Modèle hyperexponentiel en temps continu et en temps discret pour l'évaluation de la croissance de la sûreté de fonctionnement

Mohamed Kaâniche

▶ **To cite this version:**

Mohamed Kaâniche. Modèle hyperexponentiel en temps continu et en temps discret pour l'évaluation de la croissance de la sûreté de fonctionnement. Réseaux et télécommunications [cs.NI]. Institut National Polytechnique de Toulouse - INPT, 1992. Français. NNT : . tel-00142181

HAL Id: tel-00142181

https://theses.hal.science/tel-00142181

Submitted on 17 Apr 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° d'ordre : 519 Année 1992

THÈSE

présentée au

Laboratoire d'Automatique et d'Analyse des Systèmes du CNRS

en vue d'obtenir le titre de

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE

SPéCIALITé : INFORMATIQUE

par

Mohamed KAANICHE

INGéNIEUR ENAC

MODÈLE HYPEREXPONENTIEL EN TEMPS CONTINU ET EN TEMPS DISCRET POUR L'éVALUATION DE LA CROISSANCE DE LA SÛRETé DE FONCTIONNEMENT

Soutenue le 13 Janvier 1992 devant le Jury composé de:

M.	J.C.	LAPRIE	Président
M.	B.	BARRE	Examinateur
M.	P.	BESCOND	Examinateur
M.	A.	COSTES	Examinateur
Mme	K.	KANOUN	Rapporteur
M.	Y.	LEVENDEL	Examinateur
M.	S.	NATKIN	Rapporteur

Rapport de Recherche LAAS n° 92.002

Cette thèse a été préparée au LAAS- CNRS , 7, avenue du Colonel Roche, 31077 TOULOUSE CEDEX

A mes parents
A tous les miens

AVANT-PROPOS

Les travaux présentés dans ce mémoire ont été effectués au Laboratoire d'Automatique et d'Analyse des Systèmes du Centre National de la Recherche Scientifique (LAAS-CNRS), à Toulouse. Je tiens à exprimer ma profonde gratitude à Monsieur le Professeur Alain COSTES, Directeur du LAAS, pour la confiance qu'il m'a accordée en m'accueillant au sein de ce laboratoire.

Je tiens à remercier tout particulièrement Monsieur Jean-Claude LAPRIE, Directeur de Recherche au CNRS, responsable du groupe "Tolérance aux fautes et Sûreté de Fonctionnement informatique" (TSF) du LAAS, pour m'avoir accueilli dans ce groupe et dirigé mes travaux. Ses conseils, ses critiques, ses encouragements et sa large compétence m'ont permis de mener cette thèse à son terme. Il m'a offert l'environnement idéal pour m'initier à la recherche et me confronter à la communauté scientifique tant nationale qu'internationale de très haut niveau. Son dévouement pour la recherche, son dynamisme et sa recherche de la perfection ont été pour moi un grand stimulateur pour continuer dans cette voie

Je tiens à exprimer ma sincère reconnaissance à Madame Karama KANOUN, Chargée de Recherche au CNRS, qui a dirigé mes travaux. Je lui exprime ma profonde gratitude pour son soutien permanent, l'efficacité de son aide, la qualité de ses conseils et pour la grande disponibilité dont elle a toujours fait preuve à mon égard. Ses critiques et nos longues discussions ont grandement contribué à l'amélioration de la qualité de ce mémoire, qu'elle trouve ici l'expression de mes vifs remerciements.

Je tiens à leur exprimer ma profonde gratitude en leur dédiant ce mémoire, concrétisation de notre collaboration qui fut toujours agréable et très enrichissante.

Je suis très reconnaissant envers Monsieur Alain COSTES qui, en tant que membre du groupe TSF, m'a toujours encouragé durant les trois ans que j'ai passés au LAAS. J'ai pu apprécier durant mon séjour au laboratoire son dynamisme, sa compétence scientifique, sa gentillesse et sa grande modestie. Je tiens également à le remercier profondément pour ses patientes lectures de toutes les versions de ce mémoire, pour ses critiques constructives ainsi que pour la sympathie qu'il m'a toujours témoignée. Il a su me donner l'exemple de la rigueur et de la ponctualité.

Je tiens à exprimer ma profonde gratitude à Monsieur Jean-Claude LAPRIE, Directeur de Recherche au CNRS, pour l'honneur qu'il me fait en présidant le Jury de thèse, ainsi qu'à :

- Monsieur Bertrand BARRE, Directeur des Affaires Industrielles de Technicatome, Paris,
- Monsieur Pierre BESCOND, Directeur Central de la Qualité au CNES,
- Monsieur Alain COSTES, Professeur à l'Institut National Polytechnique de Toulouse, Directeur du LAAS-CNRS,
- Madame Karama KANOUN, Chargée de Recherche au LAAS-CNRS,
- Monsieur Ytzhak LEVENDEL, Responsable du Département "Testing Process and Technologies" d'AT&T Bell Labs à Naperville, Illinois, USA,
- Monsieur Stéphane NATKIN, Professeur au Centre National des Arts et Métiers, CNAM-CEDRIC, PARIS,

pour l'honneur qu'ils me font en participant à ce Jury, et particulièrement à Madame KANOUN et Monsieur NATKIN qui ont accepté la charge d'être rapporteurs.

Certains aspects des travaux théoriques présentés dans ce mémoire, n'auraient pu être mis en pratique ni étayés sans l'aide du CNET-LANNION et de la compagnie brésilienne des télécommunications TELEBRAS CPqD. Je tiens à remercier les membres de ces organismes pour leur coopération.

Enfin, je ne saurais commencer ce mémoire sans remercier tous ceux qui, par leur compétence, leur gentillesse et leurs conseils judicieux, m'ont apporté une aide généreuse.

Tout d'abord, je tiens à témoigner ma sincère reconnaissance à Christian BEOUNES pour sa précieuse et non moins sympathique collaboration. Il s'est intéressé à mon travail, l'a suivi et m'a judicieusement éclairé à de maintes reprises.

Je remercie aussi vivement, tous les autres membres de l'équipe TSF, permanents et doctorants :

. Pascale Thévenod et Hélène Waeselynck, qui ont su, aux moments difficiles, troquer leur statut de collègues de bureau en celui de "supporters", pour me prodiguer, sans compter, leurs encouragements et leur bonne humeur.
. Jean Arlat et Yves Crouzet, pour leur disponibilité, la fiabilité de leurs conseils, et par dessus tout, pour leur grande sympathie.
. Dimitri Avreski, Jean Charles Fabre, Yves Deswarte et David Powell, pour leur soutien amical et leurs conseils pertinents.
. Joëlle Penavayre, pour sa gentillesse et sa grande disponibilité.
. Tous les doctorants du groupe TSF dont la compagnie chaleureuse et amicale m'a apporté le tonus et l'énergie indispensable pour venir à bout des obstacles qui jalonnaient mon "parcours du combattant". Je citerai : Laurent Blain, Eric Jenn, Eliane Martins, et Hélène Waeselynck, et, avec une mention spéciale, Marc Dacier, Christine Mazuet et Gilles Trouessin, pour leur lecture attentive de ce mémoire.
. Sylvain Metge, actuellement à CEP-Systèmes, pour le travail réalisé en commun lors de son séjour au LAAS.

Mes remerciements s'adressent également à tous les membres des services "Reception-Standard", "Documentation-Edition" et "Direction-Gestion" qui m'ont permis d'effectuer mon travail de façon efficace et qui ont facilité la réalisation de ce mémoire.

J'adresse aussi une pensée particulière à mes parents ainsi qu'à tous mes amis proches pour leur soutien moral et leur présence continue à mes côtés : Denis Carbonne, Gilles Cohenca, Atef Fertani, Martine Hyounet, Hamid Laamouri, Gilles Mascaras, Josine Mansour, Ridha Touzi et Thierry Trémas.

Je ne peux finir cet avant propos, sans remercier MM. Markov, Petri et Macintosh dont les contributions ont fait aboutir bien des thèses, dont celle-ci…

INTRODUCTION GÉNÉRALE

La modernisation de notre société durant les dernières décennies a conduit à une intégration massive des techniques informatiques dans la plupart des domaines de la vie courante. Les systèmes informatiques sont devenus des outils indispensables dans tous les domaines d'application civils et militaires : dans les systèmes de gestion d'information, dans les systèmes de transport, les systèmes de communication, etc….. Compte tenu de la dépendance croissante de la société actuelle des systèmes informatiques, la sûreté de fonctionnement de ces systèmes est devenue un enjeux important qui concerne tant les concepteurs, les réalisateurs et les fournisseurs que les utilisateurs des services délivrés par ces systèmes.

La défaillance d'un système informatique peut avoir des impacts catastrophiques sur son environnement par des conséquences de nature humaine ou bien économiques. Par exemple, les défaillances des systèmes informatiques constituent actuellement, la première source de sinistres industriels tels que perçus par les compagnies d'assurance, et leur coût excède, en France, les bénéfices de l'ensemble de l'industrie informatique.

Un système informatique défaille à cause de la présence de fautes aussi bien dans le matériel que dans le logiciel. L'introduction massive du logiciel dans les systèmes informatiques s'est accompagnée d'une augmentation de la complexité de ces systèmes et, par suite, des coûts du développement et de la maintenance.

De ce fait et compte tenu des conséquences catastrophiques que peuvent engendrer les fautes activées dans le logiciel, il est indispensable de disposer de techniques et de méthodes permettant d'obtenir et de fournir des systèmes sûrs de fonctionnement afin de limiter la présence des fautes et surtout leurs impacts sur le service délivré aux utilisateurs.

L'utilisation de techniques évoluées de développement, de validation et de maintenance des systèmes informatiques permet de limiter la présence et les conséquences des fautes introduites dans ces systèmes. Cependant, à cause de la complexité des applications mises en œuvre par ces systèmes et de l'imperfection de toute activité humaine, de telles techniques ne peuvent conduire à un système qui soit exempt de fautes. Il est nécessaire donc de définir des mesures et des méthodes permettant :

- d'évaluer l'aptitude du système à délivrer des services conformes aux spécifications,
- d'aider les concepteurs à suivre l'évolution de la sûreté de fonctionnement du système au cours de sa validation et pendant sa vie opérationnelle.

Ceci fait l'objet des études portant sur la modélisation probabiliste de la sûreté de fonctionnement des systèmes informatiques pour la **prévision de fautes**. Les deux mesures principales qui sont considérées pour évaluer le comportement des systèmes sont la fiabilité et la disponibilité.

La modélisation de la sûreté de fonctionnement des systèmes informatiques a donné naissance à de nombreux travaux dont une large proportion a porté sur l'étude du comportement du matériel. Les travaux effectuées sur le matériel ont été focalisés sur le développement de

modèles de comportement d'un système en tenant compte de sa structure. Ces modèles permettent d'étudier la sûreté de fonctionnement d'un système au cours de sa vie opérationnelle en fonction de celle de ses différents composants en considérant une approche **"boîte blanche"**. Cependant, à quelques rares exceptions, les modèles développés considèrent les fautes physiques comme étant la source principale de défaillance d'un système et ne tiennent pas compte des fautes de conception.

Les études de la sûreté de fonctionnement du logiciel pour la prévision de fautes sont plus récentes et portent essentiellement sur la modélisation probabiliste, selon une approche **"boîte noire"**, du phénomène de **croissance de fiabilité** en mettant l'accent plus particulièrement sur les phases de développement et de validation : la croissance de fiabilité du logiciel résulte de l'élimination progressive des fautes de conception du logiciel au cours de son cycle de vie. Cependant, en dépit de l'importance majeure de la disponibilité de service pour certains types de logiciels tels que les systèmes de télécommunication et les logiciels de messagerie électronique par exemple, cette mesure n'a pas été considérée, à notre connaissance, dans les modèles de croissance de fiabilité.

Ainsi, la modélisation de la sûreté de fonctionnement du matériel et celle du logiciel ont suivi des orientations différentes. Or, tant utilisateurs que concepteurs des systèmes informatiques sont intéressés par des évaluations qui concernent la fiabilité et la disponibilité du système global matériel et logiciel en considérant aussi bien les fautes physiques que les fautes de conception. De telles évaluations ne sont pas courantes dans le domaine, l'un des objectifs des travaux que nous présentons dans ce mémoire est d'unifier la modélisation de la sûreté de fonctionnement du matériel et du logiciel en considérant les deux approches possibles de l'étude d'un système : l'approche "boîte noire" et l'approche "boîte blanche". Plus précisément, nous présentons un modèle, le modèle hyperexponentiel, et une approche de modélisation basée sur ce modèle, permettant d'évaluer la croissance de fiabilité et la croissance de disponibilité de systèmes matériels et logiciels en considérant à la fois les fautes physiques et les fautes de conception.

Par ailleurs, l'évaluation de la croissance de fiabilité du logiciel a donné naissance à une multitude de modèles, une quarantaine environ, dont la majorité sont des modèles en **temps continu** qui caractérisent l'évolution dans le temps du comportement du logiciel dans un **environnement donné**. Ces modèles ne sont pas bien adaptés pour caractériser la fiabilité de certains systèmes tels que par exemple les systèmes transactionnels, les logiciels de commande de missile, etc.... Pour ces systèmes, il est plus significatif de représenter l'évolution de la fiabilité en **temps discret,** c'est-à-dire en fonction du nombre d'exécutions effectuées.

De plus, un logiciel est souvent destiné à être utilisé dans des environnements qui diffèrent par la nature des services sollicités par les utilisateurs et par les caractéristiques matérielles des systèmes sur lesquels le logiciel est mis en œuvre. Or, il est inconcevable de valider un logiciel par rapport à tous les environnements dans lequel il est utilisé. Le problème qui se pose est "comment estimer les mesures de sûreté de fonctionnement d'un logiciel utilisé dans un environnement donné à partir des mesures évaluées dans un autre environnement ?"

Les premières tentatives pour résoudre ce problème ont été basées sur l'utilisation des modèles proportionnels de Cox. Ces modèles sont obtenus à partir des modèles classiques de

croissance de fiabilité en introduisant des variables supplémentaires dites "variables explicatives" qui caractérisent l'environnement d'utilisation du logiciel. L'approche que nous proposons dans le cadre de ce mémoire est différente ; elle repose sur l'idée qui consiste à distinguer les deux sources suivantes de la variation dans le temps des mesures de fiabilité du logiciel :

. la première source est liée à l'évolution de la probabilité de défaillance à l'exécution du logiciel suite à l'élimination progressive des fautes de conception qui sont activées au fur et à mesure de l'exécution du logiciel,

. la deuxième source est liée à la variation du taux d'exécution du logiciel quand il est utilisé dans des environnements qui diffèrent par la fréquence de sollicitation des services délivrés ou bien par les caractéristiques matérielles des systèmes sur lesquels le logiciel est mis en œuvre (des machines de puissances différentes par exemple) .

Pour modéliser l'évolution de la probabilité de défaillance à l'exécution en fonction du nombre d'exécutions du logiciel et des corrections introduites, nous définissons un nouveau modèle de croissance de fiabilité : **le modèle hyperexponentiel en temps discret**. En tenant compte du taux d'exécution du logiciel dans un environnement donné, nous déduisons ensuite les mesures de fiabilité qui caractérisent l'évolution du comportement du logiciel dans le temps, c'est à dire tel qu'il est perçu par ses utilisateurs. De cette façon, dans le cas où un même logiciel est installé dans deux environnements différents, nous pouvons évaluer son comportement tel qu'il est perçu par ses utilisateurs dans chacun des environnements considérés.

Dans le cadre de cette étude, nous appliquons l'approche proposée dans le cas où un logiciel est considéré comme "une boîte noire" et également dans le cas d'un logiciel multi-composant (approche "boîte blanche").

Les travaux que nous présentons dans ce mémoire s'inscrivent dans le cadre de la continuité des études effectuées par le groupe "Tolérance aux fautes et Sûreté de Fonctionnement informatique" du LAAS-CNRS, depuis plus de dix ans sur la croissance de la sûreté de fonctionnement. Nos travaux s'articulent autour de deux modèles de croissance de fiabilité : le modèle hyperexponentiel en temps continu et le modèle hyperexponentiel en temps discret, en considérant deux approches de modélisation de la croissance de sûreté de fonctionnement des systèmes : une approche "boîte noire" et une approche "boîte blanche".

Le premier modèle a été introduit dans [Laprie 84-b, Kanoun 85] pour modéliser la croissance de sûreté de fonctionnement d'un système informatique selon une approche "boîte noire". Dans ce mémoire, nous étudions de façon plus approfondie les caractéristiques de ce modèle et nous définissons une approche de modélisation "boîte blanche" basée sur ce modèle pour évaluer la croissance de sûreté de fonctionnement de systèmes multi-composant. Les fondements de cette approche de modélisation ont été publiés dans [Laprie 90-a, Laprie 91-a].

Le second modèle n'a pas encore fait l'objet de publication ; les principales motivations, brièvement évoquées précédemment, ainsi que le développement et l'application de ce modèle à des systèmes réels font l'objet d'une présentation détaillée dans le cadre de ce mémoire .

Ce mémoire est structuré en deux parties : la première partie est consacrée à la caractérisation et à la modélisation selon une approche "boîte noire" de la croissance de sûreté de fonctionnement d'un système informatique en mettant l'accent plus particulièrement sur le

logiciel, et la deuxième partie porte sur la modélisation selon une approche "boîte blanche" de la croissance de sûreté de fonctionnement de systèmes multi-composant.

La première partie est composée de trois chapitres :

Dans le premier chapitre, nous introduisons les concepts de base et les éléments qui sont nécessaires à l'établissement et à l'application à des données expérimentales du modèle hyperexponentiel en temps continu et en temps discret. Nous introduisons d'abord la terminologie et les concepts de base de la sûreté de fonctionnement. Nous présentons ensuite les différents facteurs qui agissent sur la sûreté de fonctionnement du logiciel à travers une modélisation conceptuelle de son comportement au cours de son cycle de vie, en mettant l'accent plus particulièrement sur son profil d'utilisation. Nous présentons ensuite les principales mesures de la fiabilité du logiciel en temps continu et en temps discret. Les modèles existants de croissance de fiabilité sont ensuite présentés et analysés afin de situer, par rapport à ces modèles, le modèle hyperexponentiel en temps continu et le modèle hyperexponentiel en temps discret. L'application des modèles de croissance de fiabilité à des données expérimentales est discutée à la fin de ce chapitre où nous présentons la méthode que nous utilisons pour analyser la variation de la croissance de fiabilité du logiciel en fonction des données collectées, et appliquer les modèles de croissance de fiabilité pour évaluer les mesures de la sûreté de fonctionnement.

Le deuxième chapitre est consacré au modèle hyperexponentiel en temps continu. Nous présentons d'abord les caractéristiques du modèle et nous établissons les expressions des mesures de fiabilité associées. Nous nous intéressons ensuite à la modélisation de la disponibilité en tenant compte du phénomène de croissance de fiabilité. Nous proposons ensuite une généralisation du modèle pour augmenter sa puissance de modélisation et nous étudions les propriétés du modèle généralisé ainsi obtenu. La dernière partie de ce chapitre est consacrée à l'application du modèle à deux logiciels de télécommunication pour illustrer son aptitude à évaluer la croissance de sûreté de fonctionnement des systèmes.

Le troisième chapitre est consacré à la présentation du modèle hyperexponentiel en temps discret. Nous établissons d'abord les expressions des principales mesures permettant de caractériser l'évolution de la croissance de fiabilité du logiciel en fonction du nombre d'exécutions effectuées. Nous étudions ensuite le lien entre ce modèle et le modèle hyperexponentiel en temps discret et nous abordons le problème de la prise en compte des caractéristiques de l'environnement d'utilisation du logiciel dans l'évaluation de ses mesures de sûreté de fonctionnement. L'application du modèle à des données réelles est effectuée à la fin de ce chapitre.

La deuxième partie est composée de deux chapitres :

Dans le premier chapitre (chapitre IV), nous exposons une approche de modélisation basée sur le modèle hyperexponentiel en temps continu permettant d'évaluer la croissance de fiabilité et la croissance de disponibilité de systèmes multi-composant matériels et logiciels. Nous introduisons d'abord l'approche de modélisation à travers deux exemples simples. Nous définissons ensuite une approche générale basée sur les réseaux de Petri stochastiques généralisés. L'approche ainsi définie est ensuite appliquée à deux systèmes plus complexes : le

premier est un système duplex intégrant des mécanismes de détection et de recouvrement d'erreurs, le deuxième est un logiciel tolérant aux fautes constitué d'un bloc de recouvrement.

Enfin, dans le deuxième chapitre (chapitre V), nous proposons une approche de modélisation basée sur le modèle hyperexponentiel en temps discret permettant d'une part, de modéliser la croissance de fiabilité d'un logiciel multi-composant en fonction du nombre d'exécutions effectuées et d'autre part de tenir compte des taux d'exécution des différents composants du logiciel pour évaluer l'évolution dans le temps des mesures de fiabilité du logiciel en tenant compte des caractéristiques de l'environnement dans lequel il est utilisé.

PREMIÈRE PARTIE

MODÈLE HYPEREXPONENTIEL EN TEMPS CONTINU ET EN TEMPS DISCRET

Cette première partie a pour objectifs la présentation et l'application du modèle hyperexponentiel de croissance de fiabilité pour l'évaluation de la sûreté de fonctionnement du logiciel. Deux types de représentation du comportement du logiciel sont considérés : une représentation en temps continu où les mesures de la sûreté de fonctionnement du logiciel sont exprimées en fonction du temps, et une représentation en temps discret où les mesures évaluées sont exprimées en fonction du nombre d'exécutions du logiciel.

Cette partie est constituée de trois chapitres.

Dans le premier chapitre, nous introduisons et rappelons les concepts de base qui sont nécessaires à l'établissement du modèle hyperexponentiel en temps continu et en temps discret. Nous analysons d'abord les différents facteurs qui agissent sur le comportement du logiciel en mettant l'accent plus particulièrement sur l'influence de la variation du profil d'utilisation du logiciel et de son domaine de défaillance. Nous présentons ensuite les définitions des différentes mesures qui seront considérées par la suite pour évaluer la sûreté de fonctionnement du logiciel. Nous présentons et analysons enfin les modèles de croissance de fiabilité existants afin de situer, par rapport à ces modèles, le modèle hyperexponentiel sous ses deux aspects, temps continu et temps discret. La dernière partie de ce chapitre est consacrée à la présentation d'une méthode d'analyse et d'évaluation de la croissance de fiabilité du logiciel, méthode où le modèle hyperexponentiel est utilisé.

Le deuxième chapitre, de cette partie est consacré à la présentation des propriétés du modèle hyperexponentiel en temps continu et à l'établissement des expressions des mesures de fiabilité et de disponibilité associées. Ce modèle est ensuite appliqué à deux logiciels de télécommunication afin d'illustrer son aptitude à évaluer les mesures de sûreté de fonctionnement d'un système.

Le troisième chapitre, est consacré à la présentation du modèle hyperexponentiel en temps discret. Nous établissons d'abord les expressions des mesures de fiabilité associées et nous étudions ensuite le lien entre ce modèle et le modèle hyperexponentiel en temps continu. Nous appliquons finalement le modèle à deux sous-ensembles de données collectées sur des logiciels en phase de validation pour illustrer son aptitude à caractériser et évaluer le comportement de logiciels réels en fonction du nombre d'exécutions effectuées.

CHAPITRE I

CADRE GÉNÉRAL ET NOTIONS DE BASES

Ce chapitre présente brièvement dans une première partie la terminologie et les concepts de base de la sûreté de fonctionnement en mettant l'accent sur ceux qui sont utilisés dans ce mémoire. Nous présentons ensuite les différents facteurs qui agissent sur la sûreté de fonctionnement du logiciel à travers une modélisation conceptuelle du comportement du logiciel au cours de son cycle de vie. Nous donnons ensuite les raisons qui nous ont amené à considérer deux types de représentation du comportement du logiciel : en fonction du temps ou en fonction du nombre d'exécutions. Les mesures de la sûreté de fonctionnement en temps continu et en temps discret ainsi que les modèles existants de croissance de fiabilité du logiciel sont ensuite introduits et analysés. L'application des modèles de croissance du logiciel à des données expérimentales est discutée à la fin de ce chapitre où nous présentons une méthode globale d'analyse et d'évaluation de la fiabilité du logiciel qui a pour objectif d'assurer une meilleure utilisation des modèles de croissance de fiabilité pour l'évaluation et le suivi de la sûreté de fonctionnement du logiciel au cours de son cycle de vie.

I.1 CONCEPTS DE BASE DE LA SÛRETé DE FONCTIONNEMENT : RAPPELS

Nous nous limitons dans cette présentation de la terminologie liée à la sûreté de fonctionnement aux concepts de base qui sont utilisés dans ce mémoire. Une présentation plus complète des concepts de base et de la terminologie de la sûreté de fonctionnement se trouve dans [Laprie 91-c].

La **sûreté de fonctionnement** est la propriété d'un système qui permet de placer une confiance justifiée dans le service qu'il délivre.

Un **système** est une entité ayant interagi ou interféré, interagissant ou interférant, ou susceptible d'interagir ou d'interférer avec d'autres entités. Ces entités sont d'autres systèmes, humains ou physiques, qui ont constitué, constituent ou constitueront l'**environnement** du système considéré.

Le **service** délivré par un système est son comportement tel que perçu par son, ou ses, utilisateur(s) ; un **utilisateur** est une partie de l'environnement qui interagit avec ce dernier.

Une **défaillance** du système survient lorsque le service délivré n'est plus conforme à la spécification, la **spécification** étant une description agréée de la fonction ou du service attendu du système. Une **erreur** est la partie de l'état du système qui est susceptible d'entraîner une défaillance. La cause adjugée ou supposée d'une erreur est une **faute**.

Les fautes et leurs sources sont extrêmement diverses. Nous distinguerons les deux types de fautes suivants :

- les **fautes physiques**, qui sont dues à des phénomènes physiques adverses qui peuvent être soit internes au système soit liés à son environnement,
- les **fautes de conception** qui résultent d'imperfections humaines commises soit au cours du développement du système soit au cours de modifications ultérieures.

Le développement d'un système sûr de fonctionnement passe par l'utilisation combinée d'un ensemble de méthodes qui peuvent être classées en :

- **prévention de fautes** : comment empêcher l'occurrence ou l'introduction de fautes,
- **tolérance aux fautes** : comment fournir un service conforme aux spécifications en dépit des fautes,
- **élimination de fautes** : comment réduire la présence (nombre, sévérité) des fautes,
- **prévision de fautes** : comment estimer la présence, la création et les conséquences des fautes.

Prévention des fautes et tolérance aux fautes peuvent être vues comme constituant l'**obtention** de la sûreté de fonctionnement : comment *fournir* au système l'aptitude à fournir un service conforme à la spécification ; élimination des fautes et prévision des fautes peuvent être vues comme constituant la **validation** de la sûreté de fonctionnement des systèmes : comment *avoir confiance* dans l'aptitude du système à délivrer un service conforme à la spécification.

Le comportement d'un système est perçu par son, ou ses utilisateurs comme une alternance entre deux états du service délivré par rapport au service spécifié :

- **service correct**, où le service délivré est conforme à la spécification,
- **service incorrect**, où le service délivré n'est pas conforme à la spécification.

Les événements qui conduisent aux transitions entre les deux états du service sont respectivement la défaillance, transition de service correct à service incorrect, et la **restauration**, transition de service incorrect à service correct.

L'évaluation du comportement d'un système par rapport à l'alternance service correct et service incorrect conduit aux deux principales mesures de sûreté de fonctionnement[1]:

- la **fiabilité** qui mesure la délivrance continue d'un service correct, ou, de façon équivalente, du temps jusqu'à défaillance,
- la **disponibilité** qui mesure la délivrance d'un service correct par rapport à l'alternance service correct-service incorrect.

---

[1] En plus de ces deux mesures de base de la sûreté de fonctionnement, il existe d'autres mesures qui caractérisent d'autres attributs de la sûreté de fonctionnement des systèmes : maintenabilité, performabilité, sécurité, etc…

Dans les définitions précédentes, un système a été implicitement considéré comme un tout selon une vue "boîte noire". Une définition d'un système d'un point de vue structurel ("boîte blanche" ou "boîte de verre") est la suivante : un système est un ensemble de composants interconnectés en vue d'interagir ; un **composant** est un autre système, etc. Cette définition est récursive ; la décomposition s'arrête lorsqu'un système est considéré comme étant un système atomique : aucune décomposition ultérieure n'est envisagée, soit par nature soit parce que dénuée de tout intérêt. Selon cette définition, un système informatique peut être vu comme un ensemble de composants matériels et logiciels interagissant entre eux ; ces derniers étant eux mêmes considérés comme des systèmes.

L'évaluation des mesures de sûreté de fonctionnement d'un système peut être effectuée en étudiant son comportement selon une vue "boîte noire" ou bien selon une vue "boîte blanche". Dans le cadre de ce mémoire, nous considérons les deux approches en mettant l'accent plus particulièrement sur le logiciel et en nous intéressant à la modélisation de la sûreté de fonctionnement pour la prévision de fautes en considérant essentiellement les fautes de conception.

I.2 ANALYSE DU COMPORTEMENT DU LOGICIEL

D'une manière conceptuelle, le comportement d'un logiciel peut être vu comme la transformation par un programme **P** d'un espace d'entrées **E** en un espace de sorties **S**.

Une **exécution** du logiciel consiste à sélectionner des données de l'espace des entrées et se traduit par une trajectoire dans l'espace E. Chaque entrée de l'espace E est transformée en une seule sortie de l'espace S si on fait l'hypothèse que les sorties qui sont utilisées pour une prochaine exécution du logiciel font partie à la fois de E et S (c'est le cas par exemple des logiciels de contrôle et de commande de processus industriel).

Au cours de son cycle de vie, le logiciel n'est pas exécuté en continu ; il est sollicité par ses utilisateurs selon un profil d'utilisation caractéristique de l'environnement dans lequel il est exécuté. Le **profil d'utilisation** caractérise la répartition dans le temps des instants de sollicitation du logiciel et détermine les entrées fournies au programme parmi toutes les entrées possibles. Du fait de la présence de fautes de conception internes dans le logiciel, l'exécution du programme P peut conduire à des sorties qui ne sont pas conformes à la spécification du logiciel.

Soient :

S_e : le sous ensemble des sorties erronées défini par les spécifications de P,

E_d : le sous-ensemble des entrées qui est tel que la transformation de E_d par P est S_e ; E_d est appelé **domaine de défaillance** du logiciel.

Le domaine de défaillance peut être décomposé en deux sous-ensembles [Laprie 89] : le premier regroupe les entrées activant les fautes du logiciel et le deuxième est constitué des entrées erronées qui sont implicitement ou explicitement définies par les spécifications.

Le domaine de défaillance du logiciel évolue au cours de son cycle de vie. Cette variation est liée d'une part, à la correction des fautes activées au fur et à mesure de l'exécution, et d'autre part, à la modification des spécifications pour mieux satisfaire les besoins des utilisateurs.

Le profil d'utilisation du logiciel et son domaine de défaillance sont les paramètres les plus importants qui caractérisent et agissent sur sa sûreté de fonctionnement. L'évolution de l'un ou l'autre de ces deux paramètres conduit à la variation du comportement du logiciel tel qu'il est perçu par ses utilisateurs. De ce fait, nous analyserons dans la suite l'influence de chacun de ces deux paramètres sur le comportement du logiciel. Nous considérerons les deux cas suivants :

- le profil d'utilisation du logiciel reste homogène au cours de son cycle de vie,
- le profil d'utilisation du logiciel varie au cours de son cycle de vie.

I.2.1 Profil d'utilisation homogène

L'analyse du comportement du logiciel vis-à-vis de l'occurrence de défaillances peut être effectuée en considérant les deux cas suivants :

- le comportement du logiciel jusqu'à occurrence d'une défaillance,
- le comportement du logiciel en tenant compte des reprises d'exécution après défaillance.

I.2.1.1 Occurrence d'une défaillance

L'exécution du logiciel s'effectue conformément aux spécifications tant que les entrées sélectionnées par les utilisateurs sont en dehors du domaine de défaillance E_d. Quand la trajectoire dans l'espace des entrées E intercepte le sous-ensemble E_d, le logiciel défaille.

Conceptuellement, on peut modéliser l'occurrence d'une défaillance du logiciel par le modèle présenté sur la figure I.1. Sur cette figure, toutes les entrées qui conduisent à la défaillance du logiciel sont regroupées dans le domaine de défaillance E_d. Si on fait l'hypothèse que chaque défaillance du logiciel est due à l'activation d'une seule faute, alors on peut partitionner E_d en plusieurs sous-domaines disjoints tels que chaque sous-domaine regroupe les entrées entraînant la défaillance correspondante[2] (la partie grisée du domaine de défaillance E_d correspond aux entrées entraînant la défaillance considérée).

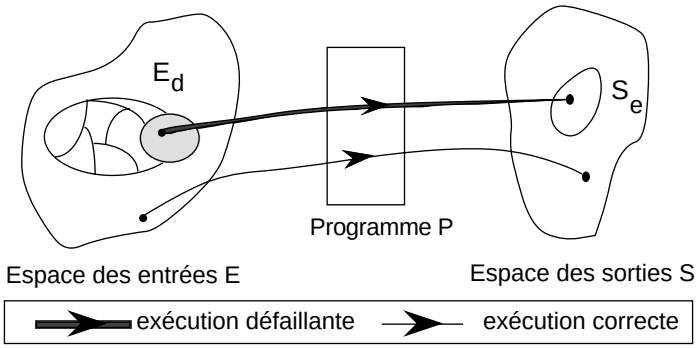


Figure I.1 : Mécanisme d'occurrence d'une défaillance

Les données de l'espace des entrées du logiciel, et en particulier celles qui font partie du domaine de défaillance, sont sélectionnées selon un mécanisme aléatoire. En effet, d'une part, le choix des entrées est conditionné par le type de services sollicité par les utilisateurs et dépend ainsi intrinsèquement du profil d'utilisation du logiciel, et d'autre part, étant donné que la

---

2 En réalité, les sous-domaines de défaillance ne sont pas disjoints car une entrée appartenant au domaine de défaillance du logiciel peut conduire à l'activation de plusieurs fautes.

sollicitation du logiciel pour un service particulier se traduit par une trajectoire dans l'espace des entrées, une telle trajectoire ne peut être connue de façon déterministe car elle peut dépendre de l'état du système avant sa sollicitation. Cette première source d'incertitude sur le comportement du logiciel est appelée "**incertitude sur les entrées**" [Littlewood 80].

L'incertitude sur le comportement du logiciel, qui est liée au mécanisme aléatoire du choix des entrées, se traduit par une distribution stochastique du temps jusqu'à occurrence d'une défaillance. La spécification d'une telle distribution permet de caractériser le comportement du logiciel jusqu'à occurrence d'une défaillance et donc sa fiabilité.

I.2.1.2 Reprise de l'exécution après défaillance

Quand une défaillance du logiciel survient, il est nécessaire de le relancer pour reprendre l'exécution. La restauration du service délivré par le logiciel peut s'effectuer de deux façons :

. soit par une simple relance du logiciel, sans aucune modification, avec des entrées situées en dehors du domaine de défaillance (réinitialisation du logiciel),

. soit par la relance du logiciel après correction de la (ou des) faute(s) ayant entraîné la défaillance.

Dans le cas où la restauration de service est effectuée par une simple relance du logiciel, celui-ci ne subit aucune transformation et son domaine de défaillance n'est pas modifié. Le comportement du logiciel vis-à-vis de l'occurrence de défaillance est alors le même avant et après la restauration de service. Le logiciel est dit en **fiabilité stabilisée**.

Considérons maintenant le cas où l'exécution du logiciel est reprise après la correction de (ou des) faute(s) qui ont entraîné la défaillance. La correction des fautes activées sur le logiciel conduit à la modification du programme P et du domaine de défaillance E_d. Si la correction est parfaite, alors les entrées qui appartenaient au sous-domaine de défaillance correspondant aux fautes corrigées ne font plus partie du nouveau domaine de défaillance associé au logiciel corrigé. Ceci est illustré sur le modèle de la figure I.2 où , comparé à la figure I.1, P' diffère de P par les corrections introduites et le domaine de défaillance est tel que :

$$E_d^{'} \subset E_d \qquad \text{(I.1)}$$

La partie grisée du domaine E_d de la figure I.1 ne fait plus partie du domaine de défaillance du logiciel après l'introduction des corrections (figure I.2).

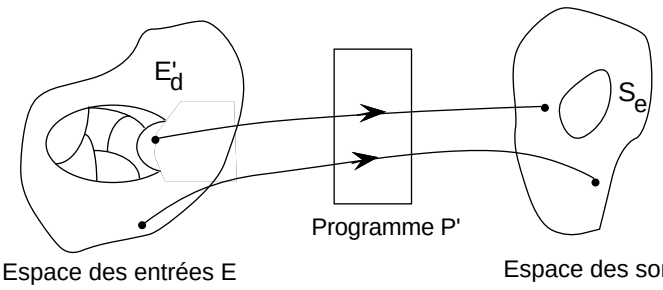


Figure I.2 : Conséquences des corrections de faute sur le domaine de défaillance

La variation du domaine de défaillance conduit alors à la variation du comportement du logiciel jusqu'à occurrence d'une nouvelle défaillance. Si on suppose que le logiciel est exécuté à nouveau selon le même profil d'utilisation, on peut s'attendre à ce que le temps jusqu'à occurrence d'une défaillance, après reprise de l'exécution, devienne de plus en plus long au fur et

à mesure de l'élimination des fautes. Ceci se traduit en d'autres termes par une fréquence d'occurrence de défaillances qui devient de plus en plus faible exprimant ainsi l'amélioration progressive de l'aptitude du logiciel à délivrer des services conformes aux spécifications. C'est le phénomène de **croissance de fiabilité**.

Néanmoins, la relation I.1 ne peut être garantie avec certitude. La correction des fautes du logiciel n'est pas toujours parfaite et peut conduire à l'introduction de nouvelles fautes dans le logiciel et à l'augmentation de la taille du domaine de défaillance. En voie de conséquence, la modification du logiciel dans ce cas peut se traduire par une diminution du temps jusqu'à occurrence d'une défaillance, comparé avec celui observé avant la reprise de l'exécution. Un tel comportement est synonyme d'une **décroissance de la fiabilité** du logiciel.

La variation du comportement du logiciel après l'élimination des fautes est intrinsèquement liée à la nature des fautes éliminées et des fautes pouvant être introduites dans le logiciel suite à des corrections imparfaites. En effet, les études effectuées sur des logiciels réels ont mis en évidence deux phénomènes importants :

- les fautes internes du logiciel ne sont pas équivalentes dans le sens où elles sont caractérisées par des taux d'activation différents [Adams 80, Finelli 91] ; l'élimination des fautes à taux d'activation élevés (respectivement faibles) conduit généralement à une variation significative (respectivement négligeable) des temps séparant l'occurrence de défaillances et par suite de la sûreté de fonctionnement du logiciel.

- les fautes du logiciel peuvent être corrélées. L'élimination d'une faute peut conduire à l'activation d'autres fautes qui étaient masquées par la faute éliminée et entraîner par la suite une augmentation du domaine de défaillance du logiciel [Ohba 84, Bishop 89].

En tenant compte de l'élimination progressive des fautes de conception, la vie du logiciel peut être vue comme une séquence de programmes $P^{(1)}$, $P^{(2)}$, …, $P^{(n)}$, …, et une séquence de domaines de défaillance $E_d^{(1)}$, $E_d^{(2)}$,…, $E_d^{(n)}$, où $P^{(i)}$ diffère de $P^{(i-1)}$ par les corrections qui lui ont été apportées. Compte tenu de la nature imprévisible des conséquences des corrections sur le logiciel, on voit apparaître une deuxième source d'incertitude sur son comportement : **l'incertitude sur les programmes** [Littlewood 80].

Cette dernière source d'incertitude traduit la nature aléatoire de l'évolution du comportement du logiciel jusqu'à défaillance au fur et à mesure de l'élimination des fautes. En d'autres termes, le processus de défaillance du logiciel peut être décrit par la suite de variables aléatoires T_i où T_i est l'intervalle de temps entre l'occurrence des défaillances i-1 et i. Le comportement du logiciel vis-à-vis de l'occurrence de défaillances est alors caractérisé par la spécification des lois des variables T_i et de la relation décrivant l'évolution de ces variables au fur et à mesure de l'élimination des fautes. Selon la nature des opérations effectuées après la défaillance (relance sans correction ou après correction), T_i et T_{i-1} peuvent être identiquement distribuées ou bien de lois différentes.

REMARQUE

Dans l'analyse effectuée, nous n'avons pas tenu compte de la variation du comportement du logiciel qui est liée au changement des spécifications. De façon analogue avec les modifications

liées à des corrections de fautes, le changement des spécifications du logiciel conduit à la modification du domaine de défaillance. Selon le type de modifications introduites, la variation du domaine de défaillance peut se traduire par une décroissance de fiabilité, une croissance de fiabilité ou une fiabilité stabilisée du logiciel [Kanoun 89].

I.2.1.3 Conclusion

L'analyse résumée du comportement du logiciel que nous venons d'effectuer amène à distinguer deux types de comportement du logiciel :

- le comportement du logiciel jusqu'à occurrence d'une première défaillance où n'intervient que la première source d'incertitude : l'incertitude sur les entrées,
- le comportement du logiciel durant son cycle de vie en tenant compte des reprises d'exécution après défaillance, où interviennent les deux sources d'incertitude : l'incertitude sur les entrées et l'incertitude sur les programmes.

Ces deux types de comportement seront considérés lors de la présentation des mesures de sûreté de fonctionnement et des modèles proposés dans la littérature pour l'évaluation de la croissance de fiabilité du logiciel.

I.2.2 Variation du profil d'utilisation

Au cours du cycle de vie du logiciel, son profil d'utilisation peut varier d'une phase à une autre, ou même à l'intérieur d'une même phase.

Au cours de la phase de validation, le profil d'utilisation du logiciel peut varier en fonction :

- de la nature de tests effectués (tests fonctionnels, tests de charge, etc.…),
- de la variation de l'effort de test appliqué,
- des caractéristiques des systèmes sur lesquels il est exécuté, etc….

Au cours de la vie opérationnelle, le logiciel est souvent utilisé dans plusieurs environnements qui diffèrent par la nature et la fréquence des services sollicités. Par exemple, les systèmes de télécommunications sont utilisés aussi bien dans des milieux ruraux où la majorité des appels effectués sont des appels téléphoniques, que dans des villes où les communications établies sont plus diversifiées : transmissions de données, communications internationales, etc…

La variation du profil d'utilisation du logiciel peut se traduire par une variation de la distribution du choix des données dans l'espace des entrées ou bien par une variation de la fréquence de sollicitation du logiciel.

Nous montrons ci-après, sur deux exemples très simplifiés, l'influence de ces deux sources de variation du profil d'utilisation du logiciel sur son comportement tel qu'il est perçu par ses utilisateurs.

I.2.2.1 Variation de la distribution du choix des entrées

Considérons deux utilisateurs A et B d'un même logiciel qui l'exécutent en moyenne à la même fréquence selon deux profils d'utilisation qui différent par la distribution du choix des données de l'espace des entrées du logiciel.

Considérons la partition de l'espace des entrées E proposée sur la figure I.3 où E_d est le domaine de défaillance du logiciel et E_1 (respectivement E_2) correspond au sous-ensemble des entrées qui sont le plus souvent sollicitées par l'utilisateur A (respectivement B).

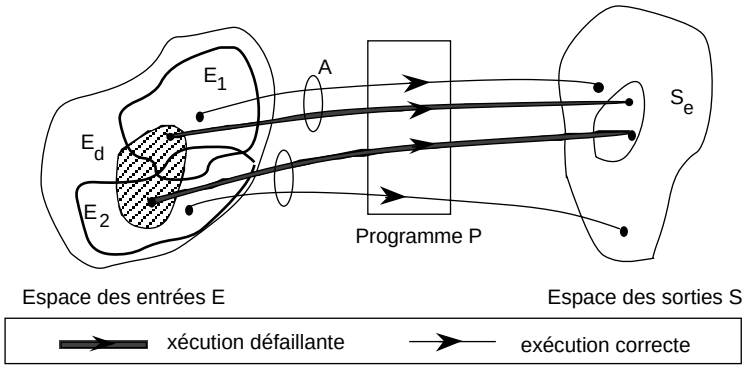


Figure I.3 : Trajectoires dans l'espace des entrées

Etudions le comportement jusqu'à défaillance du logiciel tel qu'il est perçu par les deux utilisateurs.

Nous avons vu précédemment que le comportement du logiciel jusqu'à défaillance dépend essentiellement de la nature de son domaine de défaillance. Compte tenu des hypothèses établies, tout se passe comme si on considère deux logiciels pour lesquels les domaines de défaillance sont différents. Les distributions du temps jusqu'à défaillance seront donc différentes pour les deux profils d'utilisation A et B considérés. En particulier, si l'intersection du sous-ensemble E_1 avec le domaine de défaillance du logiciel est vide, alors la probabilité que le logiciel défaille suite à son exécution selon le profil d'utilisation caractéristique de l'utilisateur A sera très faible. Par conséquent, la sûreté de fonctionnement du même logiciel sera perçue de façons différentes par les deux utilisateurs A et B.

Supposons maintenant que l'on utilise le logiciel durant l'intervalle $[0,t_1]$ selon le profil d'utilisation A, puis à partir de l'instant t_1 selon le profil d'utilisation B. La variation du sous-ensemble des entrées qui sont sélectionnées au cours de l'exécution du logiciel, qui est induite par le changement du profil d'utilisation, peut conduire à la variation de la distribution du temps jusqu'à défaillance du logiciel. Ainsi, on peut observer une augmentation ou une diminution des temps jusqu'à occurrence des défaillances ou en d'autres termes, une décroissance ou une croissance de fiabilité qui n'est pas nécessairement liée à la modification du domaine de défaillance suite à la correction de fautes dans le logiciel.

I.2.2.2 Variation de la fréquence de sollicitation du logiciel

Supposons maintenant que l'espace des entrées est parcouru de la même manière par les deux utilisateurs du logiciel ; seules les fréquences de sollicitation du logiciel sont différentes. Soit S_i l'instant d'occurrence de la i ème défaillance du logiciel.

Sur la figure I-4, nous présentons un exemple d'évolution dans le temps des instants d'occurrence de défaillances observées par les deux utilisateurs en supposant que A sollicite deux fois plus souvent le logiciel que B.

Pour simplifier la représentation, on suppose que les temps séparant deux sollicitations du logiciel par un même utilisateur sont égaux et on néglige les durées correspondant aux exécutions effectuées[3] ; les graduations de l'échelle des temps correspondent alors au nombre de sollicitations effectuées.

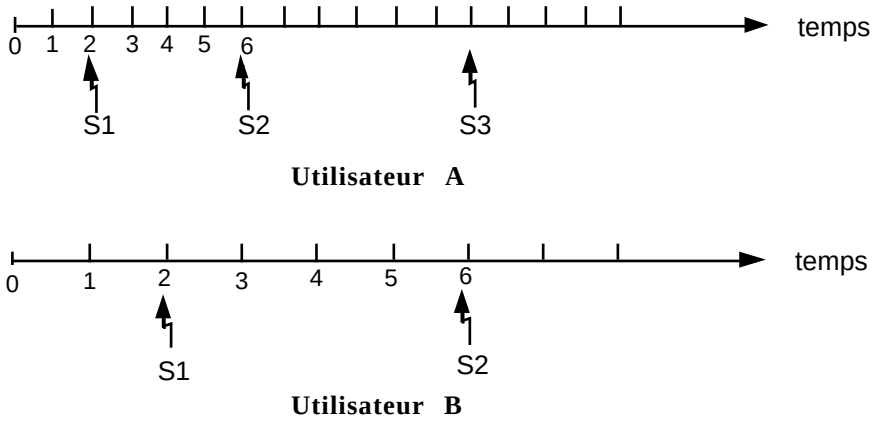


Figure I-4 : Répartition des défaillances dans le temps en fonction de la fréquence de sollicitation du logiciel

Si on considère la même référence de temps pour les deux utilisateurs du logiciel, par exemple la semaine, on peut remarquer que les intervalles de temps entre défaillances observés par A sont plus petits comparés aux intervalles de temps entre défaillances observés par B. Par conséquent, l'utilisateur B aura l'impression que la fiabilité du logiciel est meilleure, puisque les défaillances sont moins fréquentes dans le temps.

Cependant, si on considère comme référence de temps le nombre de sollicitations du logiciel, ou, de façon équivalente, le nombre d'exécutions, le comportement du logiciel sera perçu par A et B de la même manière : les intervalles de temps entre défaillances mesurés en nombre d'exécutions seront statistiquement du même ordre de grandeur. En particulier, dans le cas où le logiciel est exécuté par A et B, avec les mêmes entrées à chaque sollicitation, les intervalles de temps entre défaillances mesurés en nombre d'exécutions seront égaux.

---

[3] En réalité, les instants de sollicitation du logiciel sont répartis de façon aléatoire dans le temps, et les durées des exécutions peuvent varier selon le type des traitements effectués.

I.2.2.3 Conclusion

Les deux exemples simples que nous venons de traiter montrent que la variation du profil d'utilisation du logiciel peut se traduire par une variation de son comportement tel qu'il est perçu par ses utilisateurs.

Le deuxième exemple présenté met en évidence l'importance du choix de la référence de temps pour mesurer la sûreté de fonctionnement du logiciel. Cette remarque nous amène à distinguer deux types de représentation de la sûreté de fonctionnement du logiciel :

- une représentation en temps discret qui consiste à caractériser l'évolution du comportement du logiciel en fonction du nombre d'exécutions effectuées,
- une représentation en temps continu, qui modélise l'évolution du comportement du logiciel dans le temps. Ce type de modélisation doit intégrer aussi bien les caractéristiques intrinsèques du logiciel que les caractéristiques extrinsèques liées à son environnement d'utilisation.

I.2.3 Conclusions

L'analyse résumée du comportement du logiciel vis-à-vis de l'occurrence de défaillances et de l'élimination des fautes de conception nous amène à distinguer deux types de comportement du logiciel :

- le comportement du logiciel jusqu'à occurrence d'une défaillance où n'intervient que la première source d'incertitude : l'incertitude sur les entrées,
- le comportement du logiciel durant son cycle de vie en tenant compte des reprises d'exécution après défaillance, où interviennent les deux sources d'incertitude : l'incertitude sur les entrées et l'incertitude sur les programmes.

L'étude du processus de défaillance du logiciel a été effectuée en considérant l'évolution des temps jusqu'à défaillance au cours du cycle de vie du logiciel. Selon la nature de l'évolution des intervalles de temps jusqu'à défaillance, nous avons distingué trois types d'évolution de la fiabilité du logiciel :

- **croissance de fiabilité**, qui se traduit par une décroissance stochastique des intervalles jusqu'à défaillance du logiciel,
- **fiabilité stabilisée**, qui se traduit par des intervalles jusqu'à défaillance qui sont identiquement distribués,
- **décroissance de fiabilité**, qui se traduit par une croissance stochastique des intervalles jusqu'à défaillance.

L'étude de l'influence du profil d'utilisation du logiciel sur son comportement nous a permis de conclure que la variation du profil d'utilisation, qu'elle soit liée à la variation de la trajectoire dans l'espace des entrées ou bien à la variation de la fréquence de sollicitation, peut conduire à différentes perceptions du comportement du logiciel par ses utilisateurs. Le changement du

profil d'utilisation peut conduire à des variations de la fiabilité du logiciel se traduisant par une croissance ou une décroissance des intervalles jusqu'à défaillance du logiciel. En d'autres termes, on peut observer des périodes de croissance de fiabilité ou de décroissance de fiabilité qui ne sont pas nécessairement liées à l'introduction de modifications dans le logiciel.

I.3 MESURES DE LA SÛRETé DE FONCTIONNEMENT

Pour évaluer la sûreté de fonctionnement du logiciel, il est nécessaire de définir des mesures. Compte tenu de la nature non déterministe du comportement du logiciel au cours de son cycle de vie, les mesures établies pour caractériser sa sûreté de fonctionnement sont définies en termes de probabilités.

Selon que l'on s'intéresse au comportement du logiciel jusqu'à occurrence d'une défaillance ou bien à son comportement au cours de son cycle de vie en tenant compte des reprises d'exécution après défaillance, nous considérerons les deux variables aléatoires suivantes :

- T_i la variable aléatoire qui représente l'intervalle de temps séparant l'instant de reprise de l'exécution du logiciel après l'occurrence de la défaillance i-1 et l'instant d'occurrence de la défaillance i.
- N(t) la variable aléatoire qui représente le nombre d'occurrences de défaillances jusqu'à l'instant t d'observation du logiciel.

Nous nous limiterons à la présentation des mesures qui sont les plus utilisées dans le domaine de l'évaluation de la croissance de fiabilité en considérant les deux types de représentation du comportement du logiciel que nous avons identifiés dans le paragraphe précédent, c'est à dire en temps continu et en temps discret.

I.3.1 Mesures relatives au comportement jusqu'à défaillance

La variable aléatoire T_i caractérisant le comportement du logiciel jusqu'à occurrence d'une défaillance peut être soit une variable en temps discret mesurée par exemple en nombre d'exécutions[4], soit une variable en temps continu qui mesure le temps calendaire ou bien le temps d'exécution du logiciel[5].

Pour distinguer les deux types de représentation du comportement du logiciel, nous exprimerons les grandeurs caractéristiques de la variable aléatoire T_i respectivement en fonction de t, le temps courant, dans le cas où T_i est exprimée en temps continu, et en fonction de n, le nombre d'exécutions du logiciel, dans le cas où T_i est exprimée en temps discret.

La variable aléatoire T_i est caractérisée par un certain nombre de fonctions : la fonction densité de probabilité, la fonction de répartition (ou la distribution), la fonction de survie (qui correspond à la fiabilité du logiciel évaluée entre les défaillances i-1 et i), l'espérance

---

4 Selon le type de logiciel étudié, une exécution peut avoir plusieurs significations : par exemple, une mission, un jeu de test, une transaction, etc.…

5 Pour certains logiciels tels que les systèmes d'exploitation, il est difficile de définir ce qu'est une exécution. Dans ce cas la représentation de la variable T en temps continu semble plus appropriée.

mathématique qui correspond au temps moyen séparant l'instant de reprise de l'exécution du logiciel après l'occurrence de la défaillance i-1 et l'instant d'occurrence de la défaillance i, le taux de hasard appelé taux de défaillance, etc…

Les propriétés des grandeurs caractéristiques des variables aléatoire T_i dans le cas où T_i est une variable en temps continu sont présentées et analysées en détail dans la littérature [Gnedenko 1969, Barlow 1975, Corazza 1975, etc…]. Cependant, les travaux concernant l'étude des propriétés des mesures de fiabilité dans le cas où les variables aléatoires T_i sont spécifiées en temps discret sont plus rares et beaucoup plus récents. On peut citer, cependant, quelques références sur ce sujet : [Kalbfleisch 80, Salvia 82, Padgett 85].

Pour cette dernière raison, nous rappelons dans la figure I.5, les définitions, en temps continu et en temps discret, des fonctions caractéristiques de la variable aléatoire T_i qui sont les plus couramment utilisées : la densité de probabilité, la fiabilité et le taux de défaillance. Les différents liens qui existent entre ces fonctions sont présentés sur la figure I.6.

	Fiabilité	Densité de probabilité	Taux de défaillance
T_i continue	$R_i(t) = Pr(T_i>t)$ $0<t<\infty$	$f_i(t) = \lim_{dt->0} \frac{Pr(t\le T_i<t+dt)}{dt}$	$\lambda_i(t) = \lim_{dt->0} \frac{Pr(t\le T_i<t+dt \mid T_i\ge t)}{dt}$
T_i discrète	$R_i(n) = Pr(T_i>n)$ $n = 1, 2, \ldots$	$f_i(n)= Pr(T_i =n)$	$\lambda_i(n) = P(T_i =n \mid T_i \ge n)$

Figure I.5 : Définitions des fonctions caractéristiques de Ti

	$R_i(t)$	$f_i(t)$	$\lambda_i(t)$
$R_i(t)$	*	$\int_t^{\infty} f_i(x)dx$	$\exp\int_0^t -\lambda_i(x)dx$
$f_i(t)$	$-\frac{dR_i(t)}{dt}$	*	$\lambda_i(t)\exp\int_0^t -\lambda_i(x)dx$
$\lambda_i(t)$	$-\frac{1}{R_i(t)}\frac{dR_i(t)}{dt}$	$\frac{f_i(t)}{\int_t^{\infty} f_i(x)dx}$	*

a) temps continu

	$R_i(n)$	$f_i(n)$	$\lambda_i(n)$
$R_i(n)$	*	$\sum_{n+1}^{\infty} f_i(j)$	$\prod_{j=1}^{n}[1-\lambda_i(j)]$
$f_i(n)$	$R_i(n-1)-R_i(n)$	*	$\lambda_i(n)\prod_{j=1}^{n-1}[1-\lambda_i(j)]$
$\lambda_i(n)$	$\frac{R_i(n-1)-R_i(n)}{R_i(n-1)}$	$\frac{f_i(n)}{\sum_{n}^{\infty} f_i(j)}$	*

b) temps discret

Figure I.6 : Liens entre les fonctions caractéristiques de Ti

I.3.2 Mesures relatives au comportement après reprise d'exécution

Les mesures présentées dans le paragraphe précédent sont associées aux variables aléatoires T_i caractérisant le comportement du logiciel par rapport à l'occurrence d'une défaillance. Si on considère l'évolution du comportement du logiciel au cours de son cycle de vie en tenant compte des reprises d'exécution après occurrence de défaillances, on est amené à tenir compte de la succession des intervalles de temps jusqu'à défaillance. D'une manière équivalente, on peut décrire le comportement du logiciel par un processus stochastique

{N(t), t ≥0} où N(t) est la variable aléatoire qui représente le nombre d'occurrences de défaillances jusqu'à l'instant t d'observation du logiciel.

Les grandeurs caractéristiques de la variable aléatoire N(t) qui sont couramment considérées sont les suivantes :

- M(t), l'espérance mathématique de la variable aléatoire N(t) qui représente le nombre moyen de défaillances survenant dans l'intervalle [0,t] :

$$M(t) = E[N(t)] \tag{I.5}$$

- h(t), l'intensité de défaillance, définie par :

$$h(t) = \lim_{dt->0} \frac{Pr\{N(t+dt) - N(t) = 1\}}{dt} \tag{I.6}$$

Dans le cas où la probabilité d'occurrence simultanée de plusieurs défaillances est négligeable, c'est-à-dire : $Pr\{N(t+dt) - N(t) \geq 2\} = o(t)\,dt$ où o(t) est une fonction qui tend vers zéro plus vite que t quand t tend vers zéro, le processus stochastique {N(t), t ≥ 0} est régulier [Ascher 84]. h(t) et M(t) vérifient alors la relation suivante :

$$h(t) = \frac{dM(t)}{dt} \tag{I.7}$$

Les définitions (I.5 et I.6) supposent que le processus de défaillance est modélisé par un processus stochastique en temps continu. Dans le cas où le comportement du logiciel est décrit par un processus en temps discret, on peut définir des mesures équivalentes. Il suffit en fait, d'exprimer les mesures considérées en temps discret en fonction de l'unité de temps choisie.

En particulier, on associera à la variable aléatoire N(n), indiquant le nombre d'occurrences de défaillance du logiciel après n exécutions, les deux grandeurs suivantes :

- M(n), le nombre moyen de défaillances cumulé après n exécutions :

$$M(n) = E[N(n)] \tag{I.8}$$

- P(n), la probabilité d'occurrence d'une défaillance au cours de la n ième exécution du logiciel :

$$P(n) = Pr\{N(n) - N(n-1) = 1\} \tag{I.9}$$

Pour toutes les mesures précédentes, nous nous sommes intéressé à la caractérisation de l'évolution de la fiabilité du logiciel au cours de son cycle de vie. Pour cette raison, les temps jusqu'à restauration du logiciel sont supposés négligeables et ne sont pas pris en compte. Quand les durées d'arrêt du logiciel suite à l'occurrence de défaillances sont importantes, il est intéressant d'évaluer la disponibilité du logiciel pour mesurer l'impact des arrêts des traitements sur le service délivré aux utilisateurs. Les définitions relatives à la disponibilité sont classiques et peuvent se trouver dans [Barlow 75].

I.3.3 Processus de Poisson Non Homogènes (NHPP)

Parmi les classes de processus stochastiques qui sont très utilisés dans la modélisation de la croissance de fiabilité du logiciel, on distingue les processus de Poisson non homogènes (NHPP). Compte tenu de l'importance des modèles NHPP dans le domaine de l'évaluation de la fiabilité du logiciel, et plus particulièrement dans les travaux qui sont présentés dans ce mémoire, nous présentons dans ce paragraphe leurs principales propriétés ainsi que les expressions des mesures de fiabilité dans le cas où le processus de défaillance du logiciel est modélisé par un NHPP.

Un processus stochastique $\{N(t), t \geq 0\}$ est un processus NHPP caractérisé par la fonction intensité h(t) s'il est régulier et s'il est à accroissements indépendants [Cox 66].

Un processus stochastique est dit régulier si la probabilité d'occurrence simultanée de deux événements est négligeable.

$\{N(t), t \geq 0\}$ est un processus à accroissements indépendants si, pour $t_0 \leq t_1 \leq \ldots \leq t_n$ les variables aléatoires $N(t_0)$, $N(t_1) - N(t_0)$, …, $N(t_n) - N(t_{n-1})$ sont indépendantes. Cette propriété signifie que le nombre d'occurrences du processus sur un intervalle donné n'est pas influencé par le nombre d'occurrences du processus sur un autre intervalle disjoint.

Dans le cas où le processus de défaillance du logiciel est modélisé par un NHPP, le nombre d'occurrences de défaillances sur un intervalle $[t_1, t_2]$ suit une loi de Poisson de moyenne égale à :

$$E[N(t_2) - N(t_1)] = \int_{t_1}^{t_2} h(x)\,dx = M(t_2) - M(t_1) \qquad \text{(I.10)}$$

Posons : $H(t) = \int_0^t h(x)\,dx$

Compte tenu de la relation (I.7), H(t) est égal à M(t), le nombre cumulé moyen d'occurrences de défaillances dans l'intervalle [0,t].

Considérons maintenant les mesures de fiabilité caractérisant le comportement d'un logiciel qui est modélisé par un processus NHPP.

Soient :

s l'instant d'occurrence de la défaillance i-1,

t le temps courant, avec t > s,

t' le temps écoulé depuis l'occurrence de la défaillance i-1 : t'= t-s.

La fonction de survie (ou de fiabilité) $R_i(t'|\, s)$, associée à l'occurrence de la défaillance i est donnée par :

$$R_i(t'|s) = \exp -[H(s+t') - H(s)] \qquad \text{(I.11)}$$

Compte tenu des relations présentées sur la figure I.6, on peut obtenir à partir de la relation (I.11) respectivement les expressions de la densité de probabilité $f_i(t'|s)$ et du taux de défaillance $\lambda_i(t'|s)$. On trouve :

$$f_i(t'|s) = h(s+t') \exp -[H(s+t') - H(s)] \quad \text{(I.12)}$$

$$\lambda_i(t'|s) = h(s+t') \quad \text{(I.13)}$$

On remarque, d'après la relation (I.13), que dans le cas où le processus de défaillance du logiciel est modélisé par un processus NHPP, le taux de défaillance est une fonction continue qui a la même expression que l'intensité de défaillance du processus ; seul l'instant d'origine change. La relation entre taux de défaillance et intensité de défaillance est illustrée par la figure I.7.

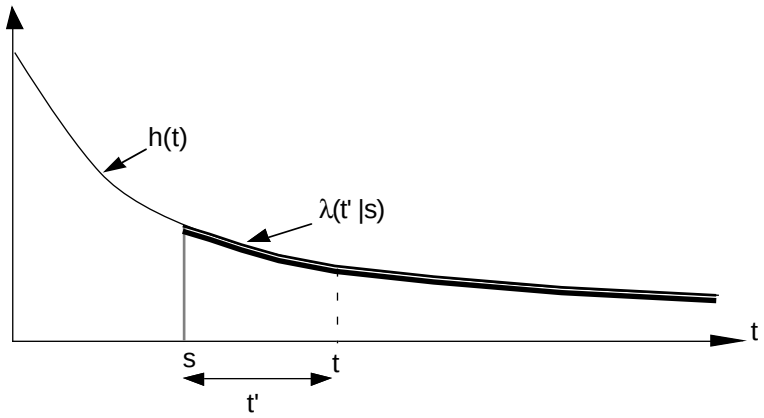


Figure I.7 : Relation entre taux de défaillance et intensité de défaillance pour un NHPP

Outre les mesures précédentes, on peut obtenir aussi l'expression de $MTTF_i$, le temps moyen jusqu'à la défaillance i conditionné par s l'instant d'occurrence de la défaillance i-1, défini par :

$$MTTF_i = \int_0^{\infty} t' f_i(t'|s) dt' \quad \text{(I.14)}$$

Si on intègre par parties la relation (I.14), on remarque que dans le cas où :

$$\lim_{t' \to 0} t' R_i(t'|s) = 0$$

MTTFi et $R_i(t'|s)$ sont liés par la relation suivante :

$$MTTF_i = \int_0^{\infty} R_i(t'|s) dt' \quad \text{(I.15)}$$

En tenant compte de la relation (I.11), on obtient finalement :

$$MTTF_i = \int_0^{\infty} \exp -[H(s+t') - H(s)] \, dt' \quad \text{(I.16)}$$

I.3.4 Conclusion

Les différentes mesures présentées dans ce paragraphe sont celles qui sont couramment utilisées dans le domaine de l'évaluation de la croissance de fiabilité du logiciel pour, d'une part, suivre l'évolution du nombre de défaillances du logiciel pendant sa validation et au cours de sa vie opérationnelle afin d'estimer l'effort de test et de maintenance nécessaire pour corriger les fautes résiduelles du logiciel, et, d'autre part, pour évaluer l'aptitude du logiciel à fournir aux utilisateurs des services conformément aux critères de sûreté de fonctionnement qui sont définis dans les spécifications.

Dans un contexte réel, il n'est pas nécessaire d'évaluer toutes les mesures présentées. En fonction des objectifs fixés par les concepteurs du logiciel, on peut être amené à effectuer un choix entre ces différentes mesures. Plus particulièrement, l'intensité de défaillance et le nombre cumulé de défaillances sont plus appropriés pour suivre l'évolution du nombre de défaillances du logiciel au cours de son cycle de vie et le taux de défaillance et le MTTF, sont plus adaptés pour suivre l'évolution de la fiabilité du logiciel au cours de son cycle de vie et vérifier, par exemple, avant la mise en opération du logiciel, si les critères spécifiés de la sûreté de fonctionnement sont satisfaits.

I.4 MODÈLES DE CROISSANCE DE FIABILITé DU LOGICIEL

Les mesures d'évaluation de la croissance de fiabilité étant introduites, il est maintenant indispensable de présenter et d'analyser les modèles existants qui permettent d'évaluer ces mesures.

Le domaine de l'évaluation de la croissance de fiabilité du logiciel a donné naissance, depuis une vingtaine d'années, à plusieurs modèles qui diffèrent essentiellement par les hypothèses établies concernant la nature et l'évolution des lois relatives aux variables aléatoires caractérisant le processus de défaillance du logiciel.

On dénombre actuellement plus de quarante modèles de croissance de fiabilité dont une revue détaillée peut se trouver dans [Goel 83]. Compte tenu du nombre important de modèles de croissance de fiabilité, plusieurs auteurs ont procédé à une classification des modèles proposés par rapport à différents critères. Parmi les classifications proposées, on peut citer :

- la classification effectuée dans [Ramamoorthy 82] qui est basée sur l'application des modèles au cours des différentes phases du cycle de vie du logiciel,
- la classification proposée dans [Goel 85] qui distingue les modèles à dénombrement de défaillance des modèles à taux de défaillance,
- la classification de [Miller 86] qui a développé un modèle général basé sur la représentation des instants d'occurrence des défaillances par des statistiques d'ordre de variables de lois exponentielles ; il a ensuite classé les modèles selon la nature et l'allure d'évolution des paramètres de ces lois,

- la classification de [Laprie 84-b, Kanoun 89] qui distingue les modèles à taux de défaillance et les modèles NHPP à intensité de défaillance,
- la classification de [Gaudoin 90] qui classe les modèles selon les propriétés de la fonction intensité de défaillance.

L'intérêt de procéder à une classification des modèles de croissance de fiabilité est d'identifier ceux qui sont basés sur des hypothèses équivalentes afin d'éviter d'utiliser plusieurs modèles qui correspondent au même type de comportement du système considéré et qui conduisent par conséquent à des résultats équivalents.

Dans la suite nous nous baserons sur la classification proposée dans [Laprie 84-b, Kanoun 89] pour présenter brièvement les modèles de croissance de fiabilité. Notre objectif n'est pas de présenter en détail tous les modèles existants mais plus d'analyser les points communs et les différences qui existent entre ces modèles et surtout de situer le modèle hyperexponentiel sous ses deux aspects, temps continu et temps discret par rapport à ces modèles.

Conformément aux mesures présentées dans le paragraphe précédent, nous considérerons deux types de modèles : les modèles en temps continu et les modèles en temps discret.

I.4.1 Modèles de croissance de fiabilité en temps continu

La plupart des modèles de croissance de fiabilité définis dans la littérature sont des modèles en temps continu qui modélisent l'évolution dans le temps de la fiabilité du logiciel. Le temps est soit le temps calendaire soit le temps d'exécution du logiciel [Musa 79].

[Laprie 84-b, Kanoun 89] distinguent deux types de modèles qui sont basés sur deux approches différentes pour la modélisation du processus de défaillance du logiciel : les modèles à taux de défaillance et les modèles NHPP à intensité de défaillance.

I.4.1.1 Modèles à taux de défaillance

Les modèles à taux de défaillance caractérisent l'évolution du comportement du logiciel entre l'occurrence de deux défaillances successives par la spécification d'une part, de l'expression du taux de défaillance du logiciel conditionné par le passé du processus jusqu'à l'occurrence de la dernière défaillance, et d'autre part, de la relation qui existe entre les taux successifs. Le passé du processus se réduit pour la plupart des modèles proposés au nombre de défaillances observées ; certains modèles considèrent également le temps écoulé depuis l'occurrence de la dernière défaillance.

