# Évaluation d'indicateurs de sûreté de fonctionnement d'architectures de contrôle-commande dans un contexte dynamique et incertain

Grâce Boyer

UNIVERSITÉ DE LORRAINE

École Doctorale IAEM Lorraine

# Évaluation d'indicateurs de sûreté de fonctionnement d'architectures de contrôle-commande dans un contexte dynamique et incertain

## THÈSE

Soutenue le 25/02/2021

pour l'obtention du

**Doctorat de l'Université de Lorraine**
(Mention Automatique, Génie Informatique et Traitement du Signal)

par

**Grâce BOYER**

**Composition du jury :**

*Rapporteurs :*

| | |
|---|---|
| Eric ZAMAI | Professeur, AMPERE, INSA Lyon |
| Zineb SIMEU-ABAZI | Professeure, G-SCOP, Université Grenoble Alpes |

*Examinateurs :*

| | |
|---|---|
| Frédéric KRATZ | Professeur, PRISME, INSA Centre Val de Loire |
| Jean-Marc THIRIET | Professeur, GIPSA-Lab, Université Grenoble Alpes |

*Directeurs de thèse :*

| | |
|---|---|
| Jean-François PÉTIN | Professeur, CRAN, Université de Lorraine |
| Nicolae BRÎNZEI | Maître de Conférences, CRAN, Université de Lorraine |

*Encadrant :*

| | |
|---|---|
| Jacques CAMERINI | Schneider Electric France |

*Invités :*

| | |
|---|---|
| Moulaye NDIAYE | Schneider Electric France |
| Marcel CHEVALIER | Schneider Electric France |

CRAN cnrs Schneider Electric

# Table des matières

# Glossaire

| | |
|---|---|
| AdD | Arbre de Défaillances |
| API | Automate Programmable Industriel (PLC en anglais) |
| ASH | Automate Stochastique Hybride |
| BDMP | Boolean logic Driven Markov Process |
| CdM | Chaîne de Markov |
| CPN | Coloured Petri Nets selon le formalisme de Jensen (Jensen & Kristensen, 2009) |
| CPU | Processeur (*Central Processing Unit*) |
| DCC | Défaillance de Cause Commune |
| E/S | Entrées/Sorties (pour les modules dédiés à la lecture d'entrée et/ou de sorties de variables) |
| ERP | Enterprise Ressource Planning |
| FMDS | Fiabilité, Maintenabilité, Disponibilité, Sécurité |
| I/O | Input/Output (equivalent anglaise de E/S) |
| MES | Manufacturing Execution System |
| MTBF | Mean Time Between Failures |
| MTTF | Mean Time To Failure |
| MTTR | Mean Time To Repair |
| MDT | Mean Down Time |
| MUT | Mean Up Time |
| PFD | Process Flow Diagram |
| PLC | Programmable Logic Controller |
| RAMS | Reliability, Availability, Maintainability, Safety |
| RBD | Reliability Block Diagram |
| RdP | Réseaux de Petri |
| SCADA | Système de commande et d'acquisition des données (*Supervisory Control And Data Acquisition* en anglais) |
| SdF | Sûreté de Fonctionnement |
| SI | Système Intégrateur (ingénieur en avant-vente responsable de la conception des architectures de contrôle-commande) |
| SIL | Safety Integrity Level |
| SIS | Système Instrumenté de Sécurité |
| UML | Unified Modeling Language |

# Liste des figures

# Liste des tableaux

# Introduction générale

Les travaux de recherche développés dans cette thèse ont été réalisés dans le cadre d'une convention CIFRE (Convention Industrielle de Formation par la Recherche) menée entre le Centre de Recherche en Automatique de Nancy (CRAN – UMR n°7039 du CNRS et de l'Université de Lorraine) et l'entité "Industrial Automation" de l'entreprise Schneider Electric France.

Une des activités de Schneider Electric est de fournir des offres en automatismes industriels. En phase d'avant-vente, l'entreprise, à travers les systèmes intégrateurs (SI) qui constituent la force de vente, propose à ses clients industriels des systèmes du type architecture de contrôle-commande permettant de faire fonctionner un procédé industriel, en adéquation avec les besoins du milieu dans lequel le système va évoluer.

La proposition commerciale qui est faite par le SI doit également respecter les exigences formulées par le client sur les performances de l'architecture candidate au projet industriel, notamment les performances entrant dans le champ de la sûreté de fonctionnement (SdF). Atteindre les performances fiabilistes demandées dans le projet, c'est assurer de proposer une offre de qualité avec un système robuste, tolérant aux fautes et assurant une continuité de fonctionnement dans la réalisation du procédé industriel (Zio, 2009).

Il est donc primordial, en plus des autres performances comme celles relatives aux temps de réponse par exemple, de pouvoir évaluer les performances de SdF d'une architecture de contrôle-commande en phase d'avant-vente. L'évaluation fournira un apport non négligeable au SI pour argumenter sur ses choix techniques dans l'élaboration de l'architecture, et par conséquent d'augmenter le niveau de confiance dans la proposition commerciale qui est faite, et de gagner des parts de marché ou des appels d'offres.

Actuellement, pour garantir le respect des exigences du projet en termes de performances de SdF, les SI rencontrent des difficultés qui complexifient la mise en œuvre de l'évaluation de ces performances, ainsi que la validation des résultats. Ceci est principalement causé par :

- Une connaissance limitée des architectures de contrôle-commande que nous pouvons proposer en avant-vente. À ce stade, ce qui est connu de l'architecture se restreint à une traduction des missions du système dans le fonctionnement d'un procédé industriel en un ensemble d'équipements liés par un réseau de communication (automates programmables industriels API, des modules d'entrées/sorties E/S, des équipements terminaux comme des variateurs de vitesse, etc.).
- Des ressources limitées pour réaliser l'évaluation des performances de SdF sur les différentes propositions commerciales. L'évaluation doit se faire dans un laps de temps relativement court alors même que ce dernier requiert une expertise en SdF. Elle peut également se faire sur un grand nombre d'architectures pouvant être proposé en tant que solution pour une même offre, ce qui incrémente le nombre de fois où nous devons réaliser l'évaluation.

Pour pallier partiellement ces difficultés, les pratiques actuelles reposent sur un surdimensionnement du système. Le recours à la redondance sur les équipements de l'architecture de contrôle-commande reste la solution la plus adoptée, mais n'est pas forcément la réponse la plus adaptée car la redondance peut ne pas être appliquée sur les équipements les plus critiques. L'impact sur les performances de SdF du système peut être donc insignifiant, mais le coût financier demandé sera élevé, ce qui augmente le risque de perdre l'appel d'offres et des parts de marché.

Au vu de ce contexte, la thèse s'inscrit dans une optique de recherche orientée vers l'industrie et aura pour objectif de proposer un outil d'aide à la décision à destination principalement de la force de vente de Schneider Electric que sont les SI. Cet outil permettra d'élaborer l'architecture de contrôle-commande la plus fiable et la plus robuste tout en respectant les exigences qui ont été exprimées par le client du projet industriel. Il devra donc permettre l'évaluation des performances fiabilistes des architectures et compléter ainsi l'estimation des performances temporelles développée dans les travaux de Moulaye Ndiaye (Ndiaye, 2017). L'outil devra intégrer les contraintes d'ingénierie en avant-vente énoncées précédemment, à savoir permettre une évaluation de multiples architectures candidates à une même offre, dans un laps de temps court, et demander peu ou pas d'expertise en SdF de la part des SI.

Cette problématique peut se décliner en trois questions scientifiques auquel le mémoire va s'attacher à apporter une réponse :

1. Comment définir formellement le système à partir d'une description incomplète de l'architecture de contrôle-commande et prendre en compte tous les aspects nécessaires pour la mise en place d'une évaluation des performances de SdF en avant-vente ?
2. Quels sont les indicateurs de SdF pertinents qui vont nous permettre d'évaluer simplement les performances de SdF de l'architecture ?
3. Comment mettre en place l'évaluation des performances de SdF d'une architecture de contrôle-commande à partir de sa définition et des indicateurs choisis ?

La réponse apportée à ces trois questions se traduit par une proposition d'un cadre formel de modélisation et d'évaluation des indicateurs de SdF d'une architecture de contrôle-commande. Compte tenu du contexte particulier de l'avant-vente, ce cadre privilégiera la recherche de généricité et le recours à des bibliothèques de modèles paramétrables et instanciables afin de réduire le plus possible le temps consacré à l'élaboration des modèles sur de multiples architectures. Enfin, l'évaluation des architectures devra tenir compte de leurs environnements opérationnels (contexte d'usage, agressivité du milieu dans lequel elles opèrent…) et de leur durée d'exploitation (en tenant compte notamment des phénomènes de vieillissement).

Le premier chapitre introduit le contexte et les objectifs industriels de la thèse. Nous posons les définitions nécessaires à la compréhension du sujet et des éléments à prendre en compte pour répondre à la problématique. Nous précisons par exemple la définition d'une architecture de contrôle-commande, et la définition des performances SdF demandés dans les projets industriels. En complément, nous détaillons les critères à respecter pour la mise en place d'une évaluation des performances SdF d'une architecture composée d'équipements réparables et sujette à

reconfiguration suite à l'occurrence d'aléas ou de pannes, qui justifieront notamment le recours à des modèles dynamiques.

Le deuxième chapitre porte sur la spécification formelle du besoin exprimé par Schneider Electric. Après une analyse des pratiques actuelles de Schneider Electric en termes d'évaluation des performances de SdF, le besoin industriel est formalisé au travers de représentations de type UML (Object Management Group, 2015). Les informations relatives aux éléments des architectures de contrôle-commande sont décrites à l'aide de diagrammes de classes incluant les paramètres dysfonctionnels des équipements ; ces informations serviront à alimenter le processus de génération des modèles décrit au chapitre suivant. Les états dysfonctionnels et/ou dégradés des équipements ainsi que les liens de dépendance dysfonctionnels entre les équipements sont décrits respectivement à l'aide de diagrammes d'états et de séquences. L'ensemble de ces modèles constitue le point d'entrée de la démarche de modélisation et d'évaluation des indicateurs de SdF présentée au chapitre suivant.

Le troisième chapitre est consacrée à l'état de l'art relatif aux trois questions que nous avons identifiées précédemment.

En termes de modélisation, il existe une littérature abondante sur le sujet qui peut se décliner en deux grandes familles : les modèles dysfonctionnels statiques et les modèles dynamiques à états et à paramètres stochastiques. Compte tenu des besoins exprimés aux chapitres 1 et 2, les modèles dynamiques sont ceux retenus dans la suite de nos travaux. Les critères ayant conduit au choix d'un formalisme parmi ceux existants dans la littérature à base de chaîne de Markov (Behrends, 2000), de réseaux de Petri (RdP) ou d'automates à états finis (Cassandras & Lafortune, 2009), sont relatifs à la modularité, la hiérarchie ainsi que les possibilités offertes en termes de paramétrage et d'instanciation des modèles. Des formalismes tels que les réseaux de Petri hiérarchiques (Fehling, 1991), stochastiques (Marsan, Balbo, Conte, Donatelli, & Franceschinis, 1998) et colorés (Peterson, 1980) (Jensen, 1981) ou les automates temporisés (Alur & Dill, 1994) répondent à ces critères et permettent d'envisager la génération automatique ou semi-automatique des modèles à partir d'une description informelle du système à modéliser (Ndiaye, 2017) (Chevalier, Vinuesa, Buchsbaum, Folleau, & Thomas, 2020). Dans la continuité des travaux de, notre choix s'est porté sur une classe particulière des réseaux de Petri colorés et temporisés définie par (Jensen & Kristensen, 2015).

En termes d'indicateurs SdF, la littérature scientifique nous donne encore ici un panel de possibilités, entre les quatre grandes composantes de la SdF, la Disponibilité, la Fiabilité, la Maintenabilité, la Sécurité et leurs métriques (Laprie J. , 1992). Dans le contexte d'avant-vente, ces indicateurs classiques pourront avantageusement être complétés par facteurs d'importance (Kuo & Zhu, 2012) qui guideront les ingénieurs vers les équipements les plus critiques. Parmi les différents facteurs d'importance définis dans la littérature, ceux définis par (Birnbaum, 1968) et leur différentes déclinaisons répondent le mieux à notre besoin industriel et sont les plus facilement adaptables à notre cadre de modélisation.

Enfin la dernière partie porte sur la méthode d'évaluation. Si les calculs analytiques sont une possibilité, nous avons choisi de coupler le modèle à une approche par simulation de Monte Carlo (Zio, 2013) (Papadopoulos & Yeung, 2001) pour résoudre le problème lié au contexte incertain et aux ressources limitées en avant-vente et la taille complexe des systèmes à évaluer.

