Availability of CAUTRA, a Subset of the French Air Traffic Control System

Karama Kanoun, Marie Borrel, Thierry Morteveille, Alain Peytavin

Availability of CAUTRA,

a Subset of the French Air Traffic Control System

Karama Kanoun◊ **Marie Borrel**◊* **Thierry Morteveille**+ **Alain Peytavin**++

++CENA
7, Avenue Edouard Belin
31055 Toulouse Cedex - France

◊LAAS-CNRS
7, Avenue du Colonel Roche
31077 Toulouse Cedex 4 - France

+ SRTI SYSTEM
Rue Max Planck, B.P. 457
31315 Labège Cedex - France

* The work presented in this paper has been partially performed while Marie Borrel was with SRTI SYSTEM.

Abstract

The aim of our work is to provide a quantified means helping in the definition of a new architecture for CAUTRA, a subset of the French Air Traffic Control system. In this paper we define a set of alternative architectures, give some elements for constructing their dependability models and compare their availability. Modeling is carried out following a modular and systematic approach, based on the derivation of block models at a high level of abstraction. In a second step, the blocks are replaced by their equivalent Generalized Stochastic Petri Nets to build up the detailed model of the architecture. The evaluations performed permit identification of a subset of architectures whose availability meets the dependability requirements and also identification of the best architecture among this subset.

Key Words: Dependability modeling, Generalized Stochastic Petri Nets, Markov Chains, model composition

1. Introduction

Due to the growth in air traffic and the saturation of computational facilities, the French Directorate of Air Navigation has commissioned the design and implementation of new architectures for the Air Traffic Control system (ATC). The work presented in this paper is part of an overall program aimed at ATC automation. It more specifically addresses the sub-system of CAUTRA (Coordinateur AUtomatique du TRafic Aérien) located in a Regional Center for Air Navigation. The ultimate aim is to provide a quantified means helping in the definition of a new architecture. To do this, starting from the current architecture, we define alternative architectures whose availability is compared. CAUTRA is a distributed fault-tolerant system whose functions are vital to ATC. It specifically belongs to the category of real-time computer systems demanding a high level of availability. Fault tolerance techniques (i.e., hardware redundancy and software replication) enable the availability of this system to be at the required level. The complexity of the system behavior results from

several interactions between hardware components and software replicas. These interactions induce dependencies that are usually stochastic in nature making modeling difficult: as the associated models have to account for the components' behavior and their interactions. We follow a modular and structured modeling approach based on Generalized Stochastic Petri Nets (GSPNs) in which interactions are explicitly considered. Emphasis is placed on clearly defining the interactions between the hardware and software components and on deriving GSPNs that are as generic as possible so as to be used in the largest number of alternatives. More details can be found in [1-3])

Several papers dealt with the dependability of ATC systems. These are mainly focused on the specification or design of fault tolerance procedures [4-7] , while papers on dependability evaluation are few (see e. g., [8, 9] where a qualitative evaluation of an ATC system is performed). Likewise, a number of papers dealing with performance and dependability modeling of real-life systems using Markov chains, GSPNs or their offspring's have been published [10-17] . Most of them only consider the hardware part of a system and, when the software is considered, the interactions between hardware and software are not explicitly modeled. Even though dependability evaluation of combined hardware and software systems are not yet of current practice, some relevant papers have been published [18-21] .

The paper is organized in seven sections. Section 2 describes CAUTRA. The modeling approach and assumptions are presented in Section 3. Example of GSPNs are given in Section 4. Section 5 outlines the benefits and drawbacks of the modeling approach. Section 6 discusses some numerical results.

2. System description

CAUTRA gathers together the computerized processing means for flight plans and the radar data of a Regional Center for Air Navigation. Two main functions can be distinguished: Flight Plan Processing (FPP) and Radar Data Processing (RDP). The former handles and updates the flight plans in the Regional Center. It provides air traffic controllers with the data regarding the planes crossing their airspace and handles any information that may be supplied to other air traffic controllers. Based on the radar data, the RDP builds up a synthetic picture representing the air traffic situation. Through the RDP-FPP dialogue, flight plan correlation allows the RDP to enhance the picture by supplying more data derived from the flight plans to those planes detected by the radar. We first present the current architecture of CAUTRA (which was in use at the beginning of the study); then the alternatives are defined taking the latter as reference.

The **current architecture** comprises two redundant Data General computers, DG1 and DG2. The software of each application is replicated leading to four replicas: RDP principal (RDPpal), RDP standby or secondary (RDPsec), FPP principal (FPPpal) and FPP standby (FPPsec). Replicas are distributed as follows:

RDPpal and FPPsec run on the same computer while RDPsec and FPPpal run on the other. FPPpal carries out a preliminary processing which is transmitted to FPPsec; also FPPpal dialogs both ways with RDP replicas for the flight plan correlation.

Error processing in each replica (provided by exception handling mechanisms) allows recovery of temporary software faults. The permanent faults in pal replica are tolerated by switching from the replica pal to sec. Reconfiguration is carried out as follows: after switching the roles of the replicas and restart of the sec replica, the replicas switch back to resume their initial roles as shown in Figure 1. Also, the switch of RDPpal is required after the communication medium failure without FPPpal failure.

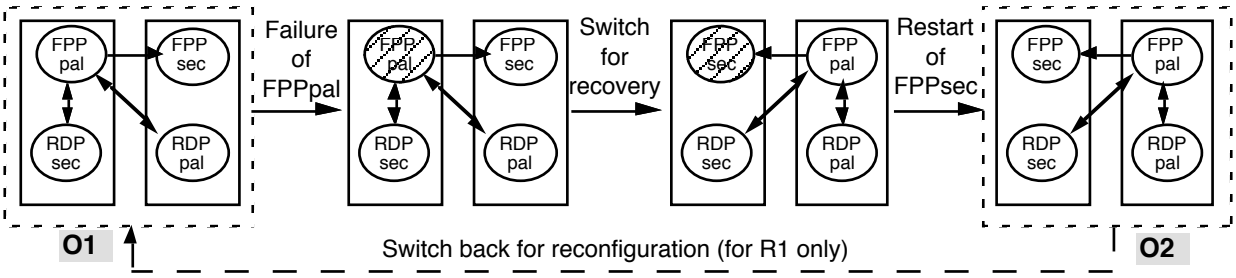


Figure 1: Reconfiguration after software failure

Starting from this architecture, several **alternatives** are proposed with the same composition but differing in terms of organization (distribution of the software replicas onto the hardware computers), the software reconfiguration and the hardware fault tolerance procedures.

