods to obtain safety indicators must be developed.

2.2 Related work

Evaluation of continuous-time Markov processes (Stewart, 1994) is usually done by translating the Markov process into a set of differential equations. Solving this differential set leads to compute the various probabilities associated to the identified states of the system. To solve the set of equations, analytical (Bolch et al., 2006) or numerical methods (Rauzy, 2004) exist. The cost of solving a set of differential equations is very high when the number of states of the Markov chain increases.

Evaluation of *large* Markov chains is an important challenge. To solve this problem, reduction of the Markov chain is usually done. Some authors propose exact methods (a smaller Markov chain *equivalent* to a bigger one is produced) and others, approximate methods.

Lal and Bhat (1988) considered the inversion of transition matrices and optimized calculations of steady state probabilities by partitioning the transition matrix. It is not strictly speaking a reduction of the model to a smaller model, but it is yet a reduction of the complexity towards reward assessment. Pribadi et al. (2001) introduced a method to reduce Markov chains in size while guaranteeing the exact probabilistic assessment of rewards associated to states. It is efficient to reduce the chain toward the calculation of the reward but this relies on ergodicity of the chain to be reduced, which may be a strong hypothesis. Fourneau et al. (2007) censored the Markov chain to the states which give non-null reward, but as shown by Zhao and Liu (1996), the equivalence between the censored Markov chain and the initial Markov chain is complex to calculate. Thus, a method to approximate the censored Markov chain is developed by Fourneau et al. (2007), which gives bounds to the reward.

For all these efficient methods, the input data is only a Markov chain without complementary information about the represented system. In our case, the method exploits information on the studied system to partially build the state space of the failure/repair behavior of the system. We directly obtain a shorter Markov chain, easier to evaluate. It is important to note that our method is essentially heuristic and produces a good level of approximation.

2.3 Proposed method

Fig. 2 presents the strategy we propose to calculate safety indicators of a given architecture. The input data of our method is a description of the dynamic of the system expressed with a safety model (as AltaRica models (Prosvirnova et al., 2013), Dynamic Fault Tree models (DFT, Dugan et al. (1992)), Boolean logic Driven Markov Processes (BDMP, Bouissou and Bon (2003)), Safety Analysis Modeling Language (SAML, Gudemann and Ortmeier (2011)), ...) or more generalist model as Petri Nets (PN). To apply our method, the only requirement is to be able to construct the state space of the system regarding the failure/repair behaviour. To avoid combinatorial explosion during the evaluation of the Markov chain, we propose to build only a part of the state space by selecting the most significant states. In order to do so, we propose a Relevance Factor, which is defined in Section 3.1.

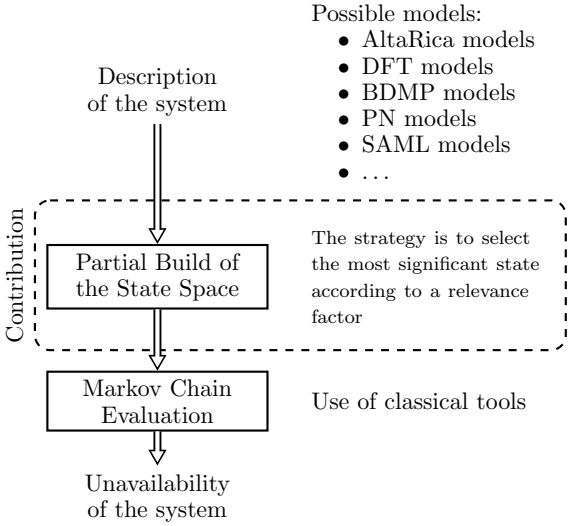


Fig. 2. Proposed method to obtain the unavailability of a system

The description of the system is used to build its state space partially, based on failure/repair events. A Markov chain is then generated from the state space, knowing which failure/repair event leads from a state to another. The computation of the unavailability of the system is made with the Markov chain. For this step, classical tools can be used.