Le quatrième chapitre constitue le cœur de la contribution de thèse autour de la proposition d'un cadre formel de modélisation et d'évaluation des indicateurs de SdF.

Dans une première partie, le chapitre expose la proposition méthodologique permettant la construction automatique d'un modèle CPN d'architecture de contrôle-commande par instanciation d'un ensemble de composants élémentaires (API, boîtier d'entrées/sorties, SCADA, poste client, réseau, …). L'originalité de la proposition tient dans l'utilisation d'un jeton coloré embarquant l'identification des modèles élémentaires à instancier ainsi que leurs paramètres. Son intérêt réside dans la construction automatique des modèles CPN qui se traduit par des gains de temps très importants pour l'évaluation de multiples solutions d'architecture.

La seconde partie du chapitre est consacrée à la définition d'observateurs génériques couvrant les besoins exprimés par les concepteurs d'architecture et qui couvrent les indicateurs classiques de SdF (fiabilité, disponibilité notamment) mais aussi des facteurs d'importance permettant d'orienter les choix des concepteurs en phase d'élaboration d'une proposition d'architecture en réponse à un appel d'offre. Des algorithmes exploitant la définition de ces indicateurs dans le cadre de simulations de Monte-Carlo sont proposés.

Le cinquième chapitre montre l'application de l'évaluation des performances de SdF d'une architecture, que nous avons développé dans le chapitre précédent, dans un cadre industriel.

La première partie du chapitre sera consacrée à l'implémentation du cadre de modélisation défini et de l'évaluation des indicateurs dans un prototype d'outil d'aide à la décision en conception en avant-vente d'architectures Schneider Electric dédié à la SdF. L'interface entre l'outil, la génération du modèle en adéquation avec les caractéristiques de l'architecture à évaluer, et la récupération des données brutes de la simulation pour estimer les indicateurs SdF demandés ainsi que leur affichage dans l'outil sont décrits dans cette partie.

La deuxième partie présente un cas d'étude industriel, celui d'une cimenterie, pour illustrer l'ensemble de la démarche de modélisation et d'évaluation des indicateurs. Cette partie montre également comment les résultats affichés sur les performances fiabilistes permettent de modifier et d'affiner l'architecture de contrôle-commande proposée pour le projet afin de respecter facilement et de façon pertinente les exigences posées par le porteur du projet industriel.

Le mémoire se conclue par une présentation des principales contributions de nos travaux de recherche ainsi que des propositions de perspectives afin de pouvoir établir une extension du cadre formel de l'évaluation et qui s'inscrit dans la continuité des travaux.

# Chapitre 1
# Contexte et objectif de la thèse

## 1.1. Introduction

Comme il a été mentionné dans l'introduction générale, les travaux de recherche présentés dans ce mémoire porte sur l'évaluation des performances de sûreté de fonctionnement (SdF) des architectures de contrôle-commande de Schneider Electric, dans un contexte d'avant-vente.

Ce chapitre introduit le contexte de ces travaux ainsi que les notions nécessaires à la compréhension du sujet de thèse et des besoins qui en résultent pour répondre à la fois à une demande industrielle et à un problème scientifique.

Ici, nous définissons dans un premier temps l'objet de l'étude à savoir les architectures de contrôle-commande. Dans un deuxième temps, nous présentons les aspects de SdF liés à ces architectures et quels sont les bénéfices attendus par rapport à la mise en place d'une évaluation sur les performances de SdF de ces architectures. Nous présentons également succinctement les indicateurs de SdF que les clients et les systèmes intégrateurs (SI) demandent dans leurs spécifications de projet. Enfin, les dernières sections sont dédiées aux caractéristiques à considérer dans la description des architectures de contrôle-commande et celles liées au contexte d'avant-vente.

Ce chapitre permet ainsi de définir les bases de l'étude afin d'introduire les problématiques industrielle et scientifique qui en découlent.

## 1.2. Définition d'une architecture de contrôle-commande

Une architecture de contrôle-commande est un ensemble de composants, à la fois matériels et logiciels, permettant de réaliser l'automatisation d'un procédé industriel (Schneider Electric, 2018). C'est sur ce type de système de contrôle-commande que nous allons développer notre évaluation des performances de SdF. L'automatisation du procédé industriel se base sur trois axes principaux (cf. Fig. 1.1) :

- Une mesure et une collecte des différents paramètres nécessaires au fonctionnement du procédé, à l'aide de capteurs et d'équipements intelligents principalement en amont du système,
- Une commande du procédé pour le maintenir dans un état de fonctionnement nominal, grâce à un ou plusieurs automates programmables industriels (API) pilotant l'ensemble des équipements et des actionneurs contrôlés par ces derniers,
- Une surveillance permanente de l'état du procédé et un contrôle dans le cas où des actions supplémentaires sont requises, par exemple lorsque le fonctionnement n'est plus considéré comme nominal.

Trois points de vue sont utilisés pour définir une architecture de contrôle commande. Selon les informations nécessaires, il sera plus judicieux de choisir une description par rapport à une autre. Les trois points de vue sont définis et expliqués dans les sous-sections suivantes.

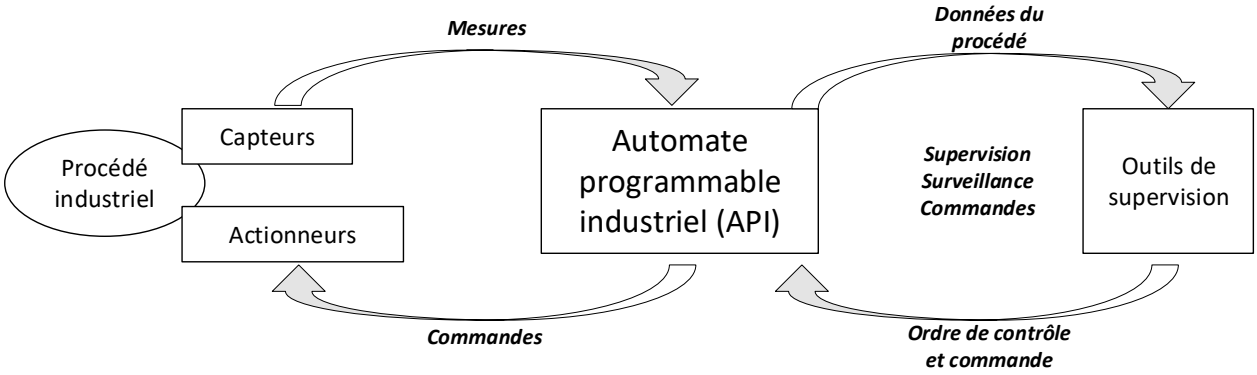



*Fig. 1.1 : Fonctionnement d'une architecture de contrôle-commande*

### 1.2.1. Architecture fonctionnelle

Le premier point de vue représente le système grâce à une architecture fonctionnelle qui décrit les différentes étapes du procédé et les fonctions associées. Elle permet de définir et d'identifier l'ensemble des services de contrôle-commande permettant de piloter le procédé industriel grâce à des fonctions liées entre elles et hiérarchisées, celles-ci donnant les informations sur les différentes missions ainsi que la sécurité de l'installation qui fera fonctionner le procédé industriel. Ces informations sont données dans un diagramme qu'on appelle PFD (*Process Flow Diagram*), comme sur la Fig. 1.2.

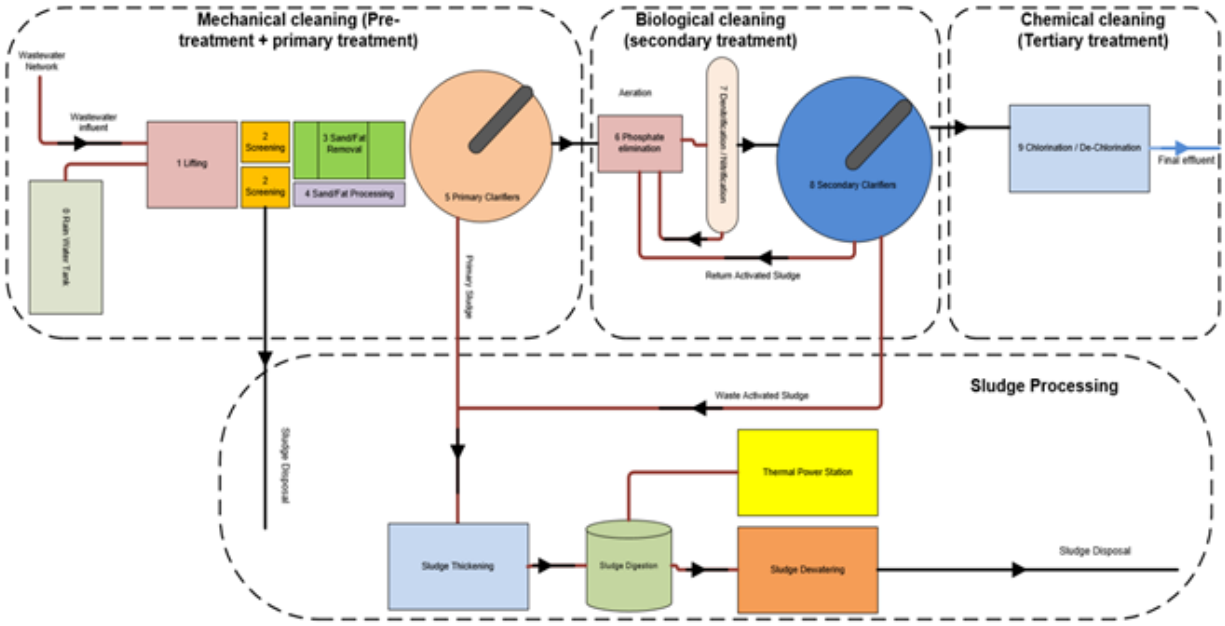



*Fig. 1.2 : Architecture fonctionnelle pour une usine de traitement des eaux usées pilotée par une architecture de contrôle-commande (Schneider Electric, 2018)*

La hiérarchisation des fonctions de contrôle-commande leur donne une structure et les place sur différents niveaux d'abstraction. La norme ANSI/ISA-95 (Scholten, 2007) (Johnsson, 2004) offre cinq niveaux fonctionnels, disponibles sur la Fig. 1.3. Cet organisation est valable pour un cas général, mais peut être modifié et ajusté pour s'adapter au type du procédé industriel automatisé ainsi que des pratiques industrielles rencontrées et peut donc être variable selon les contextes. Les cinq niveaux sont :

- **Niveau 0** : rassemble les équipements de terrain pour le pilotage du procédé tourné par l'architecture, qui permettent de récupérer les informations liées au procédé.
- **Niveau 1** : regroupe les composants permettant de lire les entrées et d'écrire les sorties et équipements dits intelligents pour le contrôle direct de la production et du flux.
- **Niveau 2** : concerne la supervision et la surveillance du procédé, en collectant les informations à partir des composants du niveau 1.
- **Niveau 3** : permet le contrôle de la production, avec le suivi des objectifs de production et de surveillance. Les données de production du procédé sont collectées pour être analysées dans le but de fournir des informations pour la traçabilité, le contrôle de la qualité, le suivi de production, l'ordonnancement et la maintenance préventive et curative.
- **Niveau 4** : dernier niveau pour l'ordonnancement de la production, et sa planification au niveau de l'usine. Il s'agit de la logistique principalement.