Based on the current **organization**, denoted O1, we define organization O2 in which RDPpal and FPPpal are hosted by the same computer. Both O1 and O2 are indicated on figure 1; the main difference lies in the communication means between RDPpal and FPPpal: for O2 this communication is internal whereas for O1 it is achieved through an external communication medium.

The existing **software reconfiguration** procedure, denoted R1, requires switching back for reconfiguration. A second reconfiguration procedure is defined, denoted R2, in which the replicas retain their new roles after replica sec restart (the switch back from O2 to O1 in figure 1 is not performed). Note that when R2 is considered, whatever the initial organization, the system alternates between O1 and O2 after recovery of software permanent failures.

The current hardware fault tolerance strategy, referred to as strategy S0, is based on two redundant computers. The presence of a third computer in the center allocated to background tasks can be taken advantage of and used as a spare. Two additional **hardware fault tolerance** procedures (S1 and S2) corresponding to using the spare respectively after the first computer failure or after failure of both of them are considered. Table 1 summarizes the considered procedures. Combining the 2 organizations with the 2

reconfigurations and the 3 hardware fault tolerance procedures leads to 12 alternatives whose availability will be evaluated.

Table 1: Definition of the considered procedures

Organization

O1	FPPpal and RDPsec onto a computer, FPPsec and RDPpal onto the other
O2	FPPpal and RDPpal onto a computer, FPPsec and RDPsec onto the other

Software reconfiguration

R1	after switching the replica roles and restart of sec, the replicas switch back for reconfiguration
R2	after switching the replica roles and restart of sec, the replicas retain their new roles

Hardware fault tolerance

S0	current hardware fault tolerance procedure (0 spare)
S1	switching of the spare to the main applications is performed after 1 failure (DG1 or DG2 failure)
S2	switching of the spare to the main applications is performed after 2 failures (DG1 and DG2 failures)

3. Modeling approach and assumptions

It is assumed that software and hardware components are in stable reliability, i.e., the failure rates are constant. With respect to the rates associated with hardware maintenance, software restart and fault tolerance procedures, the duration of these procedures is short relative to the times to failures, and previous studies have shown constant rates to be a good assumption [22] . The evaluations are thus based on Markov processes. However, model construction is based on GSPNs due to their ability to cope with modularity and hierarchy. The measures evaluated are unavailability of RDP and unavailability of FPP.

The difficulty of modeling stems from the conjunction of two things: the numerous interactions between the components and the number of alternative architectures to be modeled. Using an ad hoc approach would be cumbersome. These considerations urged us to use a modular and hierarchical approach, taking advantage of the similarities between the alternatives. This is still reinforced by the fact that the alternatives have the same composition and many interactions between components are the same in many alternatives.

From the system composition and the interactions between components, a high level behavioral model (referred to as block model) composed of blocks linked by arrows is derived for each alternative. A block represents a GSPN describing either a component behavior (component net) or an interaction (dependency net); the arrows indicate the direction of the links between the nets. The GSPNs of the blocks are derived in a second step to form the global model; i.e., the GSPN of an architecture is thus obtained by composition of the GSPNs of the components with those representing their interactions. The block models of all alternatives are derived together to clearly identify the blocks that are common or similar in several alternatives, for re-use purpose. Re-use of the blocks is one of the main advantages of our approach. Also, this approach allows for a

progressive construction of the global model so as to master its complexity as in [14, 17] . The first step in constructing the models consists thus in identifying the interactions between the components.

3.1 Interactions between the components

The interactions between hardware and software components are directly related to the assumptions made about their behavior. Owing to the importance of the impact of temporary faults on the behavior of hardware and software components, both permanent and temporary faults are considered. It is assumed that the activation of a fault may lead to the following dependencies:

- Following activation of a hardware fault:
  - an error due to the activation of a temporary hardware fault may propagate to the hosted software replicas,
  - an error due to the activation of a permanent fault in a computer leads to ***stop*** the hosted software replicas that are restarted after the end of repair.
- Following activation of a software fault: owing to the dialog between the replicas, an error in a replica due to a permanent fault — usually referred to as solid fault — may ***propagate*** to the replicas with which it dialogs [23] (it is assumed that errors due to temporary faults — usually referred to as soft faults — are confined and do not propagate to the other replicas). It is worth noting that replicas pal and sec do not perform exactly the same tasks at a given time. It is thus assumed that common mode failures are induced by error propagation due to the dialog between the replicas.
- Following failures of the communication medium: the system has to switch from organization O1 to O2 if it is in O1, and the switching from O2 to O1 is not allowed as long as the communication is in failure.

Dependencies induced by fault tolerance and maintenance procedures are as follows:

- Between two software replicas: dependency due to fault tolerance of permanent software faults, i.e., ***reconfiguration*** from sec to pal, following RDPpal or FPPpal failure.
- Between two hardware computers: dependency due to ***fault tolerance*** and ***repair***.
- Between all components: coordination of fault tolerance and maintenance actions to form a global ***recovery strategy*** when several components are in failure. For example, in case of failure of the computer hosting pal and failure of sec, sec is restarted as pal, the new sec is restarted after computer repair.