Our method is entirely based on the following practical statement of fact:

- In the complete state space of the system, some states are more interesting toward assessment of safety in-

dicators than others. Those with important steady-state probabilities are more influential in the assessment than those with small steady-state probabilities.
- By suppressing states with small steady-state probabilities, probabilistic indicators assessment is quicker with little loss of accuracy.

In our case, we build the state space partially by selecting directly the most interesting states.

3. LIMITED BUILT OF A MARKOV CHAIN

This section presents the main contribution of this proposal. The Relevance Factor defined Section 3.1, enables the identification of the states with the highest steady-state probabilities *while* building the state space.

Our method exploits properties of the studied systems and some characteristics of Markov chains:

- The nominal state of the system is the state with the highest sojourn time.
- For each state of a Markov chain, possible evolutions are *weighted* according to their transition rate. That is why the sojourn times of all states are not equal.

The partial build of the state space is based on the conservation of the states for which the Relevance Factor is superior to a threshold τ. The others states are ignored.

3.1 Relevance Factor

The method we propose is based on a *Relevance Factor* (Eq. 1) associated to each state. Its value (between 0 and 1) predicts the *influence* of the state within the state space. The most influential state has the highest value for its Relevance Factor. In our case, the most influential state is the nominal state. Its Relevance Factor is fixed to 1.0.

Let s_j be the state whose Relevance Factor $R(s_j)$ is to be evaluated. Let $q_{i\to j}$ be the transition rate from state s_i to state s_j. Let s_{init} be the nominal state. For all states s_j ($s_j \neq s_{init}$, as $R(s_{init}) = 1.0$), the relevance factor $R(s_j)$ is defined as follows:

$$R(s_j) = \max_{s_i \in \text{parent states}} \left(\underbrace{R(s_i) \times \frac{\overbrace{q_{i\to j}}^{\text{transition rate}}}{\sum_k q_{i\to k}}}_{R\text{-contribution from parent } i} \right) \quad (1)$$

The Relevance Factor is defined to decrease according to the exploration depth of the state space. The term $\frac{q_{i\to j}}{\sum_k q_{i\to k}}$ is inspired by the conditional probability to go from a state to another in continuous-time Markov processes. By construction, it is always inferior to 1. This aspect enables the decrease of the Relevance Factor from a parent s_i to its children s_j. Moreover, this term also takes into account the weight of outgoing transitions from s_i according to their transition rate $q_{i\to k}$.

Like differential equations of a Markov chain, the proposed Relevance Factor has a local definition only. In our case, its value only depends on the Relevance Factor of parent states: $R(s_i) \times \frac{q_{i\to j}}{\sum_k q_{i\to k}}$. By introducing *max* evaluation, we favor the influence of the most important parent state. It is important to note that the *max* evaluation is a key-feature of the method. It enables the limited exploration of the states.

3.2 Limited Build Algorithm

To avoid the calculation of the complete state space and then its reduction, Algorithm 1 has been specifically developed. It automatically obtains the most influential states without completely generating the state space.

This algorithm is based on neighbour discovery. It is inspired by the famous Dijkstra algorithm (Dijkstra, 1959) to specify the order in which states are explored.