Selon les hypothèses de modélisation considérées, on peut distinguer deux catégories de modèles à taux de défaillance :

- les modèles "déterministes" qui tiennent compte uniquement de la première source d'incertitude sur le comportement du logiciel : l'incertitude sur les entrées ; ils supposent que les variables aléatoires T_i sont indépendantes et distribuées selon des lois exponentielles de paramètre λ_i.

- les modèles "doublement stochastiques" qui tiennent compte des deux sources d'incertitude sur le comportement du logiciel : l'incertitude sur les entrées et l'incertitude sur les programmes ; contrairement aux autres modèles, ils supposent que les paramètres λ_i sont des variables aléatoires.

La figure I.8 donne des exemples de modèles appartenant à chacune de ces deux catégories .

Type de Modèle	Exemples de Modèles
déterministe	Jelinski-Moranda [Jelinski 72] Musa [Musa 75] Moranda [Moranda 75] Gaudoin I[6] [Gaudoin 90]
doublement stochastique	Littlewood-Verall [Littlewood 73] Littlewood [Littlewood 81] Keiller-Littlewood [Keiller 83] Gaudoin II[7] [Gaudoin 90]

Figure I.8 : Modèles à taux de défaillance

Pour ces modèles, l'introduction des corrections dans le logiciel se traduit par une discontinuité dans la fonction taux de défaillance qui exprime la variation de la fiabilité du logiciel suite à l'élimination des fautes.

Mis à part les modèles de Gaudoin, de Littlewood-Verall et de Keiller-Littlewood, qui tiennent compte d'une possibilité éventuelle de décroissance de fiabilité suite à l'introduction des corrections, tous les autres modèles considèrent que la correction du logiciel entraîne l'amélioration de sa fiabilité.

De plus, à l'exception du modèle Gaudoin I, tous les modèles à taux de défaillance, font l'hypothèse que le taux de défaillance du logiciel tend asymptotiquement vers une valeur nulle, supposant ainsi que le logiciel tend au bout d'un certain temps vers un logiciel parfait. Le modèle Gaudoin I peut modéliser l'impact des fautes qui demeurent dans le logiciel en dépit des corrections effectuées.

I.4.1.2 Modèles NHPP

Les modèles NHPP modélisent globalement le processus d'occurrence des défaillances par la caractérisation de la fonction intensité de défaillance. Contrairement aux modèles à taux de défaillance, ces modèles ne font pas d'hypothèse explicite sur le lien entre les défaillances et les corrections et le taux de défaillance du logiciel caractéristique de ces modèles est une fonction continue dans le temps.

Selon la forme de la fonction intensité de défaillance, on distingue trois types de modèles NHPP :

- modèles à intensité croissante puis décroissante tendant vers 0 quand $t \rightarrow \infty$,

---

6 Il s'agit du modèle à double taux de correction déterministe.

7 Il s'agit du modèle proportionnel lognormal.

- modèles à intensité décroissante tendant vers 0 quand t $\rightarrow\infty$,
- modèles à intensité décroissante tendant vers une limite non nulle quand t $\rightarrow\infty$.

Sur la figure I.9 nous donnons des exemples de modèles de croissance de fiabilité appartenant à chacun des trois types de modèles considérés ainsi que l'allure d'évolution de l'intensité de défaillance h(t) correspondante.

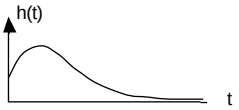
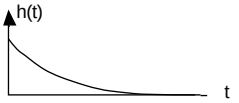
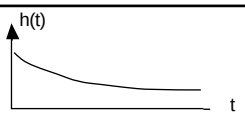
Type de Modèle	Allure de h(t)	Exemples de Modèles
h(t) croissante puis décroissante tendant vers une limite nulle		"Delayed S-shaped" [Yamada 83] "Inflexion S-shaped" [Ohba 82] "Goel généralisé " [Goel 85] "Logistic NHPP" [Fukishima 86]
h(t) décroissante tendant vers une limite nulle		"Crow" [Crow 77] "Exponentiel" [Goel 79] "Logarithmique" [Musa 84]
h(t) décroissante tendant vers une limite non nulle		"Finkelstein" [Finkelstein 79] **"Hyperexponentiel"** [Laprie 84-a, Kanoun 85]

Figure I.9 : Modèles à intensité de défaillance

I.4.1.2.1 Modèles à intensité croissante puis décroissante tendant vers une limite nulle

L'intensité de défaillance relative aux modèles appartenant à cette catégorie a une allure en forme de cloche. Ces modèles sont appelés modèles en forme de S à cause de l'allure de la fonction nombre cumulé de défaillances.

Pour ces modèles, on fait l'hypothèse qu'au début de la validation d'un logiciel, l'intensité de défaillance tend à croître. La croissance de l'intensité de défaillance, synonyme d'une décroissance de la fiabilité, est liée au nombre important de fautes qui sont généralement activées dès le début du test du logiciel. Au fur et à mesure de l'avancement de la validation, l'intensité de défaillance tend à décroître, traduisant ainsi une croissance progressive de la fiabilité du logiciel, et atteind une valeur asymptotique nulle avec l'hypothèse que toutes les fautes du logiciel peuvent être activées et corrigées.

I.4.1.2.2 Modèles à intensité décroissante tendant vers une limite nulle

Ces modèles supposent une croissance monotone de la fiabilité du logiciel telle que, asymptotiquement, le logiciel tend vers un logiciel parfait à taux de défaillance nul.

Le modèle de Crow appartenant à cette catégorie a été établi à partir de l'interprétation NHPP d'un modèle proposé par Duane [Duane 64] pour modéliser la croissance de fiabilité du matériel ; ce dernier modèle est à notre connaissance le seul modèle de croissance de fiabilité qui a été développé afin de tenir compte d'une possibilité éventuelle de croissance de fiabilité pour le matériel. Ce modèle a été utilisé également pour modéliser la croissance de fiabilité du logiciel.

I.4.1.2.3 Modèles à intensité décroissante tendant vers une limite non nulle

Ces modèles supposent que le logiciel atteint au cours de sa vie opérationnelle un comportement asymptotique en fiabilité stabilisée caractérisé par un processus de Poisson homogène à taux constant. Ils permettent ainsi de modéliser l'impact des fautes qui demeurent dans le logiciel malgré les corrections effectuées au cours de son cycle de vie.

Il n'existe à notre connaissance que deux modèles NHPP de cette catégorie : le modèle de Finkelstein [Finkelstein 79] qui est une extension du modèle de Duane obtenu en superposant à la fonction intensité un taux limite modélisant l'impact des fautes incorrigibles dans le matériel, et le modèle hyperexponentiel du LAAS qui sera présenté en détail dans le chapitre II.

I.4.2 Modèles de croissance de fiabilité en temps discret

Les modèles qui appartiennent à cette catégorie expriment l'évolution du comportement d'un système en fonction du nombre d'exécutions effectuées. La signification d'une exécution varie d'un modèle à un autre.

Pour le modèle discret de Yamada [Yamada 84] et le modèle hypergéométrique de Tohma [Tohma 90] une exécution correspond à une séquence de tests, effectuée au cours de la validation du logiciel, pendant laquelle on peut observer plusieurs défaillances.

Le modèle logarithmique de Finkelstein [Finkelstein 83] a été défini pour évaluer la fiabilité de certains systèmes qui ne sont pas utilisés en continu. Pour ce modèle, une exécution correspond à une mission effectuée par le système ; deux résultats sont possibles : soit l'exécution est effectuée avec succès soit elle entraîne la défaillance du système.

D'un point de vue conceptuel, le modèle de Finkelstein est équivalent aux modèles qui sont basés sur l'exploration de l'espace des entrées du logiciel pour lesquels une exécution correspond à une trajectoire dans l'espace des entrées. A notre connaissance, il n'existe que deux modèles de ce type, le modèle doublement stochastique de Bastani [Bastani 80, Ramamoorthy 80] et le **modèle hyperexponentiel en temps discret** que nous introduirons dans le chapitre III.

I.4.3 Commentaires sur les modèles

Mis à part le modèle hyperexponentiel qui permet d'évaluer la croissance de fiabilité et de disponibilité, tous les modèles que nous avons présentés se limitent à l'évaluation de la fiabilité.

Par ailleurs, bien que l'influence de l'environnement d'utilisation du logiciel sur sa sûreté de fonctionnement ne soit plus à démontrer [Iyer 85, Ehrlich 90], mis à part le modèle de Bastani et le modèle hyperexponentiel en temps discret, tous les modèles présentés ne prennent pas en compte la variation de l'environnement d'utilisation du logiciel dans l'évaluation des mesures de sûreté de fonctionnement. Ils supposent que l'environnement d'utilisation du logiciel est homogène tout au long de son cycle de vie.

On peut remarquer aussi que les modèles de croissance de fiabilité ne représentent que deux types de comportement du logiciel : une croissance de fiabilité, ou bien une décroissance de fiabilité suivie d'une croissance de fiabilité. Bien que, le comportement réel du logiciel au cours de son cycle de vie soit caractérisé par une succession de périodes de croissance de fiabilité, de

décroissance de fiabilité et de fiabilité stabilisée, les modèles de croissance de fiabilité proposés peuvent approcher de façon satisfaisante le comportement du logiciel au cours de son cycle de vie dans le cas où les variations de la fiabilité du logiciel sont locales et espacées dans le temps. Ceci est illustré sur la figure I.10 où la courbe (a) correspond à une allure typique de l'évolution de la fréquence d'occurrence de défaillances du logiciel, la courbe (b) correspond à une courbe typique de l'évolution de la fonction taux de défaillance relative à un modèle de croissance de fiabilité à taux de défaillance, et la courbe (c) correspond à un exemple de courbe intensité de défaillance relative à un modèle NHPP pour lequel l'intensité de défaillance décroît de façon continue dans le temps.

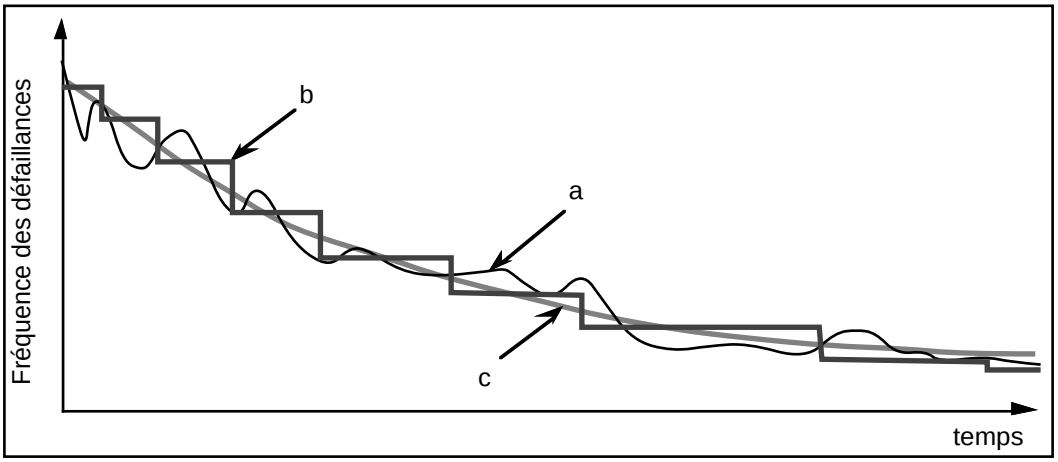


Figure I.10 : Exemple d'évolution de la fréquence d'occurrence de défaillance d'un logiciel réel

Toutefois, dans certains cas de figure, on peut observer de fortes variations locales de la fiabilité du logiciel (par exemple une forte décroissance de fiabilité résultant de la modification des spécifications). L'application des modèles de croissance de fiabilité du logiciel, dans ces conditions, peut devenir difficile sinon impossible, puisque les modèles ne tiennent pas compte d'un tel comportement.

En voie de conséquence, il est nécessaire de tenir compte de la variation de la fiabilité du logiciel au cours de son cycle de vie pour permettre une meilleure utilisation des modèles de croissance de fiabilité.

Les problèmes qui sont liés à l'application des modèles de croissance de fiabilité sont abordés dans le paragraphe suivant dans lequel nous présentons la méthode que nous utilisons pour l'analyse et l'évaluation de la fiabilité du logiciel dans laquelle les modèles de croissance de fiabilité sont appliqués.

I.5 PRéSENTATION D'UNE MéTHODE D'éVALUATION DE LA CROISSANCE DE FIABILITé DU LOGICIEL

La plupart des modèles de croissance de fiabilité sont paramétriques. Pour appliquer un modèle, il est nécessaire de collecter des données sur le logiciel étudié afin d'estimer les paramètres du modèle et de comparer les mesures évaluées à partir du modèle avec le comportement observé du logiciel.

L'approche classique qui est généralement considérée pour l'application des modèles de croissance de fiabilité consiste à les appliquer systématiquement aux données collectées sans tenir compte des hypothèses effectuées par ces modèles concernant l'évolution du comportement du logiciel au cours de son cycle de vie. L'utilisation des modèles selon cette approche s'est confrontée à plusieurs difficultés et a connu beaucoup d'échecs. Une des raisons de ces échecs est l'application des modèles de croissance de fiabilité de façon systématique sans effectuer une analyse préliminaire du comportement observé du logiciel et sans tenir compte des résultats de cette analyse lors de l'application des modèles [Kanoun 88].

La méthode d'analyse et d'évaluation de la croissance de fiabilité du logiciel proposée dans [Kanoun 88, Kanoun 89] a été définie afin d'assurer une meilleure utilisation dans un contexte industriel des modèles de croissance de fiabilité. Cette méthode est celle que nous utilisons dans le cadre de ce mémoire pour illustrer l'application des modèles de croissance de fiabilité à des données expérimentales. Pour cette dernière raison, nous présenterons dans la suite ses principales caractéristiques.

La méthode proposée est constituée de deux étapes :

. une étape spécifique au système étudié qui consiste à analyser l'architecture du système, la procédure de collecte de données considérée, l'environnement d'utilisation du logiciel, et les données collectées,

. une étape plus systématique qui consiste d'une part, à appliquer des tests de tendance pour analyser la variation de la fiabilité du logiciel, et d'autre part, à appliquer les modèles en se servant des résultats de ces tests.

Etant donné que la qualité des estimations effectuées par les modèles dépend intrinsèquement de la qualité des données collectées, la majeure partie de la première étape de la méthode proposée est consacrée, d'une part, au choix des données à collecter sur le logiciel pour évaluer les mesures de sûreté de fonctionnement, et d'autre part, à l'analyse des données collectées pour s'assurer de la qualité et de la consistance des données fournies.

La première étape de la méthode est développée en détail dans [Sabourin 87, Kanoun 88]. Cette étape de la méthode sort du cadre des travaux présentés dans ce mémoire. Nous présenterons uniquement les principales caractéristiques de la deuxième étape de la méthode que nous utiliserons par la suite.

I.5.1 Analyse de tendance

Il s'agit d'étudier la variation de fiabilité observée du logiciel. En d'autres termes, décider au vu des données collectées si, durant la période considérée, la fiabilité du logiciel croît, décroît, ou bien est restée stabilisée.

L'analyse de tendance est effectuée par l'application aux données collectées de tests de tendance, graphiques ou analytiques, qui sont définis en fonction de la variable aléatoire choisie pour représenter le comportement du logiciel : les intervalles de temps entre défaillances ou le nombre de défaillances par unité de temps.

La croissance de fiabilité du logiciel peut être testée en fonction du type de tendance observée sur la variable considérée. Par exemple, si les temps entre défaillances tendent à décroître (respectivement à croître) alors on peut conclure que la fiabilité du logiciel croît (respectivement décroît). Le même type de raisonnement peut être considéré dans le cas où on considère le nombre de défaillances par unité de temps.

I.5.1.1 Définitions formelles

De façon plus formelle, si on note par T_1, T_2, … les variables aléatoires correspondant aux intervalles de temps entre occurrences de défaillances, et $F_{Ti}(t)$ la fonction de distribution associée à T_i, une croissance de fiabilité se traduit par une croissance stochastique des intervalles de temps entre défaillances [Kanoun 91-d] :

$$T_j \underset{st}{\leq} T_i \quad \text{pour tout } j < i \tag{I.17}$$

Sous l'hypothèse d'indépendance stochastique des variables T_i, la relation (I.17) est équivalente à :

$$F_{T_j}(t) \geq F_{T_i}(t) \quad \text{pour tout } t \text{ et pour tout } j < i \tag{I.18}$$

Dans le cas où les données considérées sont sous forme de nombre de défaillances par unité de temps, l'hypothèse de croissance de fiabilité peut s'exprimer par la relation suivante :

$$N(0,t_2) \underset{st}{\geq} N(t_1,t_1+t_2) \quad \text{pour tout } t_1,\ t_2 \geq 0 \tag{I.19}$$

où $N(t_i,t_j)$ est la variable aléatoire indiquant le nombre d'occurrences de défaillances dans l'intervalle $[t_i,t_j]$.

L'inégalité (I.19) doit être stricte au moins pour un couple (t_1,t_2).

La relation (I.19) se traduit de façon équivalente par :

$$E[N(0,t_2)] \geq E[N(t_1,t_1+t_2)] \quad \text{pour tout } t_1,\ t_2 \geq 0 \tag{I.20}$$

où $E[N(t_i,t_j)]$ est l'espérance mathématique de la variable aléatoire $N(t_i,t_j)$.

I.5.1.2 Tests de Tendance : le test de Laplace

Il existe plusieurs types de tests de tendance dont une présentation détaillée se trouve dans [Kanoun 89, Gaudoin 90]. Parmi les tests proposés, on distingue le test de Laplace. Ce dernier est simple à mettre en œuvre et possède des propriétés statistiques intéressantes en particulier quand les données collectées correspondent à un processus NHPP [Gaudoin 90]. Nous en rappelons ici ses principales propriétés et nous expliquons de façon succincte comment l'utiliser pour détecter les variations de la fiabilité du logiciel à partir des données collectées.

Le test de Laplace introduit dans [Cox 66] a été utilisé dans [Ascher 78] pour détecter la tendance de croissance ou de décroissance de la variable aléatoire intervalle de temps entre

défaillances. Ce test a été adapté ensuite dans [Kanoun 89] au cas où la variable aléatoire est le nombre de défaillances par unité de temps.

Le test de Laplace consiste à évaluer à partir des données collectées une statistique, notée u et appelée facteur de Laplace, qui est interprétée comme suit :

- u positif indique une décroissance globale de la fiabilité,
- u négatif indique une croissance globale de la fiabilité,
- une faible valeur de u ($|u| < 2$) sur une période de temps indique qu'il n'y a pas globalement de tendance prononcée de croissance ou de décroissance sur cette période.

Si on évalue le facteur de Laplace progressivement pas à pas, où, à chaque pas de l'évaluation on intègre une donnée supplémentaire dans le sous-ensemble utilisé pour évaluer le facteur de Laplace, on peut obtenir une courbe u(k), appelée courbe de Laplace, où u(k) est le facteur de Laplace évalué à partir des k premières données collectées. L'analyse de la variation de la courbe de Laplace u(k) en fonction de k permet d'étudier la variation progressive de la fiabilité du logiciel au fur et à mesure de la collecte de données. Un changement du sens de variation de u(k) est synonyme d'une variation locale de la fiabilité du logiciel. La figure I.9 résume les différentes situations qui peuvent se présenter. Les points A et C correspondent à des points de changement de tendance de la fiabilité où la décroissance de u(k) en A traduit un début de croissance locale de la fiabilité et la croissance de u(k) en C traduit une décroissance locale de la fiabilité. La période correspondant aux données collectées entre A et C correspond à une période de croissance locale de fiabilité du logiciel.

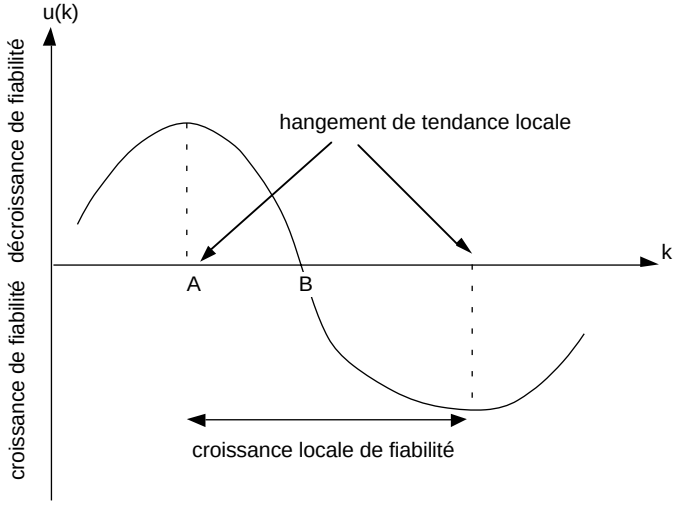


Figure I.11 : Analyse de tendance par le test de Laplace

Ainsi, l'analyse du sens de variation de la courbe de Laplace permet de suivre la variation de la fiabilité du logiciel au fur et à mesure de la collecte de données. On peut ainsi vérifier si le comportement du logiciel pendant la période considérée correspond au comportement modélisé par les modèles de croissance de fiabilité [Kanoun 89].

I.5.2 Application des modèles de croissance de fiabilité

La dernière étape de la méthode globale d'évaluation de la fiabilité du logiciel consiste à appliquer les modèles sélectionnés aux données collectées pour estimer les mesures de fiabilité correspondant aux objectifs fixés. Dans la suite nous développerons les deux points suivants :

- comment utiliser les résultats de l'analyse de tendance pour appliquer les modèles,
- comment appliquer les modèles pour évaluer les mesures de fiabilité.

I.5.2.1 Utilisation des tests de tendance pour l'application des modèles

L'analyse de la variation de la fiabilité du logiciel par les tests de tendance est déterminante pour l'application des modèles. Elle conditionne d'une part, le choix des modèles à appliquer en fonction du comportement observé du logiciel et, d'autre part, le choix des données à utiliser pour appliquer les modèles. Ceci est illustré par les exemples suivants :

- si on remarque que la fiabilité du logiciel tend à décroître de façon monotone durant la période considérée, alors on peut conclure qu'il est inutile d'appliquer les modèles de croissance de fiabilité puisqu'aucun des modèles existants ne prend en compte ce type de comportement. Un retour est nécessaire vers les équipes de maintenance et de développement du logiciel pour analyser les raisons d'un tel comportement du logiciel,

- si on observe une décroissance de fiabilité suivie d'une croissance de fiabilité, alors on peut appliquer un modèle pour lequel l'intensité de défaillance est en forme de cloche. Si on décide d'appliquer un modèle qui décrit une croissance monotone de fiabilité, il est préférable de n'appliquer le modèle qu'à partir du moment où on observe une croissance de fiabilité,

- si au cours de l'application d'un modèle de croissance de fiabilité sur une période donnée on observe un changement brutal de tendance dans l'évolution de la fiabilité qui se traduit par une décroissance locale de fiabilité, alors il est préférable d'arrêter l'application de modèle et attendre jusqu'à ce que le logiciel manifeste une croissance de fiabilité.

Ainsi, l'application des modèles de croissance de fiabilité en fonction de la tendance observée peut se ramener à un partitionnement des données collectées par période de croissance où le comportement observé du logiciel sur une période donnée correspond aux hypothèses établies par le modèle.

I.5.2.2 Calibrage et validation des modèles

L'application des modèles de croissance de fiabilité aux données sélectionnées après l'analyse de tendance s'effectue en trois phases :

- utilisation d'une procédure d'inférence pour calibrer le modèle (c'est-à-dire estimer ses paramètres),
- utilisation d'une procédure de prévision permettant d'évaluer les mesures choisies,
- application d'un ou de plusieurs critères de validation afin d'apprécier la qualité des estimations effectuées.

La procédure de prévision ne pose aucun problème particulier ; elle consiste à remplacer les paramètres intervenant dans les expressions des mesures associées au modèle considéré par les valeurs estimées à partir des données.

Nous nous limiterons donc à présenter de façon succincte les procédures d'inférence et les critères de validation.

I.5.2.2.1 Procédures d'inférence

La procédure d'inférence permet d'estimer les paramètres d'un modèle à partir des données collectées. Ils existe trois types de méthodes pour l'estimation des paramètres des modèles de croissance de fiabilité : la méthode des moindres carrés, la méthode du maximum de vraisemblance et la méthode bayesienne. Les deux premières sont les plus couramment utilisées. La méthode bayesienne, bien que très intéressante sur le plan théorique, est plus compliquée à mettre en œuvre et son application en pratique est très limitée [Mellor 87, Littlewood 89].

Le choix entre méthode des moindres carrés ou maximum de vraisemblance n'est pas évident et ne peut être effectué dans l'absolu. Pour certains modèles, on peut obtenir les expressions analytiques des estimateurs des paramètres associés au modèle par l'une ou l'autre des deux méthodes, ce qui détermine le choix entre ces deux méthodes d'estimation. Ceci n'est pas le cas de tous les modèles de croissance de fiabilité ; l'application des procédures d'inférence se ramène alors à l'utilisation de procédures numériques d'optimisation dont des exemples sont données dans [Rao 78].

I.5.2.2.2 Critères de validation et de comparaison de modèles

Ces critères sont utilisés pour apprécier l'adéquation des modèles de croissance de fiabilité pour représenter le comportement observé du logiciel.

L'adéquation peut être considérée selon deux aspects :

- la capacité réplicative qui traduit l'aptitude du modèle à reproduire le comportement passé du logiciel,
- la capacité prévisionnelle qui représente la capacité du modèle à prévoir le comportement futur du logiciel en fonction de son comportement passé.

Sur le plan pratique, pour étudier la capacité réplicative on calibre le modèle avec toutes les données observées. Pour étudier la capacité prévisionnelle, on utilise la première partie des données collectées pour estimer les paramètres du modèle et évaluer les mesures de sûreté de fonctionnement relatives à la deuxième partie. La capacité prévisionnelle est alors étudiée sur cette dernière.

Les critères de validation qui sont couramment utilisés dans le domaine de la croissance de fiabilité du logiciel sont le critère de Kolmogorov-Smirnof, appelé aussi "u plot", le critère de la vraisemblance préquentielle et le critère des résidus. Ces critères sont présentés en détail dans [Littlewood 89, Kanoun 89]. On peut citer aussi le critère de Akaike [Akaike 74] utilisé récemment dans [Khoshgoftaar 91] pour comparer plusieurs modèles.

Comparé aux critères de Kolmogorov-Smirnof et au critère de la vraisemblance préquentielle qui sont basés sur des théories statistiques, le critère des résidus est basé sur l'évaluation des écarts entre les valeurs observées et les valeurs estimées par les modèles. Il est d'une part, plus simple à mettre en œuvre, et d'autre part directement lié à la grandeur physique observée. Plusieurs types de résidus peuvent être considérés selon le type de comparaison choisie [Kanoun 89] : on peut considérer, par exemple, la somme des valeurs des résidus pour analyser

si, globalement, le modèle donne des estimations optimistes ou pessimistes, le résidu moyen pour évaluer en moyenne le comportement du modèle par rapport aux données observées, etc…

Toutefois, le critère des résidus n'est pas un critère statistique et ne peut pas être utilisé pour évaluer dans l'absolu la validité d'un modèle telle qu'elle est définie en statistiques [Saporta 78], où un modèle est, soit accepté, soit rejeté, avec un niveau de confiance donné. L'évaluation de la validité dans l'absolu d'un modèle de croissance de fiabilité reste encore un sujet ouvert dans l'état actuel des recherches effectuées dans le domaine de l'évaluation de la croissance de fiabilité. Par conséquent, les critères de validation considérés doivent être utilisés avec précaution et il est préférable d'utiliser au cours de l'application des modèles de croissance de fiabilité plusieurs critères de validation à la fois pour s'assurer de la cohérence et de la validité des résultats obtenus.

I.5.3 Conclusion

Dans ce paragraphe, nous avons présenté de façon succincte la méthode que nous utilisons pour appliquer les modèles de croissance de fiabilité. La méthode proposée a été mise en œuvre dans le logiciel SoRel [Kanoun 91-a] dans lequel sont développés plusieurs tests de tendance ainsi que différents modèles de croissance de fiabilité.

En plus de la méthode présentée dans ce paragraphe, il existe d'autres approches qui ont pour objectif de permettre aux utilisateurs des modèles de croissance de fiabilité d'obtenir des estimations satisfaisantes des mesures de fiabilité du logiciel. On peut citer par exemple l'approche adaptative présentée dans [Brocklehurst 90] qui consiste à utiliser des critères statistiques permettant d'améliorer les prévisions futures d'un modèle en fonction des résultats du modèle sur une période antérieure, et les approches multi-modèles. Ces dernières approches consistent à appliquer simultanément plusieurs modèles et à utiliser des critères de comparaison permettant de sélectionner à chaque étape de la prévision le modèle qui fournit les meilleures estimations. Des exemples d'approches multi-modèles peuvent se trouver dans [Littlewood 89, Khoshgoftaar 91, Keiller 91, Knafl 91].

CONCLUSION

Dans ce premier chapitre nous avons :

- analysé les mécanismes d'occurrence de défaillances du logiciel suite à l'activation de fautes de conception et nous avons défini trois types de comportement du logiciel : croissance de fiabilité, décroissance de fiabilité et fiabilité stabilisée,
- étudié l'influence de la variation du profil d'utilisation du logiciel sur sa sûreté de fonctionnement. Ceci nous a amené à distinguer deux types de représentation du comportement du logiciel : une représentation en temps discret caractérisant l'évolution de la sûreté de fonctionnement en fonction du nombre d'exécutions du logiciel, et une représentation en temps continu caractérisant son comportement tel qu'il est perçu par ses utilisateurs,

- présenté les mesures de la sûreté de fonctionnement en temps discret et en temps continu que nous utiliserons dans le cadre de ce mémoire, en mettant l'accent plus particulièrement sur les mesures relatives aux processus NHPP,
- introduit brièvement les modèles de croissance de fiabilité, et analysé les points communs et les différences qui existent entre ces modèles,
- présenté une méthode d'évaluation de la fiabilité du logiciel au sein de laquelle les modèles de croissance de fiabilité sont appliqués.

Les deux types de représentation du comportement du logiciel que nous avons identifiés dans ce chapitre seront considérés dans la suite de ce mémoire où nous proposerons deux modèles de croissance de fiabilité : le modèle hyperexponentiel en temps continu et le modèle hyperexponentiel en temps discret. Nous aborderons également les problèmes qui sont liés à la prise en compte de la variation du profil d'utilisation du logiciel dans l'évaluation de ses mesures de sûreté de fonctionnement.

Dans ce chapitre, nous nous sommes consacré à l'étude et à la caractérisation de la croissance de fiabilité du logiciel. Néanmoins, les analyses effectuées s'appliquent également au matériel quand il s'agit d'étudier son comportement en considérant à la fois les fautes physiques et les fautes de conception [Laprie 89]. En effet, en considérant les fautes physiques, la modélisation de la sûreté de fonctionnement du matériel se ramène à l'étude de son comportement en fiabilité stabilisée puisque la restauration du système considéré suite à l'activation d'une faute dans un composant du système se traduit par le remplacement du composant correspondant : le système est alors identique à ce qu'il était avant l'occurrence de de la défaillance correspondante. Concernant les fautes de conception, les conséquences de la restauration de service, avec ou sans élimination de ces fautes dans le matériel, conduit, de façon analogue au logiciel, à trois types de comportements possibles : croissance de fiabilité, fiabilité stabilisée ou décroissance de fiabilité.

En conséquence, tous les modèles de croissance de fiabilité pouvant prendre en compte à la fois le comportement d'un système en fiabilité stabilisée et son comportement en considérant le phénomène de croissance de fiabilité, peuvent être appliqués pour modéliser et évaluer la sûreté de fonctionnement du matériel.

CHAPITRE II

MODÈLE HYPEREXPONENTIEL EN TEMPS CONTINU

INTRODUCTION

L'évaluation de la fiabilité et de la disponibilité, des systèmes informatiques, au cours de leur vie opérationnelle, prenant en compte toutes les sources de défaillance quelque soit leur origine - fautes physiques et fautes de conception, est de toute première importance tant pour les utilisateurs que pour les concepteurs.

Il faut cependant noter que les études de modélisation de la sûreté de fonctionnement des systèmes informatiques intégrant l'évaluation du matériel et du logiciel sont très rares ; la raison en est que la modélisation de la sûreté de fonctionnement du matériel et celle du logiciel ont été pendant longtemps considérées comme des disciplines distinctes [Laprie 89].

En effet, la modélisation du matériel, a porté essentiellement sur le développement de modèles structurels permettant d'évaluer la sûreté de fonctionnement d'un système au cours de sa vie opérationnelle en tenant compte de la sûreté de fonctionnement de ses différents composants. Cependant, mis à part [Duane 64], de telles études se sont limitées à la modélisation des processus de défaillance et de restauration du matériel vis-à-vis de l'activation de fautes physiques sans tenir compte des fautes de conception internes. En d'autres termes, l'évaluation de la sûreté de fonctionnement du matériel s'est limitée à l'étude du comportement en fiabilité stabilisée des systèmes en ignorant que la fiabilité du matériel croît aussi en raison de la présence et de l'élimination des fautes de conception : les données collectées sur des systèmes réels mettent en évidence un tel phénomène [Bauer 85, Bastos 90].

Les études relatives à l'évaluation du logiciel pour la prévision de fautes se sont essentiellement axées sur la modélisation du phénomène de croissance de fiabilité en mettant l'accent plus particulièrement sur les phases de développement et de validation. Cependant, en dépit, de l'importance majeure de la disponibilité de service pour certains types de logiciels tels que les systèmes de télécommunications et les logiciels de messagerie électronique par exemple, cette mesure n'a pas été considérée dans les modèles de croissance de fiabilité.

Le modèle hyperexponentiel que nous présentons dans ce chapitre a pour objectif d'unifier la modélisation de la sûreté de fonctionnement du matériel et du logiciel en considérant simultanément les deux mesures complémentaires de la sûreté de fonctionnement que sont la fiabilité et la disponibilité. Plus précisément, ce modèle permet de modéliser la croissance de fiabilité et la croissance de disponibilité d'un système matériel et logiciel en tenant compte des fautes physiques et des fautes de conception pouvant être activées aussi bien au cours de la phase de validation que pendant la vie opérationnelle. Ce modèle permet également de considérer les deux approches possibles pour étudier le comportement d'un système :

l'approche "boîte noire" dans laquelle le système étudié est considéré comme un ensemble auquel on soumet des entrées et on observe des sorties, et l'approche "boîte blanche" dans laquelle on tient compte de la structure du système afin d'évaluer son comportement en fonction de celui de ses différents composants.

Le modèle hyperexponentiel a été introduit dans [Laprie 84-a, Kanoun 85] pour modéliser le comportement d'un système selon une vue "boîte noire". Dans ce mémoire, nous utilisons ce modèle dans le cadre d'une approche permettant d'évaluer également les mesures de sûreté de fonctionnement d'un système en tenant compte de sa structure, approche qui fera l'objet du chapitre IV.

Ce chapitre est consacré à la présentation du modèle hyperexponentiel. Nous présentons d'abord les principales propriétés du modèle et nous établissons les expressions des mesures permettant de caractériser l'évolution de la croissance de fiabilité d'un système. Nous nous intéressons ensuite à la modélisation de la croissance de disponibilité. En nous basant sur les propriétés du modèle hyperexponentiel, nous proposons une généralisation du modèle afin d'augmenter sa puissance de modélisation et nous étudions les caractéristiques du modèle obtenu. Nous présentons finalement quelques exemples d'application du modèle à des systèmes réels pour illustrer son aptitude à modéliser la fiabilité et la disponibilité des systèmes en tenant compte du phénomène de croissance de fiabilité.

II.1. PRéSENTATION DU MODÈLE

Le modèle hyperexponentiel fait partie de la classe des modèles NHPP dont les principales propriétés ont été présentées dans le paragraphe I.3.3.

L'intensité de défaillance qui caractérise le modèle est basée sur une loi de Cox hyperexponentielle ; son expression est la suivante :

$$h(t) = \frac{\omega\,\zeta_{sup}\,e^{-\zeta_{sup}t} + \varpi\,\zeta_{inf}\,e^{-\zeta_{inf}t}}{\omega\,e^{-\zeta_{sup}t} + \varpi\,e^{-\zeta_{inf}t}} \qquad \text{(II.1)}$$

avec $0 \le \omega \le 1$, $\omega + \varpi = 1$ et $\zeta_{inf} \le \zeta_{sup}$

L'intensité de défaillance h(t) est une fonction continue non croissante dans le temps (figure II-1), variant entre $h(0) = \omega\,\zeta_{sup} + \varpi\,\zeta_{inf}$ et $h(\infty) = \zeta_{inf}$.

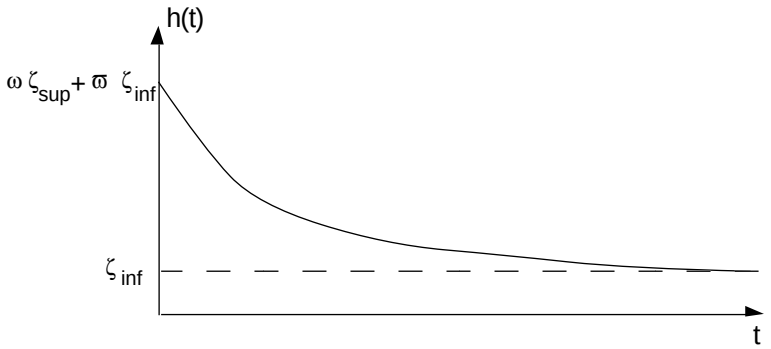


Figure II.1 : Intensité de défaillance du modèle hyperexponentiel

La dérivée première et la dérivée seconde de la fonction h(t) sont données respectivement par :

$$\frac{d\,h(t)}{dt} = -\frac{\omega\,\varpi\,(\zeta_{sup} - \zeta_{inf})^2\, e^{-(\zeta_{sup} + \zeta_{inf})t}}{\left\{\ \omega\, e^{-\zeta_{sup}\,t} + \varpi\, e^{-\zeta_{inf}t}\ \right\}^2} \qquad \text{(II.2)}$$

$$\frac{d^2\,h(t)}{dt^2} = -\frac{\omega\varpi\,(\zeta_{sup} - \zeta_{inf})^3\, e^{-(\zeta_{sup} + \zeta_{inf})t}\left\{\ \omega\, e^{-\zeta_{sup}t} - \varpi\, e^{-\zeta_{inf}t}\ \right\}}{\left\{\ \omega\, e^{-\zeta_{sup}\,t} + \varpi\, e^{-\zeta_{inf}t}\ \right\}^3} \qquad \text{(II.3)}$$

La dérivée première est toujours négative et la dérivée seconde peut changer de signe selon la valeur du paramètre ω :

- si $\omega \leq \varpi$, la dérivée seconde est toujours positive,
- si $\omega > \varpi$, la dérivée seconde est d'abord négative, et devient positive à partir de l'instant t donné par :

$$t = \frac{1}{\zeta_{sup} - \zeta_{inf}} \mathrm{Ln}\left(\frac{\omega}{\varpi}\right) \qquad \text{(II.4)}$$

Selon les valeurs des paramètres ω, ζ_{sup} et ζ_{inf}, le modèle peut représenter plusieurs types de comportement. En particulier, il peut modéliser les deux cas limites suivants :

- une intensité de défaillance constante, quand $\zeta_{sup} = \zeta_{inf}$ ou $\omega \in \{0,1\}$,
- une intensité de défaillance décroissante tendant asymptotiquement vers zéro, quand $\zeta_{inf} = 0$, $\zeta_{sup} \neq 0$ et $\omega \notin \{0,1\}$.

Le premier cas limite traduit un comportement en fiabilité stabilisée du système. Il permet de modéliser :

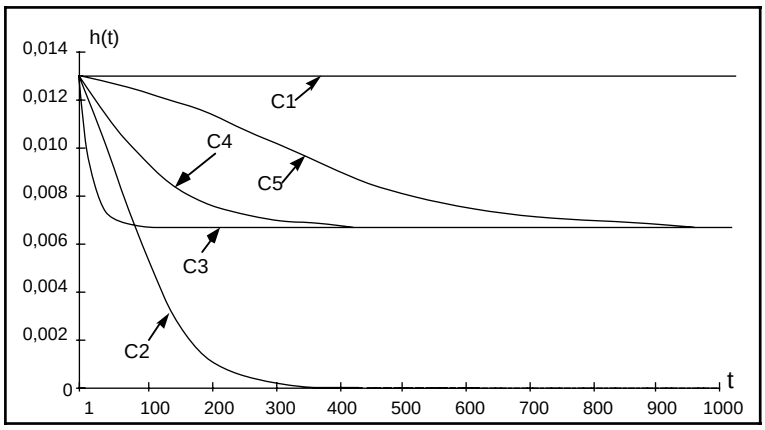
- le comportement du matériel vis-à-vis de l'activation de fautes physiques pour lequel on néglige les fautes de conception,
- le comportement d'un système matériel ou logiciel au cours de sa vie opérationnelle pendant laquelle les fautes de conception ne sont pas corrigées ou bien pour lequel la correction de ces fautes n'entraîne pas de modification significative de son taux de défaillance.

Le deuxième cas limite correspond au type de comportement modélisé par la plupart des modèles de croissance de fiabilité.

Sur la figure I.2, on présente des exemples d'évolution de l'intensité de défaillance du modèle hyperexponentiel couvrant plusieurs types de comportement du logiciel :

- C1 représente une fiabilité stabilisée,
- C2 correspond à une intensité de défaillance caractérisée par un taux asymptotique nul,
- C3 modélise une croissance de fiabilité très rapide,

- C4 traduit une croissance de fiabilité progressive,
- C5 représente une croissance de fiabilité lente.



Courbe	ω	ζ_{sup}	ζ_{inf}
C1	1,00	$1,3\ 10^{2}$	-
C2	0,75	$1,8\ 10^{2}$	0
C3	0,10	$7,0\ 10^{2}$	$7\ 10^{-3}$
C4	0,50	$2,0\ 10^{2}$	$7\ 10^{-3}$
C5	0,90	$1,4\ 10^{2}$	$7\ 10^{-3}$

Figure II.2 : Exemples de courbes intensité de défaillance du modèle hyperexponentiel

II.2. MESURES DE FIABILITé

Posons :

$$G(t) = \omega\, e^{-\zeta_{sup} t} + \varpi\, e^{-\zeta_{inf} t} \qquad \text{(II.5)}$$

Les mesures de fiabilité caractéristiques du modèle hyperexponentiel s'expriment d'une manière simple à l'aide la fonction G(t) :

$$h(t) = -\frac{G'(t)}{G(t)} \text{ avec } G'(t) = \frac{dG(t)}{dt} \qquad \text{(II.6)}$$

$$H(t) = -\operatorname{Ln} G(t) \qquad \text{(II.7)}$$

Compte tenu des notations introduites dans le paragraphe I.3.3 et des relations (I.11) à (I.13), on obtient les expressions suivantes des mesures de fiabilité associées au modèle hyperexponentiel qui caractérisent le comportement du système considéré après l'occurrence de la défaillance i-1 :

$$R_i(t'|s) = \frac{G(s+t')}{G(s)} \qquad \text{(II.8)}$$

$$f_i(t'|s) = -\frac{G'(s+t')}{G(s)} \qquad \text{(II.9)}$$

$$\lambda_i(t'|s) = -\frac{G'(s+t')}{G(s+t')} = h(s+t') \qquad \text{(II.10)}$$

Compte tenu de l'expression de $R_i(t'|s)$, on a : $\lim_{t' \to 0} t'\, R_i(t'|s) = 0$.

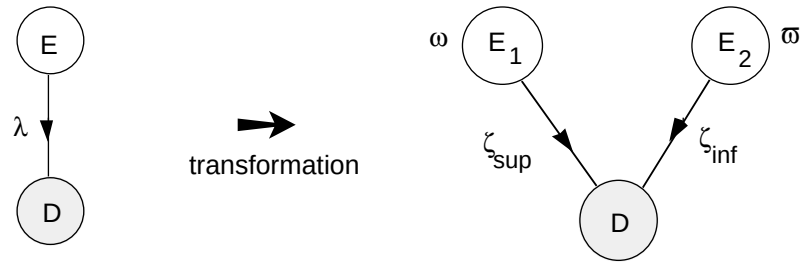
On peut alors utiliser la relation (I.16) pour obtenir l'expression de $MTTF_i$, le temps moyen jusqu'à occurrence de la défaillance i :

$$MTTF_i = \frac{\frac{\omega}{\zeta_{sup}} e^{-\zeta_{sup} s} + \frac{\varpi}{\zeta_{inf}} e^{-\zeta_{inf} s}}{\omega e^{-\zeta_{sup} s} + \varpi e^{-\zeta_{inf} s}} \qquad \text{(II.11)}$$

II.3. INTERPRéTATION MARKOVIENNE DU MODÈLE

Une propriété importante des modèles qui sont basés sur les processus NHPP est que l'intensité de défaillance et le taux de défaillance peuvent s'exprimer par la même fonction : seul l'instant d'origine change (cf paragraphe I.3.3) ; ceci est illustré pour le modèle hyperexponentiel par la relation II.10. En se basant sur cette propriété, on peut considérer que l'intensité de défaillance du modèle hyperexponentiel est dérivée d'une loi de Cox hyperexponentielle à deux étages [Cox 65]. Le modèle peut être alors représenté par une chaîne de Markov formée de trois états : un état absorbant D et deux états transitoires E_1 et E_2, avec des probabilités initiales d'occupation ω et ϖ et des taux de transition associés ζ_{sup} et ζ_{inf} (figure II.3-b).

La modélisation du phénomène de croissance de fiabilité peut se ramener alors à la transformation d'une chaîne de Markov classique caractérisant le comportement en fiabilité stabilisée du système formée de deux états (figure II.3-a) : un état transitoire E et un état absorbant D, en une chaîne de Markov à trois états (figure II.3-b) permettant de représenter la croissance de fiabilité du système. La transformation est illustrée sur la figure II-3.



(a) fiabilité stabilisée (b) croissance de fiabilité

Figure II.3 : Représentation markovienne du modèle hyperexponentiel

II.4. MODéLISATION DE LA CROISSANCE DE DISPONIBILITé

La modélisation de la disponibilité d'un système nécessite la prise en compte des intervalles de temps jusqu'à défaillance et des intervalles de temps jusqu'à restauration du service.

Les intervalles de temps jusqu'à restauration de service varient d'une manière non déterministe au cours de la vie d'un système informatique. En effet, suite à l'occurrence d'une défaillance, le temps nécessaire pour reprendre l'exécution du système peut être très court dans le cas où la restauration de service est effectuée par une simple relance du système, ou bien beaucoup plus long dans le cas où la restauration de service est effectuée après la correction des fautes activées.

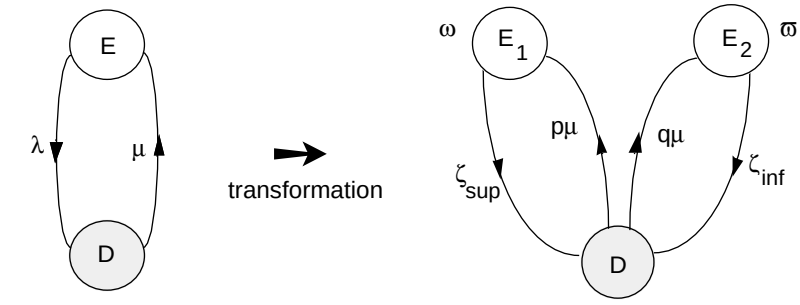
Le temps nécessaire pour corriger les fautes de conception activées dans un système varie en fonction de plusieurs facteurs. Il dépend de :

- la nature des fautes activées : les fautes, telles que les conditions d'activation sont faciles à reproduire, sont plus faciles à corriger que les fautes qui ne peuvent être activées que suite à l'occurrence de plusieurs événements se produisant très rarement au cours de l'exécution du système,
- l'efficacité des équipes de maintenance : c'est-à-dire le degré de familiarisation et le niveau de connaissance du fonctionnement du système considéré des ingénieurs,
- l'effort de maintenance alloué,
- les politiques de maintenance considérées dans l'entreprise (correction immédiate, correction différée, etc…).

Le manque de données d'expérience concernant les temps de correction des fautes activées dans des systèmes réels constitue un réel handicap pour modéliser l'évolution des taux de restauration d'un système au cours de son cycle de vie. Cependant, les études effectuées dans [Laprie 75, Costes 78] concernant l'influence de la distribution des temps de restauration d'un système informatique sur l'évolution de sa disponibilité, montrent que l'hypothèse d'une distribution exponentielle à taux de restauration constant est satisfaisante dans le cas où les intervalles de temps jusqu'à restauration de service sont très petits devant les intervalles de temps jusqu'à défaillance. A notre connaissance, cette hypothèse est vérifiée pour la majorité des systèmes informatiques en vie opérationnelle.

II.4.1. Etablissement du modèle de disponibilité

En utilisant l'interprétation markovienne du modèle hyperexponentiel introduite dans le paragraphe précédent, et en supposant que les intervalles de temps jusqu'à restauration de service sont distribués selon une loi exponentielle de paramètre μ constant, on peut modéliser la croissance de disponibilité d'un système par le modèle présenté sur la figure II.4.



a) fiabilité stabilisée b) croissance de fiabilité

Figure II.4 : Modèle de disponibilité

La figure II.4-a correspond à la chaîne de Markov modélisant un comportement du système en fiabilité stabilisée et la figure II.4-b correspond à la chaîne de Markov prenant en compte le phénomène de croissance de fiabilité. Les paramètres p et q sont tels que $0 \leq p \leq 1$ et $p+q=1$.

Le problème qui se pose est de choisir les paramètres p et q pour prendre en compte l'amélioration progressive de la disponibilité d'un système suite à l'élimination des fautes de conception.

Soit A(t) la disponibilité du système évaluée à l'instant t et $\overline{A}(t) = 1 - A(t)$ l'indisponibilité.

Considérons la chaîne de Markov de la figure II.4-a. On a par définition :

$A(t) = P_{E1}(t) + P_{E2}(t)$ et $\overline{A}(t) = P_D(t)$

Le traitement de la chaîne de Markov par les méthodes classiques [Howard 71] conduit, en supposant que $\mu >> \zeta_{sup}$, à l'expression de l'indisponibilité :

$$\overline{A}(t) = \alpha + \beta \exp\left(-\left(q\ \zeta_{sup} + p\ \zeta_{inf} + o\left(\frac{\zeta_{sup}}{\mu}, \frac{\zeta_{inf}}{\mu}\right)\right)t\right) + \gamma \exp\left(-\left(\mu + o\left(\frac{\zeta_{sup}}{\mu}, \frac{\zeta_{inf}}{\mu}\right)\right)t\right) \qquad \text{(II.12)}$$

avec :

$$\alpha = \frac{\zeta_{sup}\ \zeta_{inf}}{\mu\ (q\ \zeta_{sup} + p\ \zeta_{inf})} + o\left(\frac{\zeta_{sup}}{\mu}, \frac{\zeta_{inf}}{\mu}\right)$$

$$\beta = \frac{(\omega\ \zeta_{sup} + \varpi\ \zeta_{inf})\ (q\ \zeta_{sup} + p\ \zeta_{inf}) - \zeta_{sup}\ \zeta_{inf}}{\mu\ (q\ \zeta_{sup} + p\ \zeta_{inf})} + o\left(\frac{\zeta_{sup}}{\mu}, \frac{\zeta_{inf}}{\mu}\right)$$

$$\gamma = -\ \frac{\omega\ \zeta_{sup} + \overline{\omega}\ \zeta_{inf}}{\mu} + o\left(\frac{\zeta_{sup}}{\mu}, \frac{\zeta_{inf}}{\mu}\right)$$

$o(x)$ est telle que $\lim_{x \to 0} \frac{o(x)}{x} = 0$

L'indisponibilité $\overline{A}(t)$ vérifie les propriétés suivantes :

- $\overline{A}(0) = 0$
- $\overline{A}(t)$ est d'abord croissante puis décroissante à partir de l'instant t_{max} donné par :

$$t_{max} = \frac{1}{\mu} \operatorname{Ln}\left\{\frac{(\omega\ \zeta_{sup} + \varpi\ \zeta_{inf})\ \mu}{(\omega\ \zeta_{sup} + \varpi\ \zeta_{inf})(q\ \zeta_{sup} + p\ \zeta_{inf}) - \zeta_{sup}\ \zeta_{inf}}\right\} + o\left(\frac{\zeta_{sup}}{\mu}, \frac{\zeta_{inf}}{\mu}\right) \qquad \text{(II.13)}$$

t_{max} est de l'ordre de quelques $\frac{1}{\mu}$; compte tenu du fait que μ est généralement très grand, la valeur maximale de l'indisponibilité est atteinte très rapidement.

La valeur de l'indisponibilité maximale évaluée à l'instant t_{max} est donnée par :

$$\overline{A}_{max} = \frac{\omega\ \zeta_{sup} + \varpi\ \zeta_{inf}}{\mu} + o\left(\frac{\zeta_{sup}}{\mu}, \frac{\zeta_{inf}}{\mu}\right) \qquad \text{(II.14)}$$

Or, $\omega\ \zeta_{sup} + \varpi\ \zeta_{inf} = h(0)$ qui est la valeur maximale de l'intensité de défaillance.

On remarque alors que le modèle considéré pour l'évaluation de la disponibilité est cohérent avec le modèle hyperexponentiel pour l'évaluation de la fiabilité puisque l'indisponibilité est maximale quand l'intensité de défaillance est maximale.

Considérons maintenant la valeur asymptotique $\overline{A}_\infty$ de l'indisponibilité $\overline{A}(t)$:

$$\overline{A}_\infty = \alpha \qquad \text{(II.15)}$$

Le modèle hyperexponentiel est basé sur l'hypothèse que le logiciel tend asymptotiquement vers un comportement en fiabilité stabilisée. En d'autres termes, un tel comportement peut être modélisé par la chaîne de Markov de la figure II.4-a où le taux de défaillance du système considéré est égal à $h(\infty) = \zeta_{inf}$. L'indisponibilité évaluée à partir de cette chaîne de Markov tend rapidement vers $\frac{\zeta_{inf}}{\mu}$.

Par conséquent, le modèle de disponibilité proposé doit satisfaire la condition suivante :

$$\overline{A}_\infty = \frac{h(\infty)}{\mu} = \frac{\zeta_{inf}}{\mu} = \alpha \qquad \text{(II.16)}$$

La résolution de l'équation (II-16) conduit aux deux solutions suivantes :

. q=1 et p=0 (II.17)

. $\zeta_{sup} = \zeta_{inf}$ (II.18)

La deuxième solution est à rejeter puisque elle traduit un comportement du logiciel en fiabilité stabilisée au cours de son cycle de vie ce qui est contraire à l'hypothèse de croissance de fiabilité.

Le modèle de croissance de disponibilité ainsi obtenu est présenté sur la figure (II.5).

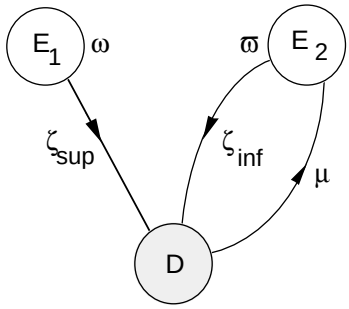


Figure II.5 : Modèle de croissance de disponibilité

L'expression de l'indisponibilité associée à ce modèle, en supposant $\mu >> \zeta_{sup}$, est la suivante :

$$\overline{A}(t) \approx \frac{\zeta_{inf}}{\mu} + \frac{\omega(\zeta_{sup}-\zeta_{inf})}{\mu} e^{-\zeta_{sup}t} - \frac{\omega\zeta_{sup}+\varpi\,\zeta_{inf}}{\mu} e^{-\mu t} \qquad \text{(II.19)}$$

La figure II.6 (courbe *a*) donne une allure typique de l'évolution de l'indisponibilité correspondant à la relation II.19. La courbe *b* (resp. la courbe *c*) de la figure II.6 correspond à la courbe indisponibilité obtenue en supposant un comportement en fiabilité stabilisée du système caractérisé par un taux de restauration μ constant et un taux de défaillance λ constant

égal à la valeur maximale (resp. minimale) de l'intensité de défaillance associée au modèle hyperexponentiel.

En comparant la courbe *a* aux courbes *b* et *c*, on peut constater l'écart qui existe entre ces différentes courbes. Un tel écart traduit les erreurs d'estimation effectuées dans le cas où on évalue la disponibilité d'un système sans tenir compte du phénomène de croissance de fiabilité.

L'hypothèse d'évolution stabilisée de la disponibilité, couramment considérée dans les modèles d'évaluation de la disponibilité des systèmes informatiques, ne correspond en réalité qu'au régime asymptotique du système. Les données collectées sur des systèmes réels montrent que la disponibilité peut varier d'une manière significative au cours du cycle de vie d'un système informatique avant d'atteindre un comportement stabilisé. On peut citer par exemple, le système d'exploitation RTR [Wallace 84] pour lequel on a observé une croissance de disponibilité de l'ordre 250 au cours de trois ans et demi d'exploitation.

Par conséquent, pour effectuer des estimations réalistes de la disponibilité, il est important de modéliser l'évolution de la disponibilité des systèmes en tenant compte du phénomène de croissance de fiabilité.

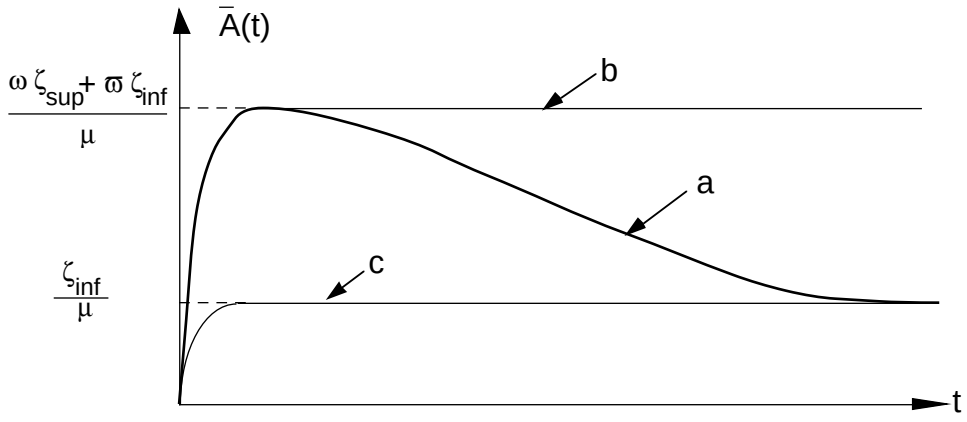


Figure II.6 - Allure d'évolution de l'indisponibilité $\overline{A}(t)$

Les courbes d'indisponibilité présentées sur la figure II.6 ont une allure similaire aux courbes d'indisponibilité obtenues à partir des modèles de connaissance présentés dans [Costes 78] qui sont basés sur une représentation semi-markovienne du comportement d'un système, des modèles présentés dans [Laprie 84-a] basés sur une formulation markovienne et des modèles présentés dans [Kanoun 89, Laprie 91-a] qui sont basés sur les processus de renouvellement généralisés.

Outre l'évaluation de l'indisponibilité instantanée d'un système, on peut évaluer aussi son indisponibilité moyenne sur un intervalle donnée [0,t], c'est-à-dire la proportion de temps au cours duquel le système considéré n'est pas opérationnel. Cette mesure est intéressante car elle peut être évaluée à partir de l'observation du système.

L'indisponibilité moyenne sur [0,t], notée $\overline{A}_m(t)$, est définie par l'expression [Barlow 75] :

$$\overline{A}_m(t) = \frac{1}{t}\int_0^t \overline{A}_m(\tau)\,d\tau \qquad \text{(II.20)}$$

En utilisant la relation (II.19), on obtient pour le modèle hyperexponentiel :

$$\overline{A}_m(t) = \frac{\zeta_{inf}}{\mu} + \frac{\omega(\zeta_{sup}-\zeta_{inf})}{\mu}\ \frac{1-e^{-\zeta_{sup}t}}{\zeta_{sup}\,t} - \frac{\omega\,\zeta_{sup}+\varpi\,\zeta_{inf}}{\mu}\ \frac{1-e^{-\mu t}}{\mu\,t} \qquad \text{(II.21)}$$

REMARQUE

Pour éviter toute confusion entre l'approche de transformation que nous venons de définir pour modéliser le phénomène de croissance de fiabilité et l'approche classique de transformation de chaînes de Markov qui est basée sur l'utilisation de la méthode des états fictifs [Laprie 75], il nous semble important de préciser les différences entre ces deux approches. La différence fondamentale réside dans le fait que dans le cas de la méthode des états fictifs, on simule une fonction de distribution caractérisée par un taux de défaillance qui décroît de façon continue dans le temps, par contre dans notre cas, nous proposons un artifice mathématique qui permet de modéliser un processus de défaillance qui est caractérisé par une fonction intensité de défaillance qui décroît dans le temps selon une allure qui est celle du modèle hyperexponentiel. Le modèle qu'on aurait obtenu en appliquant la méthode des états fictifs au modèle de la figure II.4-a est celui de la figure II.7-b :

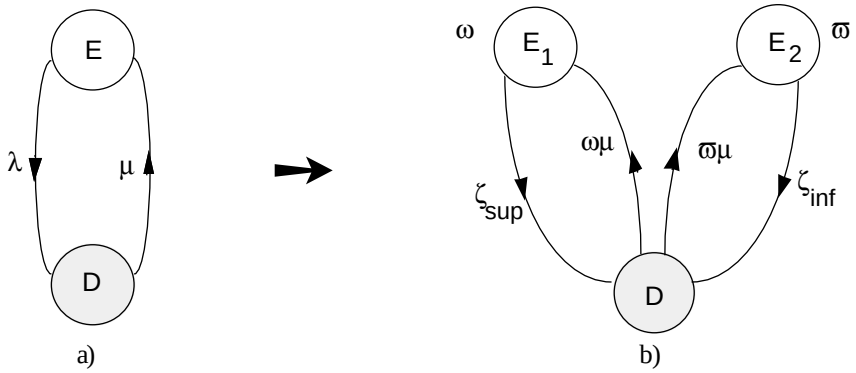


Figure II.7 : Transformation de modèle par la méthode des états fictifs

En comparant la chaîne de Markov de la figure II.7-b à celle qui est présentée sur la figure II.5, on peut remarquer que la différence entre les deux approches de transformation se traduit par des taux de transition de l'état D vers les états E_1 et E_2 qui sont différents : le retour est dissymétrique dans le cas de l'approche de transformation qui est proposée dans ce chapitre.

II.4.2. Variation de la disponibilité en fonction des paramètres du modèle

Etant donné l'expression de l'indisponibilité donnée par la relation II.19, on se propose d'étudier la variation de la croissance de disponibilité d'un système en fonction des paramètres du modèle hyperexponentiel.

On appelle facteur de croissance, qu'on notera k, le rapport entre l'intensité de défaillance maximale et l'intensité de défaillance asymptotique :

$$k = \frac{h(0)}{h(\infty)} = \frac{\omega\,\zeta_{sup} + \varpi\,\zeta_{inf}}{\zeta_{inf}} \qquad \text{(II.22)}$$

En fonction de k, la relation II-19 peut s'écrire sous la forme :

$$\overline{A}(t) = \overline{A}(\infty) \left\{ 1 + (k - 1)\, e^{-\zeta_{sup} t} - k\, e^{-\mu t} \right\} \qquad \text{(II.23)}$$

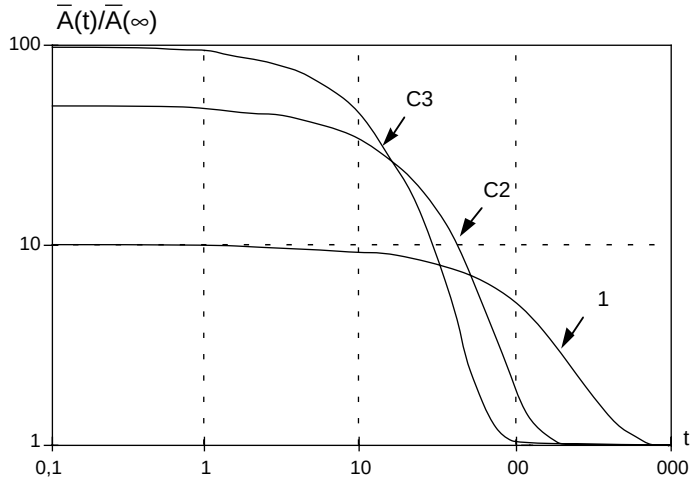
$$\overline{A}(t) \approx \overline{A}(\infty) \left\{ 1 + (k - 1)\, e^{-\frac{(k - 1)h(\infty)}{\omega} t} \right\} \qquad \text{(II.24)}$$

En particulier, pour t = 0, on a :

$$\overline{A}(0) = k\, \overline{A}(\infty) \qquad \text{(II.25)}$$

On vérifie ainsi que l'on obtient le même facteur de croissance pour l'intensité de défaillance et pour l'indisponibilité.

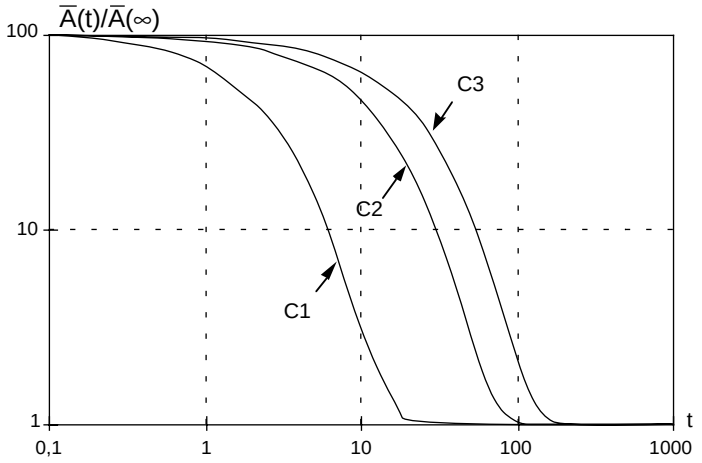
Il est intéressant d'étudier maintenant l'influence de k et de ω sur l'évolution de l'indisponibilité. Sur la figure II.8, on présente des exemples d'évolution de la fonction $\overline{A}(t)/\overline{A}(\infty)$ pour différentes valeurs de k ; toutes les courbes présentées sont tracées pour les mêmes valeurs de ω et h(∞).



Courbe	k
C1	10
C2	50
C3	100
ω = 0,5	$\zeta_{inf} = 410^{-4}$

Figure II.8 : Influence de k sur l'indisponibilité (échelle log-log)

La figure II.9 permet d'illustrer, l'influence de la variation de ω sur l'évolution de $\overline{A}(t)/\overline{A}(\infty)$ dans le cas où k et h(∞) sont fixés ; considérer différentes valeurs de ω revient à considérer différentes allures d'évolution du comportement du logiciel vers le comportement stabilisé.



Courbe	ω
C1	0,1
C2	0,5
C3	0,9
k = 100	$\zeta_{inf} = 410^{-4}$

Figure II.9 : Influence de ω sur l'indisponibilité (échelle log-log)

On peut remarquer que k et ω agissent de façon inverse sur la vitesse d'évolution de l'indisponibilité. En effet, plus k est grand, plus $\overline{A}(t)$ tend rapidement vers $\overline{A}(\infty)$ et plus ω est petit, plus la vitesse d'évolution vers le comportement asymptotique est rapide. On peut vérifier ainsi que la disponibilité évaluée à partir du modèle hyperexponentiel a une allure similaire à celle de l'intensité de défaillance. Ceci était prévisible, puisque compte tenu du fait que $e^{-\mu t}$ tend beaucoup plus vite vers zéro que $e^{-\zeta_{sup} t}$, le taux de restauration μ a peu d'influence sur l'évolution de l'indisponibilité dans le temps (voir relation II.23).

II.4.3. Conclusion

Dans ce paragraphe, nous nous sommes intéressé à la modélisation, par le modèle hyperexponentiel, de la croissance de disponibilité des systèmes. Le modèle établi est basé sur l'interprétation markovienne du modèle qui nous a permis d'utiliser les techniques markoviennes classiques pour évaluer la disponibilité.

L'analyse de la variation de la disponibilité en fonction des paramètres du modèle montre que le modèle de disponibilité est cohérent avec le modèle de fiabilité. Les deux modèles représentent le même type de comportement : une amélioration progressive du système suivie d'un comportement asymptotique stabilisé, et l'allure d'évolution du système vers le régime stabilisé est similaire dans les deux cas. On modélise ainsi la croissance de disponibilité d'un système qui résulte de sa croissance de fiabilité et non la croissance de disponibilité qui est due à la décroissance stochastique des intervalles de temps jusqu'à restauration (phénomène d'apprentissage) ou bien de l'effet conjugué des deux.

II.5. GéNéRALISATION DU MODÈLE

Le modèle hyperexponentiel est basé sur une loi de Cox-hyperexponentielle à deux étages. On peut généraliser le modèle en considérant une loi de Cox à plusieurs étages, en d'autres termes en introduisant des paramètres supplémentaires dans le modèle.

L'intensité de défaillance du modèle généralisé s'exprime sous la forme suivante :

$$h(t) = \frac{\sum_{i=1}^{k} \omega_i \zeta_i e^{-\zeta_i t}}{\sum_{i=1}^{k} \omega_i e^{-\zeta_i t}} \quad \text{avec } 0 \le \omega_i \text{ et } \sum_{i=1}^{k} \omega_i = 1 \qquad \text{(II.26)}$$

L'introduction de paramètres supplémentaires dans un modèle permet d'améliorer son aptitude à représenter le comportement du processus modélisé.

Bien que la généralisation du modèle hyperexponentiel ait déjà été évoquée dans [Laprie 84-a, Kanoun 89], les propriétés de ce modèle n'ont jamais été analysées. Dans le paragraphe suivant, nous analysons les propriétés du modèle hyperexponentiel à 3 étages (donc à 5 paramètres) ; ce modèle sera appelé modèle hyperexponentiel généralisé. Nous comparerons le modèle hyperexponentiel généralisé au modèle hyperexponentiel présenté dans les paragraphes précédents en considérant les deux mesures de la sûreté de fonctionnement : la fiabilité et la disponibilité.