3.2 Block models of the architectures

Consider first the **current architecture**. Table 2 lists the name of the nets associated with the component blocks. Dependency blocks are directly derived from the dependencies identified above. Figure 2 shows the block models for the RDP and FPP: all blocks of RDP are used for FPP as well, however FPP has two extra

blocks, (2xN'_{Prop}) for error propagation between replicas. Models of figure 2 are obtained by **composition** of the block nets introduced in table 2. Due to the exchange of information between RDP and FPP (with possible error propagation), the dependability of RDP and FPP cannot be evaluated separately: a global model has to be derived as shown in figure 3. The latter is obtained by composing the models of figure 2 and making the following modifications.

- The set of blocks (N_{DG1}, N_{DG2}, N_{Rep}) associated with the computers is considered only once.

Table 2: Component and dependency nets

N_{DG1} , N_{DG2}	model computers DG1 and DG2 behavior (identical)
N_{RDPi}, N_{FPPi}	model RDP and FPP replicas behavior (identical)
N_C	models communication medium behavior
N_{Prop}	models the propagation of a hardware error to the hosted software replica
N_{Stop}	models the software stop after activation of a permanent fault in the hosting hardware
N'_{Prop}	models propagation of a software error to a communicating software replica
N_{RecRDP}, N_{RecFPP} N'_{RecRDP}, N'_{RecFPP}	model RDP and FPP software reconfiguration from sec to pal
N_{Rep} , N'_{Rep}	model hardware repair (sharing of a repair man)
N_{SynRDP}, N_{SynFPP}	model the synchronization between hardware and software recovery actions
N_{Req}	models the switch request following the communication medium failure
N_{Strat}	models the global reconfiguration strategy according to the states of all resources
N_{R1}	models the switch back in the case of reconfiguration R1

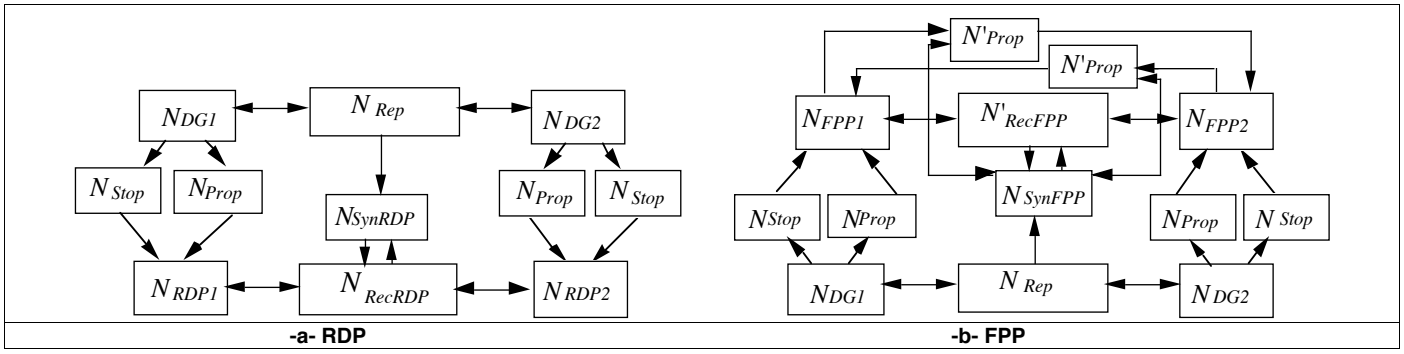


Figure 2: Block models of RDP and FPP

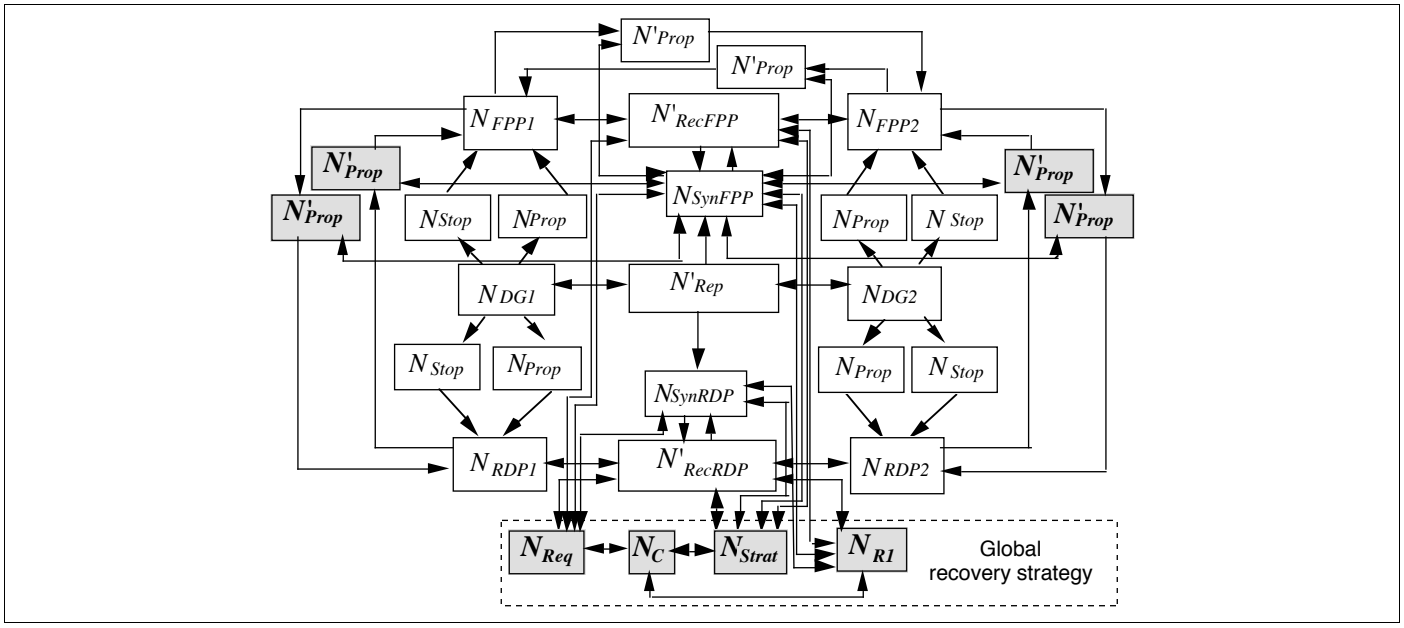


Figure 3: Block model of the current architecture (A.1.1.0)

- N_{Rep}, N_{RecRDP} and N_{RecFPP} have to be adapted to take into account the co-existence of two replicas on the same computer; they are denoted respectively N'_{Rep}, N'_{RecRDP} and N'_{RecFPP}.
- The following blocks are added: error propagation between RDP and FPP replicas (4xN'_{Prop}), the recovery strategy (N_{Req}, N_{Strat}, N_{R1}) and the communication medium, N_C.

