MODÉLISATION DE LA SÛRETÉ DE FONCTIONNEMENT DU CAUTRA (Coordinateur AUtomatique du TRafic Aérien) AVEC PRISE EN COMPTE DES DÉFAILLANCES DU MATÉRIEL ET DU LOGICIEL

M Borrel, Karama Kanoun, T Morteveille, A Peytavin°

Paru dans les Actes du 9ème Colloque International de Fiabilité et de Maintenabilité - La Baule 30 Mai - 3 Juin 1994

MODÉLISATION DE LA SÛRETÉ DE FONCTIONNEMENT DU CAUTRA (Coordinateur AUtomatique du TRafic Aérien) AVEC PRISE EN COMPTE DES DÉFAILLANCES DU MATÉRIEL ET DU LOGICIEL

M. Borrel*◊ K. Kanoun◊ T. Morteveille* A. Peytavin°

° CENA
7, Avenue Edouard Belin B.P.4005
31055 Toulouse Cedex
Tél : (33) 62 25 95 00

◊LAAS-CNRS
7, Avenue du Colonel Roche
31077 Toulouse Cedex
Tél : (33) 61 33 62 00

**SRTI SYSTEM*
Rue Max Planck B.P 457
31315 Labège Cedex
Tél : (33) 62 24 52 28

RÉSUMÉ - Ce papier présente une approche originale de la modélisation de la sûreté de fonctionnement de systèmes informatiques complexes, par les réseaux de Petri stochastiques et les chaînes de Markov ; approche tenant compte, du caractère permanent ou temporaire des fautes du matériel et du logiciel, et de la propagation d'erreur du matériel au logiciel et entre composants logiciel. L'application de cette approche au CAUTRA (Coordinateur AUtomatique du TRafic Aérien) permet de comparer les résultats d'indisponibilité pour quatre architectures possibles de ce système.

ABSTRACT - *In this paper we present an original approach for modeling the dependability of complex computer systems using stochastic Petri nets and Markov chains. This approach takes into account the temporary and permanent character of both hardware and software faults. Moreover, the propagation of errors from the hardware to the software or between software components is considered. The application of the presented modeling approach to the CAUTRA system (the French automated Air Traffic Control system) allows the results of unavailability of four possible architectures of this system to be compared.*

1 - INTRODUCTION

Le système d'assistance automatisée au contrôle du trafic aérien, dénommé CAUTRA, regroupe les moyens informatiques de traitement des plans de vol et des données radar d'un Centre Régional de la Navigation Aérienne (CRNA). Il entre dans la catégorie des systèmes informatiques temps réel dont les fonctions sont critiques et dont le niveau de disponibilité doit être très élevé. Les techniques de tolérance aux fautes, telles que la redondance du matériel et du logiciel, permettent d'améliorer la disponibilité de tels systèmes. Cependant elles augmentent la complexité de ces systèmes et rendent plus difficile l'évaluation de leur disponibilité. Une modélisation globale de la sûreté de fonctionnement de ces systèmes, vis-à-vis des défaillances du matériel et du logiciel, s'avère donc indispensable pour le choix et la validation de l'architecture finale.

L'étude présentée dans ce papier fait partie d'un projet, plus global, de conception de nouvelles architectures du système de contrôle du trafic aérien. Elle concerne la modélisation de la sûreté de fonctionnement du CAUTRA par les réseaux de Petri stochastiques généralisés et les chaînes de Markov. Une approche originale de modélisation est développée : deux réseaux représentant respectivement le comportement d'un composant logiciel et d'un calculateur sont établis ; ces réseaux servent de base pour la construction progressive des modèles d'architectures possibles pour le CAUTRA. A noter qu'il est tenu compte à la fois des fautes temporaires et des fautes permanentes du matériel et du logiciel ainsi que de la propagation possible des erreurs du matériel au logiciel ou d'un composant logiciel vers un autre composant logiciel.

Le papier est organisé de la façon suivante : le paragraphe 2 présente le CAUTRA, l'approche de la modélisation et les modèles de base sont décrits au paragraphe 3, le paragraphe 4 est consacré à la modélisation de quatre architectures possibles du CAUTRA et présente les résultats de comparaison de ces architectures ainsi qu'une étude de sensibilité aux variations des valeurs numériques des paramètres de l'architecture retenue.

2 - PRÉSENTATION DU CAUTRA

Les deux principales applications du CAUTRA sont le Système de Traitement des Plans de Vol (STPV) et le Système de Traitement Radar (STR). Le STPV est chargé de faire vivre le plan de vol dans le CRNA. Il fournit aux contrôleurs les informations relatives aux avions de leurs secteurs de contrôle et traite toute information que peut lui transmettre un contrôleur via l'écran tactile de son Digitatron. Le STR élabore, à partir des données issues des radars, une image synthétique et représentative de la situation aérienne. La corrélation plan de vol, grâce à la communication du STR avec le STPV, permet au STR d'enrichir l'image avec des renseignements obtenus dans les Plans de vol sur les avions détectés par les radars. L'image ainsi "habillée" est transmise au contrôleur sur l'écran radar.

Jusqu'au printemps 1993, l'architecture matérielle du CAUTRA était calquée sur la structure du logiciel. Chacun des cinq logiciels, STPV principal, STPV secours (redondance passive), STR principal, STR secours (redondance active) et STR de test, était mis en œuvre sur un des cinq calculateurs MITRA. Le basculement entre les logiciels principal et secours du STPV se faisait manuellement, tandis que le basculement entre le STR principal et le STR secours était automatique.

Suite à l'accroissement du trafic aérien et à la saturation des calculateurs, la Direction de la Navigation Aérienne a décidé dans un premier temps de porter à l'identique la structure du logiciel présentée ci-dessus, sur trois calculateurs DATA GENERAL de la gamme ECLIPSE MV, qui permettent la cohabitation des logiciels STR et STPV sur un même calculateur. A terme (vers 1998) les logiciels seront découpés en plusieurs entités distribuées sur les calculateurs par le biais de "machines virtuelles". Le basculement des entités principales en entités de secours se fera automatiquement, aussi bien pour le STR que pour le STPV, par des logiciels de supervision mis en œuvre pour une partie sur les calculateurs DATA GENERAL et pour l'autre sur des stations de travail UNIX [Borr 93].

