Dependability of Fault-Tolerant Systems - Explicit Modeling of the Interactions Between Hardware and Software Components

Karama Kanoun, Marie Borrel

2nd Annual IEEE International Computer Performance and Dependability Symposium (IPDS'96), Urbana-Champaign (USA), 4-6 septemb
1996, pp.252-261

Dependability of Fault-Tolerant Systems — Explicit Modeling of the Interactions Between Hardware and Software Components

Karama Kanoun and Marie Borrel

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

Abstract

'his paper addresses the dependability modeling of !ware and software fault-tolerant systems taking into unt explicitly the interactions between the various ponents. It presents a framework for modeling these actions based on Generalized Stochastic Petri Nets PNs). The modeling approach is modular: the vior of each component and each interaction is esented by its own GSPN, while the system model is ined by composition of these GSPNs. The position rules are defined and formalized through identification of the interfaces between the ponent and the dependency nets. In addition to ularity, the formalism brings flexibility and re- ility. This approach is applied to a simple, but still esentative, example.

ntroduction

ı the context of computer system dependability, the for addressing simultaneously both hardware and vare dependability aspects has now been recognized. 'ever, even though a number of publications have devoted to the dependability of combined hardware software systems (see e.g. [5, 6, 13, 14]), work on aspects dealt with at the same time is not prevalent. , it is noteworthy that, when they are considered ther for real-life systems, the interactions between the ponents are not usually modeled explicitly (see e.g. 16, 20]).

'his paper addresses the dependability modeling of ware and software fault-tolerant systems taking into unt the interactions between the various components. se interactions result for example from components munications for functional purposes (i.e., *functional actions*), or from the structure of the system, mainly distribution of software components onto the ware components (i.e., *structural interactions)*, or fault tolerance and maintenance strategies induce dependencies between at least two compone that are usually stochastic in nature. As a result, syst dependability cannot simply be obtained by combini the dependability of its components. An overall mo accounting for these dependencies is thus needed. C aim is to model explicitly these dependencies so as quantify their influence on system dependability. This of prime importance during the design of a new system while upgrading an already existing one. The desig can make different assumptions about the interactio between the components and compare the dependabil of the resulting alternative solutions through sensitiv studies. As the nature of interactions is strongly linked the modeling level considered and the assumptions ma at the considered level, it is not possible to model all interactions that could take place for any fault-toler system. Rather, we define a framework for modeli these interactions in a systematic way and, mo generally, we define a framework to build up the depe ability model of a fault-tolerant system explicitly taki into account these interactions. To do this, we follow modular approach based on Generalized Stochastic Pe Nets (GSPNs) due to their ability to handle modular and hierarchy. Note that modular approaches usi GSPNs or their offsprings are widely used (see e.g., 18]). Our contribution lies in modeling the interactio between hardware and software components and giving formal description of these dependencies.

The paper is organized in five sections: Section presents the framework for modeling interactio between hardware and software components. Section gives a formal description of the various types dependency nets while Section 4 illustrates the appro on a duplex system with several interactions. Section concludes.

2. Modeling Framework

The modeling approach consists in identifying, bas on the analysis of the system's behavior, dependenc al or structural interactions or by interactions due to em reconfiguration and maintenance. Some examples ependencies due to these interactions are given in the wing. Error propagation between two software com- nts is an example of stochastic dependency resulting functional interactions (exchange of data or transfer ntermediate results from one component to another). halting of the software activities following a perma- failure of the hosting hardware is an example of hastic dependency induced by a structural interaction. ing of a single repairman by the two hardware com- rs leads to a maintenance dependency whereas ching from an active component (hardware or vare) to a spare component following a permanent re of the active component leads to a reconfiguration ndency. In this paper, we consider interactions that driven by events occurring in a component whose rrence may impact the behavior of other components. high level model of the system is first derived based e previous analysis. It is made of blocks and arrows: ck stands for the component model (*component net)* dependency model (*dependency net*), and an arrow s the direction of the dependency. The system model us obtained by composition of the component and ndency models. In a second step, each block is ced with its detailed GSPN. To allow for a matic build up of dependency nets, rules that will to be followed during model construction are ed. These rules manage the interfaces between the ndency and component nets and are prerequisite for ularity, hierarchical modeling and re-usability (re- ility is a valuable concept when it comes to doing itivity studies about certain assumptions regarding a em's behavior or when several alternative solutions being considered). Also, these rules allow an easy lation of the global model. In the rest of the section, give the characteristics of the component and ndency nets and present the various types of ndency nets together with the rules that have to be wed to build up the GSPNs.

nponents nets: A component net represents the vior of a component as resulting from the activation aults in this component and the subsequent error essing, restart or repair actions. The assumptions e and the degree of detail considered are usually ed by the interactions with other components one s to exploit (such as the consequences of non cted errors or activation of temporary faults). A ponent net is designed to be a standalone net with its al marking, it is live and bounded. It can be nected with dependency nets only following well de- fined rules as explained hereafter; connections must alter the initial structure of the component nets.

Dependency nets: A dependency net is linked to at le two adjacent nets: an *initializing* and a *target* net t could be component or dependency nets. To forma describe dependency nets and to promote re-usability, define as much common characteristics as possible. A result, whatever is the kind of interaction modeled, all dependency nets are initialized and interfaced with adjacent nets following the same rules; they only differ their *effects* on the target net. The common characterist and the different effects on the target nets are introduc in figure 1 where a hypothetical dependency net with types of effects is given (the notations are introduced w the formalism). They are summarized hereafter.