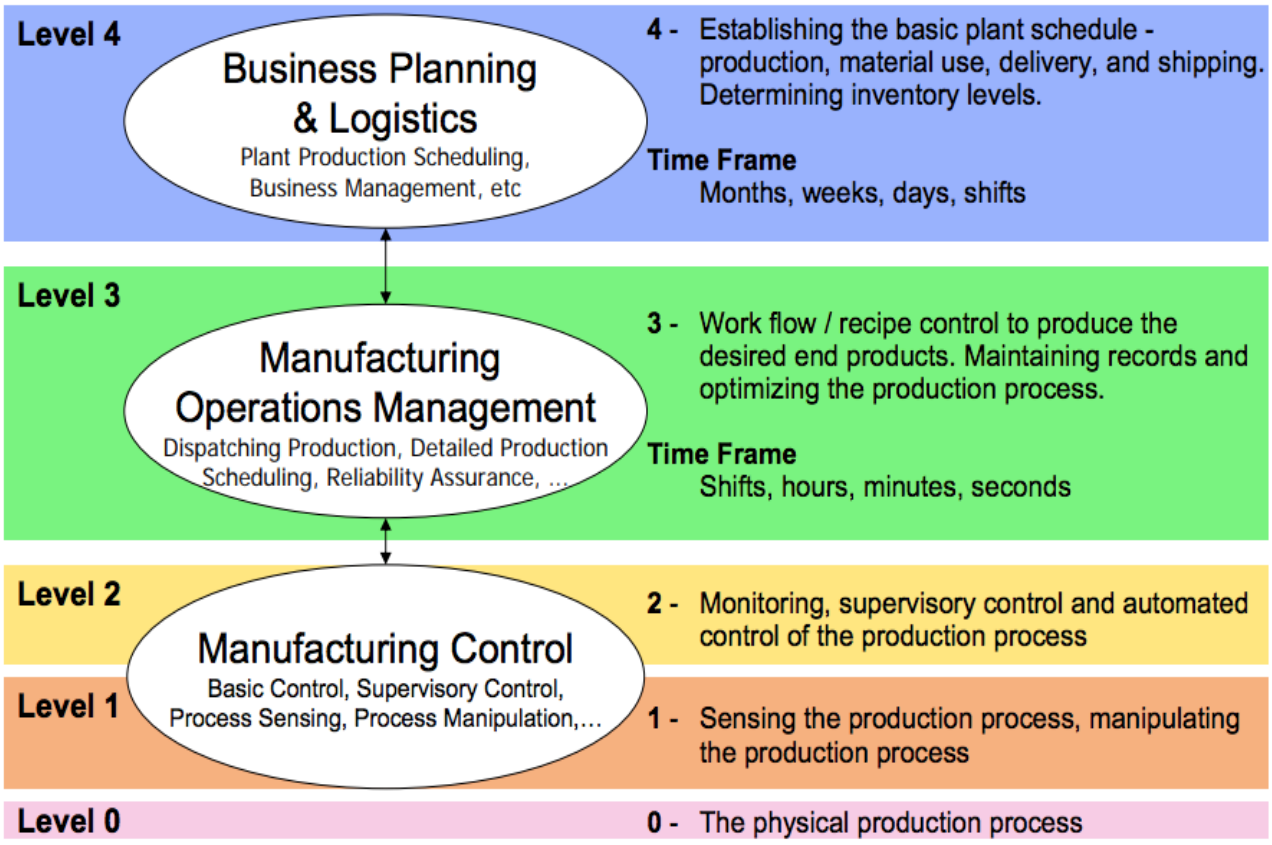



*Fig. 1.3 : Niveaux fonctionnels pour une architecture de contrôle-commande selon la norme ANSI/ISA-95*

### 1.2.2. Architecture matérielle

L'architecture matérielle décrit le système d'automatisme en présentant tous les composants ou matériels utilisés pour faire fonctionner le procédé ainsi que leurs interconnections (cf. Fig. 1.4). Il s'agit principalement d'API (ou PLC pour *Programmable Logic Controller* en anglais), de modules de communication, de modules d'entrées-sorties (E/S), de commutateurs Ethernet et d'équipements terminaux tels que des variateurs de vitesse par exemple.

**Procédé industriel : Traitement des eaux usées (Usine)**

**Architecture de contrôle-commande automatisant le procédé**

*Fig. 1.4 : Architecture de contrôle-commande pour une usine de traitement des eaux usées*

La structure du système d'automatisme représente également les canaux de communication et les interactions entre les composants. Cette structure comprend généralement le matériel séparé selon les cinq niveaux décrits dans la norme ISA-95 (cf. section 1.2.1 ).

Le premier niveau qui regroupe les composants terminaux dédiés à l'exécution des tâches du procédé industriel (capteurs, actionneurs), le second niveau qui correspond aux composants pour commander le procédé (API) et/ou pour faire le lien entre la production et le centre de supervision, et enfin les niveaux supérieurs qui intègrent les composants pour superviser le procédé (système de commande et d'acquisition des données SCADA), et les composants logiciels pour l'optimisation du procédé (cf. Fig. 1.5).

- **Niveau 0** : capteurs, actionneurs etc.
- **Niveau 1** : modules E/S et équipements dits intelligents.
- **Niveau 2** : l'API étant le dispositif destiné à la commande du procédé industriel. Il envoie des ordres aux actionneurs à partir des données d'entrée récupérées à partir des capteurs.
- **Niveau 3** : le SCADA qui permet à un opérateur de contrôler le procédé.
- Le MES (*Manufacturing Execution System*) correspondant essentiellement à un logiciel de pilotage et d'optimisation de la production entre le niveau automatisme de l'usine et le logiciel de planification du niveau supérieur.

- **Niveau 4** : l'ERP (*Enterprise Ressource Planning*) qui est la gestion de la production, de la comptabilité et finance, de la logistique et gestion des stocks, des ressources humaines et du management de projet.

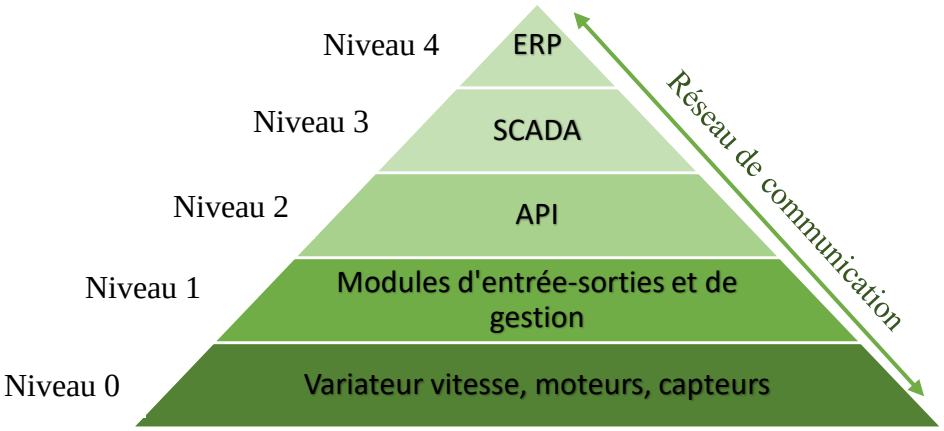



*Fig. 1.5 : Niveaux fonctionnels d'un point de vue matériel et logiciel pour une architecture de contrôle-commande selon la norme ANSI/ISA-95*

### 1.2.3. Architecture opérationnelle

La dernière description est l'architecture opérationnelle. Il s'agit d'une combinaison des deux précédents types de définition d'architectures, fonctionnelle et matérielle, avec la projection de l'architecture fonctionnelle sur l'architecture matérielle (cf. Fig. 1.6). Les fonctions, permettant de décrire le procédé industriel, sont associées aux composants. Les flux de communication sont associés aux liens physiques entre les composants. Il s'agit de pouvoir déterminer quels composants sont en charge d'une fonction, de caractériser les missions à partir du matériel disponible.

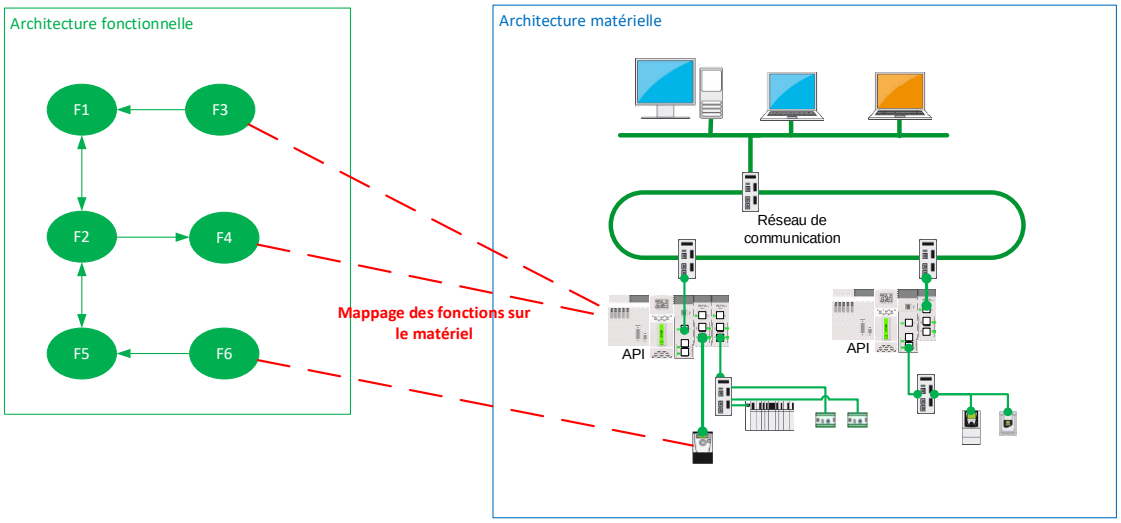



*Fig. 1.6 : Principe de l'architecture opérationnelle, combinaison de l'architecture fonctionnelle et matérielle*

Le scope de notre travail de recherche ne prendra pas en compte tous les niveaux cités précédemment. En effet, nous nous plaçons dans la phase d'avant-vente d'un projet c'est-à-dire

lors de la conception de l'architecture de contrôle-commande pour l'automatisation d'un procédé industriel tel que (traitement de l'eau, industrie minière, secteur du transport etc.). Cette phase d'avant-vente, dont la description se fera dans une prochaine section, se déroule dans un contexte particulier qui nous amène à ne considérer que trois niveaux, ceux au cœur du contrôle du procédé (cf. Fig. 1.7) :

- **Niveau 0-1** : Equipements terminaux du type modules E/S, variateurs de vitesse, gérant les départs moteur, etc.
- **Niveau 2** : API

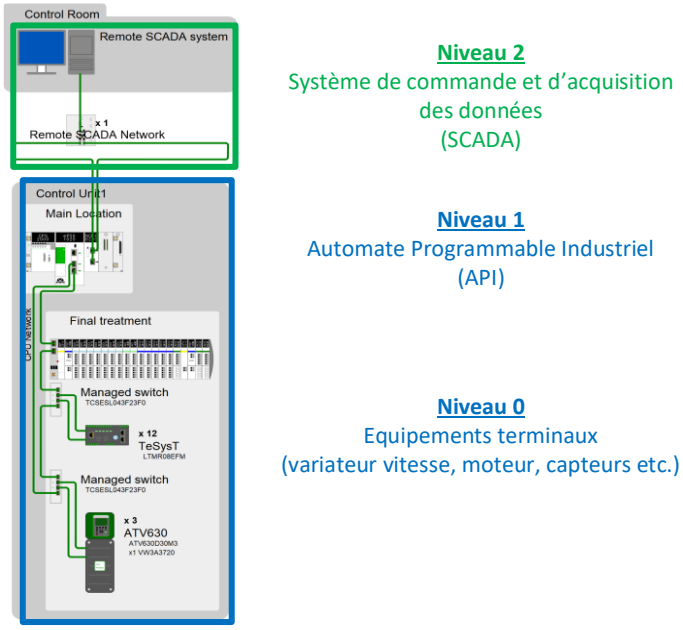



*Fig. 1.7 : Architecture matérielle et ses trois niveaux*

### 1.3. Intérêt de la SdF pour une architecture de contrôle-commande

Lors de la conception en phase d'avant-vente, il est très souvent demandé de pouvoir évaluer les performances du système pour vérifier sa conformité aux requêtes exprimées dans le cahier des charges. Mettre en place une évaluation de ces performances et les quantifier permet de déterminer plus facilement si oui ou non le système est conforme aux requêtes industrielles formulées. Parmi ces performances se trouvent celles qui doivent respecter des exigences et des contraintes en matière de SdF (Zio, 2009) (Bertsche, 2008) (Birolini, 2004).

Les architectures d'automatisme ont une durée de vie de plusieurs dizaines d'années (environ entre 30 et 40 ans). Il faut donc pouvoir estimer les défaillances pouvant arriver sur cet intervalle de temps et prédire le comportement dysfonctionnel du système sur sa période d'utilisation, afin de limiter les arrêts du système et/ou les dégradations du procédé industriel que l'architecture pilote. C'est donc sur ces performances fiabilistes que notre étude va se focaliser. Pouvoir les évaluer en avant-vente permettra de modifier la configuration de l'architecture afin de répondre au mieux aux attentes du client, auteur du projet industriel soumis.

La SdF est la science qui étudie les défaillances (Villemeur, 1988). Elle permet d'étudier la capacité d'un système à éviter ou diminuer l'apparition de défaillances qui entrainerait une dégradation du procédé industriel plus graves qu'acceptable. Cette science couvre les notions de fiabilité, disponibilité, maintenabilité et de sécurité, regroupés sous le sigle FMDS (ou RAMS en anglais pour *Reliability, Availability, Maintainability, Safety*) et c'est la science au cœur de nos travaux de recherche.