II.5.1. Mesures de Fiabilité

L'intensité de défaillance qui caractérise le modèle hyperexponentiel généralisé est donnée par :

$$h_g(t) = \frac{\sum_{i=1}^{3} \omega_i \zeta_i e^{-\zeta_i t}}{\sum_{i=1}^{3} \omega_i e^{-\zeta_i t}} \qquad \text{avec } 0 \le \omega_i, \sum_{i=1}^{3} \omega_i = 1 \text{ et } \zeta_1 \ge \zeta_2 \ge \zeta_3 \qquad \text{(II.27)}$$

Etant donné que $\sum_{i=1}^{3} \omega_i = 1$, l'intensité de défaillance du modèle est entièrement déterminée par la donnée des 5 paramètres ω_1, ω_2 , ζ_1, ζ_2, et ζ_3.

L'intensité de défaillance $h_g(t)$ est une fonction continue non croissante dans le temps, variant entre $h_g(0) = \sum_{i=1}^{3} \omega_i \zeta_i$ et $h_g(\infty) = \inf(\zeta_1,\zeta_2,\zeta_3) = \zeta_3$.

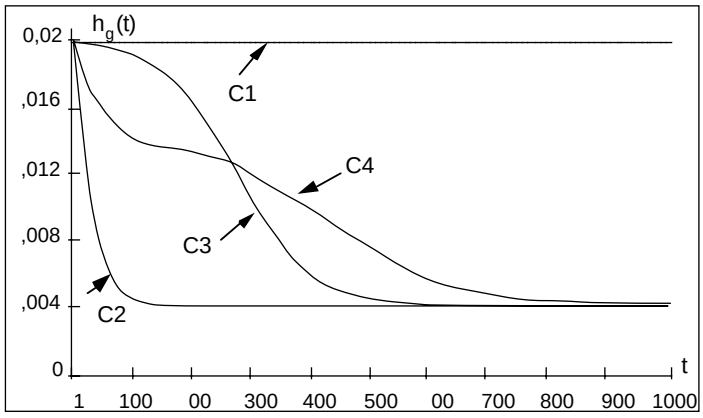
En effet, la dérivée première de $h_g(t)$ est toujours négative. Son expression est donnée par :

$$\frac{dh_g(t)}{dt} = - \frac{\sum_{\substack{i,j=1 \\ i<j}}^{3} \omega_i \omega_j (\zeta_i - \zeta_j)^2 e^{-(\zeta_i + \zeta_j)t}}{\left\{ \sum_{i=1}^{3} \omega_i e^{-\zeta_i t} \right\}^2} \qquad \text{(II.28)}$$

La dérivée seconde de $h_g(t)$ peut changer de signe selon les valeurs des paramètres du modèle comme l'illustre la figure II.10 qui présente différentes courbes caractéristiques de l'évolution de l'intensité de défaillance du modèle hyperexponentiel généralisé. Si on compare les courbes présentées sur la figure II.10 avec celles du modèle hyperexponentiel présentées sur la figure II.2, on remarque que l'introduction de paramètres supplémentaires dans le modèle hyperexponentiel permet de modéliser un nouveau type de comportement (courbe C4) :

deux vitesses d'évolution vers le comportement asymptotique : rapide—lente—rapide—lente

Cette dernière propriété est intéressante car elle permet au modèle généralisé de mieux prendre en compte les variations de la croissance de fiabilité d'un système au cours de son cycle de vie (cf figure I.10).



Courbe	ω_1	ω_2	ζ_1	ζ_2	ζ_3
C1	1,00	-	0,020	0,020	0,004
C2	0,50	0,400	0,021	0,022	0,004
C3	0,97	0,002	0,020	0,018	0,004
C4	0,23	0,760	0,041	0,014	0,004

Figure II.10 : Courbes intensité de défaillance du modèle hyperexponentiel généralisé

Considérons maintenant les mesures de fiabilité associées au modèle hyperexponentiel généralisé.

Posons :

$$G_g(t) = \sum_{i=1}^{3} \omega_i \, e^{-\zeta_i t} \qquad \text{(II.29)}$$

L'intensité de défaillance relative au modèle s'écrit alors :

$$h_g(t) = -\frac{G_g'(t)}{G_g(t)} \quad \text{avec} \quad G_g'(t) = \frac{dG_g(t)}{dt} \qquad \text{(II.30)}$$

Les mesures de fiabilité $R_i(t'|s)$, $f_i(t'|s)$, $\lambda_i(t'|s)$ caractérisant le comportement d'un système entre l'occurrence des défaillances i-1 et i et évaluées à partir du modèle hyperexponentiel généralisé ont des expressions équivalentes à celles obtenues à partir du modèle hyperexponentiel. Elles s'obtiennent directement à partir des relations II.7 à II.10 en remplaçant $G(t)$ par $G_g(t)$.

Le temps moyen jusqu'à occurrence de la défaillance i est donné par :

$$MTTF_i = \frac{\sum_{i=1}^{3} \frac{\omega_i}{\zeta_i} e^{-\zeta_i s}}{\sum_{i=1}^{3} \omega_i e^{-\zeta_i s}} \qquad \text{(II.31)}$$

II.5.2. Disponibilité

D'une façon analogue au modèle hyperexponentiel, on peut représenter le modèle hyperexponentiel généralisé par une chaîne de Markov. En appliquant la même démarche que celle considérée dans le paragraphe II.4.1, on aboutit à la chaîne de Markov présentée sur la figure II.11 permettant de modéliser une croissance de disponibilité.

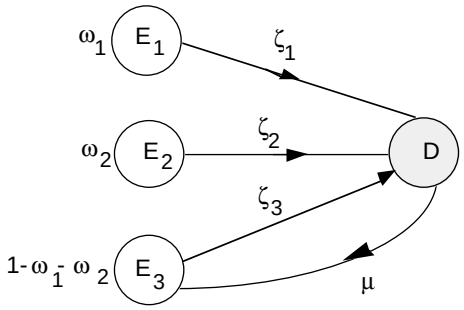


Figure II.11 : Modèle de disponibilité relatif au modèle hyperexponentiel généralisé

Sous l'hypothèse $\mu >> (\zeta_1,\zeta_2,\zeta_3)$, la résolution de la chaîne de Markov conduit à l'expression suivante de l'indisponibilité :

$$\overline{A}_g(t) \approx \frac{\zeta_3}{\mu} - \frac{\omega_1(\zeta_1-\zeta_3)}{\mu} e^{-\zeta_1 t} - \frac{\omega_2(\zeta_2-\zeta_3)}{\mu} e^{-\zeta_2 t} + \frac{\omega_1\zeta_1+\omega_2\zeta_2+\omega_3\zeta_3}{\mu} e^{-\mu t} \qquad \text{(II.32)}$$

La relation II.32 peut s'écrire en fonction du facteur de croissance $k_g = \frac{h_g(0)}{h_g(\infty)}$ sous la forme :

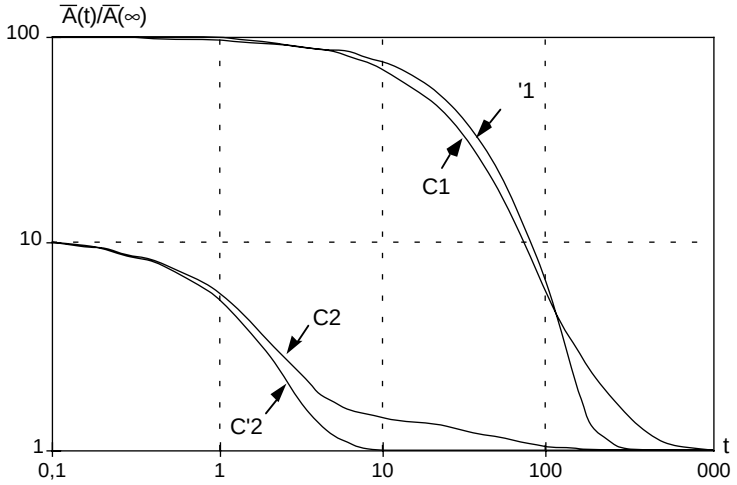
$$\overline{A}_g(t) \approx \overline{A}_g(\infty) \left\{1 + (k_g - 1) e^{-\zeta_1 t} + \omega_2 \left(\frac{\zeta_2}{h_g(\infty)} - 1\right) \left(e^{-\zeta_2 t} - e^{-\zeta_1 t} \right)\right\} \qquad \text{(II.33)}$$

Si on pose $\omega_2 = 0$, $\zeta_1 = \zeta_{sup}$ et $\zeta_3 = \zeta_{inf}$, on retrouve l'expression de l'indisponibilité associée au modèle hyperexponentiel à trois paramètres (relation II.24).

La relation II.33 permet ainsi de comparer le modèle hyperexponentiel généralisé au modèle hyperexponentiel à trois paramètres. Dans le cas du modèle à trois paramètres, la vitesse d'évolution de l'indisponibilité vers le comportement asymptotique est déterminée par ζ_{sup} ; le régime asymptotique est atteint au bout de quelques $1/\zeta_{sup}$. Par contre, dans le cas du modèle généralisé, le comportement asymptotique est atteint beaucoup plus tard : au bout de quelques $1/\zeta_2$, $(\zeta_2 \leq (\zeta_1,\zeta_{sup}))$.

La figure II.12 permet d'illustrer les différences qui existent entre les courbes indisponibilité relatives au modèle hyperexponentiel à trois paramètres et au modèle hyperexponentiel généralisé. Les courbes présentées sont tracées en considérant le même facteur de croissance :

$$k = \frac{h(0)}{h(\infty)} = \frac{h_g(0)}{h_g(\infty)} = k_g$$



Courbe	k	ω_1	ω_2	ζ_1	ζ_2	ζ_3
1	100	0,699	0,25	0,0398	,009	0,0003
2	10	0,178	0,63	0,7980	,029	0,0152

Courbe		ω	ζ_{up}	ζ_{inf}
C'1	100	0,977	,03	0,0003
C'2	10	0,180	,85	0,0152

Figure II.12 : Courbes d'indisponibilité : comparaison du modèle hyperexponentiel et du modèle hyperexponentiel généralisé (échelle log-log)

Pour les deux valeurs considérées du facteur de croissance : k = 10 et k = 100, on peut remarquer que les courbes C'i, correspondant au modèle hyperexponentiel, et Ci, correspondant au modèle généralisé, sont pratiquement confondues sauf au voisinage de l'asymptote où les courbes associées au modèle hyperexponentiel généralisé se stabilisent moins vite que celles du modèle hyperexponentiel.

Par ailleurs, on peut remarquer que contrairement au modèle hyperexponentiel, la courbe $\overline{A}_g(t)$ peut changer de concavité au cours de son évolution entre le moment où le maximum est atteint et le moment où la courbe tend vers sa valeur asymptotique (courbe C2).

II.5.3. Conclusion

Le modèle hyperexponentiel généralisé permet d'améliorer les dynamiques des courbes intensité de défaillance et indisponibilité relatives au modèle hyperexponentiel. En particulier, concernant l'intensité de défaillance, le modèle généralisé permet de modéliser deux périodes successives de croissance de fiabilité rapide séparées par deux périodes au cours desquelles la croissance de fiabilité est lente. Une telle propriété est intéressante car elle permet au modèle de mieux prendre en compte les variations de fiabilité et de disponibilité caractéristiques du comportement d'un système au cours de son cycle de vie.

Néanmoins, l'application du modèle généralisé nécessite l'estimation de cinq paramètres ; par conséquent, comparé au modèle hyperexponentiel, les procédures d'inférence associées sont plus difficiles à mettre en œuvre.

II.6. APPLICATION DU MODÈLE HYPEREXPONENTIEL à DES SYSTÈMES RéELS

Dans les paragraphes précédents, nous avons présenté le modèle hyperexponentiel en considérant les deux mesures de base de la sûreté de fonctionnement : la fiabilité et la disponibilité. Afin d'illustrer l'utilisation du modèle dans un contexte industriel, nous présentons, des résultats concernant l'application du modèle pour l'évaluation de la sûreté de fonctionnement des logiciels de deux systèmes de télécommunications. Pour le premier système, le modèle hyperexponentiel est utilisé pour suivre la validation du logiciel et évaluer ses mesures de fiabilité avant sa mise en opération. Pour le deuxième système, le modèle est appliqué pour évaluer la disponibilité moyenne du logiciel au cours de sa vie opérationnelle.

II.6.1. évaluation de la fiabilité d'un logiciel de télécommunication

Le logiciel étudié fait partie d'un autocommutateur téléphonique [Kaâniche 90-a]. Il est constitué de plusieurs modules qui sont structurés selon une architecture répartie et qui communiquent entre eux par échange de messages. Chaque module est destiné à accomplir une fonction élémentaire du logiciel pouvant être spécifiée, codée et testée séparément. Trois langages ont été utilisés pour le développement du logiciel : PLM 86, Assembleur et un langage spécifique : LDMD (Langage de Description et de Manipulations de Données). Le volume total du logiciel est de l'ordre de 1400 kilo-lignes d'instructions.

Le développement du logiciel a duré 4 ans dont 17 mois ont été consacrés à l'application de tests unitaires, de test fonctionnels et de tests d'intégration pour la validation du logiciel avant sa mise en opération. Au cours de cette période, une procédure de collecte de données a été mise en place pour enregistrer les défaillances observées sur le logiciel. En tout, 2146 relevés de défaillance ont été collectés ; ces relevés nous ont été fournis par le Centre National de Télécommunication de LANNION pour analyser et évaluer la fiabilité du logiciel au cours de la période considérée.

L'étude à été divisée en deux étapes conformément à l'approche globale présentée dans le chapitre I. La première étape a été consacrée à l'analyse des relevés de défaillance pour vérifier la consistance des données fournies et extraire les informations qui sont utiles pour l'étude de fiabilité. La deuxième étape a porté sur l'application des modèles de croissance de fiabilité pour le suivi de la validation du logiciel et l'évaluation de sa fiabilité avant sa mise en opération.

L'étude complète est présentée dans [Kaâniche 89-a, Kaâniche 90-a, Kaâniche 90-b]. Dans la suite, nous nous limiterons à la présentation de quelques résultats qui ont été obtenus dans le cadre de cette étude en mettant l'accent plus particulièrement sur l'application du modèle hyperexponentiel.

II.6.1.1. Analyse des relevés de défaillance

Les relevés de défaillance (RD) correspondent à des fiches qui sont rédigées soit pour signaler des défaillances observées au cours de l'exécution du logiciel soit pour suggérer des corrections ou bien des améliorations de certaines fonctionnalités du logiciel. Ils constituent en fait un moyen de communication entre toutes les personnes qui sont concernées par le système.

Un RD permet de préciser la date de sa création (cette date ne correspond pas toujours à la date d'occurrence de la défaillance signalée), de décrire les conditions d'occurrence de la défaillance, d'indiquer les différentes fonctions affectées et d'estimer la gravité de la défaillance en fonction des conséquences observées sur le service délivré par le logiciel.

Afin d'extraire les informations qui sont utiles pour une étude de fiabilité, nous avons procédé à une analyse du contenu des RD. Cette étape s'est terminée par l'élimination de 45% des RD. La plupart des RD qui ont été éliminés contiennent des informations redondantes : par exemple, plusieurs relevés font référence à la même défaillance du logiciel. Les critères considérés pour le filtrage des relevés de défaillance ainsi que les analyses statistiques effectuées sont détaillés dans [Kaâniche 89-a, Kaâniche 90-a]. On peut noter qu'un pourcentage de RD éliminés par rapport aux RD collectés du même ordre de grandeur a été observé sur un autre système [Levendel 89]. L'élimination d'un nombre important de RD collectées nous a amené à proposer dans [Kaâniche 89-b] des recommandations pour améliorer la procédure de collecte de données afin d'éviter dans le futur de collecter des données qui ne sont d'aucun intérêt pour les études de fiabilité et par suite limiter les coûts de la collecte de données et simplifier la tâche de tri des relevés de défaillances.

1172 relevés ont ainsi été retenus pour l'étude de fiabilité dont 174 uniquement correspondent à la période au cours de laquelle le logiciel a été intégré dans sa totalité. Les autres relevés sont relatifs aux tests fonctionnels effectués sur le logiciel. Parmi les 174 RD qui ont été collectés au cours de l'intégration du logiciel, 42 seulement correspondent à des défaillances critiques qui auraient pu affecter de façon significative le service délivré aux utilisateurs si elles étaient apparues en vie opérationnelle.

Les résultats que nous présentons dans la suite concernent uniquement les données collectées après l'intégration du logiciel. Pour caractériser le comportement du logiciel au cours de la période considérée, nous nous intéresserons dans un premier temps, à toutes les données toutes conséquences confondues, puis nous considérerons uniquement les défaillances critiques.

Notre objectif étant d'illustrer différents aspects du modèle hyperexponentiel, nous travaillerons, d'abord sur des données groupées en nombre de défaillances par unité de temps en considérant toutes les données collectées et nous appliquerons le modèle pour suivre l'évolution du nombre cumulé de défaillances. Nous considérerons ensuite uniquement les défaillances critiques qui seront présentées sous forme d'intervalles de temps entre défaillances et nous appliquerons le modèle pour suivre l'évolution du MTTF du logiciel avant sa mise en opération.

Conformément à l'approche globale d'évaluation de la fiabilité du logiciel que nous avons présentée dans le chapitre I, nous appliquerons d'abord le test de Laplace pour analyser la variation de la fiabilité du logiciel au cours de la période considérée puis nous appliquerons le modèle hyperexponentiel en nous basant sur les résultats obtenus à partir de l'analyse de tendance.

II.6.1.2. Analyse de tendance

II.6.1.2.1. Données groupées en "nombre de défaillances par unité de temps"

La courbe de Laplace calculée à partir des données collectées est présentée sur la figure II.13.

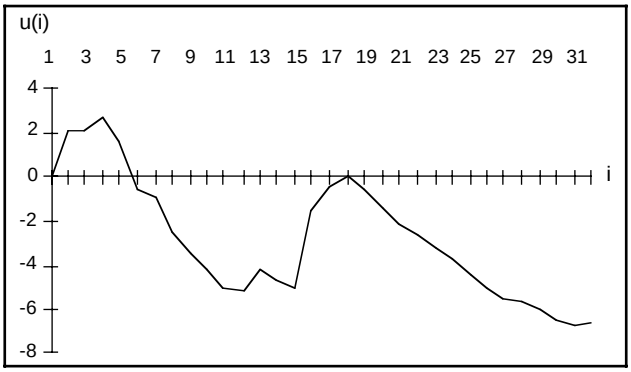


Figure II.13 : Application du test de Laplace aux données de la figure II.13

On remarque que pendant les quatre premières semaines de l'intégration du logiciel, le facteur de Laplace est positif et tend à croître. On observe, ainsi une décroissance globale de fiabilité qui est due à l'augmentation progressive de l'intensité de défaillance du logiciel pendant cette période. Un tel comportement du logiciel est classique, le passage des tests fonctionnels vers les tests d'intégration se traduit par une croissance locale de l'intensité de défaillance qui est due à l'activation des fautes localisées au niveau de l'interfaçage entre les différents composants du logiciel.

A partir de la quatrième semaine, le facteur de Laplace décroît progressivement traduisant ainsi une croissance presque monotone de la fiabilité du logiciel. Une telle croissance se poursuit jusqu'à la semaine 16. A cette date, on peut observer une forte tendance du facteur de Laplace à la croissance qui révèle une tendance de décroissance locale de la fiabilité du logiciel. Ce comportement est dû à la découverte d'un nombre important de fautes dans les manuels d'exploitation du logiciel. Cette décroissance de fiabilité est ensuite progressivement atténuée et on peut remarquer que le facteur de Laplace tend ensuite à décroître à nouveau jusqu'à la fin de l'intégration du logiciel traduisant une croissance progressive et pratiquement monotone de la fiabilité du logiciel avant sa mise en opération.

II.6.1.2.2. Données "intervalles de temps entre défaillances"

Les résultats de l'application du test de Laplace aux données relatives aux défaillances critiques sont présentés sur la figure II.14.

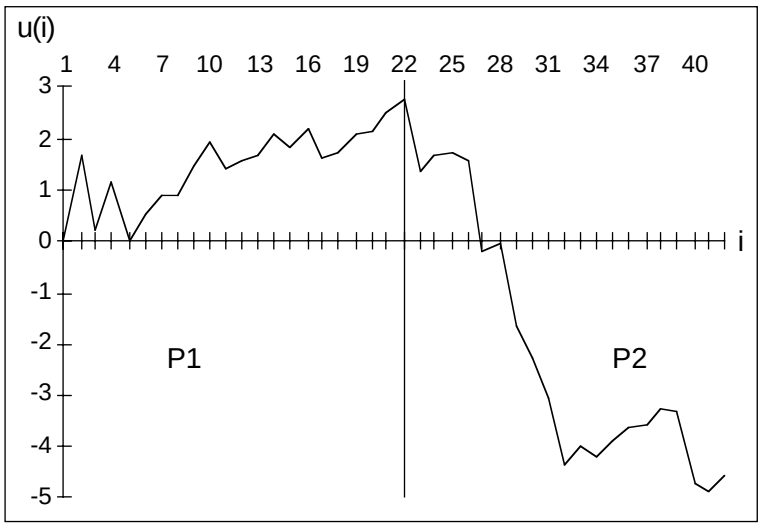


Figure II.14 : Application du test de Laplace aux "intervalles de temps entre défaillances"

On distingue deux périodes de croissance de fiabilité du logiciel : la première, notée **P1**, correspond aux 21 premières défaillances, et la deuxième, notée **P2**, est relative aux 21 dernières défaillances.

Au cours de la première période, on observe une légère décroissance de fiabilité : le facteur de Laplace est positif et tend à croître. Au cours de la deuxième période, le facteur de Laplace tend à décroître traduisant ainsi une croissance progressive de la fiabilité du logiciel. Cependant, une telle croissance n'est pas monotone et on peut observer en particulier une décroissance locale de fiabilité suite à l'occurrence de la défaillance 33.

II.6.1.3. Application du modèle hyperexponentiel

Nous appliquerons d'abord le modèle aux données groupées en nombre de défaillances par semaine pour suivre la validation du logiciel par l'évaluation de l'évolution du nombre cumulé de défaillances. Ceci permet aux concepteurs du logiciel, au vu des estimations effectuées par le modèle, de planifier l'effort de test nécessaire pour corriger les fautes qui seront activées sur le logiciel pendant les périodes suivantes. Ensuite, le modèle sera appliqué aux données "intervalles de temps entre défaillances" pour évaluer le MTTF du logiciel et estimer son taux de défaillance vers la fin de la validation. Ceci permet d'évaluer la fiabilité du logiciel avant sa mise en opération.

II.6.1.3.1. évaluation du nombre cumulé de défaillances

On présente sur la figure II.15, d'une part, l'évolution du nombre cumulé de défaillances N(i) observées sur le logiciel, et d'autre part, l'évolution du nombre cumulé de défaillances H(i) estimé par le modèle hyperexponentiel. Compte tenu des résultats du test de Laplace présentés sur la figure II.13 qui montrent que le logiciel ne manifeste une tendance de croissance de fiabilité qu'à partir de la quatrième semaine, nous n'avons appliqué le modèle qu'à partir de cette date. Les paramètres du modèle sont d'abord estimés à partir des données collectées durant le deuxième mois de l'intégration du logiciel, le modèle est ensuite recalibré toutes les deux semaines pour prendre en compte progressivement les données collectées au fur et à mesure de la validation du logiciel.

On remarque que le modèle donne de bonnes prévisions jusqu'à la semaine 16. A partir de cette date, les estimations effectuées par le modèle s'écartent de manière significative des valeurs observées. Un tel comportement est dû à la forte décroissance locale de fiabilité observée à cette date et détectée par le test de Laplace.

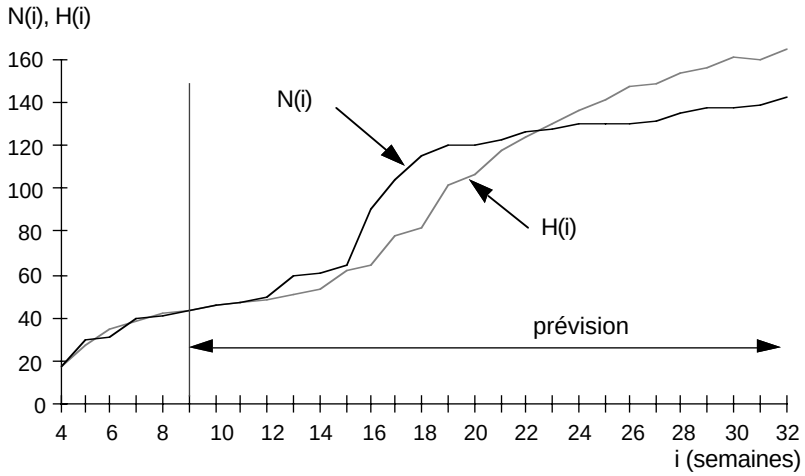


Figure II.15 : Estimation du nombre cumulé de défaillances par le modèle

Les modèles de croissance de fiabilité ne peuvent pas modéliser un tel changement de tendance. Pour améliorer les estimations effectuées, on peut appliquer le modèle par période de croissance de fiabilité. On distingue deux périodes de croissance de fiabilité : la première est relative aux données collectées jusqu'à la semaine 16 et la deuxième allant de cette date jusqu'à la fin de l'intégration du logiciel.

Si on applique à nouveau le modèle par période de croissance de fiabilité, on obtient la figure II.16. On observe alors une très bonne adéquation entre les données observées et les prévisions effectuées par le modèle durant les deux derniers mois de la validation.

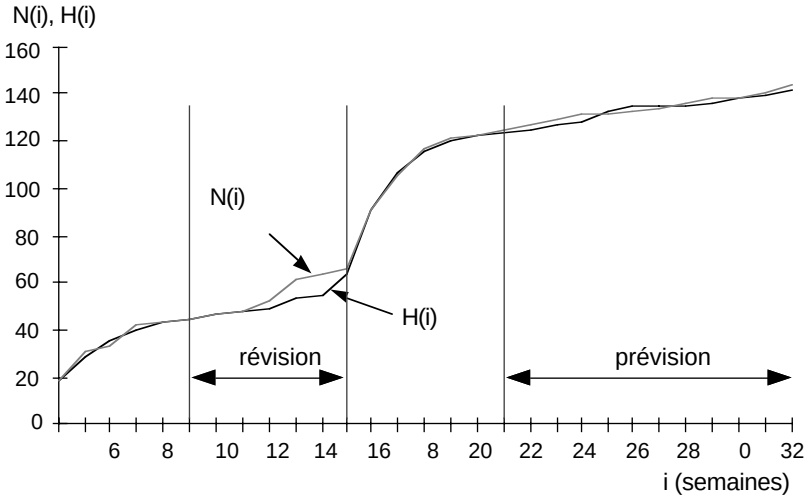


Figure II.16 : Application du modèle par période de croissance de fiabilité

II.6.1.3.2. Comparaison du modèle hyperexponentiel à d'autres modèles

Afin de comparer le modèle hyperexponentiel (HE) à d'autres modèles de croissance de fiabilité, nous avons choisi :

- un modèle en forme de S appartenant à la classe des modèles décrivant une décroissance de fiabilité suivie d'une croissance de fiabilité : le modèle SS [Yamada 83],
- un modèle appartenant à la classe des modèles décrivant une croissance monotone de fiabilité : le modèle exponentiel (EXP) [Goel 79].
- le modèle hyperexponentiel généralisé (HEg).

Nous avons appliqué les quatre modèles durant la période allant de la semaine 16 à la semaine 32. Le critère de comparaison considéré est le résidu moyen : c'est à dire, la valeur moyenne des écarts entre les valeurs observées du nombre cumulé de défaillances et les valeurs estimées par les modèles durant la période considérée.

Nous nous sommes intéressé d'une part, à la capacité réplicative des modèles pour apprécier leur aptitude à reproduire le comportement du logiciel, et d'autre part à leur capacité prévisionnelle pour estimer la qualité des prévisions effectuées. Les résultats sont donnés sur les figures II.17-a et II.17-b.

Pour évaluer la capacité réplicative des différents modèles, nous avons utilisé les données collectées entre les semaines 16 et 32 pour estimer les paramètres des modèles, puis nous avons évalué le résidu moyen sur toute la période considérée.

Pour évaluer la capacité prévisionnelle du modèle, nous avons estimé les paramètres des modèles à partir des données observées entre les semaines 16 et 20. Les modèles ainsi calibrés sont utilisés pour estimer le nombre cumulé de défaillances pour les semaines 21 jusqu'à 32. Sur la figure II.17-b nous donnons les valeurs des résidus moyens dans trois cas de figure :

1- le modèle est calibré une seule fois,
2- le modèle est recalibré toutes les 2 semaines,
3- le modèle est recalibré toutes les 4 semaines.

HE	HEg	SS	EXP
0,9	0,8	8,4	5,4

a) Application des modèles en réplicatif

Calibration des modèles	HE	HEg	SS	EXP
1 seule fois	9,1	1,4	11,1	8,3
toutes les 2 semaines	0,7	0,7	2,4	2,0
toutes les 4 semaines	2,3	1,1	3,7	2,9

b) Application des modèles en prévisionnel

figure II.17 : Valeurs des résidus moyens

L'analyse des deux figures montre que les modèles HE et HEg possèdent une meilleure capacité réplicative que les modèles SS et EXP. En prévisionnel, les résultats de ces deux

modèles sont comparables, le modèle HEg donne cependant en moyenne de meilleures prévisions que les autres modèles. On peut remarquer ainsi, que les résultats du modèle hyperexponentiel généralisé confirment les résultats théoriques présentés dans le paragraphe II.5. L'introduction de paramètres supplémentaires dans le modèle hyperexponentiel permet d'améliorer les estimations effectuées par le modèle. Il faut remarquer, toutefois, que l'application du modèle généralisé nécessite l'estimation de cinq paramètres et que, comparé au modèle hyperexponentiel, les procédures d'inférence associées sont plus difficiles à mettre en œuvre .

II.6.1.3.3. Suivi de la fiabilité du logiciel avant sa mise en opération

Avant la mise en opération du logiciel, il est nécessaire d'évaluer sa fiabilité afin d'évaluer le comportement du logiciel tel qu'il sera perçu par ses utilisateurs en supposant que l'environnement d'utilisation du logiciel au cours de sa vie opérationnelle est équivalent à celui dans lequel le logiciel a été validé.

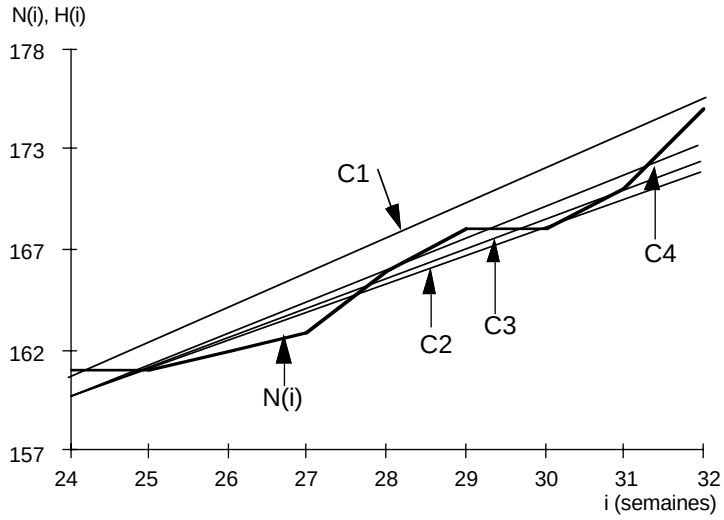
Pour suivre l'évolution du comportement du logiciel avant sa mise en opération, nous avons appliqué progressivement le modèle hyperexponentiel en prévisionnel à partir de la semaine 16 et étudié l'évolution des estimations effectuées par rapport au comportement observé du logiciel.

Les résultats sont résumés sur la figure II.18 où N(i) est le nombre cumulé de défaillances observées jusqu'à la semaine i ; les courbes C1, C2, C3, C4 sont telles que :

- Pour C1, le modèle est calibré avec les données collectées entre les semaines 16 et 24.
- Pour C2, le modèle est calibré avec les données collectées entre les semaines 16 et 26. Quand le modèle est calibré avec les données collectées entre les semaines 16 et 28, on obtient une courbe qui est pratiquement confondue avec C2.
- Pour C3, le modèle est calibré avec les données collectées entre les semaines 16 et 30.
- La courbe C4 correspond à l'application du modèle en réplicatif sur la période considérée. Elle caractérise le comportement du logiciel juste avant sa mise en opération.

Par souci de clarté, seules les parties des courbes correspondant aux semaines 24 à 32 sont présentées.

On peut remarquer que toutes les courbes estimées par le modèle sont des droites qui représentent bien l'évolution du comportement du logiciel durant la période considérée. Ceci traduit une évolution stabilisée de la fiabilité du logiciel vers la fin de sa validation. Le taux de défaillance λ_S du logiciel tend alors à être constant durant cette période, il est donné par la pente de la courbe H(i) estimée par le modèle.



Courbe	λ_s
C1	$1{,}03\ 10^{-2}$ /h
C2	$8{,}20\ 10^{-3}$ /h
C'2	$8{,}15\ 10^{-3}$ /h
C3	$8{,}57\ 10^{-3}$ /h
C4	$9{,}10\ 10^{-3}$ /h

Figure II.18 : Evolution des prévisions effectuées par le modèle vers la fin de la validation du logiciel

Les valeurs des taux de défaillance λ_s relatifs aux différentes courbes estimées par le modèle hyperexponentiel sont également présentées sur la figure II.19. Nous donnons aussi la valeur du taux de défaillance évalué par le modèle quand il est calibré avec les données collectées entre les semaines 16 et 28 (courbe C'2). On peut remarquer, que les valeurs obtenues sont du même ordre de grandeur. Le taux de défaillance relatif à la courbe C1 est plus élevé que les autres valeurs à cause de la faible croissance locale de fiabilité observée au niveau des semaines 25 et 26. Une telle croissance, matérialisée par une variation de la pente de la courbe de Laplace présentée sur la figure II.13, est prise en compte dans les estimations du modèle après la recalibration du modèle à la semaine 26 (courbe C2) : taux de défaillance estimé plus faible.

Les valeurs des taux de défaillance présentées sur la figure II.19 sont relativement élevées si on les compare aux taux de défaillance relatifs à des autocommutateurs téléphoniques évalués dans [Sabourin 87, Kanoun 91-b][1]. Néanmoins, contrairement à ces deux derniers systèmes pour lesquels les taux de défaillance ont été évalués à partir des données collectées au cours de la vie opérationnelle, les données considérées ici concernent uniquement la fin de la phase de validation. Au cours de cette période, le logiciel a été installé sur quatre sites expérimentaux et a subi essentiellement des tests de charge. Par conséquent, le taux de défaillance estimé par le modèle hyperexponentiel est pessimiste comparé au taux de défaillance qui sera observé au cours de la vie opérationnelle du logiciel : celui-ci sera au plus égal à $\frac{\lambda_s}{4}$.

II.6.1.3.4. Evaluation du MTTF du logiciel relatif aux défaillances critiques

Considérons maintenant les données collectées sous forme d'intervalles entre défaillances. Nous appliquerons le modèle hyperexponentiel pour suivre l'évolution du MTTF du logiciel et évaluer son taux de défaillance asymptotique vis-à-vis de l'occurrence de défaillances critiques.

---

1 Dans [Sabourin 87], le taux de défaillance évalué est de l'ordre de $4{,}8\ 10^{-6}$/h pour un parc de 1400 autocommutateurs observés sur 3 ans de vie opérationnelle, et dans [Kanoun 91-b] il est de l'ordre de $1{,}3\ 10^{-4}$/h pour un parc de 15 systèmes observés durant la première année de vie opérationnelle.

Ceci permet de caractériser le comportement du logiciel tel qu'il est perçu par ses utilisateurs en considérant uniquement les défaillances qui affectent de façon significative le service délivré par le logiciel : c'est-à-dire celles qui conduisent à la perte de certaines fonctionnalités du système ou bien à l'interruption totale du système.

Si on applique le modèle hyperexponentiel en prévisionnel pas à pas après l'observation de la 15 ième défaillance sans tenir compte des résultats du test de Laplace présentés sur la figure II.14, on obtient les résultats présentés sur la figure II.19 :

. t_i est l'intervalle de temps observé entre l'occurrence des défaillances i-1 et i,

. $MTTF_i$ est l'intervalle de temps estimé par le modèle,

. λ_s est la valeur du taux de défaillance asymptotique du logiciel évalué vers la fin de la période considérée.

. $R_{16,42}$ est le résidu relatif évalué sur la période de prévision défini par :

$$R_{16,42} = \frac{\sum_{i=16}^{42} (t_i - MTTF_i)}{\sum_{i=16}^{42} t_i}$$

. KS est la distance de Kolmogorov-Smirnof qui permet d'apprécier la qualité des estimations effectuées par le modèle[2].

i	t_i	$MTTF_i$	i	t_i	$MTTF_i$
16	0	1,87	30	9	2,64
17	3	1,75	31	14	2,93
18	1	1,82	32	27	3,42
19	0	1,78	33	1	4,46
20	1	1,68	34	11	4,31
21	0	1,65	35	1	4,59
22	0	1,57	36	2	4,45
23	6	1,50	37	6	4,35
24	0	1,70	38	0	4,41
25	1	1,63	39	7	4,25
26	2	1,60	40	42	4,35
27	10	1,62	41	13	5,65
28	1	2,09	42	2	5,90
29	14	2,03			

$R_{16,42}$ = 50,75 % KS = 0.401 $\lambda_s = 6.410^{-3}$/h

Figure II.19 : Evaluation des MTTF par le modèle hyperexponentiel

Compte tenu des valeurs du résidu relatif et de la distance de KS obtenues, on peut conclure que les estimations effectuées par le modèle s'écartent de façon significative des valeurs observées : le résidu relatif est très grand et le test de KS est significatif au niveau de confiance α = 1%.

---

[2] Pour la définition de la distance de Kolmogorov-Smirnof et son utilisation pour tester l'adéquation des estimations effectuées par un modèle de croissance de fiabilité avec les données observées on peut consulter [Littlewood 89, Kanoun 89].

Par ailleurs, la valeur du taux de défaillance asymptotique évalué par le modèle est de 6,4 10^{-3}/h (0,15 /jour) ce qui correspond à une défaillance en moyenne tous les 6 jours. Cet intervalle est plus faible que celui qui a été effectivement observé : en considérant les quatre dernières données, l'intervalle de temps entre défaillances observées est de l'ordre de 16 jours.

Un tel comportement du modèle s'explique par l'absence de croissance de fiabilité durant le début de la période considérée comme nous l'avons remarqué suite à l'application du test de Laplace. Le modèle hyperexponentiel étant un modèle à croissance de fiabilité monotone, il ne peut pas donner dans de telles conditions de bonnes estimations.

Compte tenu des résultats du test de Laplace, il est préférable de n'appliquer le modèle qu'à partir du moment où le logiciel manifeste une tendance de croissance de fiabilité ; c'est-à-dire durant la période P2 identifiée par le test de Laplace (figure II.14). Les résultats de l'application du modèle durant cette période sont présentés sur la figure II.20.

i	t_i	$MTTF_i$
30		7,80
31	14	8,11
32	27	9,31
33		12,30
34	11	10,67
35		10,67
36		9,62
37		8,85
38		8,59
39		7,87
40	42	7,80
41	13	10,26
42		10,44

$R_{30,42}$ = 9.42 % KS = 0.197 $\lambda_s = 410^{-3}$ /heure

Figure II.20 : Application du modèle à P2

On peut remarquer alors une nette amélioration des estimations du modèle. Le résidu relatif est plus faible et le test de KS est non significatif au niveau de confiance $\alpha = 5\%$.

Le taux de défaillance asymptotique évalué par le modèle est 410^{-3}/h (0,096/jour) soit environ 1 défaillance tous les 11 jours. Cette valeur correspond mieux aux taux de défaillance du logiciel observé vers la fin de la validation.

II.6.1.4. Conclusion

Dans ce paragraphe, nous avons appliqué la méthode d'évaluation de la fiabilité du logiciel introduite dans le chapitre I au logiciel d'un autocommutateur téléphonique. Nous avons mis l'accent plus particulièrement sur différents aspects de l'application du modèle hyperexponentiel pour l'évaluation des mesures de fiabilité. Nous avons appliqué le modèle pour :

. suivre l'évolution du nombre de défaillances au cours de la validation du logiciel,

. évaluer le taux de défaillance asymptotique du logiciel avant sa mise en opération,

. estimer l'évolution du MTTF quand les données collectées sont sous forme d'intervalles de temps entre défaillances.

Par ailleurs, nous avons montré qu'une analyse de tendance basée sur le test de Laplace peut être très utile pour guider l'application des modèles de croissance de fiabilité et améliorer les estimations effectuées.

Nous avons aussi remarqué que le modèle hyperexponentiel et le modèle hyperexponentiel généralisé donnent des estimations satisfaisantes qui sont comparables sinon meilleures que les estimations effectuées par d'autres modèles de croissance de fiabilité. Ces résultats concernent uniquement l'application considérée et ne peuvent être généralisés dans l'absolu car il n'existe pas de modèle universel de croissance de fiabilité qui donne de bonnes estimations dans tous les cas de figure : les performances d'un modèle de croissance de fiabilité peuvent varier d'un ensemble de données à un autre [Keiller 91]. Toutefois, l'utilisation des résultats des tests de tendance permet d'améliorer l'aptitude des modèles à fournir des estimations qui sont représentatives du comportement des systèmes.

II.6.2. évaluation de la disponibilité du logiciel de l'autocommutateur TROPICO-R 4096

Les données collectées sur le logiciel de télécommunication étudié dans le paragraphe précédent concernent uniquement la phase de validation. Pour illustrer l'application du modèle hyperexponentiel pour l'évaluation de la disponibilité, nous considérerons un logiciel pour lequel des données de défaillance ont été collectées au cours de sa vie opérationnelle. Nous appliquerons plus particulièrement le modèle pour évaluer l'indisponibilité moyenne du logiciel considéré.

II.6.2.1. Présentation du système et des données

Il s'agit du logiciel correspondant à la version 4096 de l'autocommutateur TROPICO-R développé par la compagnie brésilienne des télécommunications TELEBRAS. Le TROPICO-R est un centre de commutation local basé sur une structure matérielle et logicielle distribuée.

Le logiciel du TROPICO-R est divisé en deux parties [Mello 86, Moreira 86] : le logiciel d'application qui regroupe tous les traitements permettant l'accomplissement des fonctions téléphoniques, et l'exécutif qui offre l'environnement nécessaire à la réalisation des services demandées par le logiciel d'application.

Le volume global du logiciel est de l'ordre de 330 kilo-octets.

Au cours des 15 premiers mois d'utilisation du système sur des sites opérationnels, on a observé 50 défaillances logicielles. Durant la période considérée, le nombre de sites dans lequel le logiciel a été installé a évolué de 4 à 42 sites.

Sur la figure II.21 nous donnons la répartition des défaillances observées par semaine ainsi que l'évolution du nombre de systèmes installés au cours de la période considérée.

i	n(i)	NS(i)	i	n(i)	NS(i)	i	n(i)	NS(i)
1	2	4	24	1	36	7		42
2	0	10	25	1	36	8		42
3	2	10	26	0	36	9		42
4	1	10	27	0	36	0		42
5	1	10	28	1	38	1		42
6	0	12	29	1	40	2		42
7	2	12	30	0	40	3		42
8	1	12	31	0	40	4		42
9	2	12	32	0	40	5		42
10	5	12	33	0	42	6		42
11	2	13	34	0	42	7		42
12	1	13	35	0	42	8		42
13	2	13	36	1	42	9		42
14	0	13	37	0	42	0		42
15	0	21	38	0	42	1		42
16	0	21	39	0	42	2		42
17	0	21	40	0	42	3		42
18	1	21	41	0	42	4		42
19	1	21	42	0	42	5		42
20	2	28	43	1	42	6		42
21	1	28	44	2	42	7		42
22	0	28	45	0	42			
23	4	28	46	1	42			

i = numéro de semaine, n(i) = nombre de défaillances/semaine ; NS(i) = nombre de systèmes installés

Figure II.21 : répartition des données de défaillance par semaine

Les fautes activées durant l'exploitation du logiciel ont été corrigées hors ligne. La restauration de service sur site a été effectuée soit par une relance du logiciel soit par une réinitialisation du système. Les intervalles de temps jusqu'à restauration du logiciel suite à l'occurrence de défaillance n'ont pas été indiqués dans les relevés de défaillance fournis. Néanmoins, il a été remarqué que la restauration du logiciel durant la période considérée durait en général entre 1 et 5 minutes. Afin de compléter les informations contenues dans les relevés de défaillance collectées, nous avons utilisé un générateur aléatoire pour générer des intervalles de temps jusqu'à restauration variant selon une distribution uniforme entre 1 et 5 minutes. Les valeurs obtenues sont données sur la figure II.22 où ξ_i est l'intervalle de temps jusqu'à restauration après occurrence de la défaillance i du logiciel.

i	ξ_i	i	ξ_i	i	ξ_i
1	1,1	18	3,5	35	4,1
2	2,6	19	4,3	36	4,1
3	1,6	20	2,4	37	3,4
4	2,4	21	3,8	38	1,3
5	4,0	22	3,2	39	1,3
6	2,8	23	1,3	40	3,9
7	4,5	24	1,0	41	1,2
8	1,2	25	2,8	42	2,9
9	4,6	26	3,2	43	2,3
10	2,3	27	4,3	44	4,9
11	3,1	28	2,6	45	4,7
12	2,1	29	3,4	46	2,3
13	1,1	30	4,2	47	1,8
14	2,6	31	3,3	48	1,5
15	3,2	32	4,0	49	1,7
16	1,7	33	3,9	50	1,9
17	4,0	34	3,9		

Figure II.22 : Intervalles de temps (en minutes) jusqu'à restauration du logiciel

II.6.2.2. Evaluation de l'indisponibilité moyenne du logiciel

Pour évaluer l'indisponibilité moyenne du logiciel, nous avons procédé en deux étapes. La première étape de l'évaluation consiste à trouver un estimateur statistique permettant de caractériser l'évolution de la disponibilité moyenne du logiciel durant la période considérée. La deuxième étape de l'évaluation consiste à utiliser les valeurs obtenues à partir de l'estimateur statistique pour estimer les paramètres du modèle hyperexponentiel et ensuite comparer les estimations obtenues à partir du modèle à celles qui sont évaluées à partir de l'estimateur statistique et vérifier si le modèle suit bien l'évolution de la disponibilité moyenne du logiciel.

La disponibilité moyenne du logiciel sur un intervalle [0,t] peut être interprétée comme la proportion du temps pendant lequel il est opérationnel. Si t_i dénotent les intervalles de temps jusqu'à défaillance, un estimateur statistique de la disponibilité moyenne du logiciel est donné par :

$$\hat{A}_m(t) = \frac{1}{t} \sum_{i=1}^{n} t_i \quad \text{où n est le nombre de défaillances observées dans [0,t]} \qquad \text{(II.34)}$$

Par suite, l'indisponibilité moyenne observée du logiciel dans [0, t] peut être évaluée par l'estimateur donné par l'expression suivante :

$$\bar{\hat{A}}_m(t) = \frac{1}{t} \sum_{i=1}^{n} \xi_i \qquad \text{(II.35)}$$

Compte tenu de l'évolution du nombre de systèmes installés durant la période considérée, $\bar{\hat{A}}_m(t)$ sera évalué pour un système moyen ce qui permet de caractériser la disponibilité telle quelle est perçue par un utilisateur donné du logiciel.

La relation II.36 donne l'expression de $\bar{\hat{A}}_m^*$, l'estimateur de l'indisponibilité moyenne du logiciel pour un système moyen où :

- Δt est l'intervalle de temps correspondant à une semaine,
- NS(i) est le nombre total de systèmes opérationnels au cours de la semaine i,
- m_i est le nombre total de défaillances observées après i semaines d'utilisation du logiciel.

$$\bar{\hat{A}}_m^*(k\,\Delta t) = \frac{1}{k} \sum_{i=1}^{k} \frac{\sum_{m_{i-1}+1}^{m_i} \xi_j}{\Delta t\, NS(i)} \qquad k = 1 \ldots 67 \text{ et } m_0 = 0 \qquad \text{(II.36)}$$

On peut appliquer maintenant le modèle hyperexponentiel en utilisant l'expression de l'indisponibilité moyenne donnée par la relation II.21 pour vérifier si le modèle proposé suit bien l'évolution de l'indisponibilité moyenne du logiciel observée durant la période considérée.

L'utilisation de la méthode des moindres carrés pour l'estimation des paramètres du modèle et l'évaluation de l'indisponibilité moyenne du logiciel par le modèle hyperexponentiel donne les résultats qui sont présentés sur la figure II.23 : C1 correspond à l'indisponibilité moyenne

évaluée par le modèle hyperexponentiel (donnée par la relation II.21) et C2 est l'indisponibilité moyenne observée obtenue à partir de la relation II.36.

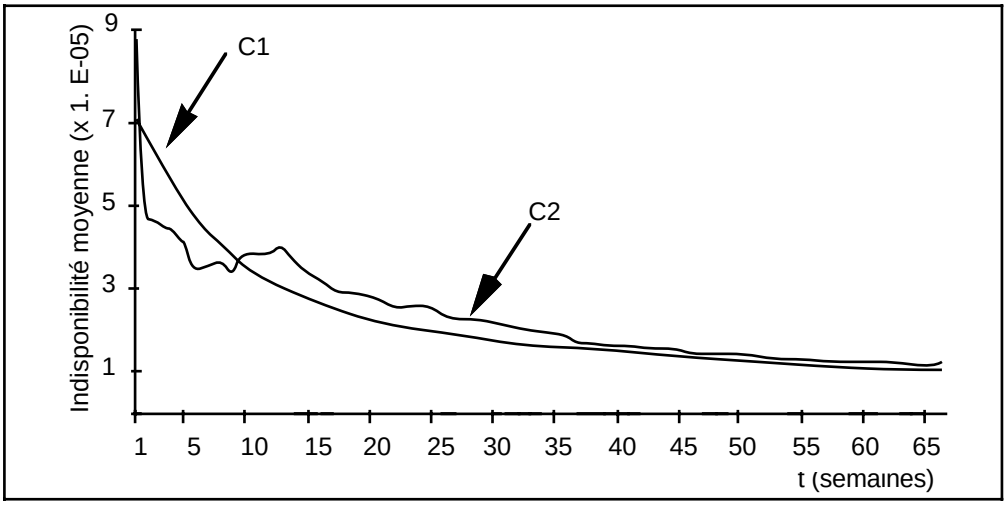


Figure II.23 : Indisponibilité moyenne du logiciel TROPICO-R 4096

On peut remarquer d'une part, que les estimations effectuées par le modèle hyperexponentiel sont satisfaisantes et, d'autre part, que l'allure d'évolution de l'indisponibilité moyenne observée du logiciel correspond bien aux hypothèses établies par le modèle hyperexponentiel : on observe d'abord une croissance de disponibilité avant d'atteindre le comportement stabilisé du logiciel.

II.6.2.3. Conclusion

Nous venons d'illustrer sur un cas réel l'application du modèle hyperexponentiel pour évaluer l'indisponibilité moyenne d'un système. La méthode choisie pour appliquer le modèle consiste à évaluer d'abord un estimateur statistique de l'indisponibilité observée du système et d'utiliser les valeurs obtenues pour estimer les paramètres du modèle par la méthode des moindres carrées. Une autre alternative serait d'appliquer le modèle pour suivre l'évolution de l'intensité de défaillance du système et utiliser les paramètres estimés dans l'expression du modèle relative à la disponibilité.

Dans [Laprie 91-a], le modèle a été appliqué aux mêmes données qui sont présentées dans ce paragraphe pour estimer l'évolution de l'intensité de défaillance et pour évaluer l'indisponibilité moyenne du système en considérant la même approche que celles que nous venons de présenter. Cette application a permis de vérifier la cohérence du modèle vis-à-vis des deux mesures : les valeurs des paramètres du modèle estimées par la méthode des moindres carrés dans les deux cas sont du même ordre de grandeur.

CONCLUSION

Dans ce chapitre, nous avons introduit le modèle hyperexponentiel et présenté ses principales caractéristiques. Par rapport aux modèles de croissance de fiabilité existants, ce modèle se distingue par son aptitude, d'une part, à modéliser une croissance de fiabilité suivie d'une fiabilité stabilisée, et d'autre part, à évaluer les deux mesures principales de la sûreté de

fonctionnement d'un système : la fiabilité et la disponibilité. La modélisation de la disponibilité des systèmes en tenant compte du phénomène de croissance de fiabilité a été effectuée en se basant sur l'interprétation markovienne du modèle.

Nous avons aussi étudié les propriétés du modèle hyperexponentiel généralisé, et nous avons remarqué que ce modèle possède des propriétés intéressantes qui lui permettent de mieux prendre en compte les variations de la fiabilité et de la disponibilité du logiciel au cours de son cycle de vie.

Afin d'illustrer l'utilisation du modèle hyperexponentiel dans un contexte réel, nous avons présenté quelques résultats concernant l'application du modèle pour l'évaluation des mesures de sûreté de fonctionnement de deux logiciels de télécommunication. Les différentes applications effectuées ont permis d'illustrer différents aspects de l'utilisation du modèle. Nous avons montré, par application à des données réelles, comment utiliser le modèle :

- pour suivre le comportement du logiciel au cours de sa validation par l'estimation de l'évolution du nombre cumulé de défaillances,
- pour évaluer le MTTF et le taux de défaillance du logiciel et estimer la fiabilité du logiciel avant sa mise en opération,
- pour évaluer l'indisponibilité moyenne du logiciel en tenant compte du phénomène de croissance de fiabilité.

Nous avons montré dans les différents exemples traités qu'il est préférable d'effectuer une analyse de tendance avant d'appliquer les modèles de croissance de fiabilité afin d'obtenir des estimations qui sont représentatives du comportement du système considéré.

D'autres exemples d'application du modèle hyperexponentiel à des systèmes réels se trouvent dans [Kanoun 85, Kanoun 87, Gaudoin 90, Metge 90, Laprie 91-a]. En particulier, dans [Laprie 91-a], le modèle hyperexponentiel est appliqué aux données de défaillance relatives à un sous-système matériel de l'autocommutateur 5ESS d'AT&T [Bauer 85] pour illustrer l'aptitude du modèle à prendre en compte la croissance de fiabilité du matériel.

i	n(i)	NS(i)	i	n(i)	NS(i)	i	n(i)	NS(i)
1	2	4	24	1	36	47	0	42
2	0	10	25	1	36	48	0	42
3	2	10	26	0	36	49	0	42
4	1	10	27	0	36	50	1	42
5	1	10	28	1	38	51	0	42
6	0	12	29	1	40	52	1	42
7	2	12	30	0	40	53	1	42
8	1	12	31	0	40	54	0	42
9	2	12	32	0	40	55	0	42
10	5	12	33	0	42	56	1	42
11	2	13	34	0	42	57	1	42
12	1	13	35	0	42	58	6	42
13	2	13	36	1	42	59	0	42
14	0	13	37	0	42	60	0	42
15	0	21	38	0	42	61	0	42
16	0	21	39	0	42	62	1	42
17	0	21	40	0	42	63	0	42
18	1	21	41	0	42	64	0	42
19	1	21	42	0	42	65	0	42
20	2	28	43	1	42	66	1	42
21	1	28	44	2	42	67	0	42
22	0	28	45	0	42			
23	4	28	46	1	42			

i	ξ_i	i	ξ_i	i	ξ_i
1	1,1	18	3,5	35	4,1
2	2,6	19	4,3	36	4,1
3	1,6	20	2,4	37	3,4
4	2,4	21	3,8	38	1,3
5	4,0	22	3,2	39	1,3
6	2,8	23	1,3	40	3,9
7	4,5	24	1,0	41	1,2
8	1,2	25	2,8	42	2,9
9	4,6	26	3,2	43	2,3
10	2,3	27	4,3	44	4,9
11	3,1	28	2,6	45	4,7
12	2,1	29	3,4	46	2,3
13	1,1	30	4,2	47	1,8
14	2,6	31	3,3	48	1,5
15	3,2	32	4,0	49	1,7
16	1,7	33	3,9	50	1,9
17	4,0	34	3,9		

i	t_i	$MTTF_i$
30	9	7,80
31	14	8,11
32	27	9,31
33	1	12,30
34	11	10,67
35	1	10,67
36	2	9,62
37	6	8,85
38	0	8,59
39	7	7,87
40	42	7,80
41	13	10,26
42	2	10,44

i	t_i	$MTTF_i$	i	t_i	$MTTF_i$
16	0	1,87	30	9	2,64
17	3	1,75	31	14	2,93
18	1	1,82	32	27	3,42
19	0	1,78	33	1	4,46
20	1	1,68	34	11	4,31
21	0	1,65	35	1	4,59
22	0	1,57	36	2	4,45
23	6	1,50	37	6	4,35
24	0	1,70	38	0	4,41
25	1	1,63	39	7	4,25
26	2	1,60	40	42	4,35
27	10	1,62	41	13	5,65
28	1	2,09	42	2	5,90
29	14	2,03			

CHAPITRE III

MODÈLE HYPEREXPONENTIEL EN TEMPS DISCRET

INTRODUCTION

Le modèle hyperexponentiel présenté dans le chapitre II ainsi que la plupart des modèles de croissance de fiabilité caractérisent l'évolution dans le temps du comportement du logiciel vis-à-vis de l'activation et de l'élimination des fautes de conception. Ce type de modèles n'est pas bien approprié pour représenter le comportement de certains systèmes qui ne sont sollicités qu'occasionnellement par leur environnement (par exemple les systèmes transactionnels, les logiciels de commande de missiles, …). Pour ces systèmes, il est plus significatif d'évaluer l'évolution de la sûreté de fonctionnement en fonction du nombre d'exécutions effectuées et non en fonction du temps.

Le modèle hyperexponentiel en temps discret que nous proposons dans ce chapitre est basé sur une représentation de la croissance de fiabilité du logiciel en fonction du nombre d'exécutions effectuées en tenant compte des corrections introduites au cours de son cycle de vie. Il permet plus particulièrement de modéliser l'évolution de la probabilité de défaillance du logiciel en fonction du nombre d'exécutions effectuées. Le modèle est basé sur des hypothèses qui sont équivalentes à celles que nous avons considérées dans le chapitre II pour l'établissement du modèle hyperexponentiel en temps continu. En effet, ce modèle permet de représenter une probabilité de défaillance à l'exécution qui décroît en fonction du nombre d'exécutions du logiciel et qui tend asymptotiquement vers une limite non nulle. Il modélise une croissance de fiabilité du logiciel suivie d'une fiabilité stabilisée permettant ainsi, d'une part, de prendre en compte l'influence de l'élimination des fautes de conception du logiciel sur son domaine de défaillance, et d'autre part, de modéliser l'impact, sur le comportement du logiciel, des fautes résiduelles qui demeurent dans le système au cours de sa vie opérationnelle en dépit des corrections effectuées.

Ce chapitre est consacré au modèle hyperexponentiel en temps discret. Nous présentons d'abord les propriétés du modèle et nous établissons les expressions des mesures de fiabilité associées. Nous étudions ensuite le lien entre ce modèle et le modèle hyperexponentiel en temps continu présenté dans le chapitre II. L'application du modèle à des données réelles est considérée dans la dernière partie de ce chapitre.

III.1 PRéSENTATION DU MODÈLE

Le modèle hyperexponentiel en temps discret décrit l'évolution de la probabilité de défaillance du logiciel à l'exécution en fonction du nombre d'exécutions effectuées. Il est basé sur des hypothèses équivalentes à celles du modèle hyperexponentiel en temps continu (cf. paragraphe II.1).

Soit P(n) la probabilité de défaillance du logiciel au cours de la n ième exécution. Le modèle que nous proposons est basé sur l'hypothèse que la probabilité P(n) est indépendante de l'occurrence de défaillances au cours des (n-1) premières exécutions. Cette hypothèse ne signifie pas que ponctuellement à chaque exécution la probabilité de défaillance décroît mais traduit plutôt, que l'évolution de la probabilité de défaillance à l'exécution du logiciel au cours de son cycle de vie peut être approchée par une courbe qui décroît au fur et à mesure que le nombre d'exécutions du logiciel augmente.

P(n) est donc l'équivalent en temps discret de la fonction intensité de défaillance caractéristique des modèles de Poisson non homogènes.

Le modèle de croissance de fiabilité en temps discret est défini par la fonction P(n) :

$$P(n) = \frac{\theta p_{sup}(1-p_{sup})^{n-1} + \overline{\theta} p_{inf}(1-p_{inf})^{n-1}}{\theta(1-p_{sup})^{n-1} + \overline{\theta}(1-p_{inf})^{n-1}} \qquad \text{(III.1)}$$

avec $0 \leq \theta \leq 1$ $\overline{\theta} = 1-\theta$ et $p_{inf} \leq p_{sup}$

P(n) est une fonction à trois paramètres θ, p_{sup} et p_{inf} et vérifie les propriétés suivantes :

- $P(1) = \theta\, p_{sup} + \overline{\theta}\, p_{inf}$ et $P(\infty) = p_{inf}$
- la suite définie par P(n) est décroissante.
- le sens de variation de P(n) dépend du paramètre θ.

En effet, soit :

$$\Delta P(n+1) = P(n+1) - P(n) \qquad \text{(III.2)}$$

$$\Delta P(n+1) = -\frac{\theta\overline{\theta}\,(p_{sup}-p_{inf})^2(1-p_{sup})^{n-1}(1-p_{inf})^{n-1}}{\prod_{i=n-1}^{n}\left\{\theta(1-p_{sup})^{i} + \overline{\theta}(1-p_{inf})^{i}\right\}} \leq 0 \quad \text{pour tout } n \geq 1 \qquad \text{(III.3)}$$

$$\Delta P(n+1) - \Delta P(n) = -\frac{\theta\overline{\theta}\,(p_{sup}-p_{inf})^3(1-p_{sup})^{n-2}(1-p_{inf})^{n-2}\left\{\theta(1-p_{sup})^{n-1} - \overline{\theta}(1-p_{inf})^{n-1}\right\}}{\prod_{i=n-2}^{n}\left\{\theta(1-p_{sup})^{i} + \overline{\theta}(1-p_{inf})^{i}\right\}} \qquad \text{(III.4)}$$

Le sens de variation de P(n) détermine l'allure d'évolution de la probabilité de défaillance du logiciel considéré au cours de son cycle de vie. Il est donné par le signe de $\Delta P(n+1) - \Delta P(n)$.

L'étude de la relation III.4 montre que :

- si $\theta \leq \overline{\theta}$ alors $\Delta P(n+1) < \Delta P(n)$

- si $\theta > \overline{\theta}$ et si on pose

$$n_0 = 1 + \frac{1}{\mathrm{Ln}\left(\frac{1-p_{inf}}{1-p_{sup}}\right)} \mathrm{Ln}\left(\frac{\theta}{\overline{\theta}}\right) \qquad \text{(III.5)}$$

alors

- $\Delta P(n+1) > \Delta P(n)$ pour $n < n_0$
- $\Delta P(n+1) \leq \Delta P(n)$ pour $n \geq n_0$

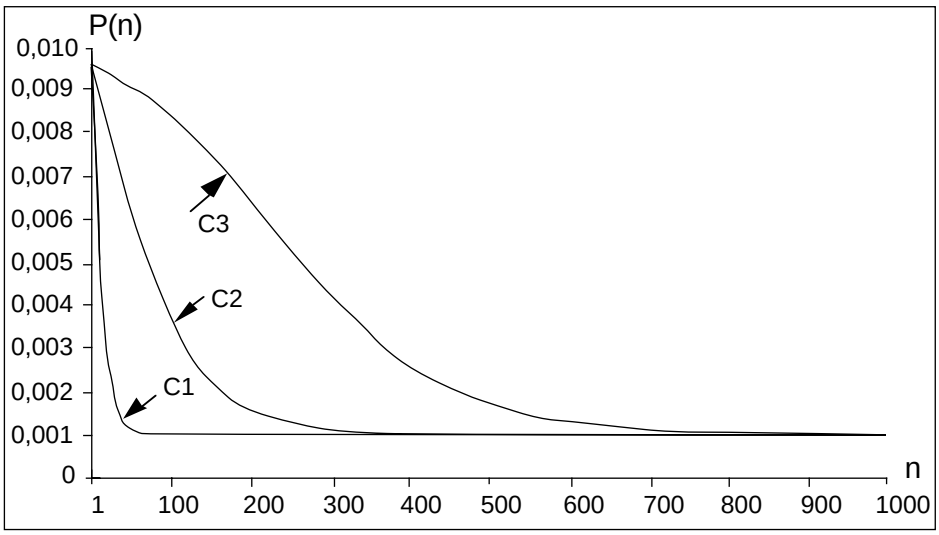
D'une façon générale, la variation de la probabilité de défaillance à l'exécution est beaucoup plus importante au cours de la phase d'intégration et au début de la phase opérationnelle du logiciel que vers la fin de son cycle de vie. En effet, au début de la vie opérationnelle, l'exécution du logiciel conduit essentiellement à l'activation des fautes dont la probabilité d'occurrence est grande. L'élimination de ces fautes conduit à une réduction significative du domaine de défaillance du logiciel et par suite à la décroissance de la probabilité de défaillance à l'exécution. Vers la fin du cycle de vie du logiciel, les fautes qui demeurent dans le logiciel sont celles qui sont caractérisées par des probabilités d'activation faibles. La fiabilité du logiciel tend alors vers un comportement stabilisé qui se traduit par une variation très faible de la probabilité de défaillance à l'exécution.

Le modèle hyperexponentiel en temps discret modélise une probabilité de défaillance à l'exécution qui décroît avec le nombre d'exécutions effectuées et qui tend vers une valeur asymptotique non nulle. La courbe donnant l'évolution de la probabilité de défaillance à l'exécution P(n) a donc la même allure que la courbe intensité de défaillance du modèle hyperexponentiel.

La figure III.1 illustre différentes allures de la probabilité de défaillance à l'exécution pouvant modéliser différents comportements d'un logiciel :

- croissance de fiabilité très rapide (C1),
- croissance de fiabilité progressive (C2),
- croissance de fiabilité lente (C3).

Les courbes C1, C2 et C3 sont similaires aux courbes caractéristiques du modèle hyperexponentiel en temps continu qui sont présentées sur la figure II.2.



Courbe	θ	p_{sup}	p_{inf}
C1	0,1	0,0900	0,001
C2	0,5	0,0180	0,001
C3	0,9	0,0105	0,001

Figure III.1 : Allures de la probabilité de défaillance à l'exécution P(n)

III.2 MESURES DE FIABILITé

Le processus d'exécution du logiciel peut être vu comme une séquence d'épreuves de Bernouilli, où une épreuve correspond à une exécution du logiciel. Soit Z_i la variable aléatoire qui caractérise l'exécution i. Z_i est une variable binaire telle que :

$$\begin{cases} Z_i = 0 \text{ quand l'exécution i est effectuée sans défaillance} \\ Z_i = 1 \text{ quand l'exécution i entraîne la défaillance du logiciel} \end{cases} \quad \text{(III.6)}$$

Les probabilités associées aux événements $\{Z_i = 1\}$ et $\{Z_i = 0\}$ sont données par :

$$\begin{cases} P(i) = Pr\ \{Z_i = 1\} \\ Q(i) = Pr\ \{Z_i = 0\} = 1 - P(i) \end{cases} \quad \text{(III.7)}$$

où P(i) est la probabilité de défaillance à l'exécution i donnée par la relation (III.1).

P(i) caractérise l'exécution i ; elle ne dépend pas explicitement de l'occurrence de défaillances au cours des (i-1) premières exécutions. Par analogie avec les modèles NHPP en temps continu, P(i) peut être interprétée de la même manière que la probabilité h(t)dt d'occurrence d'une défaillance dans l'intervalle [t,t+dt].

Connaissant P(i), on peut établir les expressions des différentes mesures de fiabilité du logiciel en fonction du nombre d'exécutions effectuées.

III.2.1 Nombre cumulé de défaillances

Désignons par Y_n la variable aléatoire "nombre d'occurrences de défaillances au cours de n exécutions".

Y_n s'écrit en fonction des variables aléatoires Z_i sous la forme :

$$Y_n = Z_1 + Z_2 + \ldots + Z_n$$

Connaissant les propriétés des variables aléatoires de Bernouilli, on peut alors obtenir directement l'expression de H(n), l'espérance mathématique de la variable Y_n [Feller 68] :

$$H(n) = E[Y_n] = \sum_{i=1}^{n} E[Z_i] = \sum_{i=1}^{n} P(i) \qquad \text{(III.8)}$$

$$H(n) = \sum_{i=1}^{n} \frac{\theta p_{sup}(1-p_{sup})^{i-1} + \overline{\theta}\, p_{inf}(1-p_{inf})^{i-1}}{\theta(1-p_{sup})^{i-1} + \overline{\theta}(1-p_{inf})^{i-1}} \qquad \text{(III.9)}$$

L'expression de H(n) est relativement complexe. Compte tenu du fait que les paramètres p_{inf} et p_{sup} sont tels p_{inf} et $p_{sup} << 1$, on peut écrire H(n) sous une forme plus simple en effectuant un développement limité de l'expression (III.9) par rapport à p_{inf} et p_{sup}. On trouve :

$$H(n) = -Ln \left\{ \theta(1-p_{sup})^{n} + \overline{\theta}\,(1-p_{inf})^{n} \right\} + o(p_{inf}, p_{sup}) \qquad \text{(III.10)}$$

En effet, si p_{inf} et $p_{sup} << 1$, alors :

$$1-P(i) \approx \exp(-P(i)) + o(p_{inf}, p_{sup}) \text{ pour } i = 1, 2\ldots \qquad \text{(III.11)}$$

$$\prod_{i=1}^{n} (1-P(i)) \approx \prod_{i=1}^{n} \exp(-P(i)) + o(p_{inf}, p_{sup}) \qquad \text{(III.12)}$$

$$\prod_{i=1}^{n} (1-P(i)) \approx \exp\left(-\sum_{i=1}^{n} P(i) \right) + o(p_{inf}, p_{sup}) = \exp(-H(n)) \qquad \text{(III.13)}$$

$$\text{Or } \prod_{i=1}^{n} (1-P(i)) = \theta(1-p_{sup})^{n} + \overline{\theta}\,(1-p_{inf})^{n} \qquad \text{(III.14)}$$

En utilisant les relations (III.13) et (III.14) on obtient finalement la relation (III.10).

REMARQUE

La distribution de probabilité de la variable aléatoire Y_n est difficile à obtenir directement car les probabilités P(i) sont variables. Néanmoins, compte tenu de l'ordre de grandeur de P(i), la loi de Y_n peut être approchée par une loi de Poisson de paramètre H(n) [Serfling 78] :

$$H(n) = \sum_{i=1}^{n} P(i) \qquad \text{(III.15)}$$

Cette dernière approximation est d'autant meilleure que les probabilités P(i) sont petites.