This model has been derived after some iterations and refinements between the block models of the various alternatives. In particular, the global recovery strategy could have been modeled by a single block. It has been split into three blocks (N_{Req}, N_{Strat}, N_{R1}) only for re-usability and clarity: N_{Strat} and N_{Req} are the same for all architectures; they are distinguished only because their roles are different; N_{R1} is to be omitted for R2 .

With respect to the **other alternative architectures**, using the notations of table 1, let A.o.r.s denote an architecture with o = O1, O2; r = R1, R2; s = S0, S1, S2. For the sake of simplicity, this is shortened as: o = 1, 2; r = 1, 2; s = 0, 1, 2. We give hereafter the list of blocks that have to be added, removed or simply adapted for each architecture, taking the current one, A.1.1.0, as reference (figure 3):

- A.2.1.0: adapt N_{R1} and N_{SynFPP} to comply with the O2 assumption.
- A.1.2.0 and A.2.2.0: remove N_{R1} (which is associated with R1 only), and adapt N_{SynFPP} for A.2.2.0 to comply with the O2 assumption.
- Set (R2, S1 and R2, S2) = {A.1.2.1, A.2.2.1, A.1.2.2 and A.2.2.2}: remove N_{R1}, adapt N_{SynFPP} for A.2.2.1 and A.2.2.2 to comply with the O2 assumption, adapt N'_{Rep} and add N_{Spare}.

- Set (R1, S1 and R1, S2) = {A.1.1.1, A.2.1.1, A.1.1.2 and A.2.1.2}: adapt N_{R1} and N_{SynFPP} for A.2.1.1 and A.2.1.2 according to the O2 assumption, adapt N'_{Rep} and add a block corresponding to the behavior of the spare N_{Spare} which is directly linked to the new N'_{Rep}.

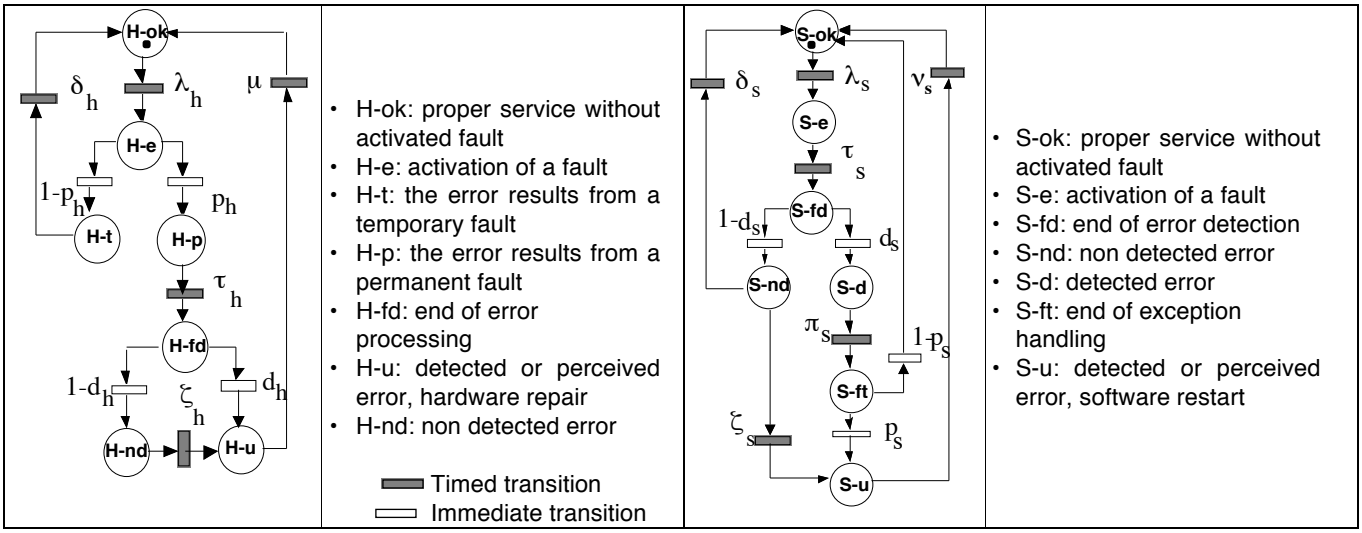
As a result, modeling the 11 other architectures requires only the addition of one component net (N_{Spare}) and adaptation of 6 dependency nets: those associated with the reconfiguration and fault tolerance strategy.

4. GSPN construction

To assist re-usability, we defined rules for constructing the GSPNs of the blocks. Together with the conventional rules of GSPNs, these rules allow a) for an easy interfacing of the dependency nets — a prerequisite for modularity, hierarchical modeling and re-usability, b) controlling the token creation and absorption in order to generate from the beginning nets that are bounded [1-3] . The main rules are summarized hereafter.

- A **component net** models the behavior of a component as resulting from the activation of its own faults and from local error detection, fault tolerance mechanisms, and restoration actions. It is bounded and alive.
- A **dependency net** (DN) is linked to at least an initializing and a target net; it is defined according to specific rules, among which:
  - It is activated by an initializing net, via entry place(s) whose initial marking is 0, after firing of specific transitions (firing transitions) in the initializing net.
  - It has well defined interfaces with target net(s).
  - At the initialization of the DN, an additional token is generated, it has to be absorbed when leaving the net;
  - DNs should not alter the basic structure of the component nets (i.e., a component net has always the same places and the same transitions between these places, the only interfaces with the DNs are constituted by additional arcs). This is because several dependency nets can be connected to the same component net.