Algorithm 1: Limited Exploration Algorithm

Input: Description of the system
Input: τ the threshold on the relevance factor
Output: $G = (\Sigma, \Theta)$ where Σ is the set of explored states and Θ is the set of explored transitions
Local: C the set of *candidate states*: *discovered* but not *explored* yet
Local: $R(s_i)$ the Relevance Factor of state s_i
begin
  `// Initialisation`
  $C \leftarrow \{s_{init}\}$
  $\Sigma \leftarrow \emptyset$
  $\Theta \leftarrow \emptyset$
  $R(s_{init}) = 1.0$
  `// Construction of the state space`
  while $C \neq \emptyset$ *and* $\max\{R(s_k), s_k \in C\} \geq \tau$ **do**
    `// Choose the candidate with highest Relevance Factor`
    $s_i = \arg\max\{R(s_k), s_k \in C\}$
    $C \leftarrow C \setminus \{s_i\}$
    $\Sigma \leftarrow \Sigma \cup \{s_i\}$
    `// Calculation of the possible evolutions from` s_i
    Discovery of all reachable states s_k from s_i
    $outRate = \sum_k q_{i\to k}$
    `// Calculation of` $R(s_j)$
    foreach *reachable state* s_j *from* s_i **do**
      if $s_j \notin \Sigma$ **then**
        $R = R(s_i) \times \frac{q_{i\to j}}{outRate}$
        if $s_j \in C$ **then**
          $R(s_j) = \max\{R(s_j), R\}$
        else
          $R(s_j) = R$
          `// Record of the state`
          $C \leftarrow C \cup \{s_j\}$
      `// Record of the transition`
      $\Theta \leftarrow \Theta \cup \{(s_i, (\text{event}, q_{i\to j}), s_j)\}$
  `// Clean transitions which are not leading to a state of` Σ
  foreach $t = (s_i, (event, q_{i\to j}), s_j) \in \Theta$ **do**
    if $s_j \notin \Sigma$ **then**
      $\Theta \leftarrow \Theta \setminus \{t\}$

The main input of Algorithm 1 is the description of the system with a safety model or a Petri net. The

input model is used to discover new reachable states from previously discovered ones. In Algorithm 1, only instruction "Discovery of all reachable states s_j from s_i" is specific to the description model given as input data. The Relevance Factor is calculated alongside the discovery of new states.

The build of the system's state space begins from its nominal state. This state is chosen by the system designer and usually corresponds to the state with all components up. In Algorithm 1, states are separated into two distinct families: *candidate states* (C) and *explored states* (Σ). Candidate states are states which are only *discovered*, but not *explored* yet. They are explored according to their Relevance Factor.

The state chosen to pursue the build of the state space is the candidate with the higher Relevance Factor. After its selection, this state becomes an *explored state*. Then, all reachable states from state s_i are calculated. Three cases are possible for each reachable state s_j:

- $s_j \in \Sigma$: s_j has already been explored.
- $s_j \in C$: s_j has already been discovered. The Relevance Factor of s_j may change.
- $s_j \notin (C \cup \Sigma)$: s_j is a new candidate state.

In any case, transition from s_i to s_j must be recorded.

When all the candidate states have their Relevance Factor below threshold τ, the exploration is stopped. The limited state space of the system is only the set of explored states (remaining candidate states are discarded). As some recorded transitions lead to candidate states, it is necessary to delete them at the end of the process.

The obtained partial state space contains only the most influential states of the state space of the system.

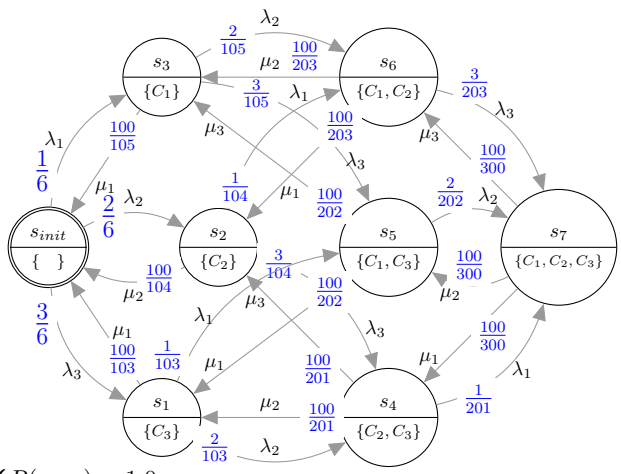
3.3 Illustration

Let us consider an illustrative system composed of 3 independent components C_1, C_2, C_3. Each component has a failure event (λ_i) and a repair event (μ_i). This system with 3 components has only 8 (2^3) states. Fig. 3 represents the complete state space obtained by applying the proposed algorithm with a threshold fixed to 0. States are labeled according to the order of discovery by our algorithm, and also, with the set of failed components. λ_i and μ_i are as follows: $\lambda_2 = 2 \cdot \lambda_1$, $\lambda_3 = 3 \cdot \lambda_1$, $\mu_i = 100 \cdot \lambda_1$. For didactic aspects, we have noted, at the beginning of each transition, the value of $\frac{q_{i\to j}}{\sum_k q_{i\to k}}$. The detail of the values of the Relevance Factor for each state is given below the state space.