Ainsi, le processus d'occurrence des défaillances peut être décrit par les variables aléatoires en temps discret $\{Y_n, n \geq 0\}$ où Y_n est la variable aléatoire donnant le nombre d'occurrences de défaillances au cours de n exécutions.

Soient $s_1<s_2<\ldots<s_k$ une suite d'exécutions. Si on tient compte de l'approximation de la loi de Y_n par une loi de Poisson alors, la variable aléatoire $Y_{s_k} - Y_{s_{k-1}}$ donnant le nombre d'occurrences de défaillances dans l'intervalle d'exécution $[s_{k-1}, s_k]$ est distribuée selon une loi de Poisson :

$$\Pr\{Y_{s_k} - Y_{s_{k-1}} = i\} = \frac{[H(s_k) - H(s_{k-1})]^i}{i!} \exp\{-[H(s_k) - H(s_{k-1})]\} \quad \text{(III.16)}$$

III.2.2 Fiabilité

Soient :

- N_i, la variable aléatoire "nombre d'exécutions entre l'occurrence des défaillances (i-1) et i". On note par n_i la réalisation de N_i,
- M_i la variable aléatoire "nombre cumulé d'exécutions jusqu'à occurrence de la défaillance i". On note par m_i la réalisation de M_i, m_i est appelé instant d'occurrence de la défaillance i.

$$M_i = \sum_{j=1}^{i} N_j \quad \text{(III.17)}$$

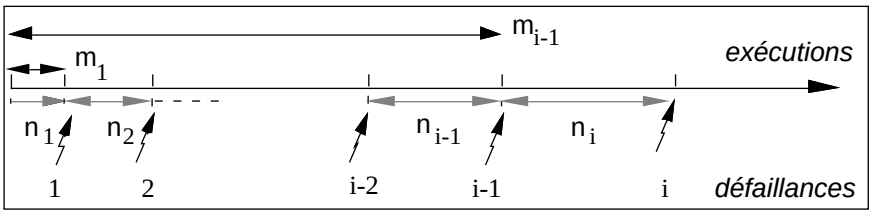


Figure III.2 : Lien entre les variables aléatoires N_i et M_i

Soit $R(n_i \mid m_{i-1})$ la fonction de fiabilité associée à l'occurrence de la défaillance i et conditionnée par l'événement $M_{i-1}=m_{i-1}$ caractérisant l'occurrence de la i-1 ème défaillance.

$R(n_i \mid m_{i-1})$ est donnée par :

$$R(n_i \mid m_{i-1}) = \Pr\{N_i \geq n_i \mid M_{i-1}=m_{i-1}\} \quad \text{(III.18)}$$

$$R(n_i \mid m_{i-1}) = \prod_{j=m_{i-1}+1}^{m_{i-1}+n_i} [1 - P(j)] \quad \text{(III.19)}$$

Compte tenu de l'expression de P(i), on obtient :

$$R(n_i \mid m_{i-1}) = \frac{\theta(1-p_{sup})^{m_{i-1}+n_i} + \overline{\theta}(1-p_{inf})^{m_{i-1}+n_i}}{\theta(1-p_{sup})^{m_{i-1}} + \overline{\theta}(1-p_{inf})^{m_{i-1}}} \quad \text{(III.20)}$$

Ainsi, la fiabilité $R(n_i \mid m_{i-1})$ dépend de l'instant d'occurrence de la dernière défaillance m_{i-1}.

Soit $f(n_i \mid m_{i-1})$ la fonction probabilité de masse associée à la variable N_i et conditionnée par l'événement $M_{i-1}=m_{i-1}$.

$$f(n_i \mid m_{i-1}) = Pr\{N_i = n_i \mid M_{i-1}=m_{i-1}\} \quad \text{(III.21)}$$

$$f(n_i \mid m_{i-1}) = Pr\{N_i = n_i \text{ et 0 défaillances entre } m_{i-1} \text{ et } m_{i-1}+ n_i -1\} \quad \text{(III.22)}$$

Ce qui donne :

$$f(n_i \mid m_{i-1}) = P(m_{i-1}+n_i)\ \ R(n_i -1 \mid m_{i-1}) \quad \text{(III.23)}$$

$P(m_{i-1} + n_i)$ étant la probabilité de défaillance à l'exécution $m_{i-1}+n_i$.

III.2.3 Nombre moyen d'exécutions jusqu'à défaillance

Soit $MTTF_i$ le nombre moyen d'exécutions jusqu'à occurrence de la défaillance i sachant que la défaillance (i-1) a eu lieu à l'exécution m_{i-1}.

$$MTTF_i = E[N_i] = \sum_{n_i=1}^{\infty} n_i f(n_i \mid m_{i-1}) \quad \text{(III.24)}$$

En utilisant la relation (III.23), on obtient :

$$MTTF_i = \sum_{n_i=1}^{\infty} n_i P(m_{i-1}+ n_i)\ \ R(n_i -1 \mid m_{i-1}) \quad \text{(III.25)}$$

Or, $R(n_i \mid m_{i-1})$ peut s'écrire sous la forme :

$$R(n_i \mid m_{i-1}) = R(n_i -1 \mid m_{i-1})\ \{1-P(m_{i-1}+ n_i)\} \quad \text{(III.26)}$$

D'où :

$$MTTF_i = \sum_{n_i=1}^{\infty} n_i \left\{R(n_i -1 \mid m_{i-1}) - R(n_i \mid m_{i-1})\right\} \quad \text{(III.27)}$$

$$MTTF_i = \sum_{n_i=1}^{\infty} R(n_i \mid m_{i-1}) \quad \text{(III.28)}$$

Si on utilise la relation (III.20) on obtient finalement :

$$MTTF_i = \frac{\theta \frac{1-p_{sup}}{p_{sup}} (1-p_{sup})^{m_{i-1}} + \overline{\theta} \frac{1-p_{inf}}{p_{inf}} (1-p_{inf})^{m_{i-1}}}{\theta(1-p_{sup})^{m_{i-1}} + \overline{\theta}(1-p_{inf})^{m_{i-1}}} \quad \text{(III.29)}$$

III.2.4 Taux de défaillance en temps discret

Soit $\lambda(n_i \mid m_{i-1})$ la fonction de hasard associée à la variable aléatoire N_i. $\lambda(n_i \mid m_{i-1})$ est le taux de défaillance en temps discret du logiciel après occurrence de la défaillance i-1.

Par définition:

$$\lambda(n_i \mid m_{i-1}) = \frac{f(n_i \mid m_{i-1})}{R(n_i - 1 \mid m_{i-1})} \quad \text{(III.30)}$$

Il vient donc à partir de la relation (III.23) :

$$\lambda(n_i \mid m_{i-1}) = P(m_{i-1}+n_i) \quad \text{(III.31)}$$

Ainsi, le taux de défaillance en temps discret associée à l'occurrence de la défaillance i a la même expression que celle de la probabilité de défaillance P(n) caractérisant le modèle hyperexponentiel en temps discret. En particulier, la probabilité de défaillance P(n) est égale au taux de défaillance en temps discret associé à l'occurrence de la première défaillance.

On retrouve ainsi une propriété équivalente à celle des processus NHPP en temps continu concernant le lien qui existe entre intensité de défaillance et taux de défaillance.

Cette propriété est importante car elle permet d'exprimer le taux de défaillance en temps discret directement à partir de la probabilité de défaillance à l'exécution.

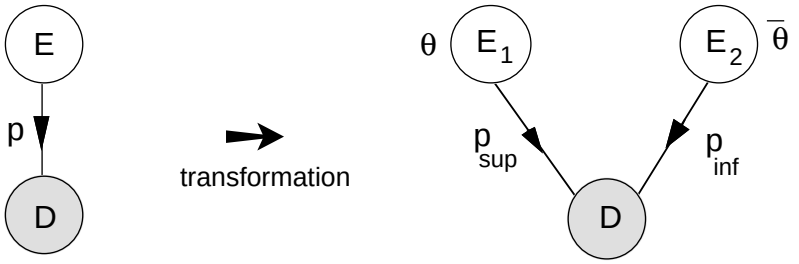
III.2.5 Interprétation markovienne du modèle

Par analogie avec le modèle hyperexponentiel, en utilisant la propriété précédente, on peut représenter le modèle hyperexponentiel en temps discret sous forme d'une chaîne de Markov discrète à trois états $\{E_1, E_2, D\}$ où D est un état absorbant et E_1 et E_2 sont deux états transitoires caractérisés par les probabilités initiales d'occupation θ et $\bar{\theta}$ et les probabilités de transition vers l'état absorbant p_{sup} et p_{inf}. La chaîne de Markov associée au modèle est représentée sur la figure III.3-b.

La modélisation du phénomène de croissance de fiabilité peut se ramener alors à la transformation d'une chaîne de Markov en temps discret classique formée de deux états : un état transitoire E avec une probabilité de transition p, et un état absorbant D (figure III.3-a), caractérisant le comportement en fiabilité stabilisée du logiciel, en une chaîne de Markov à trois états (figure III.3-b) permettant de représenter le phénomène de croissance de fiabilité. La transformation consiste à transformer l'état E en deux états E_1, E_2 avec des probabilités initiales

d'occupation θ et $\bar{\theta}$. Cette transformation est illustrée par la figure III.3 ; pour simplifier la représentation, les probabilités de transition d'un état vers lui même ne sont pas représentées.

La chaîne de Markov transformée permet de simuler l'évolution de la probabilité de défaillance P(n) caractérisant le processus d'occurrence de défaillance en fonction du nombre d'exécutions du logiciel au cours de son cycle de vie.



(a) : fiabilité stabilisée (b) : croissance de fiabilité

Figure III.3 : Interprétation markovienne du modèle hyperexponentiel en temps discret

III.2.6 Conclusion

Dans les paragraphes précédents nous avons établi les expressions des mesures de fiabilité associées au modèle hyperexponentiel en temps discret permettant de suivre le comportement du logiciel en fonction du nombre d'exécutions effectuées. L'analyse des expressions obtenues (relations III.10, III.20, III.29 et III.31) et leur comparaison avec celles qui sont établies à partir du modèle hyperexponentiel en temps continu (relations II.7, II.8, II.10 et II.11), montrent l'analogie entre ces expressions. Ceci était prévisible compte tenu du fait que les deux modèles sont basés sur des hypothèses équivalentes.

Dans le paragraphe suivant, nous analyserons le lien qui existe entre ces deux modèles et nous étudierons le problème du passage du modèle hyperexponentiel en temps discret en un modèle en temps continu permettant de représenter le comportement du logiciel tel qu'il est perçu par ses utilisateurs dans un environnement donné.

III.3 LIEN ENTRE LES MODÈLES HYPEREXPONENTIEL EN TEMPS DISCRET ET EN TEMPS CONTINU

Nous analysons tout d'abord le lien qui existe entre les expressions analytiques de la probabilité de défaillance à l'exécution du modèle en temps discret et de l'intensité de défaillance du modèle en temps continu. Nous étudions ensuite le lien entre les deux modèles à partir de leur représentation markovienne.

III.3.1 Lien entre les expressions analytiques

Le modèle hyperexponentiel en temps discret modélise l'évolution de la fiabilité du logiciel en fonction du nombre d'exécutions effectuées sans tenir compte des durées de ces exécutions.

Pour étudier la relation qui existe entre ce modèle et le modèle hyperexponentiel en temps continu, il est nécessaire de tenir compte de ce paramètre.

Considérons d'abord le cas d'un logiciel qui s'exécute en continu ; c'est-à-dire pour lequel nous supposons que le temps séparant la fin d'une exécution et le début de l'exécution suivante est négligeable.

Soit t_e le temps correspondant à la durée d'une exécution. On suppose que t_e est constant Dans le cas où la durée d'une exécution est une variable aléatoire on remplacera t_e par sa valeur moyenne.

Soit t le temps écoulé après n-1 exécutions du logiciel : $t = (n-1)\, t_e$

Supposons qu'il existe des limites finies pour les quantités $\frac{p_{sup}}{t_e}$ et $\frac{p_{inf}}{t_e}$.

Soient ζ_{sup} et ζ_{inf} ces limites respectives :

$$\begin{cases} \zeta_{sup} = \lim\limits_{t_e=0} \frac{p_{sup}}{t_e} & \text{soit} \quad p_{sup} = \zeta_{sup} t_e + o(t_e) \\ \zeta_{inf} = \lim\limits_{t_e=0} \frac{p_{inf}}{t_e} & \text{soit} \quad p_{inf} = \zeta_{inf} t_e + o(t_e) \end{cases} \tag{III.32}$$

où o(x) dénote une quantité tendant vers zéro plus vite que x : $\lim\limits_{x=0} \frac{o(x)}{x}$

Posons :

$$\omega = \theta \tag{III.33}$$

En utilisant les relations (III.32) et (III.33), P(n) s'écrit :

$$P(n) = \frac{\omega\left\{\zeta_{sup}t_e+o(t_e)\right\}\left\{1-\zeta_{sup}t_e+o(t_e)\right\}^{t/t_e} + \varpi\left\{\zeta_{inf}t_e+o(t_e)\right\}\left\{1-\zeta_{inf}t_e+o(t_e)\right\}^{t/t_e}}{\omega\left\{1-\zeta_{sup}t_e+o(t_e)\right\}^{t/t_e} + \varpi\left\{1-\zeta_{inf}t_e+o(t_e)\right\}^{t/t_e}} \tag{III.34}$$

Or :

$$\left\{1-\zeta_{sup}t_e+o(t_e)\right\}^{t/t_e} = \exp\left(\mathrm{Ln}\left[\left\{1-\zeta_{sup}t_e+o(t_e)\right\}^{t/t_e}\right]\right) \approx \exp\left\{-\zeta_{sup}t+o(t_e)\right\} \tag{III.35}$$

$$\left\{1-\zeta_{inf}t_e+o(t_e)\right\}^{t/t_e} = \exp\left(\mathrm{Ln}\left[\left\{1-\zeta_{inf}t_e+o(t_e)\right\}^{t/t_e}\right]\right) \approx \exp\left\{-\zeta_{inf}t+o(t_e)\right\} \tag{III.36}$$

On obtient, finalement :

$$\lim_{t_e=0} \frac{P(n)}{t_e} = \frac{\omega\zeta_{sup}\exp(-\zeta_{sup}t) + \varpi\zeta_{inf}\exp(-\zeta_{inf}t)}{\omega\exp(-\zeta_{sup}t) + \varpi\exp(-\zeta_{inf}t)} = h(t) \tag{III.37}$$

On retrouve ainsi l'expression de l'intensité de défaillance relative au modèle hyperexponentiel en temps continu.

Considérons maintenant le cas d'une exécution discontinue. Soit t_r le temps de repos cumulé jusqu'à la n ième exécution : t_r correspond à la somme des temps séparant la fin d'une exécution et le début de l'exécution suivante.

Soit t le temps écoulé jusqu'au début de l'exécution n. On a :

$$t = (n-1)t_e + t_r \qquad \text{(III.38)}$$

Si on procède de manière identique à celle du cas précédent, en effectuant un développement limité de P(n) par rapport à $\frac{p_{sup}}{t_e}$ et $\frac{p_{inf}}{t_e}$, on trouve :

$$\lim_{t_e=0} \frac{P(n)}{t_e} = \frac{\omega\zeta_{sup}\exp(-\zeta_{sup}(t-t_r)) + \varpi\zeta_{inf}\exp(-\zeta_{inf}(t-t_r))}{\omega\exp(-\zeta_{sup}(t-t_r)) + \varpi\exp(-\zeta_{inf}(t-t_r))} = h(\pi t) \qquad \text{(III.39)}$$

où $\pi = \frac{t - t_r}{t} = \frac{(n-1)t_e}{t}$ est la proportion du temps pendant lequel le logiciel est actif.

Ainsi, le modèle de croissance de fiabilité en temps discret peut être interprété comme étant la version en temps discret du modèle hyperexponentiel. Cette approximation n'est valable que si le temps d'exécution du logiciel t_e est très petit devant le temps courant t. En d'autres termes l'approximation est valable si le nombre d'exécutions effectuées jusqu'à l'instant t est suffisamment grand.

III.3.2 Lien entre les représentations markoviennes

Etudions maintenant le lien qui existe entre les deux modèles à partir de leurs représentations markovienne.

Considérons d'abord les chaînes de Markov en temps continu de la figure III.4 caractérisant le comportement en fiabilité stabilisée d'un logiciel vue comme une "boîte noire" [Laprie 84-b] : on suppose que le logiciel n'est pas sollicité en continu par son environnement.

Soient :

- A l'état du logiciel quand il n'est pas activé,
- E l'état caractérisant l'exécution du logiciel,
- D l'état défaillant qui est atteint quand le service délivré par le logiciel est non conforme aux spécifications,
- η le taux de sollicitation ; $1/\eta$ est l'intervalle de temps moyen entre deux sollicitations du logiciel,
- γ le taux d'exécution du logiciel ; $1/\gamma$ est la durée moyenne d'une exécution,
- γ^* le taux d'exécution sans défaillance du logiciel ; $1/\gamma^*$ est la durée moyenne d'une exécution sans défaillance,
- p la probabilité de défaillance à l'exécution,

- μ le taux de restauration du logiciel.

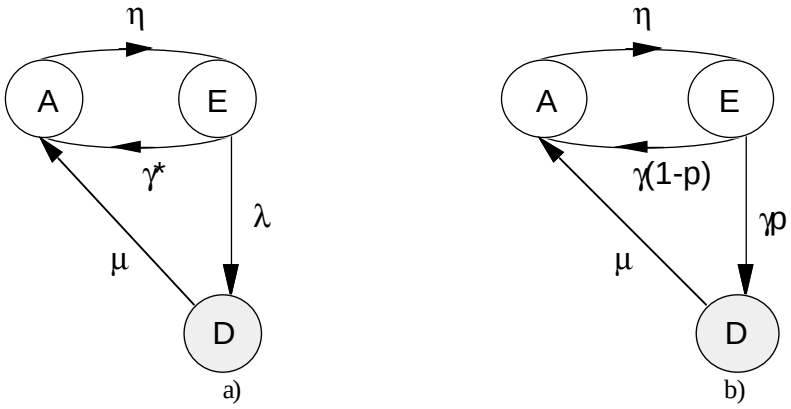


Figure III.4 : Modèles de comportement du logiciel en fiabilité stabilisée

Les modèles a et b de la figure III.4 sont équivalents. Il suffit de poser $\lambda = \gamma p$ et de tenir compte du fait que le taux de défaillance du logiciel est très petit devant son taux d'exécution sans défaillance :

$$\gamma(1-p) >> \gamma p \quad \text{soit} \quad \gamma(1-p) \approx \gamma \approx \gamma^* \tag{III.40}$$

Cette approximation exprime que la durée moyenne d'une exécution est très inférieure au temps moyen entre défaillances. Cette propriété est généralement satisfaite car sinon le logiciel étudié est sans intérêt.

L'avantage d'une modélisation du type de la figure III.4-b est qu'elle permet de distinguer entre les processus d'activation des événements qui régissent les transitions entre les différents états du système et les processus temporels liés au temps de séjour dans les états. Les transitions entre les états du système sont décrites par la chaîne immergée déduite de la chaîne de Markov en temps continu [Howard 71].

Pour prendre en compte le phénomène de croissance de fiabilité, on peut utiliser la représentation markovienne du modèle hyperexponentiel en temps continu et la technique de transformation introduite dans paragraphes II.3 et II.4.1.

Si on applique la technique de transformation à la chaîne de Markov présentée sur la figure III.4-a, on obtient la chaîne de Markov présentée sur la figure III.5-a.

Posons :

$$\begin{cases} \zeta_{sup} = \gamma p_{sup} \\ \zeta_{inf} = \gamma p_{inf} \\ \omega = \theta \end{cases} \tag{III.41}$$

Si on tient compte des approximations :

$$\gamma^* \approx \gamma \approx \gamma(1-p_{sup}) \approx \gamma(1-p_{inf}) \tag{III.42}$$

la chaîne de Markov de la figure III.5-a est équivalente à la chaîne de Markov présentée sur la figure III.5-b.

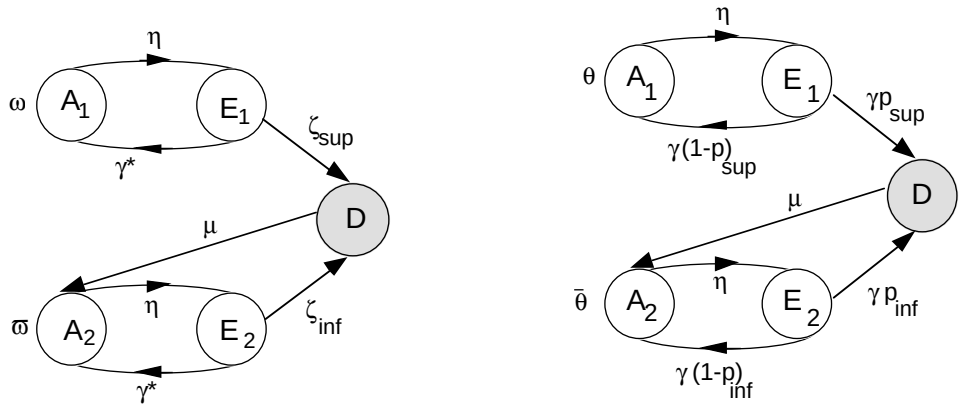
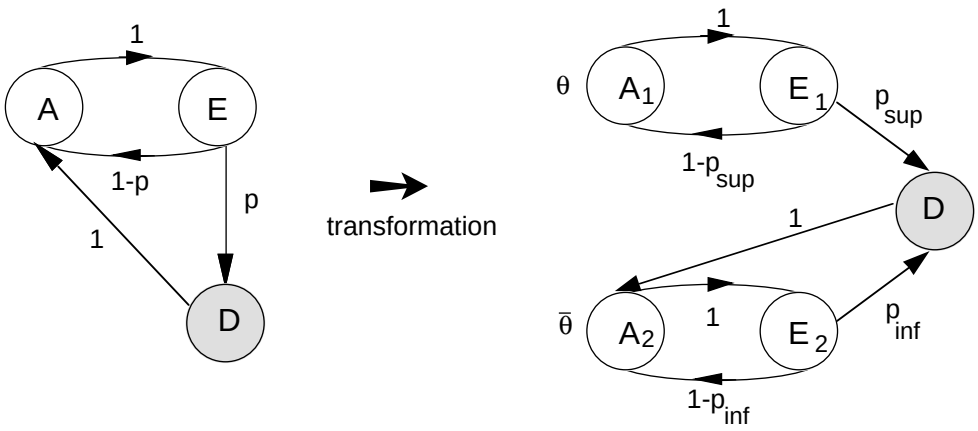


Figure III.5 : Modèles de comportement du logiciel en croissance de fiabilité

Les chaînes immergées associées aux chaînes de Markov présentées sur les figures III.4-b et III.5-b, sont données respectivement sur les figures III.6-a et III.6-b. On peut remarquer alors que l'application de la technique de transformation relative au modèle hyperexponentiel en temps continu peut se ramener à l'application de la technique de transformation, relative au modèle hyperexponentiel en temps discret (paragraphe III.2.4), à la chaîne immergée associée à la chaîne de Markov en temps continu caractérisant le comportement du logiciel en fiabilité stabilisée. Une telle transformation est illustrée sur la figure III.6.



a) fiabilité stabilisée b) croissance de fiabilité

Figure III.6 : Transformation des chaînes de Markov immergées par le modèle hyperexponentiel en temps discret

III.3.3 Conclusion

Dans le paragraphe III.3, nous avons étudié le lien qui existe entre le modèle hyperexponentiel en temps discret et le modèle hyperexponentiel en temps continu. A partir des expressions analytiques des deux modèles, nous avons montré que l'on peut passer du modèle en temps continu vers le modèle en temps discret en discrétisant le temps. De plus, en utilisant les interprétations markoviennes des deux modèles, nous avons montré que l'application de la technique de transformation pour transformer une chaîne de Markov en temps continu

représentant le comportement d'un système en fiabilité stabilisée en une chaîne de Markov modélisant le phénomène de croissance de fiabilité peut se ramener à l'application d'une transformation équivalente en temps discret à la chaîne immergée extraite de la chaîne de Markov en temps continu.

Compte tenu de ces propriétés, on peut modéliser d'abord l'évolution du comportement du logiciel en fonction du nombre d'exécutions effectuées pour caractériser le phénomène de croissance de fiabilité et transformer ensuite le modèle obtenu en tenant compte des caractéristiques de l'environnement dans lequel le logiciel est exécuté pour caractériser son comportement tel qu'il est perçu par ses utilisateurs. De ce fait, si on considère plusieurs environnements d'utilisation du logiciel on peut évaluer de cette façon, les mesures de la sûreté de fonctionnement du logiciel qui sont propres à chacun des environnements considérés. Ce dernier résultat n'est valable que si la variation de l'environnement d'utilisation du logiciel se traduit uniquement par une variation de son taux d'exécution sans qu'il y ait une modification importante de la trajectoire dans l'espace des entrées du logiciel dans les différents environnements considérés.

III.4 APPLICATION DU MODÈLE à DES DONNéES RéELLES

Les données collectées à partir du système peuvent être fournies sous deux formes :

- nombre d'exécutions entre défaillances : ces données correspondent aux "intervalles entre défaillances",
- nombre de défaillances par séquence d'exécution : ces données sont similaires aux données "nombre de défaillances par unité de temps" considérées en temps continu.

Le deuxième type de données est plus facile à collecter dans un contexte industriel car il n'est pas nécessaire d'indiquer l'ordre dans lequel les exécutions ont été effectuées, seul le nombre de défaillances observées au cours d'une séquence d'exécutions doit être précisé.

L'application du modèle hyperexponentiel en temps discret aux données collectées s'effectue de la même façon que dans le cas des modèles en temps continu, c'est-à-dire en trois phases : utilisation d'une procédure d'inférence pour l'estimation des paramètres, évaluation des mesures de fiabilité du logiciel en utilisant les valeurs estimées des paramètres et application d'un ou plusieurs critères de validation pour analyser la qualité des estimations effectuées par le modèle.

III.4.1 Estimation des paramètres du modèle

L'estimation des paramètres du modèle peut être effectuée par les deux méthodes classiques : les moindres carrés et le maximum de vraisemblance. L'estimation par la méthode des moindres carrés est plus simple à mettre en œuvre : il suffit d'utiliser une procédure numérique pour évaluer les valeurs des paramètres qui minimisent l'écart entre la mesure de fiabilité choisie, évaluée par le modèle, et la valeur correspondante évaluée à partir des données collectées sur le logiciel. L'estimation par la deuxième méthode nécessite l'évaluation de la fonction de vraisemblance.

Nous donnons dans la suite l'expression de la fonction de vraisemblance relative au modèle hyperexponentiel en temps discret et correspondant aux deux types de données collectées sur le logiciel.

III.4.1.1 Données : "nombre d'exécutions entre défaillances"

Supposons que l'on a observé r défaillances avec (m_1, m_2,……, m_r) les instants d'occurrence des défaillances observées. La fonction de vraisemblance associée à l'échantillon (m_1, m_2,……, m_r) est donnée par la fonction densité de probabilité conjointe notée $f(m_1, m_2,\ldots\ldots, m_r)$ [Lawless 82].

La probabilité de défaillance à l'exécution i, P(i), est supposée par hypothèse indépendante de l'occurrence de défaillances au cours des i-1 exécutions précédentes. Compte tenu de cette hypothèse, la fonction de vraisemblance s'écrit :

$$L = f(m_1, m_2,\ldots\ldots, m_r) = \prod_{i=1}^{r} f(n_i \mid m_{i-1}) \qquad \text{(III.43)}$$

où n_i est le nombre d'exécutions effectuées entre l'occurrence des défaillances i-1 et i.

Il vient à partir des relations III.19 et III.23 :

$$L = \left\{ P(m_1) \prod_{j=1}^{m_1-1} [1-P(j)] \right\} * \left\{ P(m_2) \prod_{j=m_1+1}^{m_2-1} [1-P(j)] \right\} *\ldots* \left\{ P(m_r) \prod_{j=m_{r-1}+1}^{m_r-1} [1-P(j)] \right\} \qquad \text{(III.44)}$$

La relation III.44 peut se mettre sous une forme plus simple :

$$L = \prod_{j=1}^{m_r} [1-P(j)] \prod_{j=1}^{r} \frac{P(m_j)}{1-P(m_j)} \qquad \text{(III.45)}$$

Pour estimer les paramètres θ, p_{sup} et p_{inf} du modèle discret, il faudra évaluer le maximum de la fonction de vraisemblance. Généralement, on considère plutôt le logarithme de la fonction de vraisemblance. Celui-ci, noté LL, obtenu à partir de la relation III.45, est donné par :

$$LL = \sum_{j=1}^{r} Ln\left\{ \theta p_{sup}(1-p_{sup})^{m_j-1} + \overline{\theta}\, p_{inf}(1-p_{inf})^{m_j-1} \right\} - \sum_{j=1}^{r-1} Ln\left\{ \theta(1-p_{sup})^{m_j} + \overline{\theta}(1-p_{inf})^{m_j} \right\} \qquad \text{(III.46)}$$

L'expression de la fonction de vraisemblance est très complexe et ne permet pas d'obtenir analytiquement les valeurs des paramètres θ, p_{sup} et p_{inf} donnant le maximum de la fonction de vraisemblance. Ces paramètres peuvent être obtenus numériquement en utilisant les techniques classiques d'optimisation.

III.4.1.2 Données : "Nombre de défaillances par séquence d'exécution"

Considérons maintenant le cas où les données sont collectées sous la forme (s_i,y_i), i = 1,2,…k où y_i est le nombre cumulé de défaillances observées après s_i exécutions du logiciel.

Pour évaluer la fonction de vraisemblance, il faut calculer la densité de probabilité conjointe associée à l'échantillon $\{(s_1,y_1), (s_2,y_2) \ldots(s_k,y_k)\}$. Cette fonction peut être évaluée si on connaît

la distribution de la variable aléatoire Y_n, définie dans le paragraphe III.2.1, qui correspond au nombre d'occurrences de défaillances au cours de n exécutions du logiciel. La loi de Y_n est très complexe, cependant, on peut utiliser l'approximation de la loi de Y_n par une loi de Poisson.

Si on utilise l'hypothèse d'indépendance, la fonction de vraisemblance relative à l'échantillon $\{(s_1,y_1), (s_2,y_2) \ldots (s_k,y_k)\}$ s'écrit :

$$L = \exp(- H(s_k)) \prod_{i=1}^{k} \frac{\{H(s_i) - H(s_{i-1})\}^{y_i - y_{i-1}}}{(y_i - y_{i-1})!} \qquad \text{(III.47)}$$

Le logarithme de la fonction de vraisemblance est donné par :

$$LL = \sum_{j=1}^{k} (y_i - y_{i-1}) \, Ln\{H(s_i) - H(s_{i-1})\} - Ln\{(y_i - y_{i-1})!\} - H(s_k) \qquad \text{(III.48)}$$

$H(s_i)$ étant donné par la relation (III.10).

III.4.2 Critères de validation

Mis à part le critère de Kolmogorov-Smirnof qui est basé sur l'hypothèse que la fonction de distribution relative à la variable aléatoire caractérisant le processus modélisé est continue, tous les autres critères cités dans le paragraphe I.5.2 peuvent être appliqués pour apprécier la qualité du modèle hyperexponentiel en temps discret. En particulier, on peut utiliser le critère des résidus, pour comparer les valeurs estimées par le modèle au comportement observé du logiciel, ou bien appliquer le critère de la vraisemblance préquentielle et le critère de Akaike pour comparer le modèle à d'autres modèles de croissance de fiabilité en temps discret.

III.4.3 Application du modèle à deux exemples de données

Pour illustrer l'application du modèle en temps discret à des données d'expérience, nous considérerons deux ensembles de données qui ont été publiés respectivement dans [Yamada 84] et [Tohma 91]. Ces données sont les seules que nous avons trouvées dans la littérature. Les données publiées sont généralement soit, sous forme d'intervalles de temps entre défaillances soit, sous forme de nombre de défaillances par unité de temps.

III.4.3.1 Données de Yamada

Les données considérées ont été collectées au cours de la validation d'un logiciel d'application écrit en assembleur et en PL/I dont le volume est de l'ordre de 50 000 lignes d'instructions. Les données collectées sont présentées sur la figure III.7 où :

- i correspond à la i ème séquence d'exécution du logiciel,
- s_i est le nombre cumulé d'exécutions effectuées jusqu'à la séquence i,
- y_i est le nombre cumulé de défaillances observées après i séquences d'exécution du logiciel.

Durant la période considérée, 73 défaillances ont été observées au cours de 773 exécutions du logiciel[1].

i	s_i	y_i	$y_i - y_{i-1}$
1	14	5	5
2	28	8	3
3	57	18	10
4	71	20	2
5	114	27	7
6	143	29	2
7	186	31	2
8	243	39	8
9	286	42	3
10	300	47	5
11	358	52	5
12	393	53	1
13	457	60	7
14	571	63	3
15	600	66	3
16	743	69	3
17	758	71	2
18	773	73	2

Figure III.7 : Données de Yamada

Les informations collectées sont insuffisantes pour appliquer le modèle hyperexponentiel afin de suivre l'évolution du MTTF du logiciel. En effet, compte tenu de la relation III.29, l'évaluation du MTTF associé à l'occurrence de la défaillance i du logiciel nécessite la connaissance de l'instant d'occurrence de la défaillance i-1.

Nous nous limiterons donc à l'application du modèle hyperexponentiel en temps discret pour suivre l'évolution du nombre cumulé de défaillances en fonction du nombre d'exécutions du logiciel.

Sur les figures III.8 et III.9 nous donnons respectivement les résultats de l'application du modèle en réplicatif et en prévisionnel pas-à-pas aux données considérées :

- N(n) est le nombre cumulé de défaillances observées après n exécutions du logiciel,
- H(n) est le nombre cumulé de défaillances estimées par le modèle hyperexponentiel en temps discret.

[1] Ces données ont été échantillonnées à partir de la courbe, présentée dans [Yamada 84], donnant l'évolution du nombre cumulé de défaillances observées en fonction du nombre d'exécutions effectuées.

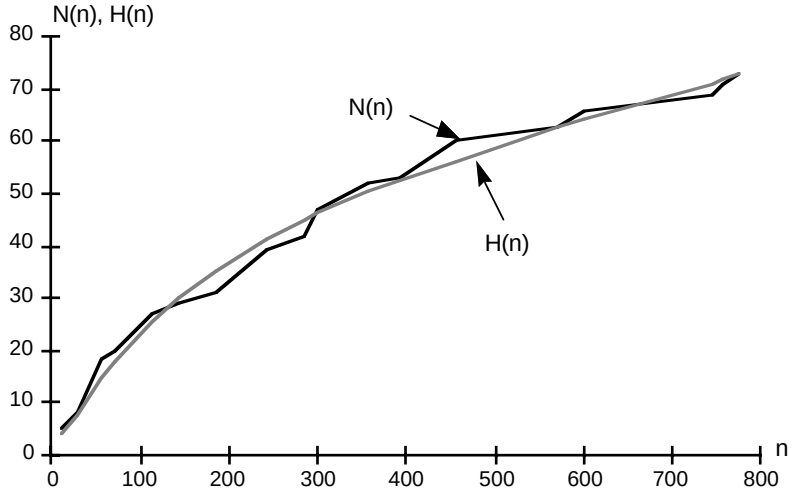


Figure III.8 : Application en réplicatif du modèle hyperexponentiel en temps discret aux données de Yamada

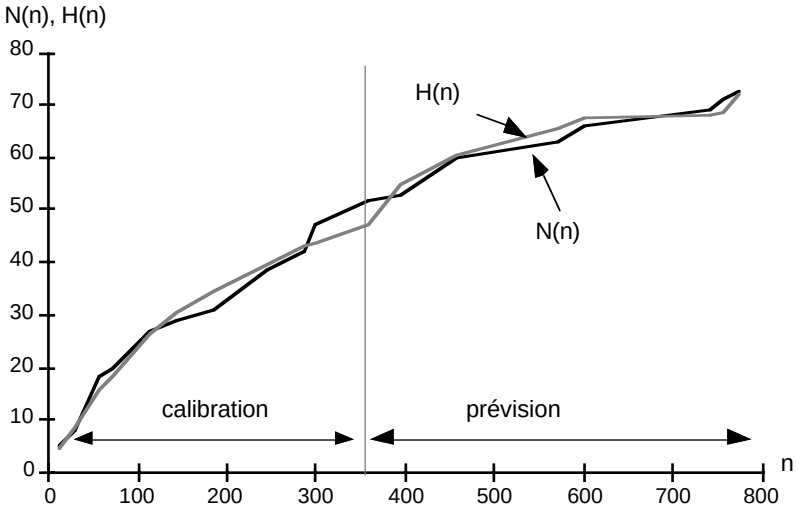


Figure III.9 : Application en prévisionnel du modèle hyperexponentiel en temps discret aux données de Yamada

Compte tenu de l'allure d'évolution de la courbe N(n), on peut remarquer que la fiabilité du logiciel croît globalement en fonction du nombre d'exécutions effectuées au cours de la période considérée. Une telle croissance de fiabilité est bien prise en compte par le modèle hyperexponentiel en temps discret aussi bien en réplicatif qu'en prévisionnel.

III.4.3.2 Données de Tohma

Les données considérées ont été collectées au cours de la phase de validation d'un logiciel pour lequel on a observé 137 défaillances au cours de 418 exécutions. Ces données sont présentées sur la figure III.10 où i, s_i et y_i sont définies de la même façon que dans le cas précédent.

Si on applique le modèle hyperexponentiel en temps discret de la même manière que dans le cas précédent en réplicatif et en prévisionnel, on obtient les figures III.11 et III.12.

On peut remarquer que dans ce cas aussi le modèle suit bien l'évolution du nombre cumulé de défaillances.

L'analyse de la courbe N(n) montre que la pente de la courbe tend à être constante ce qui traduit une évolution stabilisée du logiciel en fonction du nombre d'exécutions effectuées.

i	s_i	y_i	$y_i - y_{i-1}$
1	33	30	30
2	51	45	15
3	59	47	2
4	74	48	1
5	105	55	7
6	106	55	0
7	163	77	22
8	190	79	2
9	225	84	5
10	251	96	12
11	287	110	14
12	315	115	5
13	337	117	2
14	341	117	0
15	349	124	7
16	354	127	3
17	381	127	0
18	387	127	0
19	393	127	0
20	397	127	0
21	398	132	5
22	403	134	2
23	408	137	3
24	416	137	0
25	418	137	0

Figure III.10 : Données de Tohma

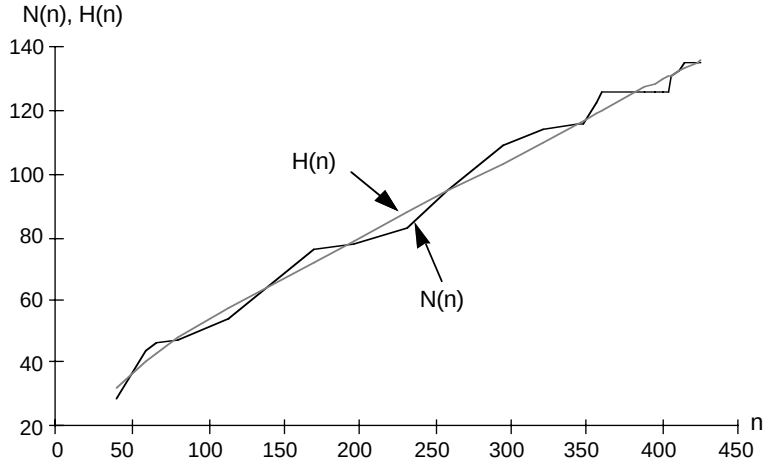


Figure III.11 : Application du modèle en réplicatif aux données de Tohma

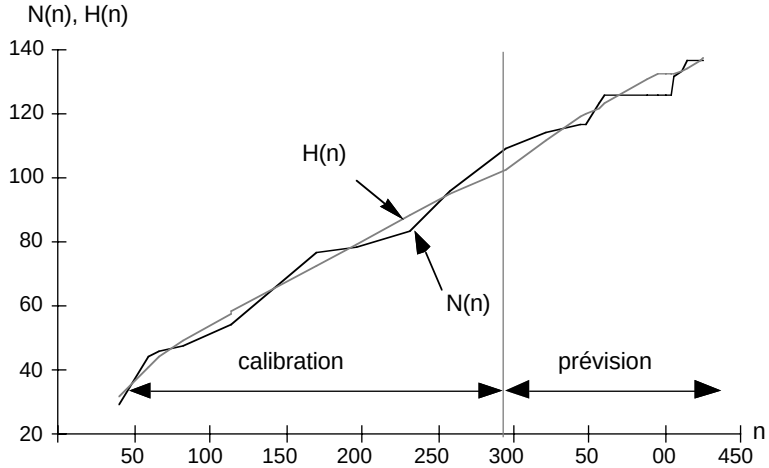


Figure III.12 : Application du modèle en prévisionnel aux données de Tohma

III.4.4 Conclusion

Les deux exemples de données que nous venons de traiter ont permis d'illustrer l'application du modèle hyperexponentiel en temps discret à des données réelles pour suivre l'évolution du nombre cumulé de défaillances en fonction du nombre d'exécutions du logiciel. Le volume de données traitées n'est pas très important et les informations fournies sont insuffisantes pour appliquer le modèle afin d'évaluer d'autres mesures de fiabilité du logiciel (MTTF par exemple). Une collecte de données plus importante est nécessaire afin de permettre une validation plus significative du modèle.

CONCLUSION

Dans ce chapitre, nous avons défini un modèle de croissance de fiabilité en temps discret qui modélise l'évolution de la probabilité de défaillance à l'exécution en fonction du nombre d'exécutions effectuées et nous avons établi les expressions des mesures de fiabilité qui lui sont associées. Nous avons remarqué que le modèle possède des propriétés similaires à celles du modèle hyperexponentiel en temps continu en particulier en ce qui concerne le lien qui existe entre probabilité non conditionnelle de défaillance à l'exécution et fonction de hasard par analogie avec le lien entre intensité de défaillance et taux de défaillance. En se basant sur ces propriétés nous avons associé au modèle une interprétation markovienne qui consiste à représenter le modèle sous forme d'une chaîne de Markov en temps discret. Afin de modéliser le comportement du logiciel tel qu'il est perçu par ses utilisateurs, nous avons étudié la transformation du modèle en temps discret en un modèle en temps continu permettant d'évaluer les mesures de fiabilité du logiciel en fonction du temps. Une telle transformation a été effectuée en tenant compte du taux d'exécution du logiciel qui est caractéristique de l'environnement dans lequel il est exécuté. Cette dernière propriété du modèle lui permet de tenir compte de certaines caractéristiques de l'environnement d'utilisation du logiciel dans l'évaluation des mesures de sûreté de fonctionnement.

Dans ce chapitre, nous avons étudié le logiciel selon une approche "boîte noire" en supposant que la variation de l'environnement se traduit uniquement par une modification du taux d'exécution du logiciel global. Nous présenterons dans le chapitre V une approche plus générale dans laquelle on tient compte de la structure du logiciel pour caractériser son environnement d'utilisation et évaluer ses mesures de sûreté de fonctionnement en fonction de celles de ses différents composants.

DEUXIÈME PARTIE

MODÉLISATION DE LA SÛRETÉ DE FONCTIONNEMENT DE SYSTÈMES MULTI-COMPOSANT

Avec l'augmentation croissante de la complexité des systèmes informatiques, il devient indispensable de développer des méthodes d'évaluation et de prévision de fautes permettant de tenir compte de la structure des systèmes afin :

- d'évaluer l'impact des différents composants d'un système sur l'évolution de ses mesures de sûreté de fonctionnement,
- d'identifier les composants les moins fiables afin de mettre l'accent sur ces composants durant la validation du système considéré et au cours de sa vie opérationnelle, ou bien sur des composants qui sont de même nature lors du développement de futurs systèmes semblables.

La modélisation de la sûreté de fonctionnement des systèmes informatiques selon une vue "boîte blanche" a été essentiellement axée sur l'évaluation du matériel. Cependant, de telles études se sont limitées à la modélisation des processus de défaillance et de restauration du matériel vis-à-vis de l'activation de fautes physiques sans tenir compte des fautes de conception internes. L'évaluation de la sûreté de fonctionnement du matériel s'est donc limitée à l'étude du comportement en fiabilité stabilisée des systèmes en ignorant que la fiabilité du matériel croît aussi au cours de la vie des systèmes en raison de la présence et de l'élimination des fautes de conception.

Les approches "boîte blanche" portant sur l'évaluation du logiciel ont donné lieu à beaucoup moins de travaux et se sont généralement limitées aux logiciels en fiabilité stabilisée qu'il s'agisse de logiciels non tolérant aux fautes [Littlewood 79, Cheung 80, Laprie 83-b, Siegrist 88] ou de logiciels tolérant aux fautes [Hecht 79, Grnarov 80, Laprie 84-b, Arlat 88-a]. Les approches "boîte blanche" prenant en compte le phénomène de croissance de fiabilité sont très rares et se limitent, à notre connaissance, au modèle incrémental d'IBM [Currit 86], au modèle hyperexponentiel d'Ohba [Ohba 84] et au modèle hypergéométrique de Tohma [Tohma 89]. Ces derniers modèles ne considèrent que la fiabilité des systèmes et ne tiennent pas compte de la disponibilité.

Par ailleurs, actuellement, les études portant sur la modélisation de la sûreté de fonctionnement de systèmes constitués de composants matériels et logiciels sont très rares ([Costes 78, Laprie 84-a, Sumita 86, Stark 87, Laprie 91-a]). La raison en est que la modélisation du matériel et la modélisation du logiciel ont été pendant longtemps considérées comme des disciplines distinctes [Laprie 89]. Néanmoins, au cours de la vie opérationnelle d'un système informatique, le souhait des utilisateurs est de disposer de résultats concernant

l'évaluation de la fiabilité et de la disponibilité du matériel et du logiciel en considérant aussi bien les fautes physiques que les fautes de conception.

Le problème que nous venons de soulever permet d'introduire le sujet que nous traitons dans la deuxième partie de ce mémoire. En effet, dans cette partie, nous nous consacrons à la modélisation de la croissance de sûreté de fonctionnement d'un système multi-composant à partir de celle de ses différents composants. Les deux modèles que nous avons présentés dans la première partie de ce mémoire permettent de caractériser le comportement d'un système selon une vue "boîte noire" en tenant compte du phénomène de croissance de fiabilité. En considérant un système multi-composant, on peut utiliser les modèles proposés pour caractériser, en temps continu ou en temps discret, le comportement de chacun des composants du système. Dans la deuxième partie de ce mémoire, nous proposons une approche de modélisation qui permet de construire le modèle d'un système à partir des modèles de comportement de ses différents composants en tenant compte du phénomène de croissance de fiabilité.

Cette partie est constituée de deux chapitres. Dans le premier chapitre (chapitre IV du mémoire), nous définissons une approche, basée sur le modèle hyperexponentiel en temps continu, permettant de modéliser la **croissance de fiabilité** et la **croissance de disponibilité** de systèmes constitués de composants **matériels** et **logiciels**. Dans le deuxième chapitre (chapitre V), nous proposons une approche, basée sur le modèle hyperexponentiel en temps discret, visant à modéliser le comportement de logiciels constitués de composants qui sont exécutés de façon séquentielle, en fonction du nombre d'exécutions effectuées. Nous abordons également dans ce chapitre, le problème de la prise en compte des caractéristiques de l'environnement d'utilisation d'un système dans l'évaluation de ses mesures de sûreté de fonctionnement.

CHAPITRE IV

APPROCHE GLOBALE DE MODéLISATION DE SYSTÈMES MULTI-COMPOSANT BASéE SUR LE MODÈLE HYPEREXPONENTIEL EN TEMPS CONTINU

INTRODUCTION

La méthode la plus naturelle pour maîtriser la complexité d'un système et modéliser son comportement est de le décomposer, de décrire le comportement de chacun de ses composants par un sous-modèle et de construire finalement le modèle du système global en agrégeant les sous-modèles et en tenant compte des dépendances qui existent entre les comportements des différents composants. On distingue principalement deux types de dépendances [Arlat 88-b] :

- les dépendances fonctionnelles essentiellement liées à la structure du système,
- les dépendances stochastiques induites par la représentation du comportement du système.

Les dépendances stochastiques entre les composants d'un système qui sont liées à la sûreté de fonctionnement peuvent résulter [Laprie 83-a] :

- de la nature du processus d'exécution : certains composants du système ne sont activés, par exemple, que suite à la défaillance d'autres composants,
- de la nature des fautes : les défaillances des différents composants du système peuvent résulter par exemple de l'activation de fautes indépendantes ou de fautes corrélées [Laprie 90-b]. Contrairement aux fautes indépendantes, les fautes corrélées peuvent entraîner la défaillance simultanée de plusieurs composants du système.
- de la nature des procédures de restauration de service et de traitement d'erreurs : pour certains systèmes, la restauration d'un composant défaillant ne peut être effectuée qu'après la restauration d'autres composants du système. Par ailleurs, la distribution du temps jusqu'à restauration d'un composant peut dépendre de la nature des fautes activées, des conséquences des défaillances sur l'environnement d'utilisation du système et des ressources disponibles pour effectuer la restauration du système.

La modélisation du comportement d'un système par un graphe d'états, et en particulier par une chaîne de Markov, est la méthode la plus courante et la plus adaptée pour prendre en compte les deux types de dépendances [Laprie 75, Pagès 80].

La description du comportement d'un système par une chaîne de Markov suppose que les événements qui régissent les transitions entre les états du système (défaillances, restaurations sollicitations) sont distribués selon des lois exponentielles à taux constant. Cet outil de modélisation est donc bien adapté pour représenter le comportement d'un système en fiabilité stabilisée.

Notre objectif est de prendre en compte la croissance de sûreté de fonctionnement des composants d'un système pour l'évaluation de sa sûreté de fonctionnement. La représentation markovienne du modèle hyperexponentiel introduite dans le chapitre II permet de décrire la croissance de sûreté de fonctionnement d'un système en boîte noire (ou d'un composant) par la **transformation** d'une chaîne de Markov en fiabilité stabilisée en une autre chaîne de Markov caractérisant le phénomène de croissance de sûreté de fonctionnement.

On peut ainsi représenter la croissance de sûreté de fonctionnement de chaque composant d'un système par une chaîne de Markov. Notre objectif maintenant est **d'étendre la transformation,** définie dans le chapitre II, au cas d'un système multi-composant. Le problème à résoudre est le suivant : "comment construire une chaîne de Markov caractérisant la croissance de sûreté de fonctionnement d'un système, à partir des chaînes de Markov modélisant la croissance de sûreté de fonctionnement de ses différents composants". C'est l'objet de l'approche que nous présentons dans ce chapitre.

Ce chapitre est constitué de trois paragraphes. Dans le premier paragraphe, nous introduisons l'approche de transformation à travers deux exemples simples. Dans le deuxième paragraphe, nous définissons une approche générale basée sur les réseaux de Petri stochastiques généralisés permettant de modéliser la croissance de sûreté de fonctionnement d'un système à partir de celle de ses différents composants. L'approche proposée est ensuite illustrée, dans le troisième paragraphe, sur deux systèmes : le premier est un système constitué de deux composants redondants intégrant des mécanismes internes de détection et de recouvrement d'erreurs et le deuxième est un logiciel tolérant aux fautes.

Les travaux que nous présentons dans ce chapitre ont fait l'objet de publications [Laprie 90-a, Laprie 91-a, Kanoun 91-c]. Les fondements de l'approche de modélisation et les principes de la technique de transformation ont été publiés dans [Laprie 90-a, Laprie 91-a] où notre contribution personnelle a porté essentiellement sur la définition et la mise en œuvre de la technique de transformation par les réseaux de Petri stochastiques généralisés. Dans [Kanoun 91-c], nous avons, d'une part, étendu les possibilités de la technique de transformation dans l'objectif de la rendre plus mécanique et de faciliter sa mise en œuvre dans le cas de systèmes complexes et, d'autre part, appliqué cette approche pour la modélisation et l'évaluation de la croissance de fiabilité de logiciels tolérant aux fautes en considérant les trois approches principales pour la tolérance aux fautes logicielles : les blocs de recouvrement, la programmation en N-versions, et la programmation N-auto-testable. Dans ce chapitre, nous présenterons uniquement les résultats qui sont relatifs à l'application de l'approche de modélisation à un logiciel constitué d'un bloc de recouvrement.

IV.1 INTRODUCTION DE L'APPROCHE DE TRANSFORMATION

Pour introduire l'approche de transformation, nous considérerons deux exemples simples. Le premier exemple concerne un système redondant constitué de deux composants identiques qui s'exécutent en parallèle et le deuxième exemple porte sur l'étude du comportement d'un système constitué de deux composants différents qui s'exécutent en parallèle et qui sont tels que le système défaille si l'un des ses deux composants défaille (système série).

IV.1.1 Exemple 1 : système redondant

Considérons un système constitué de deux composants identiques qui sont exécutés en parallèle. Les deux composants, supposés stochastiquement indépendants, sont soumis à deux processus : défaillance et restauration. Soient λ et μ respectivement le taux de défaillance et le taux de restauration associés à chaque composant.

Le comportement en fiabilité stabilisée de chaque composant du système est alors décrit par la chaîne de Markov présentée sur la figure IV.1.a.

Compte tenu de l'indépendance stochastique du comportement des deux composants du système, on peut utiliser une technique algébrique basée sur le produit et la somme de Kronecker, dont les définitions sont rappelées dans l'Annexe, pour obtenir directement la chaîne de Markov caractérisant le comportement en fiabilité stabilisée du système considéré à partir des chaînes de Markov caractéristiques du comportement de chacun de ses composants [Amoia 81, Mazars 81].

Soient $\mathbb{A}_C$ la matrice des taux de transition associée à la chaîne de Markov décrivant le comportement de chacun des composants du système, et $\mathbb{A}_S$ la matrice des taux de transition relative au système global. Compte tenu des propriétés de la somme et du produit de Kronecker présentées dans l'Annexe, on a :

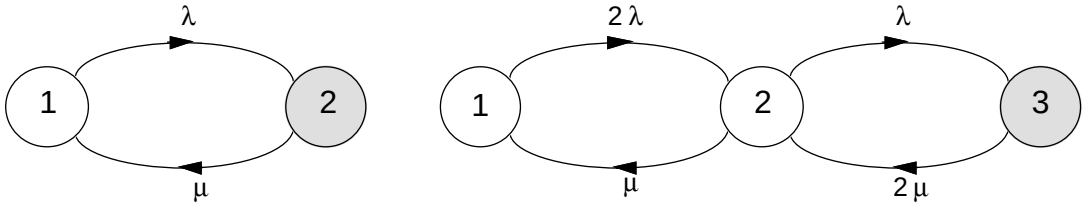
$$\mathbb{A}_S = \mathbb{A}_C \oplus \mathbb{A}_C \qquad \text{(IV.1)}$$

où $\oplus$ est l'opérateur "somme de Kronecker".

Pour l'exemple considéré, la matrice de transition $\mathbb{A}_C$ associée à la chaîne de Markov de la figure IV.1-a caractérisant le comportement en fiabilité stabilisée d'un composant est la suivante :

$$\mathbb{A}_C = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix} \qquad \text{(IV.2)}$$

Si on applique la relation IV.1 et le théorème de Kemeny-Snell d'agrégation des états d'une chaîne de Markov [Kemeny 59], on obtient la chaîne de Markov présentée sur la figure IV.1-b qui caractérise le comportement du système en fiabilité stabilisée.



a) Composant b) Système

Figure IV.1 : Chaînes de Markov en fiabilité stabilisée d'un composant et du système

Le théorème de Kemeny-Snell permet de réduire l'espace des états d'une chaîne de Markov dans le cas où certains états du système sont équivalents. C'est le cas ici puisqu'on a considéré que les deux composants du système sont identiques ; la défaillance de l'un des deux composants se traduit par une transition du système vers le même état : l'état 2 qui correspond au cas où un composant est actif et l'autre composant est défaillant.

Considérons maintenant le cas où la fiabilité de chacun des composants du système croît au cours de son cycle de vie. On suppose que la croissance de fiabilité de chacun des composants peut être décrite par un modèle hyperexponentiel de paramètres ω, ζ_{sup} et ζ_{inf}. Si on utilise la représentation markovienne du modèle hyperexponentiel introduite dans le chapitre II (que nous rappelons sur la figure IV.2-a), alors, sous l'hypothèse de croissance de fiabilité, la matrice des taux de transition caractéristique du comportement de chacun des composants est donnée par :

$$\mathbb{\Lambda}_C = \begin{pmatrix} -\zeta_{sup} & 0 & \zeta_{sup} \\ 0 & -\zeta_{inf} & \zeta_{inf} \\ 0 & \mu & -\mu \end{pmatrix} \qquad \text{(IV.3)}$$

Soit $\mathbb{P}_C(0)$ le vecteur des probabilités initiales d'occupation des états associé à un composant :

$$\mathbb{P}_C(0) = \begin{pmatrix} \omega & \overline{\omega} & 0 \end{pmatrix} \qquad \text{(IV.4)}$$

Le vecteur des probabilités initiales d'occupation des états associé au système est donné par (cf l'Annexe) :

$$\mathbb{P}_S(0) = \mathbb{P}_C(0) \otimes \mathbb{P}_C(0) \qquad \text{(IV.5)}$$

où $\otimes$ est l'opérateur "produit de Kronecker".

On peut alors obtenir, à partir des relations (IV.1), (IV.3) et (IV.5), la matrice des taux de transition relative à la chaîne de Markov qui décrit la croissance de fiabilité du système considéré. L'application du théorème de Kemeny-Snell d'agrégation d'états conduit à la chaîne de Markov du système présentée sur la figure IV.2-b.

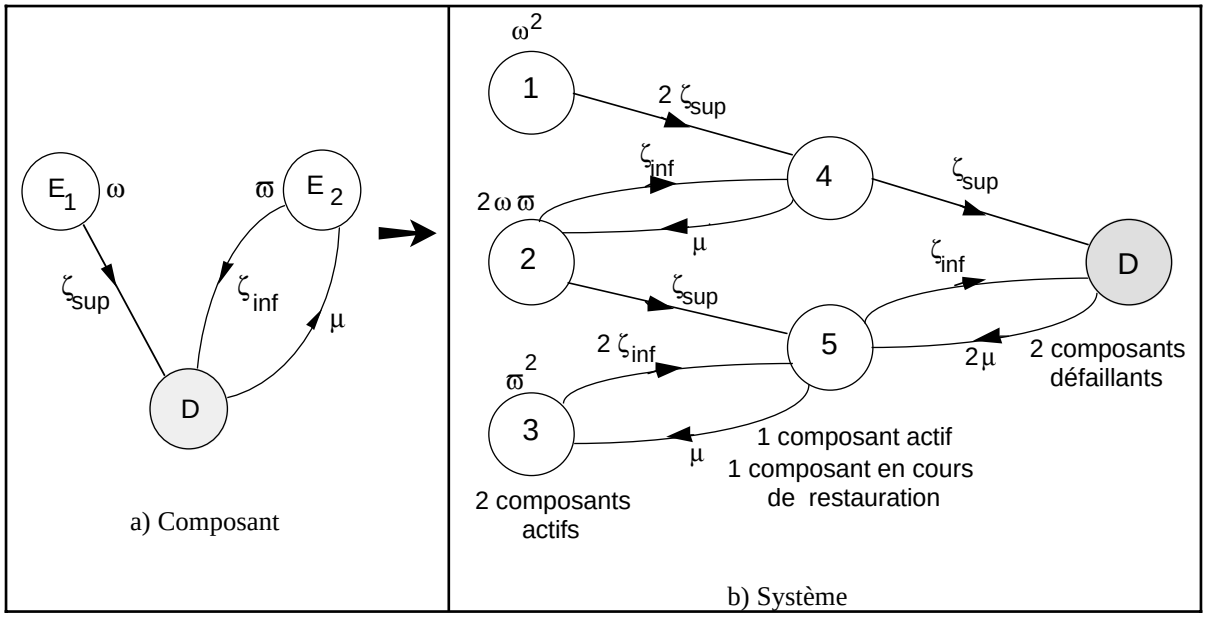


Figure IV.2 : Chaîne de Markov d'un composant et du système redondant en croissance de fiabilité

La chaîne de Markov ainsi obtenue résulte de la transformation de la chaîne de Markov du système en fiabilité stabilisée (figure V.1-b) pour prendre en compte la croissance de fiabilité de ses différents composants.

On peut traiter maintenant la chaîne de Markov transformée par les techniques classiques de traitement des chaînes de Markov, pour évaluer les mesures de sûreté de fonctionnement du système caractérisant son comportement dans le cas où la sûreté de fonctionnement de ses composants croît au cours de son cycle de vie.

Considérons par exemple la fonction indisponibilité $\overline{A}(t)$ du système.

Avec l'hypothèse $\mu >> \zeta_{sup} > \zeta_{inf}$, on obtient :

$$\overline{A}(t) = \frac{\zeta_{inf}^2}{\mu^2} + 2\,\frac{\omega\zeta_{inf}(\zeta_{sup}-\zeta_{inf})}{\mu^2}\,e^{-\zeta_{sup}t} + \frac{\omega^2\,(\zeta_{sup}-\zeta_{inf})^2}{\mu^2}\,e^{-2\zeta_{sup}t} - 2\,\frac{(\omega\zeta_{sup}+\overline{\omega}\,\zeta_{inf})^2}{\mu^2}\,e^{-\mu t} \qquad \text{(IV.6)}$$

A partir de la relation IV.6, on peut aussi obtenir l'expression $\overline{A}^*(t)$ de l'indisponibilité qui correspond au cas où les composants du système sont en fiabilité stabilisée. Il suffit de poser $\omega = 0$ et $\zeta_{inf} = \lambda$. On trouve alors :

$$\overline{A}^*(t) = \frac{\lambda^2}{\mu^2} - 2\,\frac{\lambda^2}{\mu^2}\,e^{-\mu t} \qquad \text{(IV.7)}$$

La relation IV.7 peut être obtenue également en traitant directement la chaîne de Markov du système en fiabilité stabilisée (figure IV.1-b).

Etant donné qu'en général $\mu t >> 1$, on peut négliger les termes en $e^{-\mu t}$. Si on compare, avec cette hypothèse, les relations IV.6 et IV.7, on peut remarquer que dans le cas où on ne tient pas

compte du phénomène de croissance de fiabilité, l'indisponibilité tend très vite vers une valeur constante : le régime asymptotique est atteint à un instant t qui est de l'ordre de quelques $\frac{1}{\mu}$.

Cependant, en tenant compte du phénomène de croissance de fiabilité, on remarque que $\overline{A}(t)$ décroît progressivement dans le temps avant d'atteindre un comportement asymptotique stable : celui-ci est atteint beaucoup plus lentement que dans le cas où on suppose un comportement du système en fiabilité stabilisée : l'instant de stabilisation de la courbe $\overline{A}(t)$ est de l'ordre de quelques $\frac{1}{\zeta_{sup}}$. La décroissance de la courbe $\overline{A}(t)$ est synonyme d'une croissance de la disponibilité du système.

IV.1.2 Exemple 2 : système série

Considérons un système constitué de deux composants différents stochastiquement indépendants qui s'exécutent en même temps. Chacun des deux composants est soumis à deux processus : défaillance et restauration, et la défaillance de l'un d'eux conduit à la défaillance du système.

Soient λ_1, μ_1 et λ_2, μ_2 les taux de défaillance et de restauration des composants C_1 et C_2 constituant le système. C_1 et C_2 peuvent être aussi bien des composants matériels que des composants logiciels.

L'application à ce système de l'approche de transformation est résumée sur la figure IV.3. La figure IV.3-a correspond à la chaîne de Markov caractérisant le comportement du système en fiabilité stabilisée et la figure IV.3-b représente la chaîne de Markov du système obtenue après l'application de la transformation où ω_j, ζ_{supj} et ζ_{infj} sont les paramètres du modèle hyperexponentiel qui caractérisent la croissance de fiabilité du composant C_j.

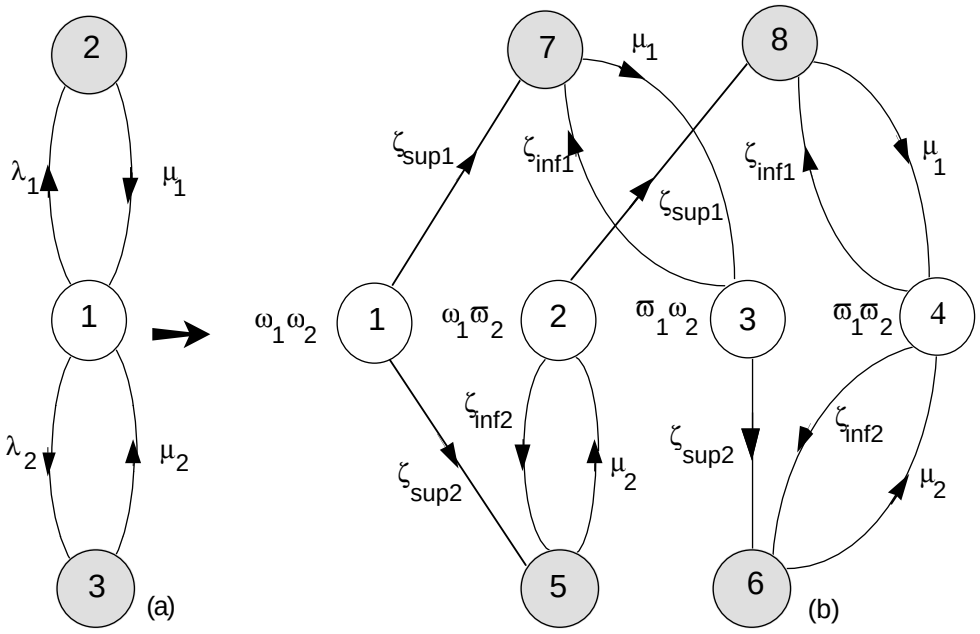


Figure IV.3 : Application de la transformation à un système série

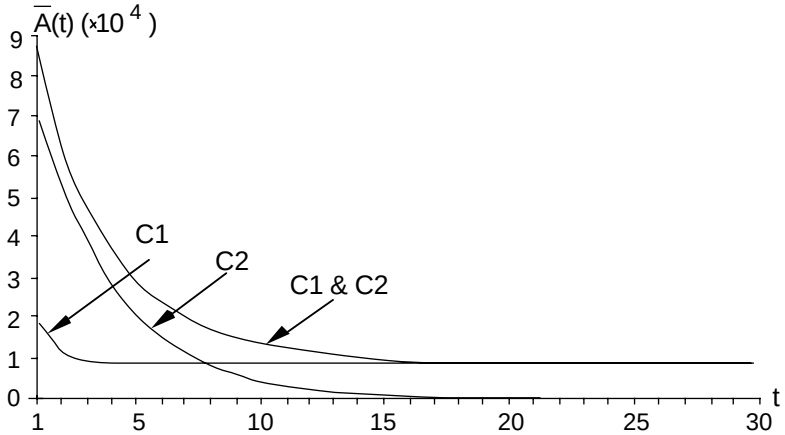
Le traitement de la chaîne de Markov transformée conduit à l'expression suivante de l'indisponibilité $\overline{A}(t)$ du système :

$$\overline{A}(t)=1-\prod_{j=1}^{2}\left\{1-\frac{\zeta_{infj}}{\mu_j}-\frac{\omega_j(\zeta_{supj}-\zeta_{infj})}{\mu_j}\,e^{-\zeta_{supj}t}+\frac{\omega_j\zeta_{supj}+\overline{\omega_j}\,\zeta_{infj}}{\mu_j}\,e^{-\mu_j t}\right\} \qquad \text{(IV.8)}$$

On peut alors étudier la variation de l'indisponibilité du système en fonction des paramètres caractérisant la croissance de sûreté de fonctionnement de ses deux composants. Sur la figure IV.4, on présente un exemple d'évolution de la fonction indisponibilité $\overline{A}(t)$ donnée par la relation IV.8. La courbe C1 (respectivement C2) caractérise l'évolution de l'indisponibilité du composant C_1 (respectivement C_2) et la courbe C1&C2 donne l'évolution de l'indisponibilité du système en tenant compte de la croissance de fiabilité de ses deux composants.

Pour les valeurs choisies des paramètres, on peut remarquer qu'au début de la période considérée, c'est le composant C_2 qui est essentiellement à l'origine de l'indisponibilité observée du système. Cependant, cette tendance est inversée au voisinage du comportement asymptotique.

L'étude de sensibilité que nous venons d'effectuer est intéressante, en particulier, pour les concepteurs des systèmes car elle permet de suivre l'évolution de la sûreté de fonctionnement d'un système en fonction de l'évolution de la sûreté de fonctionnement de ses différents composants. En fonction des résultats obtenus, les concepteurs peuvent être amenés à modifier le processus de validation et de maintenance du système considéré pour mettre l'accent plus particulièrement sur certains composants du système.



Composant C1	ω_1 =0,5	ζ_{sup1} =10^{-2}/h	ζ_{inf1}=10^{-3}/h	μ_1 =10 /h
Composant C2	ω_2=0,5	ζ_{sup2} =2 10^{-3}/h	ζ_{inf2} =10^{-5}/h	μ_2 =1 /h

Figure IV.4 : indisponibilité d'un système formé de deux composants

IV.1.3 Conclusion

Les deux exemples que nous venons de traiter ont permis d'introduire l'approche que nous proposons pour modéliser la croissance de sûreté de fonctionnement d'un système à partir de celle de ses différents composants. Cette approche comporte trois étapes :

- une première étape portant sur la **construction** d'une chaîne de Markov décrivant le comportement en fiabilité stabilisée du système à partir de celui de ses différents composants,
- une deuxième étape consacrée à la **transformation** de la chaîne de Markov du système en fiabilité stabilisée pour générer une chaîne de Markov décrivant la croissance de sûreté de fonctionnement du système,
- une troisième étape consacrée au **traitement** de la chaîne de Markov transformée pour évaluer les mesures de sûreté de fonctionnement du système.

Par rapport à l'approche classique de modélisation des systèmes complexes par des processus markoviens homogènes [Laprie 83-a], notre approche se distingue en particulier par l'étape consacrée à l'application de la transformation.