The **component nets** are given in Figure 4. The difference between these nets lies in that for hardware, temporary and permanent faults are differentiated by their respective consequences following activation, whereas for software, they can only be distinguished after specific processing.



-a- N_{DG1} , N_{DG2} -b- N_{RDP1}, N_{RDP2}

Figure 4: Hardware computer and software replica nets

The hardware model (for computers DG1 and DG2) is based on the following assumptions:

- Faults are activated with rate λ_h.
- With probability p_h the fault is permanent, (probability of a temporary fault $(1-p_h)$).
- An error due to a permanent fault is either detected with probability d_h, or non detected $(1-d_h)$; error processing rate: τ_h.
- The effects of an error due to a temporary fault are eliminated within a short time $1/\delta_h$.
- The effects of a non detected error resulting from a permanent fault may be perceived later (rate ζ_h).
- The repair rate is μ.

Equivalent assumptions are made regarding the behavior of the software replicas:

- Faults are activated with rate λ_s (that can be specified as λ_{RDP} and λ_{FPP} respectively for RDP and FPP).
- An error is either detected with probability d_s, or non detected $(1-d_s)$; detection rate τ_s.
- The detected error is processed by means of exception handling mechanisms during a short time $1/\pi_s$. At the end of error processing, 1) if the fault is temporary (probability $(1-p_s)$) its effects are eliminated and the software resumes its normal mode of operation, or 2) if the fault is permanent (probability p_s); the software has to be restarted (restart rate: ν_s, denoted ν_{FPP}, ν_{RDP} for FPP and RDP) to eliminate its effects.
- The effects of a non detected error may be eliminated (rate δ_s), or perceived (perception rate ζ_s), in which case the software replica has to be restarted.

Due to space limitations and to the complexity of the dependency nets [2] , only the **error propagation net** is shown in Figure 5 (this is the smallest and easiest one). It is initialized via the entry place Prop after

firing of transition 1-d_h (non detected error) or of transition 1-p_h (an error due to a temporary fault) in the hardware net. With probability 1-p_{ph}, the error is not propagated and with p_{ph} it is propagated. The effect on the software replica net is to move the token from S-ok to S-e if it is in S-ok; processing of the induced error is carried out in the same way as when the fault is activated without propagation (i.e., through λ_s in figure 4-b).

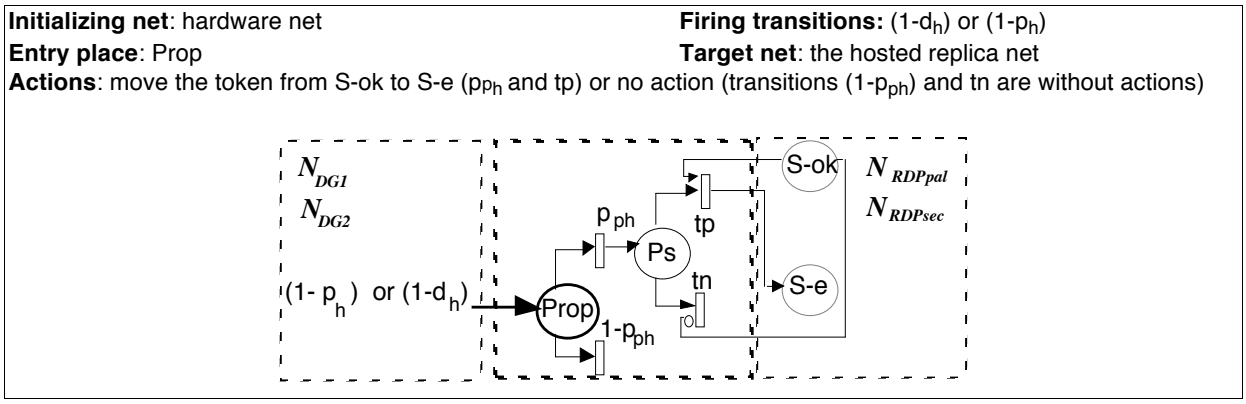


Figure 5: Error propagation net, N_{Prop}

5. Benefits and drawbacks

One of the major benefits of the approach is the ease of model validation due to the rules that have been imposed for block composition. For instance, the rules controlling token creation and absorption when entering and leaving a dependency net allow construction of a global model that is bounded. Each dependency net is validated with its initializing and target nets. Possible problems can thus be detected early before building up the whole GSPN. The resulting models can be processed by any tool for dependability evaluation based on Petri nets. In our case, we used SURF-2 [24] for model construction and validation.

Dependency nets are validated only once when they are re-used without adaptation for other alternatives. By way of example, consider the RDP model of figure 2-a. The validation of the global model can be carried out as follows: N_{Prop} is validated with N_{DG1} and N_{RDP1}; N_{Stop} is validated with N_{DG1} and N_{RDP1}; N_{Rep} is validated with N_{DG1} and N_{DG2}; N_{RecRDP} is validated with N_{RDP1} and N_{RDP2}, and N_{SynRDP} is to be validated with all the other GSPNs (the whole model, in which all the blocks except N_{SynRDP} have been validated).

Model composition and validation are facilitated by re-usability (with or without adaptation). Indeed, the explicit modeling of the interactions leads us to analyze them in detail and generally to fractionate them into elementary operations. Most of the time, the adaptation of a GSPN in order to model similar interactions consists in modifying a part of these elementary operations. Also, re-use can be supported by the creation of a library from which the elementary nets can be automatically drawn. This reinforces further the advantages of the modeling approach.

Due to the presence of several immediate transitions, the state space of the resulting reachability set is very large. Fortunately, several techniques are available for suppressing the immediate transitions, (see e.g., [25-27]). Another problem arises from the existence of fast transitions, leading to stiff Markov chains and making it more difficult to evaluate dependability. The state aggregation technique proposed in [28] and the place aggregation technique achieved at the GSPN level in [25] allow a non-stiff Markov chain with a smaller state space to be obtained.