L'approche et les résultats présentés dans ce papier constituent une première étape pour la modélisation de la future architecture. Les résultats concernent l'architecture existante, architecture intermédiaire constituée par les calculateurs DATA GENERAL et les logiciels actuels. Seuls deux calculateurs sont réservés aux applications temps réel. L'étude concerne la modélisation de ces deux calculateurs et des quatre logiciels : STR principal et secours, STPV principal et secours. Des "chiens de garde" détectent une partie des défaillances du matériel, au niveau de chaque calculateur. Des procédures de traitement d'erreur (traitement d'exception), internes aux logiciels, permettent d'éliminer les fautes temporaires également appelées fautes non solides ou douces. Les fautes volatiles qui disparaissent d'elles mêmes entrent dans la catégorie de ces fautes douces. Par contre les fautes permanentes nécessitent d'être passivées, par une relance du logiciel : elles sont dites solides ou dures. Les fautes solides du logiciel sont en partie tolérées par le mécanisme de basculement, automatique pour le STR, manuel pour le STPV, entre les versions principale et secours de chaque application.

3 - APPROCHE DE MODÉLISATION

Bien que l'utilisateur final d'un système soit intéressé en premier lieu par une évaluation globale de la sûreté de fonctionnement de son système, il n'y a eu que très peu d'études tenant compte à la fois du matériel et du logiciel [Cost 78, Angu 82, Star 87, Lapr 92, Duga 94] comparé au nombre relativement élevé de modélisations dédiées au matériel.

L'originalité de notre approche consiste à construire deux réseaux de Petri stochastiques modélisant respectivement le comportement d'un composant logiciel et d'un calculateur vis-à-vis des fautes temporaires et permanentes. Basés sur ces deux réseaux, nous construisons successivement 1) le modèle de comportement de deux calculateurs, puis 2) de deux composants logiciel communiquants, ensuite 3) de deux répliques d'un même composant logiciel (communiquantes ou non), 4) d'un calculateur et d'un composant logiciel, et enfin 5) le modèle de deux composants logiciel répliqués sur deux calculateurs. Les dépendances entre les composants sont gérées à l'aide de réseaux de connexion assurant les liens entre les différents réseaux de base. Pour une architecture du matériel et une architecture du logiciel données, les différentes possibilités de répartition du logiciel sur le matériel n'entraînent aucune modification du modèle du matériel et du modèle du logiciel. Par contre les réseaux de connexion relient différemment les deux

modèles. Ces réseaux de connexion permettent ainsi de modéliser a) la propagation des erreurs d'un calculateur vers un composant logiciel ou d'un composant logiciel vers un autre composant logiciel, b) la reconfiguration d'un composant de secours vers le composant principal, c) l'arrêt d'utilisation du logiciel en cas de défaillance du calculateur, d) la priorité de réparation des calculateurs ou de relance d'un logiciel après restauration du matériel. A chaque étape, les chaînes de Markov générées par l'outil SURF-2 [Metg 94] sont validées. Les modèles du CAUTRA utilisés dans le paragraphe 4 sont construits à partir des deux réseaux de base et des réseaux de connexion.

3.1 - Modélisation d'un calculateur

Les fautes temporaires constituent la grande majorité des fautes du matériel [Lapr 93]. Ces fautes ne conduisent pas toujours le calculateur à défaillance. Elles induisent des erreurs qui peuvent propager d'autres erreurs aux logiciels avant de disparaître d'elles-mêmes. Le modèle du matériel, donné à la Fig.1.a, distingue les fautes temporaires des fautes permanentes. Les différents états d'un calculateur sont modélisés de la façon suivante (Fig.1.a) :

- La place "C_ok" correspond à l'état de bon fonctionnement du calculateur, sans faute activée.
- Au niveau de la place "C_e" l'activation d'une faute, par un taux λ_m , produit une erreur dans l'état interne du calculateur.
- Cette erreur provient, soit de l'activation d'une faute temporaire avec une probabilité 1-p_m : place "C_t", soit de l'activation d'une faute permanente avec une probabilité p_m : place "C_d".
- L'erreur provenant de l'activation d'une faute temporaire disparaît d'elle-même dans un bref délai $1/\delta_m$, cependant, nous le verrons au paragraphe 3.6, elle peut propager une erreur aux composants logiciel mis en œuvre sur le calculateur.
- Une faute permanente conduit à une défaillance du calculateur nécessitant la réparation de ce dernier avec un taux μ_m .

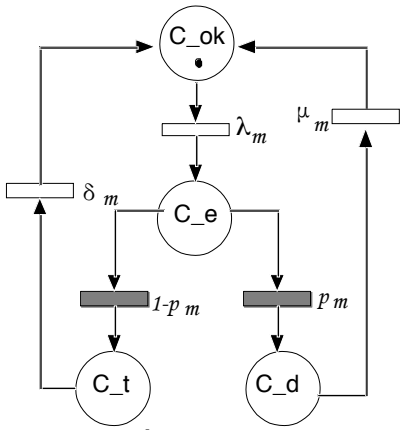


Fig. 1.a : Modèle d'un calculateur

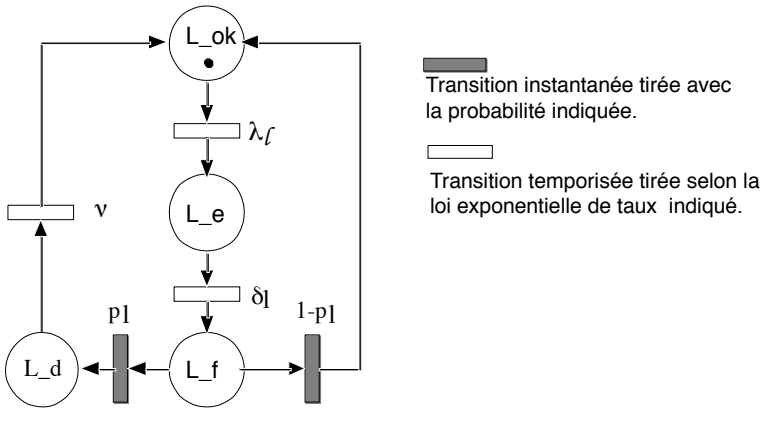


Fig. 1.b : Modèle d'un composant logiciel

3.2 - Modélisation d'un composant logiciel

Les fautes temporaires ne concernent pas uniquement le matériel [Lapr 93]. La majorité des fautes du logiciel, présentes pendant la vie opérationnelle, sont aussi des fautes temporaires [Elme 72, Gray 86]. Le modèle du matériel proposé admet que les fautes temporaires et permanentes sont distinguables dès leur activation par leurs conséquences respectives : disparition avec propagation possible d'erreur vers le logiciel, ou défaillance et arrêt du calculateur. Par contre, dans le cas du logiciel, la distinction est souvent plus complexe, et ne sera faite qu'après le traitement d'erreur. En outre, si l'activation d'une faute temporaire du matériel peut propager une erreur à un composant logiciel, l'hypothèse est faite qu'une erreur d'un composant logiciel peut propager une erreur à un autre composant logiciel uniquement si elle n'a pas été

éliminée par le traitement d'erreur. De ce fait nous proposons un modèle du logiciel différent de celui du matériel. Dans ce modèle nous ne parlons plus de fautes permanentes et temporaires mais de fautes dures (ou solides) et de fautes douces (ou non solides) par référence au processus qu'il faut mettre en œuvre pour les éliminer (respectivement passivation ou traitement d'erreur). Le réseau de Petri de la Fig.1.b modélise le comportement d'un composant logiciel vis-à-vis des fautes solides et non solides.