For this illustrative system, an algebraic resolution is also possible. Let $P_i^n(t)$ and $P_i^f(t)$ be the probabilities for component C_i to be in the nominal mode or in the failure mode at date t. These probabilities are:

$$\begin{cases} P_i^n(t) = \dfrac{1}{\lambda_i + \mu_i}\left(\mu_i + \lambda_i\, e^{-(\lambda_i+\mu_i)t}\right) \\ P_i^f(t) = \dfrac{1}{\lambda_i + \mu_i}\left(\lambda_i - \lambda_i\, e^{-(\lambda_i+\mu_i)t}\right) \end{cases}$$

They are obtained by solving the following set of differential equations (Eq. 2) which corresponds to the Markov chain model of a reparable component.



$$\begin{cases} R(s_{init}) = 1.0 \\ R(s_1) = {}^3\!/_6 = 0.5 \\ R(s_2) = {}^2\!/_6 \approx 0.333 \\ R(s_3) = {}^1\!/_6 \approx 0.167 \\ R(s_4) = R(s_1) \times {}^2\!/_{103} = {}^1\!/_2 \times {}^2\!/_{103} \approx 9.71 \times 10^{-3} \\ R(s_5) = R(s_1) \times {}^1\!/_{103} = {}^1\!/_2 \times {}^1\!/_{103} \approx 4.85 \times 10^{-3} \\ R(s_6) = R(s_2) \times {}^1\!/_{104} = {}^1\!/_3 \times {}^1\!/_{104} \approx 3.21 \times 10^{-3} \\ R(s_7) = R(s_4) \times {}^1\!/_{201} = {}^1\!/_2 \times {}^2\!/_{103} \times {}^1\!/_{201} \approx 4.83 \times 10^{-5} \end{cases}$$

Fig. 3. Application of the proposed algorithm on a system with three independent components: C_1, C_2, C_3 ($\lambda_2 = 2 \cdot \lambda_1$, $\lambda_3 = 3 \cdot \lambda_1$, $\mu_i = 100 \cdot \lambda_1$)

$$\begin{cases} \dfrac{\partial}{\partial t} P_i^n(t) = -\lambda_i P_i^n(t) + \mu_i P_i^f(t) \\ \dfrac{\partial}{\partial t} P_i^f(t) = \lambda_i P_i^n(t) - \mu_i P_i^f(t) \\ P_i^n(0) = 1 \end{cases} \tag{2}$$

As components are independent, the probability to be in a state is calculated by composing previous probabilities. The set of equations (3) presents the probability to be in each state at $t = \infty$ according to the algebraic model.

$$\begin{cases} P_{s_{init}} = \dfrac{\mu_1}{\lambda_1+\mu_1} \times \dfrac{\mu_2}{\lambda_2+\mu_2} \times \dfrac{\mu_3}{\lambda_3+\mu_3} \approx 0.942 \\ P_{s_1} = \dfrac{\mu_1}{\lambda_1+\mu_1} \times \dfrac{\mu_2}{\lambda_2+\mu_2} \times \dfrac{\lambda_3}{\lambda_3+\mu_3} \approx 2.82 \times 10^{-2} \\ P_{s_2} = \dfrac{\mu_1}{\lambda_1+\mu_1} \times \dfrac{\lambda_2}{\lambda_2+\mu_2} \times \dfrac{\mu_3}{\lambda_3+\mu_3} \approx 1.88 \times 10^{-2} \\ P_{s_3} = \dfrac{\lambda_1}{\lambda_1+\mu_1} \times \dfrac{\mu_2}{\lambda_2+\mu_2} \times \dfrac{\mu_3}{\lambda_3+\mu_3} \approx 9.42 \times 10^{-3} \\ P_{s_4} = \dfrac{\mu_1}{\lambda_1+\mu_1} \times \dfrac{\lambda_2}{\lambda_2+\mu_2} \times \dfrac{\lambda_3}{\lambda_3+\mu_3} \approx 5.65 \times 10^{-4} \\ P_{s_5} = \dfrac{\lambda_1}{\lambda_1+\mu_1} \times \dfrac{\mu_2}{\lambda_2+\mu_2} \times \dfrac{\lambda_3}{\lambda_3+\mu_3} \approx 2.82 \times 10^{-4} \\ P_{s_6} = \dfrac{\lambda_1}{\lambda_1+\mu_1} \times \dfrac{\lambda_2}{\lambda_2+\mu_2} \times \dfrac{\mu_3}{\lambda_3+\mu_3} \approx 1.88 \times 10^{-4} \\ P_{s_7} = \dfrac{\lambda_1}{\lambda_1+\mu_1} \times \dfrac{\lambda_2}{\lambda_2+\mu_2} \times \dfrac{\lambda_3}{\lambda_3+\mu_3} \approx 5.65 \times 10^{-6} \end{cases} \tag{3}$$