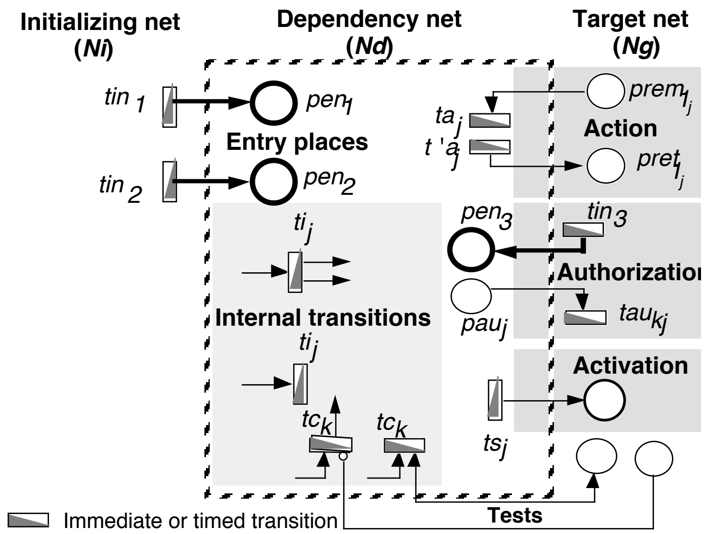


Figure 1: Characteristics of dependency nets

Dependency net initialization

- A dependency net is initialized through the marking one or more *entry places* by the initializing net(following firing of *initializing transitions* in the initializing net(s) (as interactions are event driven).
- The initial marking of the entry places is zero.
- When the initializing net is a component net, consequences of initializing transition(s) firing on component behavior are modeled within component net (marking of one or several inter places) and an additional token is generated a deposited in the entry place of the dependency net activate the interaction.
- If the initializing net is a dependency net, the tok deposited in the entry place could be either the c generated when entering the initializing depender net (corresponding to a series of success interactions) or an additional one newly genera (corresponding to the initialization in parallel of two more interactions).

rnal transitions: The dependency net has internal sitions (timed or instantaneous) whose firing may be pendent from the marking of the adjacent nets ependent transitions) or conditioned by the marking laces in the adjacent nets. A condition is modeled by nhibitor arc or an arc from and towards the tested e: the marking of this place is not changed. The faces with the adjacent nets (excluding initializing and effects on the target nets) are thus only tituted by tests on the marking of specific places. An nal transition could be absorbing (i.e., the tokens are rbed).

cts on target nets: These effects are strongly related ne type of interaction modeled. Thus, three such cts have been identified:

the interaction consists in changing the state of nother component (the target net is necessarily a omponent), the effect at the GSPN level involves emoval of a token from a stable place (i.e., a place ollowed by timed transitions, this condition stems om the fact that only stable places correspond to ates of the components) in the target net and return f the token to the same place or to another stable lace of the target net (immediately or after firing nternal transitions in the dependency net). This is eferred to as an *action net*.

the interaction consists in performing or synchronizing reconfiguration or maintenance actions, the effects epend on the nature of the initializing net:

if the initializing net is a component net, the interaction consists in coordinating the component restart (or repair) action with the restart (or repair) action of the components to which it is linked: it requests permission before undertaking internal actions, these actions are enabled by the dependency net (immediately or after firing of internal transitions). The target net is necessarily the same as the initializing net. At the GSPN level, the effect consists of enabling a transition in the component net through the marking of a place in the dependency net. Since the component net is a standalone net, this means that, in the component net, this transition has also to be enabled by the marking of at least an internal place. This is an *authorization net*,

if the initializing net is a dependency net, the interaction consists in activating another interaction with other linked components; at the GSPN level, this consists of initializing another dependency net by depositing a token in its entry place following the firing of an initializing transition in the initializing dependency net. As previously stated, depending on additional one, both dependency nets run in paral or in series. This is an *activation net*.

The previous rules are intended to manage the *sta* links between dependency and adjacent nets. Furtl rules have to be considered to control the *dynan behavior* of the nets (i.e., the tokens generation and th flow). They are given together with the formalism in t next section. Note that model construction is an iterat process with information flow in both directions from dependency nets to/from components nets: in t component models, care should be taken to inclu potential dependencies.

Interaction origin and dependency net type: T interactions have been attributed to three possible origi functional, structural and those due to system configuration and maintenance. Functional and structu interactions are usually accompanied by a state chan the associated nets are thus action nets. Dependencies d to reconfiguration and maintenance may induce a st change and they could be any kind of dependency net.

3. Nets formalization and validation

The aim of this section is to give a formal descripti of the various rules introduced in the previous section a to address model validation. We first give the ma notations, the other notations being defined in table 1.

General Notations

$pen_k \in P_d$	Entry place (EP) of Nd.
$tin_j \in T_i$	Transition of Ni that initializes Nd by marking an EI
Mi_j	Initializing marking of a dependency net
$ti_j \in T_d$	Internal independent transition in Nd
$tc_k \in T_d$	Internal conditioned transition in Nd

Interfaces for an action net

$ta_j, t'a_j \in T_d$	Removing and returning action transitions in Nd
$prem_{k_j} \in T_g$	Input place in Ng of the removing action trans. ta_j
$pret_{k_j} \in T_g$	Output place in Ng of the returning action trans. $t'a$

Interfaces for an authorization net

$pau_j \in P_d$	Authorization place of Nd

Interfaces for an activation net

$ts_k \in T_d$	Activation transition of Nd

Table 1: Notations

Let Ni, Nd and Ng denote respectively an initializing dependency, and a target net, and let Nx be any of the nets ($x = i, d, g$). $Nx = (P_x, T_x, I_x, O_x, pr_x, pa_x)$ where:

- P_x is the set of places of Nx ,.
- $T_x = Ttim_x \cup Timm_x$ is the set of transitions of Nx : Tti is the set of timed transitions and $Timm_x$ is the set immediate transitions.
- $I_x : P_x \times T_x \rightarrow \mathbf{N} \cup \{-1\}$ is the input function,

$_x: T_x \times P_x \rightarrow \mathbf{N}$ is the output function ($\mathbf{N}$ is the set of atural integers).

r_x the set of timed transitions rates, pa_x the set of ring probabilities of immediate transitions.

'he set of places and transitions are such that: $P_d = \varnothing$, $P_i \cap P_g = \varnothing$, $P_d \cap P_g = \varnothing$, $T_i \cap T_d = \varnothing$, $T_g = \varnothing$ and $T_d \cap T_g = \varnothing$

'he interfaces of a dependency net Nd with an alizing net Ni and a target net Ng are the input and ut functions, I_{id}, I_{dg}, O_{id}, O_{dg}, that connect places transitions of Nd to places and transitions of Ni or These functions are defined as follows:

$_d: P_i \times T_d \rightarrow \mathbf{N} \cup \{-1\}$ $\quad I_{dg}: P_d \times T_g \rightarrow \mathbf{N} \cup \{-1\}$

$'_{id}: T_i \times P_d \rightarrow \mathbf{N}$ $\quad O_{dg}: T_d \times P_g \rightarrow \mathbf{N}$

Vhen it is not necessary to distinguish between alizing and target nets, indices *i, d, g* are omitted.