En évaluant les performances de SdF de l'architecture, nous serons en mesure de faciliter la mise en place d'un plan de maintenance, mais également de réduire les temps d'arrêt et ainsi diminuer les coûts liés à la réparation. Il sera possible aussi de proposer une architecture plus robuste par rapport à l'environnement dans lequel elle va évoluer et d'être tolérante aux fautes. Faire cette évaluation donne la possibilité de proposer une architecture qui soit optimale à la fois pour faire fonctionner le procédé industriel et pour assurer sa continuité avec le moins d'interruptions au cours de son opération.

Les bénéfices d'une telle évaluation en phase d'avant-vente sont multiples (Schneider Electric, 2019) :

- Un bénéfice économique, car en anticipant le comportement dysfonctionnel de l'architecture de contrôle-commande, nous sommes en mesure de réduire les coûts de réparation (réduction du stock de pièces de rechange au strict minimum, opérateur de maintenance non sollicité excessivement, etc.),
- Une assurance sur la sécurité des biens et des personnes afin d'éviter une catastrophe ou un danger trop important qui auraient pu être évités si une étude a été réalisé en amont,
- Une robustesse renforcée de l'architecture d'automatisme en montrant à partir de l'évaluation les limites envisageables liées au matériel choisi et ainsi proposer des composants plus pertinentes pour l'application ou une architecture différente pour le système avant son installation sur site,
- Un apport technique important permettant d'appuyer la proposition d'architectures basées sur des preuves et des explications détaillées face à des exigences formulées par le client.

L'évaluation sera faite principalement en se basant sur l'architecture matérielle avec un complément sur les fonctionnalités ne nécessitant pas de connaitre en détails les fonctions implémentées, à travers le contexte d'utilisation et la connaissance du comportement général des composants du système. L'analyse complète et approfondie des fonctions sera écartée de l'étude, sachant qu'en phase d'avant-vente, il est rare de connaître les fonctions qui seront implémentées dans les composants.

### 1.4. Liste des indicateurs de SdF pour une architecture de contrôle-commande

À l'issue d'une série d'analyses sur les besoins industriels liés à la sûreté de fonctionnement, plusieurs indicateurs de SdF ont été retenus pour évaluer les performances fiabilistes et feront l'objet des prochaines sections (Schneider Electric, 2019) (Verma, Ajit, & Karanki, 2010) (Villemeur, 1988). Ici, nous donnons une définition succincte des indicateurs tels qu'ils sont perçus

dans le milieu industriel. Les définitions complète seront vues dans le chapitre 3, lors de nos investigations dans la littérature scientifique.

Pour les définitions données dans cette section, nous considérons une entité comme étant l'un des points suivants :

- Un composant, matériel ou logiciel, présent dans l'architecture de contrôle-commande.
- Un ensemble de composants formant un sous-système dans le système global.
- Le système global.

### 1.4.1. Disponibilité d'un système

Le premier indicateur demandé est la disponibilité. Dans le cadre de l'évaluation des performances fiabilistes d'une architecture de contrôle-commande, il est important de connaitre la valeur de cet indicateur.

En effet, la disponibilité permet de donner une appréciation de la capacité de l'architecture d'être opérationnelle à l'instant t. Pour les systèmes de contrôle-commande, cet indicateur est déterminant pour surveiller l'état de la communication entre les contrôleurs et les équipements terminaux, afin de ne pas interrompre la supervision et le contrôle du processus industriel. Il mesure l'état de la commande à un instant donné.

Par exemple, une disponibilité de 0,99 à 1 an pour un module de communication, entre l'API et un composant terminal indique que la commande venant du contrôleur pour le device à travers le module, a 99% de chances de ne pas être défaillante à 1 an d'utilisation.

### 1.4.2. Fiabilité d'un système

Le deuxième indicateur retenu est la fiabilité. La fiabilité caractérise la capacité de l'architecture de contrôle-commande à fonctionner sans interruption sur un intervalle de temps qui peut être critique pour le procédé industriel.

Il est primordial de l'évaluer pour s'assurer de la continuité des missions attribuées à l'architecture mais également pour anticiper à quel moment il est le plus judicieux de changer certains composants afin d'éviter une panne qui pourrait être fatale au système. Pour un système de contrôle-commande, il permet de suivre l'évolution du contrôle du procédé industriel, dans quelle mesure celui-ci sera réalisé sans changement sur le système pendant un laps de temps.

Par exemple un API ayant une fiabilité de 0,80 à 1 an signifie que ce dernier a 80% de chances de ne pas avoir eu d'interruption sur 1 an pour réaliser sa mission.

Son complémentaire est appelé défiabilité.

### 1.4.3. Nombre de défaillances et temps d'indisponibilité pour un système

Une défaillance est un événement caractérisant l'interruption d'une entité à accomplir une fonction requise et est la conséquence en général d'un défaut de cette entité qui aurait pu survenir

depuis sa mise en service (défaut de fabrication) ou dû à des conditions favorables à sa dégradation (phénomène de vieillissement).

Une défaillance peut être classifiée par exemple, en fonction de sa nature, comme c'est le cas avec la norme pour la sécurité fonctionnelle, la norme CEI 61508 (CEI, 2010). Cette norme définit les exigences de sécurité d'un système de contrôle commande grâce à la mise en place d'un système instrumenté de sécurité (SIS), exclusif à la sécurité. Ce SIS surveille le système de contrôle-commande, détecte les conditions d'un danger imminent grâce à un système de vote, et réagit en actionnant une mise en sécurité du système si ce dernier s'avère être un danger potentiel.

La norme CEI 61508 répertorie les défaillances en deux types :

- **Une défaillance sûre** : il s'agit d'une défaillance qui ne met pas en danger le SIS ou qui ne le met pas dans l'impossibilité d'exécuter sa fonction de sécurité pour le système de contrôle-commande
- **Une défaillance non sûre** : c'est une défaillance qui peut amener un danger potentiel pour le SIS en lui-même ou qui peut bloquer la fonction de sécurité et donc poser un risque pour le système de contrôle-commande et le procédé industriel qu'il commande.

Il est également possible de classer les défaillances en deux autres types, toujours en lien avec le SIS, mais avec une optique et une approche différente :

- **Une défaillance détectée** : c'est une défaillance que le SIS a pu repérer grâce à son diagnostic et son système de vote.
- **Une défaillance non détectée** : il s'agit d'une défaillance non détectée par le SIS au moment du diagnostic, ce qui peut être un danger potentiel pour le système de contrôle commande.

Les deux classifications proposées par la norme montrent une différenciation portée par rapport au SIS, et restent orientées uniquement à la sécurité du système. Or notre étude s'intéresse à la partie contrôle-commande du système ainsi qu'à son organisation pour faire fonctionner l'automatisation du procédé industriel. Ainsi, au lieu de faire appel à la définition d'une défaillance sûre et non sûre ou à la définition d'une défaillance détectée ou non détectée, qui est caractéristique de la sécurité fonctionnelle, nous allons classer les défaillances en deux autres types. Ces types seront valables pour la partie applicative, et par rapport aux besoins que le client pourrait avoir.

Nous allons donc définir les deux types de défaillance suivants (cf. Fig. 1.8) :

- **Une défaillance non critique** qui est une défaillance n'engendrant pas une dégradation ou un arrêt de l'entité, donc avec un impact mineur sur le système (s'apparente à une défaillance sûre dans le cadre de la sécurité fonctionnelle)
- **Une défaillance critique** qui est une défaillance pouvant engendrer une dégradation ou un arrêt de l'entité, qu'il faut absolument répertorier pour éviter les interruptions potentielles sur le procédé industriel, donc avec un impact majeur sur le système (s'apparente à une défaillance non sûre dans le cadre de la sécurité fonctionnelle).

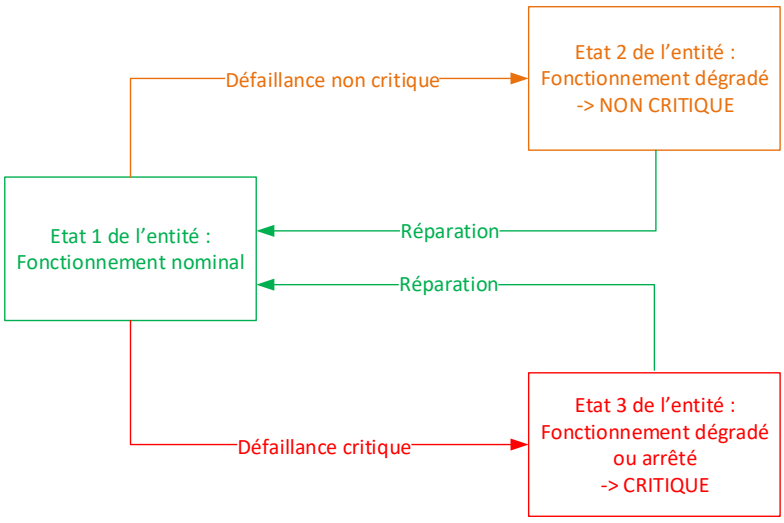



*Fig. 1.8 : Lien entre les défaillances et les états pour une entité*

Estimer le nombre de défaillances dites critiques s'avère une nécessité pour connaître en moyenne le nombre de fois où le système tournera dans un état dégradé ou arrêté avec un impact plus ou moins majeur sur le procédé industriel. Néanmoins, il ne faut pas écarter l'évaluation du nombre de défaillances non critiques car l'addition des deux catégories permettra de prévoir le nombre de pièces de rechange nécessaire pour assurer à chaque fois la réparation la plus optimale à la suite d'une défaillance.

Il est primordial également d'estimer le temps d'indisponibilité de l'architecture lié à l'occurrence de défaillances pendant le temps d'opération du système. Toutes ces données permettent de pouvoir prédire le plan de maintenance, à savoir le nombre de pièces de rechange à prévoir pour chaque composant et combien de temps le système va rester dans un état de dégradation, état qui pourrait être problématique pour le procédé industriel. Par exemple, nous pourrons déterminer si l'API tombe en panne, combien de fois et sur combien de temps, ce qui donne une estimation du nombre de pièces à mettre dans le stock de maintenance et l'intervalle de temps où le procédé industriel ne sera pas réalisé dans des conditions nominales.

Ces deux performances permettront également de valider ou non la conception de l'architecture de contrôle-commande en fonction des besoins formulés dans le cahier des charges. Si le système a un nombre de défaillances dépassant le seuil accepté, il sera alors nécessaire de retravailler sur la conception de l'architecture pour respecter ces spécifications.

### 1.4.4. Impact des composants dans un système

Les défaillances qui touchent un composant de l'architecture de contrôle-commande n'engendrent pas les mêmes conséquences sur le système global et dépendent fortement de la famille du composant. Une défaillance sur un équipement en charge d'exécuter une tâche (lire une entrée comme la hauteur de l'eau dans un réservoir) n'est pas forcément aussi critique qu'une défaillance sur un équipement en charge de commander une partie importante du procédé industriel comme l'API (en charge de donner des instructions à un ensemble d'équipements terminaux).

Dès lors, il paraît indispensable d'être en mesure de cibler les composants qui sont le plus sujet à des dégradations pendant toute la période d'utilisation du système. Il est important de connaître leur impact sur le système global. Cela permettra ainsi non seulement de pouvoir identifier les points critiques de l'architecture d'automatisme, mais également d'affiner la gestion du stock pour les pièces de rechange en ayant la possibilité de savoir à l'avance quels composants sont les plus susceptibles d'avoir besoin d'être remplacés.

L'impact des composants sur le système se basera sur la notion de facteurs d'importance, introduits dans le chapitre 3 et déployés dans le chapitre 4, expliquant le développement de l'évaluation avec la justification des facteurs sélectionnés pour notre étude.

## 1.5. Caractéristiques d'une architecture de contrôle-commande

Une architecture de contrôle commande est un système dont le comportement dysfonctionnel ne peut être déduit directement de l'état fonctionnel et/ou dysfonctionnel de ses propres composants. De plus l'état, le type et la complexité des échanges entre les modules ne permet pas de déterminer l'état du système global sachant seulement qu'un composant a changé d'état. Ainsi, d'un point de vue fiabiliste, l'architecture n'est pas considérée comme un système dit statique.

Comme les autres systèmes dits dynamiques, ces données ne suffisent pas pour déterminer l'état dysfonctionnel du système global. Il faudra également connaitre les événements qui surviennent et qui changent l'état des composants, ainsi que les interactions entre ces derniers. Une architecture de contrôle-commande respecte plusieurs critères que l'on retrouve généralement dans les systèmes dynamiques. Ces derniers font l'objet des sous-sections suivantes (Piriou, 2015).