For this example, it is important to note that the states sorted by their Relevance Factor are in the same order as states sorted by their steady-state probability at $t = \infty$.

Applied to this illustrative example, our method identifies without error the states with the highest steady-state probabilities.

3.4 Value of the Threshold

In our approach, the size of the partially built state space depends on the value of the threshold on the Relevance Factor. If it is too high, influential states will be discarded, and the assessment done with the limited Markov chain will be inaccurate. On the opposite, if the threshold is too low, lots of useless states will be explored, and time to build and then assess the chain will be high.

According to previous experiments, a threshold of 10^{-4} suits most problems. We currently analyze the practical reasons for which this value is well-adapted.

4. CASE STUDY

This section presents an academic case study for which the unreliability will be assessed with several techniques. The objective is to test our partial build method against other existing methods. This example comes from Malhotra and Trivedi (1995). It was used by Montani et al. (2006) to compare three safety tools that assess unreliability of a system, based on different approaches: DBNet (Montani et al., 2006), DRPFTproc (Bobbio and Raiteri, 2004) and Galileo (Dugan et al., 2000).

The safety model, technical data and proposed mission times come from Montani et al. (2006). To be able to evaluate the accuracy of our method on this academic example, we have assessed the unreliability of the system with the complete Markov chain (1056 states) and with the partial Markov chain (74 states) obtained by Algorithm 1 with a threshold fixed to 10^{-4}.

4.1 Structure of the studied system

The system (Fig. 4) is a multiprocessor computing system composed by two computing modules CM1 and CM2.

- CM1 consists of a processor P1, a memory M1, a primary hard disk D11 and a backup disk D12.
- CM2 is formed similarly with P2, M2, D21 and D22.
- M3 is a spare memory. It can replace M1 or M2 in case of failure, but not both.
- A unique bus connects CM1, CM2 and M3.
- The power supply PS is used by both processors.

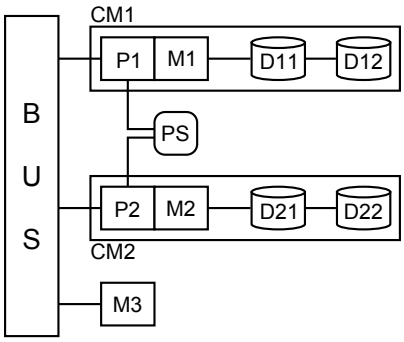


Fig. 4. Structure of the multiprocessor computing system (Montani et al., 2006)

The disks and the memory are *warm spare* components which deteriorate, even when unused. Table 1 regroups technical data of components.

Component	Failure rate (h^{-1})	Dormancy factor
BUS	2.0×10^{-9}	-
P1, P2	5.0×10^{-7}	-
PS	6.0×10^{-6}	-
D11, D12, D21, D22	8.0×10^{-5}	0.5
M1, M2, M3	3.0×10^{-8}	0.5

Table 1. Failures rates and dormancy factors

4.2 Results of the study

Table 2 shows the assessment of the system's unreliability for several mission times. Values given for DBNet, DRPFTproc, and Galileo come from Montani et al. (2006). We have assessed the unreliability with the complete Markov chain (1056 states) and the partial Markov chain (74 states). We have used the same amount of significant figures to be able to compare values.