alization: Initializing transitions, entry places and initializing marking of a dependency net Nd are ned as follows: let $tin_j \in T_i$ and $p_k \in P_d$ such that $in_j, p_k) > 0$, $I_{di}(p_k, tin_j) = 0$, if $\forall p_l \in P_d, p_l \neq p_k$, we have $_l, tin_j) = 0$ then tin_j is an initializing transition of Nd p_k is an entry place of Nd, denoted pen_k. An entry e can be initialized by several transitions. In order for nsition tin_j of a net Ni to be fired, one must have: $\in P_i$ $\quad I_{id}(p_m, tin_j) \geq 0 \Rightarrow Mi_n(p_m) \geq I_{id}(p_m, tin_j)$ (Mi_n: Nd alizing marking) and $I_{id}(p_m, tin_j) = -1 \Rightarrow Mi_n(p_m) = 0$ [1].

rnal transitions: Internal transitions of a dependency can be independent or conditioned by the marking of es in adjacent nets. They are defined as follows:

$_j \in T_d$ is an independent transition if $\forall p_k$ such that $(p_k, ti_j) > 0$ or $O(ti_j, p_k) > 0$ then $p_k \in P_d$, $I = I_d$, $O = O_d$.

$_k \in T_d$ is a conditioned transition if the two ollowing conditions are verified:

) $\exists p_j \in (P_g \cup P_i)$ such that: $I(p_j, tc_k) = O(tc_k, p_j) > 0$ or $I(p_j, tc_k) = -1$

) $\forall p_n \in (P_g \cup P_i)$ such that $I(p_n, tc_k) > 0$ or $O(tc_k, p_n) > 0$ then $I(p_n, tc_k) = O(tc_k, p_n)$.

n internal transition can be an absorption transition. ndependent or a conditioned transition ti_j or tc_j is an rption transition if: $\forall p_n \in P$, $O(ti_j, p_n) = 0$ or $_j, p_n) = 0$ with $P = (P_d \cup P_i \cup P_c)$.

on nets: An action net ends with a transition which es a token to be removed from a stable place of the et net and to be returned to the same or another stable e of the same net. Removal and return can be done ugh two distinct transitions (with internal transitions and places between them) or through the same transiti The number of tokens in the target net rema unchanged. Transitions ta_j and $t'a_j$ are action transitic if the four following conditions are met:

1) $\forall p_h \in P_i$, $I(p_h, ta_j) = 0$ and $\forall p_l \in P_i$, $O(ta_j, p_l) = 0$
2) $\exists p_k \in P_g$ such that: $O_{dg}(ta_j, p_k) = 0$ a $I_{dg}(p_k, ta_j) > 0$ (ta_j: removing transition) $\exists p'_k \in P_g$ such that: $O_{dg}(t'a_j, p'_k) > 0$ a $I_{dg}(p'_k, t'a_j) = 0$ ($t'a_j$: returning transition)
3) $\sum_{i=1}^{|P_g|} O(t'a_j, p'_i) = \sum_{i=1}^{|P_g|} I(p_i, ta_j)$
4) p_k or p'_k is a stable place: $\exists t \in Ttim_g$ such tl $I_d(p_k, t) > 0$ and $\forall t \in Timm_g$ such that $I_d(p_k, t)$: then $I_d(p_k, t) = O_d(t, p_k)$, p'_k must verify equivalent relation.

p_k is then the input place of the removing action t being denoted $prem_{kj}$ and p'_k ,the output place of returning transition $t'a_j$, is denoted $pret_{kj}$.

Authorization nets: An authorization net ends with c or several places enabling firing of transitions in target net(s), authorization places. In this case, the tar net is necessarily the initializing net.

$pau_j \in P_d$ is an authorization place of Nd if: $\forall t \in T_d$ such that $I(pau_j, t) = 0$ then $\exists t_{jk} \in (T_i \cup T_g)$ su that $I(pau_j, t_{jk}) > 0$ and $O(t_{jk}, pau_j) = 0$; t_{jk} is then authorized transition in Nc, it is denoted tau_{kj}.

Activation nets: An activation net allows linking dependency nets (i.e. synchronize the relat interactions). It ends with a transition, synchronizati transition, that sends one or several tokens in one several other dependency nets (but does not remc tokens from these nets). $ts_j \in T_d$ is an activation transiti if $\forall p_m \in P_g$ $\; I_{gd}(p_m, ts_j) = 0$ and if $\exists p_k \in P_g$ such tl $O_{dg}(ts_j, p_k) > 0$.

Dynamic behavior: The generation, moving a absorption of the tokens has to be controlled wh building up a dependency net so as to ensure that resulting global net is bounded and live. Each tok generated upon dependency net initialization by marking of an entry place must thus be removed either the dependency net itself or through the effect on target net. It is then necessary that as long as dependency net place is marked, whatever the glol marking, there is a transition that can be fired and tl removes a token from this place. This condition must formalized for the internal places of all types dependency nets.

Let P_y be such that $P_y = P_d$ for action and activati nets, $P_y = \{P_d - P_{aut}\}$ for authorization nets. The inter

---

$,t) \geq 0 \Rightarrow Mn_x(p) \geq I(p,t)$ means that there must be enough ens in all the input places of t to enable it.

$,t) = -1 \Rightarrow Mn_x(p) = 0$ means that if there is an inhibitor arc very place has at least one transition that removes ›kens: $\forall\, p \in P_y\ \exists\, t \in T_d \mid I(p,t) > 0$ and $O(t,p) = 0$

' there exists an arc with multiplicity x from a place to transition, there exist x-1 other arcs with multiplicity to x-1, from the same place to x-1 other transitions ith the same input and output as the preceding ansition:

$\forall t \in T_d$, and $p \in P_y$ with $I(p,t) > 0$ and $O(t,p) = 0$ if $p' \in P_d$ such that $I(p,t) = x,\ x \in \mathbf{N}$ then $\exists x$ transitions $\in T_d$ such that $I(p',t_j) = j,\ j = 1,\ldots,x$, $I(p,t_j) > 0$ and $(t_j,p) = 0$.

ll these transitions are independent internal ansitions.

' there exists a test arc with multiplicity x between a lace and a transition, there exist x-1 other arcs with ıultiplicity 1 to x-1, from the same place to x-1 other ansitions with the same input and output as the receding transition.

$\forall p \in P_y$ and $\forall p \in P_y$ with $I(p,t) > 0$ and $O(t,p) = 0$, if $p' \in P$ such that $I(p',t) = O(t,p') = x$, then $\exists x$ ansitions t_j such that $I(p',t_i) = O(t_j,p') = j,\ j = 1,\ldots,x$ $(p,t_j) > 0$ and $O(t_j,p) = 0$.