### 1.5.1. Système réparable

La prise en compte de composants réparables, ce qui est le cas dans une architecture de contrôle-commande, implique souvent des difficultés dans la description du comportement du système global.

Lorsque le composant est indépendant des autres, c'est-à-dire que son état n'a pas d'impact sur celui des autres (les défaillances sont décorrélées entre elles), le scénario de réparation reste relativement simple.

En revanche, si le composant est lié fonctionnellement avec un autre, une redondance passive par exemple, il faut connaitre les règles de basculement pour savoir comment se comporte le système face à une défaillance sur un composant redondant. Si le premier tombe en panne et se met en réparation, le deuxième prend le relais ; une fois la réparation terminée, est-ce que le composant réparé reprend le rôle principal ou se met en mode passif et devient le composant secondaire ? Nous constatons ici que les deux possibilités n'impliquent pas la même conséquence pour le système entier, et donc sur son comportement dysfonctionnel.

Savoir comment réagissent les composants à la suite d'une défaillance puis d'une réparation aura un impact non négligeable sur l'estimation des performances fiabilistes du système global de contrôle-commande, et selon le degré de connaissances que nous avons sur le système réparable, il faudra considérer des critères qui affineront le calcul des performances.

### 1.5.2. Système avec composants à multi-états

Le comportement le plus classique pour un composant est d'être binaire : soit il fonctionne et assure sa mission soit il ne fonctionne pas et donc défaillant. Dans ce cas, les calculs peuvent être simplifiés et la déduction des groupes de composants qui engendrent la dégradation du système global peut être réalisées plus facilement.

Mais il est courant dans le milieu industriel d'avoir des systèmes dont les composants sont à multi-états : ils peuvent avoir plusieurs niveaux de dégradations par exemple, telle que la position de repli où le composant applique des valeurs par défaut pour garantir la continuité d'un procédé industriel en mode dégradé.

Prendre en compte ces différents états, qui ne se limitent plus à du tout ou rien, est à la fois primordial et complexe dans le calcul des performances du système global. Il faut déterminer quels états donnent lieu à une défaillance ou à une dégradation du système, ce qui apporte une granularité plus importante pour la description des états dysfonctionnels de l'architecture.

Par exemple, la perte de la fonction de contrôle des d'entrées sorties par le processeur (CPU) dans l'API, état étant considéré comme une dégradation du composant CPU, engendrent des conséquences totalement différentes, par rapport à une panne ou arrêt de la CPU donc une défaillance, pour le procédé industriel géré par le système de contrôle commande dans lequel on retrouve la CPU.

### 1.5.3. Système à fiabilité dynamique

On définit la fiabilité d'un système comme étant statique lorsque les données élémentaires considérées pour caractériser celle-ci sont des constantes dans le temps. Il en découle naturellement qu'une fiabilité dite dynamique dépend de données variables dans le temps.

Pour un système à fiabilité dynamique, il faudra modéliser les phénomènes de vieillissement et de dégradation des composants, un processus en continu dans le temps, où le taux de défaillance ne sera donc plus une constante mais une variable qui évolue au cours du temps. Il faudra également considérer les conséquences de cette évolution en continu de variables du procédé tourné par l'architecture de contrôle-commande (certaines variables en franchissant un certain seuil peuvent impacter le taux de défaillances des composants du système) et l'évolution discrète de l'état fonctionnel ou dysfonctionnel des composants.

Dans les pratiques actuelles de Schneider Electric, nous avons des composants électroniques dans une architecture dont on ne considère que des variables statiques, donc une fiabilité statique, mais il faudra par la suite introduire des composants à fiabilité dynamique, car les phénomènes de vieillissement peuvent être intégrés dans la description du comportement dynamique du système. Il faudra alors prendre en compte ces informations dans l'estimation fiabiliste des performances du système avec la mise à jour du niveau de dégradation des composants.

À la suite d'un événement, il est possible qu'un composant soit remis à neuf (après réparation) ou qu'il vieillisse rapidement. Il faut prendre en compte ces évènements dans le calcul. Avoir un modèle de fiabilité dynamique permet de représenter les interactions entre des événements

stochastiques discrets comme l'occurrence d'une défaillance lorsqu'un composant est dans état *i*, et des variables continues comme le niveau de dégradation d'un composant au cours du temps.

Un système à fiabilité dynamique est donc caractérisé par le couplage d'événements aléatoires discrets d'une part (configuration du système comme la transition entre les modes de défaillances, les états de marche etc.) et par des phénomènes déterministes continus (paramètres du type pression, température ou changement de niveau d'usure etc.).

### 1.5.4. Système reconfigurable

Il s'agit d'un système disposant de mécanismes de commutation qui lui permet de modifier de façon dynamique sa structure et/ou le comportement de ses composants lorsqu'ils changent d'état. Les systèmes reconfigurables regroupent généralement l'ensemble des caractéristiques décrites dans les sous-sections précédentes. Une architecture de contrôle-commande est généralement un système reconfigurable, avec des commutations possibles entre certains composants. Les reconfigurations sont importantes entre autres pour des composants avec un composant secours qui prend le relais dans le cas où une panne intervient sur le composant principal.

Ces commutations et donc reconfigurations se font pour plusieurs raisons possibles, qui sont souvent liées à la criticité du procédé industriel exécuté par le système, dont :

- Fonctionnelles (changement de régime de fonctionnement, par exemple un composant en redondance qui devient maître du processus),
- De production (pour accélérer ou réduire le flux de production, réduire les temps d'arrêt),
- De maintenance (s'il est nécessaire de réaliser un test périodique ou une maintenance préventive par exemple),
- De sécurité (si on doit recalibrer le système vers un état sûr),
- etc.

Il existe un impact non négligeable sur les performances du système lorsque ce dernier est reconfigurable. Il faut prendre en compte par exemple les redondances présentes pour améliorer les performances de sûreté de fonctionnement de l'architecture ou les changements de mode des composants qui impliquent un changement dans le comportement dysfonctionnel du système global, que ce soit vers une amélioration (comme avec les redondances) ou vers une dégradation (comme la surcharge d'un composant ou sa dégradation dans le temps).

Il faut également faire attention aux commutations qui rajoutent un nouveau type de défaillance possible : les défaillances à la sollicitation. Si la commutation échoue, alors on perd non seulement le composant défaillant mais également le composant qui devait prendre le relais, ce qui rend caduque la reconfiguration.

### 1.5.5. Conclusion sur les caractéristiques

La complexité du comportement dynamique que nous retrouvons dans les architectures de contrôle-commande, entraine à considérer ce contexte particulier pour calculer les indicateurs répertoriés. Il faudra définir les bases du mécanisme sur comment se comporte le système face à une défaillance ou un basculement sur un nouveau mode de dégradation, tout en sachant que des

événements se déroulent dans le même temps dû aux interactions entre les composants à un instant donné.

Ainsi le choix de la méthode pour non seulement décrire mais aussi modéliser le système devra prendre en compte son comportement dynamique. Il faudra donc choisir parmi les modèles dynamiques, et nous verrons dans les chapitres suivants, les candidats potentiels et ceux que nous écarterons car ne répondant pas au besoin formulé.

### 1.6. Caractéristiques du contexte d'avant-vente

La première partie d'un projet d'automatisation se fait dans la concrétisation d'un accord avec le client pour déployer son système devant piloter un procédé industriel particulier. Il s'agit d'une étape où les besoins du client sont définis dans une spécification technique ou les performances, notamment les performances liées à la SdF sont précisées. Il nous incombe alors de proposer une architecture de contrôle-commande qui puisse répondre à ces spécifications et ces performances afin de sécuriser le projet : c'est la phase d'avant-vente (cf. Fig. 1.9).

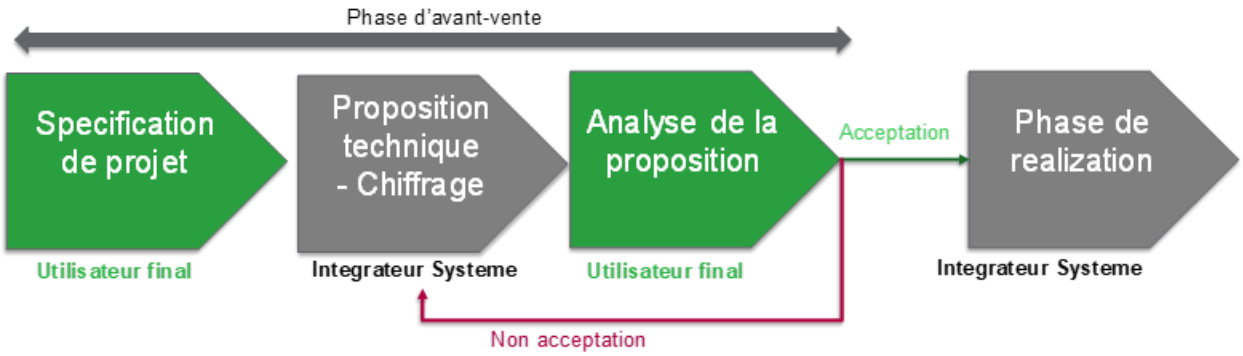



*Fig. 1.9 : Les différentes étapes la phase d'avant-vente*

Nous sommes donc tout en amont du processus pour gagner le projet industriel, et la réussite repose sur la proposition technique et l'estimation des performances que nous aurons développées. Cette estimation, qui sera le cœur de notre travail, se base sur un cahier des charges qui comprend deux éléments :

- Des spécifications du procédé avec son type, sa nature et ses caractéristiques ainsi que des informations sur l'environnement de déploiement de la future architecture ;
- Des spécifications fonctionnelles, plus ou moins complètes, qui décrivent les services de commande, de surveillance, ou de supervision à développer ainsi que les performances attendues.

Pour garantir ces performances de SdF, il est nécessaire de les évaluer pendant la phase d'avant-vente. Cependant, cela reste difficile en raison de la faible quantité d'informations sur le projet en avant-vente qui se limitent au PFD et aux spécifications dans le cahier des charges. Les ressources humaines et financières sont également limitées ainsi que le délai restreint pour la soumission d'une proposition d'architecture au client.

Actuellement, l'approche utilisée par Schneider Electric pour évaluer les performances fiabilistes lors de la phase d'avant-vente consiste à envoyer la conception de l'architecture proposé par le SI à un expert en gestion des risques et en SdF. Cet expert analyse l'architecture par le biais d'outils et de méthodologies internes, réalise des analyses comparatives pour valider ou réfuter le design proposé. Cette approche est efficace pour concevoir la meilleure architecture pour un projet, mais elle nécessite des ressources considérables en moyens humains, un temps jugé trop important en phase d'avant-vente pour que l'expert puisse fournir des résultats et elle n'est pas flexible.

De plus, si l'architecture déployée ne répond pas aux spécifications du client après l'évaluation des performances faite par l'expert, il faut redéfinir le conception et soumettre la nouvelle architecture à une nouvelle évaluation, ce qui implique non seulement un coût pour le SI mais aussi pour l'expert et par conséquent aussi pour le client (ou l'utilisateur final).

Le contexte d'avant-vente est donc caractérisé par les critères suivants :

- Une connaissance partielle des architectures due à des informations imprécises ou non disponibles,
- Des contraintes en termes de temps et de personnels fortement limitées,
- La nécessité d'élaborer une offre commerciale pertinente constituée de plusieurs solutions d'architectures correspondant, pour le client, à différents compromis coûts / satisfaction des exigences.

Le processus permettant d'évaluer les performances fiabilistes devra donc prendre en compte ces caractéristiques afin de pouvoir fournir un outil qui soit exploitable en phase d'avant-vente, avec ses limites en ressources et en temps.

### 1.7. Conclusion

Dans ce chapitre nous avons introduit le contexte et les besoins de l'étude : évaluer les performances fiabilistes d'une architecture de contrôle-commande avec un comportement dynamique, durant la phase d'avant-vente. Nous avons alors listé les indicateurs qui nous intéressent et pourquoi ils sont importants à évaluer. Nous avons enfin posé les conditions que nous allons prendre en considération dans le développement de notre évaluation fiabiliste, le comportement dynamique de l'architecture et le contexte d'avant-vente. Nous verrons dans le chapitre suivant quel est le besoin industriel attaché à cette demande et comment nous allons y répondre. Nous allons voir comment développer les formalismes de modélisation nécessaires pour réaliser l'évaluation et respecter les exigences liées au contexte d'utilisation.