For the mission times proposed for this academic example, results obtained with the five methods are very similar.

To deepen this first result, we measured the *percent error* of the unreliability given by the partial Markov chain (74 states) with respect to the result given by the complete Markov chain (1056 states), which is considered as the exact value. This percent error (δ) is defined as follows:

$$\delta = \left| \frac{v_{CompleteMC} - v_{PartialMC}}{v_{CompleteMC}} \right| \times 100$$

where v is the unreliability of the system.

Fig. 5 presents the evolution of δ according to the mission time. This plot is drawn from 1000 measures (one measure every 50 hours of the mission time).

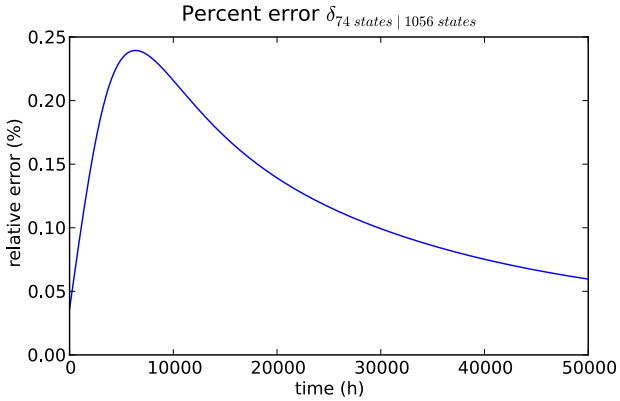


Fig. 5. Percent error δ of unreliability assessed with the partial Markov chain (74 states) with reference to unreliability assessed with the complete Markov chain (1056 states)

With a threshold value fixed to 10^{-4}, the percent error is always lower than 0.25%. The evolution of the percent error is induced by the presence of absorbing states in the Markov chain. These states are due to the presence of non-reparable components in this system.

For this example proposed by Malhotra and Trivedi (1995) and used by Montani et al. (2006) as benchmark, the partial Markov chain proposed by our algorithm is sufficient to assess safety indicators for preliminary system safety analysis.

Mission Time (hours)	Assessment tool				
	DBNet	DRPFTproc	Galileo	Full state space exploration (1056 states)	Limited state space exploration (74 states)
1,000	0.0060086	0.0060088	0.0060088	0.0060088	0.0060033
2,000	0.0122452	0.0122455	0.0122455	0.0122456	0.0122280
3,000	0.0191820	0.0191832	0.0191832	0.0191833	0.0191477
4,000	0.0273523	0.0273548	0.0273548	0.0273548	0.0272960
5,000	0.0372379	0.0372413	0.0372413	0.0372413	0.0371549

Table 2. Unreliability of the system, assessed with different tools.

5. CONCLUSION

In this article, we proposed a method to build approximate Markov chains from higher level description. The idea is to keep only the most representative states of the full chain. These states are selected by means of a Relevance Factor. The Relevance Factor of states is calculated on the fly thanks to an algorithm derived from Dijkstra's algorithm to calculate shortest paths in a graph. We showed on an already published test case made with 11 components, that it is possible to obtain the unreliability with a relative error lower than 0.25% while limiting state space to only 74 states instead of $1,056$. Applied to another system with 21 components, the state space limitation is even more impressive. The complete Markov chain has $962,552$ states and $10,768,622$ transitions. With a partial state space limited to only 131 states and 436 transitions, the unavailability of the system is obtained with a relative error of 0.6%.

As a future work, we plan to integrate our method in the AltaRica 3.0 project (Fig. 6) (Prosvirnova et al., 2013) and apply it onto industrial sized test cases.