IV.2 APPROCHE GéNéRALE

Les deux exemples que nous avons considérés dans les paragraphes précédents sont relativement simples ; l'étape consacrée à la construction de la chaîne de Markov du système en fiabilité stabilisée et l'étape portant sur l'application de la transformation peuvent être effectuées sans grande difficulté. Dans le cas d'un système complexe, le graphe d'états du système peut atteindre quelques milliers d'états. Il est indispensable donc de disposer d'un outil permettant d'une part, de formaliser la construction du graphe d'états du système et de décharger partiellement les concepteurs des modèles de la tâche pénible qui consiste à énumérer tous les états du système, et d'autre part, d'effectuer la transformation de façon mécanique et donc la rendre transparente à l'utilisateur.

Actuellement, il existe deux types d'approches pour formaliser la construction des chaînes de Markov [Laprie 83-a] :

- l'approche algébrique, par utilisation des sommes et produits de Kronecker [Amoia 81], ou de leur extensions [Mazars 82],
- l'approche graphique, par utilisation des réseaux de Petri stochastiques [Natkin 80, Molloy 82, Florin 85] et de leur extension [Dugan 84, Marsan 84, Marsan 86, Holliday 87, Roux 87].

L'approche que nous proposons est basée sur les réseaux de Petri stochastiques, et plus précisément sur les réseaux de Petri stochastiques généralisés (RdPSG) [Marsan 84]. Les principales raisons qui nous ont fait retenir comme modèle de base les réseaux de Petri sont les suivantes [Beounes 85] :

- leur puissance de modélisation et en particulier leur aptitude à prendre en compte les différents types de dépendances qui caractérisent le comportement d'un système informatique : la concurrence, le parallélisme et la synchronisation,
- leur puissance d'analyse des propriétés du système et de validation du modèle établi,
- leur aptitude à évaluer des mesures du comportement du système et en particulier des mesures de sûreté de fonctionnement,

- la disponibilité d'un ensemble important de résultats théoriques et pratiques concernant l'application des RdPSG pour l'évaluation des systèmes [Balbo 87, Marsan 87, Blakemore 89, Marsan 91].

Les RdPSG sont présentés en détail dans la littérature [Marsan 84, Marsan 87, Marsan 91] ; nous nous limiterons à présenter ici les principales propriétés de ces modèles et à introduire quelques définitions qui seront utiles dans la suite.

IV.2.1 Quelques définitions relatives aux Réseaux de Petri et aux RdPSG

Un réseau de Petri est généralement représenté sous la forme d'un graphe orienté biparti. Les nœuds du graphe sont les places (matérialisées par des cercles) et les transitions (matérialisées par des barres). Les places et les transitions sont reliées entre elles par des arcs.

Les places du réseau peuvent contenir des marques qui sont matérialisées sur le graphe par des jetons.

Le marquage du réseau de Petri est alors défini par le nombre de jetons contenus dans les différentes places du réseau.

L'ensemble des marquages possibles du réseau constitue le graphe des marquages.

Lorsque l'on modélise le comportement d'un système informatique par un réseau de Petri, les places correspondent aux états des différents éléments modélisés du système et les transitions représentent l'occurrence des différents événements qui conduisent à l'évolution du système d'un état à un autre tels que, par exemple les défaillances et les restaurations des composants du système.

Le graphe des marquages du réseau de Petri identifie l'ensemble des états du système modélisé : calculer le graphe des marquages équivaut donc à simuler le fonctionnement du système.

La définition des réseaux de Petri classiques n'inclut aucune information temporelle concernant la durée des différentes activités effectuées par le système modélisé. L'introduction du paramètre "temps" dans les réseaux de Petri a conduit à la définition de plusieurs extensions aux réseaux de Petri classiques, parmi lesquelles on distingue les RdPSG.

Pour ces réseaux, on distingue deux types de transitions : les transitions **immédiates** et les transitions **temporisées**.

Les transitions immédiates sont prioritaires par rapport aux transitions temporisées et sont franchissables immédiatement à partir du moment où elles sont sensibilisées. Elles permettent de décrire certains types de dépendance entre le comportement des composants d'un système. Les transitions temporisées sont caractérisées par des durées de franchissement qui sont exponentiellement distribuées. Elles permettent de modéliser la nature stochastique des événements qui agissent sur le comportement du système.

Une propriété importante des RdPSG est la suivante : "Le graphe des marquages associé est isomorphe à une chaîne de Markov en temps continu et à espace d'états discret".

Le traitement de la chaîne de Markov générée à partir du graphe des marquages permet alors d'effectuer une analyse quantitative du comportement dynamique du système.

Il existe actuellement des logiciels pour la construction de modèles basés sur les RdPSG, parmi lesquels le logiciel SURF2 développé au LAAS [Arlat 89].

IV.2.2 Utilisation des RdPSG dans l'approche de transformation

L'approche de transformation pour l'évaluation de la croissance de sûreté de fonctionnement d'un système multi-composant est basée, pour les raisons que nous avons indiquées, sur l'utilisation des RdPSG. Cet outil de modélisation est utilisé pour :

- modéliser le comportement du système en fiabilité stabilisée et générer la chaîne de Markov en temps continu associée à partir du graphe des marquages du RdPSG,
- transformer la chaîne de Markov en fiabilité stabilisée pour prendre en compte la croissance de sûreté de fonctionnement des différents composants du système.

La transformation d'une chaîne de Markov en fiabilité stabilisée en une chaîne de Markov en croissance de fiabilité est basée sur la représentation du modèle hyperexponentiel caractérisant la croissance de fiabilité d'un composant du système sous forme d'un RdPSG.

Un modèle RdPSG équivalent au modèle hyperexponentiel et illustrant la transformation est présenté sur la figure IV.5. Ce modèle est plus général que celui que nous avons proposé dans [Laprie 91-a] ; il est mieux adapté pour effectuer la transformation de façon mécanique dans le cas où on considère un système multi-composant.

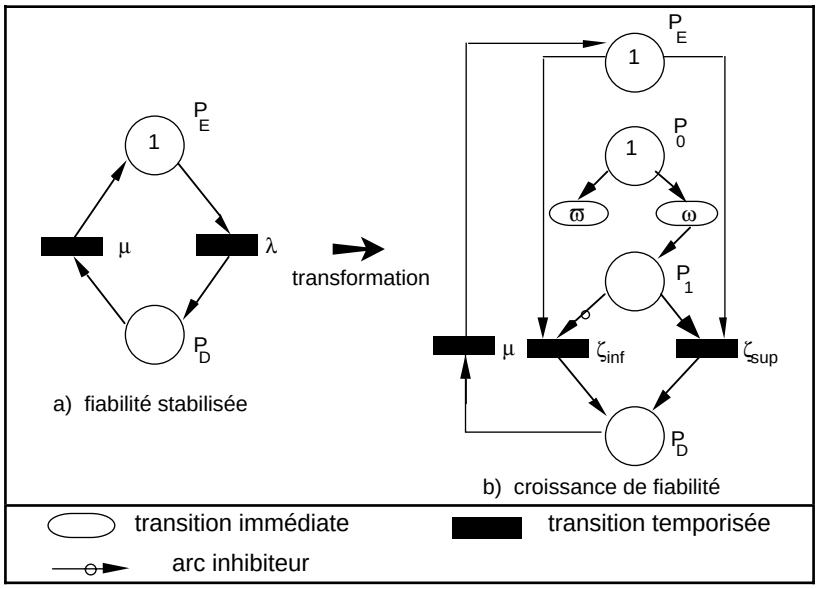
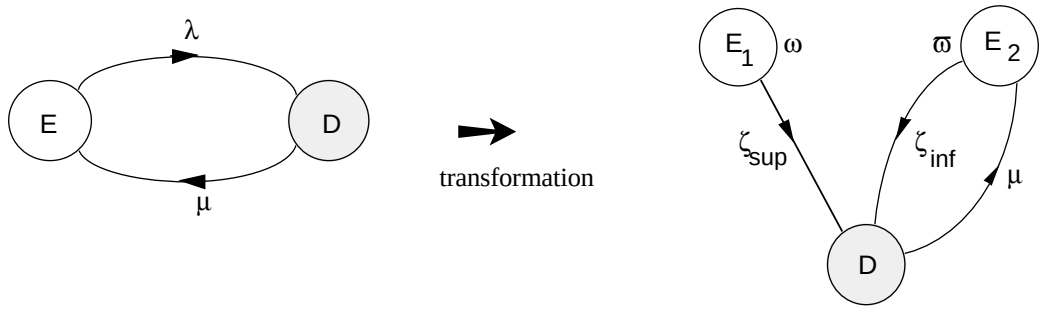


Figure IV.5 : Modèles RdPSG relatifs à un composant[1]

1 Un arc inhibiteur est tel que la transition associée n'est franchissable que si la place en entrée de la transition correspondante n'est pas marquée.

Les noms associés aux transitions correspondent dans le cas des transitions temporisées aux taux d'occurrence des événements correspondants, et dans le cas des transitions immédiates aux probabilités de franchissement des transitions correspondantes.

Les chaînes de Markov générées à partir des graphes des marquages associés aux modèles RdPSG de la figure IV.5 (a) et (b) sont respectivement celles qui sont présentées sur les figures IV.6-a et IV.6-b.



a) fiabilité stabilisée

b) croissance de fiabilité

Figure IV.6 : Chaînes de Markov associées à un composant

La transformation d'une chaîne de Markov représentant le comportement en fiabilité stabilisée d'un système en une chaîne de Markov décrivant son comportement en croissance de fiabilité se ramène alors à la transformation du RdPSG associé selon le modèle présenté sur la figure IV.5. La transformation se traduit de façon formelle par la substitution de la transition temporisée de taux associé λ par le sous réseau de Petri constitué des places P_0 et P_1, des transitions temporisées ζ_{sup} et ζ_{inf} et des transitions immédiates ω et ϖ.

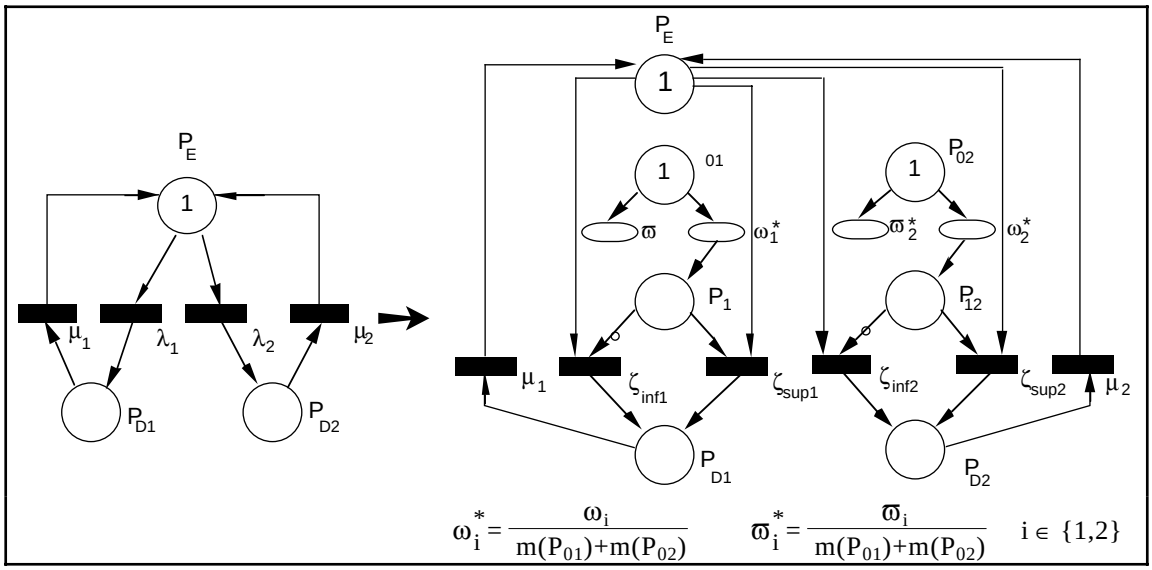
Les transitions immédiates utilisées dans le modèle RdPSG de la figure IV.5 (b) permettent d'associer les probabilités initiales ω et ϖ aux états E_1 et E_2 de la chaîne de Markov de la figure IV.6-b. On a introduit ainsi une extension supplémentaire aux RdPSG classiques qui ne considèrent que des probabilités initiales d'occupation d'états qui sont soit égales à 0 soit égales à 1.

Dans le cas d'un système multi-composant, l'approche de transformation peut se résumer par les trois étapes suivantes :

- construction du RdPSG modélisant le comportement du système en fiabilité stabilisée,
- transformation du RdPSG conformément au modèle de la figure IV.5 pour modéliser la croissance de fiabilité du système,
- génération du graphe des marquages du modèle RdPSG transformé pour obtenir la chaîne de Markov du système en croissance de fiabilité.

Le traitement de la chaîne de Markov transformée par des méthodes analytiques ou numériques permet ensuite d'évaluer les mesures de sûreté de fonctionnement du système en prenant en compte le phénomène de croissance de fiabilité.

Si on applique cette approche à l'exemple 2 (système série) présenté dans le paragraphe IV.1.2, on obtient les RdPSG du système en fiabilité stabilisée et en croissance de fiabilité présentés respectivement sur les figures IV.7-a et IV.7-b. Les chaînes de Markov générées à partir de ces RdPSG sont celles qui ont été présentées sur la figure IV.4.



a) fiabilité stabilisée b) croissance de fiabilité

Figure IV.7 : Modèles RdPSG d'un système série

IV.2.3 Conclusion

Nous avons présenté dans ce paragraphe une approche générale basée sur les RdPSG permettant de modéliser la croissance de sûreté de fonctionnement d'un système à partir de celle de ses différents composants. L'utilisation des RdPSG permet de tenir compte des dépendances stochastiques qui existent entre les comportements des différents composants d'un système et d'effectuer la transformation de façon mécanique.

L'approche de modélisation proposée dans ce chapitre est similaire à l'approche classique de modélisation de systèmes complexes par des processus markoviens homogènes ; seule l'étape concernant l'application de la transformation pour générer le modèle en croissance de fiabilité du système considéré a été introduite. Cette approche permet par conséquent de réutiliser tous les résultats existants concernant la construction et le traitement de modèles markoviens.

Néanmoins, on peut remarquer que l'application de la transformation conduit à l'augmentation de l'espace des états du système en comparaison avec les chaînes de Markov du même système en fiabilité stabilisée. En effet, si on considère par exemple, un système constitué de n composants stochastiquement indépendants qui sont tels que chaque composant est soit actif, soit défaillant, alors la cardinalité de l'espace des états de la chaîne de Markov caractérisant le comportement en fiabilité stabilisée du système varie entre n+1 (cas où tous les composants sont identiques) et 2^n (cas où tous les composants sont différents). Après l'application de la transformation, la cardinalité de l'espace des états du système varie alors entre (n+1)(n+2)/2 et 3^n. Cette explosion de l'espace des états n'est pas très contraignante puisqu'il existe actuellement des techniques et des outils puissants permettant de traiter des chaînes de Markov à plusieurs milliers d'états [Gross 84, Stewart 85, Goyal 86, Reibman 89].

IV.3 EXEMPLES D'APPLICATION DE L'APPROCHE GéNéRALE

Nous illustrerons l'approche générale sur deux systèmes. Le premier est un système duplex intégrant des mécanismes internes de détection et de recouvrement d'erreurs ; nous appliquerons l'approche proposée pour évaluer la croissance de disponibilité du système. Le deuxième système est un logiciel tolérant aux fautes ; nous appliquerons l'approche pour analyser la variation de la croissance de fiabilité du système en fonction de celle de ses différents composants.

IV.3.1 Système duplex avec couverture imparfaite

Considérons un système formé de deux composants redondants (système duplex) intégrant des mécanismes internes de détection et de recouvrement d'erreurs permettant de restaurer l'état du système dans le cas où un composant défaille. Les mécanismes de recouvrement d'erreurs sont supposés imparfaits ; leur défaillance conduit alors à la défaillance du système global.

IV.3.1.1 Construction des RdPSG en fiabilité stabilisée et en croissance de fiabilité

On note par :

- λ_c le taux de défaillance d'un composant du système correspondant aux erreurs couvertes,
- $\lambda_{\bar{c}}$ le taux de défaillance du système correspondant aux erreurs non couvertes ; $\lambda_{\bar{c}}$ peut être interprété comme le taux de défaillance des mécanismes de recouvrement d'erreurs,
- μ_1, μ_2, et μ_3, respectivement, les taux de restauration du système après occurrence :
    i) d'une première défaillance d'un composant qui est couverte par le système,
    ii) d'une défaillance non couverte,
    iii) de deux défaillances couvertes successives.

Le modèle RdPSG du système en fiabilité stabilisée est donné sur la figure IV.8.

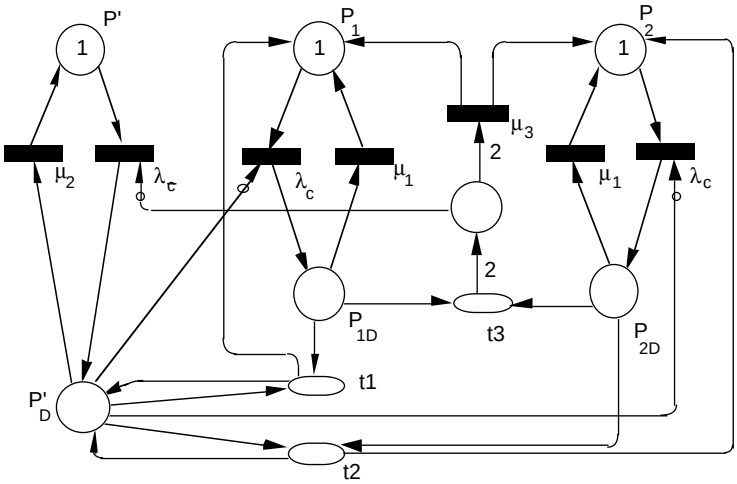


Figure IV.8 : Modèle RdPSG du système en fiabilité stabilisée.

Les comportements des deux composants du système sont modélisés respectivement par les sous-réseaux de Petri constitués des places P_1 et P_{12} (respectivement P_2 et P_{22}) et des transitions temporisées de taux associés λ_c et μ_1. Les mécanismes de détection et de

recouvrement d'erreurs sont modélisés par le sous-réseau de Petri constitué des places P'_1 et P'_2 et des transitions temporisées de taux associés $\lambda_{\bar{c}}$ et μ_2.

Les dépendances stochastiques entre les comportements des composants sont représentées d'une part, par les arcs inhibiteurs et d'autre part, par les transitions immédiates t_{11} , t_{12} et t_2. Les transitions t_{11} et t_{12} expriment que le système est défaillant si le deuxième composant du système défaille avant la restauration du premier composant qui a défailli. La transition t_2 exprime que le système est défaillant dès qu'il y a une défaillance non couverte de l'un des deux composants.

Etant donné que les deux composants du système sont identiques, on peut simplifier le RdPSG de la figure IV.8 en représentant le comportement des deux composants du système par un même sous-réseau de Petri. Le RdPSG réduit ainsi obtenu est présenté sur la figure IV.9. Les processus de défaillance et de restauration correspondant aux défaillances des composants qui sont couvertes par le système sont alors modélisés par le même sous-réseau de Petri constitué par les places P_1 et P_2 et les transitions temporisées de taux associés respectifs $m(P_1)\lambda_c$ et μ_1 où $m(P_1)$ est le marquage de la place P_1.

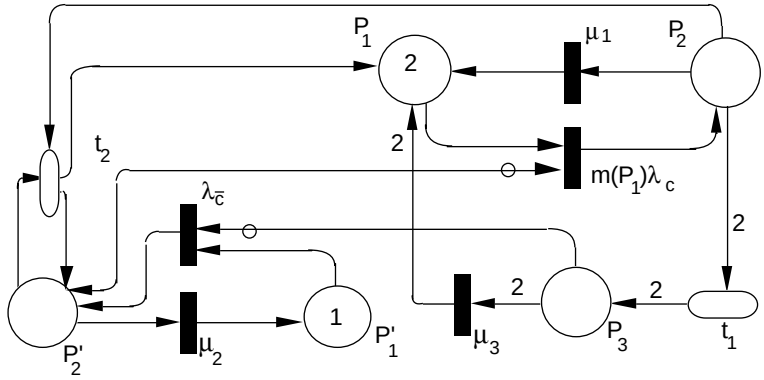
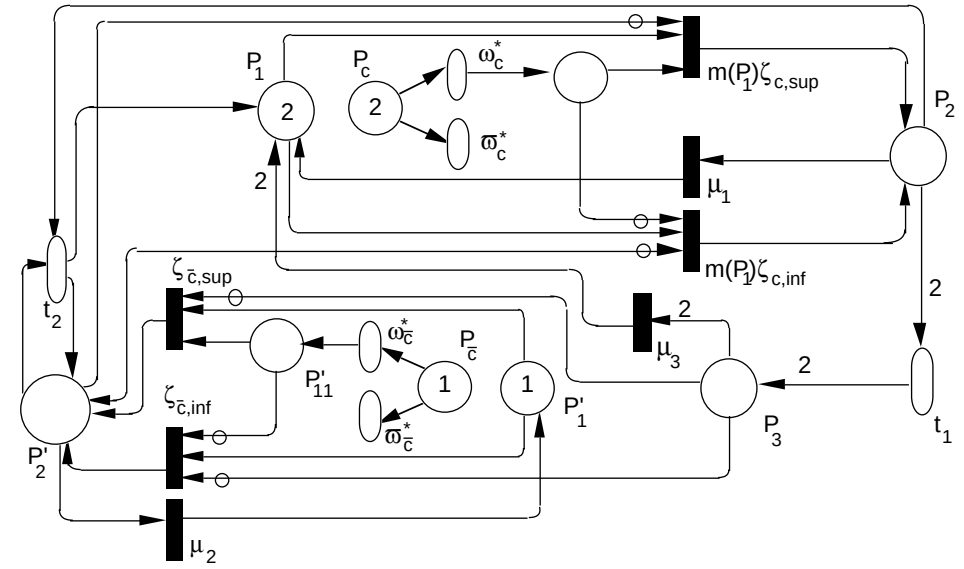


Figure IV.9 : Modèle RdPSG réduit du système en fiabilité stabilisée.

L'application de l'approche de transformation selon le modèle de la figure IV.5, conduit au modèle RdPSG transformé présenté sur la figure IV.9 prenant en compte le phénomène de croissance de fiabilité relatif aux différents composants du système (figure IV.9) où $\{\omega_c, \varpi_c, \zeta_{c,sup}, \zeta_{c,inf}\}$ résultent de la transformation de λ_c et $\{\omega_{\bar{c}}, \varpi_{\bar{c}}, \zeta_{\bar{c},sup}, \zeta_{\bar{c},inf}\}$ résultent de la transformation de $\lambda_{\bar{c}}$.

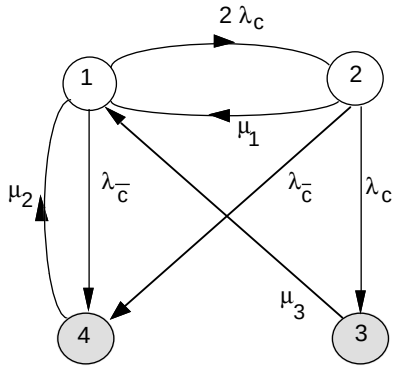


$$\omega^*_x = \frac{\omega_x}{m(P_c)+m(P_{\bar{c}})} \qquad \varpi^*_x = \frac{\varpi_x}{m(P_c)+m(P_{\bar{c}})} \qquad x \in \{c,\bar{c}\}$$

Figure IV.9 : Modèle RdPSG du système en croissance de fiabilité.

IV.3.1.2 Génération des chaînes de Markov en fiabilité stabilisée et en croissance de fiabilité

La chaîne de Markov du système en fiabilité stabilisée calculée à partir du graphe des marquages du RdPSG correspondant est présentée sur la figure IV.10. Nous donnons aussi sur cette figure les marquages, relatifs au RdPSG de la figure IV.9, qui sont associés aux différents états de la chaîne de Markov afin de faire le lien entre le modèle RdPSG et la chaîne de Markov correspondante.



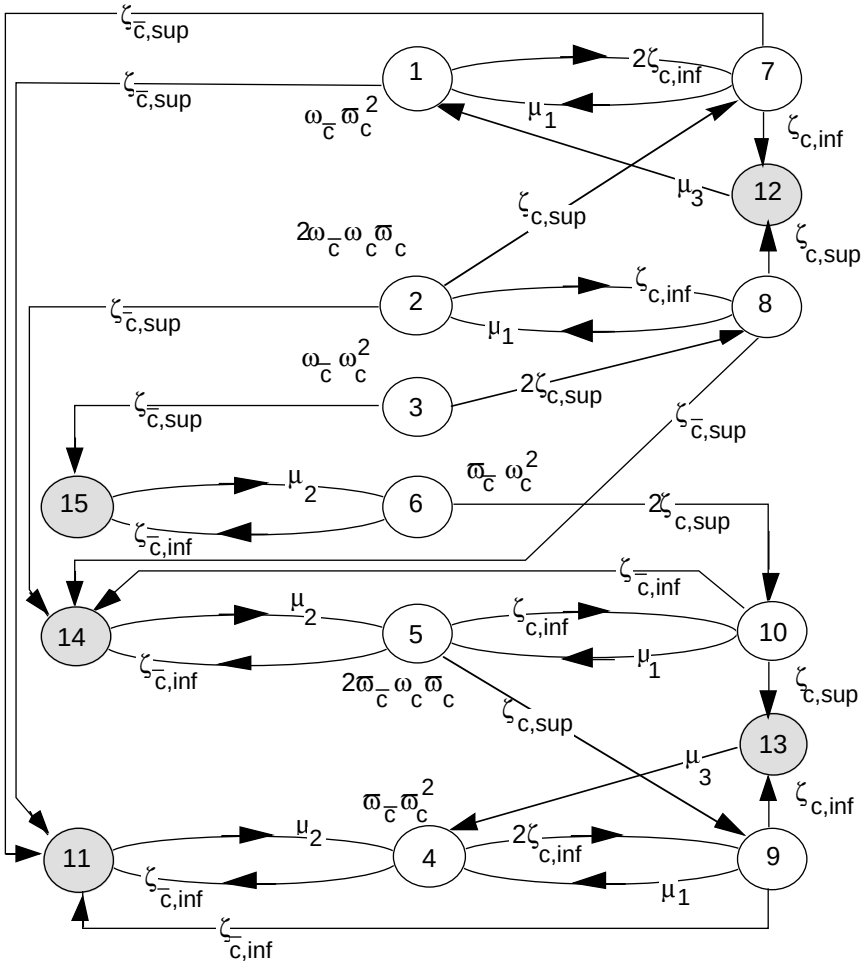
Etat	Marquage
1	{2P1,P'1}
2	{P1,P2,P'1}
3	{2P3,P'1}
4	{2P1,P'2}

Etat 1	état de bon fonctionnement du système (deux composants actifs)
Etat 2	un composant actif et un composant défaillant suite à occurrence d'une défaillance couverte
Etat 3	système défaillant suite à occurrence d'une défaillance couverte du 2ième composant avant restauration du premier composant
Etat 4	système défaillant suite à occurrence d'une défaillance non couverte

Figure IV.10 : Chaîne de Markov du système en fiabilité stabilisée.

La chaîne de Markov du système en croissance de fiabilité générée à partir du graphe des marquages du RdPSG transformé est donnée sur la figure IV.11. En comparant les chaînes de

Markov du système en fiabilité stabilisée et en croissance de fiabilité, on peut remarquer que l'application de la transformation conduit à l'augmentation de l'espace des états du système.



Etat	Marquage
1	{2P1,P'1,P'11}
2	{2P1,P11,P'1,P'11}
3	{2P1,2P11,P'1,P'11}
4	{2P1,P'1}
5	{2P1,P11,P'1}
6	{2P1,2P11,P'1}
7	{P1,P2,P'1,P'11}
8	{P1,P11,P2,P'1,P'11}
9	{P1,P2,P'1}
10	{P1,P11,P2,P'1}
11	{2P1,P'2}
12	{2P3,P'1,P'11}
13	{2P3,P'1}
14	{2P1,P11,P'2}
15	{2P1,2P11,P'2}

Figure IV.11 : Chaîne de Markov du système en croissance de fiabilité.

IV.3.1.3 Traitement de la chaîne de Markov transformée

On se propose d'étudier l'évolution de la disponibilité du système en tenant compte du phénomène de croissance. Pour évaluer la disponibilité du système, on peut traiter la chaîne de Markov transformée par le logiciel SURF développé au LAAS.

Considérons les valeurs numériques suivantes des paramètres du modèle :

Valeurs des paramètres	Commentaires
$\omega_c = 0,5$ $\omega_{\bar{c}} = 0,5$	Croissance de fiabilité progressive.
$\zeta_{c,inf} = 10^{-4}$/h $\zeta_{\bar{c},inf} = 5\ 10^{-7}$/h	$\zeta_{\bar{c},inf}/\zeta_{c,inf} = 5\ 10^{-3}$: le taux d'occurrence de défaillance non couvertes est beaucoup plus faible que celui des défaillances couvertes.
$10^{-4} \le \lambda_c \le 10^{-3}$ $510^{-7} \le \lambda_{\bar{c}} \le 5\ 10^{-6}$	Facteur de croissance de 10 pour les deux sources de défaillance.
$\zeta_{c,inf}/\mu_1 = 0,5\ 10^{-3}$ $\zeta_{c,inf}/\mu_2 = \zeta_{c,inf}/\mu_3 = 10^{-3}$	On suppose que la restauration du système est plus rapide quand le système continue à fonctionner et qu'un seul composant est défaillant, comparé au cas où le système est défaillant.

Le traitement de la chaîne de Markov transformée avec les valeurs numériques considérées donne les résultats présentés sur la figure IV.12 :

- la courbe C1 (respectivement C5) correspond à un comportement en fiabilité stabilisée du système, où λ_c et $\lambda_{\bar{c}}$ sont égales à leur valeur minimale (respectivement maximale),
- C4 correspond à un comportement en croissance de fiabilité du système ; où λ_c et $\lambda_{\bar{c}}$ tendent à décroître progressivement dans le temps de leur valeur maximale vers leur valeur minimale,
- la courbe C2 (respectivement C3) représente l'évolution de la décroissance de l'indisponibilité du système quand on considère uniquement une croissance de fiabilité du système liée aux défaillances couvertes (respectivement non couvertes). En réalité ces deux phénomènes sont corrélés ; l'intérêt de représenter C2 et C3 est essentiellement d'analyser la sensibilité de l'évolution de la croissance de disponibilité du système en fonction des différents processus qui agissent sur son comportement.

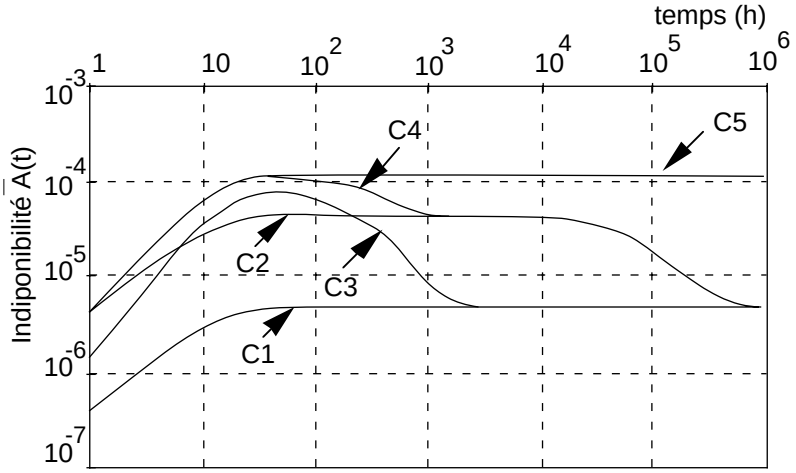


Figure IV.12 : Courbes d'indisponibilité associées au modèle de la figure IV.11.

L'analyse des différentes courbes présentées sur la figure IV.11 montre qu'il existe une différence significative entre l'allure d'évolution des courbes d'indisponibilité basées sur l'hypothèse d'un comportement en fiabilité stabilisée du système (C1 et C5) et les courbes prenant en compte le phénomène de croissance de fiabilité (C2 , C3 et C4). Par conséquent, ne pas tenir compte de la croissance de fiabilité des composants du système peut conduire à des résultats qui ne sont pas représentatifs du comportement du système modélisé. L'erreur commise dans l'estimation dépend des valeurs considérées pour représenter le comportement en fiabilité stabilisée du système. Les courbes C1 et C5 correspondent aux valeurs minimales et maximales des valeurs des taux de défaillance considérés ; dans le cas où l'indisponibilité en fiabilité stabilisée est évaluée avec d'autres valeurs intermédiaires en supposant toujours que les différents taux de défaillance sont constants, les courbes correspondantes auront la même allure et seront situées entre C1 et C5 et l'erreur commise dans l'estimation dépendra des valeurs choisies par rapport aux valeurs maximale et minimale.

IV.3.2 Logiciel tolérant aux fautes

Le système que nous venons de considérer est typiquement **matériel** et la mesure considérée est la **disponibilité**. L'exemple que nous présentons dans ce paragraphe, porte sur l'évaluation d'un **logiciel** tolérant aux fautes ; nous nous intéresserons plus particulièrement à l'évaluation de la croissance de **fiabilité**.

Il existe trois approches principales pour la tolérance aux fautes logicielles qui sont les blocs de recouvrement [Randell 75], la programmation en N-versions [Chen 78] et la programmation N-autotestable [Laprie 90-b].

Dans ce paragraphe, nous appliquons l'approche de transformation à un logiciel constitué d'un bloc de recouvrement. L'application de l'approche de transformation à des logiciels structurés selon les deux autres approches est effectuée dans [Kanoun 91-c].

Considérons donc un bloc de recouvrement (BR) formé de deux alternants, un primaire (P) et un secondaire (S), et d'un test d'acceptation (T). Le test d'acceptation est destiné à juger de l'acceptabilité des résultats fournis par les alternants.

La structure de base d'un BR peut être résumée de la façon suivante :

satisfaire T

par <P>

sinon par <S >

sinon erreur

Notre objectif est d'étudier l'influence sur l'évolution de la fiabilité du logiciel global, de la croissance de fiabilité de ses deux alternants et du test d'acceptation résultant de l'élimination progressive des fautes introduites au niveau de ces composants .

IV.3.2.1 Hypothèses de fautes et de modélisation

Nous distinguerons les deux types de fautes suivants :

- les **fautes indépendantes**, qui sont introduites au cours des spécifications et des mises en œuvre indépendantes des alternants et du test d'acceptation ; l'activation d'une faute indépendante conduit à la défaillance du composant concerné,
- les **fautes corrélées**, qui peuvent résulter de l'introduction de fautes au cours de la spécification commune des différents composants du BR ; elle peuvent être introduites aussi pendant la conception et la mise en œuvre de ces alternants. L'activation d'une faute corrélée conduit à la défaillance combinée des composants concernés.

Le détail des différentes fautes ainsi que les conséquences de leur activation, indépendante ou combinée, sur le comportement du BR est donné dans ce qui suit.

IV.3.2.1.1 Fautes indépendantes et hypothèses de modélisation

- Une faute indépendante dans P peut être tolérée en l'absence de faute indépendante dans S qui est activée par les mêmes données d'entrée ou de faute corrélée dans P et S.
- Une faute indépendante ne peut être activée dans S que si T déclare erroné le résultat délivré par P et fait appel à S.
- Une faute indépendante est activée dans T quand celui-ci déclare défaillant un alternant qui ne l'est pas.

IV.3.2.1.2 Fautes corrélées et hypothèses de modélisation

- Une faute corrélée dans P et S conduit, lors de son activation, à la défaillance du BR. Une telle défaillance est détectée par le test d'acceptation (en supposant que T n'est pas défaillant).
- Une faute corrélée dans P et T ou dans S et T (après activation d'une faute dans P) entraîne la défaillance du BR. Dans ce cas, le résultat délivré par l'alternant défaillant est déclaré non erroné par le test d'acceptation et par suite, la défaillance de l'alternant correspondant n'est pas détectée.

Dans la suite, nous nous intéresserons à l'évaluation de la fiabilité du BR ; par conséquent, les états de défaillances détectées et de défaillances non détectées du BR seront regroupées en un seul état : défaillance du BR.

IV.3.2.1.3 Autres hypothèses de modélisation

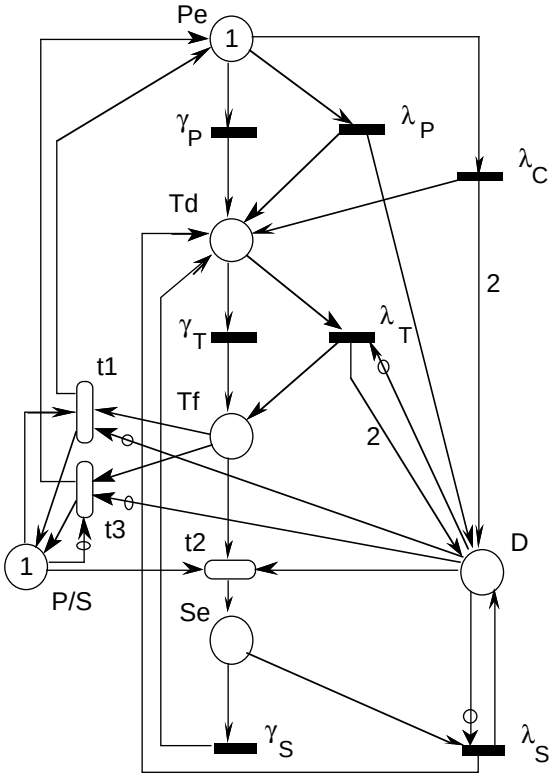
En plus des hypothèses que nous venons d'établir, nous supposerons que :

- l'activation d'une faute corrélée, quelle que soit sa source, conduit à la défaillance du BR,
- l'activation d'une faute dans T, indépendante ou corrélée, entraîne la défaillance du BR. Nous négligerons ainsi le phénomène de compensation d'erreurs ; c'est-à-dire, si T rejette un résultat correct délivré par P alors il déclarera erroné également le résultat délivré par S,
- un seul type de faute, indépendante ou corrélée, peut être activé au cours d'une exécution du BR. Dans ce cas, on peut regrouper toutes les fautes corrélées quelles que soit leur origine en un seul type de fautes corrélées, noté C, qui correspond à l'activation d'une faute corrélée dans deux alternants ou bien dans un alternant et le test d'acceptation [Arlat 88-b].

IV.3.2.2 Modèles RdPSG du BR en fiabilité stabilisée et en croissance de fiabilité

En se basant sur les hypothèses de fautes et de modélisation que nous venons d'établir, on peut modéliser le comportement du BR par le réseau de Petri stochastique généralisé présenté sur la figure IV.13. où γ_X et λ_X sont respectivement, le taux d'exécution de X et le taux d'activation de fautes indépendantes dans ce composant, $X \in \{P, S, T\}$, et λ_C est le taux d'activation de fautes corrélées dans le BR. Nous donnons également sur cette figure,

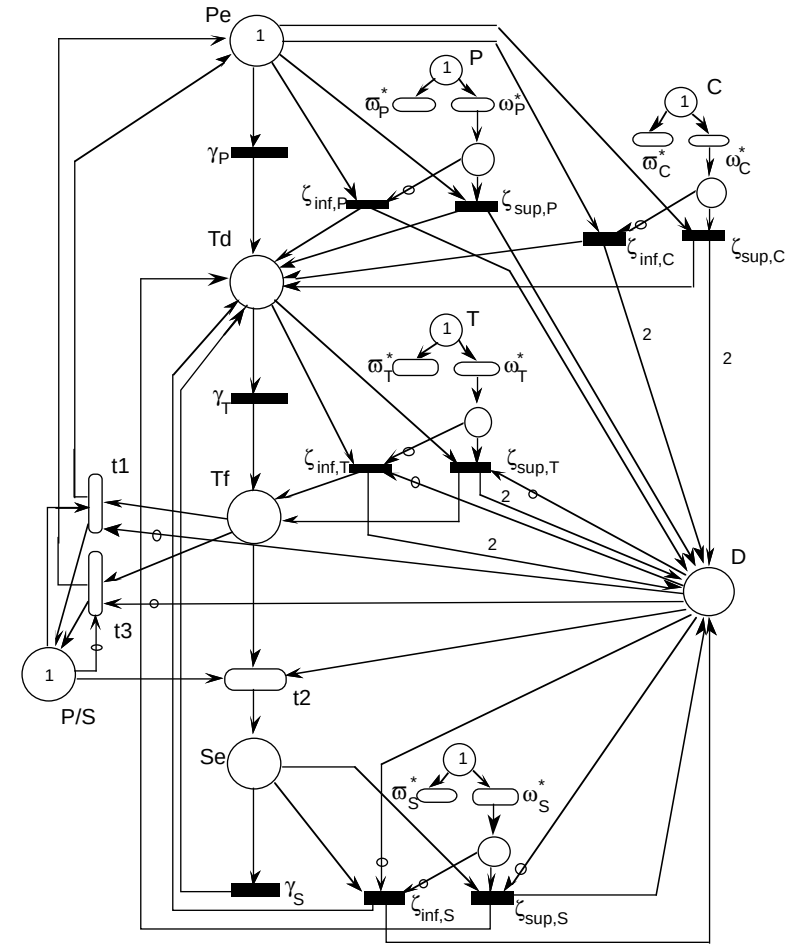
l'interprétation des places et des transitions du RdPSG afin de comprendre l'évolution du comportement du BR au cours de son exécution en fonction des événements qui gouvernent les transitions entre les états du système.



Pe	P en cours d'exécution	γ_P	Exécution correcte de P
		λ_P	Activation d'une faute indépendante dans P
		λ_C	Activation d'une faute corrélée
Td	T en cours d'exécution	γ_T	Exécution correcte de T
		λ_T	Activation d'une faute indépendante dans T
Se	S en cours d'exécution	γ_S	Exécution correcte de S
		λ_S	Activation d'une faute indépendante dans S
Tf	Choix de l'exécution suivante	t1, t3	Sélection de P
		t2	Sélection de S
P/S	Sélection de P ou S		
D	Type de faute activée: m(D) = 1 : faute indépendante dans P ou S m(D) = 2 : faute indépendante dans T ou faute corrélée		

Figure IV.13 : RdPSG du BR en fiabilité stabilisée.

L'application de la transformation au RdPSG de la figure IV.13 conduit au RdPSG transformé présenté sur la figure de la figure IV.14 décrivant la croissance de fiabilité du BR en tenant compte de la croissance de fiabilité de P, S et T résultant de l'élimination progressive des fautes indépendantes et des fautes corrélées : ω_X, $\zeta_{sup,X}$ et $\zeta_{inf,X}$ sont les paramètres du modèle hyperexponentiel qui décrivent la croissance de fiabilité de X par rapport à l'activation de fautes indépendantes, $X \in \{P, S, T\}$, et ω_C, $\zeta_{sup,C}$ et $\zeta_{inf,C}$ sont les paramètres du modèle hyperexponentiel qui décrivent la croissance de fiabilité du BR par rapport à l'activation de fautes corrélées.



$$\omega_X^* = \frac{\omega_X}{\sum m(X)} \qquad \varpi_X^* = \frac{\varpi_X}{\sum m(X)} \qquad X \in \{P,T,S,C\}$$

Figure IV.14 : RdPSG du BR en croissance de fiabilité.

IV.3.2.3 Chaînes de Markov du BR en fiabilité stabilisée et en croissance de fiabilité

Les chaînes de Markov qui sont générées à partir des graphes des marquages des RdPSG du BR en fiabilité stabilisée et en croissance de fiabilité sont présentées respectivement sur les figures IV.15 et IV.16.

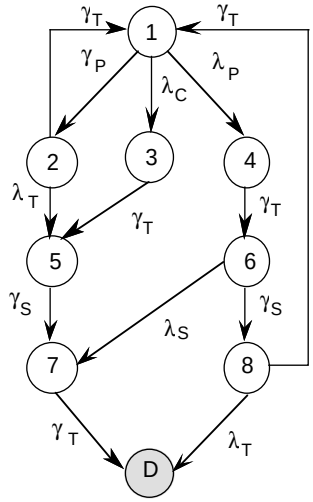


Figure IV.15 : Chaîne de Markov du BR en fiabilité stabilisée.

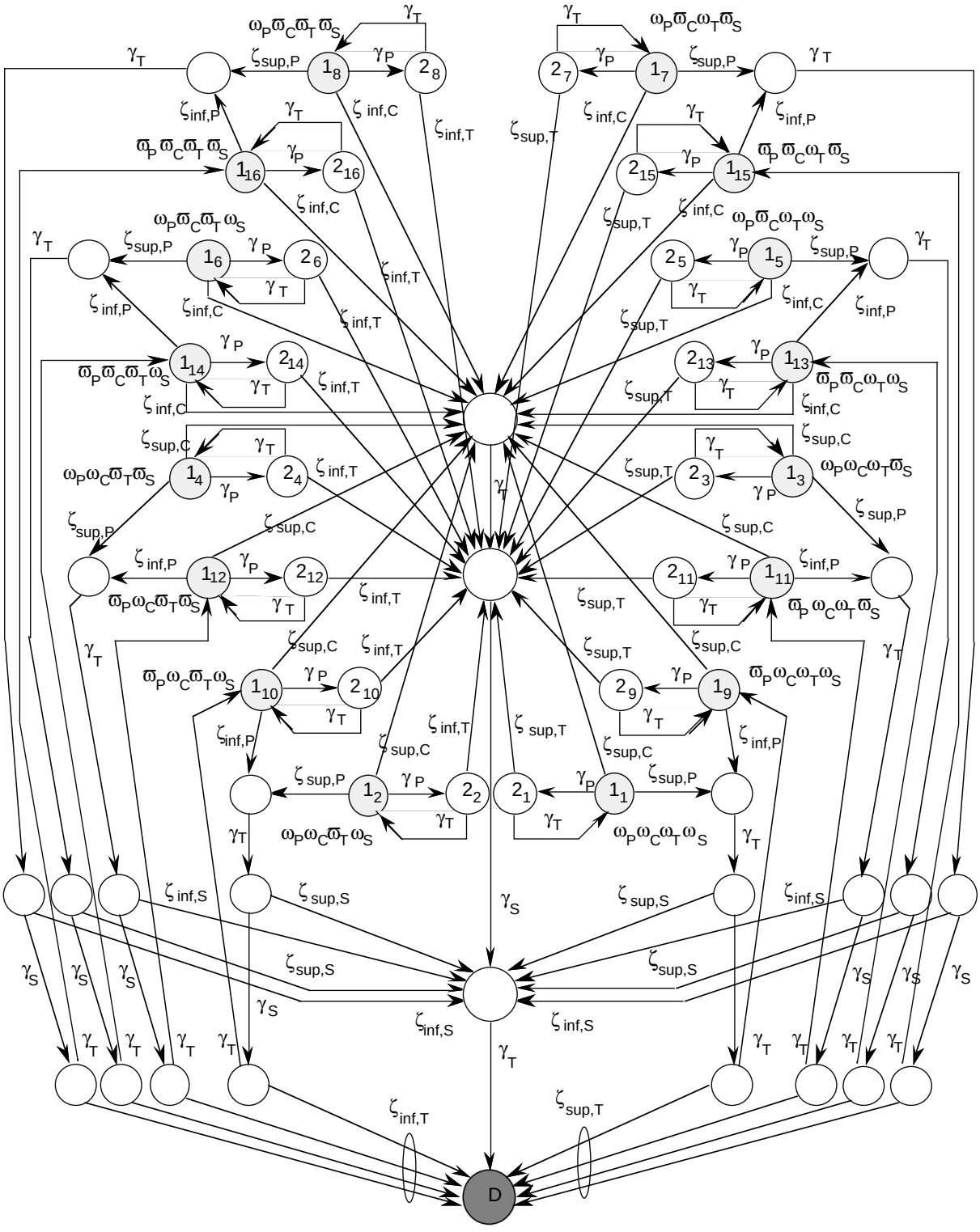


Figure IV.16 : Chaîne de Markov du BR en croissance de fiabilité.

La chaîne de Markov de la figure IV.16 caractérise le comportement du BR en considérant une croissance de fiabilité par rapport à tous les processus considérés. A partir de cette chaîne de Markov, on peut aussi obtenir les chaînes de Markov du BR en considérant une croissance de fiabilité du logiciel par rapport à un ou plusieurs processus à la fois. Il suffit de poser : $\omega_X = 0$ et $\zeta_{inf,X} = \lambda_X$, X = {P, S, T, C} pour tous les processus pour lesquels on suppose une fiabilité stabilisée. Par exemple, si on veut étudier le comportement du BR en tenant compte uniquement

de la croissance de fiabilité résultant de l'élimination progressive des fautes indépendantes dans P, il suffit de poser $\omega_X = 0$ et $\zeta_{inf,X} = \lambda_X$, X = {S, T, C}.

En particulier, la chaîne de Markov du BR en fiabilité stabilisée peut s'obtenir à partir de la chaîne transformée en posant : $\omega_X = 0$ et $\zeta_{inf,X} = \lambda_X$, X = {P, S, T, C}.

L'intérêt de considérer les chaînes de Markov caractérisant le comportement du BR en tenant compte de la croissance de fiabilité d'un ou plusieurs composants du système est d'effectuer une étude de sensibilité sur le comportement du système afin d'analyser l'évolution de la sûreté de fonctionnement du BR en fonction de la croissance de fiabilité de celle de ses composants.

IV.3.2.4 Méthode d'évaluation de la fiabilité du bloc de recouvrement

Notre objectif est d'évaluer la fiabilité d'un logiciel formé d'un BR au cours de sa vie opérationnelle en tenant compte de la croissance de fiabilité de ses différents composants. Plus particulièrement, nous allons étudier l'évolution dans le temps de la probabilité de bon fonctionnement du logiciel pour une durée de fonctionnement u. u peut être interprétée comme la durée d'une mission effectuée par le logiciel tolérant aux fautes.

IV.3.2.4.1 Présentation de la mesure

Si on note par t le temps courant et u une durée fixe, la mesure qu'on cherche à évaluer est R(t,t+u) définie par :

$$R(t,t+u) = Pr\{\text{exécution du BR sans défaillance dans l'intervalle } [t, t+u]\}$$

Cette mesure permet d'évaluer la sûreté de fonctionnement, vis-à-vis de la continuité de service, des systèmes qui ne sont utilisés que durant des périodes de temps u (par exemple les systèmes avioniques). Généralement pour ces systèmes, la durée de mission u est très faible par rapport au temps courant t.

En tenant compte du fait que u << t, on démontre dans [Laprie 91-b], que dans le cas où le processus de défaillance du logiciel est représenté par un processus de renouvellement généralisé et plus particulièrement dans le cas où le logiciel manifeste une croissance de fiabilité, on a :

$$R(t, t+u) \approx 1- h(t)u + o(u) \qquad \text{(IV.9)}$$

Considérons les fonctions $r(\tau)$ et $\lambda(\tau)$ qui correspondent respectivement à la fiabilité et au taux de défaillance du BR à un instant τ donné qui sont évalués à partir de la chaîne transformée caractérisant la croissance de fiabilité du BR.

On peut écrire à partir des définitions classiques de $r(\tau)$ et $\lambda(\tau)$:

$$\frac{r(t+u)}{r(t)} = \exp - \int_{t}^{t+u} \lambda(\tau)\, d\tau \qquad \text{(IV.10)}$$

En supposant t >> u , la relation (IV.10) devient :

$$\frac{r(t+u)}{r(t)} \approx 1 - \lambda(\tau)\, u + o(u) \qquad \text{(IV.11)}$$

Compte tenu du fait que pour un processus NHPP le taux de défaillance et l'intensité de défaillance sont indissociables, le taux de défaillance évalué à partir de la chaîne de Markov transformée peut être interprété comme l'intensité de défaillance caractérisant la croissance de fiabilité du BR. On obtient ainsi :

$$R(t,t+u) = \frac{r(t+u)}{r(t)} \qquad \text{(IV.12)}$$

Il suffit donc d'évaluer la fonction $r(\tau)$ à partir de la chaîne transformée pour évaluer $R(t,t+u)$.

IV.3.2.4.2 Traitement de la chaîne de Markov transformée : technique d'agrégation

Compte tenu du fait que les taux de restauration et les taux d'exécution du BR sont très grands par rapport aux taux de défaillance du logiciel, nous nous trouvons dans le cas d'une chaîne de Markov "raide" [Reibman 89]. Dans ces conditions, il est difficile d'obtenir par les méthodes numériques classiques, des estimations des mesures de fiabilité avec une précision acceptable.

Actuellement, il existe deux principales approches permettant d'évaluer des mesures transitoires à partir de chaînes de Markov "raides" [Bobbio 90]:

. la première approche consiste à améliorer les techniques numériques classiques de traitement de chaîne de Markov dans le cas où la chaîne considérée est raide [Reibman 89],

. la deuxième approche est basée sur une technique dite "d'agrégation" qui consiste à transformer la chaîne de Markov considérée en une autre chaîne qui est telle que tous les taux de transition sont du même ordre de grandeur [Bobbio 86].

La deuxième approche possède deux avantages : elle permet d'une part, de transformer la chaîne de Markov initiale en une chaîne de Markov non raide qui peut être traitée par les méthodes numériques classiques, et d'autre part, de réduire l'espace des états du système considéré en procédant par décomposition du modèle initial en sous-modèles pouvant être traités séparément, et en évaluant les mesures caractéristiques du système à partir des solutions obtenues à partir des différents sous-modèles. Dans le cas où le nombre d'états de la chaîne réduite obtenue n'est pas très important, on peut obtenir les expressions analytiques des mesures transitoires caractérisant le comportement du système considéré : par exemple la fiabilité.

Compte tenu des avantages de cette approche, nous présenterons brièvement dans la suite ses principales propriétés puis nous l'appliquerons au BR pour obtenir l'expression analytique de $R(t,t+u)$ et étudier l'évolution de la fiabilité du BR en fonction de la croissance de fiabilité de ses différents composants.

PRéSENTATION DE LA TECHNIQUE D'AGRéGATION

La technique d'agrégation proposée dans [Bobbio 86] peut être résumée comme suit : selon l'ordre de grandeur des taux de transition de la chaîne de Markov considérée, on distingue des taux de transition "rapides" et des taux de transition "lents". On effectue alors une partition de la chaîne de Markov en deux sous-ensembles : le sous-ensemble des états "rapides" et le sous-ensemble des états "lents" :

- Un état est dit "rapide" quand il possède au moins un taux de transition "rapide" en sortie.
- Un état est dit "lent" quand toutes les transitions en sortie de l'état considéré sont telles que les taux associés sont lents.

Parmi les états rapides, on distingue ensuite, conformément à la classification classique des états d'une chaîne de Markov [Kemeny 59], les sous-ensembles d'états récurrents et les sous-ensembles d'états transitoires.

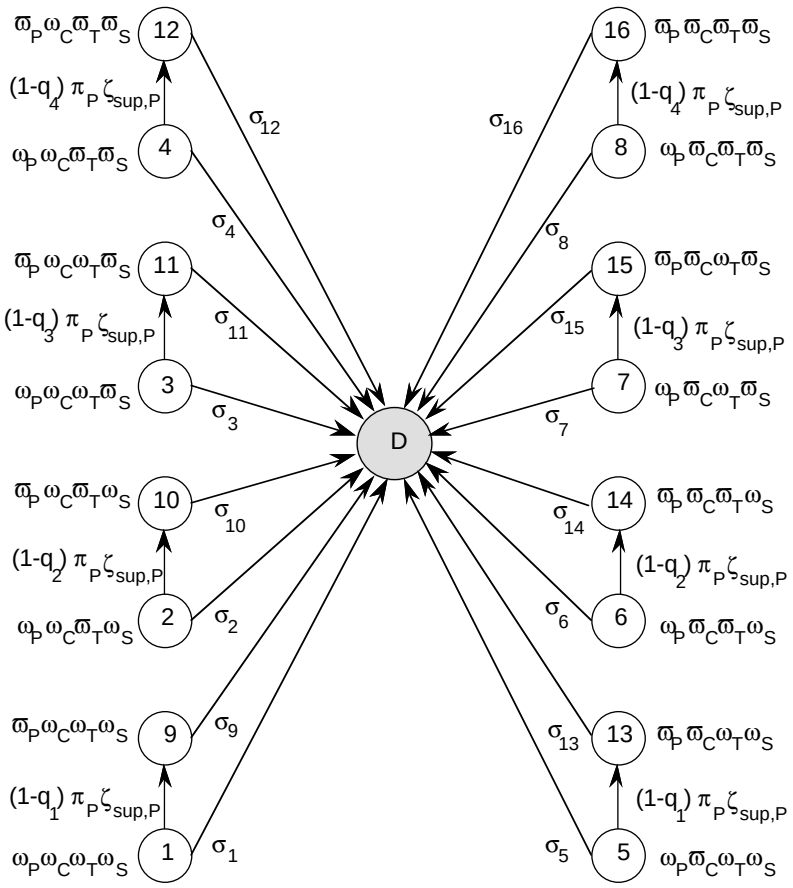
Chaque sous-ensemble d'états récurrents rapides est ensuite agrégé en un seul état lent. On élimine par la suite le sous-ensemble des états transitoires et on calcule les nouveaux taux de transition entre les états lents de la chaîne de Markov en tenant compte des taux de transitions associés aux états transitoires éliminés. On obtient finalement une nouvelle chaîne de Markov qui est telle que, d'une part, l'espace des états est plus petit que celui de la chaîne de Markov initiale, et d'autre part, les taux de transition de la chaîne obtenue sont tous du même ordre de grandeur.

IV.3.2.5 Application de la méthode d'évaluation au bloc de recouvrement

En considérant la méthode que nous venons de présenter, on peut obtenir l'expression analytique de la fiabilité R(t,t+u) caractérisant la croissance de fiabilité du BR et étudier la variation de la fiabilité du BR en tenant compte de la croissance de fiabilité de ses différents composants.

IV.3.2.5.1 Evaluation de R(t,t+u)

Considérons la chaîne de Markov transformée du BR présentée sur la figure IV.16. Conformément aux définitions présentées précédemment, les états $\{1_i, 2_i\}$, $i \in \{1\ldots16\}$, constituent des états récurrents rapides. Mis à part l'état absorbant qui est un état lent, tous les autres états de la chaîne de Markov considérée sont des états transitoires rapides. L'application de la technique d'agrégation à la chaîne de Markov transformée conduit alors à la chaîne de Markov réduite présentée sur la figure IV.17.



$\sigma_1 = (q_1\,\zeta_{sup,P} + \zeta_{Sup,C})\,\pi_P + \pi_T\,\zeta_{Sup,T}$ $\quad$ $\sigma_2 = (q_2\,\zeta_{sup,P} + \zeta_{sup,C})\,\pi_P + \pi_T\,\zeta_{inf,T}$

$\sigma_3 = (q_3\,\zeta_{sup,P} + \zeta_{sup,C})\,\pi_P + \pi_T\,\zeta_{sup,T}$ $\quad$ $\sigma_4 = (q_4\,\zeta_{sup,P} + \zeta_{sup,C})\,\pi_P + \pi_T\,\zeta_{inf,T}$

$\sigma_5 = (q_1\,\zeta_{sup,P} + \zeta_{inf,C})\,\pi_P + \pi_T\,\zeta_{Sup,T}$ $\quad$ $\sigma_6 = (q_2\,\zeta_{sup,P} + \zeta_{inf,C})\,\pi_P + \pi_T\,\zeta_{inf,T}$

$\sigma_7 = (q_3\,\zeta_{sup,P} + \zeta_{inf,C})\,\pi_P + \pi_T\,\zeta_{sup,T}$ $\quad$ $\sigma_8 = (q_4\,\zeta_{sup,P} + \zeta_{inf,C})\,\pi_P + \pi_T\,\zeta_{inf,T}$

$\sigma_9 = (q_1\,\zeta_{inf,P} + \zeta_{sup,C})\,\pi_P + \pi_T\,\zeta_{sup,T}$ $\quad$ $\sigma_{10} = (q_2\,\zeta_{inf,P} + \zeta_{sup,C})\,\pi_P + \pi_T\,\zeta_{inf,T}$

$\sigma_{11} = (q_3\,\zeta_{inf,P} + \zeta_{sup,C})\,\pi_P + \pi_T\,\zeta_{sup,T}$ $\quad$ $\sigma_{12} = (q_4\,\zeta_{inf,P} + \zeta_{sup,C})\,\pi_P + \pi_T\,\zeta_{inf,T}$

$\sigma_{13} = (q_1\,\zeta_{inf,P} + \zeta_{inf,C})\,\pi_P + \pi_T\,\zeta_{sup,T}$ $\quad$ $\sigma_{14} = (q_2\,\zeta_{inf,P} + \zeta_{inf,C})\,\pi_P + \pi_T\,\zeta_{inf,T}$

$\sigma_{15} = (q_3\,\zeta_{inf,P} + \zeta_{inf,C})\,\pi_P + \pi_T\,\zeta_{sup,T}$ $\quad$ $\sigma_{16} = (q_4\,\zeta_{inf,P} + \zeta_{inf,C})\,\pi_P + \pi_T\,\zeta_{inf,T}$

$\pi_P = \dfrac{\gamma_T}{\gamma_T+\gamma_P}$ = proportion de temps de séjour dans P en absence de défaillance par rapport au temps de séjour dans P et T.

$\pi_T = 1-\pi_P$ = proportion de temps de séjour dans T en absence de défaillance par rapport au temps de séjour dans P et T.

$$q_1 = \frac{\zeta_{Sup,S}}{\zeta_{sup,S} + \gamma_S} + \frac{\zeta_{Sup,T}\,\gamma_S}{(\zeta_{Sup,S} + \gamma_S)(\zeta_{Sup,T} + \gamma_T)}$$

$$q_2 = \frac{\zeta_{sup,S}}{\zeta_{sup,S} + \gamma_S} + \frac{\zeta_{inf,T}\,\gamma_S}{(\zeta_{sup,S} + \gamma_S)(\zeta_{inf,T} + \gamma_T)}$$

$$q_3 = \frac{\zeta_{inf,S}}{\zeta_{inf,S} + \gamma_S} + \frac{\zeta_{sup,T}\,\gamma_S}{(\zeta_{inf,S} + \gamma_S)(\zeta_{sup,T} + \gamma_T)}$$

$$q_4 = \frac{\zeta_{inf,S}}{\zeta_{inf,S} + \gamma_S} + \frac{\zeta_{inf,T}\,\gamma_S}{(\zeta_{inf,S} + \gamma_S)(\zeta_{inf,T} + \gamma_T)}$$

Figure IV.17 : Chaîne de Markov réduite du BR en croissance de fiabilité.

Les états $\{1_i, 2_i\}$, $i \in \{1\ldots16\}$ de la chaîne transformée sont remplacés par les états i, dans la chaîne de Markov réduite de la figure IV.17. On peut remarquer que l'application de la technique d'agrégation conduit à la réduction de l'espace des états du système ; le nombre des états du BR est passé de 60 états (chaîne transformée) à 17 états (chaîne réduite).

Pour obtenir l'expression de R(t,t+u), il suffit d'évaluer la fonction de fiabilité $r(\tau)$ à partir de la chaîne de Markov réduite et d'appliquer ensuite la relation IV.12. L'expression obtenue pour $r(\tau)$ est la suivante :

$$r(\tau) = \sum_{i=1}^{4} \omega_P\, \delta_i\, (1+A_i)\ \exp\left(-(1-q_i)\pi_P\zeta_{sup,P}\tau\right) \left\{\ \omega_C \exp(-\sigma_i\tau) + \varpi_C \exp(-\sigma_{i+4}\tau)\right\}$$

$$+ \sum_{i=1}^{4} (\varpi_P - \omega_P A_i)\, \delta_i \left\{\omega_C \exp(-\sigma_{i+8}\tau) + \varpi_C \exp(-\sigma_{i+12}\tau)\right\} \qquad \text{(IV.13)}$$

avec :

$$\begin{cases} \delta_1 = \omega_S\omega_T \qquad \delta_2 = \omega_S\varpi_T \qquad \delta_3 = \varpi_S\omega_T \qquad \delta_4 = \varpi_S\varpi_T \\ A_i = \dfrac{(1-q_i)\zeta_{sup,P}}{(q_i\zeta_{inf,P} - \zeta_{sup,P})} \end{cases}$$

$\sum_{i=1}^{4} \delta_i\, q_i$ est la probabilité conditionnelle de défaillance du BR après activation d'une faute indépendante dans P.

La relation (IV.13) peut s'écrire sous la forme :

$$r(\tau) = \sum_{i=1}^{16} \alpha_i \exp(-\beta_i\tau)) \quad \text{avec} \quad \sum_{i=1}^{16} \alpha_i = 1 \ \text{ et } \ \alpha_i,\ \beta_i \geq 0 \qquad \text{(IV.14)}$$

En utilisant les relations (IV.12) et (IV.14), on obtient l'expression suivante de R(t,t+u) :

$$R(t,t+u) = \frac{\sum_{i=1}^{16} \alpha_i \exp(-\beta_i\,(t+u))}{\sum_{i=1}^{16} \alpha_i \exp(-\beta_i t)} \qquad \text{(IV.15)}$$

Compte tenu de l'hypothèse $u << t$, on peut approcher R(t,t+u) par l'expression :

$$R(t,t+u) = 1 - h(t)\, u \qquad \text{(IV.16)}$$

où $h(t) = \dfrac{\sum_{i=1}^{16} \alpha_i\, \beta_i \exp(-\beta_i t)}{\sum_{i=1}^{16} \alpha_i \exp(-\beta_i t)}$

h(t) est l'intensité de défaillance caractérisant le comportement du BR.

Etant donné que $\sum_{i=1}^{16} \alpha_i = 1$ et $\alpha_i, \beta_i \geq 0$, on remarque alors que h(t) a une expression similaire à celle d'un modèle hyperexponentiel.

REMARQUE

A partir de la relation (IV.14), on peut obtenir directement l'expression de $r^*(\tau)$ correspondant au cas où on suppose un comportement en fiabilité stabilisée du BR par rapport à tous les processus de défaillance considérés. Il suffit de poser : $\omega_X = 0$ et $\zeta_{inf,X} = \lambda_X$, $X \in \{P, S, T, C\}$. On trouve :

$$r^*(\tau) = \exp -\left(\left(\left(\frac{\lambda_S}{\lambda_S+\gamma_S} + \frac{\lambda_T}{\lambda_T+\gamma_T}\frac{\gamma_S}{\lambda_S+\gamma_S}\right)\lambda_P+\lambda_C\right)\pi_P + \lambda_T \pi_T\right)\tau = \exp -(\lambda_{eq}\tau) \qquad \text{(IV.17)}$$

λ_{eq} est le taux de défaillance équivalent associé au BR en fiabilité stabilisée[2].

La relation (IV.17) peut être obtenue aussi en appliquant directement la technique d'agrégation à la chaîne de Markov du logiciel en fiabilité stabilisée de la figure IV.15. On peut remarquer à partir de la relation (IV.17) que le taux de défaillance équivalent du BR en fiabilité stabilisée est proportionnel :

i) au produit des taux de défaillance de P et S et de P et T,

ii) au taux d'activation des faute corrélées,

iii) et au taux de défaillance de T.

Ce résultat est analogue à celui obtenu dans [Arlat 88-b], où le comportement du BR est modélisé par une chaîne de Markov discrète à espace d'état discret ; le taux de défaillance équivalent du BR étant évalué en tenant compte du taux de sollicitation du logiciel.

A partir de la relation (IV.17), on obtient l'expression suivante de la fiabilité $R^*(t,t+u)$ du BR en fiabilité stabilisée :

$$R^*(t,t+u) = \frac{\exp-(\lambda_{eq}(t+u))}{\exp-(\lambda_{eq}t)} = \exp -(\lambda_{eq}u) \approx 1 - \lambda_{eq}u \qquad \text{(IV.18)}$$

Contrairement à R(t,t+u), la fiabilité du BR en fiabilité stabilisée ne dépend pas du temps et dépend uniquement de la durée de mission u.

IV.3.2.5.2 Variation de R(t,t+u) en fonction des paramètres du modèle

Afin d'étudier l'évolution de la fiabilité R(t,t+u) en fonction des différents paramètres du modèle, considérons l'exemple numérique suivant :

---

[2] Dans le cas où la chaîne de Markov caractérisant le comportement d'un système est telle que les états correspondant aux états non défaillants constituent un ensemble irréductible (fortement connexe), on peut évaluer un taux de défaillance équivalent λ_{eq} du système tel que la fiabilité $R(\tau)$ s'écrit [Pagès 80] :

$R(\tau) = \exp -(\lambda_{eq}\tau)$

$5 \le k_P \le 10$	$2 \le k_T \le 5$	$2 \le k_C \le 5$	$1 \le k_S \le 5$
$10^{-1} \le \frac{\lambda_T}{\lambda_P} \le 10^{-3}$	$10^{-2} \le \frac{\lambda_C}{\lambda_P} \le 10^{-3}$	$\gamma_P = \gamma_S = 10^6$ / h	$\gamma_T = 10^7$ / h

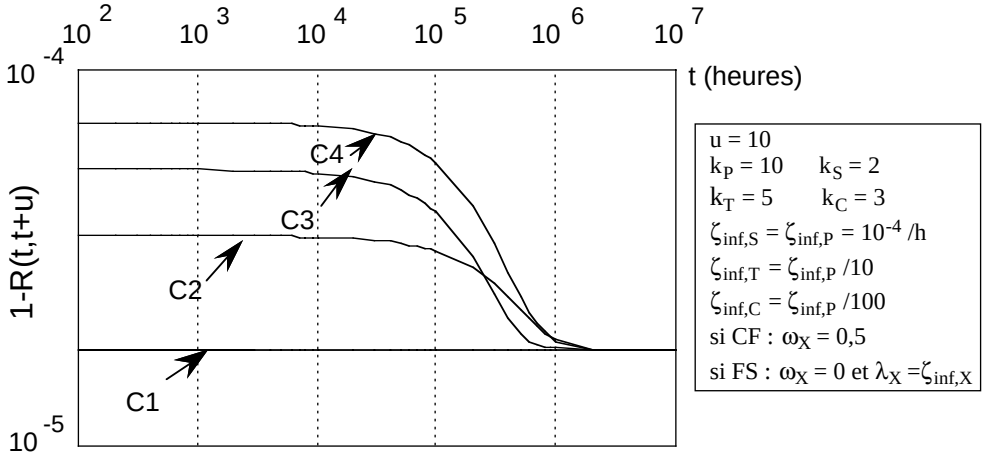
où k_X, $X \in \{P, S, T, C\}$ est le facteur de croissance associé à X défini par :

$$k_X = \omega_X \frac{\zeta_{sup,X}}{\zeta_{inf,X}} + \varpi_X$$

Compte tenu du fonctionnement d'un BR, P est exécuté plus souvent que S. Il est donc vraisemblable que l'exécution du BR conduise à l'activation et à la correction de plus de fautes dans P que dans S. C'est ce qui justifie l'ordre de grandeur choisi entre k_P et k_S.