# Chapitre 2
# Formalisation du besoin industriel

## 2.1. Introduction

Ce chapitre introduit le besoin industriel résultant de la nécessité d'évaluer les performances fiabilistes d'une architecture de contrôle-commande formulée dans le chapitre précédent.

Nous allons, dans un premier temps, répertorier les pratiques industrielles actuelles et pourquoi nous devons développer une nouvelle approche pour estimer les performances afin d'intégrer de nouveaux axes d'amélioration. Nous verrons également comment répondre à cela en définissant les spécifications de la nouvelle évaluation en considérant le contexte particulier dans lequel l'estimation doit être réalisée, à savoir le comportement dynamique des architectures de contrôle-commande ainsi que le contexte d'avant-vente. Par ailleurs, nous fournissons les données en entrée que nous devons prendre en compte pour la mise en place de la nouvelle évaluation.

La problématique sera alors posée ainsi que les critères à respecter dans le cadre du développement de la nouvelle approche d'évaluation des performances de sûreté de fonctionnement.

## 2.2. Pratiques actuelles chez Schneider Electric

Évaluer les performances fiabilistes d'un système de contrôle-commande est primordial pour définir une proposition d'architecture qui couvre l'ensemble des exigences du client. C'est une étape cruciale en phase d'avant-vente, faisant l'objet d'une réflexion sur la meilleure approche pour répondre le plus rapidement possible afin de gagner le projet. Actuellement, cette évaluation peut être conduite de trois manières différentes selon la nature du projet, la criticité des performances fiabiliste ou le temps que les intégrateurs peuvent se permettre de consacrer à cette analyse.

### 2.2.1. Estimation des équipements critiques par les intégrateurs (SI)

La première approche consiste à réaliser une estimation des composants a priori critiques et à proposer des architectures redondantes sur ces composants.

Cette approche peut conduire à un surdimensionnement de l'architecture dans le cas où certains équipements se révèlent moins critiques que ce qu'ils avaient été estimés ou un sous-dimensionnement dans le cas où des composants critiques n'aient pas été identifiés ou si les redondances proposées sont insuffisantes.

Cette approche ne repose sur aucune modélisation, son seul intérêt réside dans le temps extrêmement court dévolu à l'analyse SdF mais peut bien sûr aboutir à des erreurs économiquement ou fonctionnellement graves. Lorsque cela est possible, les deux alternatives suivantes seront donc privilégiées.

### 2.2.2. Approche via les diagrammes de fiabilité par les intégrateurs (SI)

Cette approche repose sur un modèle de représentation, le diagramme de fiabilité (ou *Reliability Block Diagrams* RBD en anglais) proche de la structure de l'architecture. Il est donc naturel pour les intégrateurs en avant-vente et peut être construit sans connaissance experte en sûreté de fonctionnement.

Comme son nom l'indique, le diagramme de fiabilité (ou *Reliability Block Diagrams* RBD en anglais) est une méthode utilisée pour analyser et calculer principalement la fiabilité d'un système (Villemeur, 1988), (Pagès & Gondran, 1980) (Cocozza-Thivent, 1997) (Cepin, 2011). Il s'agit d'une représentation graphique du système étudié ayant pour but de déterminer l'état du système global en fonction des états de ses composants.

Il représente l'architecture du système sous la forme de blocs, chaque bloc correspondant généralement à un composant ou un ensemble de composants, ou à des fonctions si l'analyse est fonctionnelle et non sur le matériel. Ces blocs sont reliés entre eux lorsque des liens dysfonctionnels existent entre les composants. Ainsi nous avons la description du comportement dysfonctionnel du système global comme dans la Fig. 2. 1.

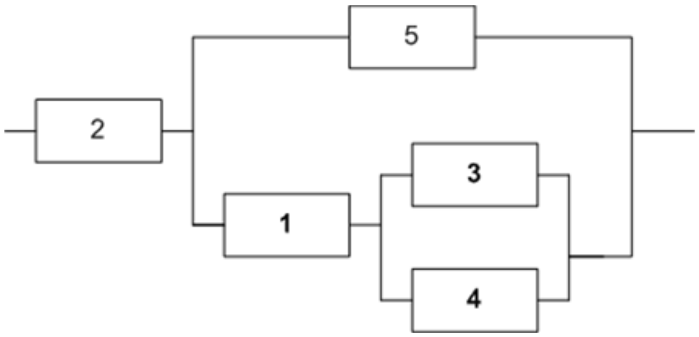



*Fig. 2. 1 : Exemple de diagramme de fiabilité constitué de cinq composants*

Par analogie avec un circuit électrique, le bloc représentant un composant est considéré comme un interrupteur fermé si le composant en question est dans un état de marche et ouvert si le composant est dans un état de panne. Le système est considéré comme étant opérationnel, c'est-à-dire en état de fonctionnement, s'il est possible de trouver un chemin permettant de relier le point d'entrée du diagramme au point de sortie.

De ce fait, il est possible de savoir dans quelles conditions le système est fonctionnel ou non, si nous trouvons un chemin permettant de relier le point d'entrée au point de sortie ou non, selon l'état des composants (cf. Fig. 2. 2 et Fig. 2. 3).

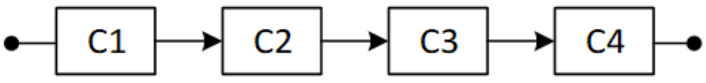



*Fig. 2. 2 : Diagramme de fiabilité d'un système de quatre composants où la panne d'un rend le système dysfonctionnel (composants en série)*

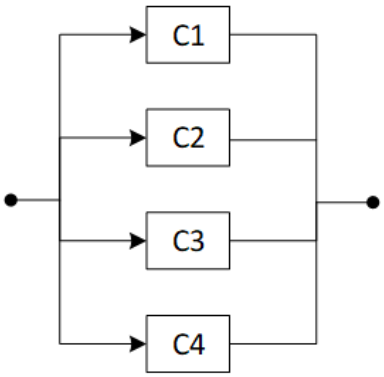



*Fig. 2. 3 : Diagramme de fiabilité d'un système de quatre composants où la panne simultanée des quatre rend le système dysfonctionnel (composants en parallèle)*

Se servir des diagrammes de fiabilité permet d'avoir une analyse qualitative et quantitative. Qualitative car nous déterminons les chemins qui mènent à la réussite de la mission du système, et donc nous recherchons les composants les plus problématiques, ceux qui se retrouvent avec la plus grande fréquence dans les combinaisons de composants conduisant à l'échec de la mission, et potentiellement à des incidents. Quantitative également puisqu'une probabilité de bon fonctionnement du système est calculée, avec la fiabilité des composants suivant une loi exponentielle pour leur durée de vie.

Trois configurations sont possibles pour décrire les liens entre les composants, ces configurations pouvant être utilisés pour décrire un système mixte (cf. Fig. 2. 4) :

- En série, lorsqu'un des composants entraine la panne du système complet,
- En parallèle, lorsque la panne du système arrive si tous les composants sont défaillants,
- En redondance k parmi n (k/n), s'il faut que k composants sur n soit défaillants pour que le système complet soit considéré en panne.

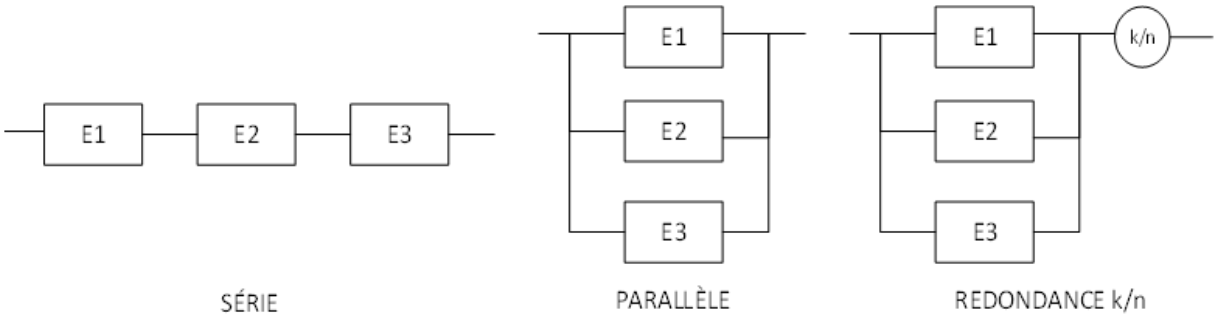



*Fig. 2. 4 : Configuration des composants dans un diagramme de fiabilité*

Dans le cadre d'un projet industriel soumis en avant-vente à Schneider Electric, tous les composants de l'architecture de contrôle-commande sont considérés en série et/ou en parallèle (cf. Fig. 2. 5). Autrement dit, un composant défaillant entraine automatiquement une détérioration du système complet s'il est en série, et dans le cas où il est en parallèle, il faut que tous les composants soient défaillants pour dégrader le système. Dans une configuration en série, il est donc primordial de s'assurer que tous les modules soient en état de fonctionnement durant toute la période

opérationnel de l'architecture de contrôle-commande. Dans une configuration en parallèle, il faut s'assurer qu'au moins un composant soit opérationnel pour que l'architecture ne soit pas défaillante. Il faudra donc non seulement fournir des composants robustes, mais également mettre en place une structure qui ne diminuent pas la robustesse que peuvent fournir les produits au système.

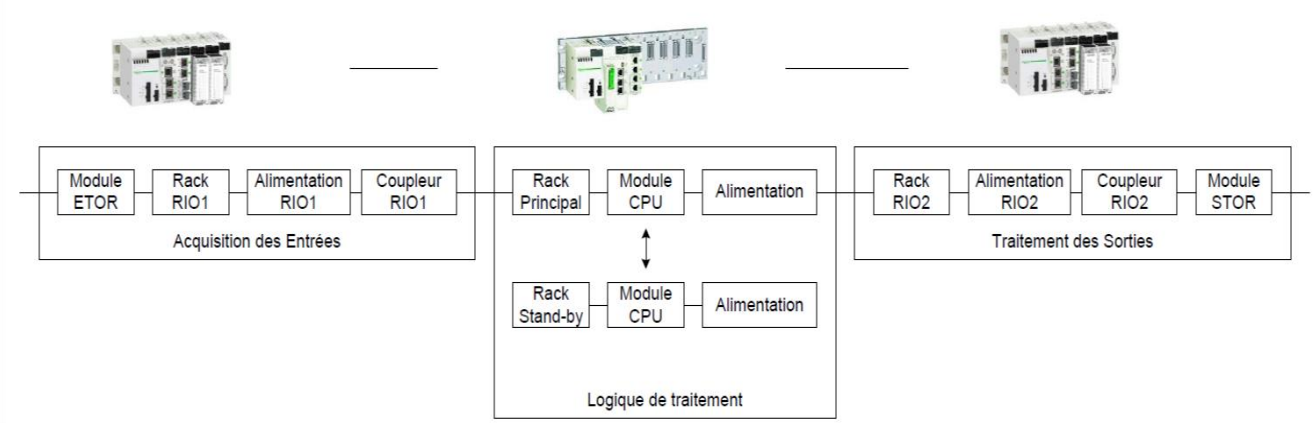



*Fig. 2. 5 : Exemple de diagramme de fiabilité utilisé à Schneider Electric*

A partir de cette représentation, les intégrateurs en avant-vente pourront donc calculer des métriques classiques telles que les MTBF, MTTF… sans faire appel à un expert externe.

Cette représentation est néanmoins limitée car l'approche est statique, elle ne se base que sur des structures logiques ne dépendant pas du temps, et booléenne car elle ne permet de prendre en compte uniquement que des composants à deux états, état de marche ou état de panne.

De plus, les diagrammes de fiabilité élaborés par les intégrateurs placent ra défaut les composants de l'architecture en série, à l'exception des composants redondants placés en parallèle. Cela revient à considérer que toutes les défaillances des composants en série sont au même niveau et ont un impact équivalent sur la défaillance globale de l'architecture, ce qui n'est pas tout à fait conforme à la réalité.

De plus, cette analyse est uniquement basée sur la structure de l'architecture et ne tient pas compte d'éventuelles situations dangereuses ou d'événements redoutés face auxquels l'architecture de contrôle-commande doit résister de manière appropriée. Ceci nécessite de réaliser une analyse plus poussée qui nécessite une analyse par des experts SdF qui ne font pas partie des équipes d'intégrateurs en avant-vente. Ce type d'analyse, réalisée à l'aide d'arbres de défaillances fournira donc des informations plus précises mais nécessitera, en contrepartie, un temps de traitement plus long.