6. Some results

Considering the models of CAUTRA alternatives, using the composition rules of Section 4 and suppressing fast transitions corresponding to error detection and processing rates [τ_h, δ_h and τ_s, π_s, δ_s] (as the duration of the associated events are in the range of seconds, to be compared with the times to failures), the number of states of the Markov chains obtained is 104 states for the architectures without a spare and 390 states for those with a spare. Several verifications have been performed to check their validity before model processing and also by sensitivity studies for semantic verifications. Examples of such studies are given in the remainder of the section. However, additional parameters have to be introduced as they do not appear in the previous models:

- β_h, the switching rate of DG1 or DG2 onto the spare and c_h, the associated coverage factor
- β_{FPP}, the switching rate from FPPpal to FPPsec, β_{RDP} its equivalent and c_s, the associated coverage factor
- λ_C the failure rate of the communication medium.

With respect to the numerical values, some of them are derived from observations on the current system and the others are assigned nominal values from which sensitivity studies are performed. Unless otherwise stated, the numerical values considered are those of Tables 3 and 4. Note that the nominal permanent failure rate is $2\ 10^{-4}$ /h for the computers and RDP replicas, and 10^{-3} / h for FPP replicas. The difference in the RDP and FPP switching times is due to the fact that in the current system, RDP switching is automatic, whereas for FPP it is performed after the operator acknowledgment. Also, It is not possible to have an FPP restart time less than 10 to 5 minutes owing to the data stream: it is created once, infrequently updated and difficult to reconstitute, while the RDP data stream is more frequently created, updated at a faster rate, and easier to reconstitute, leading to a short restart time.

Table 3: Nominal values of transition rates

λ_h	λ_{RDP} (λ_s)	λ_{FPP} (λ_s)	λ_C	1 / μ
0.01 / h	0.01 / h	0.05 / h	10^{-5} /h	10 h
1 / ν_{RDP}	1 / ν_{FPP}	1 / β_{RDP}	1 / β_{FPP}	1 / β_h
1 min	10 min	1 s	1 min	1 min

Table 4: Nominal values of probabilities

p_h	p_s	p_{ph}	p_{ps}	c_s	c_h	d_h	d_s
0.02	0.02	0.85	0.7	0.98	0.95	1	1

In the rest of the section, we first give the unavailability figures for the defined architectures, then we show some examples of sensitivity analyses.

6.1 Comparison of the alternative architectures

Tables 5 and 6 show the FPP and RDP unavailability (UA) of the twelve alternative architectures using the nominal values of the parameters. These tables deserve several comments.

- The FPP unavailability is much more higher than that of RDP. This difference is due to three factors: the software failure rate and the software switching and restart rates (if the switching fails or after the two replicas' failure). All of them are better for RDP, as discussed above.
- For both FPP and RDP, the hardware fault tolerance procedure is the most influent factor. However it impacts RDP more than FPP. For FPP, using a spare computer does not reduce unavailability by more than 20%, even for S1 where the spare is used after the failure of the first computer. On the other hand, the unavailability of RDP is divided by almost 10 from S2 to S1. Therefore, using a spare, it is more useful to switch after the failure of DG1 or DG2 than to wait for both DG1 and DG2 to fail.
- For both RDP and FPP, with R2 the organization has no impact, whatever the hardware fault tolerance procedure. This is due to the fact that after a long operational time, and without a strategy based on systematically exchanging the role of replicas after system reconfiguration to retain the initial distribution of the replicas, the time spent in O1 is equivalent to that spent in O2.
- For RDP, the software reconfiguration impacts differently the unavailability for both organizations, whatever the hardware fault tolerance procedure. For O1, moving from R1 to R2 decreases the unavailability, while for O2, it increases it . Thus, it may be inferred that for O1, the best reconfiguration is R2, and for O2, R1.
- Note that the dependability requirement of less than 5 min per year for the RDP is met only for S1 whatever the organization and the software reconfiguration.

Table 5: Unavailability (UA) per year of FPP

Hard. fault tolerance	A.1.1.s	A.2.1.s	A.1.2.s	A.2.2.s
Without a spare (s = 0)	2 h 42 min	2 h 41 min	2 h 42 min	2 h 42 min
With a spare (s = 1)	2 h 17 min	2 h 17 min	2 h 17 min	2 h 17 min
With a spare (s = 2)	2 h 36 min	2 h 36 min	2 h 36 min	2 h 36 min

Table 6: Unavailability per year of RDP

Hard. fault tolerance	A.1.1.s	A.2.1.s	A.1.2.s	A.2.2.s

Without a spare (s = 0)	21 min 36 s	20 min 49 s	21 min 11 s	21 min 11 s
With a spare (s = 1)	1 min 48 s	1 min 25 s	1 min 36 s	1 min 36 s
With a spare (s = 2)	16 min 18 s	15 min 40 s	15 min 56 s	15 min 56s

6.2 Sensitivity analyses

Hardware coverage factor and repair duration

Table 7 shows that RDP unavailability is sensitive to the hardware fault tolerance coverage c_h, only for S1, because switching after the failure of DG1 or DG2 occurs more frequently than switching after the failure of DG1 and DG2. Tables 6 and 7 show that for the nominal value of repair duration (10 hours) the unavailability of S1 and S2 is always lower than that of S0. Indeed fault tolerance strategy is influenced by the coverage factor, c_h, and at the same time by the repair duration, $1/\mu$ as shown by table 8 and figure 6. When the repair duration decreases, table 8 shows that the unavailability of S0 can be slightly lower than that of S1: for $c_h = 0.5$, S0 is better than S1 for a repair duration less than 2h, whereas for $c_h = 0.8$, S0 is better when the repair duration is less than 1h 30 min (the results are given for organization O1 and reconfiguration R2). Figure 6 and table 8 allow quantification of the tradeoff between using a spare or decreasing the repair duration.