- La place "L_ok" correspond à l'état de bon fonctionnement du composant logiciel, sans faute activée.
- Les fautes sont activées avec un taux λ_f qui donne lieu à une erreur : place "L_e".
- L'erreur est traitée, par un traitement d'exception, pendant un laps de temps très court $1/\delta_f$. A la fin du traitement d'exception, visualisé par la place "L_f", si la faute est non solide avec une probabilité $1-p_f$, elle est éliminée et le composant logiciel retrouve son fonctionnement normal. Par contre, si la faute est solide avec une probabilité p_f, elle doit être passivée et le composant logiciel est considéré défaillant : place "L_d". La faute est passivée par une relance du composant logiciel de taux ν.

Dans le cas d'un composant logiciel sans traitement d'erreur, le réseau modélise simplement son comportement face aux fautes volatiles et aux fautes permanentes.

3.3 - Modélisation de deux calculateurs

On considère deux calculateurs redondants C1 et C2. Leur modèle de comportement construit à partir de deux réseaux de Petri (Fig.1.a), modifiés et reliés par un réseau de synchronisation des réparations respectives de chaque calculateur, est donné Fig.2. Le marquage de la place "Rép" indique le nombre de réparateurs disponibles. Les transitions instantanées, en amont des transitions de taux μ_m, permettent de gérer une file d'attente en réparation du type "premier arrivé premier servi".

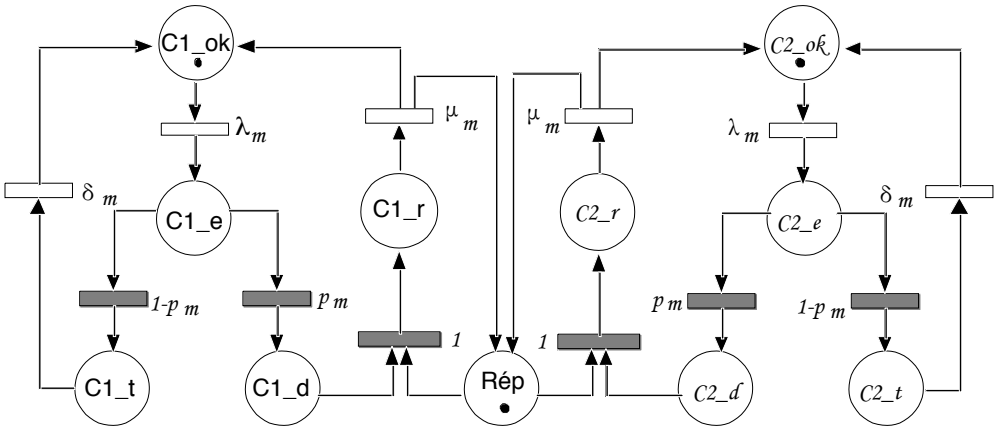


Fig. 2 : Modélisation de deux calculateurs avec un réparateur

3.4 - Modélisation de deux composants logiciel communiquants

Considérons deux composants logiciel L1 et L2. Si L1 effectue un traitement qu'il transmet à L2, les erreurs de L1 sont susceptibles de propager d'autres erreurs à L2. Le réseau de Petri de la Fig.3, modélisant la propagation d'erreur du logiciel, est construit à partir de deux réseaux (Fig.1.b), reliés par un autre réseau appelé "réseau de propagation d'erreur" de L1 vers L2. En cas de propagation bilatérale, il faut rajouter un réseau "propagation d'erreur" symétrique de celui présenté.

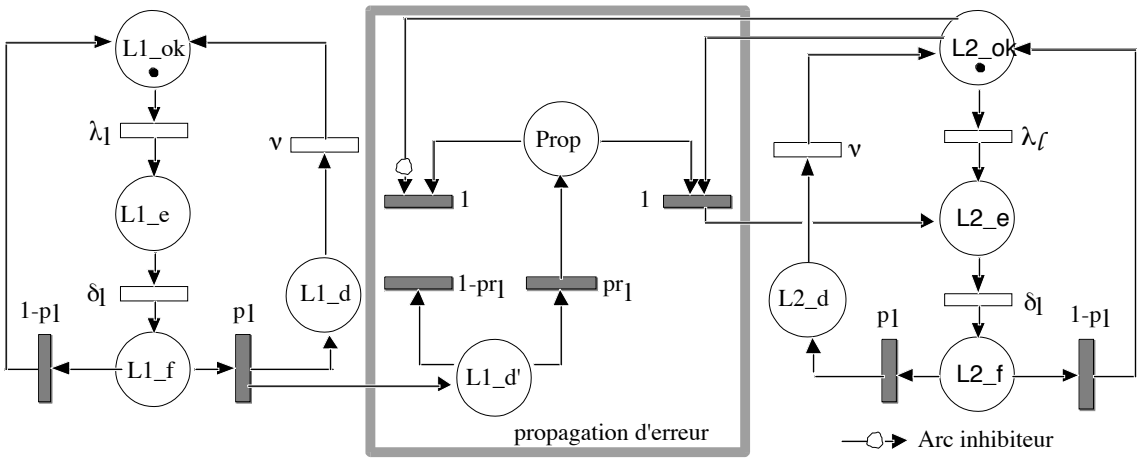


Fig. 3 : Modélisation de la propagation d'erreur de L1 vers L2

Par hypothèse, une erreur ne se propage pas si elle a été éliminée par le processus de traitement d'erreur (voir §3.2). Une erreur de L1 non éliminée se propage avec une probabilité pr_f (place Prop) ou ne se propage pas avec une probabilité $1\text{-}pr_f$. L'erreur propagée à L2 est traitée au même titre que les autres erreurs de L2.

3.5 - Modélisation de deux répliques d'un même composant logiciel

L'objectif du réseau de Petri de la Fig.4, est de modéliser les principes de tolérance aux fautes entre deux répliques d'un même composant logiciel, dénotées principal (pal) et secours (sos), conformément aux hypothèses suivantes :

- si le sos est défaillant, il est relancé ;
- si le pal est défaillant, il passe en mode sos, réciproquement le sos passe en mode pal, le pal devenu sos est relancé, conformément à l'action précédente ;
- si le pal et le sos sont défaillants, le pal est relancé en priorité ;
- le basculement peut échouer (mode commun de défaillance par exemple).