' $p' \in P_d$, t is an independent transition, if $p' \notin P_d$, t is a ɔnditioned transition.

' there exists an inhibitor arc from a place to a ansition then there must exist an arc from the same lace to another transition.

$t \in T_d$ and $p \in P_y$ with $I(p,t) > 0$ and $O(t,p) = 0$, if $\exists p' \in P$ ıch that $I(p,t) = -1$, then $\exists t'$ such that $I(p',t') = 0$, $(p,t') > 0$ and $O(t',p) = 0$.

' $p' \in P_d$, t is an independent transition, if $p' \notin P_d$, t is conditioned transition.

he sum of firing probabilities of immediate ansitions in conflict is always equal to 1:

$p \in P_d$, if there are $\{t_1 \ldots t_i \ldots t_u\} \subset Timm_d$ such that $(p,t_i) > 0$, and $\forall p'$ such that $I(p',t_{i=x}) =$ constant and $(p',t_{i \neq x}) = 0$, then $\sum_{i=1}^{n} \mathrm{pa}(t_i) = 1$.

' $p' \in P_d$, t is an independent transition, if $p' \notin P_d$, t a conditioned transition.

ı some situations, depending on the marking of the ırget net, the token must be removed by an absorbing ansition if it cannot be removed by the target or ıitializing net: $\forall p \in P$ if $\exists t \in T_d$ such that $|\, I(p,t) \geq 0$ or $(p,t) = -1$ then $O(t,p) = 0$.

lel validation: Several verifications are needed to h confidence in the model; they are usually grouped two categories: syntactic and semantic validation . Syntactic validation consists in checking that the el represents the dependability of *a* system; it mainly consists of checking that the model represents ı dependability of *the* system under validation; it requi comparing the system and the model behaviors w respect to variations of the underlying assumptio Usually it is performed through sensitivity studies. Due the scope of the paper, we concentrate on the structu validation. The rules for interfacing dependency a component nets and for managing the dynamic behav allow us to obtain, by construction models that ; structurally valid (i.e., live and bound). Structu validation is progressively done, starting from ı component nets and gradually adding dependency nє Identification of possible problems is thus easy. Al these verifications can be achieved automatically computation of place- and transition-invariants checking necessary or sufficient conditions of livelin and boundedness with a tool such as SURF-2 developed at LAAS-CNRS.

4. Application to the duplex system

Let us consider a duplex system composed of t hardware computers (H1 and H2) and two identi software replicas: each replica is implemented on computer. We assume semi-active replication [17]: ı leader replica (L) processes all input messages a provides output messages while the follower replica (does not produce output messages. The internal state o is updated by means of notifications from L completed direct processing. Temporary faults in the software ; tolerated by exception handling mechanisms associa with each replica, whereas the activation of perman faults leads to restart the replica. To reduce syst unavailability, after detection of an error due to permanent fault in L, the software replicas switch th roles: processing is performed on the new leader bef restarting the new follower. If L and F fail, L is restar first. Also, in case of failure of the hardware hosting (identified as H1), the replicas switch their roles; ı computer hosting the new follower is then repaired. W respect to hardware repair policy when the two comput are in failure, we consider two assumptions: R1: the t computers share a single repairman and priority of given to H1 and R2: two repairmen are available.

4.1. High level modeling

Interactions are directly related to the assumptic made about the components' behavior. Owing to ı importance of the impact of temporary faults on ı behavior of hardware and software components [7, 8, 19], both permanent and temporary faults are conside in this example.

: is assumed that the activation of a fault may lead to 'ollowing dependencies:

ollowing activation of a hardware fault:

an error due to the activation of a temporary fault in a hardware computer may ***propagate*** to the hosted software replica,

an error due to the activation of a permanent fault in a hardware computer leads to ***stopping*** the hosted software replica that is restarted after the end of hardware repair.

ollowing activation of a software fault: owing to the otifications sent from the leader to the follower, an :ror in the leader due to a permanent fault — usually :ferred to as solid fault — may ***propagate*** to the)llower (it is assumed that errors due to temporary ults — usually referred to as soft faults — are onfined and do not propagate).

)ependencies induced by fault tolerance and itenance strategies are as follows:

etween software replicas: dependency due to fault)lerance of permanent software faults, i.e., ***:configuration*** from F to L.

etween hardware computers: dependency due to ***:configuration and repair***.

etween all components: coordination of fault)lerance and maintenance actions to form a global ***:covery strategy*** when several components are in ilure.

'hese dependencies are summarized in table 2 ther with the names of the associated nets which are to build up the high level model of the duplex :m. The latter is given in figure 2 where N_{Hard} and t represent a computer and a software replica model :ctively.

Prop	models propagation of a hardware error to the hosted software replica
Stop	models software stop after activation of a permanent fault in the hosting hardware
'Prop	models propagation of a software error
Rep	models hardware reconfiguration and repair
Rec	models software reconfiguration
Strat	models the global recovery strategy

Table 2: Dependency nets

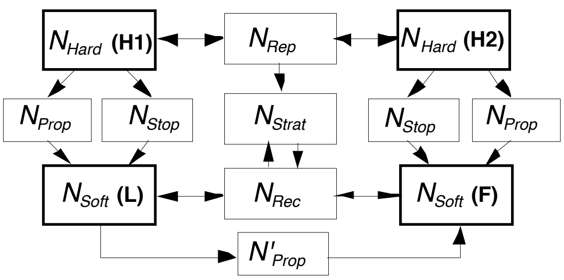


Figure 2: High level model of the duplex system

The corresponding GSPNs are built up following t rules and formal description presented in Sections 2 a 3; they are successively given in the remainder of t section.