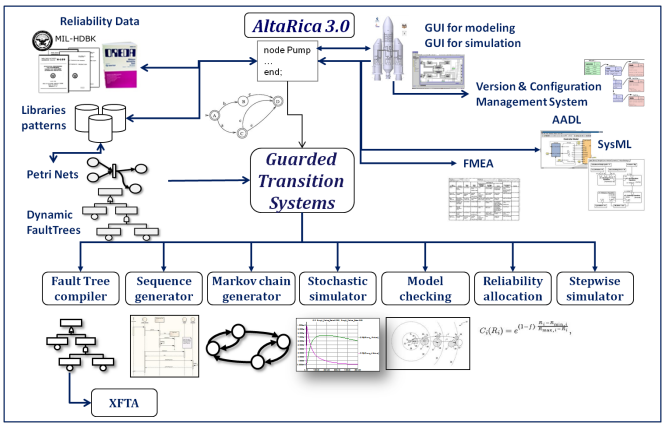


Fig. 6. Overview of the AltaRica 3.0 project

REFERENCES

Ajmone Marsan, M., Balbo, G., Conte, G., Donatelli, S., and Franceschinis, G. (1994). *Modelling with Generalized Stochastic Petri Nets*. Wiley Series in Parallel Computing. John Wiley and Sons.

Bobbio, A. and Raiteri, D.C. (2004). Parametric fault trees with dynamic gates and repair boxes. In *Reliability and Maintainability, 2004 Annual Symposium-RAMS*, 459–465. IEEE.

Bolch, G., Greiner, S., de Meer, H., and Trivedi, K.S. (2006). *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Wiley-Interscience.

Bouissou, M. and Bon, J. (2003). A new formalism that combines advantages of fault-trees and markov models: Boolean logic driven markov processes. *Reliability Engineering & System Safety*, 82(2), 149–163.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1), 269–271.

Dugan, J., Bavuso, S., and Boyd, M. (1992). Dynamic fault-tree models for fault-tolerant computer systems. *Reliability, IEEE Transactions on*, 41(3), 363–377.

Dugan, J.B., Sullivan, K.J., and Coppit, D. (2000). Developing a low-cost high-quality software tool for dynamic fault-tree analysis. *Reliability, IEEE Transactions on*, 49(1), 49–59.

Fourneau, J.M., Pekergin, N., and Youns, S. (2007). Censoring markov chains and stochastic bounds. In *Formal Methods and Stochastic Models for Performance Evaluation*, volume 4748 of *Lecture Notes in Computer Science*, 213–227. Springer Berlin Heidelberg.

Gudemann, M. and Ortmeier, F. (2011). Towards model-driven safety analysis. In *Dependable Control of Discrete Systems (DCDS), 2011 3rd International Workshop on*, 53–58. IEEE.

Lal, R. and Bhat, U. (1988). Reduced system algorithms for markov chains. *Management Science*, 34(10), 1202–1220.

Malhotra, M. and Trivedi, K.S. (1995). Dependability modeling using Petri-nets. *Reliability, IEEE Transactions on*, 44(3), 428–440.

Montani, S., Portinale, L., Bobbio, A., Varesio, M., and Codetta-Raiteri, D. (2006). A tool for automatically translating dynamic fault trees into dynamic bayesian networks. In *Reliability and Maintainability Symposium, 2006. RAMS'06. Annual*, 434–441. IEEE.

Pribadi, Y., Voeten, J., and Theelen, B. (2001). Reducing markov chains for performance evaluation. In *Proceedings of PROGRESS'01*.

Prosvirnova, T., Batteux, M., Brameret, P.A., Kloul, L., Cherfi, A., Friedlhuber, T., Roussel, J.M., and Rauzy, A. (2013). The altarica 3.0 project for model-based safety assessment. In *Dependable Control of Discrete Systems (DCDS), 2013 4th IFAC Workshop on*. IEEE. To be published.

Rauzy, A. (2004). An experimental study on iterative methods to compute transient solutions of large markov models. *Reliability Engineering & System Safety*, 86(1), 105–115.

Stewart, W. (1994). *Introduction to the numerical solution of Markov chains*, volume 1. Princeton University Press.

Zhao, Y.Q. and Liu, D. (1996). The censored markov chain and the best augmentation. *Journal of Applied Probability*, 33(3), 623–629.