Le modèle de la Fig.4 est construit à partir de deux réseaux identiques (Fig.1.b) reliés par un réseau appelé "réseau de reconfiguration". Selon que les répliques communiquent ou effectuent leurs traitements de façon indépendante, le "réseau de propagation d'erreur" est présent ou non. Seule la reconfiguration suite à la défaillance de L_pal est présentée sur la figure pour des raisons de clarté. Pour avoir la reconfiguration complète du logiciel, il faut rajouter le réseau symétrique modélisant la reconfiguration suite à la défaillance de L'_pal.

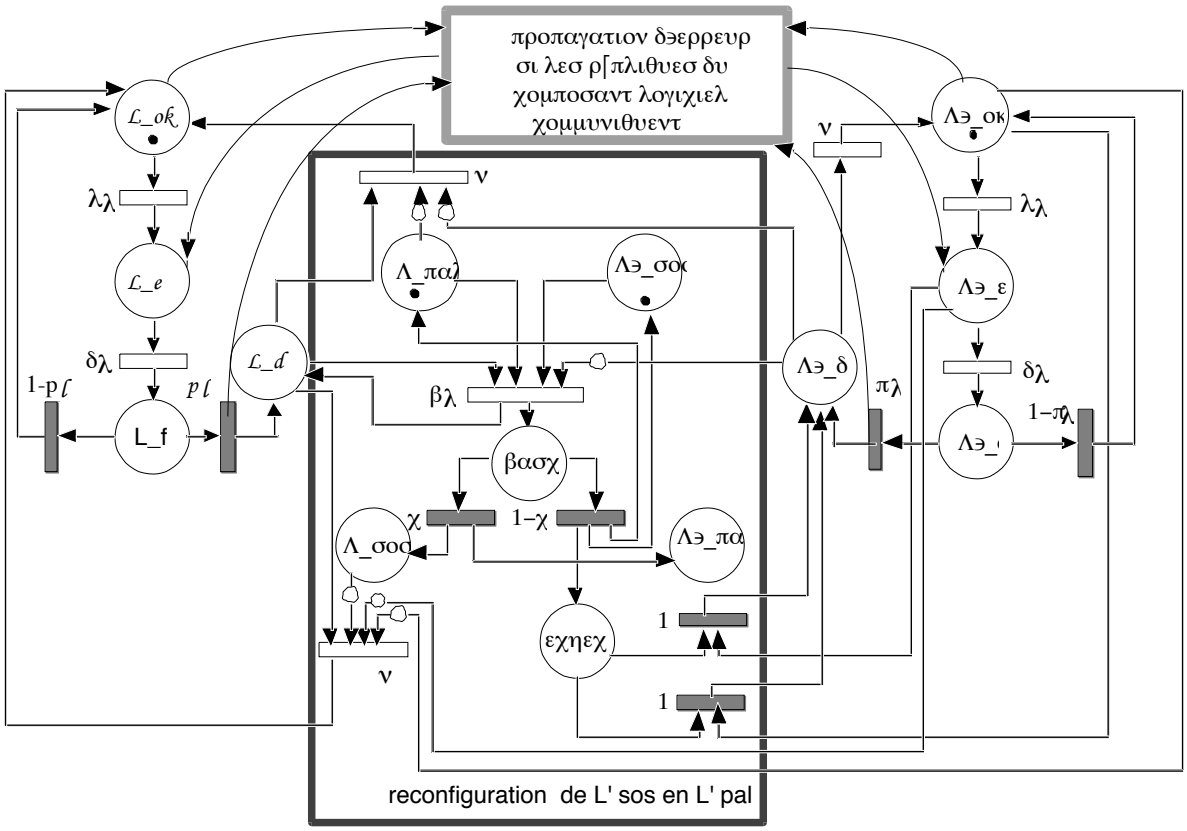


Fig. 4 : Modélisation de la reconfiguration entre deux répliques de logiciel

Deux places "pal" et "sos" sont affectées à chaque réplique, leur marquage indique à tout instant les modes de fonctionnement des répliques. Si L est défaillant et en mode pal, L' non défaillant et en mode sos, une tentative de basculement a lieu, de durée moyenne $1/b_f$, place "basc". Le basculement peut réussir avec une probabilité c, correspondant au facteur de couverture de la reconfiguration du logiciel. L passe en mode sos et L' passe en mode pal. Le basculement peut échouer avec une probabilité 1-c, les répliques du composant logiciel gardent leur mode de fonctionnement respectif et L' est considéré défaillant (hypothèse pessimiste) pour que L puisse être relancé en mode pal. Les arcs inhibiteurs sur les transitions de relance, ne permettent de relancer une réplique que si elle était

en mode pal et que le sos est défaillant ou que si elle est en mode sos et que le pal n'est pas défaillant. Le réseau complet comprend un réseau de reconfiguration symétrique, relatif à la défaillance de L'.

3.6 - Modélisation d'un calculateur et d'un composant logiciel

Le réseau présenté à la Fig.5 modélise les dépendances entre un calculateur et un composant logiciel. Il est construit à partir des réseaux matériel et logiciel de base (Fig 1.a et 1.b), reliés par deux réseaux de connexion. Le premier, appelé "réseau de propagation d'erreur" (qui est identique à celui de la Fig.3) modélise la propagation d'erreur du matériel au logiciel, lorsque par exemple une faute temporaire du matériel provoque avant sa disparition l'écriture d'une donnée erronée dans un registre mémoire qui sera lu par le composant logiciel, avant qu'une autre donnée soit réécrite au même emplacement mémoire. Le deuxième, appelé "réseau arrêt ou indisponibilité du composant logiciel suite à la défaillance du matériel", modélise l'indisponibilité du composant logiciel lorsque le calculateur sur lequel il est mis en œuvre est défaillant.