### 2.2.3. Approche via les arbres de défaillances par des équipes expertes en SdF

L'analyse à partir d'un arbre de défaillances (AdD) est une autre méthode que nous pouvons classer dans la catégorie des modèles dysfonctionnelles et statiques (Cocozza-Thivent, 1997) (Villemeur, 1988) (Limnios, 2005) (Birnbaum, Esary, & Saunders, 1961) (Pagès & Gondran, 1980) (Duroeulx, 2020) (Rauzy, 1993).

Ces arbres représentent l'architecture du système sous la forme d'événements qui sont reliés entre eux par des opérateurs logiques ou portes, le but étant de déterminer toutes les combinaisons d'événements qui entraine un événement redouté, par exemple la défaillance du système complet (cf. Fig. 2. 6).

L'analyse se fait en définissant un événement redouté, à l'aide d'une analyse préliminaire des risques du système, et en déterminant quels sont les événements et leurs combinaisons qui aboutissent à l'événement redouté. Comme les diagrammes de fiabilité, ce sont les mêmes bases en terme probabiliste et booléenne, mais au lieu de se concentrer sur le bon fonctionnement du système, l'AdD modélise la défaillance du système.

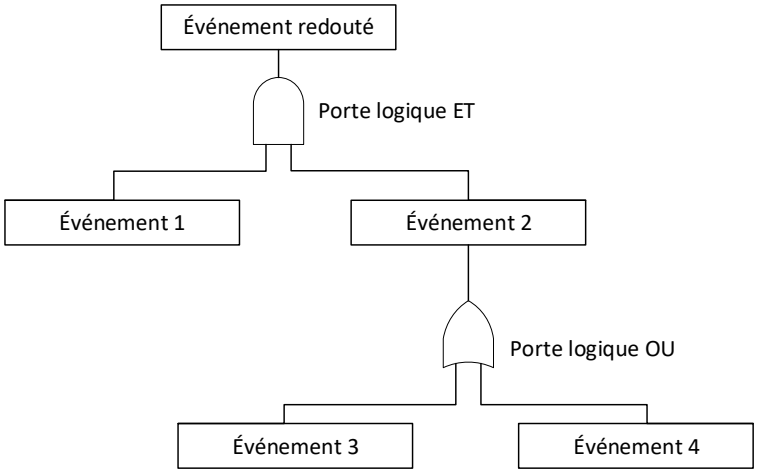



*Fig. 2. 6 : Exemple d'arbre de défaillances et de combinaisons d'événements conduisant à l'événement redouté*

A partir de cet arbre, les coupes permettent de représenter les différentes combinaisons d'événements entrainant l'événement redouté. Une coupe est définie comme un ensemble de composants tel que la combinaison de ces composants dans un état défaillant implique l'occurrence de l'événement redouté, à savoir la panne du système global. Les coupes minimales sont les coupes pour lesquelles tous les événements sont requis pour entrainer l'événement redouté (la suppression d'un événement dans une coupe minimal aboutit à un ensemble d'événements qui n'est plus une coupe).

Il est possible de représenter le système par un AdD s'il est cohérent. Nous considérons un système comme cohérent, s'il vérifie les conditions suivantes :

- Lorsque le système est défaillant, l'arrivée d'une nouvelle défaillance sur un de ses composants n'entraîne pas le rétablissement du système dans un état de bon fonctionnement,
- La panne de tous les composants implique forcément la panne du système global,
- Le bon fonctionnement de tous les composants entraine forcément le bon fonctionnement du système global.

Pour chaque événement de base, c'est-à-dire pour un composant élémentaire et non un sous-système, il est possible d'associer une probabilité d'occurrence qui, combinées entre elles, permet

de déterminer la probabilité d'occurrence de l'événement "système en panne", en considérant le type d'opérateurs logiques liant ces derniers.

Pour les portes, les principales sont les portes ET, OU et kooN (cf. Fig. 2. 7) :

- La porte ET est active si tous les événements en entrée de la porte sont actifs,
- La porte OU est active si au moins un des événements en entrée est actif,
- La porte kooN (K parmi N) est active si au moins k événements en entrée sur N événements sont actifs. Par exemple pour une porte 2oo3, cela signifie qu'il 2 des 3 événements actifs pour que la porte soit active.

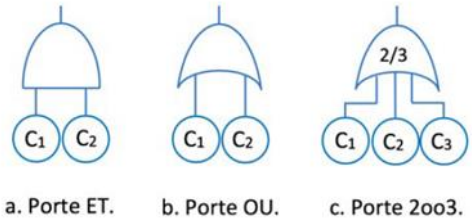



*Fig. 2. 7 : Opérateurs logiques dans les arbres de défaillances (Duroeulx, 2020)*

Un arbre de défaillances peut se traduire par un diagramme de fiabilité et vice versa, les deux modèles partageant les mêmes bases booléennes et probabiliste. La différence réside dans la finalité du modèle : si le diagramme de fiabilité représente le bon fonctionnement du système, l'arbre de défaillances modélise son complémentaire, c'est-à-dire la défaillance du système.

Ce type d'analyse réalisé par des experts en SdF permet donc de mesurer les indicateurs probabilistes de SdF des architectures face à une situation dysfonctionnelle redoutée. Elle permet donc de fournir des informations à la fois qualitative et quantitative sur les performances SdF des architectures mais nécessite un temps de traitement plus long du fait du recours aux experts.

### 2.2.4. Bilan des pratiques actuelles

Le contexte d'avant-vente nous oblige à trouver une solution peu consommatrice en temps et en ressources. Si la solution reposant sur des analyses par arbres de défaillances fournit des indicateurs pertinents, elles impliquent de disposer d'un nombre non négligeable de données sur l'aspect fiabiliste de l'architecture de contrôle-commande pas nécessairement connues en phase d'avant-vente. Par ailleurs, elles font appel à des équipes d'experts SdF externes à l'équipe d'intégrateurs en charge d'élaborer la proposition commerciale et sont couteuses en temps. A contrario, les analyses réalisées par les intégrateurs à l'aide de diagramme de fiabilité ne fournissent qu'une estimation grossière des performances qui peut se révéler insuffisante ou conduite à des surdimensionnements de l'architecture proposée.

Par ailleurs, les analyses par diagramme de fiabilité ou arbres de défaillances restent des analyses statiques avec toutes leurs limites. Il n'est par exemple pas possible de modéliser des séquences ordonnées de défaillances conduisant à la défaillance globale de l'architecture (par exemple, l'événement *a* suivi de l'événement *b* entraine la défaillance du système alors que l'événement *b* suivi de l'événement *a* n'a lui aucun impact).

D’autre part, l’occurrence d’une défaillance sur un composant pourra avoir pour effet de provoquer le passage d’un autre composant dans un nouvel état sur lequel à partir duquel d’autres événements dysfonctionnels pourraient survenir. Des composants peuvent être dépendants, à travers des liens physiques comme l’alimentation, impliquant des défaillances de cause commune entre deux ou plusieurs composants. Une défaillance peut impacter en même temps plusieurs composants et les rendre non opérationnels.

Enfin, les architectures de contrôle-commande opèrent dans des environnements qui peuvent varier au cours du temps (température, pression…) et peuvent être le théâtre de phénomène de vieillissement ou de dégradation. Tous ces éléments peuvent se traduire par une variation des caractéristiques et paramètres dysfonctionnels des équipements, par exemple la mise à jour du taux de défaillance des composants, donc de leur fiabilité, lorsqu’il y a reconfiguration ou changement de mode de dégradation, ce qui peut arriver dans un système dynamique.

L’objectif de Schneider Electric est donc de proposer des voies d’amélioration aux processus actuellement en vigueur :

- en proposant une démarche qui concilie à la fois la pertinence et la qualité des indicateurs fournis (à l’instar des approches par arbres de défaillances) tout en préservant les ressources humaines et financières (notamment en termes de temps d’analyse), c’est à dire sans faire appel à des équipes externes d’experts SdF,
- en intégrant les aspects dynamiques des architectures, notamment en termes de reconfiguration, d’identification de séquences (et non pas d’ensembles) de défaillances conduisant à la défaillance de l’architecture et de variation des paramètres dysfonctionnels des équipements (contexte d’usage, vieillissement, dégradation).

## 2.3. Spécifications pour la nouvelle évaluation sur les performances de SdF

### 2.3.1. Principe de la nouvelle évaluation

Comme abordé précédemment, l’objectif de notre travail est de pouvoir évaluer dynamiquement les performances de SdF d’une architecture de contrôle-commande. Le résultat doit être à la fois adaptée à un contexte d’avant-vente et simple d’utilisation. Parce que les ressources sont limitées et que le comportement dynamique rend complexe la récupération et la compréhension des performances avec les pratiques actuelles, il est nécessaire de développer une nouvel approche pour réaliser cette évaluation.

Cette évaluation va se baser sur trois axes.

Le premier axe est la description informelle de l’architecture de contrôle-commande (cf. Fig. 2. 8). Cette description permet de récupérer la configuration du système, sa structure ainsi que les connexions entre les composants élémentaires qui constituent l’architecture. La conception et la configuration de l’architecture est le résultat de l’interprétation des exigences du client pour son procédé industriel, à travers le cahier des charges disponible lors de l’appel d’offres, par un système intégrateur. Nous y retrouvons la liste des composants, ainsi que leur emplacement dans le système et les liens entre eux.

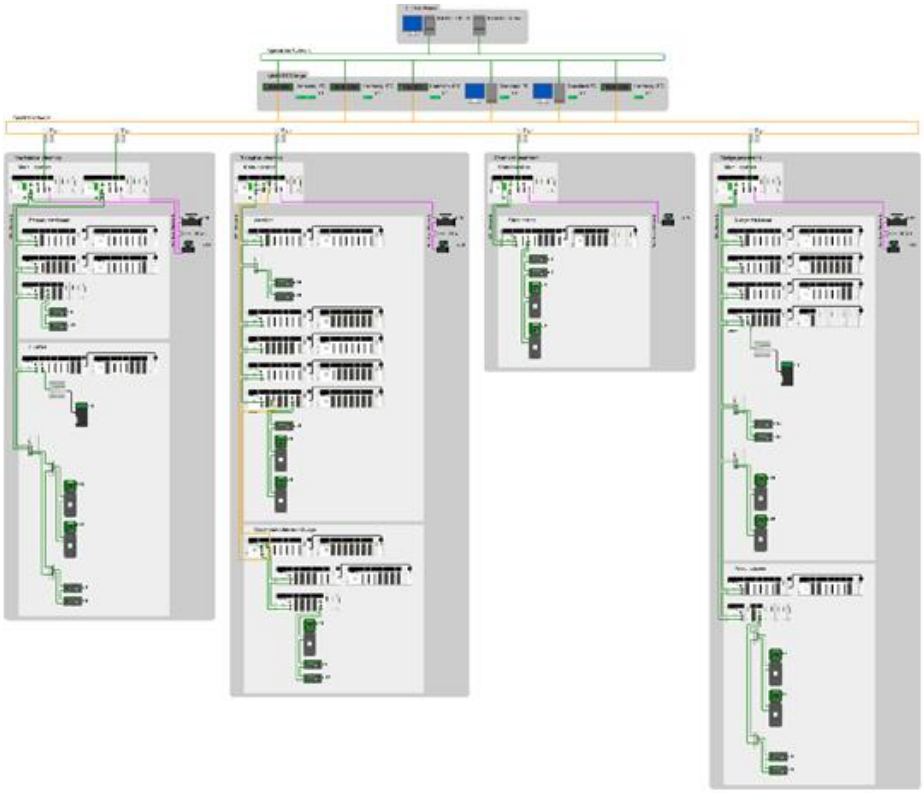


*Fig. 2. 8 : Exemple de description informelle d'une architecture de contrôle-commande*

Le deuxième axe concerne les relations de dépendance dysfonctionnelle entre équipements. Il se base sur une connaissance des produits d'automatisme et de leurs comportements fonctionnels et dysfonctionnels. Ainsi, nous sommes en mesure de connaître quels types de composants peuvent être connectés entre eux, ceux qui sont dépendants d'un autre type (par exemple, un module de communication est dépendant du support sur lequel il se pose pour assurer son alimentation électrique), et comment la panne d'un type de composant impacte les autres. Toutes ces informations sont regroupées dans un bloc que nous allons appeler "spécifications techniques".

Enfin, le dernier axe établit la liste des indicateurs de sûreté de fonctionnement que l'on souhaite estimer dans notre évaluation. Comme évoqué dans le premier chapitre, nous souhaitons estimer :

- La disponibilité de l'architecture, sa moyenne et son évolution au cours du temps,
- La fiabilité du système complétée le temps moyen de fonctionnement,
- Le nombre de défaillances qui arrivent pendant la période d'utilisation et le temps d'indisponibilité incombé au système,
- L'impact des composants sur les performances du système.