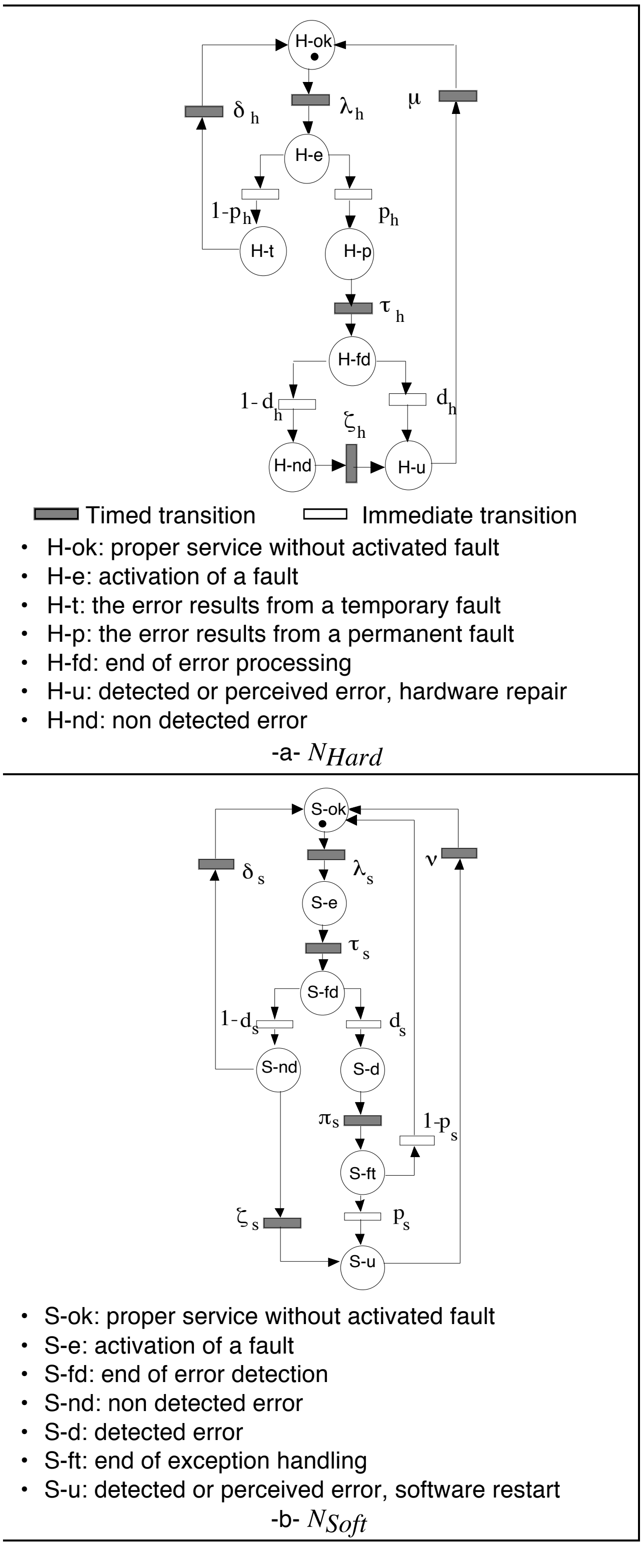
4.2. Hardware and software component nets

Figure 3 gives the component nets. The hardwa model is based on the following assumptions:

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

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

- Faults are activated with rate λ_s.
- An error is either detected with probability d_s, or n detected $(1-d_s)$; detection rate τ_s.
- The detected error is processed by means of excepti handling mechanisms during a short time $1/\pi_s$. At t end of error processing, 1) if the fault is tempora (probability $(1-p_s)$) its effects are eliminated and t software resumes its normal mode of operation, 2) the fault is permanent (probability p_s); the software h to be restarted (rate: ν) to eliminate its effects.(1- measures the efficiency of fault containme procedures [8, 11].
- The effects of a non detected error may eliminated



'igure 3: Hardware computer and software replica nets

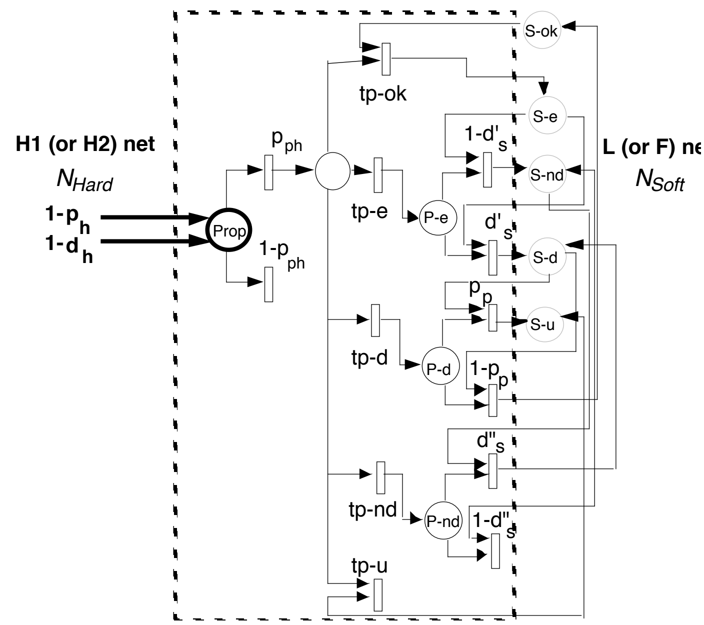
:limination rate δ_s), or perceived (perception rate ζ_s) ı which case the software replica has to be restarted.

'he difference between these nets lies in that for ware, temporary and permanent faults are :rentiated by their respective consequences following 'ation, whereas for software, they can only be nguished after specific processing [12].

Error propagation nets

n hardware to software: It is assumed that only propagate from a hardware computer to the host software replica. The error propagation net, shown figure 4, is initialized by the marking of place Pr following the firing of transition 1-d_h (undetected err or of transition 1-p_h (an error due to a temporary fault) the hardware net (initializing net). With probability 1-p the error is not propagated and with probability p_{ph} it N_{Prop} is *an action net*, whose effects on the software r (target net) are as follows:

- If the token is in S-ok, it is returned to S-e, the induc error is then processed in the same way as when t fault is activated without propagation (through λ_s figure 3-b).
- If the token is in S-e, since a fault is already activat in the software, the probability of error detection m be reduced ($d'_s \leq d_s$), if the errors are detected, the tok is returned to S-d; if they are non detected (w probability 1-d'_s) the token is returned to S-nd.
- If the token is in S-nd (an internal error is non detect in the replica) the propagated error and the intern error are detected with probability d''_s ($d''_s \leq d'_s$, owi to the perturbation due to the first error) the token returned to S-d; the errors remain undetected with d''_s.
- If the token is in S-d the propagated error c compromise error processing and prevent the recove of an error due to a temporary fault. The internal a propagated errors are recovered with probability 1- (1-p_p < 1-p_s).
- If the token is in S-u, the software replica is alrea under restart, the token of N_{Prop} is absorbed throu tp-u and the token of N_{Soft} is kept in S-u.