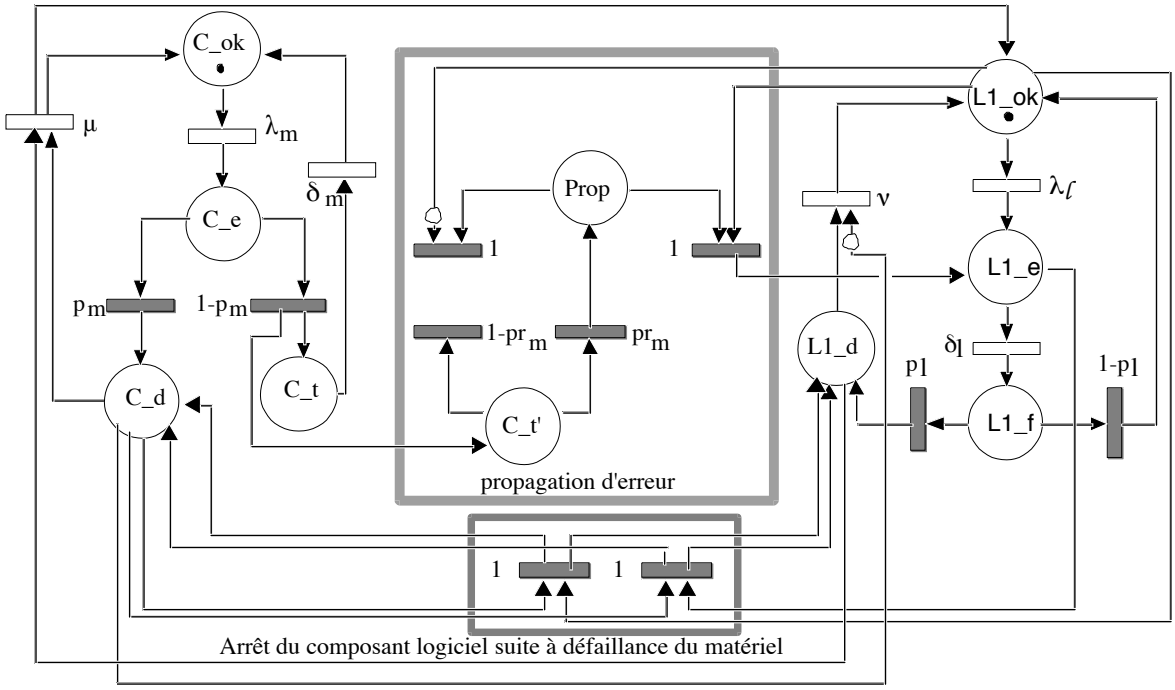


Fig. 5 : Les dépendances entre un calculateur et un composant logiciel

Le taux de réparation μ comprend le taux de réparation μ_m du calculateur et le taux de relance ν du composant logiciel. L'arc inhibiteur reliant la place c_d, où le calculateur est défaillant, et la transition de relance d'un composant logiciel mis en œuvre sur ce calculateur, n'autorise la relance du composant logiciel que si le calculateur n'est pas défaillant.

3.7 - Modélisation de deux répliques d'un composant logiciel sur deux calculateurs

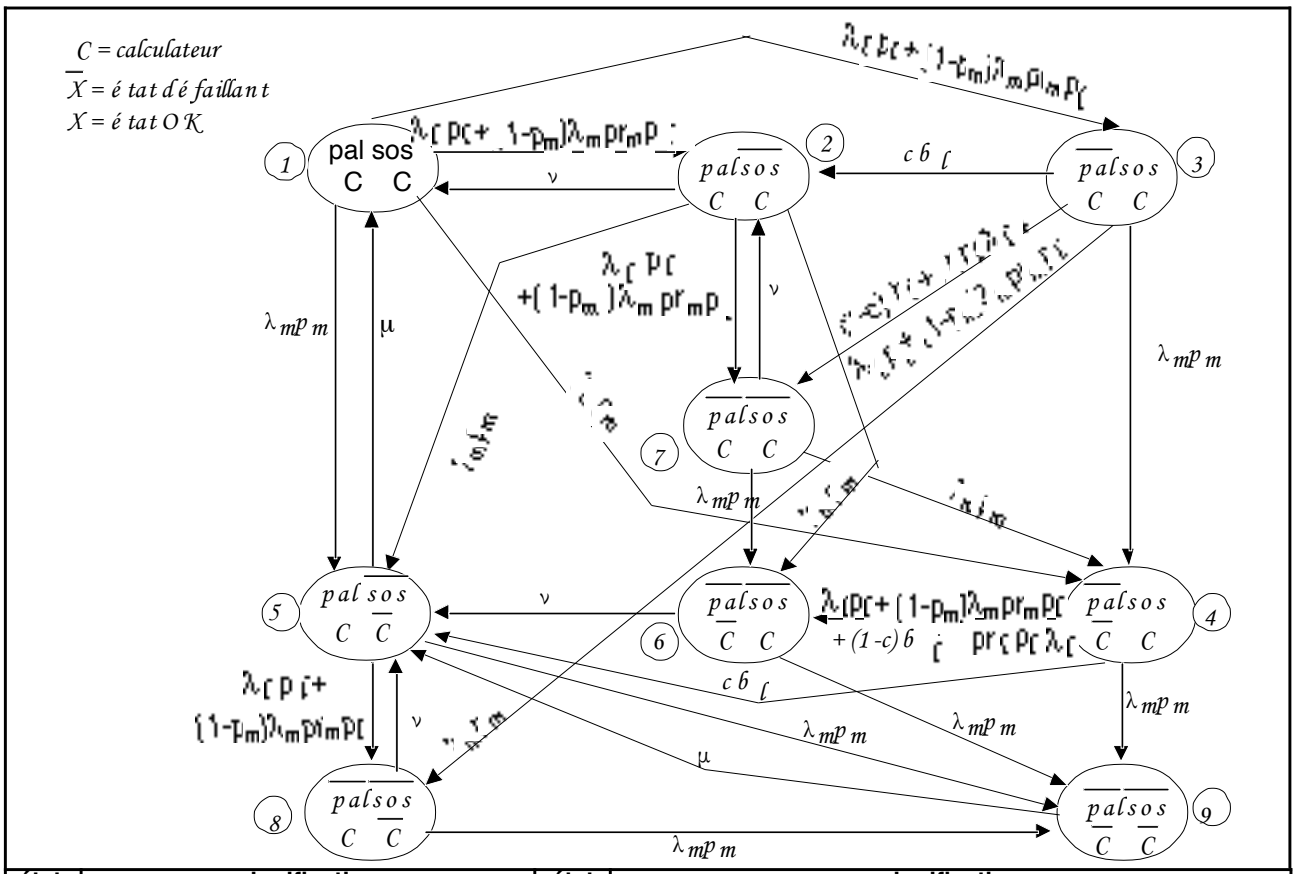
Afin d'augmenter sa disponibilité, les versions pal et sos d'une application du logiciel sont souvent mises en œuvre sur deux calculateurs différents. Nous supposons ici que le pal peut propager des erreurs au sos (communication unilatérale). Le réseau de Petri modélisant cette architecture est construit à partir des réseaux des Fig.2, 4 et 5 qui illustrent respectivement les relations de dépendance entre les deux calculateurs, les deux répliques d'un même composant logiciel et entre un calculateur et un composant logiciel. A ces dépendances, il faut ajouter des dépendances dues aux priorités induites par les procédures de restauration : relance d'un composant logiciel et réparation d'un calculateur. Elles sont généralement modélisées par des arcs inhibiteurs. Ces relations de dépendances traduisent les hypothèses suivantes :

- le calculateur du composant logiciel sos est relancé si l'homologue pal n'est pas défaillant,
- si les deux calculateurs sont défaillants, le calculateur du pal est relancé en premier,
- si le calculateur du pal est défaillant et que le sos présente une défaillance du logiciel, ce dernier

est relancé en mode pal, le pal deviendra sos après la restauration du calculateur,

- la relance des composants logiciel ne peut se faire sur un calculateur défaillant.