Concernant le test d'acceptation T, compte tenu d'une part, de la simplicité de sa mise en œuvre comparé aux alternants et, d'autre part, de l'importance du test d'acceptation et de l'attention particulière qu'on lui réserve au cours de la conception et de la validation du BR, on peut supposer que la croissance de fiabilité de T est plus faible que celle des alternants.

On présente sur la figure IV.18, des courbes d'évolution de la fonction 1-R(t,t+u) en fonction du temps t. Ces courbes permettent d'observer l'évolution de la fiabilité du BR en tenant compte de la croissance de fiabilité de ses composants. Les différents cas considérés sont résumés sur cette figure.



Courbe	P	S	T	C	Commentaires
C1	FS	FS	FS	FS	on ne tient pas compte du phénomène de croissance de fiabilité
C2	FS	FS	FS	CF	on considère uniquement la croissance de fiabilité résultant de l'élimination des fautes corrélées
C3	FS	FS	CF	FS	on considère uniquement la croissance de fiabilité résultant de l'élimination des fautes indépendantes dans T
C4	CF	CF	CF	CF	on tient compte de la croissance de fiabilité résultant de l'élimination de toutes les fautes considérées

FS : fiabilité stabilisée CF : croissance de fiabilité

Figure IV.18 : Courbes d'évolution de 1- R(t,t+u) en fonction du temps.

La courbe C1 correspond au comportement du BR en fiabilité stabilisée ; R(t,t+u) étant évaluée en considérant les valeurs minimales de λ_X, $X \in \{P, S, T, C\}$. Les courbes C2, C3 et C4 illustrent l'évolution de la fiabilité du BR dans le cas où on tient compte de la croissance de fiabilité d'un ou plusieurs composants du logiciel. Les courbes correspondant à l'évolution de 1-R(t,t+u) en tenant compte de la croissance de fiabilité de P uniquement, ou bien de celle de P et de S à la fois, sont confondues avec la courbe C1. Ainsi, pour les valeurs choisies des

paramètres, la croissance de fiabilité de P et de S n'a pas d'influence significative sur l'évolution de 1-R(t,t+u).

En comparant la courbe C1 aux autres courbes, on peut remarquer qu'il existe une différence significative entre les estimations effectuées en tenant compte du phénomène de croissance de fiabilité et celles obtenues en supposant un comportement du BR en fiabilité stabilisée.

Une étude de sensibilité sur l'influence des différents paramètres qui agissent sur le comportement du BR conduit aux principales conclusions suivantes :

- la croissance de fiabilité du BR est essentiellement conditionnée par la croissance de fiabilité qui est liée à l'élimination des fautes corrélées et aussi à la croissance de fiabilité du test d'acceptation. Ce résultat est prévisible puisque l'activation de fautes corrélées ou bien de fautes dans le test conduit à la défaillance du BR. La différence qui existe entre les courbes C4 et C1 dépend essentiellement des valeurs de k_T et k_C, c'est-à-dire de la croissance de fiabilité observée, par rapport à ses deux sources de défaillance, durant la période considérée,

- la croissance de fiabilité de P ou de S n'a d'influence significative sur l'évolution de la fiabilité du BR que si λ_P est très grand par rapport à λ_T et λ_C et que les taux d'exécution γ_P et γ_S ne sont pas très élevés (soit quand λ_P λ_T et λ_P λ_S sont du même ordre de grandeur que λ_T et λ_C),

- La vitesse d'évolution du BR vers le comportement asymptotique en fiabilité stabilisée est conditionnée par l'allure d'évolution des intensités de défaillance $h_C(t)$ et $h_T(t)$: le comportement en fiabilité stabilisée du BR est atteint au bout d'un temps t qui est de l'ordre de l'inverse de la valeur minimale de $h_C(\infty)$ et $h_T(\infty)$.

IV.3.2.6 Conclusion

Dans ce paragraphe, nous avons appliqué l'approche de modélisation de systèmes multi-composant pour l'évaluation de la croissance de fiabilité d'un bloc de recouvrement. Nous avons étudié plus particulièrement l'influence de la croissance de fiabilité des deux alternants du système et du test d'acceptation sur l'évolution de la croissance de fiabilité du bloc de recouvrement, en considérant deux types de fautes : les fautes indépendantes et les fautes corrélées. Nous avons remarqué qu'il existe une différence significative entre les estimations effectuées en tenant compte du phénomène de croissance de fiabilité et celles qui sont obtenues en faisant l'hypothèse d'un comportement du BR en fiabilité stabilisée ; une telle différence est due essentiellement au phénomène de croissance de fiabilité résultant de l'élimination progressive des fautes corrélées et des fautes indépendantes introduites dans le test d'acceptation.

Compte tenu du fait que la fiabilité du BR croît au cours de son cycle de vie en raison de l'élimination des fautes de conception, il est nécessaire de tenir compte du phénomène de croissance de fiabilité afin d'obtenir des estimations réalistes du comportement du BR. A notre connaissance, l'évaluation et la modélisation de la croissance de fiabilité des logiciels tolérants aux fautes n'a pas été considérée dans la littérature. Les études publiées sur la modélisation de

ce type de logiciels se sont limitées à l'analyse de leur comportement en fiabilité stabilisée [Hecht 79, Grnarov 80, Laprie 84-c, Muluzzani 85, Cha 86, Tso 86, Scott 87, Arlat 88-a, Csenki 89, Pucci 90]. L'étude que nous venons d'effectuer constitue ainsi une nouveauté dans le domaine.

La même approche qui a été considérée dans ce paragraphe peut être utilisée pour évaluer la croissance de fiabilité d'un bloc de recouvrement en distinguant les défaillances bénignes qui résultent de la manifestation d'erreurs détectées par le test d'acceptation et les défaillances catastrophiques qui sont dues à la manifestation d'erreurs qui ne sont pas détectées par le test (c'est-à-dire la croissance de sécurité du BR). On peut aussi appliquer l'approche proposée en considérant un bloc de recouvrement qui est constitué de plus de deux alternants pour étudier dans quelle mesure l'introduction davantage d'alternants peut conduire à l'amélioration de la sûreté de fonctionnement du logiciel tolérant aux fautes.

CONCLUSION

Dans ce chapitre, nous avons d'abord présenté une approche générale permettant de modéliser la croissance de sûreté de fonctionnement d'un système multi-composant à partir de celle de ses différents composants, nous avons ensuite illustré l'approche proposée sur deux systèmes en considérant les deux principales mesures de la sûreté de fonctionnement : la fiabilité et la disponibilité.

L'approche proposée est applicable à des systèmes constitués de composants matériels et logiciels. Elle est basée sur l'utilisation des réseaux de Petri stochastiques généralisés pour, d'une part, générer une chaîne de Markov caractérisant le comportement en fiabilité stabilisée du système considéré, et d'autre part, appliquer de façon mécanique, et donc transparente aux utilisateurs, une transformation basée sur la représentation markovienne du modèle hyperexponentiel en temps continu permettant de prendre en compte la croissance de fiabilité des différents composants du système pour évaluer ses mesures de sûreté de fonctionnement. L'utilisation d'une approche markovienne permet de réutiliser tous les résultats existants concernant la construction et le traitement de modèles de comportement de systèmes complexes qui sont basés sur les processus de Markov homogènes.

L'approche de modélisation définie dans ce chapitre, permet d'intégrer dans un même modèle l'évaluation de la croissance de fiabilité et de la croissance de disponibilité du matériel et du logiciel, et de fournir, ainsi, aux utilisateurs des mesures concernant la sûreté de fonctionnement du système global. Elle offre également aux concepteurs le moyen de suivre l'évolution de la sûreté de fonctionnement de leur système en tenant compte de la croissance de sûreté de fonctionnement de ses différents composants.

CHAPITRE V

APPROCHE DE MODéLISATION D'UN LOGICIEL MULTI-COMPOSANT BASéE SUR LE MODÈLE HYPEREXPONENTIEL EN TEMPS DISCRET

INTRODUCTION

Dans ce chapitre, nous définissons une approche de modélisation basée sur le modèle hyperexponentiel en temps discret qui vise à atteindre les deux objectifs suivants :

- modéliser la croissance de fiabilité d'un logiciel multi-composant en fonction du nombre d'exécutions effectuées et de la croissance de fiabilité de ses différents composants,
- prendre en compte les taux d'exécution des différents composants du logiciel pour évaluer la fiabilité du logiciel telle qu'elle est perçue dans le temps par ses utilisateurs.

Les études effectuées depuis une vingtaine d'années dans le domaine de la croissance de fiabilité du logiciel se sont essentiellement focalisées sur le problème de l'évaluation et de la prévision de mesures de sûreté de fonctionnement du logiciel dans un environnement donné sans tenir compte explicitement des caractéristiques de l'environnement dans lequel il est utilisé. L'hypothèse considérée dans la plupart des modèles de croissance de fiabilité est que l'environnement dans lequel le logiciel est validé avant sa mise en opération est équivalent à l'environnement dans lequel il est utilisé pendant sa vie opérationnelle.

Or, un logiciel est généralement destiné à être utilisé dans plusieurs environnements qui diffèrent :

- par le profil d'utilisation du logiciel qui est déterminé par la distribution du domaine des entrées et par la fréquence de sollicitation des différents services délivrés par le logiciel,
- par les caractéristiques matérielles des systèmes sur lesquels le logiciel est exécuté.

Le comportement d'un logiciel est ainsi lié aux caractéristiques de l'environnement dans lequel il est utilisé. Comme nous l'avons montré dans le chapitre I (cf. paragraphe I.2.2.2), le changement de l'environnement d'utilisation du logiciel peut se traduire par une variation des mesures qui caractérisent l'évolution dans le temps de sa sûreté de fonctionnement. C'est le cas par exemple quand on passe de l'environnement de validation du logiciel vers la vie opérationnelle, l'intensité de défaillance estimée à partir des données collectées au cours de la phase de validation peut changer d'allure et avoir des allures différentes dans les différents sites dans lequel le logiciel est installé [Ehrlich 90, Jones 91].

Pratiquement, il est très difficile et inconcevable de valider un logiciel par rapport à tous ses profils d'utilisation possibles. Le problème qui se pose alors est comment estimer les mesures de sûreté de fonctionnement d'un logiciel pour un environnement donné à partir des mesures effectuées dans un autre environnement ?

La raison pour laquelle les modèles en temps continu peuvent difficilement donner une réponse à ce problème est que ces modèles caractérisent l'évolution de la fiabilité du logiciel telle qu'elle est perçue par les utilisateurs du logiciel dans un environnement donné sans tenir compte de façon explicite des caractéristiques de l'environnement considéré.

Le modèle de croissance de fiabilité en temps discret que nous avons défini dans le chapitre III peut donner un début de réponse à ce problème. En effet, ce modèle caractérise l'évolution du comportement du logiciel en fonction du nombre d'exécutions effectuées en tenant compte des corrections introduites sur le logiciel. Il modélise en fait le phénomène de croissance de fiabilité qui est lié à l'évolution du domaine de défaillance du logiciel au cours de son cycle de vie. Néanmoins, comme nous l'avons constaté dans le chapitre III, l'enrichissement du modèle par la prise en compte du taux d'exécution du logiciel dans l'environnement d'utilisation considéré permet d'évaluer les mesures de sûreté de fonctionnement du logiciel qui caractérisent son comportement tel qu'il est perçu dans le temps par ses utilisateurs. Dans le chapitre III, nous avons étudié le comportement du logiciel selon une vue "boîte noire", dans ce chapitre, nous considérons le cas d'un logiciel multi-composant ; l'environnement d'utilisation du logiciel est alors caractérisé par les taux d'exécution des différents composants du logiciel.

Cependant, dans le cadre de l'approche que nous proposons pour prendre en compte les caractéristiques de l'environnement d'utilisation du logiciel dans l'évaluation de ses mesures de sûreté de fonctionnement, nous supposons que la sélection des données du domaine des entrées du logiciel s'effectue de façon identique en probabilité dans les différents environnements d'utilisation du logiciel et que cette probabilité ne varie pas dans le temps. Une telle hypothèse est un peu restrictive compte tenu du fait que pour certains logiciels, le domaine de défaillance peut évoluer en fonction des conditions de sollicitation du système [Laprie 89] : par exemple, certaines fautes du logiciel ne peuvent être activées que suite à l'occurrence combinée de plusieurs événements, occurrence qui n'a lieu que si la charge appliquée au système dépasse un certain seuil. Par conséquent, étant donné que l'approche que nous proposons dans ce chapitre ne tient compte que de la variation de la fréquence de sollicitation des différents composants du logiciel, elle doit être vue comme une première étape dans la résolution du problème très complexe que nous venons de soulever.

Dans ce chapitre, nous considérons des logiciels multi-composant qui sont tels qu'à chaque instant d'utilisation du logiciel un seul composant est activé.

Nous commencerons dans une première étape par introduire l'approche proposée à travers un exemple simple. Nous généraliserons l'approche, dans une deuxième étape, et nous l'appliquerons pour évaluer la croissance de fiabilité d'un logiciel non tolérant aux fautes et également pour effectuer une étude détaillée de la fiabilité d'un bloc de recouvrement.

V.1 INTRODUCTION DE L'APPROCHE

Considérons un bloc de recouvrement (BR) constitué de deux alternants : un primaire et un secondaire et supposons que le test d'acceptation associé est parfait. On se propose d'étudier le comportement du BR par rapport à l'activation de fautes indépendantes dans le primaire et dans le secondaire. On néglige, dans ce cas, l'impact des fautes corrélées sur le comportement du logiciel.

La chaîne de Markov décrivant le comportement en temps discret du logiciel en fiabilité stabilisée est donnée par la figure V.1 : p_p et p_S sont respectivement les probabilités de défaillance à l'exécution du primaire et du secondaire.

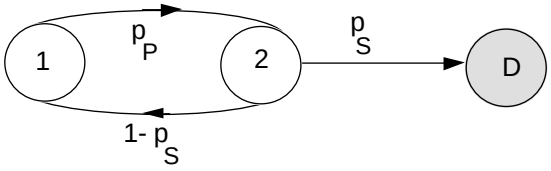


Figure V.1 : Bloc de recouvrement en fiabilité stabilisée

Supposons que la fiabilité du primaire croisse au cours de son cycle de vie et qu'une telle croissance de fiabilité soit décrite par un modèle hyperexponentiel en temps discret de paramètres θ_P, $p_{sup,P}$ et $p_{inf,P}$. La question qui se pose maintenant est comment tenir compte de la croissance de fiabilité du primaire dans l'évaluation des mesures de fiabilité du BR ?

V.1.1 Technique de transformation dans le cadre d'une approche multi-composant

En tenant compte de l'interprétation markovienne du modèle hyperexponentiel en temps discret et en faisant le lien avec le modèle hyperexponentiel en temps continu, nous pouvons définir une approche similaire à celle que nous avons proposée pour modéliser en temps continu la croissance de sûreté de fonctionnement d'un système multi-composant à partir de celle de ses différents composants. La figure V.2 résume les différentes étapes considérées pour introduire la transformation dans le cas discret en se basant sur l'approche présentée dans le chapitre IV. Les différents modèles présentés sont commentés dans la suite.

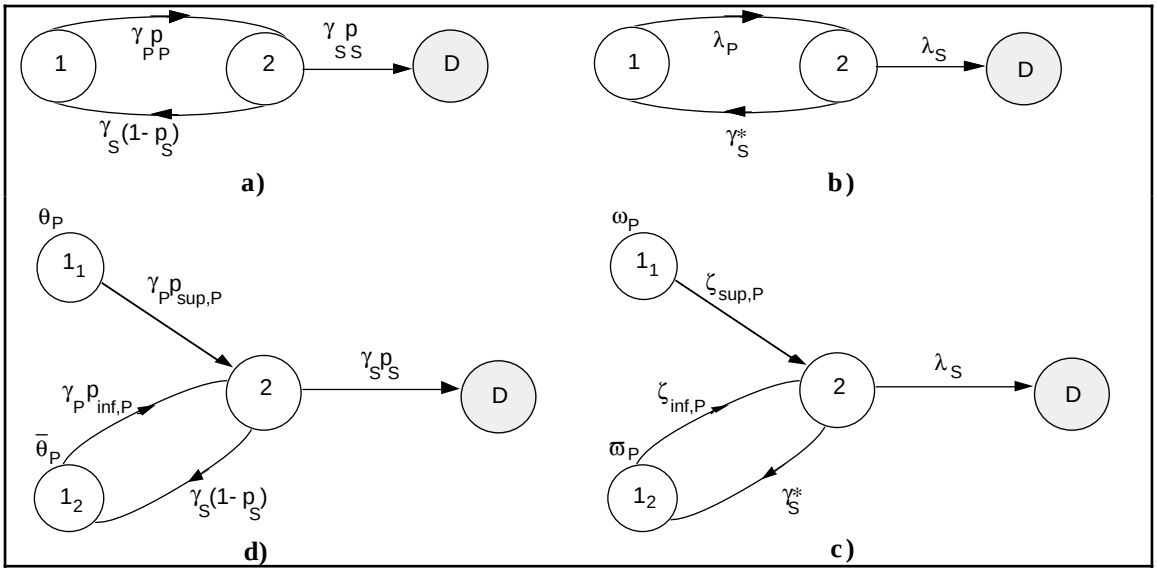


Figure V.2 : Chaînes de Markov en temps continu du BR en fiabilité stabilisée et en croissance de fiabilité

La chaîne de Markov en temps discret présentée sur la figure V.1 peut être interprétée comme la chaîne immergée déduite de la chaîne de Markov en temps continu présentée sur la

figure V.2-a. Cette dernière caractérise le comportement du BR en tenant compte des temps d'exécution du primaire et du secondaire.

Soient γ_P et γ_S respectivement les taux d'exécution du primaire et du secondaire. Posons :

$$\begin{cases} \lambda_P = \gamma_P p_P = \text{taux de défaillance du primaire} \\ \lambda_S = \gamma_S p_S = \text{taux de défaillance du secondaire} \\ \gamma_S^* \approx (1-p_S)\gamma_S \approx \gamma_S = \text{taux d'exécution sans défaillance du secondaire} \end{cases}$$

Avec cette notation, la chaîne de Markov de la figure V.2-a est identique à la chaîne de Markov présentée sur la figure V.2-b. La différence qui existe entre les deux modèles est : dans le premier modèle, nous modélisons le comportement en fiabilité stabilisée du BR en distinguant les probabilités d'activation des fautes associées à un composant du taux d'exécution du composant correspondant, et dans le deuxième modèle ces deux paramètres sont agrégés.

Considérons maintenant le cas où la fiabilité du primaire croît au cours de son cycle de vie. Si on applique la technique de transformation basée sur le modèle hyperexponentiel en temps continu définie dans le chapitre IV à la chaîne de Markov de la figure V.2-b, on obtient la chaîne de Markov présentée sur la figure V.2-c où ω_P, $\zeta_{sup,P}$, $\zeta_{inf,P}$ sont les paramètres qui caractérisent la croissance de fiabilité du primaire en temps continu.

En exprimant $\zeta_{sup,P}$ et $\zeta_{inf,P}$ en fonction de γ_P, la chaîne de Markov de la figure V.2-c est alors équivalente à la chaîne de Markov de la figure V.2-d dans laquelle :

$$\begin{cases} \zeta_{sup,P} = \gamma_P p_{sup,P} \\ \zeta_{inf,P} = \gamma_P p_{inf,P} \\ \omega_P = \theta_P \end{cases}$$

En considérant respectivement les figures V.2-a et V.2-d, on peut conclure que la chaîne de Markov de la figure V.2-d résulte de la transformation de la chaîne de Markov de la figure V.2-a en considérant la représentation markovienne du modèle hyperexponentiel en temps discret pour prendre en compte la croissance de fiabilité du primaire. La transformation est illustrée sur la figure V.3 dans laquelle les chaînes de Markov V.3-a et V.3-b sont respectivement les chaînes immergées associées aux chaînes de Markov caractérisant le comportement du BR en fiabilité stabilisée (figure V.2-a) et en croissance de fiabilité (figure V-2-d).

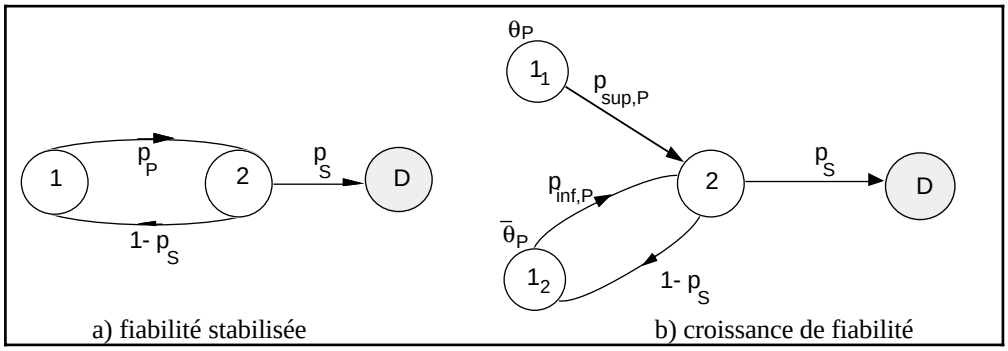


Figure V.3 : Chaînes de Markov en temps discret d'un BR en croissance de fiabilité

Ainsi, la modélisation de la croissance de fiabilité d'un logiciel multi-composant en fonction de celle de ses différents composants est effectuée de la même manière que dans le cas continu : au lieu de transformer des taux de défaillance en utilisant la représentation markovienne du modèle hyperexponentiel en temps continu, on transforme des probabilités de défaillance à l'exécution en considérant l'interprétation markoviennne du modèle hyperexponentiel en temps discret.

Les avantages de dissocier les probabilités de défaillance à l'exécution des taux d'exécution du système sont :

- d'une part, d'évaluer les mesures de fiabilité d'un système en fonction du nombre d'exécutions effectuées en traitant la chaîne de Markov transformée en temps discret (figure V.3-b),

- d'autre part, de tenir compte des caractéristiques de l'environnement d'utilisation du système considéré par la prise en compte des taux d'exécution des ses composants, dans l'évaluation des mesures de sûreté de fonctionnement (traitement de la chaîne de Markov de la figure V.2-d).

L'évaluation des mesures de fiabilité du logiciel en temps continu peut être effectuée en considérant la même approche que celle que nous avons considérée dans le chapitre IV : c'est-à-dire en utilisant les techniques classiques, numériques ou analytiques, de traitement de chaînes de Markov. Cependant, l'évaluation des mesures de fiabilité du logiciel en fonction du nombre d'exécutions du logiciel global est moins immédiate. Nous développerons ce point dans le paragraphe suivant.

V.1.2 évaluation des mesures de fiabilité en temps discret

Considérons la chaîne de Markov transformée de la figure V.3-b. Les états 1_1 et 1_2 de la chaîne de Markov caractérisent l'exécution du bloc de recouvrement global et l'état 2 caractérise l'exécution du secondaire. On se propose d'étudier l'évolution de la croissance de fiabilité du bloc de recouvrement en fonction du nombre total d'exécutions du logiciel.

L'évaluation de la probabilité de défaillance à l'exécution qui caractérise le comportement du système en fonction du nombre total d'exécutions du BR, se ramène à l'évaluation de la distribution de probabilité du nombre de visites dans une classe d'états du système : ceux qui identifient la sollicitation du système pour une nouvelle exécution. Le nombre d'exécutions du BR correspond au nombre d'exécutions du primaire. Ainsi, l'évaluation de la probabilité de défaillance à l'exécution du BR en fonction du nombre d'exécutions effectuées se ramène à l'évaluation de la distribution du nombre de visites dans les états 1_1 et 1_2 de la chaîne de Markov considérée ; l'état 2 correspond quant à lui à l'exécution du secondaire.

Les techniques classiques de traitement d'une chaîne de Markov en temps discret permettent de calculer la distribution des probabilités d'occupation des états du système en fonction du nombre total de transitions effectuées au sein de la chaîne. Or, dans le cas qui nous concerne, cette mesure n'est pas intéressante car on cherche à évaluer les mesures de fiabilité d'un système

en fonction du nombre de visites dans un sous-ensemble d'états qui caractérisent l'exécution du système et non pas en fonction du nombre total de transitions effectuées.

L'évaluation de la distribution du nombre total de visites dans un sous-ensemble d'états transitoires s'est généralement limitée au cas où ce sous-ensemble se réduit à un seul état [Kemeny 59]. A notre connaissance, l'évaluation de la distribution du nombre total de visites dans un sous-ensemble quelconque d'états transitoires n'a été considérée que dans [Marie 86].

Nous présenterons dans la suite, une méthode d'évaluation des mesures de fiabilité caractérisant le comportement d'un logiciel multi-composant en fonction du nombre total d'exécutions effectuées qui est basée sur les résultats présentés dans [Marie 86]. Nous appliquerons ensuite cette méthode à l'exemple que nous venons de présenter.

V.1.2.1 Présentation de la méthode d'évaluation des mesures de fiabilité

Le comportement d'un système est décrit par une chaîne de Markov en temps discret et à espace d'états discret E.

Soit E ={1, 2,…, m, m+1}l'ensemble des états du système où les m premiers états sont transitoires et le m+1 ième état est absorbant.

Soit B le sous-ensemble des états transitoires qui caractérisent le début d'exécution du système. On peut alors partitionner l'espace E sous la forme $E = B \cup B^c \cup D$ où :

$B = \{1, 2,\ldots, l\} \qquad 1 \leq l \leq m$

$B^c = \{l+1,\ldots, m\}$ si $l < m$

et $D=\{m+1\}$

Soit :

- $P = (p_{i,j})$ $1 \leq i \leq m+1$ et $1 \leq j \leq m+1$, la matrice des probabilités de transition associée à la chaîne de Markov en temps discret,
- Q la matrice déduite de P en supprimant la (m+1)ième ligne et la (m+1)ième colonne de P,
- Q_l, D, C, Q_{m-l}, les matrices de tailles respectives (l,l), (l,m-l), (m-l,l) et (m-l,m-l) telles que :

  $$Q = \begin{pmatrix} Q_l & D \\ C & Q_{m-l} \end{pmatrix}$$

- I_x la matrice identité de dimension (x,x),
- $\mathbb{1}_x$ le vecteur colonne de dimension x dont toutes les composantes sont égales à 1,
- $\underline{\alpha} = (\alpha_1, \alpha_2,\ldots, \alpha_l, \alpha_{l+1},\ldots, \alpha_m, \alpha_{m+1})$ le vecteur ligne des probabilités d'occupation initiale des états de la chaîne de Markov,
- $\underline{\alpha}_B = (\alpha_1, \alpha_2,\ldots,\alpha_l)$ et $\underline{\alpha}_{B^c} = (\alpha_{l+1}, \ldots,\alpha_m)$,

. N_B la variable aléatoire "nombre de visites dans les états de B avant absorption". En d'autres termes, N_B indique le nombre d'exécutions du système avant défaillance.

La fonction de probabilité associée à la variable aléatoire N_B est définie par :

$f_{N_B}(n) = Pr(N_B = n)$

$f_{N_B}(n)$ est la probabilité de visiter n fois le sous-ensemble B avant d'atteindre l'état absorbant. En d'autres termes, $f_{N_B}(n)$ correspond à la probabilité de défaillance du système après n exécutions.

$f_{N_B}(n)$ est donnée par le théorème suivant [Marie 86] :

THéORèME :

$$f_{N_B}(n) = \underline{\beta}\, H^{n-1} (I - H)\underline{1}_l \qquad n \geq 1 \qquad (V.1)$$

où :

$\underline{\beta} = \underline{\alpha}_B + \underline{\alpha}_B{}^c (I_{m-l} - Q_{m-l})^{-1} C$ (vecteur ligne de dimension l)

$H = Q_l + D(I_{m-l} - Q_{m-l})^{-1} C$ (matrice carrée de (l,l))

$\underline{\beta} = (\beta_j)$ j= 1…l où β_j est la probabilité que le premier état de B visité soit l'état j.

Si $\underline{\alpha}_B{}^c = 0$ alors $\underline{\beta}$ correspond au vecteur des probabilités d'occupation initiale des états de B.

$H^{n-1}(i,j)$ est la probabilité que, partant de $i \in B$, le n-ième état de B visité soit l'état j sachant qu'avant d'atteindre l'état j le système a évolué librement dans l'espace E.

Connaissant la fonction probabilité de masse de la variable aléatoire N_B, on peut évaluer la distribution de probabilité associée, $F_{N_B}(n)$.

$$F_{N_B}(n) = Pr(N_B \leq n) = \sum_{s=1}^{n} P(N_B = s) = \sum_{s=1}^{n} \underline{\beta}\, H^{s-1} (I - H)\underline{1}_l \qquad (V.2)$$

$$F_{N_B}(n) = \underline{\beta} \left\{ I - H^n \right\} \underline{1}_l \qquad (V.3)$$

On peut alors obtenir la fonction de survie r(n) donnant la probabilité que la chaîne ne soit pas absorbée après n visites dans le sous-ensemble B. r(n) caractérise la fiabilité du système considéré après n exécutions. On a :

$$r(n) = 1 - F_{N_B}(n) = 1 - \underline{\beta} \left\{ I - H^n \right\} \underline{1}_l \qquad (V.4)$$

si $\sum_{j=1}^{+} \mathbb{B}_j = 1$, alors :

$$r(n) = \mathbb{B}\, H^n\, \mathbb{1}| \qquad (V.5)$$

Soit $\lambda(n)$ la fonction de hasard associée à la variable aléatoire N_B. $\lambda(n)$ est le taux de défaillance en temps discret évalué à partir de la chaîne de Markov considérée.

Par définition, $\lambda(n)$ est donnée par :

$$\lambda(n) = \frac{f_{N_B}(n)}{r(n-1)} \qquad (V.6)$$

Sachant que $f_{N_B}(n) = r(n-1) - r(n)$, l'expression (V.6) devient :

$$\lambda(n) = 1 - \frac{r(n)}{r(n-1)} \qquad (V.7)$$

D'après les propriétés du modèle hyperexponentiel en temps discret, le taux de défaillance en temps discret $\lambda(n)$ à la même allure que P(n), la probabilité de défaillance du système à l'exécution n. Le taux de défaillance en temps discret $\lambda(n)$ calculé à partir de la chaîne de Markov transformée peut être interprété comme la probabilité de défaillance non conditionnelle caractérisant l'évolution de la croissance de fiabilité du système à partir de celle de ses différents composants.

Soit n le nombre d'exécutions effectuées par le système. Si on prend en compte l'hypothèse d'indépendance de la probabilité de défaillance à l'exécution du système global vis-à-vis du processus d'occurrence de défaillance, on peut évaluer $R(n,n+n_0)$ qui est la probabilité de fonctionnement sans défaillance du système global durant n_0 exécutions à partir de l'instant n.

Compte tenu des remarques précédentes, $R(n,n+n_0)$ est donnée par l'expression suivante :

$$R(n,n+n_0) = \prod_{i=n+1}^{n+n_0} 1- P(i) = \prod_{i=n+1}^{n+n_0} \frac{r(i)}{r(i-1)} \qquad (V.8)$$

soit :

$$R(n,n+n_0) = \frac{r(n+n_0)}{r(n)} \qquad (V.9)$$

On trouve ainsi une relation équivalente à celle obtenue dans le cas continu où : $R(t,t+u) = \frac{r(t+u)}{r(t)}$.

RéDUCTION DE LA CHAÎNE TRANSFORMéE

L'application de la technique de transformation pour la modélisation de la croissance de fiabilité d'un système multi-composant conduit à l'augmentation de l'espace des états du système considéré. L'utilisation de techniques d'agrégation pour réduire l'espace des états du système permet de faciliter la tâche de traitement de la chaîne transformée pour évaluer les mesures de fiabilité.

Compte tenu du théorème précédent, on peut définir à partir d'une chaîne de Markov d'espace des états E une chaîne de Markov réduite constituée d'un état absorbant et du sous-ensemble d'états B, défini précédemment, qui est telle que la distribution du nombre de transitions avant d'atteindre l'état absorbant est égale à la distribution du nombre de visites dans le sous-ensemble B de la chaîne initiale. La chaîne de Markov réduite ainsi définie est caractérisée par la matrice des probabilités de transition $T = (t_{ij})$ $1 \leq i \leq l+1$, $1 \leq j \leq l+1$, donnée par :

$$T = \begin{pmatrix} H & (I - H)\mathbb{1}_l \\ 0 & 1 \end{pmatrix} \qquad \text{(V.10)}$$

où H est la matrice des probabilités de transition définie précédemment.

On peut vérifier que l'expression de la distribution du nombre de transitions avant absorption obtenue à partir de la matrice T est égale à celle calculée directement à partir de la chaîne de Markov initiale qui est donnée par la relation (V.3).

Ainsi, pour évaluer les mesures de fiabilité caractérisant la croissance de fiabilité d'un système multi-composant on peut utiliser cette dernière propriété pour réduire l'espace des états du système et évaluer directement les mesures de fiabilité à partir de la chaîne réduite.

V.1.2.2 Application de la méthode

Considérons la chaîne de Markov transformée décrivant le comportement du BR dans le cas où on tient compte de la croissance de fiabilité du primaire uniquement (figure V.3-b). Conformément aux notations considérées dans le paragraphe précédent, on a :

$B = \{1,2\}$, $l = 2$ et $m=3$

La matrice P des probabilités de transition associée à la chaîne de Markov transformée s'écrit :

$$P = \begin{pmatrix} 1-p_{sup,P} & 0 & p_{sup,P} & 0 \\ 0 & 1-p_{inf,P} & p_{inf,P} & 0 \\ 0 & 1-p_S & 0 & p_S \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

La matrice Q déduite de P en éliminant la ligne et la colonne qui caractérisent l'état défaillant est donnée par :

$$Q = \begin{pmatrix} Q_2 & D \\ C & Q_1 \end{pmatrix} \quad \text{avec :}$$

$$Q_2 = \begin{pmatrix} 1-p_{sup,P} & 0 \\ 0 & 1-p_{inf,P} \end{pmatrix} \quad D = \begin{pmatrix} p_{sup,P} \\ p_{inf,P} \end{pmatrix} \quad C = \begin{pmatrix} 0 & 1-p_S \end{pmatrix} \quad \text{et} \quad Q_1 = (0)$$

D'où : $H = Q_2 + D(I_1 - Q_1)^{-1}C$

$$H = \begin{pmatrix} 1-p_{sup,P} & p_{sup,P}(1-p_S) \\ 0 & 1-p_{inf,P}p_S \end{pmatrix} \qquad (V.11)$$

Par suite :

$$(I - H)\mathbb{1}_2 = \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 1-p_{sup,P} & p_{sup,P}(1-p_S) \\ 0 & 1-p_{inf,P}p_S \end{pmatrix}\right)\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} p_S p_{sup,P} \\ p_S p_{inf,P} \end{pmatrix} \qquad (V.12)$$

Pour évaluer la fonction de survie r(n) définie par la relation (V.5), il suffit d'évaluer H^n. Pour cela, on peut utiliser la transformée géométrique, dite aussi transformée en z [Howard 71].

Soit $\mathcal{T}_{\mathcal{G}}^{-1}$ l'opérateur transformée géométrique inverse. On a :

$$H^n = \mathcal{T}_{\mathcal{G}}^{-1}(I - Hz)^{-1} \qquad (V.13)$$

$$I\text{-}Hz = \frac{1}{1-(1-p_{sup,P})\,z}\begin{pmatrix} 0 & \frac{p_{sup,P}(1-p_S)}{p_{inf,P}p_S - p_{sup,P}} \\ 0 & 0 \end{pmatrix} + \frac{1}{1-(1-p_{inf,P}p_S)\,z}\begin{pmatrix} 0 & \frac{-p_{sup,P}(1-p_S)}{p_{inf,P}p_S - p_{sup,P}} \\ 0 & 1 \end{pmatrix}$$

D'où :

$$H^n = (1-p_{sup,P})^n\begin{pmatrix} 1 & \frac{p_{sup,P}(1-p_S)}{p_{inf,P}p_S - p_{sup,P}} \\ 0 & 0 \end{pmatrix} + (1-p_{inf,P}p_S)^n\begin{pmatrix} 0 & \frac{-p_{sup,P}(1-p_S)}{p_{inf,P}p_S - p_{sup,P}} \\ 0 & 1 \end{pmatrix} \qquad (V.14)$$

Le vecteur des probabilités initiales d'occupation des états 1_1 et 1_2 est donné par :

$$\mathbb{B} = (\theta_P, \overline{\theta}_P) \qquad (V.15)$$

En utilisant les relations (V.5), (V.15) et (V.15), on obtient l'expression suivante de la fonction de survie r(n) :

$$r(n) = \theta_P(1-p_{sup,P})^n + \overline{\theta}_P(1-p_{inf,P}p_S)^n + \theta_P\frac{p_{sup,P}(1-p_S)}{p_{inf,P}\,p_S - p_{sup,P}}\left\{(1-p_{sup,P})^n - (1-p_{inf,P}p_S)^n\right\} \qquad (V.16)$$

Posons :

$$\Phi = \frac{p_S(p_{inf,P} - p_{sup,P})}{p_{inf,P}\, p_S - p_{sup,P}} \qquad (V.17)$$

La relation (V.16) s'écrit alors sous la forme :

$$r(n)= \Phi\theta_P(1-p_{sup,P})^n + (1-\Phi\theta_P)(1-p_{inf,P}p_S)^n \qquad (V.18)$$

Compte tenu des relations (V.7) et (V.18), la probabilité de défaillance à l'exécution n du bloc de recouvrement est donnée par :

$$P(n) = \lambda(n) = \frac{\Phi\theta_P p_{sup,P}(1-p_{sup,P})^{n-1} + (1-\Phi\theta_P)p_{inf,P}p_S(1-p_{inf,P}p_S)^{n-1}}{\Phi\theta_P(1-p_{sup,P})^{n-1} + (1-\Phi\theta_P)(1-p_{inf,P}p_S)^{n-1}} \qquad (V.19)$$

$\Phi\theta_P$ vérifie la condition suivante :

$$0 \le \Phi\theta_P \le 1 \qquad (V.20)$$

On remarque ainsi que la probabilité de défaillance à l'exécution du bloc de recouvrement correspond à un modèle hyperexponentiel en temps discret de paramètres $\Phi\theta_P$, $p_{sup,P}$ et $p_{inf,P}p_S$. Par conséquent, P(n) décroît au fur et à mesure que le nombre d'exécutions du bloc de recouvrement augmente et tend asymptotiquement vers une constante. En d'autres termes, la fiabilité du BR tend à croître au fur et à mesure de l'exécution du logiciel et atteint un comportement asymptotique en fiabilité stabilisée. Les valeurs maximale et asymptotique de P(n) sont données respectivement par :

$$\begin{cases} P(1) = \Phi\theta_P p_{sup,P} + (1-\Phi\theta_P)p_{inf,P}\, p_S = \theta_P\, p_{sup,P}\, p_S + \overline{\theta}_P\, p_{inf,P}\, p_S \\ P(\infty) = p_{inf,P}\, p_S \end{cases} \qquad (V.21)$$

Soit k (respectivement k_P) le facteur de croissance caractérisant l'évolution de la probabilité de défaillance à l'exécution du bloc de recouvrement (respectivement du primaire) :

$$k = \frac{\Phi\theta_P p_{sup,P} + (1-\Phi\theta_P)p_{inf,P}\, p_S}{p_{inf,P}\, p_S} \qquad (V.22)$$

$$k_P = \frac{\theta_P\, p_{sup,P} + \overline{\theta}_P\, p_{inf,P}}{p_{inf,P}} \qquad (V.23)$$

Compte tenu de la relation (V.21), on a : $k = k_P$. On vérifie ainsi que l'écart entre la probabilité P(n) est la valeur asymptotique P(∞) est conditionné par la croissance de fiabilité du primaire. Par ailleurs, si on pose $\theta_P = 0$ et $p_{inf,P} = p_P$, on retrouve l'expression de la probabilité de défaillance P*(n) caractérisant le comportement du BR dans le cas où la fiabilité du primaire et la fiabilité du secondaire sont stabilisées :

$$P^*(n) = p_P p_S \qquad (V.24)$$

Soit $R(n, n+n_0)$ la fonction caractérisant l'évolution de la fiabilité du BR entre les instants n et $n+n_0$. En utilisant les relations (V.9) et (V.18) on trouve :

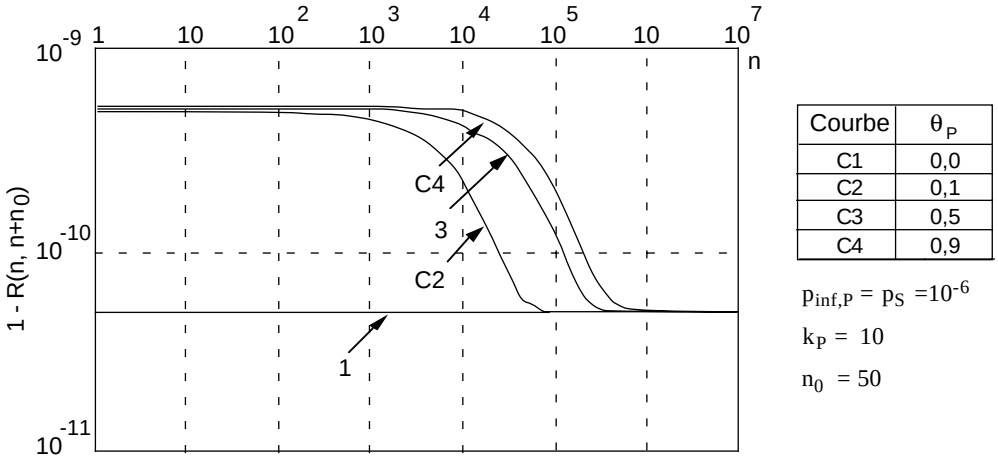
$$R(n,n+n_0) = \frac{\Phi\theta_P(1-p_{sup,P})^{n+n_0} + (1-\Phi\theta_P)(1-p_{inf,P}p_S)^{n+n_0}}{\Phi\theta_P(1-p_{sup,P})^{n} + (1-\Phi\theta_P)(1-p_{inf,P}p_S)^{n}} \qquad (V.25)$$

Soit $R^*(n, n+n_0)$ la fonction caractérisant l'évolution de la fiabilité du BR entre les instants n et $n+n_0$ dans le cas où on suppose que la fiabilité du primaire et la fiabilité du secondaire sont stabilisées. $R^*(n, n+n_0)$ s'obtient directement à partir de (V.25) en posant $\theta_P = 0$ et $p_{inf,P} = p_P$. On trouve :

$$R^*(n,n+n_0) = (1-p_P p_S)^{n_0} \qquad (V.26)$$

Ainsi, dans le cas où on ne tient pas compte du phénomène de croissance de fiabilité, la fiabilité du BR durant une mission de durée fixé n_0 est constante quelque soit l'instant n considéré. Cependant, dans le cas où la fiabilité du primaire croît au cours de son cycle de vie, la probabilité $R(n, n+n_0)$ croît en fonction de n.

La figure V.4 présente différentes courbes d'évolution de la fonction $1- R(n,n+n_0)$: C1 correspond au cas où on ne tient pas compte de la croissance de fiabilité du primaire, et C2, C3 et C4 caractérisent l'évolution de la croissance de fiabilité du BR en considérant différentes allures d'évolution de la probabilité de défaillance à l'exécution du primaire.



Courbe	θ_P
C1	0,0
C2	0,1
C3	0,5
C4	0,9

$p_{inf,P} = p_S = 10^{-6}$

$k_P = 10$

$n_0 = 50$

Figure V.4 : Courbes d'évolution de 1- $R(n,n+n_0)$

REMARQUE

Toutes les mesures évaluées dans ce paragraphe peuvent être obtenues également en considérant la chaîne de Markov réduite déduite de la chaîne présentée sur la figure V.3-b.

La matrice T des probabilités de transition associées à la chaîne réduite s'écrit, à partir des relations (V.10), (V.11) et (V.12), sous la forme :

$$T = \begin{pmatrix} 1-p_{sup,P} & p_{sup,P}(1-p_S) & p_S p_{sup,P} \\ 0 & 1-p_{sup,P}p_S & p_S p_{inf,P} \\ 0 & 0 & 1 \end{pmatrix} \qquad (V.27)$$

La chaîne de Markov ainsi obtenue est présentée sur la figure V.5.

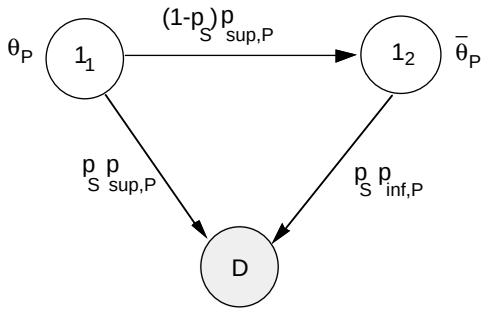


Figure V.5 : Chaîne de Markov réduite

On peut vérifier qu'en traitant directement la chaîne de Markov réduite, on obtient la même expression pour la fonction r(n).

V.1.3 Prise en compte des caractéristiques de l'environnement d'utilisation du logiciel

Considérons la chaîne de Markov en temps continu présentée sur la figure V.6[1] qui est telle que la chaîne immergée associée est la chaîne de Markov caractérisant la croissance de fiabilité du BR en temps discret. La prise en compte des taux d'exécution du primaire et du secondaire permet d'évaluer l'évolution dans le temps des mesures de fiabilité du BR. L'avantage de dissocier les probabilités conditionnelles de défaillance à l'exécution du primaire et du secondaire de la distribution du temps d'exécution de ces composants est d'évaluer le comportement du BR tel qu'il est perçu dans chacun des environnements d'utilisation du logiciel ; γ_P et γ_S pouvant varier d'un environnement à un autre.

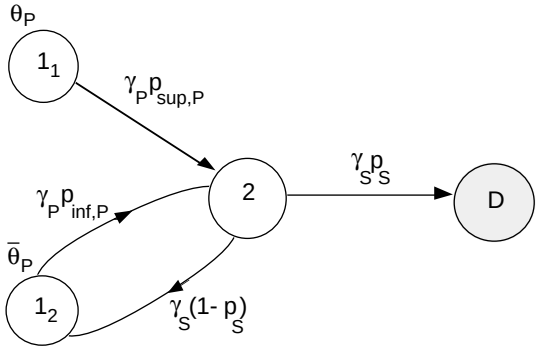


Figure V.6 : Chaîne de Markov transformée (temps continu)

On se propose d'évaluer la fonction R(t,t+u) caractérisant l'évolution dans le temps de la fiabilité du BR pour une durée de mission u.

En considérant la même approche que celle que nous avons présentée dans le chapitre IV (paragraphe IV.3.2.3), on peut obtenir une chaîne de Markov réduite à partir de la chaîne de Markov de la figure V.6 et déduire l'expression analytique de R(t,t+u) à partir de celle-ci.

---

1 La chaîne de Markov présentée sur cette figure correspond à celle qui est donnée par la figure V.2-d.

Etant donné que : $\gamma_S(1- p_S) >> \gamma_S p_S$, l'état 2 de la chaîne de Markov considérée est un état transitoire rapide et les états 1_1 et 1_2 et D sont des états lents. L'application de la technique d'agrégation conduit alors à la chaîne de Markov réduite présentée sur la figure V.7.

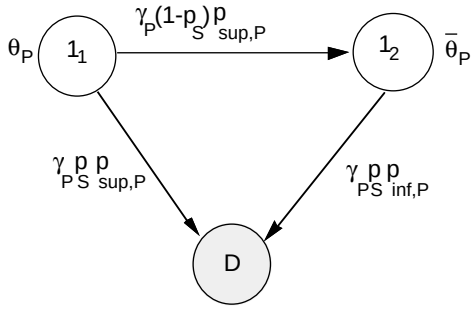


Figure V.7 : Chaîne de Markov réduite (temps continu)

Le traitement de la chaîne réduite conduit à l'expression suivante de R(t,t+u) :

$$R(t,t+u) = \frac{\Phi\theta_P \exp(-\gamma_P p_{sup,P}(t+u)) + (1-\Phi\theta_P) \exp(-\gamma_P p_{inf,P} p_S(t+u))}{\Phi\theta_P \exp(-\gamma_P p_{sup,P} t) + (1-\Phi\theta_P) \exp(-\gamma_P p_{inf,P} p_S t)} \qquad (V.28)$$

où Φ est donné par la relation (V.17).

On remarque que γ_S n'intervient pas dans l'expression de R(t,t+u). La raison en est que l'algorithme d'agrégation utilisé pour obtenir la chaîne de Markov réduite est basé sur l'hypothèse que les temps de séjour dans les états transitoires rapides sont négligeables devant les temps de séjour dans les autres états de la chaîne de Markov considérée. Les états transitoires rapides n'interviennent dans la chaîne de Markov caractérisant le comportement du système considéré que par les probabilités conditionnelles de transition vers les autres états de la chaîne. Pour cette raison, seules les probabilités p_S et $1-p_S$ interviennent dans l'expression de R(t,t+u).

L'approximation effectuée est satisfaisante. En effet, si on calcule directement R(t,t+u) à partir de la chaîne de Markov de la figure V.6, on obtient :

$$R(t,t+u) \approx \frac{\Phi\theta_P \exp(-\gamma_P p_{sup,P}(t+u)) + (1-\Phi\theta_P) \exp(-\gamma_P p_{inf,P} p_S(t+u)) + \Psi \exp(-\gamma_S(t+u))}{\Phi\theta_P \exp(-\gamma_P p_{sup,P} t) + (1-\Phi\theta_P) \exp(-\gamma_P p_{inf,P} p_S t) + \Psi \exp(-\gamma_S t)} + o(1/\gamma_S) \qquad (V.29)$$

$$\text{avec } \Psi \approx \frac{\theta_P (2-p_S)\gamma_P p_{sup,P}}{\gamma_S} + \frac{\overline{\theta}_P \gamma_P p_{inf,P}}{\gamma_S} + o(1/\gamma_S)$$

Etant donné que $\gamma_S >> (\gamma_P p_{sup,P}, \gamma_P p_{inf,P} p_S)$, on peut négliger les termes $\Psi\exp(-\gamma_S t)$ dans l'expression de R(t,t+u) et on retrouve ainsi la relation (V.28).

Ainsi, la variation de la fiabilité R(t,t+u) du BR dans différents environnements d'utilisation du logiciel en fonction des caractéristiques de ces environnements, est conditionnée uniquement par la valeur de γ_P.

Sur la figure V.8, nous présentons des exemples d'évolution de la fonction 1- R(t,t+u) pour différentes valeurs de γ_P en considérant les mêmes paramètres du modèle hyperexponentiel en

temps discret qui caractérisent la croissance de fiabilité du primaire en fonction du nombre d'exécutions effectuées. Les courbes C1, C2 et C3 donnent l'évolution de la fiabilité du BR pour une durée de mission fixe u =50h telle qu'elle est perçue par ses utilisateurs dans trois environnements qui sont caractérisés par des valeurs γ_P différentes. On peut remarquer que plus la variation de γ_P est importante plus l'écart entre les estimations de la fiabilité dans les différents environnements considérés est important.

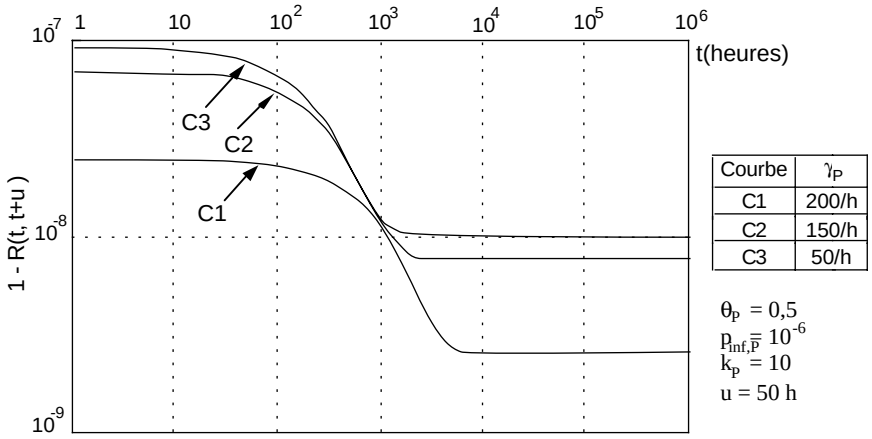


Figure V.8 : Influence de γ_P sur l'évolution de 1- R(t,t+u)

V.1.4 Conclusion

Dans ce paragraphe, nous avons introduit une approche basée sur le modèle hyperexponentiel en temps discret qui permet d'une part, de modéliser la croissance de fiabilité d'un logiciel à partir de celle de ses différents composants en fonction du nombre d'exécutions effectuées, et d'autre part, de déduire les mesures de fiabilité qui caractérisent le comportement du système considéré tel qu'il est perçu par ses utilisateurs en tenant compte des caractéristiques de l'environnement dans lequel il est utilisé.

L'approche que nous avons proposée dans ce paragraphe s'applique à des logiciels multi-composant qui sont tels que chaque instant d'utilisation du logiciel correspond à l'exécution d'un seul composant. Cette approche est similaire à celle que nous avons définie dans le chapitre IV, elle se résume en trois étapes :

- représenter le comportement du système en fiabilité stabilisée par une chaîne de Markov en temps continu telle que, pour chaque état de la chaîne de Markov considérée on distingue le taux d'exécution du composant en cours d'exécution de la probabilité d'activation des fautes conduisant à la défaillance de ce composant,

- transformer la chaîne de Markov en fiabilité stabilisée en considérant la représentation markovienne du modèle hyperexponentiel en temps discret, pour prendre en compte le phénomène de croissance de fiabilité,

- évaluer les mesures de fiabilité en temps discret et en temps continu à partir de la chaîne transformée ; les mesures en temps discret étant obtenues en considérant la chaîne immergée déduite de la chaîne de Markov transformée. Les mesures en temps continu permettent d'évaluer le comportement du logiciel en tenant compte des taux d'exécution de ses différents composants.

Dans la suite de ce chapitre, nous généralisons cette approche en utilisant les réseaux de Petri stochastiques généralisés, puis nous appliquons l'approche proposée à un bloc de recouvrement constitué d'un primaire, d'un secondaire et d'un test d'acceptation.

V.2 GéNéRALISATION DE L'APPROCHE DE MODéLISATION

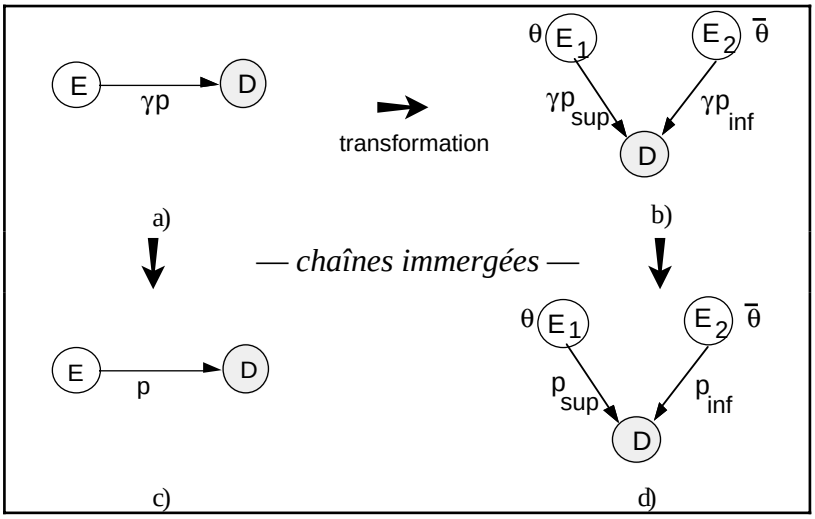
Afin de faciliter la mise en œuvre de l'approche proposée pour la modélisation de la croissance de fiabilité d'un système multi-composant, il est nécessaire d'utiliser, de façon analogue à l'approche de modélisation en temps continu, un outil permettant de construire une chaîne de Markov caractérisant le comportement en fiabilité stabilisée du système considéré, et d'effectuer ensuite la transformation de façon transparente à l'utilisateur pour obtenir la chaîne de Markov caractérisant la croissance de fiabilité du système.

Pour les mêmes raisons que celles évoquées dans le chapitre IV, notre choix s'est fixé sur les réseaux de Petri stochastiques généralisés.

V.2.1 Mise en œuvre de l'approche par les RdPSG

La première étape de l'approche consiste à définir un RdPSG permettant de modéliser la croissance de fiabilité d'un composant en tenant compte de la représentation markovienne du modèle hyperexponentiel en temps discret.

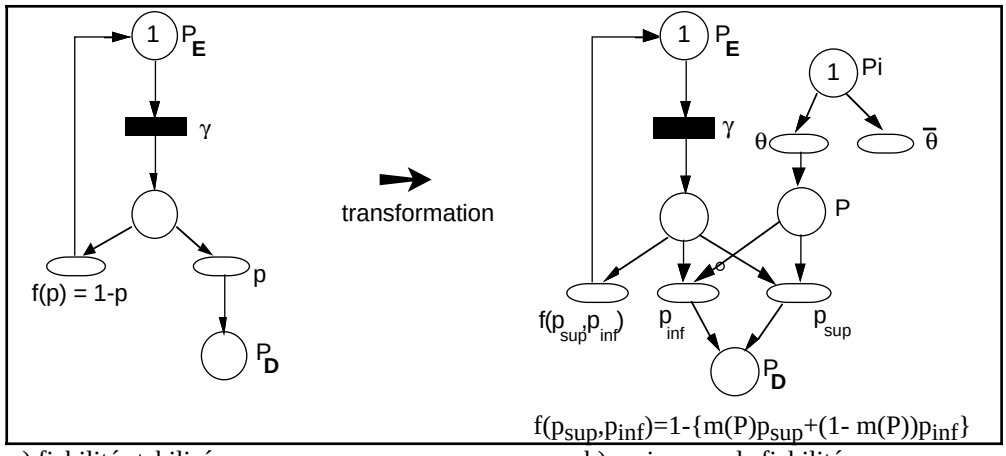
L'application de l'approche de transformation à un composant est résumée sur la figure V.9. Les figures V.9-a et V.9-b correspondent respectivement aux chaînes de Markov en temps continu caractérisant le comportement du composant considéré en fiabilité stabilisée et en croissance de fiabilité, et les figures V.9-c et V.9-d correspondent aux chaînes de Markov immergées déduites à partir des chaînes de Markov en temps continu permettant de caractériser son comportement en fonction du nombre d'exécutions effectuées.



a) fiabilité stabilisée b) croissance de fiabilité

Figure V.9 : Chaînes de Markov en temps continu et chaînes immergées associées d'un composant en fiabilité stabilisée et en croissance de fiabilité

Un modèle RdPSG illustrant la transformation dans le cas d'un composant est présenté sur la figure V.10. Les chaînes de Markov générées à partir des RdPSG présentés sur les figures V.10-a et V.10-b sont celles qui sont données respectivement sur les figures V.9-a et V.9-b.



a) fiabilité stabilisée b) croissance de fiabilité

Figure V.10 : RdPSG d'un composant en fiabilité stabilisée et en croissance de fiabilité

La modélisation du phénomène de croissance de fiabilité se traduit de façon formelle par la substitution des transitions immédiates de probabilités associées p et f(p)=1-p, par le sous-réseau de Petri constitué des places Pi et P, des transitions immédiates de probabilités de franchissement associées θ et $\overline{\theta}$ permettant de représenter les probabilités initiales d'occupation des états E_1 et E_2, et des transitions immédiates de probabilités de franchissement associées p_{sup}, p_{inf} et $f(p_{sup},p_{inf})$ permettant de modéliser les probabilités conditionnelles de transition relatives à la chaîne de Markov associée.

La valeur de la probabilité $f(p_{sup}, p_{inf})$ dépend du marquage de la place P : si celle-ci est marquée alors $f(p_{sup},p_{inf}) = 1- p_{sup}$, sinon $f(p_{sup}, p_{inf}) = 1- p_{inf}$.

Dans le cas d'un logiciel multi-composant, l'approche de modélisation basée sur la transformation se résume par les étapes suivantes :

- construction du RdPSG modélisant le comportement du système en fiabilité stabilisée conformément au modèle de la figure V.10-a,
- transformation du RdPSG conformément au modèle de la figure V.10-b pour modéliser la croissance de fiabilité du système,
- génération du graphe des marquages du modèle RdPSG transformé pour obtenir la chaîne de Markov en temps continu du système en croissance de fiabilité,
- génération de la chaîne de Markov immergée associée à la chaîne de Markov transformée.

Le traitement de la chaîne de Markov immergée en utilisant la méthode présentée dans le paragraphe V.1.2.1 permet d'obtenir les mesures de fiabilité du logiciel considéré en fonction du nombre d'exécutions effectuées. Le traitement de la chaîne de Markov en temps continu permet d'évaluer les mesures caractérisant le comportement du logiciel en tenant compte des taux d'exécution de ses composants.

Si on applique cette approche à l'exemple traité précédemment concernant un bloc de recouvrement constitué d'un primaire et d'un secondaire, on obtient les RdPSG en fiabilité stabilisée et en croissance de fiabilité présentés respectivement sur les figures V.11-a et V.11-b. Les chaînes de Markov générées à partir de ces RdPSG sont celles qui sont présentées sur les figures V.2-a et V.2-d respectivement.

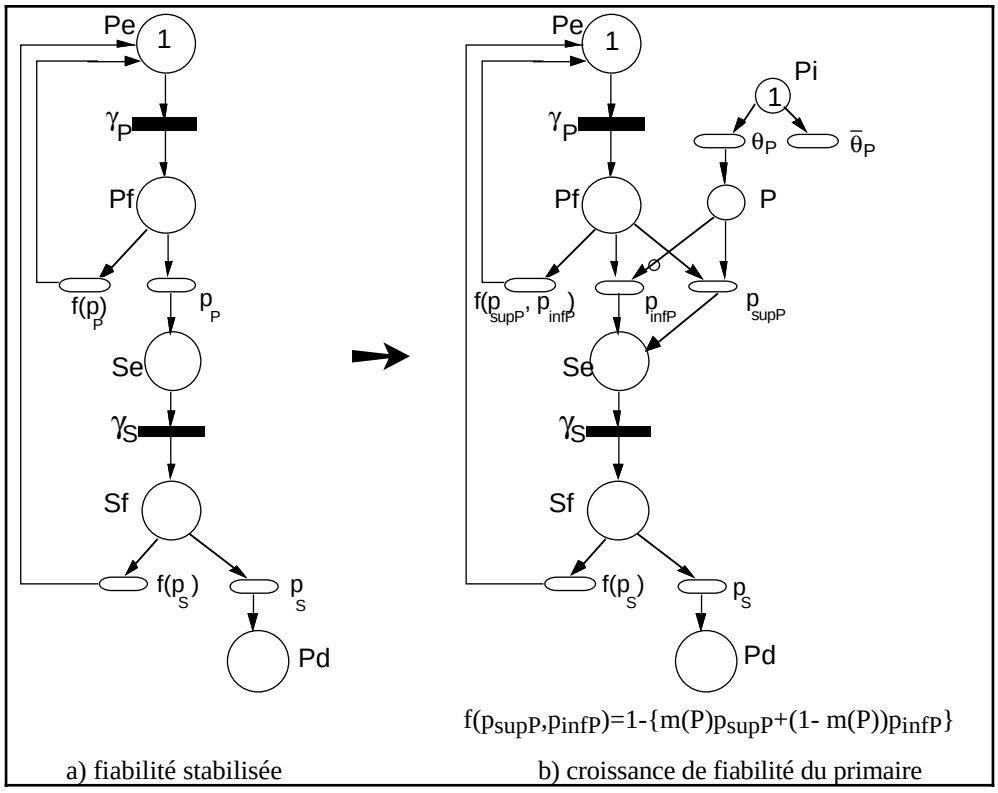


Figure V.11 : Modèles RdPSG d'un BR constitué d'un primaire et d'un secondaire

V.2.2 Application de l'approche générale à un logiciel non tolérant aux fautes

Considérons un système logiciel non tolérant aux fautes (NTF) constitué d'un système d'exploitation (composant n°1) et de N-1 autres composants constituant le logiciel d'application. Lorsque l'exécution d'un composant est terminée, le système d'exploitation est exécuté afin de déterminer le prochain composant d'application à exécuter.

Soient :

. $1/\gamma_j$: temps moyen d'exécution du composant j, j =1, …, N,

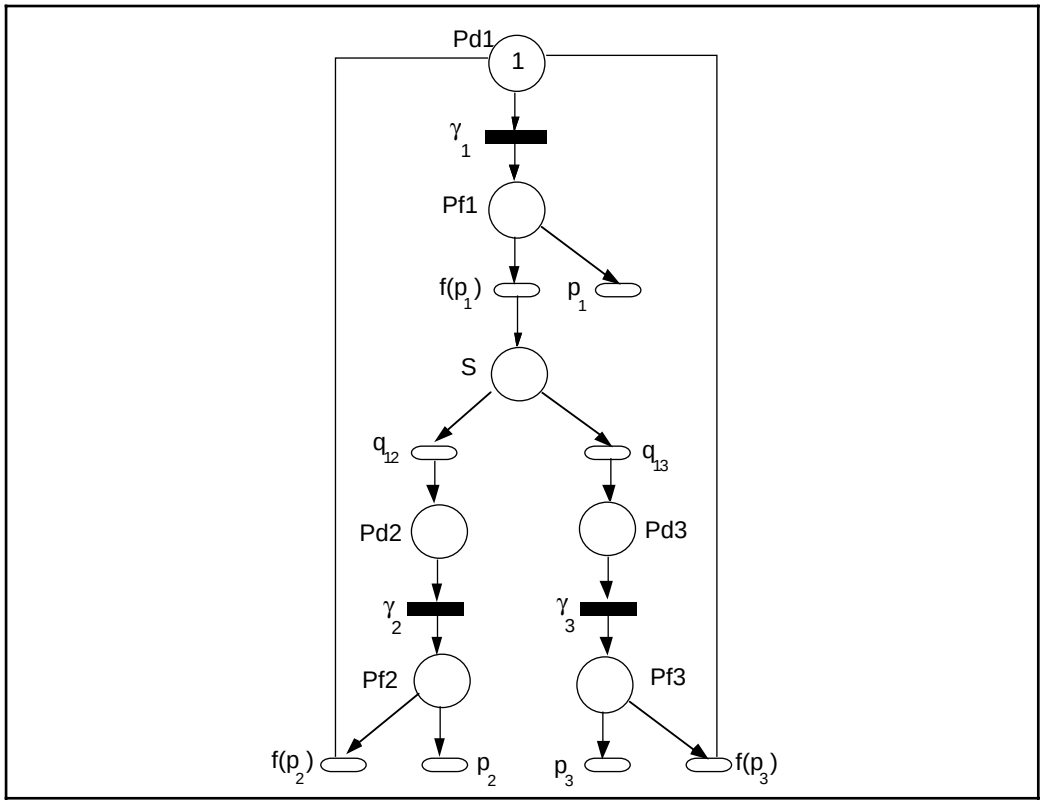
. q_{1j} = Pr { le composant d'application j est sélectionné par le logiciel d'exploitation en l'absence de défaillance }, j = 2, …, N-1,

. p_j = probabilité de défaillance à l'exécution du composant j, j =1, …, N.

On se propose d'évaluer la croissance de fiabilité du système non tolérant aux fautes en fonction de celle de ses différents composants en appliquant l'approche générale présentée dans le paragraphe précédent. Nous considérerons d'abord le cas d'un système constitué de 3 composants puis nous étendrons les résultats au cas N > 3.

V.2.2.1 Modèles RdPSG en fiabilité stabilisée et en croissance de fiabilité

Le modèle RdPSG du système NTF considéré en fiabilité stabilisée est présenté sur la figure V.12 où p_j la probabilité de défaillance à l'exécution du composant j et $f(p_j) = 1- p_j$, j =1,2,3



Pd1	Logiciel d'exploitation (composant 1) en cours d'exécution	γ_1	Exécution du composant 1
Pf1	Exécution composant 1 terminée	p_1	Défaillance du composant 1
		$f(p_1)=1-p_1$	Exécution correcte du composant 1
S	Sélection d'un logiciel d'application pour l'exécution suivante	q_{12}	Sélection du composant 2
		q_{13}	Sélection du composant 3
Pd2	Logiciel d'application (composant 2) en cours d'exécution	γ_2	Exécution du composant 2
Pf2	Exécution du composant 2 terminée	p_2	Défaillance du composant 2
		$f(p_2)=1-p_2$	Exécution correcte du composant 2
Pd3	Logiciel d'application (composant 3) en cours d'exécution	γ_3	Exécution du composant 3
Pf3	Exécution du composant 3 terminée	p_3	Défaillance du composant 3
		$f(p_3)=1-p_3$	Exécution correcte du composant 3

Figure V.12 : RdPSG du système NTF en fiabilité stabilisée

La transformation du RdPSG du système NTF en fiabilité stabilisée selon le modèle de la figure V.10 conduit au RdPSG présenté sur la figure V.13. θ_j, p_{jsup} et p_{jinf} sont les paramètres du modèle hyperexponentiel en temps discret qui décrivent la croissance de fiabilité du composant j. Pour la clarté de la figure, p_{jsup} et p_{jinf} sont notés respectivement p_{js} et p_{ji}, j = 1,2,3.