Entry places and initializing arcs are indicated in bold

Figure 4: Error propagation net, N_{Prop}

n L to F: The dependency net, the target net and the :ts on the target net are exactly the same but the ılizing net is that of the software leader. It is assumed only undetected errors in L and detected errors of L to permanent faults, can propagate. The error agation net is then initialized following the firing of or p_s. The probability of error propagation is p_{ps}.
higher modeling level, error propagation from L to F ɔe regarded as common mode failures.

Software stop and restart net

ollowing a detected error or the perception of an :tected error in an hardware computer, the hosted vare replica is stopped and is restarted after repair of hardware. We assume that the repair includes the ca restart. The software stop and restart net (Figure an action net, it is initialized by the marking of STP ɪwing the firing of transition ξ_h (perception of a non :cted error due to a permanent fault) or d_h (detection ı error due to a permanent fault). Transitions t1 to t5 ɔve the token from places S-ok, S-e, S-d, S-nd or S-u :ectively. After repair of the hardware (including vare restart), RST is marked and the token is returned ·ok.

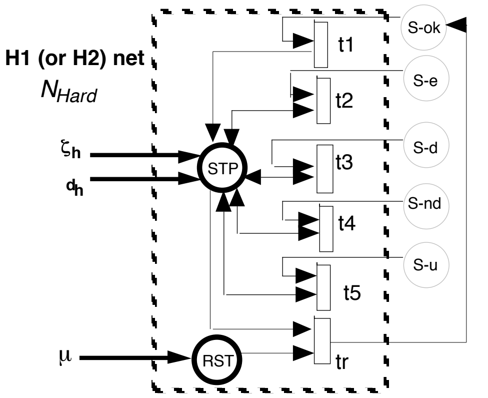


Figure 5: Software stop and restart, N_{Stop}

Hardware reconfiguration and repair net

ıs previously stated, we consider two different mptions: A1 assumes a single repairman, while A2 mes the presence of two repairmen. The :sponding nets are given in figure 6. Each net is posed of two parts corresponding respectively to nfiguration (the shaded parts on the figures) and ir. They are grouped together because the nfiguration is automatically followed by a repair. e the reconfiguration strategy is the same, the ciated nets are the same. The two nets are commented ther and, when they are different, the figure number ecified.

N_{Rep} is initialized by the marking of H (respectively H2F) following the firing of d_h, detection an error due to a permanent fault or ζ_h perception of undetected error, in the hardware hosting L, (respectively H2):

- if H1F is marked, H1 is in failure (H1 is the initializ H2 the target):
  - if H2 is not in failure (REP2 not marked) switching attempted, (β_h) and HSW is marked:
    1) Switch can succeed with probability c_h, pla SSW is then marked,

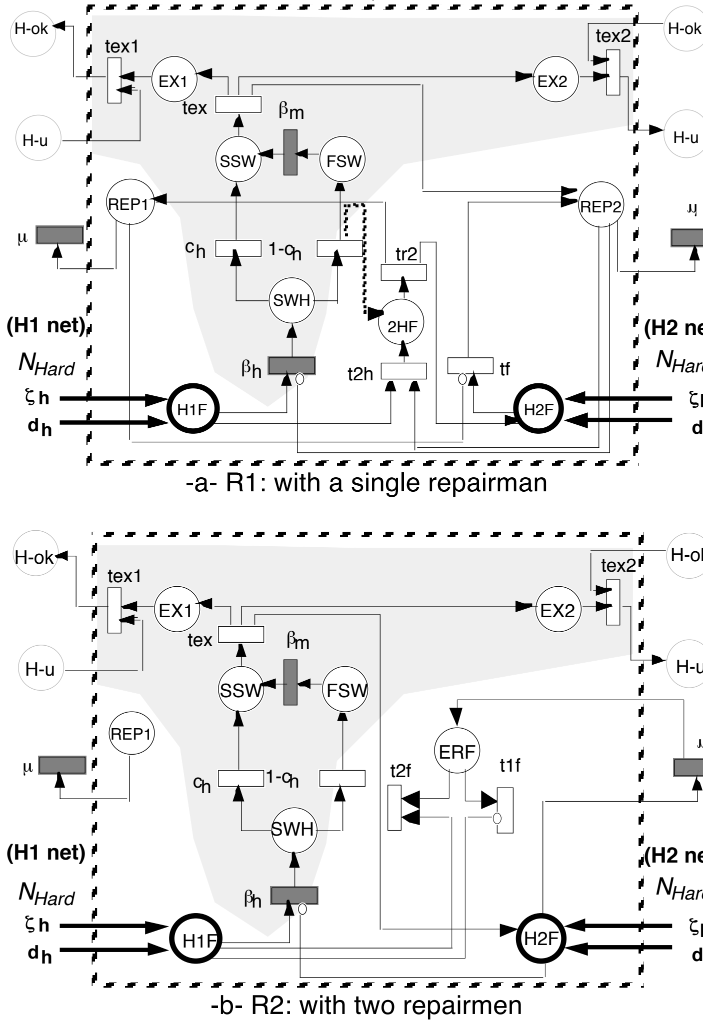


Figure 6: Hardware reconfiguration and repair nets, N_{Rep}

2) It can fail with probability 1-c_h, FSW is mark and switching is done manually[2] (β_m), SSW then marked; tex can be fired, places EX1, E and REP2 are marked; tex1 and tex2 can fired, they remove the token from H-u to H- and from H-ok to H-u, F becomes the n leader, L the new follower and it can be restart (REP2 is marked in figure 6-a for R1, H2F marked in figure 6-b for R2),

---

[2] Other possible assumption: it can be assumed that the man switch is not attempted. In this case, transition 1-c_h leads to pl

for R1: if H2 is in failure (REP2 marked): t2h is fired removing the token from REP2 to 2HF; tr2 can then be fired returning a token in REP1 and one in H2F in order to repair H1, then H2, (for R2: if H2 is in failure (H2F marked): repair of H1 and H2 are enabled; at the end of H2 repair, if H1 is still under repair H2 is restarted with the leader),

for R1 if H2F is marked: H2 is in failure (H2 is ılizer and target): if H1 is not in failure (REP1 not ced), tf can be fired and REP2 is marked, authorizing epair of H2; else the token stays in H2F until the end [1 repair; repair of H2 in then allowed through the ring of REP2 (for R2: repair of H2 is enabled without condition on H1).