Changer l'une de ces hypothèses reviendrait à changer uniquement les liaisons entre les différents réseaux de base. La chaîne de Markov générée par SURF-2 à partir du réseau de Petri construit selon les hypothèses d'interaction et de reconfiguration ci-dessus, est donnée à la Fig.6. Les états de traitement d'erreur dont les temps de séjour sont très courts ($1/\delta_m$ et $1/\delta_\ell$) ont été agrégés avec les états à temps de séjour plus long, afin de réduire le nombre d'états de la chaîne.



état	signification	état	signification
1	Aucune défaillance	6 à 9	Etats où le système est défaillant
2	Défaillance du logiciel sos	6	Défaillances du calculateur affecté au pal et du logiciel sos
3	Défaillance du logiciel pal	7	Défaillances des logiciels pal et sos
4	Défaillance du calculateur affecté au pal	8	Défaillances du calculateur affecté au sos et du logiciel pal
5	Défaillance du calculateur affecté au sos	9	Défaillances des calculateurs affectés au pal et au sos

Fig. 6 : Chaîne de Markov de deux répliques d'un même composant logiciel sur deux calculateurs

4 - APPLICATION AU CAUTRA

Notre objectif est de définir, à partir de l'architecture actuelle du CAUTRA, quelques variantes de celle-ci afin d'étudier à la fois l'influence de la structure du système et celle des procédures de reconfiguration sur l'indisponibilité du CAUTRA. Nous définissons ainsi quatre architectures possibles ayant les mêmes composants matériel et logiciel. Elles diffèrent par la répartition des composants du logiciel sur les deux calculateurs et par les procédures de reconfiguration après défaillance du système. Pour ces quatre architectures nous avons donc le même modèle du matériel et le même modèle du logiciel. La prise en compte de la structure spécifique d'une architecture et de sa procédure de reconfiguration se fait à l'aide des réseaux de connexion tels que ceux définis dans le paragraphe 3. Les modèles du logiciel et du matériel sont construits à l'aide des réseaux de base du paragraphe 3. Au fur et à mesure de la construction des modèles, les chaînes de Markov associées au réseau de Petri, générées automatiquement par SURF-2, permettent de vérifier que les modèles représentent bien le comportement réel du système.

4.1 - Architectures considérées

Chacune des quatre architectures proposées comprend deux calculateurs et deux répliques de chaque application, STPV et STR. Elles diffèrent, d'une part, par la répartition des composants logiciel sur les calculateurs, appelée structure (les deux structures sont présentées Fig.7) :

- structure 1 : STR pal et STPV sos sur un calculateur, STR sos et STPV pal sur l'autre,
- structure 2 : STR pal et STPV pal sur un calculateur, STR sos et STPV sos sur l'autre,

et d'autre part, par leur type de reconfiguration envisageable :

- reconfiguration 1 : après basculement et restauration du secours les logiciels basculent à nouveau pour revenir dans l'architecture de départ,
- reconfiguration 2 : après basculement et restauration, les composants logiciel conservent leur mode de fonctionnement.

Ceci nous amène donc à considérer les quatre architectures suivantes :

- A1.1 : structure 1 et reconfiguration 1, préconisée actuellement dans les CRNAs,
- A1.2 : structure 1 et reconfiguration 2,
- A2.1 : structure 2 et reconfiguration 1,
- A2.2 : structure 2 et reconfiguration 2.

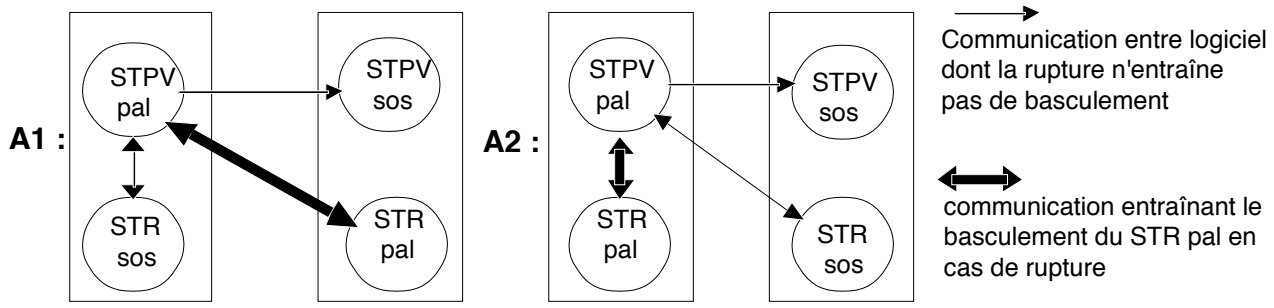


Fig. 7 : Structure des architectures considérées

4.2 - Les modèles

Le modèle du matériel, deux calculateurs et un réparateur, a été présenté Fig. 2. Pour chacune des deux applications STR ou STPV, le modèle du logiciel correspond à celui de deux répliques d'un même composant logiciel (Fig.4), avec un "réseau de propagation d'erreur" du STPV pal vers le STPV sos, car le STPV pal effectue un traitement préliminaire qu'il transmet au STPV sos. Le STPV pal communiquant de façon bilatérale avec le STR pal et le STR sos, les deux réseaux ci-dessus sont reliés par des "réseaux de propagation d'erreur" (Fig.3) qui modélisent la propagation d'erreur bilatérale entre le STPV pal et le STR pal, et, entre le STPV pal et le STR sos. Les modèles du logiciel et du matériel restent identiques quelle que soit la répartition des applications STR et STPV sur les DATA GENERAL. Ce sont les réseaux de connexion ("réseau propagation d'erreur du matériel au logiciel" et "réseau arrêt ou défaillance du logiciel suite à défaillance du matériel") qui relient différemment les logiciels aux calculateurs selon la répartition modélisée.