Table 7: RDP UA per year for S1 and S2 according to c_h

c_h	A.1.1.1	A.2.1.1	A.1.2.1 / A.2.2.1
0.7	6 min 44 s	6 min 34 s	6 min 07 s
0.9	**2 min 32 s**	**2 min 27 s**	**2 min 29 s**
0.95	**1 min 29 s**	**1 min 25 s**	**1 min 27 s**
	A.1.1.2	A.2.1.2	A.1.2.2 / A.2.2.2
0.7	15 min 59 s	15 min 37 s	15 min 49 s
0.9	15 min 59 s	15 min 37 s	15 min 49 s
0.95	15 min 59 s	15 min 37 s	15 min 46 s

Software restart and switching durations

One of the main objectives is to decrease the FPP unavailability as much as possible. As we have stated earlier, in addition to the failure rate, the availability is limited by the duration of switching from sec to pal and by the restart time.

Table 8: RDP UA for S0 and S1 according to c_h and $1/\mu$

$1/\mu$	c_h	A.2.1.1	A.2.1.0
1 h	0.5	2 min 20 s	2 min 16 s
	0.8	2 min 18 s	2 min 16 s
1 h15	0.5	2 min 51 s	2 min 45 s
	0.8	2 min 47 s	2 min 45 s

2 h	0.5	4 min 21 s	4 min 20 s
	0.8	4 min 07 s	4 min 20 s

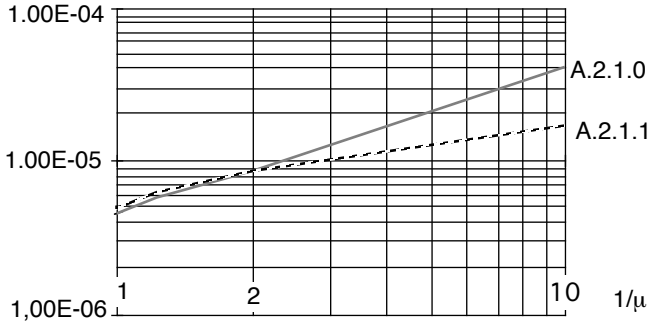


Figure 6: RDP UA for c_h= 0.5

Tables 9 and 10 show that reducing switching time is more worthwhile than reducing the restart duration. Table 9 shows that if FPP switching time were the same as that of RDP (1 s), unavailability would be almost 1 h per year for S0 and half an hour per year for S1. From 10 min to 6 min the unavailability gain is about 30%.

Table 9: FPP UA according to $1/\beta_{FPP}$ ($1/\nu_{FPP}$ = 10 min)

$1/\beta_{FPP}$	A.2.1.0	A.2.1.1	A.2.1.2
6 min	11 h 33 min	11 h 02 min	11 h 28 min
1 min	**2 h 41 min**	**2 h 17 min**	**2 h 36 min**
1 s	56 min 46 s	33 min	50 min 59 s

Table 10: FPP UA according to $1/\nu_{FPP}$ ($1/\beta_{FPP}$ = 1 min)

$1/\nu_{FPP}$	A.2.1.0	A.2.1.1	A.2.1.2
1 h	6 h 20 min	5 h 34 min	6 h 10 min
10 min	**2 h 41 min**	**2 h17 min**	**2 h 36 min**
6 min	2 h 27 min	2 h 05 min	2 h 22 min

7. Conclusion

The number of alternatives considered for CAUTRA and the complexity of the models induced by the numerous interactions between the components urged us to follow a modular and systematic approach which is particularly efficient for modeling several alternatives and well-suited for mastering this complexity. For instance, we have shown that the modeling of 11 alternatives requires only the addition of a component net to those of the current architecture and adaptation of six dependency nets among them (those related to reconfiguration and fault tolerance strategy). Even if building up the block models and validating the block's GSPNs (designed according to specific rules and to be as generic as possible) is time consuming, it is still worthwhile however since the time saved and the amount of confidence gained in creating and validating the overall model is not commensurable.

The overall results provide significant information about CAUTRA behavior. They highlight the particular importance of selecting a global fault tolerance and maintenance strategy right from the beginning. We have shown that only the architectures using a spare with switching on the spare after the failure of the first computer failure meet the requirement of less than 5 min for RDP unavailability. Among these, the best one is A.2.1.1 where the two principal replicas run onto the same computer (organization O2) and where the replicas switch back their roles after failure of the principal software replica and restart (reconfiguration R1). The results obtained from the whole study have been used for the definition of the new CAUTRA architecture.

Acknowledgments: the authors would like to thank Jean Arlat, Alain Costes, Mohamed Kaâniche and Jean-Claude Laprie for their useful comments when reading earlier versions of this paper.

References

[1] M. Borrel, "Interactions between Hardware and Software Components: Characterization, Formalization and Modeling — Application to CAUTRA Dependability", INP, Toulouse, France, PhD Dissertation (in French) 1996.

[2] K. Kanoun and M. Borrel, "Dependability of Fault-tolerant Systems — Explicit Modeling of the Interactions Between Hardware and Software Components", Proc. *2nd IEEE Int. Computer Performance and Dependability Symposium (IPDS)*, Urbana-Champaign, IL, USA, 1996, pp. 252-261.

[3] K. Kanoun, M. Borrel, T. Moreteveille, and A. Peytavin, "Modeling the Dependability of CAUTRA, a Subset of the French Air Traffic Control System", Proc. *26th IEEE Int. Symp. Fault-Tolerant Computing (FTCS-26)*, Sendai, Japan, 1996, pp. 106-115. Reduced version of LAAS-report 95-515.

[4] A. Avizienis and D. E. Ball, "On the Achievement of a High Dependable and Fault-Tolerant Air Traffic Control System", *IEEE Computer*, vol. 20, pp. 84-90, 1987.

[5] V. R. Hunt and G. V. Kloster, "The Federal Aviation Administration's Advanced Automation Program", *IEEE Computer*, vol. 20, pp. 14-17, 1987.