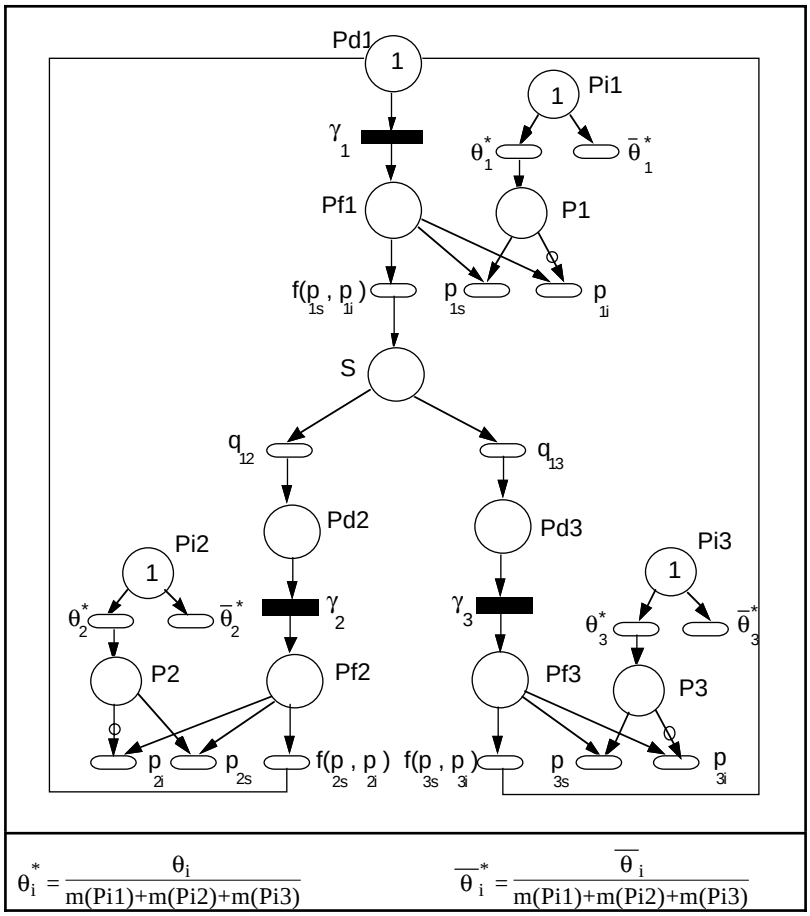
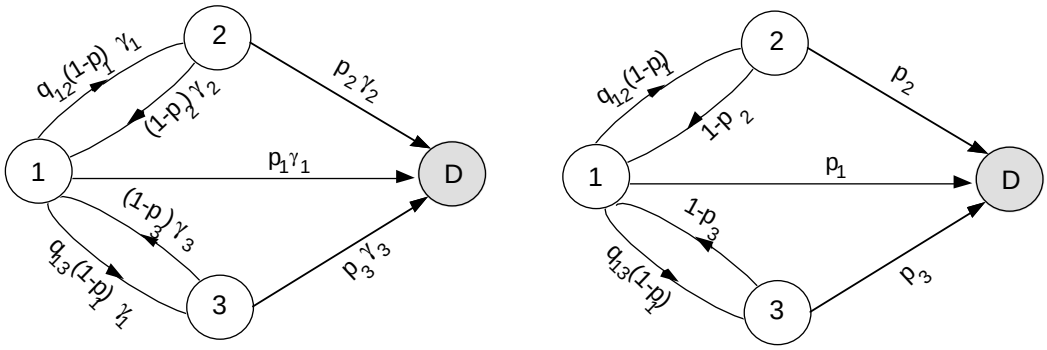


Figure V.13 : RdPSG du système en croissance de fiabilité

V.2.2.2 Chaînes de Markov en fiabilité stabilisée et en croissance de fiabilité

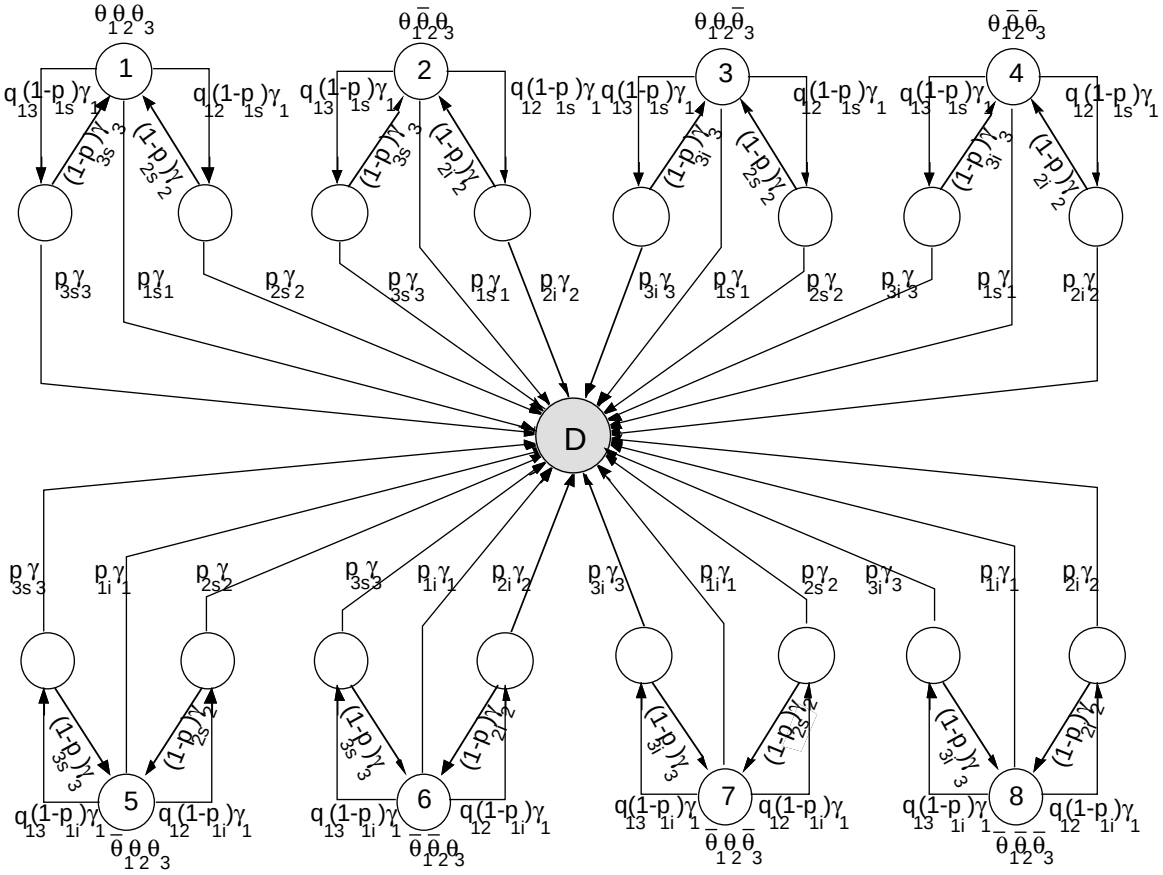
La chaîne de Markov générée à partir du graphe des marquages du RdPSG du système NTF en fiabilité stabilisée est présentée sur la figure V.14-a. La chaîne immergée associée à cette chaîne de Markov est présentée sur la figure V.14-b. Celle-ci décrit le comportement en fiabilité stabilisée du système en temps discret.



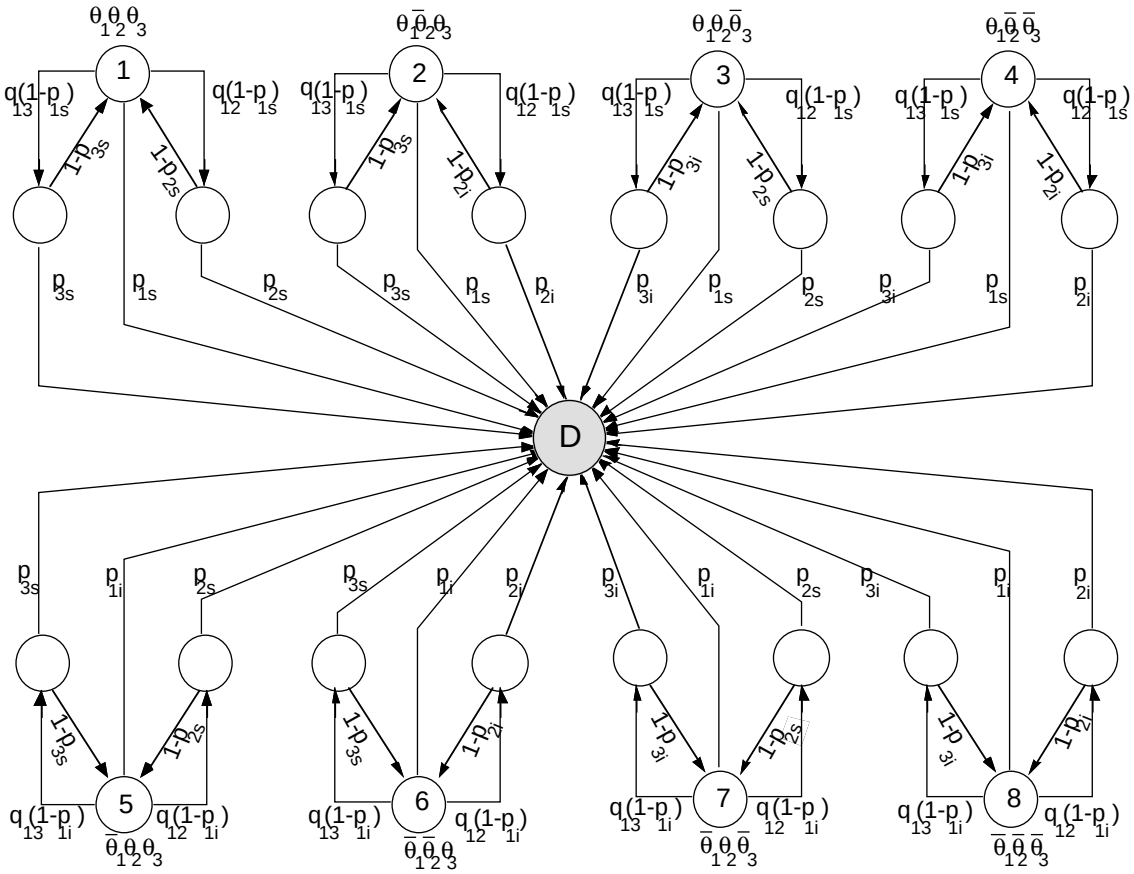
(a) temps continu (b) temps discret

Figure V.14 : Chaînes de Markov du système NTF en fiabilité stabilisée

La chaîne de Markov décrivant le comportement du système en croissance de fiabilité qui est générée à partir du graphe des marquages du RdPSG correspondant et la chaîne immergée associée sont présentées respectivement sur les figures V.15 -a et V.15-b.



(a) temps continu



(b) temps discret

Figure V.15 : Chaînes de Markov du système en croissance de fiabilité

V.2.2.3 Mesures de fiabilité

Nous évaluerons dans un premier temps, les mesures caractérisant l'évolution du comportement du système NTF en fonction du nombre d'exécutions effectuées à partir de la chaîne immergée transformée (figure V.15-b), et dans un deuxième temps, les mesures qui caractérisent l'évolution dans le temps du comportement du système NTF en tenant compte des taux d'exécution des différents composants du logiciel en considérant la chaîne de Markov en temps continu présentée sur la figure V.15-a.

V.2.2.3.1 Mesures en temps discret

Considérons la chaîne de Markov en temps discret présentée sur la figure V.15-b. On se propose d'évaluer la fiabilité $R(n,n+n_0)$ du système en utilisant la méthode présentée dans le paragraphe V.1.2.1.

Les états {1, 2…,8} constituent le sous-ensemble B des états qui caractérisent la sollicitation du système pour une nouvelle exécution. Compte tenu des résultats présentés dans V.1.2.1, on peut définir à partir de la chaîne de Markov considérée une chaîne de Markov réduite constituée du sous-ensemble d'états B et de l'état absorbant. La chaîne de Markov réduite, déduite de la chaîne de Markov transformée en temps discret est présentée sur la figure V.16. La matrice des probabilités de transition associée à la chaîne réduite est calculée à partir de la relation V.10.

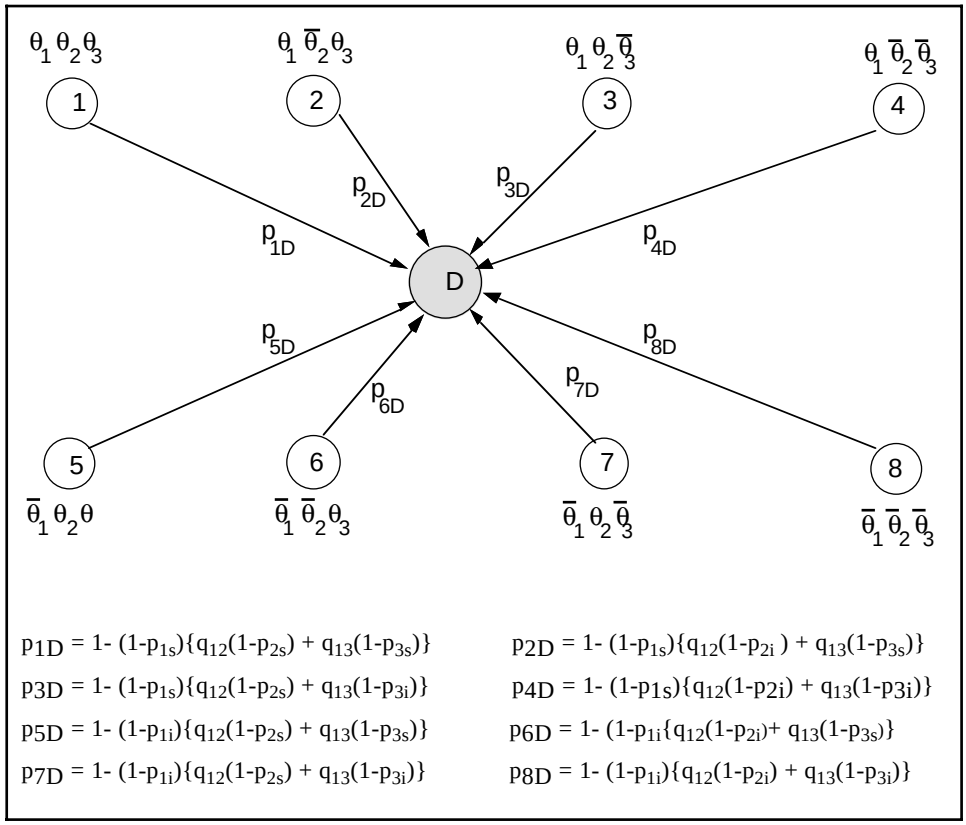


Figure V.16 : Système en croissance de fiabilité : chaîne de Markov réduite

ETABLISSEMENT DE L'EXPRESSION DE LA FIABILITé $R(n,n+n_0)$

Le calcul de la fonction de survie r(n) à partir de la chaîne de Markov réduite donne :

$$r(n) = \sum_{i \in B} \beta_i (1-p_{iD})^n \qquad \text{(V.30)}$$

β_i est la probabilité initiale d'occupation de l'état i de la figure V.16.

Soit P(n) la probabilité de défaillance à l'exécution n du système NTF : $P(n) = 1 - \frac{r(n)}{r(n-1)}$:

$$P(n) = \frac{\sum_{k \in B} \beta_k p_{kD} (1-p_{kD})^{n-1}}{\sum_{k \in B} \beta_k (1-p_{kD})^{n-1}} \qquad \text{(V.31)}$$

On remarque alors que la probabilité P(n) caractérisant l'évolution de la probabilité de défaillance du système en fonction du nombre d'exécutions effectuées a la même forme que l'expression générale de la probabilité de défaillance à l'exécution d'un modèle hyperexponentiel en temps discret.

Evaluons maintenant l'expression de la fonction de fiabilité $R(n,n+n_0) = \frac{r(n+n_0)}{r(n)}$ donnant l'évolution de la fiabilité du système pour un nombre n_0 fixe d'exécutions. On obtient à partir de la relation (V.30) :

$$R(n,n+n_0) = \frac{\sum_{k \in B} \beta_k (1-p_{kD})^{n+n_0}}{\sum_{k \in B} \beta_k (1-p_{kD})^{n_0}} \qquad (V.32)$$

A partir de la relation (V.32), on peut également déduire l'expression de la fonction $R^*(n,n+n_0)$ associée à la chaîne de Markov de la figure V.14-b caractérisant le comportement en fiabilité stabilisée du système. Il suffit de poser $\theta_j = 0$ et $p_j = p_{ij}$ pour tout j = 1, 2, 3.

On trouve :

$$R^*(n,n+n_0) = \left\{(1-p_1)\,[\,q_{12}(1-p_2) + q_{13}(1-p_3)]\right\}^{n_0} \qquad (V.33)$$

ETABLISSEMENT DE L'EXPRESSION DE LA FIABILITé $R(n,n+n_0)$ POUR $N > 3$

La relation (V.32) peut être généralisée dans le cas où le système est composé d'un logiciel d'exploitation et de N-1 composants constituant le logiciel d'application. En effet, en supposant que tous les composants considérés sont différents, l'application de l'approche générale conduit à une chaîne de Markov en croissance de fiabilité constituée de 2^N états initiaux, ces états constituent le sous-ensemble B qui caractérise l'exécution du système global. En appliquant la technique d'agrégation, on obtient une chaîne de Markov réduite constituée de $2^N + 1$ états : les états appartenant à B et un état absorbant. Le calcul de la fonction $R(n,n+n_0)$ à partir de la chaîne réduite donne alors :

$$R(n,n+n_0) = \frac{\sum_{k \in B} \beta_k (1-p_{kD})^{n+n_0}}{\sum_{k \in B} \beta_k (1-p_{kD})^{n_0}} \qquad (V.34)$$

avec :

$$\beta_k \in \Theta = \left\{ \prod_{j=1}^{N} \delta_k(\theta_j, \overline{\theta}_j) \mid \delta_k(\theta_j, \overline{\theta}_j) = \theta_j \text{ ou } \delta_k(\theta_j, \overline{\theta}_j) = \overline{\theta}_j \right\} \qquad (V.35)$$

Θ est l'ensemble des probabilités d'occupation initiale des états de B et β_k est la probabilité d'occupation initiale de l'état k de B.

$p_{kD} = 1 - (1-\nu_1)\sum_{j=2}^{N-1} q_{1j}\,(1-\pi_j)$ où π_j est tel que :

$$\begin{cases} \nu_j = p_{j\text{sup}} & \text{si } \delta_k(\theta_j, \overline{\theta}_j) = \theta_j \\ \nu_j = p_{j\text{inf}} & \text{si } \delta_k(\theta_j, \overline{\theta}_j) = \overline{\theta}_j \end{cases} \qquad (V.36)$$

p_{kD} est la probabilité de transition de l'état k vers l'état absorbant.

L'évaluation de la fonction $R(n,n+n_0)$ donnée par la relation (V.34) et l'étude de sa variation en fonction des paramètres du modèle permet de quantifier et d'analyser la croissance de fiabilité du système NTF considéré en fonction de celle de ses différents composants.

VARIATION DE $R(n,n+n_0)$ EN FONCTION DES PARAMèTRES DU MODèLE (CAS : N=3)

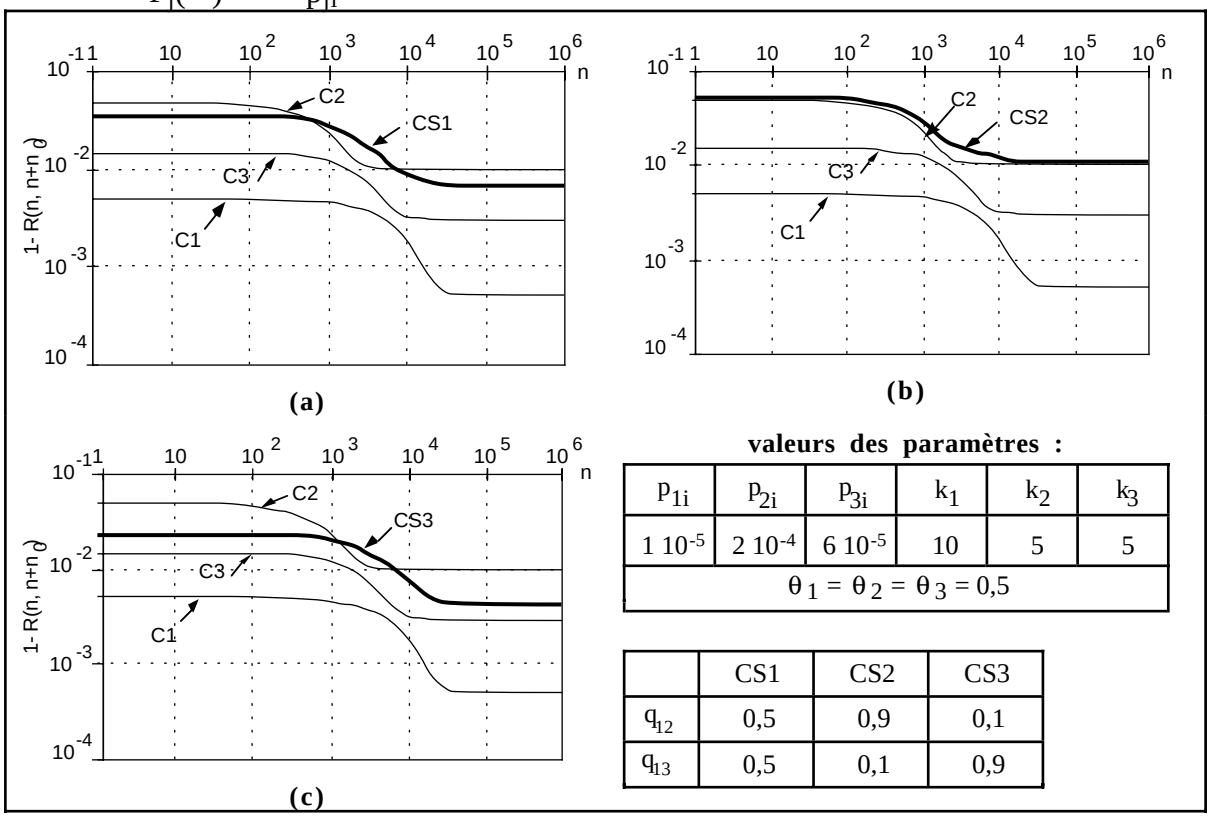
Sur la figure V.17, nous présentons des exemples d'évolution de la fonction $1-R(n,n+n_0)$ pour observer l'évolution de la croissance de fiabilité du système en fonction de celle de ses différents composants :

. Cj : donne l'évolution de la fonction $1- R(n,n+n_0)$ du composant j du système, j = 1,2,3.

. CS1, CS2 et CS3 donnent l'évolution de la fonction $1- R(n,n+n_0)$ du système en tenant compte de la croissance de fiabilité de ses différents composants et en considérant différentes valeurs des paramètres q_{12} et q_{13}.

Les valeurs considérées pour les paramètres du modèle caractérisant la croissance de fiabilité des composants du système et pour les probabilités q_{12} et q_{13} sont données sur la figure V.17. k_j est le facteur de croissance défini par :

$$k_j = \frac{P_j(1)}{P_j(\infty)} = \theta_j \frac{p_{js}}{p_{ji}} + \bar{\theta}_j \qquad j = 1,2,3 \qquad \text{(V.37)}$$



valeurs des paramètres :

p_{1i}	p_{2i}	p_{3i}	k_1	k_2	k_3
$1\ 10^{-5}$	$2\ 10^{-4}$	$6\ 10^{-5}$	10	5	5
$\theta_1 = \theta_2 = \theta_3 = 0,5$					

	CS1	CS2	CS3
q_{12}	0,5	0,9	0,1
q_{13}	0,5	0,1	0,9

Figure 17 : Exemples d'évolution de la fonction 1- R(n, n+n0) (n_0 = 50)

L'analyse de la figure V.17 permet de remarquer que les valeurs q_{12} et q_{13} conditionnent l'évolution de la croissance de fiabilité du système en fonction de celle de ses différents composants. Pour les valeurs choisies pour les paramètres du modèle, on remarque que la fiabilité du système est déterminée en grande partie par la fiabilité des composants C1 et C2.

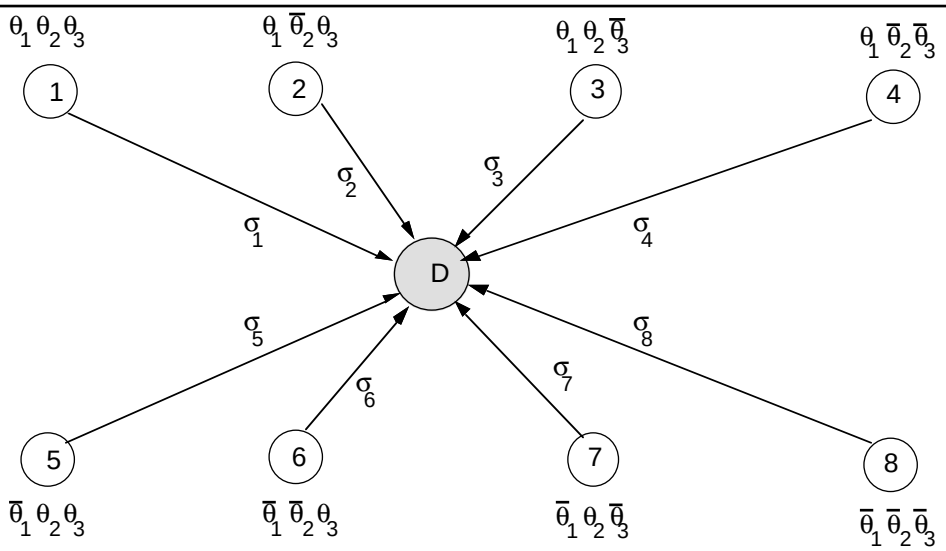
Selon les valeurs de q_{12} et q_{13}, la fiabilité du système peut être essentiellement conditionnée par celle du composant C2 (figure V.17-b), peut dépendre de celle de C2 et de C3 (figure V.17-a), ou bien être essentiellement conditionnée par celle du composant C3 (figure V.17-c). En particulier, sur la figure V 17-c, il est intéressant de remarquer que la fiabilité du système est meilleure que celle du composant C2, qui est le composant le moins fiable, ceci est du au fait que celui-ci est très rarement sollicité au cours de l'exécution du système (q12 = 0,1). Quand les composants du système sont sollicités de façon équiprobable, on observe une alternance de périodes qui sont telles que l'impact des composants C1 et C2 sur la fiabilité du système varie dans le temps (figure V.17-a).

V.2.2.3.2 Mesures en temps continu

Considérons la chaîne de Markov transformée de la figure V.16-a qui caractérise l'évolution de la croissance de fiabilité du système NTF en tenant compte des taux d'exécution de ses composants : γ_1, γ_2 et γ_3. Ces derniers paramètres, en plus des paramètres q_{12} et q_{13}, caractérisent l'environnement dans lequel le logiciel est utilisé.

On se propose d'évaluer la fonction R(t,t+u) caractérisant l'évolution dans le temps du comportement du système NTF et d'étudier la variation de R(t,t+u) en fonction de γ_1, γ_2 et γ_3.

L'application de la technique d'agrégation présentée dans le paragraphe IV.3.2.3 conduit à la chaîne de Markov réduite présentée sur la figure V.18.



$\sigma_1 = \pi_1 p_{1s}\gamma_1 + \pi_2 p_{2s}\gamma_2 + \pi_3 p_{3s}\gamma_3$ $\quad$ $\sigma_2 = \pi_4 p_{1s}\gamma_1 + \pi_5 p_{2i}\gamma_2 + \pi_6 p_{3s}\gamma_3$

$\sigma_3 = \pi_7 p_{1s}\gamma_1 + \pi_8 p_{2s}\gamma_2 + \pi_9 p_{3i}\gamma_3$ $\quad$ $\sigma_4 = \pi_{10} p_{1s}\gamma_1 + \pi_{11} p_{2i}\gamma_2 + \pi_{12} p_{3i}\gamma_3$

$\sigma_5 = \pi_{13} p_{1i}\gamma_1 + \pi_{14} p_{2s}\gamma_2 + \pi_{15} p_{3s}\gamma_3$ $\quad$ $\sigma_6 = \pi_{16} p_{1i}\gamma_1 + \pi_{17} p_{2i}\gamma_2 + \pi_{18} p_{3s}\gamma_3$

$\sigma_7 = \pi_{19} p_{1i}\gamma_1 + \pi_{20} p_{2s}\gamma_2 + \pi_{21} p_{3i}\gamma_3$ $\quad$ $\sigma_8 = \pi_{22} p_{1i}\gamma_1 + \pi_{23} p_{2i}\gamma_2 + \pi_{24} p_{3i}\gamma_3$

$$\pi_1 = \frac{1/\gamma_1(1-p_{1s})}{1/\gamma_1(1-p_{1s}) + q_{12}/\gamma_2(1-p_{2s}) + q_{13}/\gamma_3(1-p_{3s})} \qquad \pi_2 = \frac{q_{12}/\gamma_2(1-p_{2s})}{1/\gamma_1(1-p_{1s}) + q_{12}/\gamma_2(1-p_{2s}) + q_{13}/\gamma_3(1-p_{3s})}$$

$$\pi_3 = \frac{q_{13}/\gamma_3(1-p_{3s})}{1/\gamma_1(1-p_{1s}) + q_{12}/\gamma_2(1-p_{2s}) + q_{13}/\gamma_3(1-p_{3s})} \qquad \pi_4 = \frac{1/\gamma_1(1-p_{1s})}{1/\gamma_1(1-p_{1s}) + q_{12}/\gamma_2(1-p_{2i}) + q_{13}/\gamma_3(1-p_{3s})}$$

$$\pi_5 = \frac{q_{12}/\gamma_2(1-p_{2i})}{/\gamma_1(1-p_{1s}) + q_{12}/\gamma_2(1-p_{2i}) + q_{13}/\gamma_3(1-p_{3s})} \qquad \pi_6 = \frac{q_{13}/\gamma_3(1-p_{3s})}{1/\gamma_1(1-p_{1s}) + q_{12}/\gamma_2(1-p_{2i}) + q_{13}/\gamma_3(1-p_{3s})}$$

$$\pi_7 = \frac{1/\gamma 1(1-p_{1s})}{1/\gamma 1(1-p_{1s}) + q_{12}/\gamma_2(1-p_{2s}) + q_{13}/\gamma_3(1-p_{3i})} \qquad \pi_8 = \frac{q_{12}/\gamma_2(1-p_{2s})}{1/\gamma_1(1-p_{1s}) + q_{12}/\gamma_2(1-p_{2s}) + q_{13}/\gamma_3(1-p_{3i})}$$

$$\pi_9 = \frac{q_{13}/\gamma_3(1-p_{3i})}{11/\gamma_1(1-p_{1s}) + q_{12}/\gamma_2(1-p_{2s}) + q_{13}/\gamma_3(1-p_{3i})} \qquad \pi_{10} = \frac{1/\gamma_1(1-p_{1s})}{1/\gamma_1(1-p_{1s}) + q_{12}/\gamma_2(1-p_{2i}) + q_{13}/\gamma_3(1-p_{3i})}$$

$$\pi_{11} = \frac{q_{12}/\gamma_2(1-p_{2i})}{1/\gamma_1(1-p_{1s}) + q_{12}/\gamma_2(1-p_{2i}) + q_{13}/\gamma_3(1-p_{3i})} \qquad \pi_{12} = \frac{q_{13}/\gamma_3(1-p_{3i})}{1/\gamma_1(1-p_{1s}) + q_{12}/\gamma_2(1-p_{2i}) + q_{13}/\gamma_3(1-p_{3i})}$$

$\pi_{13} = \backslash F(1/\gamma_1(1-p_{1i}); 1/\gamma 1(1-p_{1i}) + q_{12}/\gamma_2(1-p_{2s}) + q_{13}/\gamma_3(1-p_{3s}))$ $\quad$ $\pi_{14} = \backslash F(q_{12}/\gamma_2(1-p_{2s}); 1/\gamma 1(1-p_{1i}) + q_{12}/\gamma_2(1-p_{2s}) + q_{13}/\gamma_3(1-p_{3s}))$

$$\pi_{15} = \frac{q_{13}/\gamma_3(1-p_{3s})}{1/\gamma 1(1-p_{1i}) + q_{12}/\gamma_2(1-p_{2s}) + q_{13}/\gamma_3(1-p_{3s})} \qquad \pi_{16} = \frac{1/\gamma_1(1-p_{1i})}{1/\gamma_1(1-p_{1i}) + q_{12}/\gamma_2(1-p_{2i}) + q_{13}/\gamma_3(1-p_{33})}$$

$$\pi_{17} = \frac{q_{12}/\gamma_2(1-p_{2i})}{/\gamma 1(1-p_{1i}) + q_{12}/\gamma_2(1-p_{2i}) + q_{13}/\gamma_3(1-p_{3s})} \qquad \pi_{18} = \frac{q_{13}/\gamma_3(1-p_{3s})}{1/\gamma 1(1-p_{1i}) + q_{12}/\gamma_2(1-p_{2i}) + q_{13}/\gamma_3(1-p_{3s})}$$

$$\pi_{19} = \frac{1/\gamma_1(1-p_{1i})}{1/\gamma_1(1-p_{1i}) + q_{12}/\gamma_2(1-p_{2s}) + q_{13}/\gamma_3(1-p_{3i})} \qquad \pi_{20} = \frac{q_{12}/\gamma_2(1-p_{2s})}{1/\gamma_1(1-p_{1i}) + q_{12}/\gamma_2(1-p_{2s}) + q_{13}/\gamma_3(1-p_{3i})}$$

$$\pi_{21} = \frac{q_{13}/\gamma_3(1-p_{3i})}{1/\gamma_1(1-p_{1i}) + q_{12}/\gamma_2(1-p_{2s}) + q_{13}/\gamma_3(1-p_{3i})} \qquad \pi_{22} = \frac{1/\gamma_1(1-p_{1i})}{1/\gamma_s(1-p_{1i}) + q_{12}/\gamma_2(1-p_{2i}) + q_{13}/\gamma 3(1-p_{3i})}$$

$$\pi_{23} = \frac{q_{12}/\gamma_2(1-p_{2i})}{1/\gamma_s(1-p_{1i}) + q_{12}/\gamma_2(1-p_{2i}) + q_{13}/\gamma 3(1-p3i)} \qquad \pi_{24} = \frac{q_{13}/\gamma_3(1-p_{3i})}{1/\gamma_s(1-p_{1i}) + q_{12}/\gamma_2(1-p_{2i}) + q_{13}/\gamma 3(1-p_{3i})}$$

Figure V.18 : Système NTF en croissance de fiabilité : chaîne de Markov réduite

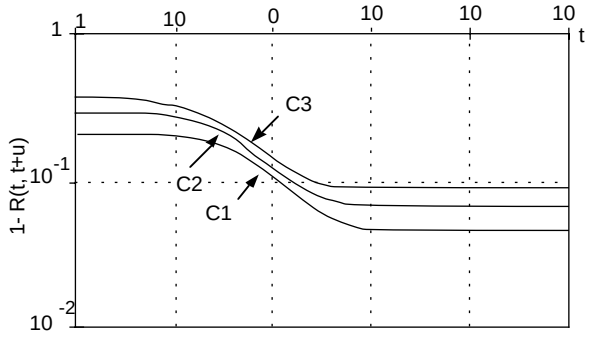
A partir de la chaîne réduite, on peut obtenir facilement l'expression de la fonction r(t), et par suite la fonction R(t,t+u) :

$$R(t,t+u) = \frac{\sum_{i=1}^{8} \beta_i \exp(-\sigma_i(t+u))}{\sum_{i=1}^{8} \beta_i \exp(-\sigma_i t)} \qquad (V.38)$$

où β_i est la probabilité d'occupation initiale de l'état i et σ_i les taux de transition associées aux états de la chaîne de Markov réduite. β_i et σ_i sont données sur la figure V.18.

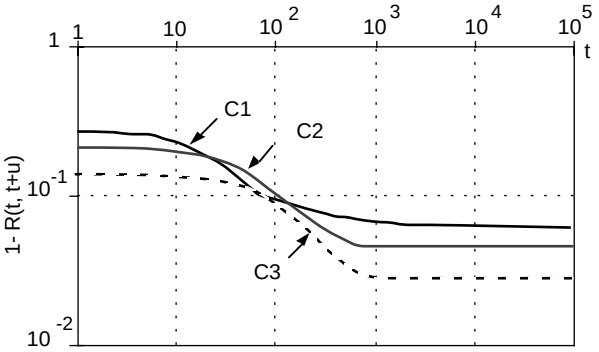
Sur la figure V.19, nous donnons des exemples d'évolution de la fonction 1- R(t,t+u) pour des valeurs γ_1, γ_2 et γ_3 données afin d'observer l'évolution, dans un environnement donné, de la croissance de fiabilité du système en fonction de celle de ses différents composants. On a considéré pour cette figure les mêmes valeurs q_{12} et q_{13} pour les courbes C1, C2 et C3, l'influence de ces deux derniers paramètres sur l'évolution de la fiabilité du système est illustrée sur la figure V.20. Les valeurs des paramètres qui caractérisent la croissance de fiabilité de chacun des composants du système sont les mêmes que celles que nous avons considérées pour évaluer la fiabilité du système NTF en temps discret.

Les courbes présentées sur la figure V.19 et V.20 montrent différentes allures d'évolution de la croissance de fiabilité du système en considérant des environnement d'utilisation du logiciel qui différent soit par les taux d'exécution des composants du système (figure V.19) soit par les valeurs des probabilités q_{12} et q_{13} (figure 20). Les différences entre les courbes présentées traduisent différentes perceptions du comportement du système NTF par ses utilisateurs. L'avantage de l'approche réside donc dans son aptitude à tenir compte de la variation de certains paramètres qui caractérisent l'environnement d'utilisation du système dans l'évaluation de sa fiabilité en tenant compte du phénomène de croissance de fiabilité.



Courbe	γ_1	$\gamma_2 = \gamma_3$
C1	100/h	50/h
C2	150/h	75/h
C3	200/h	100/h
$q_{12} = q_{13} = 0,5$		

Figure V.19 : Variation de 1-R(t,t+u) en fonction de γ_1, γ_2 et γ_3



Courbe	q_{12}	q_{13}
C1	0,9	0,1
C2	0,5	0,5
C3	0,1	0,9
$\gamma_1 = 100/h ; \gamma_2 = \gamma_3 = 50/h$		

Figure V.20 : Variation de 1-R(t,t+u) en fonction de q_{12} et q_{13}

REMARQUE

On peut généraliser l'expression de la fiabilité R(t,t+u) donnée par la relation (V.38) pour un logiciel constitué de N composants, N>3. En procédant de la même manière que dans le cas N=3, on obtient :

$$R(t,t+u) = \frac{\sum_{k=1}^{2^N} \beta_k \exp(-\sigma_k(t+u))}{\sum_{k=1}^{2^N} \beta_k \exp(-\sigma_k t)} \qquad (V.39)$$

où β_k est donné par la relation (V.35) et σ_k est donné par :

$$\sigma_k = \sum_{j=1}^{N} \frac{q_{1j} \big/ [(1-\nu_j(p_{jsup};p_{jinf}))\gamma_j]}{\sum_{m=1}^{N} q_{1m} \big/ [(1-\nu_m(p_{msup};p_{minf}))\gamma_m]} \nu_j(p_{jsup};p_{jinf})\gamma_j, \qquad q_{11} = 1. \qquad (V.40)$$

Ainsi, en évaluant l'expression V.39, on peut quantifier et analyser la croissance de fiabilité d'un système non tolérant aux fautes composé d'un logiciel d'exploitation et de N-1 composants constituant le logiciel d'application, en tenant compte des caractéristiques de l'environnement dans lequel il est utilisé.

V.2.3 Application de l'approche générale à un logiciel tolérant aux fautes

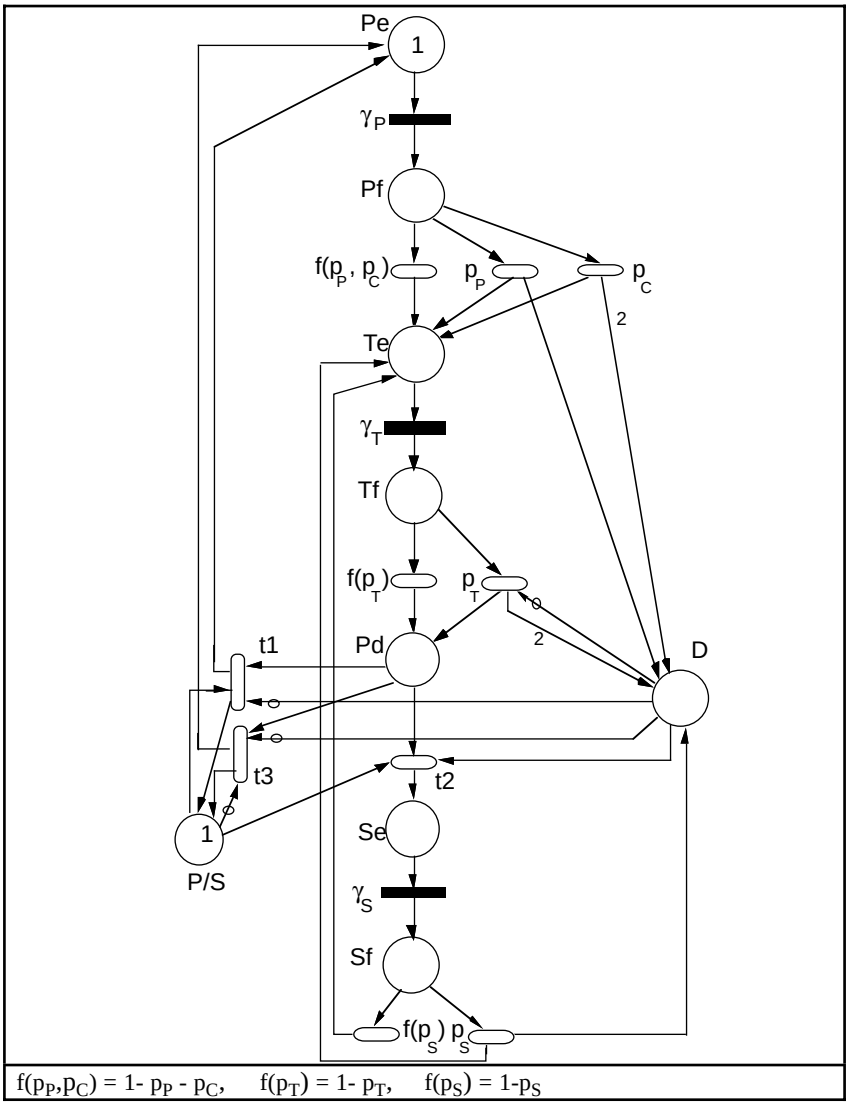
Considérons un bloc de recouvrement constitué de deux alternants : un primaire et un secondaire, et d'un test d'acceptation. On se propose d'appliquer l'approche de transformation pour étudier la croissance de fiabilité du logiciel en prenant compte du phénomène de croissance de fiabilité résultant de l'élimination progressive d'une part, des fautes indépendantes introduites dans le primaire, le secondaire et le test d'acceptation et d'autre part, des fautes corrélées activées dans le logiciel.

V.2.3.1 Modèles RdPSG en fiabilité stabilisée et en croissance de fiabilité

En considérant les mêmes hypothèses de modélisation que celles que nous avons établies dans le paragraphe IV.3.2, on peut modéliser le comportement du BR en fiabilité stabilisée par le RdPSG présenté sur la figure V.21. On donne également sur cette figure la signification des places et des transitions du RdPSG.

Les paramètres du modèle sont les suivants :

- γ_X : le taux d'exécution de X, $X \in \{P, S, T\}$,
- p_X : la probabilité d'activation d'une faute indépendante suite à l'exécution de X, $X \in \{P, S, T\}$,
- p_C : la probabilité d'activation d'une faute corrélée suite à l'exécution du BR.

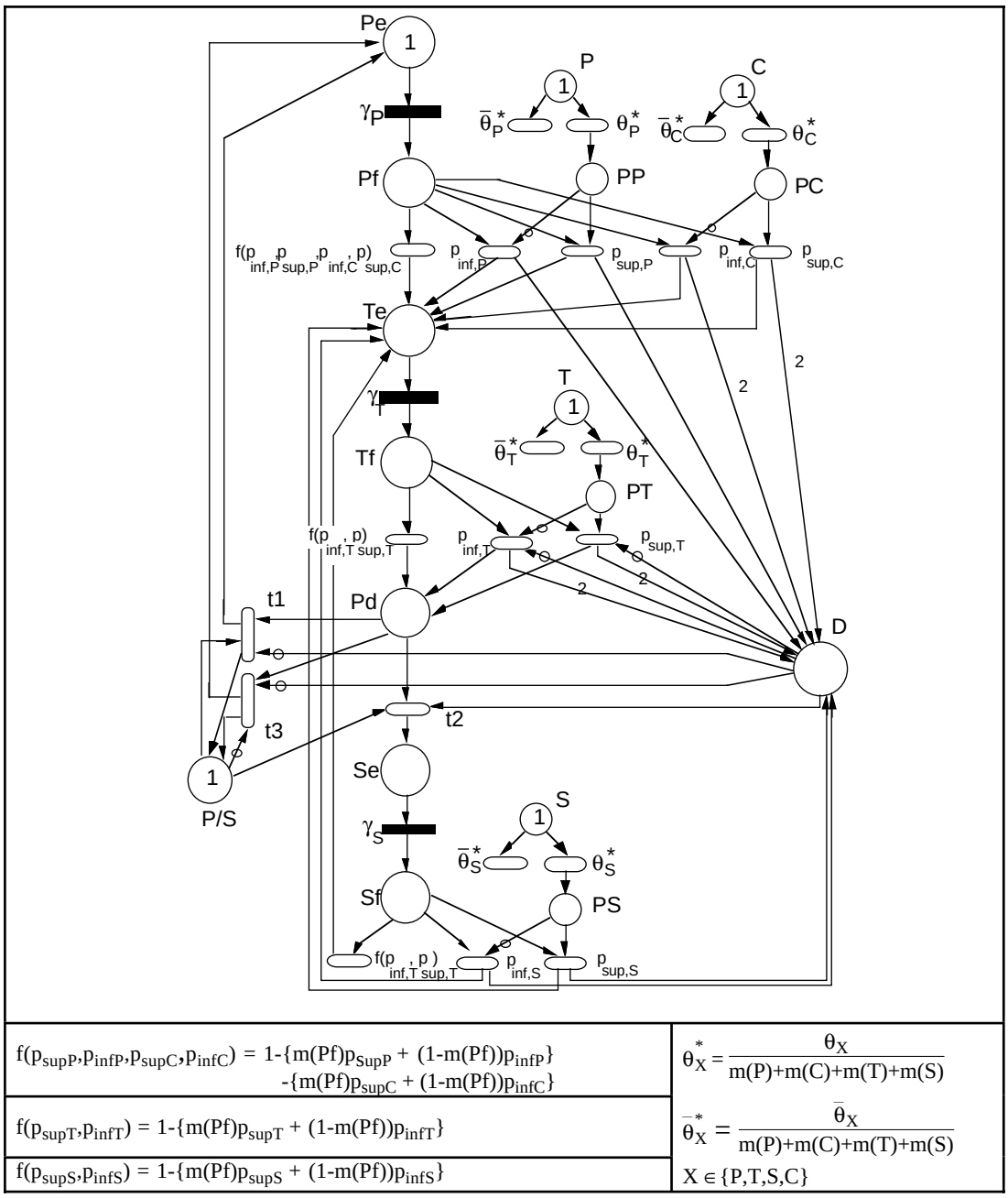


Pe	P en cours d'exécution	γ_P	Exécution de P
Pf	Exécution de P terminée	p_P	Activation d'une faute indépendante dans P
		p_C	Défaillance de P due à une faute corrélée
		$f(p_P,p_C)$	Exécution correcte de P
Te	Début d'exécution de T	γ_T	Exécution de T
Tf	Fin d'exécution de T	p_T	Activation d'une faute indépendante dans T
		$f(p_T)$	Exécution correcte de T
Se	Début d'exécution de S	γ_P	Exécution de S
Sf	Fin d'exécution de S	p_S	Activation d'une faute indépendante dans S
		$f(p_T)$	Exécution correcte de S
Pd	Choix de l'exécution suivante	t1,t3	Fin d'exécution du BR
		t2	Sélection de S
P/S	Sélection de P ou de S		
D	Type de faute activée : m(D)=1 : faute indépendante dans P ou S. m(D) = 2 : faute indépendante dans T ou faute corrélée.		

Figure V.21 : Modèle RdPSG d'un BR en fiabilité stabilisée

La transformation du RdPSG du BR en fiabilité stabilisée selon le modèle de la figure V.10 conduit au RdPSG présenté sur la figure V.22. θ_X, $p_{sup,X}$ et $p_{inf,X}$ sont les paramètres du modèle hyperexponentiel en temps discret qui décrivent la croissance de fiabilité de X par rapport à l'activation de fautes indépendantes, $X \in \{P, S, T\}$ et θ_C, $p_{sup,C}$ et $p_{inf,C}$ sont les

paramètres du modèle hyperexponentiel en temps discret qui décrivent la croissance de fiabilité du BR par rapport à l'activation de fautes corrélées.

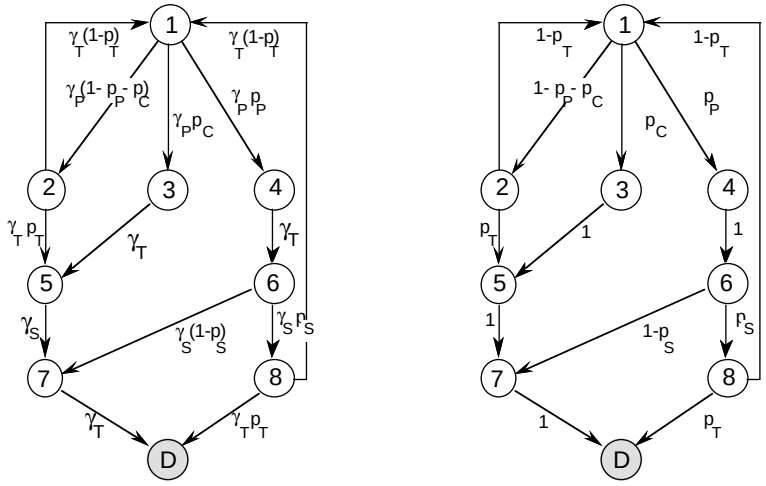


$f(p_{supP},p_{infP},p_{supC},p_{infC}) = 1-\{m(Pf)p_{supP} + (1-m(Pf))p_{infP}\} -\{m(Pf)p_{supC} + (1-m(Pf))p_{infC}\}$	$\theta_X^* = \frac{\theta_X}{m(P)+m(C)+m(T)+m(S)}$
$f(p_{supT},p_{infT}) = 1-\{m(Pf)p_{supT} + (1-m(Pf))p_{infT}\}$	$\bar{\theta}_X^* = \frac{\bar{\theta}_X}{m(P)+m(C)+m(T)+m(S)}$
$f(p_{supS},p_{infS}) = 1-\{m(Pf)p_{supS} + (1-m(Pf))p_{infS}\}$	$X \in \{P,T,S,C\}$

Figure V.22 : Modèle RdPSG d'un BR en croissance de fiabilité

V.2.3.2 Chaînes de Markov en fiabilité stabilisée et en croissance de fiabilité

La chaîne de Markov générée à partir du graphe des marquages du RdPSG du BR en fiabilité stabilisée est présentée sur la figure V.23-a. La chaîne immergée associée à cette chaîne de Markov est présentée sur la figure V.23-b. Celle-ci décrit le comportement en fiabilité stabilisée du BR en temps discret.



a) b)

Figure V.23 : Chaîne de Markov du BR en fiabilité stabilisée en temps continu (a) et chaîne immergée associée (b)

La chaîne de Markov décrivant le comportement du BR en croissance de fiabilité qui est générée à partir du graphe des marquages du RdPSG correspondant et la chaîne immergée associée sont présentées respectivement sur les figures V.24 et V.25.

Pour la clarté des figures, on note respectivement par $p_{s,X}$ et $p_{i,X}$ les paramètres $p_{sup,X}$ et $p_{inf,X}$, $X \in \{P, S, T\}$

La chaîne de Markov transformée ainsi obtenue contient 60 états. Elle est similaire à celle que nous avons obtenue dans le chapitre IV (figure IV.25) dans le cas de la modélisation du comportement du BR par le modèle hyperexponentiel en temps continu. La différence fondamentale entre ces deux chaînes de Markov est la suivante : contrairement à la chaîne de Markov transformée obtenue dans le cadre de l'application de l'approche de modélisation en temps continu, nous distinguons, dans le cas discret, les probabilités d'activation des fautes associées à un composant du taux d'exécution du composant correspondant ; ce qui permet de prendre en compte, dans l'évaluation des mesures de sûreté de fonctionnement du logiciel, la variation des taux d'exécution de ses composants quand il est utilisé dans des environnements différents.

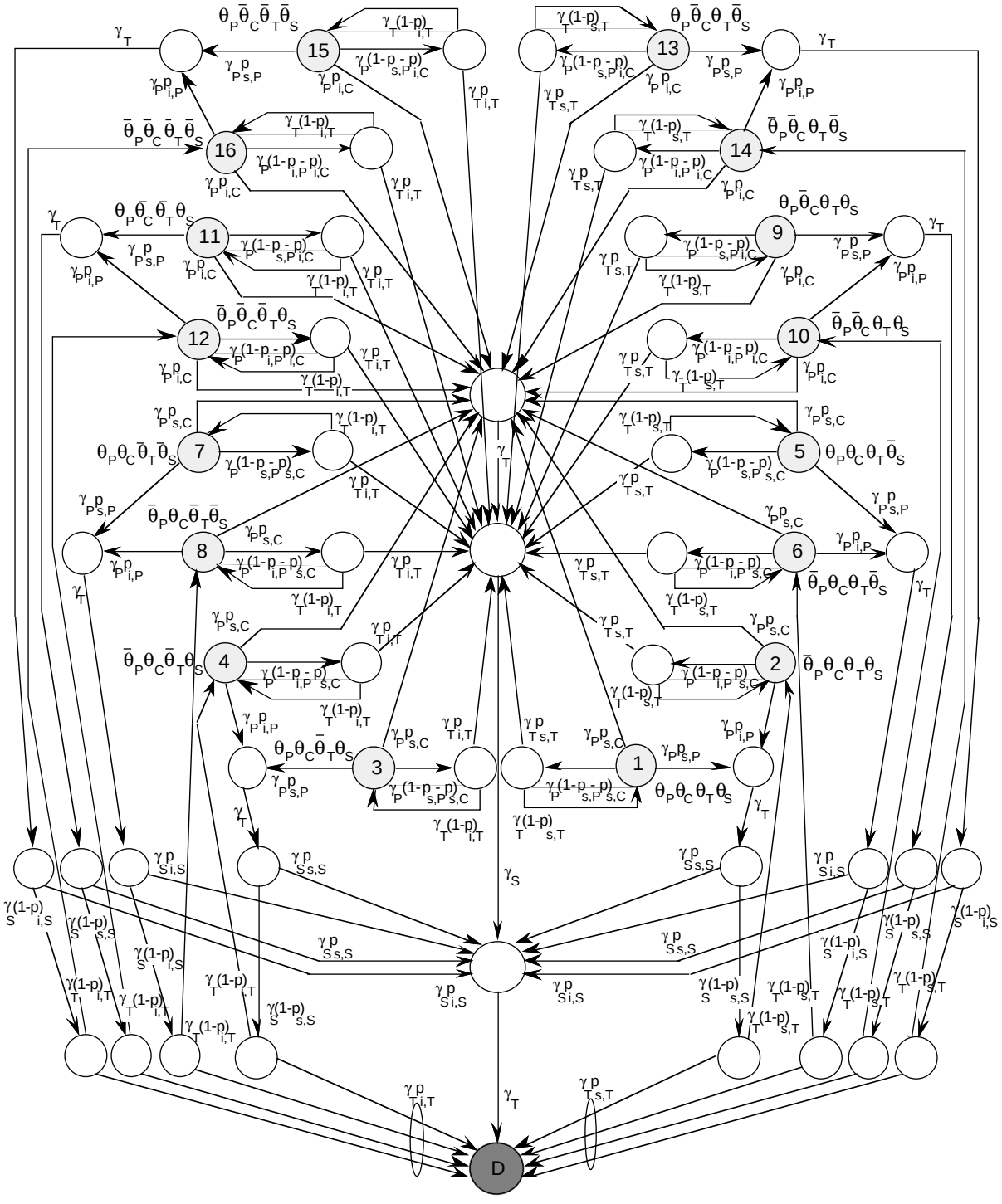


Figure V.24 : Chaîne de Markov du BR en croissance de fiabilité en temps continu

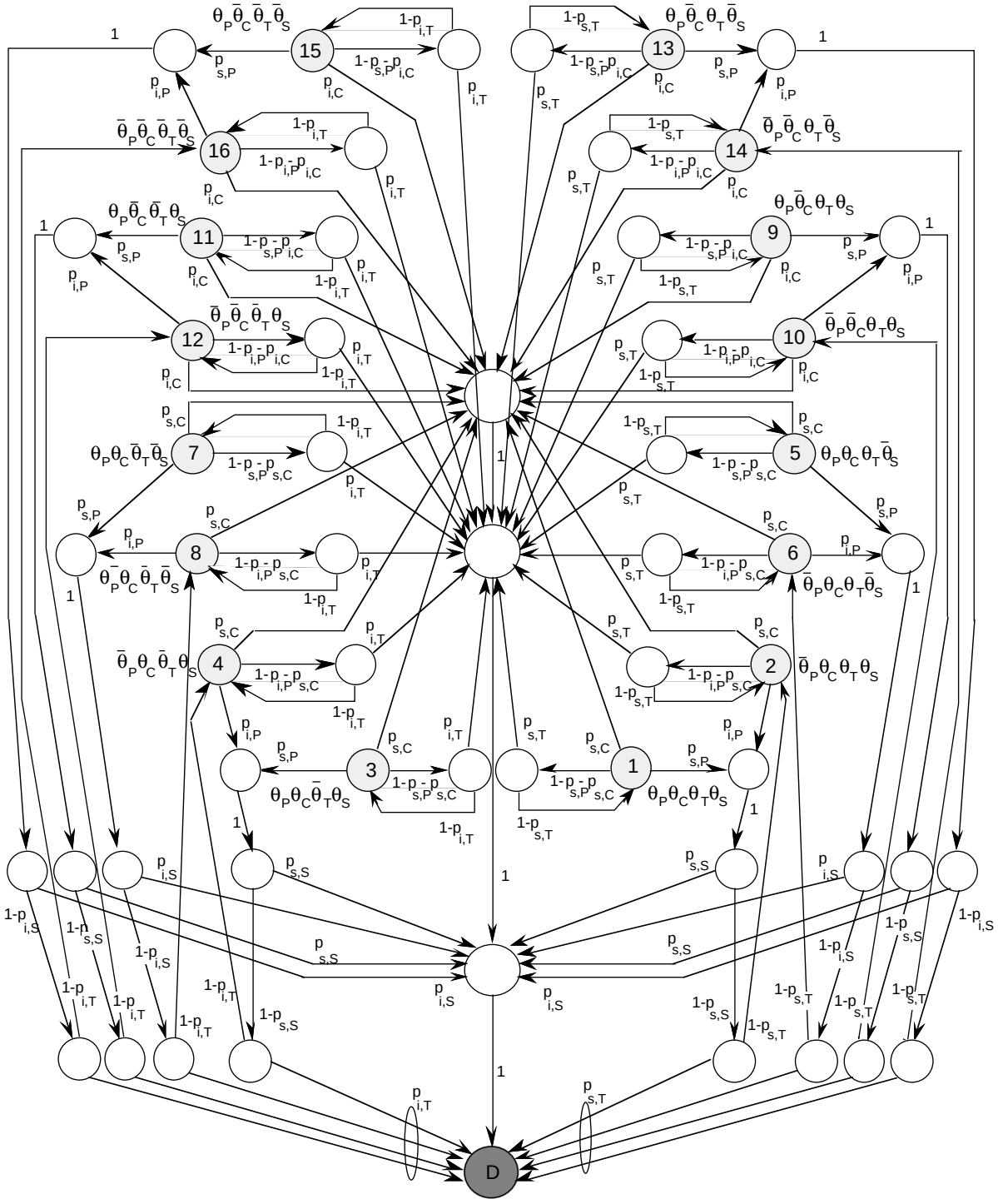


Figure V.25 : Chaîne de Markov en temps discret du BR en croissance de fiabilité

V.2.3.3 Mesures de croissance de fiabilité

Nous évaluerons dans un premier temps, les mesures caractérisant l'évolution du comportement du BR en fonction du nombre d'exécutions effectuées à partir de la chaîne immergée transformée (figure V.25), et dans un deuxième temps, les mesures qui caractérisent l'évolution dans le temps du comportement du BR en tenant compte des taux d'exécution des différents composants du logiciel en considérant la chaîne de Markov en temps continu présentée sur la figure V.24.

V.2.3.3.1 Mesures en temps discret

Considérons la chaîne de Markov en temps discret présentée sur la figure V.25. On se propose d'évaluer la fiabilité $R(n,n+n_0)$ du BR en utilisant la méthode présentée dans le paragraphe V.1.2.1.

Les états {1, 2…,16} constituent le sous-ensemble B des états qui caractérisent la sollicitation du BR pour une nouvelle exécution. Compte tenu des résultats présentés dans V.1.2.1, on peut définir à partir de la chaîne de Markov considérée une chaîne de Markov réduite constituée du sous-ensemble d'états B et de l'état absorbant. La chaîne de Markov réduite, déduite de la chaîne de Markov transformée en temps discret est présentée sur la figure V.26. La matrice des probabilités de transition associée à la chaîne agrégée est calculée à partir de la relation V.10.

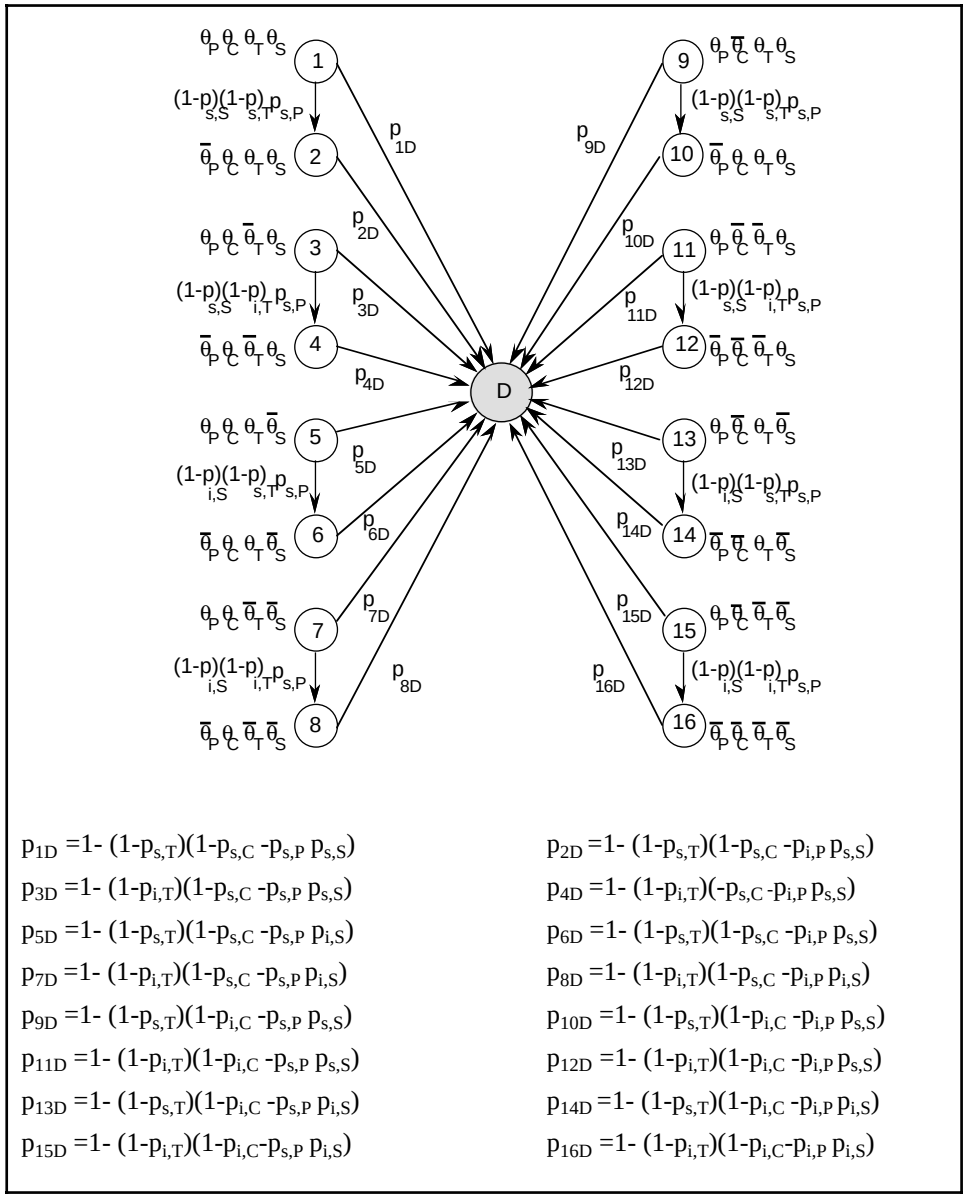


Figure V.26 : BR en croissance de fiabilité : chaîne de Markov réduite

ETABLISSEMENT DE L'EXPRESSION DE LA FIABILITé $R(n,n+n_0)$

Soit p_{ii} la probabilité de ne pas quitter l'état i suite à l'exécution effectuée :

$$p_{ii} = 1 - \sum_{j} p_{ij} \qquad i \neq j \text{ et } i,j \in B = \{1,2,\ldots 16\} \tag{V.41}$$

Le calcul de la fonction de survie r(n) à partir de la chaîne de Markov réduite donne :

$$r(n) = \sum_{i} \beta_i \left\{1 + \frac{1-p_{iD} - p_{ii}}{p_{ii} - p_{jj}}\right\} p_{ii}^n + \left\{\beta_j - \beta_i \frac{1 - p_{iD} - p_{ii}}{p_{ii} - p_{jj}}\right\} p_{jj}^n \tag{V.42}$$

avec $i \in \{1, 3, 5, 7, 9, 11, 13, 15\}$ et $j = i+1$

β_i est la probabilité initiale d'occupation de l'état i. Les expressions des β_i sont données sur la figure V.26.

La relation (V.42) peut se mettre sous la forme :

$$r(n) = \sum_{i \in B} \upsilon_i \; p_{ii}^n \tag{V.43}$$

avec

$$\begin{cases} \upsilon_i = \beta_i \left\{1 + \dfrac{1-p_{iD} - p_{ii}}{p_{ii} - p_{jj}}\right\} & j = i+1 \quad \text{et} \quad i \in \{1, 3, 5, 7, 9, 11, 13, 15\} \\ \upsilon_i = \beta_i - \beta_j \dfrac{1 - p_{jD} - p_{jj}}{p_{jj} - p_{ii}} & j = i-1 \text{ et } i \in \{2, 4, 6, 8, 10, 12, 14, 16\} \end{cases}$$

Soit P(n) la probabilité de défaillance à l'exécution n du BR : $P(n) = 1 - \dfrac{r(n)}{r(n-1)}$.

Compte tenu de la relations (V.43), on trouve :

$$P(n) = \frac{\sum_{i \in B} \upsilon_i (1 - p_{ii}) \, p_{ii}^{n-1}}{\sum_{i \in B} \upsilon_i \; p_{ii}^{n-1}} \qquad \text{avec} \sum_{i \in B} \upsilon_i = 1 \text{ et } \upsilon_i \geq 0 \tag{V.44}$$

On remarque alors que la probabilité P(n) caractérisant l'évolution de la probabilité de défaillance du BR en fonction du nombre d'exécutions effectuées a la même forme que l'expression générale de la probabilité de défaillance à l'exécution d'un modèle hyperexponentiel en temps discret. On retrouve ainsi un résultat similaire à celui obtenu dans le cas d'une modélisation de la croissance de fiabilité d'un bloc de recouvrement basée sur le modèle hyperexponentiel en temps continu.

Evaluons maintenant l'expression de la fonction de fiabilité $R(n,n+n_0) = \frac{r(n+n_0)}{r(n)}$ donnant l'évolution de la fiabilité du bloc de recouvrement pour un nombre n_0 fixe d'exécutions. n_0 peut correspondre par exemple au nombre d'exécutions effectuées par le logiciel tolérant aux fautes au cours d'une mission (cas d'un bloc de recouvrement installé sur un système avionique).

On obtient à partir de la relation (V.43) :

$$R(n,n+n_0) = \frac{\sum_{i \in B} \upsilon_i \, p_{ii}^{n+n_0}}{\sum_{i \in B} \upsilon_i \, p_{ii}^{n_0}} \qquad \text{(V.45)}$$

REMARQUES

1- On peut vérifier que l'on retrouve la relation (V.45) donnant l'expression de $R(n,n+n_0)$ relative à l'exemple traité dans le paragraphe V.1.1 concernant un BR constitué d'un primaire et d'un secondaire et pour lequel on n'a modélisé que l'influence de fautes indépendantes sur le comportement du logiciel. Il suffit de poser :

$$\begin{cases} \theta_S = 0 \text{ et } p_{i,S} = p_S \text{ : fiabilité stabilisée du secondaire} \\ \theta_X = 0 \text{ et } p_{i,X} = 0 \text{ pour } X \in \{T,C\} \end{cases}$$

2- A partir de la relation (V.45) on peut également déduire l'expression de la fonction $R^*(n,n+n_0)$ associée à la chaîne de Markov de la figure V.23-b caractérisant le comportement en fiabilité stabilisée du BR. Il suffit de poser $\theta_X = 0$ et $p_X = p_{i,X}$ pour tout $X \in \{P,S,T,C\}$.

On trouve :

$$R^*(n,n+n_0) = \left\{ (1-p_C)(1-p_T)(1-p_P\, p_S) \right\}^{n_0} \qquad \text{(V.46)}$$

Etant donné que p_P, p_S, p_T et $p_C << 1$, on a :

$$(1-p_C)(1-p_T)(1-p_P\, p_S) \approx 1- (p_C + p_T + p_P\, p_S) \qquad \text{(V.47)}$$

où $p_C + p_T + p_P\, p_S$ est la probabilité de défaillance à l'exécution du BR en fiabilité stabilisée.

On retrouve ainsi un résultat équivalent à celui obtenu dans [Arlat 88-a].

VARIATION DE LA FIABILITé $R(n,n+n_0)$ EN FONCTION DES PARAMèTRES DU MODèLE

Sur la figure V.27 , nous présentons des exemples d'évolution de la fonction $1-R(n,n+n_0)$ pour observer l'évolution de la croissance de fiabilité du BR en fonction de celle de ses différents composants .