4.3- Les hypothèses de dépendance et de modélisation

Pour chacune des applications STR ou STPV, les composants logiciel pal et sos sont reliés aux calculateurs sur lesquels ils sont mis en œuvre, de façon identique au modèle présenté Fig.5. Cependant du fait de la présence de deux applications, de la même façon que dans le paragraphe 3.7, des priorités supplémentaires sont établies à l'aide d'arcs inhibiteurs, avec les hypothèses suivantes :

- Dans tous les cas de relance de composant ou de réparation de calculateur, le STR est prioritaire par rapport au STPV. Par exemple si le STPV pal et le STR pal sont chacun sur un calculateur et si ces deux calculateurs sont défaillants, le calculateur du STR pal est relancé en priorité.
- Le basculement du STR pal vers le STR sos peut se produire sur la rupture de communication du STR pal avec le STPV pal. Le taux de défaillance de la communication entre le STR pal et le STPV pal est lié à la répartition du logiciel sur les calculateurs. Si les applications sont sur deux

calculateurs différents, il faut prendre en compte le support de communication inter-calculateur. Les places et les transitions modélisant la défaillance de cette communication inter-calculateur font partie des relations de dépendance entre les modèles du logiciel et du matériel.

En ce qui concerne les valeurs numériques des paramètres, nous considérons, sauf indication contraire, les valeurs nominales suivantes (l'ordre de grandeur de l'inverse des taux de défaillance et de réparation correspond aux durées moyennes observées dans la réalité) :

$1/\lambda_m$	$1/\mu$	$1/\lambda_{STR}$	$1/\nu_{STR}$	$1/\lambda_{STPV}$	$1/\nu_{STPV}$	$1/b_{STR}$	$1/b_{STPV}$	p_m	p_f	pr_m	c
10 000h	20h	1000h	1 min	200h	10 min	1 s	1 min	0,6	0,4	0,85	0,98

4.4 - Comparaison des quatre architectures

Les architectures comparées sont celles présentées au paragraphe 4.1. Soit λ_{c1} le taux de défaillance de la communication STPV pal /STR pal inter-calculateur pour la structure 1, et λ_{c2} celui de la communication interne au calculateur de la structure 2. Introduisons le rapport $r = \lambda_{c1}/\lambda_{c2}$. La comparaison des résultats d'indisponibilité du STR pour les quatre architectures se fait en fonction de r.

Les résultats obtenus avec SURF-2 (Tab.1 et Tab.2) montrent qu'en terme d'indisponibilité asymptotique du STPV et du STR les résultats sont égaux pour la reconfiguration 2 quelle que soit la structure (A1.2 et A2.2). Le rapport r des taux de défaillance de la communication STPV pal / STR pal, lorsque les deux logiciels sont sur le même calculateur ou sur deux calculateurs différents, influe sur l'indisponibilité du STR (Tab.2). Pour un rapport inférieur ou égal à 10, A1.1 est préférable à A2.1. Par contre, pour un rapport supérieur à 10, A2.1 donne l'indisponibilité minimale pour le STR. La valeur élevée de l'indisponibilité du STPV pour l'architecture A1.1 s'explique par les hypothèses de maintenance qui donnent la priorité de réparation au calculateur du STR pal lorsque les deux calculateurs sont défaillants. Pour cette architecture, l'indisponibilité en heure par an du STPV est directement liée à la durée de réparation du calculateur.

Tab.1 : Indisponibilité du STPV pour A1 et A2

Indisp. A1.1		Indisp. A2.1	
Prob.	h/an	Prob.	h/an
1,57E-03	13,753	1,72E-04	1,5067

Indisp. A1.2		Indisp. A2.2	
Prob.	h/an	Prob.	h/an
2,28E-04	1,9973	2,28E-04	1,9973

L'architecture A2.1 donne aussi l'indisponibilité minimale (Tab.1) pour le STPV, du fait des priorités entre STR et STPV concernant la relance des logiciels et la réparation des calculateurs. En considérant le CAUTRA dans sa globalité (STR + STPV) nous en déduisons que c'est l'architecture A2.1 qui donne l'indisponibilité minimale.

Tab.2 : Indisponibilité du STR en fonction du rapport r pour les deux structures

	Indisp. A1.1		Indisp. A2.1	
r	Prob.	h/an	Prob.	h/an
1	2,13E-05	0,1866	2,15E-05	0,1883
10	2,14E-05	0,1875	2,15E-05	0,1883
100	2,28E-05	0,1997	2,16E-05	0,1892
1000	3,51E-05	0,3075	2,16E-05	0,1892
10 000	1,59E-04	1,3928	2,17E-05	0,1901

	Indisp. A1.2		Indisp. A2.2	
r	Prob.	h/an	Prob.	h/an
1	2,16E-05	0,1892	2,16E-05	0,1892
10	2,17E-05	0,1901	2,17E-05	0,1901
100	2,22E-05	0,1945	2,22E-05	0,1945
1000	2,46E-05	0,2155	2,46E-05	0,2155
10 000	2,64E-05	0,2313	2,64E-05	0,2313

4.5 - Etudes de sensibilité

Des études de sensibilité ont été effectuées sur l'ensemble des autres paramètres, les résultats sont donnés pour l'architecture A2.1. Les modèles sont très sensibles au taux d'activation des fautes du matériel (Tab.3), et dans une moindre mesure au taux d'activation des fautes du logiciel. L'influence de l'augmentation du taux d'activation des fautes du matériel est plus importante que celle du taux d'activation des fautes du logiciel en raison de la propagation d'erreur du matériel vers le logiciel.

Tab.3 : Indisponibilité de A2.1 en fonction du taux d'activation des fautes du matériel et du logiciel

	Indisp. STR		Indisp. STPV	
λ_m	Prob.	h/an	Prob.	h/an
1E-04	2,13E-05	0,1866	1,72E-04	1,507
1E-03	4,01E-04	3,5128	1,24E-03	10,862
5E-03	5,83E-03	51,071	9,27E-03	81,205

Variation de λ_m

	Indisp. STR		Indisp. STPV	
λ_{STR}	Prob.	h/an	Prob.	h/an
1E-04	4,23E-06	0,0371	2,96E-05	0,2593
1E-03	2,13E-05	0,1866	1,72E-04	1,5067
5E-03	9,38E-05	0,8217	1,49E-03	13,052

Variation de λ_{STR} avec $\lambda_{STPV} = 5\ \lambda_{STR}$

A taux d'activation des fautes du matériel et du logiciel fixes, les résultats (Tab.4) mettent en évidence l'influence de la proportion de fautes permanentes ou solides sur l'indisponibilité du STR et du STPV. Le seuil de sensibilité se situe entre les proportions 0,2 et 0,4 : l'indisponibilité du STR est multipliée alors par un facteur 2 et celle du STPV par un facteur 1,5.