R1 and R2: if N_{Rep} is initialed by H1 only, its is an on, activation and an authorization net; when ılized by H2 only, it is an authorization net. If it is ıled by H1 then H2 (or H2 then H1) it is an orization net.

Software reconfiguration net

'he software reconfiguration net is given in figure 7. initialized by the marking of S1F (respectively S2F) wing the firing of transition p_S, a detected error due permanent fault or perception of an undetected error (respectively F):

S1F is marked, L is in failure (L is the initializer and the target):

if F is not in failure (RSTF not marked) switching is attempted (β_S) and SWS is marked.

1) Switch can succeed with probability c_S, places EXL, EXF and RSTF are then marked. Marking of EXL allows firing of tex that removes the token from S-u to S-ok in L. Marking of EXF allows the firing of one of transitions t1 to t4 that removes the token in the leader net from places S-ok, S-e, S-d or S-nd and return it to S-u. Marking of RSTF enables transition ν (in F) to restart it.
2) Switch can fail with probability $1\text{-}c_S$, places EXF and 2SF are then marked. Marking of 2SF allows transition tr2 firing that marks places RSTL and S2F. Marking of place RSTL enables transition ν in the leader to restart it. Marking of S2F allows the follower restart only after the end of the leader restart.

if F is in failure (RSTF marked) t2s is fired and 2SF is marked allowing the firing of tr2 that marks RSTL and S2F. Marking of place RSTL enables transition ν in the leader to restart it. Marking of S2F allows the restart of F only after the end of the leader restart.

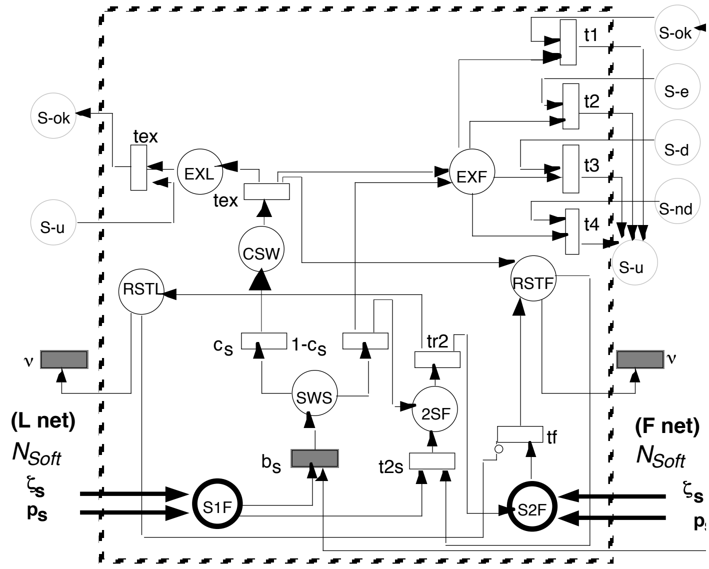


Figure 7: Software reconfiguration net, N_{Rec}

- if S2F is marked, F is in failure (F is initializer a target): if L is not in failure (RSTL not marke transition tf can be fired and RSTF is mark authorizing the restart of F; else the token stays in S until the end of L restart, restart of F is then allow via RSTF marking.

If $N_{Rec}$is initialed by L only, its is an action and authorization net. If N_{Rec} is initialized by F only, it is authorization net. If N_{Rec} is initialed by L then F (or then L) it is an authorization net.

4.7. Global recovery strategy net

The global recovery strategy net is initialized by N_{I} through F1H following the firing of tex. If F is in failu (RSTF marked) t2 removes the token from RSTF a deposits a token in RSTL and another one in S2F in orc for L to be repaired first. If F is not in failure (RSTF r marked) transition t1 deposits a token in CSW in orc that the roles of the follower and leader to be exchange N_{Strat} is an action net if place RSTF is marked and activation net if RSTF is not marked.

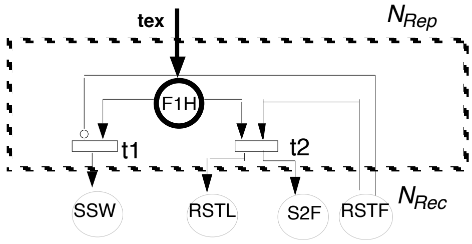


Figure 8: Global recovery strategy net, N_{Strat}

4.8. Concluding remarks and global model

Due to lack of space the formal description of t previous nets is not presented. It can be checked that t hardware and the software GSPNs are live and bound erties have to be done with the adjacent nets as :ated in figure 2, as follows

$_{Prop}$ has to be validated connected with N_{Hard} and $_{Soft}$, (N'_{Prop} is identical to N_{Prop}),

$_{Stop}$ has to be validated with N_{Hard} and N_{Soft},

$_{Rep}$ has to be validated with two N_{Hard},

$_{Rec}$ has to be validated with N_{Soft},

$_{Strat}$ has to be validated with all the other nets (that ave already been validated).

'he overall model obtained by replacing the blocks of :e 2 with their GSPNs given in figures 3 to 8 has been essed by SURF-2. The marking graph has 1200 :ings and the Markov chain 500 states without any aggregation due to symmetry.

: could be argued that the state space may be very : for more complex systems, this is inherent to the plexity of the system to be modeled and to the level etail considered. The only difficulty due specifically ur modeling approach is the number of markings; it be overcome by using an aggregation technique at the 'N level to suppress immediate (see e.g. [1]).

'onsidering again the duplex system, taking into unt the fact that the transition rates associated with : detection and processing mechanisms are very high pared to failure, repair and restart rates (the durations rror detection and processing is of the order of the nd whereas the intervals to failures are several lreds of hours), the model can be reduced to 9 states iown in figure 9. This model is to be considered as a :ing case allowing verification of the complete model is specific case.