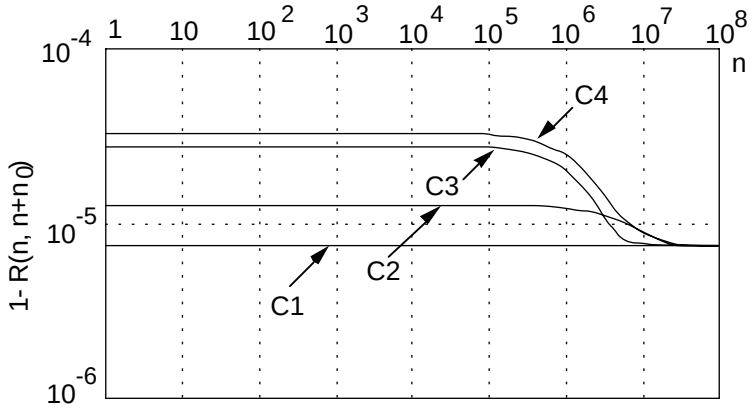
Les valeurs considérées pour les paramètres du modèle caractérisant la croissance de fiabilité du BR sont les suivantes :

$k_P = 10$	$k_S = 2$	$k_T = 5$	$k_C = 3$
$\theta_P = 0,5$	$\theta_S = 0,5$	$\theta_T = 0,5$	$\theta_C = 0,5$
$p_{infP} = 10^{-6}$	$p_{infS} = 10^{-6}$	$p_{infT} = 10^{-7}$	$p_{infC} = 510^{-8}$

k_X est le facteur de croissance défini par :

$$k_X = \frac{P_X(1)}{P_X(\infty)} = \theta_X \frac{p_{sX}}{p_{iX}} + \bar{\theta}_X \qquad X \in \{P,S,T,C\} \tag{V.48}$$

Dans le cas où on suppose une fiabilité stabilisée(FS) du BR vis-à-vis des fautes X, $X \in \{P,S,T,C\}$, alors : $\theta_X = 0$, $k_X = 1$ et $p_X = p_{infX}$.



Courbe	P	S	T	C	Commentaires
C1	FS	FS	FS	FS	on ne tient pas compte du phénomène de croissance de fiabilité
C2	FS	FS	FS	CF	on considère uniquement la croissance de fiabilité résultant de l'élimination des fautes corrélées
C3	FS	FS	CF	FS	on considère uniquement la croissance de fiabilité résultant de l'élimination des fautes indépendantes dans T
C4	CF	CF	CF	CF	on tient compte de la croissance de fiabilité résultant de l'élimination de toutes les fautes considérées

FS : fiabilité stabilisée CF : croissance de fiabilité

Figure V.27 : Courbes d'évolution de la fonction : 1-R(n,n+50)

Au vu des résultats présentés sur la figure V.27, on observe les mêmes types de comportement du BR en fonction de celui de ses différents composants que ceux observés dans le cas d'une modélisation de la croissance de fiabilité du logiciel basée sur le modèle hyperexponentiel en temps continu . En effet, on peut remarquer qu'il existe une différence significative entre les estimations effectuées en prenant en compte le phénomène de croissance de fiabilité et celles obtenues en faisant l'hypothèse que le BR est en fiabilité stabilisée. On remarque aussi que la croissance de fiabilité du bloc de recouvrement est essentiellement liée à la croissance de fiabilité du test d'acceptation et à la diminution progressive de la probabilité d'activation des fautes corrélées. La différence qui existe entre les courbes C4 et C1 dépend essentiellement des valeurs des facteurs de croissance k_T et k_C. L'influence sur l'évolution de la croissance de fiabilité du BR de l'élimination des fautes indépendantes du primaire et du secondaire est très faible. Par conséquent, l'erreur d'estimation due à la non prise en compte du

phénomène de croissance de fiabilité n'est faible que si la probabilité d'activation à l'exécution de fautes corrélées tend à être constante et si le test d'acceptation ne manifeste aucune croissance de fiabilité au cours de la période considérée.

V.2.3.3.2 Mesures en temps continu

Considérons la chaîne de Markov transformée de la figure V.15 qui caractérise l'évolution de la croissance de fiabilité du BR en tenant compte des taux d'exécution du primaire du secondaire et du test d'acceptation : γ_P, γ_S et γ_T. Ces derniers paramètres caractérisent l'environnement dans lequel le logiciel est utilisé.

On se propose d'évaluer la fonction R(t,t+u) caractérisant l'évolution dans le temps du comportement du BR et d'étudier la variation de R(t,t+u) en fonction des paramètres γ_P, γ_S et γ_T.

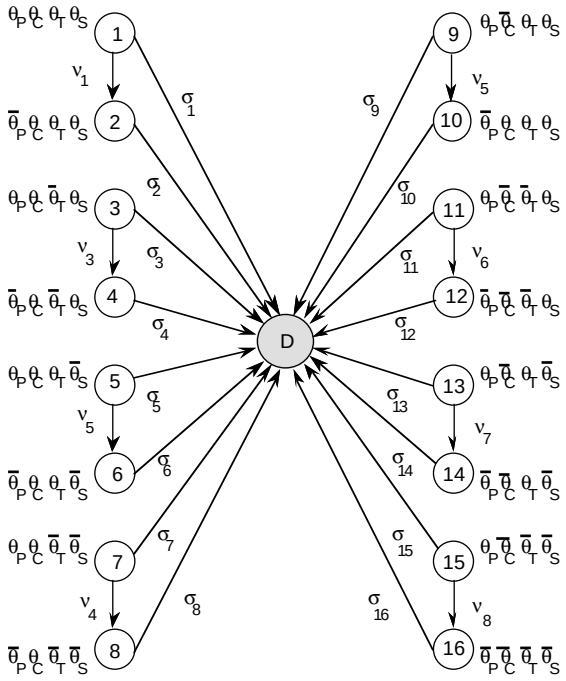
L'application de la technique d'agrégation présentée dans le paragraphe IV.3.2.3 conduit à la chaîne de Markov réduite présentée sur la figure V.28.

L'expression de la fiabilité R(t,t+u) évaluée à partir de celle-ci est la suivante :

$$R(t,t+u) = \frac{\sum_{i=1}^{8} \beta_i (1+A_i) \exp(-(\sigma_i+\nu_i)(t+u)) + (\beta_{i+1} - \beta_i A_i) \exp(-\sigma_{i+1}(t+u))}{\sum_{i=1}^{8} \beta_i (1+A_i) \exp(-(\sigma_i+\nu_i)t) + (\beta_{i+1} - \beta_i A_i) \exp(-\sigma_{i+1}t)} \qquad \text{(V.49)}$$

où $A_i = \frac{\nu_i}{\sigma_{i+1}-\sigma_i-\nu_i}$ et β_i est la probabilité initiale d'occupation de l'état i. σ_i et ν_i sont les taux de transition associées aux états de la chaîne de Markov réduite. Les expressions des paramètres β_i, σ_i et ν_i sont données sur la figure V.28.

Pour les mêmes raisons que celles évoquées dans le paragraphe V.1.3, on remarque que seuls γ_P et γ_T interviennent dans l'expression de R(t,t+u).



$\sigma_1=(p_{sC} + p_{sP}(p_{sS} + (1-p_{sS})p_{sT})\gamma_P\pi_{1P} + p_{sT}\gamma_T\pi_{1T}$

$\sigma_2 =(p_{sC}+p_{sP}(p_{sS} + (1-p_{sS})p_{iT})\gamma_P\pi_{2P} + p_{iT}\gamma_T\pi_{2T}$

$\sigma_3 =(p_{sC} + p_{sP}(p_{iS} + (1-p_{iS})p_{sT})\gamma_P\pi_{1P} + p_{sT}\gamma_T\pi_{1T}$

$\sigma_4 =(p_{sC}+p_{sP}(p_{iS} + (1-p_{iS})p_{iT})\gamma_P\pi_{2P} + p_{iT}\gamma_T\pi_{2T}$

$\sigma_5 =(p_{iC} + p_{sP}(p_{sS} + (1-p_{sS})p_{sT})\gamma_P\pi_{3P} + p_{sT}\gamma_T\pi_{3T}$

$\sigma_6 =(p_{iC}+p_{sP}(p_{sS} + (1-p_{sS})p_{iT})\gamma_P\pi_{4P} + p_{iT}\gamma_T\pi_{4T}$

$\sigma_7 =(p_{iC} + p_{sP}(p_{iS} + (1-p_{iS})p_{sT})\gamma_P\pi_{3P} + p_{sT}\gamma_T\pi_{3T}$

$\sigma_8 =(p_{iC}+p_{sP}(p_{iS} + (1-p_{iS})p_{iT})\gamma_P\pi_{4P} + p_{iT}\gamma_T\pi_{4T}$

$\sigma_9 =(p_{sC} + p_{iP}(p_{sS} + (1-p_{sS})p_{sT})\gamma_P\pi_{5P} + p_{sT}\gamma_T\pi_{5T}$

$\sigma_{10} =(p_{sC}+p_{iP}(p_{sS} + (1-p_{sS})p_{iT})\gamma_P\pi_{6P} + p_{iT}\gamma_T\pi_{6T}$

$\sigma_{11} =(p_{sC} + p_{iP}(p_{iS} + (1-p_{iS})p_{sT})\gamma_P\pi_{5P} + p_{sT}\gamma_T\pi_{5T}$

$\sigma_{12} =(p_{sC}+p_{iP}(p_{iS} + (1-p_{iS})p_{iT})\gamma_P\pi_{6P} + p_{iT}\gamma_T\pi_{6T}$

$\sigma_{13} =(p_{iC} + p_{iP}(p_{sS} + (1-p_{sS})p_{sT})\gamma_P\pi_{7P} + p_{sT}\gamma_T\pi_{7T}$

$\sigma_{14} =(p_{iC}+p_{iP}(p_{sS} + (1-p_{sS})p_{iT})\gamma_P\pi_{8P} + p_{iT}\gamma_T\pi_{8T}$

$\sigma_{15} =(p_{iC} + p_{iP}(p_{iS} + (1-p_{iS})p_{sT})\gamma_P\pi_{7P} + p_{sT}\gamma_T\pi_{7T}$

$\sigma_{16} =(p_{iC}+p_{iP}(p_{iS} + (1-p_{iS})p_{iT})\gamma_P\pi_{8P} + p_{iT}\gamma_T\pi_{8T}$

$$\pi_{1P} = \frac{(1-p_{sT})\gamma_T}{(1-p_{sT})\gamma_T+(1-p_{sC}-p_{sP})\gamma_P}$$

$$\pi_{2P} = \frac{(1-p_{iT})\gamma_T}{(1-p_{iT})\gamma_T+(1-p_{sC}-p_{sP})\gamma_P}$$

$$\pi_{3P} = \frac{(1-p_{sT})\gamma_T}{(1-p_{sT})\gamma_T+(1-p_{iC}-p_{sP})\gamma_P}$$

$$\pi_{4P} = \frac{(1-p_{iT})\gamma_T}{(1-p_{iT})\gamma_T+(1-p_{iC}-p_{sP})\gamma_P}$$

$$\pi_{5P} = \frac{(1-p_{sT})\gamma_T}{(1-p_{sT})\gamma_T+(1-p_{sC}-p_{iP})\gamma_P}$$

$$\pi_{6P} = \frac{(1-p_{iT})\gamma_T}{(1-p_{iT})\gamma_T+(1-p_{sC}-p_{iP})\gamma_P}$$

$$\pi_{7P} = \frac{(1-p_{sT})\gamma_T}{(1-p_{sT})\gamma_T+(1-p_{iC}-p_{iP})\gamma_P}$$

$$\pi_{8P} = \frac{(1-p_{iT})\gamma_T}{(1-p_{iT})\gamma_T+(1-p_{iC}-p_{iP})\gamma_P}$$

$\pi_{xT} =1-\pi_{xP} \quad x = \{1,\ldots,8\}$

$\nu_1=(1-p_{sT})(1-p_{sS})p_{sP}\gamma_P\pi_{1P}$

$\nu_2=(1-p_{iT})(1-p_{sS})p_{sP}\gamma_P\pi_{2P}$

$\nu_3=(1-p_{sT})(1-p_{iS})p_{sP}\gamma_P\pi_{1P}$

$\nu_4=(1-p_{iT})(1-p_{iS})p_{sP}\gamma_P\pi_{2P}$

$\nu_5=(1-p_{sT})(1-p_{sS})p_{sP}\gamma_P\pi_{3P}$

$\nu_6=(1-p_{iT})(1-p_{sS})p_{sP}\gamma_P\pi_{4P}$

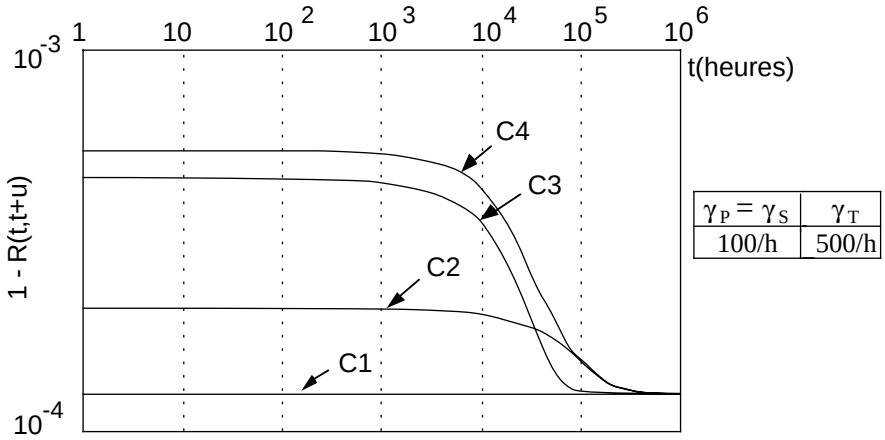
$\nu_7=(1-p_{sT})(1-p_{iS})p_{sP}\gamma_P\pi_{3P}$

$\nu_8=(1-p_{iT})(1-p_{iS})p_{sP}\gamma_P\pi_{4P}$

Figure V.28 : Chaîne de Markov réduite du BR en croissance de fiabilité.

Sur la figure V.29, nous donnons des exemples d'évolution de la fonction 1- R(t,t+u) pour des valeurs γ_P, γ_S et γ_T données afin d'observer l'évolution, dans un environnement donné, de la croissance de fiabilité du BR en fonction de celles de ses différents composants. Les paramètres caractérisant la croissance de fiabilité de ces derniers sont les mêmes que ceux que nous avons

considérés pour l'évaluation de la fiabilité du BR en fonction du nombre d'exécutions effectuées. On peut remarquer que l'on observe les mêmes types d'évolution de la fiabilité du BR qu'en temps discret. La raison en est que la croissance de fiabilité du BR résulte de la diminution des probabilités d'activation à l'exécution des fautes de conception du logiciel au fur et à mesure de l'élimination des fautes ce qui a également pour conséquence directe d'améliorer la fiabilité en temps continu du logiciel.



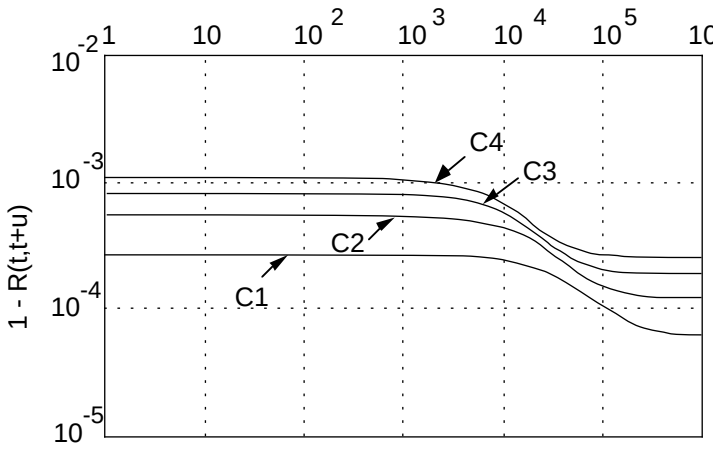
$\gamma_P = \gamma_S$	γ_T
100/h	500/h

Courbe	P	S	T	C	Commentaires
C1	FS	FS	FS	FS	on ne tient pas compte du phénomène de croissance de fiabilité
C2	FS	FS	FS	CF	on considère uniquement la croissance de fiabilité résultant de l'élimination des fautes corrélées
C3	FS	FS	CF	FS	on considère uniquement la croissance de fiabilité résultant de l'élimination des fautes indépendantes dans T
C4	CF	CF	CF	CF	on tient compte de la croissance de fiabilité résultant de l'élimination de toutes les fautes considérées

FS : fiabilité stabilisée CF : croissance de fiabilité

Figure V.29 : Courbes d'évolution de la fiabilité du BR

La figure V.30 illustre l'évolution de la fiabilité dans différents environnements d'utilisation du logiciel en considérant le cas où tous les composants du BR manifestent une croissance de fiabilité pendant la période considérée.



Courbe	$\gamma_P = \gamma_S$	γ_T
C1	50/h	250/h
C2	100/h	500/h
C3	150/h	750/h
C4	200/h	100/h

Figure V.30 : Variation de la fiabilité du BR en fonction de γ_P, γ_S et γ_T

Les différentes allures d'évolution de la fonction 1-R(t,t+u) caractérisent le comportement du BR tel qu'il est perçu par les utilisateurs du logiciel dans les environnements considérés. On remarque qu'il existe une différence significative entre les différentes courbes obtenues : plus la variation des paramètres γ_S et γ_T est importante plus l'écart entre les courbes 1-R(t,t+u) correspondantes est grand.

V.2.4 Conclusion

L'application de l'approche de transformation basée sur le modèle hyperexponentiel en temps discret pour l'évaluation de la croissance de fiabilité en temps discret et en temps continu d'un bloc de recouvrement conduit à des résultats similaires à ceux obtenus dans le chapitre IV dans le cas d'une modélisation de la croissance de fiabilité du BR basée sur le modèle hyperexponentiel en temps continu. Néanmoins, l'approche définie dans ce chapitre a l'avantage de pouvoir prendre en compte la variation des taux d'exécution des différents composants du logiciel dans l'évaluation des mesures de fiabilité. Ceci permet, dans le cas où le logiciel est utilisé dans des environnements différents, d'estimer les mesures de croissance de fiabilité du logiciel qui sont spécifiques à chacun de ces environnements.

CONCLUSION

Grâce à l'interprétation markovienne du modèle hyperexponentiel en temps discret nous avons défini dans ce chapitre une approche markovienne similaire à l'approche de modélisation en temps continu présentée dans le chapitre IV qui permet d'évaluer à la fois les mesures en temps discret et les mesures en temps continu caractérisant le comportement d'un logiciel multi-composant en tenant compte du phénomène de croissance de fiabilité. La différence entre les deux approches réside dans le fait que dans le cas de l'approche basée sur le modèle hyperexponentiel en temps continu, on transforme des taux de défaillance conformément à la représentation markovienne du modèle, et dans le cas de l'approche basée sur le modèle hyperexponentiel en temps discret on transforme des probabilités conditionnelles de défaillance à l'exécution en considérant la représentation markovienne du modèle hyperexponentiel en temps discret. La modélisation en temps continu du comportement d'un logiciel multi-composant en distinguant les probabilités de défaillance à l'exécution des composants du logiciel des taux d'exécution de ces composants permet d'évaluer les mesures en temps continu de la croissance de fiabilité du logiciel en tenant compte des caractéristiques de l'environnement dans lequel il est utilisé.

L'approche proposée nous a permis d'évaluer la croissance de fiabilité d'un bloc de recouvrement en fonction du nombre d'exécutions effectuées. Ceci constitue une nouveauté dans le domaine puisqu'à notre connaissance, toutes les études publiées sur la modélisation du comportement d'un bloc de recouvrement ne tiennent pas compte du phénomène de croissance de fiabilité [Cha 86, Arlat 88-a, Csenki 89, Pucci 90].

Néanmoins, l'approche proposée pour prendre en compte les caractéristiques de l'environnement d'utilisation d'un logiciel dans l'évaluation de son comportement tel qu'il est perçu par ses utilisateurs permet de prendre en compte de façon explicite certaines caractéristiques de l'environnement d'utilisation du logiciel : les fréquences d'exécution des différents composants du système. Nous avons supposé dans ce chapitre que le parcours de

l'espace des entrées du logiciel s'effectue de façon identique en probabilité dans les différents environnements considérés. Reste donc maintenant à étendre l'approche pour tenir compte de cette dernière source de variation de l'environnement d'utilisation du logiciel dans l'évaluation de ses mesures de fiabilité.

Par ailleurs, dans ce chapitre, nous avons étudié des logiciels multi-composant qui sont tels que, à chaque instant d'utilisation du logiciel, un seul composant est activé ; c'est-à-dire les logiciels qui s'exécutent de façon séquentielle. Nos futurs travaux consisteront à étendre cette approche pour prendre en compte le cas où plusieurs composants du logiciel s'exécutent en parallèle.

CONCLUSION GÉNÉRALE

Ce mémoire est consacré à la modélisation et à l'évaluation de la croissance de sûreté de fonctionnement des systèmes en considérant deux types de représentation de leur comportement : une représentation en temps continu permettant de caractériser et d'évaluer l'évolution dans le temps des mesures de sûreté de fonctionnement d'un système matériel et logiciel, et une représentation en temps discret, plus adaptée à l'étude du comportement du logiciel, permettant de caractériser l'évolution de la fiabilité d'un système en fonction du nombre d'exécutions effectuées. Pour ces deux types de modélisation, nous avons d'abord étudié le comportement des systèmes selon une vue "boîte noire" puis nous nous sommes intéressé à la modélisation de la croissance de sûreté de fonctionnement des systèmes en tenant compte de leur structure, c'est-à-dire en considérant une approche "boîte blanche".

Nous reprenons dans cette conclusion les principaux résultats exposés dans ce mémoire qui sont relatifs à la modélisation, en **temps continu** et en **temps discret,** de la croissance de sûreté de fonctionnement des systèmes selon une approche **"boîte noire"** et selon une approche **"boîte blanche"**, en indiquant à chaque étape les voies susceptibles de développements futurs.

*
* *

Concernant la modélisation en temps continu de la sûreté de fonctionnement des systèmes, selon une approche "boîte noire", en tenant compte de l'élimination progressive des fautes de conception durant les phases de validation et de vie opérationnelle, nous avons étudié et analysé les caractéristiques d'un modèle déjà existant, le modèle hyperexponentiel en temps continu. Le modèle considéré se distingue des modèles existants par son aptitude à modéliser à la fois la croissance de fiabilité et la croissance de disponibilité des systèmes. La modélisation de la croissance de disponibilité est effectuée en considérant les propriétés NHPP du modèle et les caractéristiques de sa fonction intensité de défaillance. Grâce à ces propriétés, le modèle est représenté sous forme d'une chaîne de Markov en temps continu et l'évaluation de la disponibilité est effectuée à partir de la chaîne de Markov ainsi définie. En considérant l'interprétation markovienne du modèle hyperexponentiel, nous avons montré que la modélisation du phénomène de croissance de fiabilité d'un système peut se ramener à la transformation d'une chaîne de Markov caractérisant le comportement d'un système en fiabilité stabilisée en une autre chaîne de Markov permettant de prendre en compte le phénomène de croissance de fiabilité. Cette transformation est fondamentale car elle constitue la base de l'approche que nous avons exposée pour modéliser la croissance de sûreté de fonctionnement d'un système à partir de celle de ses différents composants.

En nous basant sur les caractéristiques de la fonction intensité de défaillance du modèle, nous avons proposé également une généralisation du modèle et nous avons montré que le modèle généralisé permet d'augmenter la puissance de modélisation du modèle et d'améliorer son aptitude à prendre en compte les variations de la fiabilité et de la disponibilité des systèmes au cours de leur cycle de vie.

Pour illustrer l'aptitude du modèle à évaluer la croissance de fiabilité et la croissance de disponibilité des systèmes, nous l'avons appliqué à deux logiciels de télécommunications. Nous avons montré que pour obtenir des estimations qui soient représentatives du comportement des systèmes, il est préférable d'effectuer, avant l'application du modèle, une analyse de tendance permettant d'étudier les variations de la fiabilité observée du logiciel considéré et de tenir compte des résultats de cette analyse lors de l'application du modèle.

Concernant la modélisation selon une approche "boîte blanche" du comportement de systèmes multi-composant, nous avons exposé une approche de modélisation basée sur le modèle hyperexponentiel en temps continu pour modéliser la croissance de fiabilité et la croissance de disponibilité des systèmes constitués de composants matériels et de composants logiciels. L'objectif de l'approche définie est l'unification des études de modélisation de la sûreté de fonctionnement du matériel et du logiciel afin de fournir aux concepteurs et aux utilisateurs des évaluations concernant le comportement du système global en considérant les deux mesures complémentaires de la sûreté de fonctionnement que sont la fiabilité et la disponibilité et en tenant compte des deux principales sources de défaillance des systèmes informatiques que sont les fautes physiques du matériel et les fautes de conception introduites dans le matériel et dans le logiciel. De telles évaluations ne sont pas courantes bien qu'elles soient indispensables aussi bien pour les concepteurs que pour les utilisateurs des systèmes informatiques.

L'approche que nous avons exposée dans ce mémoire consiste à représenter le comportement en fiabilité stabilisée d'un système en fonction de celui de ses différents composants par une chaîne de Markov et à transformer ensuite la chaîne de Markov ainsi obtenue pour tenir compte du phénomène de croissance de fiabilité. La transformation est basée sur la représentation de la croissance de sûreté de fonctionnement de chaque composant du système par un modèle hyperexponentiel, et par suite par une chaîne de Markov. L'utilisation d'une approche similaire à l'approche classique de modélisation de systèmes complexes par des chaînes de Markov homogènes permet de réutiliser tous les résultats obtenus dans le domaine de la construction et du traitement de modèles markoviens. En particulier, nous avons utilisé comme outil de base, pour la construction de modèles des systèmes en fiabilité stabilisée et pour la mise en œuvre de la transformation, les réseaux de Petri stochastiques généralisés. Nous avons exposé une approche générale de modélisation basée sur cet outil et nous avons illustré cette approche sur plusieurs exemples pour montrer ses possibilités. Nous avons présenté, également, la méthode à suivre pour évaluer les mesures de croissance de sûreté de fonctionnement.

La technique de transformation exposée dans ce mémoire conduit à l'augmentation de l'espace des états du système considéré. Cependant, comme nous l'avons montré sur des exemples, une telle limitation n'est pas très contraignante puisqu'il existe actuellement des techniques analytiques et numériques très puissantes permettant de traiter des chaînes de Markov de très grande taille.

Nous avons montré à travers les différents exemples présentés que l'approche de modélisation proposée permet d'évaluer la croissance de sûreté de fonctionnement d'un système à partir de celle de ses différents composants et d'effectuer une étude de sensibilité concernant l'influence de chacun des composants du système sur l'évolution de la sûreté de fonctionnement du système global. Reste maintenant à valider l'approche présentée sur des systèmes réels. L'application de l'approche de modélisation de systèmes multi-composant nécessite d'abord la décomposition du

système considéré en plusieurs composants et ensuite la mise en œuvre d'une procédure de collecte de données afin de fournir les informations qui sont nécessaires pour estimer les paramètres du modèle hyperexponentiel qui caractérisent la croissance de sûreté de fonctionnement des différents composants du système considéré. La décomposition du système doit être effectuée en fonction des objectifs de l'étude de sûreté de fonctionnement entreprise :

- une décomposition trop fine du système peut, d'une part, conduire à des modèles qui sont très complexes et à des évaluations qui ne sont pas de grand intérêt pour les concepteurs, et d'autre part, avoir comme conséquence l'impossibilité d'estimer les paramètres qui caractérisent la croissance de sûreté de fonctionnement des différents composants du système si les données collectées sont insuffisantes,

- une décomposition de très haut niveau conduit à des modèles plus simples mais peut avoir l'inconvénient de ne pas mettre en évidence certains phénomènes qui agissent de façon significative sur le comportement du système.

Par conséquent, il faudra trouver le meilleur compromis entre ces deux types de décomposition. Des exemples d'approches pour décomposer des systèmes peuvent se trouver dans [Parnas 72, Anderson 81, Levendel 90].

Dans ce mémoire, nous avons défini les fondements de l'approche de transformation. Néanmoins, afin de permettre l'application de cette approche à des systèmes très complexes, il est nécessaire de poursuivre les travaux effectués dans le cadre de la thèse dans l'objectif de décrire sous une forme algorithmique la technique de transformation pour sa mise en œuvre de façon mécanique par un outil. L'objectif sera donc de développer un outil tel que la seule tâche d'un utilisateur donné sera de construire le modèle en fiabilité stabilisée du système considéré et de spécifier les mesures de sûreté de fonctionnement à évaluer ; toutes les étapes concernant la construction du modèle du système en croissance de fiabilité et l'évaluation des mesures seront effectuées mécaniquement par l'outil.

Par ailleurs, une des critiques portées aux études concernant la modélisation de la croissance de sûreté de fonctionnement du logiciel est que l'application des modèles de croissance de fiabilité ne peut être effectuée qu'à un stade avancé du développement, c'est-à-dire à partir de la phase d'intégration. La raison en est que de telles études portent sur un produit donné sans réutiliser les résultats des évaluations effectuées sur des produits antérieurs qui ont été développés dans des environnements similaires. Néanmoins, le développement d'un produit logiciel bénéfice souvent de l'expérience et des connaissances acquises au cours du développement de produits précédents. Il serait donc profitable d'intégrer de telles connaissances dans les modèles d'évaluation de la sûreté de fonctionnement du logiciel pour permettre d'effectuer des estimations plus tôt dans le cycle de vie. Plus précisément, ceci consisterait à collecter des données caractérisant à la fois **le processus** de développement des différents produits développés par un constructeur donné et **les produits** eux mêmes, et de définir des modèles intégrant de telles informations permettant d'estimer la sûreté de fonctionnement d'un produit à partir de celle de ses précédents. Ceci permettra également d'utiliser l'approche de transformation définie dans ce mémoire pour comparer différentes architectures d'un système avant son développement.

*
* *

En ce qui concerne la modélisation en temps discret de la croissance de fiabilité des systèmes, nous avons défini un nouveau modèle de croissance de fiabilité (le modèle hyperexponentiel en temps discret) qui permet d'évaluer le comportement du logiciel en fonction du nombre d'exécutions effectuées. Ce modèle permet d'une part, de caractériser et d'évaluer les mesures de croissance de fiabilité de certains types de systèmes tels que les systèmes transactionnels par exemple, et d'autre part, de distinguer les deux sources de variation des mesures de sûreté de fonctionnement d'un système : la variation qui est liée à l'amélioration du système suite à l'élimination des fautes de conception, et la variation qui est liée au changement de son environnement d'utilisation. Cette dernière source de variation est prise en compte en considérant le taux d'exécution du logiciel dans son environnement d'utilisation considéré, et en transformant le modèle de croissance de fiabilité en temps discret en un modèle en temps continu permettant d'estimer l'évolution dans le temps du comportement du logiciel tel qu'il est perçu par ses utilisateurs.

Le modèle que nous avons défini a été établi en considérant des hypothèses équivalentes à celles du modèle hyperexponentiel en temps continu. Nous avons montré que le modèle obtenu peut être interprété comme la version en temps discret du modèle hyperexponentiel en temps continu. De façon analogue à ce dernier modèle, nous avons associé au modèle hyperexponentiel en temps discret une interprétation markovienne et nous avons montré que la modélisation de la croissance de fiabilité en temps discret par le modèle hyperexponentiel peut se ramener à la transformation d'une chaîne de Markov en temps discret caractérisant le comportement en fiabilité stabilisée du système considéré en une autre chaîne de Markov prenant en compte le phénomène de croissance de fiabilité.

En considérant l'interprétation markovienne du modèle hyperexponentiel en temps discret, nous avons défini une approche permettant d'une part, de modéliser et d'évaluer, en temps discret, la croissance de fiabilité d'un logiciel multi-composant à partir de celle de ses différents composants, et d'autre part, de caractériser le comportement du logiciel tel qu'il est perçu par ses utilisateurs dans un environnement donné en tenant compte des taux d'exécution de ses différents composants. De façon analogue à l'approche de modélisation qui est basée sur le modèle hyperexponentiel en temps continu, nous avons proposé une approche générale basée sur le modèle hyperexponentiel en temps discret utilisant les réseaux de Petri stochastiques généralisés pour formaliser l'approche et mettre en œuvre de façon mécanique la transformation.

Dans le cadre de ce mémoire, nous avons considéré le cas de logiciels multi-composant qui sont tels qu'à chaque instant d'utilisation du logiciel, un seul composant est activé. Bien que, le cas considéré couvre plusieurs types de logiciels, il serait intéressant de généraliser l'approche proposée pour étudier le cas de logiciels qui sont tels que plusieurs composants peuvent s'exécuter en parallèle.

De plus, l'approche que nous avons proposée afin de tenir compte des caractéristiques de l'environnement d'utilisation du logiciel dans l'évaluation de ses mesures de sûreté de fonctionnement constitue une première étape dans la résolution du problème auquel sont confrontés les concepteurs des systèmes informatiques qui est celui de l'évaluation des mesures

de sûreté de fonctionnement d'un système tel qu'il est perçu par ses utilisateurs dans un environnement donné à partir des mesures évaluées dans un autre environnement. Nous avons considéré le cas où la variation des caractéristiques de l'environnement d'utilisation du logiciel se traduit par la variation des taux d'exécution de ses différents composants en supposant que le domaine de défaillance du logiciel ne varie pas en fonction de sa charge. Reste à considérer maintenant le cas où le domaine de défaillance varie en fonction de la charge et en tenir compte lors de l'évaluation des mesures de sûreté de fonctionnement du logiciel.

ANNEXE

SOMME ET PRODUIT DE KRONECKER

La somme et le produit de Kronecker sont souvent utilisés pour la résolution de systèmes d'équations linéaires tant algébriques que différentielles et pour la formulation de dérivées partielles de fonctions de matrices. En particulier, ils sont bien adaptés pour la résolution de systèmes d'équations de Chapman-Kolmogorov associées à des processus markoviens homogènes.

Les principales propriétés de la somme et du produit de Kronecker sont développées dans [Brewer 78] ; leur application pour le traitement de modèles markoviens est abordée dans [Amoia 81, Mazars 81].

Dans cette annexe, nous rappelons les principales définitions de la somme et du produit de Kronecker et nous présentons quelques résultats concernant les processus markoviens homogènes.

A.1 PRODUIT DE KRONECKER

Le produit de Kronecker de deux matrices $\mathbb{A} = [a_{ij}]$, de dimensions (p,q), et $\mathbb{B} = [b_{ij}]$, de dimensions (m,n), est une matrice de dimensions (pm,qn) notée $\mathbb{A} \otimes \mathbb{B}$ donnée par :

$$\mathbb{A} \otimes \mathbb{B} = \begin{pmatrix} a_{11}\,\mathbb{B} & a_{12}\,\mathbb{B} & \dots & a_{1q}\,\mathbb{B} \\ a_{21}\,\mathbb{B} & a_{22}\,\mathbb{B} & \dots & a_{2q}\,\mathbb{B} \\ \vdots & \vdots & & \vdots \\ a_{p1}\,\mathbb{B} & a_{p2}\,\mathbb{B} & \dots & a_{pq}\,\mathbb{B} \end{pmatrix} \qquad \text{(A.1)}$$

A.2 SOMME DE KRONECKER

La somme de Kronecker de deux matrices $\mathbb{C} = [c_{ij}]$, de dimensions (n,n), et $\mathbb{D} = [d_{ij}]$, de dimensions (m,m), est une matrice carrée de dimensions (mn, mn) donnée par :

$$\mathbb{C} \oplus \mathbb{D} = \mathbb{C} \otimes \mathbb{I}_m + \mathbb{I}_n \otimes \mathbb{D} \qquad \text{(A.2)}$$

où $\mathbb{I}_x$ est la matrice identité de dimensions (x,x).

A.3 APPLICATION à UN SYSTÈME MARKOVIEN MULTI-COMPOSANT

Considérons un système constitué de N composants stochastiquement indépendants tels que le comportement de chacun des composants est représenté par une chaîne de Markov en temps continu.

Soit :

- $\mathbb{P}_i(t)$ le vecteur des probabilités d'occupation des états de la chaîne de Markov décrivant le comportement du composant i du système et $\mathbb{\Lambda}_i$ la matrice des taux de transition associée.
- $\mathbb{P}_S(t)$ le vecteur des probabilités d'occupation des états de la chaîne de Markov décrivant le comportement du système et $\mathbb{\Lambda}_S$ la matrice des taux de transition associée.

Compte tenu de l'indépendance stochastique des comportements des différents composants du système, on obtient les relations suivantes :

$$\mathbb{P}_S(t) = \prod_{\otimes i=1}^{N} \mathbb{P}_i(t) = \mathbb{P}_1(t) \otimes \mathbb{P}_2(t) \otimes \ldots \otimes \mathbb{P}_N(t) \qquad \text{(A.3)}$$

$$\mathbb{\Lambda}_S = \sum_{\oplus i=1}^{N} \mathbb{\Lambda}_i = \mathbb{\Lambda}_1 \oplus \mathbb{\Lambda}_2 \oplus \ldots \oplus \mathbb{\Lambda}_N \qquad \text{(A.4)}$$

RéFéRENCES

[Adams 80] E.N. Adams, "Minimizing cost impact of software defects", IBM Research Division, Report RC 8228 (35669), Avril 1980.

[Akaike 74] H. Akaike, "A new look at the statistical model identification", *IEEE Trans. on Automatic Control,*. vol. AC-10, n° 19, Décembre 1974, pp. 716-723.

[Amoia 81] V. Amoia, G. De Micheli, M. Santomauro, "Computer-oriented formulation of transition-rate matrices via Kronecker algebra", *IEEE Trans. on Reliability,* vol. R-30, n° 2, Juin 1981.

[Anderson 81] T. Anderson, P.A. Lee, *Fault tolerance principles and practice*, Printice Hall Int., 1981.

[Arlat 88-a] J. Arlat, K. Kanoun, J.C. Laprie, "Dependability evaluation of software fault-tolerance", *Proc. 18th IEEE Int. Symp. Fault-Tolerant Computing,* Tokyo, Japon, Juin 1988, pp. 142-147; voir aussi *IEEE Trans. on Computers,* vol. 39, n° 4, Avril 1990, pp. 504-513.

[Arlat 88-b] J. Arlat, "Méthodes et outils pour l'évaluation de la sûreté de fonctionnement des systèmes informatiques tolérant les fautes", *Technique et Science Informatiques*, vol.7, n° 4, Juillet 1988, pp. 345-357.

[Arlat 89] J. Arlat, S. Bachmann, C. Beounes, J.E. Doucet, K. Kanoun, J.C. Laprie, J.Moreira de Souza, D. Powell, P. Spiesser, "SURF-2 : Specification de conception", *RR-LAAS* n° 89-098, Mars 1989.

[Ascher 78] H.Ascher, H.Feingold, "Application of Laplace's test to repairable system reliability", *Actes du 1er Colloque International de Fiabilité et de Maintenabilité*, Paris, Juin 1978, pp.219-225.

[Ascher 84] H.Ascher, H.Feingold, *Repairable Systems Reliability,* Lecture notes in statistics, vol. 7, New York and Basel: Marcel Dekker, inc, 1984.

[Balbo 87] G. Balbo, G. Chiola, G. Franceschinis, G. Molina Roet "On the efficient construction of the tangible reachability graph of Generalized Stochastic Petri Nets", *Proc. International Workshop on Petri Nets and Performance Models (PNPM 87)*, Madison, Wisconsin, Août 1987, pp. 136-145.

[Blakemore 89] A. Blakemore "The cost of elimination of vanishing markings from Generalized Stochastic Petri Nets", *Proc. International Workshop on Petri Nets and Performance Models (PNPM 89)*, Kyoto, Japon, Décembre 1989, pp. 85-92.

[Barlow 75] R.E.Barlow, F.Proschan, *Statistical Theory of Reliability and Life Testing,* Edition Holt, Rnehart and Winston, Inc, 1975.

[Bastani 80] F.B. Bastani, "An input based theory of software reliability and its application", Thèse de P.h.D, Dep. Elec. Eng. Comput. Sci., Université de Californie, USA, Berkley ,1980.

[Bastos 90] M.R. Bastos Martini, J. Moreira de Souza, "Reliability assessment of computer system design", *Annales des télécommunications*, tome 45, n° 11-12, pp. 642-647.

[Bauer 85] H.A. Bauer, L.M. Croxall, E.A. Davis, "The 5ESS switching system: system test, first-office application, and early field experience", *AT&T Technical Journal,* vol. 64, n° 6, 1985, pp. 1503-1522.

[Beounes 85] C. Beounes, J.C. Laprie, "Dependability Evaluation of Complex Computer Systems: Stochastic Petri Net Modeling", *Proc. 15th IEEE Int. Symp. Fault-Tolerant Computing,* Ann Arbor, Michigan, Juin 1985, pp. 364-369.

[Beyaert 81] B. Beyaert, G. Florin, P. Lonc, S. Natkin, "Evaluation of computer systems dependability using stochastic Petri nets", *Proc. 11th IEEE Int. Symp. Fault-Tolerant Computing,* Portland, Maine, Juin 1981, pp. 79-81.

[Bishop 89] P.G. Bishop, F.D. Pullen, "Error-masking: a source of failure dependency in multi-version programs", *Proc. 1st International Working Conf. Dependable Comput. Critical Appl.,* Santa Barbara, CA, Août 1989, pp. 25-32.

[Bobbio 86] A. Bobbio, K. Trivedi, "An aggregation technique for the transient analysis of stiff Markov chains", *IEEE Trans. on Computers*, vol. C-35, n° 9, Septembre 1986, pp. 803-814 .

[Bobbio 90] A. Bobbio, K. Trivedi, "Computing cumulative measures of stiff Markov chains using aggregation", *IEEE Trans. on Computers*, vol. C-39, n° 10, Octobre 1990, pp.803-814 .

[Brewer 78] J.W. Brewer "Kronecker products and matrix calculus in system theory", *IEEE Trans. on Circuits and Systems,* vol. CAS-25, n° 9, Septembre. 1978, pp. 772-781.

[Brocklehurst 90] S.Brocklehurst, P.Y Chan, B. Littlewood, J.Snell, "Recalibrating software reliability models", *IEEE Trans. on Software. Engineering*, vol. SE-16, n° 4, Avril 1990, pp. 458-470.

[Cha 86] S.D. Cha, "A Recovery Block Model and its Analysis", *Proc. SAFECOMP'86*, Sarlat, France, Octobre 1986, pp. 21-26.

[Chen 78] L. Chen, A. Avizienis, "N-version programming: a fault-tolerance approach to reliability of software operation", *Proc. 8th IEEE Int. Symp. on Fault Tolerant Computing*, Toulouse, France, Juin 1978, pp. 3-9.

[Clement 87] G.F. Clement, P.K. Giloth, "Evolution of of fault-tolerant switching systems in AT&T", dans *The evolution of Fault-Tolerant Computing,* A. Avizienis, H. Kopetz, J.C. Laprie, Editions Wien : Springer-Verlag, 1987, pp. 37-54

[Corazza 75] M.Corazza, *Techniques mathématiques de la fiabilité des systèmes*, Cepadues, 1975.

[Costes 78] A.Costes, C.Landrault, J.C.Laprie, "Reliability and Availability models for maintained systems featuring hardware failures and software faults", *IEEE Trans. on Computers,*` vol. C-27, Juin 1978, pp. 548-560.

[Costes 81] A. Costes, J.E. Doucet, C. Landrault, J.C. Laprie, "SURF: A Program for Dependability Evaluation of Complex Fault-tolerant Computing Systems", *Proc. 11th IEEE Int. Symp. Fault-Tolerant Computing,* Portland, Maine, Juin 1981, pp. 72-78.

[Cox 65] D.R. Cox, H.D Miller, *The theory of stochastic processes*, Methuen and Co., 1965.

[Cox 66] D.R. Cox, P.A W. Lewis, *The statistical analysis of series of events,* Londres, Chapman&Hall, 1966.

[Crow 77] L.H.Crow, "Confidence interval procedures for reliability growth analysis", Tech. report 1977, US Army Material Syst. Anal. Activity Aberdeen, MD, 1977.

[Csenki 89] A. Csenki, "Recovery block reliability analysis with failure clustering", *Proc. 1st International Working Conf. Dependable Comput. Critical Appl.,* Santa Barbara, CA, Août. 1989, pp. 33-42.

[Currit 86] P.A.Currit, M.Dyer, H.D.Mills, "Certifying the reliability of software", *IEEE Trans. on Software Engineering*, vol. SE-12, n° 1, Janvier 1986, pp. 3-11.

[Duane 64] J.T. Duane, "Learning curve approach to reliability monitoring", *IEEE Trans. on Aerospace,* vol. 2, 1964, pp. 563-566.

[Dugan 84] J.B. Dugan, K.S Trivedi, R.M Geist, V.F Nicolas, "Extended stochastic Petri nets: Applications and analysis", *Proc. 10th Int. Symp. on Computer Performance 84*, Editions E–Gelenbe, Paris, Décembre 1984, pp. 507-519.

[Ehrlich 90] W.K. Ehrlich, J.P. Stampfel, J.R. Wu, "Application of software reliability modeling to product quality and test process", *Proc. International Conference on Software Engineering.(ICSE),* Nice, Mai. 1990, pp. 108-116.

[Feller 68] W. Feller, *An introduction to probability theory and its applications,* New York: John Wiley & Sons, 3ième édition,1968.

[Finelli 91] G.B. Finelli, "NASA software characterisation experiments", *Reliability Engineering and System Safety,* vol.32, 1991, pp 155-169.

[Finkelstein 79] J.M.Finkelstein, "Starting and limiting values for reliability growth" , *IEEE Trans. on Reliability,* vol. R-28, n° 2, Juin 1979, pp. 111-113.

[Finkelstein 83] J.M.Finkelstein, "A logarithmic reliability growth model for single-mission systems", *IEEE Trans. on Reliability,* vol. R-32, n° 5, Décembre 1983, pp. 508-511.

[Florin 85] G. Florin, S. Natkin, "Les réseaux de Petri stochastiques", *Technique et Science Informatiques*, vol.4, n° 1, 1985, pp. 143-160.

[Fukishima 86] K.Fukishima, Y.Kishida, "Estimation of the bug curve using experimental regression analysis of office automation equipement software", Acte du *5ième Colloque international de Fiabilité et de Maintenabilité,* Biarritz, 6-10 Octobre 1986, pp. 87-91.

[Gaudoin 90] O.Gaudoin, "Outils statistiques pour l'évaluation de la fiabilité des logiciels", Thèse de Doctorat de l'Université Joseph Fourier- Grenoble 1, Décembre 1990.

[Geist 83] R. Geist, K. Trivedi, "Ultra-high reliability prediction for fault-tolerant computer systems", *IEEE Trans. on Computers,* vol. 32, n° 12, Décembre 1983, pp. 1118-1127.

[Goel 79] A.L.Goel, K.Okumoto, "Time-dependent Error-detection Rate Model for Software and other Performance Measures", *IEEE Trans. on Reliability*, vol. R-28, n°3, Août 1979, pp. 206-211.

[Goel 83] A.L.Goel, "A guidebook for Software reliability assessment", Data and Analysis Centre for Software, *Rome Air Development Centre (RADC)*, Rome, New York, Tech. Report RADC-TR-83-176, 1983.

[Goel 85] A.L.Goel, "Software Reliability Models : Assumptions, Limitations, and Applicability", *IEEE Trans. on Software Eng., Special issue on software Reliability*, vol. SE-11, n° 12, Décembre 1985, pp. 1411-1423.

[Goyal 86] A. Goyal, W.C. Carter, E. de Souza e Silva, S.S. Lavenberg, "The system availability estimator", *Proc. 16th IEEE Int. Symp. Fault-Tolerant Computing,* Vienna, Austria, June 1986, pp. 84-89.

[Gnedenko 69] B.V. Gnedenko, Y.K. Belyayev, A.D. Solovyev, *Mathematical methods of reliability theory,* New York: Academic Press, 1969.

[Gray 90] J.Gray, "A census of Tandem system availability between 1985 and 1990", *IEEE Trans. on Reliability,* vol. R-39, n° 4, Octobre 1990, pp. 409-418.

[Grnarov 80] A. Grnarov, J. Arlat, A. Avizienis, "On the Performance of Software Fault-Tolerance Strategies", *Proc. 10th IEEE Int. Symp. Fault-Tolerant Computing,* Kyoto, Japon, Octobre 1980, pp.251-253.

[Gross 84] D. Gross, D.R.Miller, "The Randomization Technique as a Modeling Tool and Solution Procedure for Transient Markov Processes", *Operations Research,* vol. 32, n° 2, 1984, pp. 343-361.

[Hecht 79] H. Hecht, "Fault-tolerant software", *IEEE Trans. on Reliability,* vol. R-28, Août. 1979, pp. 227-232.

[Howard 71] R. A. Howard, *Dynamic Probabilistic Systems, volume I: Markov models,* New York: John. Wiley and Sons Inc, 1971.

[Iyer 85] R.K.Iyer, D.J.Rossetti, "Effect of system workload on operating system reliability: a study on IBM 3081", *IEEE Trans. on Software Engineering, Special issue on software Reliability*, vol. SE-11, n° 12, Décembre 1985, pp. 1438-1448.

[Jelinski 72] Z.Jelinski, P.B.Moranda, "Software Reliability Research", *Proc. Statistical Methods for the Evaluation of Computer System Performance,* Academic Press, 1972, pp. 465-484.

[Jones 91] W. D. Jones, "Reliability models for very large software systems in industry" , *Proc. Int. Symp. on Software Reliability Engineering.,* Austin, Texas, Mai. 1991, pp. 35-42.

[Kaâniche 89-a] M. Kaâniche, K. Kanoun, S. Metge, "Analyse et évaluation de la croissance de fiabilité du logiciel E-10-5 Abonnés", Contrat Cnet n° 89 1B 007909245, *RR-LAAS* n° 89-328, Septembre 1989.

[Kaâniche 89-b] M.Kâaniche, K.Kanoun, S.Metge, "Proposition d'une collecte de données pour l'évaluation de la sûreté de fonctionnement d'un système informatique", Contrat Cnet n° 89 1B 007909245, *RR-LAAS* n° 89-329, Septembre 1989.

[Kaâniche 90-a] M. Kaâniche, K. Kanoun, S. Metge, "Utilisation du modèle hyperexponentiel pour le suivi de la validation d'un équipement de télécommunications", *Actes du 7ième Colloque International de Fiabilité et de Maintenabilité*, Juin 1990, Brest, France, pp.332-339.

[Kaâniche 90-b] M. Kaâniche, K. Kanoun, S. Metge, "Analyse des défaillances et suivi de la validation du logiciel d'un équipement de Télécommunications", *Annales des télécommunications*, tome 45, n° 11-12, Novembre-Décembre 1990, pp. 657-670.

[Kalbfleisch 80] J.D Kalbfleisch, R.L. Prentice, *The statistical analysis of failure time data,* John&Wiley , 1980.

[Kanoun 85] K.Kanoun, J.C.Laprie, "Modeling Software Reliability and Availability from Development-validation up to Operation", *RR-LAAS* n° 85-042, Mars 1985, révision Août 1985.

[Kanoun 87] K.Kanoun, T.Sabourin, "Software dependability of a telephone switching system", *Proc. 17th IEEE Int. Symp. on Fault Tolerant Computing,* Pittsburgh, Juin 1987, pp. 236-241.

[Kanoun 88] K.Kanoun, J.C.Laprie, T.Sabourin, "A method for software reliability analysis and assessment", Actes du : *Le Génie Logiciel & ses Applications*, Toulouse, 5-9 Décembre 1988, pp. 859-878.

[Kanoun 89] K.Kanoun, "Croissance de la sûreté de fonctionnement des logiciels : caractérisation-modélisation-evaluation", Thèse de Doctorat d'état de l'Institut National Polytechnique de Toulouse, Septembre 1989.

[Kanoun 91-a] K.Kanoun, M. Kaâniche, J.C. Laprie, S. Metge, "Méthodes et modèles mis en œuvre dans le logiciel SoRel Software Reliability", *RR-LAAS* n° 91-212, Janvier 1991.

[Kanoun 91-b] K.Kanoun, M.R Bastos Martini, J.M. de Souza, "A method for software reliability and prediction application to the TROPICO-R switching system", *IEEE Trans. on Software Engineering,* vol. SE-17, n° 4, Avril 1991, pp. 334-344.

[Kanoun 91-c] K.Kanoun, M. Kaâniche, C. Beounes, J.C. Laprie, J. Arlat, "Reliability growth of fault tolerant software", Contrat CEE PDCS n° 3092, vol. 2, Mai 1991.

[Kanoun 91-d] K.Kanoun, J.C. Laprie, "The role of trend analysis in software development and validation", *Proc. SAFECOMP'91,* Trondheim, Norvège, Octobre 1991, pp. 169- 174.

[Keiller 91] P.A.Keiller, D.R.Miller, "On the use and the performance of software reliability growth models", *Reliability Engineering and System Safety*, Elsevier Science Publishers, Angleterre, vol 32, 1991, pp. 95- 117.

[Kemeny 59] J.G Kemeny, J.L. Snell, *Finite Markov Chains.* Princetown: D. Van Nostrand, 1959.

[Khoshgoftaar 91] T.M .Khoshgoftaar, T.G.Woodcock, "Software reliability model selection: a case study", *Proc. Int. Symp. on Software Reliability Engineering (ISSRE),* Austin, Texas, Mai 1991, pp. 183-191.

[Knafl 91] G.Knafl, J. Sacks, "Poisson Processes with nearly constant failure intensity", *Proc. Int. Symp. on Software Reliability Engineering (ISSRE),* Austin, Texas, Mai 1991, pp. 60-66.

[Laprie 75] J.C.Laprie, "Prévision de la sûreté de fonctionnement et architectures de structures numériques temps réel réparables", Thèse de Doctorat d'Etat, UPS, Toulouse, Juin 1975.

[Laprie 83-a] J.C.Laprie, "Trustable evaluation of computer systems dependability", *Proc. Int. Workshop. on Applied Mathematics and Performance/Reliability Models of Computer/Communication Systems*, Pise, Italie, Septembre1983, pp. 341- 360.

[Laprie 83-b] J.C.Laprie, "Evaluation de la sûreté de fonctionnement des logiciels en opération", *Technique et Science Informatiques*, vol.2, n° 4, Décembre 1983, pp. 233-247.

[Laprie 84-a] J.C.Laprie, "Models for software availability evaluation", *2nd National Workshop on Fault Tolerant Computing Systems (NWFTCS II),* Melbourne, Australie, Fevrier 1984.

[Laprie 84-b] J.C.Laprie, "Dependability modeling and evaluation of software-and-hardware Systems", *Proc. 2nd GI-NTG-GMR Conf. on Fault Tolerant Computing,* Bonn, Allemagne Fédérale, Septembre 1984, pp. 202-215.

[Laprie 84-c] J.C.Laprie, "Dependability Evaluation of Software Systems in Operation", *IEEE Trans. on Software. Engineering.* Vol. SE-10, n° 6, Novembre 1984, pp. 701-714.

[Laprie 87] J.C. Laprie, J. Arlat, C. Beounes, K. Kanoun, C. Hourtolle, "Hardware and software fault tolerance: Definition and analysis of architectural solutions", *Proc. 17th IEEE Int. Symp. on Fault Tolerant Computing*, Pittsburgh, PA, Juillet 1987, pp.116-121.

[Laprie 88] J.C.Laprie, "Vers une théorie de la fiabilité du X-iel", *Technique et Science Informatiques*, vol.7, n° 3, 1988, pp. 315-329.

[Laprie 89] J.C. Laprie, "Hardware-and-Software dependability evaluation", *Proc. IFIP 11th World Computer Congress*, San Fransisco, USA, Août 1989, pp. 109-114.

[Laprie 90-a] J.C. Laprie, K. Kanoun, M. Kaâniche, C. Beounes, "The transformation approach to modeling and evaluation of the reliability and availability growth of systems", *Proc. 20th IEEE Int. Symp. on Fault Tolerant Computing,* Newcastle, UK, Juin 1990, pp.364-371.

[Laprie 90-b] J.C. Laprie, J. Arlat, C. Beounes, K. Kanoun "Definition and analysis of hardware- and software-fault-tolerant architectures", *IEEE Computer,* vol. 23, n° 7, Juillet 1990, pp. 39-51.

[Laprie 91-a] J.C. Laprie, K. Kanoun, C. Beounes, M. Kaâniche, "The KAT (Knowledge-Action-Transformation) approach to the modeling and evaluation of reliability and availability growth", *IEEE Trans. on Software Engineering,* vol. SE-17, n° 4, Avril 1991, pp. 370-382.

[Laprie 91-b] J.C. Laprie, K. Kanoun, "X-ware reliability and availability modeling", *RR-LAAS* n°91012, Contrat CEE PDCS n°3092, vol. 2, Mai 1991.

[Laprie 91-c] J.C. Laprie, "Dependability: Basic Concepts and Terminology", Editions : Springer-Verlag, 1991, *paru en cinq langues*.

[Levendel 89] Y. Levendel, "Defects and reliability analysis of large software systems: field experience", *Proc. 19th IEEE Int. Symp. on Fault Tolerant Computing,* Chicago, USA, Juin 1989, pp.238-244.

[Levendel 90] Y. Levendel, "The software quality improvement process: when to stop testing", Actes du : *Le Génie Logiciel & ses Applications*, Toulouse, Décembre 1990, pp. 729-748.

[Lawless 82] J.F. Lawless, *Statistical models and methods for lifetime data,* Wiley series in probability and mathematical statistics, 1982.

[Littlewood 73] B.Littlewood, J.L.Verral, "A Bayesian Reliability Growth Model for Computer Software" *J. Royal Stat. Soc.*, C(App. stat.), 22, 1973, pp. 332-336.

[Littlewood 79] B. Littlewood, "Software reliability model for modular program structure", *IEEE Trans. on Reliability,* vol. R-28, n° 3, Août. 1979, pp. 241-246.

[Littlewood 80] B. Littlewood, "Theories of software reliability", *IEEE Trans. on Software Engineering,* vol. SE-6, n° 5, Septembre. 1980, pp. 489-500.

[Littlewood 81] B. Littlewood, "Stochastic reliability growth: a model for for fault-removal in computer programs and hardware designs", *IEEE Trans. on Reliability*, vol. R-30, n° 4, Octobre. 1981, pp. 313-320.

[Littlewood 89] B. Littlewood, *"Forecasting Software reliability"*, Lecture notes in Computer Science, Springer-Verlag, pp 140-209.

[Marie 86] R. Marie, B. Sericola, "Distribution du temps total de séjour dans un sous-ensemble d'états transitoires d'un processus markovien homogène à espace d'état fini", *RR-IRISA*, n° 585. Novembre 1986.

[Marsan 84] A. Marsan, G. Balbo, G. Conte, "A class of Generalized Stochastic Petri Nets for the performance analysis of multiprocessor systems", *ACM Trans. on Computers*, vol2, n° 2, Mai 1984, pp. 93-122 .

[Marsan 86] A. Marsan, G. Balbo, G. Chiola, G. Conte, "On Petri nets with deterministic and exponential transition firing time", *7th European Workshop on Application and Theory of Petri Nets*, Oxford, Juin 1986.

[Marsan 87] A. Marsan, G. Balbo, G. Chiola, G. Conte, "Generalized Stochastic Petri Nets revisited: Random switches and priorities", *Proc. International Workshop on Petri Nets and Performance Models (PNPM 87)*, Madison, Wisconsin, Août 1987, pp. 44-53.

[Marsan 91] A. Marsan, G. Balbo, G. Chiola, G. Conte, S. Donatelli,G. Franceschinis, "An introduction to Generalized Stochastic Petri Nets", *Microelectronics and Reliability*, Pergamon Press, vol 31, n° 4, pp.699-725.

[Mazars 81] N. Mazars, "De l'indépendance et des dépendances stochastiques en théorie de la fiabilité", Thèse de Doctorat, Université Paul Sabatier, Toulouse, Juin 1981.

[Mello 86] E.B. Mello, "Software architecture of a TROPICO-R telephone exchange", *Int. Conf. on Software eng. for Telecom. Switching Systems,* Londres, Avril 1986, pp. 40-45.

[Mellor 87] P.Mellor, "Software Reliability modelling : the state of the art", *Information and Software Technology*, vol. 29, n° 2, Mars 1987, pp. 81-98.

[Metge 90] S. Metge, "Analyse et évaluation de la fiabilité de deux logiciels de télécommunication", mémoire CNAM, Conservatoire National des Arts et Métiers, Toulouse, Mai 1990.

[Miller 86] D.R. Miller, "Exponential order statistic models of software reliability growth", *IEEE Trans. on Software Engineering*, vol. SE-12, n° 1, Janvier 1986, pp. 12-24.

[Molloy 82] M. Molloy, "Performance analysis using Stochastic Petri nets", *IEEE Trans on Computers*, vol 39, n° 9, Septembre 1982, pp. 913-917 .

[Moranda 75] P.Moranda, "Predictions of software reliability during debugging", *Proc. Ann. Reliability and Maintainability Symposium*, Washington, D.C., Janvier 1975, pp. 327-332.

[Moreira 86] J. Moreira de Souza, A.C Lavelha, M.R. Bastos Martini, "Evaluation de la qualité de service d'un système à dégradation progressive", *Actes du 5ième Colloque International de Fiabilité et de Maintenabilité*, Octobre 1986, Biarritz, France.

[Muluzzani 85] M. Mulazzani, "Reliability Versus Safety", *Proc. SAFECOMP'85*, Como, Italie, Octobre 1985, pp. 141-146.

[Musa 79] J.D. Musa, "Validity of execution time theory of software reliability", *IEEE Trans. on Reliability*, vol. R-28, n° 3, Août 1979, pp. 181-191.

[Musa 84] J.D. Musa, K. Okumoto, "A logarithmic Poisson execution time model for software reliability measurement", *Proc. Compsac'84*, Chicago, 1984, pp. 230-238.

[Musa 87] J.D. Musa, A. Ianino, K. Okumoto, *Software reliability: Measurement, Prediction, Application*, McGraw-Hill, Singapour, 1987.

[Natkin 80] S. Natkin, "Les réseaux de Petri stochastiques", Thèse de Docteur ingénieur CNAM, Paris, Juin 1980.

[Ohba 82] M.Ohba, S.Yamada, K.Tekada, S.Osaki, " S-Shaped software reliability growth curve : how good is it?", *Proc. COMPSAC 82*, Chicago, Illinois, 8-12 Novembre 1982, pp. 38-44.

[Ohba 84] M. Ohba, "Software reliability analysis models", *IBM Journal of Res. and Dev.* vol 28, n° 4, juillet 1984, pp. 428-443.

[Padgett 85] W.J. Padgett, J.D. Spurrier, "On discrete failure models", *IEEE Trans. on Reliability,* vol. R-34, n° 3, Août 1985, pp. 253-256.

[Parnas 72] D. L. Parnas, "On the criteria to be used in decomposing systems into modules", *Communications of the ACM,* vol. 15, n° 12, Décembre 1972, pp. 1053-1058.

[Pagès 80] A. Pagès, M. Gondran, *Fiabilité des systèmes,* Eyrolles, Paris, 1980.

[Pucci 90] G. Pucci, "On the modeling and testing of recovery block structures", *Proc. 20th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-20)*, Newcastle, UK, Juin 1990, pp. 356-363.

[Rao 78] S. S. Rao, *Optimization Theory and Applications*, *Proc. COMPSAC 80*, Wiley Eastern Limited, 1978.

[Ramamoorthy 80] C. V. Ramamoorthy, F.B. Bastani, "Modeling of the software reliability growth process", *Proc. COMPSAC 80*,Chicago, Illinois, 1980, pp. 161-169.

[Ramamoorthy 82] C.V.Ramamoorthy, F.B.Bastani, "Software Reliability - Status and Perspectives" *IEEE Trans. on Software.Engineering*, vol. SE-8, n° 4, Juillet 1982, pp. 354-371.

[Randell 75] B. Randell, "System Structure for Software Fault Tolerance", *IEEE Trans. on Software Engineering.*, vol. SE-1, n° 2, June 1975, pp.220-232.

[Roux 87] J.L. Roux, G. Juanole, "Functional and performance analysis using extended time Petri nets", *Proc. International Workshop on Petri Nets and Performance Models (PNPM 87)*, Madison, Wisconsin, Août 1987, pp. 14-23.

[Reibman 89] A. Reibman, R. Smith, K. Trivedi, "Markov and Markov reward model transient analysis: An overview of numerical approaches", *European Journal of Operation Research,* vol. 40, n° 4, 1989, pp. 257-267.

[Sabourin 87] T.Sabourin, "Evaluation de la fiabilité du logiciel d'un autocommutateur téléphonique", Thèse de Doctorat, Institut National Polytechnique de Toulouse, INPT, Octobre 1987.

[Saglietti 86] F. Saglietti, W. Ehrenberger, "Software diversity — some considerations about its benefits and its limitations", *Proc. SAFECOMP'86*, Sarlat, France, Octobre 1986, pp. 27-34.

[Salvia 82] A.A. Salvia, R.C. Bollinger, "On discrete hazard functions", *IEEE Trans. on Reliability,* vol. R-31, n°. 5, Décembre 1982, pp. 458-459.

[Saporta 78] G. Saporta, *Théories et méthodes de la statistique*. Paris: Technip, 1978.

[Scott 87] R.K. Scott, J.W. Gault, D.F. McAllister, "Fault-Tolerant Software Reliability Modeling", *IEEE Trans. on Software Engineering*, vol. SE-13, Mai 1987,.pp. 582-592.

[Serfling 78] R.J. Serfling, "Some elementary results on Poisson approximation in a sequence of Bernouilli trials", *SIAM Review.*, vol. 20, n° 3, Juillet 1978, pp. 567-579.

[Siegrist 88] K. Siegrist, "Reliability of systems with Markov transfer of control", *IEEE Trans. on Software Engineering,* vol. SE-14, n° 7, Juillet 1988, pp. 1049-1053.

[Stark 87] G.E. Stark, "Dependability evaluation of integrated hardware/software systems", *IEEE Trans. on Reliability,* vol. R-36, n°. 4, Octobre. 1987, pp. 440-444.

[Sumita 86] U. Sumita, Y. Masuda, "Analysis of software availibility/reliability under the influence of hardware failures", *IEEE Trans. on Software. Engineering*, vol. SE-12, n° 1, Janvier 1986, pp. 32-41.

[Stewart 85] W.J. Stewart, A. Goyal, "Matrix methods in large dependability models", IBM Research Report RC-11485, YorktownHeights, NY, Novembre 1985.

[Tso 86] K.S. Tso, A. Avizienis, J.P.J. Kelly, "Error Recovery in Multi-Version Software", *Proc. SAFECOMP'86*, Sarlat, France, Octobre 1986, pp. 35-41.

[Tohma 89] Y. Tohma, K. Tokunaga, S. Nagase, Y. Murata, "Structural approach to the estimation of the number of residual software faults based on the hyper-geometric distribution", *IEEE Trans. on Soft. Engineering*, vol. SE-15, Mars 1989,.pp. 345-355.

[Tohma 91] Y. Tohma, H. Yamano, M. Ohba, R. Jacoby, "Parameter estimation of the hyper-geometric distribution for real test/debug data", *Proc. Int. Symp. on Software Reliability Engineering (ISSRE),* Austin, Texas, Mai 1991, pp. 28-34.

[Toy 85] W.N Toy, "Modular redundancy concept, problems and solutions", *EPRI Seminar: Digital Control and Fault-Tolerant.Computer Technology*, Scottsdale, Arizona, Avril 1985.

[Wallace 84] J.J. Wallace, W.W. Barnes, "Designing for ultrahigh availability: the Unix RTR operating system", *Computer*, Août. 1984, pp. 31-39.

[Yamada 83] S.Yamada, S.Osaki, "Reliability Growth Modeling for Software Error Detection", *IEEE Trans. on Reliability,* vol. R-32, n° 5, 1983, pp. 475-478.

[Yamada 84] S. Yamada, S. Osaki, H. Narihisa, "Software reliability growth modeling with number of test runs", *Trans. IECE Japan,* vol. E-67, n° 2, Février 1984, pp. 79-83.

Thèse de Monsieur Mohamed KAANICHE

"Modèle hyperexponentiel en temps continu et en temps discret pour l'évaluation de la croissance de la sûreté de fonctionnement"

RéSUMé

Ce mémoire présente des travaux et des résultats, aussi bien théoriques que pratiques, concernant la modélisation et l'évaluation de la croissance de fiabilité et de la croissance de disponibilité des systèmes informatiques. Nous considérons deux types de représentation du comportement des systèmes : d'abord, en fonction du temps, et ensuite en fonction du nombre d'exécutions effectuées.

Les travaux présentés dans ce mémoire s'articulent autour de deux modèles de croissance de fiabilité : le modèle hyperexponentiel en temps continu et le modèle hyperexponentiel en temps discret. Pour chacun de ces deux modèles, nous étudions d'abord, le cas d'un système mono-composant, puis nous considérons le cas d'un système multi-composant qui est tel que la croissance de fiabilité de chacun de ses composants est représentée par un modèle hyperexponentiel. Le modèle hyperexponentiel en temps discret est également utilisé pour prendre en compte certaines caractéristiques de l'environnement d'utilisation du logiciel dans l'évaluation de son comportement tel qu'il est perçu dans le temps par ses utilisateurs dans chacun des environnements dans lequel il est mis en œuvre.

Mots clés : croissance de fiabilité, croissance de disponibilité, temps continu, temps discret, systèmes mono-composant, systèmes multi-composant, chaînes de Markov, réseaux de Petri stochastiques

"Continuous time and discrete time hyperexponential model for dependability growth modeling"

ABSRACT

This dissertation presents theoretical and practical results on computer system reliability and availability growth modeling. Two kinds of system behavior characterizations are considered: first, with respect to time, and second, with respect to the number of executions performed.

The dissertation is centered on two reliability growth models: the continuous time hyperexponential model and the discrete time hyperexponential model. For each of these models, we consider first, single component systems, and second, multi-component systems such that the reliability growth of each component of the considered system is characterised by the hyperexponential model. The discrete time hyperexponential model is also used to account for some characteristics of the software's environment to evaluate dependability measures of the software that are particular to the environment in which it is executed.

Key words: reliability growth, availability growth, continuous time, discrete time, single component systems, multicomponent systems, Markov chains, stochastic Petri nets.