Tab.4 : Indisponibilité de A2.1 en fonction des proportions de fautes solides ou permanentes

	Indisp. STR		Indisp. STPV	
p_m	Prob.	h/an	Prob.	h/an
0,2	7,28E-06	0,064	8,38E-05	0,734
0,4	1,47E-05	0,129	1,20E-04	1,051
0,6	2,25E-05	0,197	1,56E-04	1,367
0,8	3,08E-05	0,27	1,92E-04	1,682

Variation de p_m pour λ_m = 1E-04

	Indisp. STR		Indisp. STPV	
p_f	Prob.	h/an	Prob.	h/an
0,2	1,28E-05	0,112	8,78E-05	0,769
0,4	2,53E-05	0,222	1,56E-04	1,367
0,6	3,22E-05	0,282	2,23E-04	1,954
0,8	4,18E-05	0,366	2,88E-04	2,523

Variation de p_f pour λ_{STR} = 1E-03

Pour un taux d'activation de fautes, et une proportion de fautes permanentes, fixes pour le matériel et le logiciel, les résultats (Tab.5) montrent que le STPV est plus sensible à la probabilité de propagation d'erreur entre logiciels que le STR, en raison de la propagation d'erreur entre le STPV pal et le STPV sos, mais qui n'existe pas entre le STR pal et le STR sos.

Tab.5 : Indisponibilité de A2.1 en fonction des probabilités de propagation d'erreur

	Indisp. STR		Indisp. STPV	
pr_m	Prob.	h/an	Prob.	h/an
0	2,07E-05	0,1813	2,20E-04	1,9272
0,4	2,11E-05	0,1848	2,22E-04	1,9447
0,6	2,14E-05	0,1875	2,22E-04	1,9447
1	2,18E-05	0,1910	2,23E-04	1,9535

Variation de pr_m

	Indisp. STR		Indisp. STPV	
pr_f	Prob.	h/an	Prob.	h/an
0	2,13E-05	0,1866	1,72E-04	1,5067
0,4	2,18E-05	0,1910	3,52E-04	3,0835
0,6	2,22E-05	0,1945	4,36E-04	3,8194
1	2,35E-05	0,2059	6,04E-04	5,2910

Variation de pr_f

L'influence de la variation du taux de couverture de la reconfiguration est plus importante pour l'indisponibilité du STPV (Tab.6) que pour celle du STR, car la durée de relance manuelle du logiciel, en cas d'échec de la reconfiguration du logiciel, est environ 10 fois plus élevée pour le STPV que pour le STR.

Tab.6 : Indisponibilité de A2.1 en fonction du facteur de couverture c de la reconfiguration du logiciel

	Indisp. STR		Indisp. STPV	
c	Prob.	h/an	Prob.	h/an
0,99	2,15E-05	0,1883	2,08E-04	1,8221
0,98	2,16E-05	0,1892	2,23E-04	1,9535
0,95	2,20E-05	0,1927	2,67E-04	2,3389
0,90	2,25E-05	0,1971	3,42E-04	2,9959
0,80	2,37E-05	0,2076	4,93E-04	4,3187
0,50	2,71E-05	0,2374	9,60E-04	8,4096

5 - CONCLUSION ET PERSPECTIVE

L'approche de modélisation que nous avons suivie dans ce papier nous a permis de construire et de valider progressivement les modèles de quatre architectures possibles. Les résultats d'évaluation nous ont permis d'une part d'effectuer un choix d'architecture pour le CAUTRA, et d'autre part de mettre en évidence l'impact du caractère solide ou non solide des fautes du matériel et du logiciel sur la disponibilité des applications du CAUTRA. Cependant les modèles établis concernent l'architecture actuelle du CAUTRA et supposent que les erreurs du matériel et du logiciel sont détectées avec une probabilité égale à 1. L'architecture future du CAUTRA mettra en œuvre des mécanismes de détection d'erreur sophistiqués et complexes qui comprendront plusieurs séries de logiciels, certains mis en œuvre sur les calculateurs DATA GENERAL, d'autres sur des stations de travail UNIX, ainsi que sur chaque calculateur. La conception modulaire des modèles établis jusqu'à présent va nous permettre d'y inclure les modifications apportées par la future architecture. L'objectif final est de contribuer par des évaluations quantitatives au choix de la future architecture du CAUTRA et plus généralement d'un CRNA.

REMERCIEMENTS :

Les auteurs tiennent à remercier Jean-Claude Laprie, Sylvain Metge et David Powell pour leur aide précieuse lors de la construction et validation des modèles, Alain Costes et Denis Spicq pour leurs critiques constructives lors de la relecture de ce papier, ainsi que toutes les personnes ayant participé à la relecture.

6 - BIBLIOGRAPHIE

[Angu82] J.E. Angus, L.E. James, "Combined hardware/software reliability models", *Actes Annual Reliability and Maintainability Symposium,* 1982, pp. 176-181.

[Borr 93] M.Borrel, "Description des sous-systèmes d'un Centre Régional de la Navigation Aérienne",rapport LAAS 93-229, STRI SYSTEM 27834/BRL/517, Note CENA 93-645, Juin 93.

[Cost 78] A. Costes, C. Landrault, J.C. Laprie, "Reliability and availability models for maintained systems featuring hardware failures and design faults", *IEEE Trans. on Computers, vol. C-27, no.6, juin 78, pp. 548-560*.

[Duga 94] J.B. Dugan, M. Lyu, "System-level reliability and sensitivity analysis for three fault-tolerant system architectures", *Actes 4th IFIP Int. Working Conf. on Dependable Computing for Critical Applications (DCCA4)*, San Diego, Janvier 1994, pp. 295-307.

[Elme 72] W.R. Elmendorf, "Fault-tolerant programming", *Actes 2nd IEEE Int. Symp. on Fault Tolerant Computing (FTCS-2)*, Newton, Massachusetts, Juin 1972, pp. 79-83.

[Gray86] J. Gray, "Why do computers stop and what can be done about it ?", *Actes 5th Symp. on Reliability in Distributed Software and Database Systems*, Los Angeles, Janvier 1986, pp. 3-12.

[Lapr 92] J.C. Laprie, K. Kanoun, "X-ware reliability and availability modeling", *IEEE Trans. on Software Engineering* vol. 18, no. 2, Février 1992, pp. 130-147.

[Lapr 93] J.C. Laprie, "On the temporary character of operation-persistent software faults", Actes 4th *Int. Symp. on Software Reliability Engineering,* Denver, Colorado , Novembre 93, pp. 125.

[Metg 94] S. Metge et al, "SURF-2 : outil d'évaluation de la sûreté de fonctionnement par chaînes de Markov et réseaux de Petri stochastiques", *Actes du 9ème Colloque International de Fiabilité et de Maintenabilité*, La Baule, France, 28 Mai-3 Juin 1994.

[Star 87] G.E. Stark, "Dependability evaluation of integrated hardware / software systems", *IEEE Trans. on Reliability,* vol. R-36, no. 4, 1987, pp. 440-444.