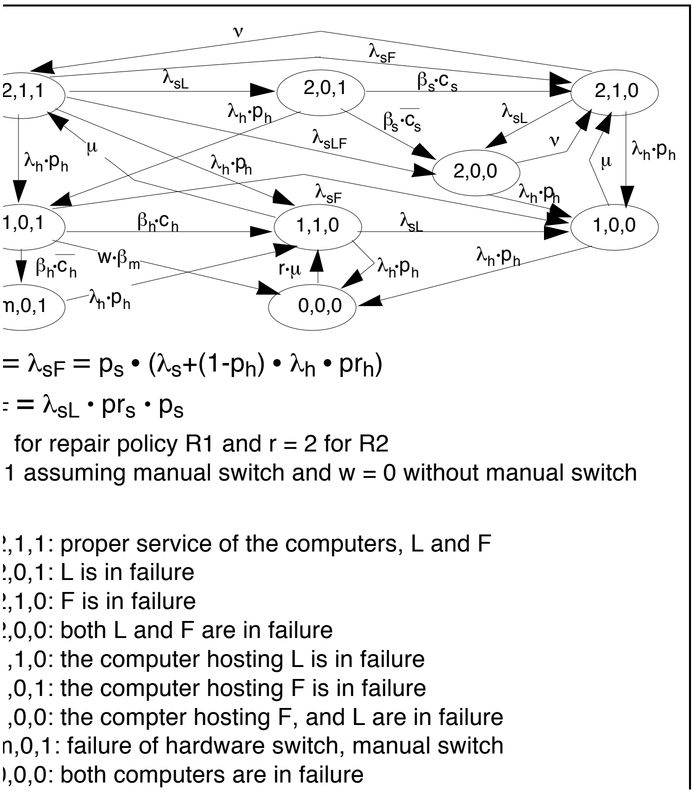


Figure 9: Reduced Markov chain of the duplex system

5. Conclusion

This work presented in this paper has allowed varic types of dependencies between hardware and softw: components of a fault-tolerant system to be identifi These dependencies may result from functional structural interactions as well as interactions due reconfiguration and maintenance strategies. T dependability model of the system is obtained composition of the components models with th associated with the dependencies. The rules interfacing the models have been clearly defined a formally described to build up easily validable syst models. The formal description facilitates composition of the various GSPNs.

The modeling approach has been illustrated by simple example, including all the types of dependenc identified: the duplex system. Modeling of this syst showed the strong dependency between components. F example: the activation of a temporary hardware fa may propagate an error to the hosted softw: component, which in turn may propagate to otl components communicating with it (without bei necessarily on the same computer). Thus the activation a hardware fault, may lead to the restart of one or m software components. Even if this has already be observed on real-life systems, it has not been mode explicitly in previous work. Also, we have shown h the modification of one or several assumptions can performed without modifying all GSPNs, considering t repair policies and two switching policies (with without manual switch).

The main advantage of the modeling approach, bas on considering explicitly the interactions, lies in efficiency for modeling several alternatives for the sa system. These alternatives may differ by th composition (number of computers or replicas) or organization (distribution of software components o the hardware) or by the fault tolerance and maintenar strategies. One can clearly identify from the beginni the components and interactions that are specific a those that are common to all alternatives. The GSPNs t are common are thus developed and validated only onc

This approach has been applied to the French Traffic Control system (the subset associated with Flight Plan Processing and Radar Data Processing) in where twelve alternative architectures have been mode and their unavailability compared to identify the m suitable one. Based on these results, additional and m detailed architectures have been modeled in [4]. T ication showed all the power of the modeling oach with the explicit modeling of the interactions.

:nowledgments

he work presented in this paper has been partially orted by the French Civil Aviation Authority and SRTI TEM while Marie Borrel was with SRTI SYSTEM and by Ξuropean Commission through the OLOS Network (A tic approach to the dependability analysis and evaluation ontrol systems involving hardware, software and human ırces).

erences

H. H. Ammar, Y. F. Huang and R. W. Liu, "Hierarchical Models for Systems Reliability, Maintainability, and Availability", *IEEE Trans. on Circuits and Syst.*, CAS-34 (6), pp. 629-38, 1987.

G. Balbo, "On the Success of Stochastic Petri Nets", in *6th International Workshop on Petri Nets and Performance Models,* (Durham, NC, USA), pp. 2-9, 1995.

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", in *23rd IEEE Int. Symp. Fault-Tolerant Computing,* (Toulouse, France), pp. 668-73, 1993.

M. Borrel, *Interactions between Hardware and Software Components: Characterization, Formalization and Modeling — Application to CAUTRA Dependability*, PhD Dissertation, N°96-001, In French, 1996.

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*, C-27 (6), pp. 548-60, 1978.

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

W. R. Elmendorf, "Fault-tolerant Programming", in *2nd IEEE Int Symp. Fault-Tolerant Computing,* (Newton, Massashusetts), pp. 79-83, 1972.

J. Gray, "Why Do Computers Stop and What Can be Done About it ?", in *5th Int. Symp. on Reliability in Distributed Software and Database Systems,* (Los Angeles, CA), pp. 3-12, 1986.

K. Kanoun, M. Borrel, T. Moreteveille and A. Peytavin, "Modeling the Dependability of CAUTRA, a Subset of the French Air Traffic Control System", in *26th Int. Symp. Fault-Tolerant Computing (FTCS-26),* (Sendai, Japan), pp. LAAS-Report, 95-515, 1996.

J.-C. Laprie, "Trustable Evaluation of Computer Systems Dependability", in *Applied Mathematics and Performance/Reliability Models of Computer/ Com-*

[11] J.-C. Laprie (Ed.), *Dependability: Basic Concepts a Terminology,* Dependable Computing and Fault-Toler Systems, 5, 265 p. , Springer Verlag, Wien-New Yo 1992.

[12] J.-C. Laprie, "On The Temporary Character Operation-persistent Software Faults", in *4th Int. Syn on Software Reliability Engineering,* (Denver, Colorad pp. 125, 1993.

[13] J.-C. Laprie and K. Kanoun, "X-ware Reliability a Availability Modeling", *IEEE Trans. on Softwa Engineering*, SE-18 (2), pp. 130-47, 1992.

[14] J.-C. Laprie, K. Kanoun, C. Béounes and M. Kaânic "The KAT (Knowledge-Action-Transformati Approach to the Modeling and Evaluation of Reliabil and Availability Growth", *IEEE Trans. Softwa Engineering*, SE-17 (4), pp. 370-82, 1991.

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

[16] P. I. Pignal, "An Analysis of Hardware and Softw Availability Exemplified on the IBM-37 Communication Controller", *IBM Journal of Resea and Development*, 32 (2), pp. 268-78, 1988.

[17] D. Powell, "Distributed Fault Tolerance: Lessons fr Delta-4", *IEEE Micro*, 14 (1), pp. 36-47, 1994.

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

[19] D. P. Siewiorek and R. S. Swarz, *The Theory a Practice of Reliable System Design,* Digital Press, 199

[20] G. E. Stark, "Dependability Evaluation of Integra Hardware/Software Systems", *IEEE Trans. Reliability*, R-36 (4), pp. 440-4, 1987.