[6] F. Cristian, B. Dancey, and J. Dehn, "Fault-Tolerance in the Advanced Automation System", Proc. *20th IEEE Int. Symp; on Fault-Tolerant Computing*, Newcastle, UK, 1990, pp. 6-17.

[7] E. Amadio, P. Iaboni, M. Lamanna, and P. Mariano, "Implementation of High Availability Mechanisms in the Air Traffic Control SIR-S System", Proc. *24th IEEE Int. Symp. on Fault-Tolerant Computing*, Austin, Texas, USA, 1994, pp. 134-136.

[8] J.-M. Garot and T. Hawker, "Evaluating Proposed Architectures for the FAA's Advanced Automation System", *IEEE Computer*, vol. 20, pp. 33-45, 1987.

[9] N. Fota, M. Kâaniche, and K. Kanoun, "Dependability Evaluation of an Air Traffic Control System", Proc. *3rd IEEE Int. Computer Performance & Dependability Symposium (IPDS)*, Durham, NC, 1998, pp. 206-215.

[10] C. Chen, H. Asada, Y. Kakuda, and T. Kikuno, "Comparison of Hybrid Modular Redundant Multiprocessor Systems with respect to Performabilities", Proc. *23rd IEEE Int. Symp. Fault-Tolerant Computing*, Toulouse, France, 1993, pp. 66-75.

[11] G. E. Stark, "Dependability Evaluation of Integrated Hardware/Software Systems", *IEEE Trans. on Reliability*, vol. R-36, pp. 440-444, 1987.

[12] P. I. Pignal, "An Analysis of Hardware and Software Availability Exemplified on the IBM-3725 Communication Controller", *IBM Journal of Research and Development*, vol. 32, pp. 268-278, 1988.

[13] J. F. Meyer, K. H. Muralidhar, and W. H. Sanders, "Performability of a Token Network under Transient Fault Conditions", Proc. *19th IEEE Int. Symp. on Fault-Tolerant Computing*, Chicago, Illinois, USA, 1989, pp. 175-182.

[14] J. K. Muppala, A. Sathaye, R. Howe, C, and K. S. Trivedi, "Dependability Modeling of a Heterogeneous VAX-cluster System Using Stochastic Reward Nets", in *Hardware and Software Fault Tolerance in Parallel Computing Systems*, D. R. Avresky, Ed., 1992, pp. 33-59.

[15] K. H. Prodromides and W. H. Sanders, "Performability Evaluation of CSMA/CD & CSMA/DCR Protocols Under Transient Faul Conditions", *IEEE Trans. on Reliability*, vol. 42 (1), pp. 116-127, 1993.

[16] L. A. Tomek and K. S. Trivedi, "Analysis Using Stochastic Reward Nets", in *Software Fault Tolerance*, M. Lyu, Ed., J. Wiley, 1995, pp. 138-165.

[17] W. Sanders and J. Meyer, "Reduced Base Model Construction Methods for Stochastic Activity Networks", *IEEE Trans. on Selected Areas in Communications*, vol. 9 (1), pp. 25-36, 1991.

[18] A. Costes, C. Landrault, and J.-C. Laprie, "Reliability and Availability Models for Maintained Systems Featuring Hardware Failures and Design Faults", *IEEE Trans. on Computers*, vol. C-27, pp. 548-560, 1978.

[19] J.-C. Laprie, C. Béounes, M. Kaâniche, and K. Kanoun, "The Transformation Approach to Modeling and Evaluation of Reliability and Availability Growth of Systems", Proc. *20th IEEE Int. Symp. Fault-Tolerant Computing*, Newcastle, UK, 1990, pp. 364-371.

[20] J.-C. Laprie and K. Kanoun, "X-ware Reliability and Availability Modeling", *IEEE Trans. on Software Engineering*, vol. SE-18, pp. 130-147, 1992.

[21] J. B. Dugan and M. Lyu, "System-level Reliability and Sensitivity Analysis for Three Fault-tolerant Architectures", Proc. *4th IFIP Int. Conference on Dependable Computing for Critical Applications*, San Diego, 1994, pp. 295-307.

[22] J.-C. Laprie, "Dependability Evaluation of Software Systems in Operation", *IEEE Trans. on Software Engineering*, vol. SE-10, pp. 701-714, 1984.

[23] I. Lee and R. K. Iyer, "Faults, Symptoms, and Software Fault Tolerance in Tandem GUARDIAN90 Operating System", Proc. *23rd IEEE Int. Symp. Fault-Tolerant Computing (FTCS-23)*, Toulouse, France, 1993, pp. 20-29.

[24] C. Béounes, M. Aguéra, J. Arlat, S. Bachman, C. Bourdeau, J. E. Doucet, K. Kanoun, J.-C. Laprie, S. Metge, J. Moreira de Souza, D. Powell, and P. Spiesser, "SURF-2: A Program for Dependability Evaluation of Complex Hardware and Software systems", Proc. *23rd IEEE Int. Symp. Fault-Tolerant Computing*, Toulouse, France, 1993, pp. 668-673.

[25] H. H. Ammar, Y. F. Huang, and R. W. Liu, "Hierarchical Models for Systems Reliability, Maintainability, and Availability", *IEEE Trans. on Circuits and Syst.*, vol. CAS-34, pp. 629-638, 1987.

[26] G. Chiola and S. Donatelli, "GSPN versus SPNs: What is the Actual Role of Immediate transitions?", Proc. *Int. Workshop on Petri Nets and Performance Models*, Los Alamitos, CA, 1991, pp. 20-31.

[27] J. F. Meyer and W. H. Sanders, "Specification and Construction of Performability Models", Proc. *Int. Workshop on Performability Modeling of Computer and Communication Systems*, Mont Saint Michel, France, 1993, pp. 1-32.

[28] A. Bobbio and K. Trivedi, "An Aggregation Technique for the Transient Analysis of Stiff Markov Chains", *IEEE Trans. on Computers*, vol. C-35, pp. 803-814, 1986.