À partir de ces trois axes, il faudra être en mesure de construire un modèle qui servira de base à l'évaluation des indicateurs SdF (cf. Fig. 2. 9). Les exigences en termes d'indicateurs ayant été présentées au chapitre 1, la suite de ce chapitre se focalise sur les spécifications relatives aux deux premiers axes.

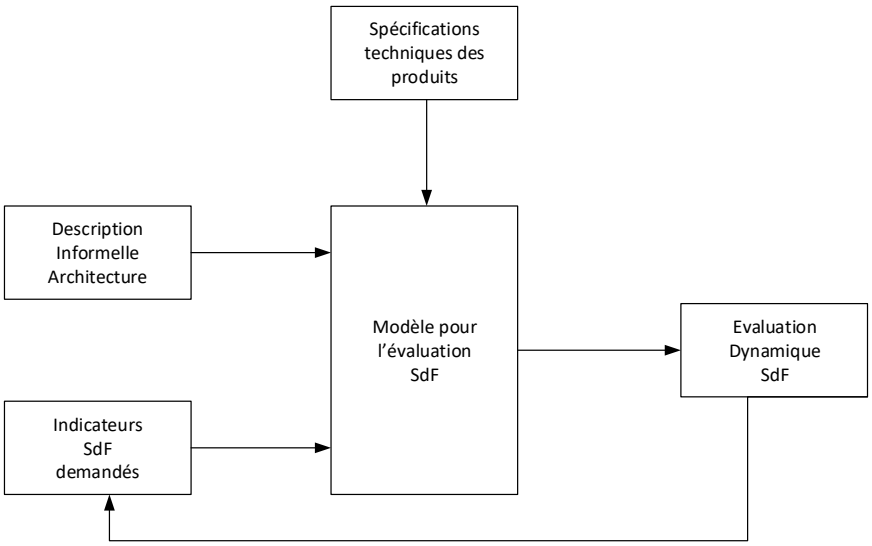



*Fig. 2. 9 : Nouvelle méthode d'évaluation des performances SdF pour une architecture de contrôle-commande*

### 2.3.2. Considérations générales sur la nouvelle évaluation

Nous avons expliqué dans la section précédente, le principe de l'évaluation que nous souhaitons mettre en place pour répondre au besoin industriel, c'est-à-dire déterminer les performances fiabilistes d'un système de contrôle-commande dans le but de piloter un procédé industriel en respectant les exigences de sécurité formulées par le client. Nous verrons un peu plus en détails dans la section suivante, les données d'entrée utilisés pour développer les modèles de description et d'évaluation, notamment les spécifications techniques.

L'évaluation sera développée sous la forme d'une solution intégrée dans un outil de configuration d'architecture, dont la prise en main pourra être réalisée par un utilisateur avec une expertise en SdF pouvant être limitée. Généralement, l'évaluation sera exécutée par un système intégrateur, qui a une connaissance plus ou moins poussé de l'architecture du projet, selon les spécifications données dans le cahier des charges, la traduction du comportement fonctionnel en une liste de matériels.

Ainsi le modèle de description et celui d'évaluation doivent pouvoir donner la possibilité d'évaluer rapidement et facilement les performances fiabilistes, pour respecter les contraintes d'avant-vente. Le processus d'évaluation devra être réalisé de manière transparente à l'utilisateur pour éviter de donner des informations qui ne sont pas nécessaires pour le système intégrateur.

Nous avons vu dans la section précédente que l'approche partira d'une description informelle. Cette description comprend la liste des composants d'automatisme et leurs liens physiques et fonctionnels. Cependant, il faudra considérer les limites que cette description implique, notamment en phase d'avant-vente, où les connaissances de l'environnement dans lequel évolue le système (température, niveau de service, niveau de réparation) ne sont pas forcément connues ou partiellement. Il faudra donc mettre en place une évaluation avec ce peu d'informations connues en entrée, que ce soit pour la description de l'architecture ou les paramètres d'utilisation du système.

L'évaluation devra également prendre en considération le comportement dynamique d'une architecture de contrôle-commande et être en mesure de décrire l'évolution de ce dernier dans le temps, d'un point de vue fiabiliste. Si le système se trouve dans un scénario possible constitué d'un ensemble d'états fonctionnels et dysfonctionnels de ses composants, nous serons en mesure de pouvoir définir l'évolution du système grâce au modèle de description mis en place et aux règles découlant des spécifications techniques.

Tous les types d'architectures d'automatisme possibles, pour tout type d'applications industrielles (traitement des eaux et désalinisation, gestion des transports dont les tunnels ferroviaires, industrie minière et cimenterie, etc.) devront être pris en charge par l'évaluation, sans distinction, ce qui implique que les modèles doivent rester de l'ordre général et ne pas se restreindre à un domaine en particulier. Il est à préciser néanmoins que si le nombre d'architectures potentiellement candidates peut être élevé, le nombre de type de composants d'automatisme est fini, ce qui nous permet de ne pas avoir d'explosions combinatoires au niveau de la description de la structure générale d'une architecture ainsi que des interactions et connexions entre les types de composants. L'instanciation de la méthode doit être rapide, automatique et adapté pour tous types d'architectures (Boyer, Brînzei, Camerini, Ndiaye, & Pétin, 2020).

Au-delà de l'évaluation en elle-même, il faut que les résultats trouvés sur les performances soient complets mais simples à analyser et à interpréter. Les utilisateurs de la nouvelle méthode n'ont pas le même niveau de connaissance en sûreté de fonctionnement, ils peuvent être des experts ou être uniquement en charge de proposer une offre d'architectures et donc de la conception du système. Des recommandations ainsi que des pistes ou aides à la décision seront à fournir en complément, dans le but d'améliorer les performances et de présenter la meilleure architecture répondant aux besoins industriels du client.

En résumé, les considérations générales pour la mise en place de la nouvelle évaluation des performances fiabilistes d'une architecture de contrôle-commande sont (Boyer, Brînzei, Camerini, Ndiaye, & Pétin, 2020) :

- Le développement du modèle d'évaluation à partir de la description informelle donnée par le système intégrateur et des paramètres environnementaux dont la connaissance peut être limitée en avant-vente,
- La sélection pertinente des composants parmi une bibliothèque de composants regroupant tous les types possibles que nous pouvons retrouver dans une architecture et permettant de considérer les exigences en termes de performances fiabilistes définies dans le cahier des charges,
- La possibilité de choisir une architecture parmi un large éventail d'architectures type possible pour gérer le procédé industriel,
- La définition d'une évaluation prenant en compte la dynamique et le comportement dysfonctionnel du système de contrôle-commande afin de garantir la mise en place d'un plan de maintenance, préventif ou correctif, en adéquation avec les ressources disponibles,
- Le type de ressources propres à la phase d'avant-vente, se caractérisant par une expertise limitée dans le domaine de la sûreté de fonctionnement et des évaluations couplée à des

contraintes de temps pour proposer rapidement un design d'architecture et valider ses performances.
- La présentation des performances de façon ergonomique, que ce soit pour un utilisateur expert ou non.

C'est donc sur ces préconisations que la recherche du type de modélisation optimal pour répondre à toutes ces exigences se fera et sera développé par la suite.

### 2.3.3. Données en entrée de la nouvelle évaluation

Nous avons vu précédemment que l'évaluation aura pour point d'entrée la description informelle de l'architecture de contrôle-commande avec un panel de paramètres en lien avec la prise en compte d'aspects fiabilistes (comme la température ou le niveau de réparation).

Cette description reste néanmoins incomplète et dépend fortement du projet et du procédé industriel à piloter ainsi que de la configuration de l'architecture qui en est déduite. Il est donc impératif de mettre en place des règles permettant de décrire de façon générale le comportement fonctionnel et dysfonctionnel d'une architecture, en connaissant les types de composants qui la constituent et les liens physiques qui sont réalisables d'un point de vue technique (David, Idasiak, & Kratz, 2010) (Cressent, David, Idasiak, & Kratz, 2013) (Thiriet, Conrard, & Robert, 2005) (Suiphon, Simeu-Abazi, & Gascard, 2014) (David, Idasiak, & Kratz, 2008).

Ces règles universelles seront ainsi valables pour toutes configurations, pour tous secteurs industriels, et pour toutes contraintes environnementales. Nous avons fait le choix de nous baser sur certaines catégories de diagrammes UML (de l'anglais *Unified Modeling Language*) (Muller & Gaertner, 2000), un langage de modélisation graphique utilisé principalement en informatique notamment en développement logiciel ou en conception orientée objet, mais qui s'avère être un outil intéressant pour décrire notre besoin actuel, car adéquats pour décrire tous les types de configurations et les liens de dépendance que nous pouvons retrouver dans les architecture de Schneider Electric.

Les diagrammes UML utilisés dans notre étude permettront de définir les spécifications techniques complétant la description informelle pour mettre en place le meilleur modèle d'évaluation en adéquation avec les demandes du client industriel. Ils sont totalement transparents à l'utilisateur, ainsi que pendant l'évaluation.

Contrairement au modèle d'évaluation sollicité en arrière-plan de l'outil donné à l'utilisateur (comme le système intégrateur), les diagrammes UML sont uniquement utilisés pour décrire le système en intégrant les aspects fiabilistes afin d'avoir une représentation formelle de ce dernier. Ces diagrammes vont formaliser la structure d'une architecture, avec les diagrammes de classes, ainsi que son comportement, avec les diagrammes d'états-transitions, et les interactions entre les composants, avec les diagrammes de séquence, et pourront donc être réutilisés pour d'autres applications hors sûreté de fonctionnement. Nous verrons dans les sections suivantes les différents types de diagrammes UML.

#### *2.3.3.1. Première entrée : diagrammes de classes*

La structure physique et les connexions possibles entre les types de composants dans l'architecture sont définies avec des diagrammes de classe (Object Management Group, 2015). Ce type de diagramme permet de fournir une représentation de la structure interne du système avec ses composants qui vont interagir pour réaliser les cas d'utilisation. Il est particulièrement adapté pour des cas d'utilisation généraux, pouvant s'appliquer pour tous types d'applications industrielles.

Les diagrammes de classe donnent une vue statique du système. Ils ne tiennent pas compte du comportement au cours du temps de ce dernier. Ici, ils nous permettent uniquement d'apprendre comment l'architecture se structure lorsque le système est opérationnel et quels composants peuvent être liés entre eux. Il s'agit d'une capture à un instant fonctionnel de la composition de l'architecture. Les interactions et les changements d'état seront donnés par d'autres diagrammes UML.

Les familles de composants sont représentées par des classes, un ensemble de fonctions et de données sous la forme d'attributs qui peuvent être définis ou non selon les besoins. Ces classes sont liées entre elles par des liens entre les composants quand la communication est possible entre deux composants. Nous utilisons également l'héritage, qui permet de donner les attributs d'une classe dite mère à une autre classe dite fille (par exemple dans la Fig. 2. 10, les classes I/O module et Expert module héritent des attributs de la classe Process module). Ici, nous considérons l'héritage comme la définition de types secondaires à un type de composant. Ainsi les modules d'E/S (I/O module) et les modules expert (Expert module) sont des sous-catégories du type Process module.

Pour plus de lisibilité, les diagrammes de classe sont structurés par localisation. Un diagramme de classe élémentaire correspond à une localisation qui comprend un support d'alimentation de modules/composants (en anglais *Backplane*) et à tous les composants que nous pouvons assigner sur celui-ci. Par exemple, dans la Fig. 2. 10, il s'agit de la localisation Primary Rack correspondant au rack où se situe le processeur de l'API (en anglais *CPU*) et de tous les modules qui peuvent se placer potentiellement sur le même rack que ce dernier.

Les diagrammes de classes élémentaires dont les classes correspondent à une famille de composants. Les diagrammes de classe non élémentaires sont les diagrammes dont les classes peuvent correspondre également à des localisations (ou emplacements). Certaines localisations sont contenues dans une autre localisation d'un niveau supérieur, englobant plusieurs localisations. Dans la Fig. 2. 11, le diagramme de classes pour la localisation Control Unit est constitué de classes à la fois pour les composants de la famille Device mais également de classes pour des localisations comme la *Local Area* qui est en réalité un diagramme, celui de la Fig. 2. 10. Le diagramme pour la *Control Unit* est un diagramme non élémentaire. Il est possible d'avoir plusieurs fois le même type de localisation, ce qui est décrit grâce à la multiplicité sur les liens d'association (nous pouvons avoir plusieurs *Remote Area* connecté à la *Local Area* par exemple dans une *Control Unit*, comme c'est affiché dans la Fig. 2. 11).

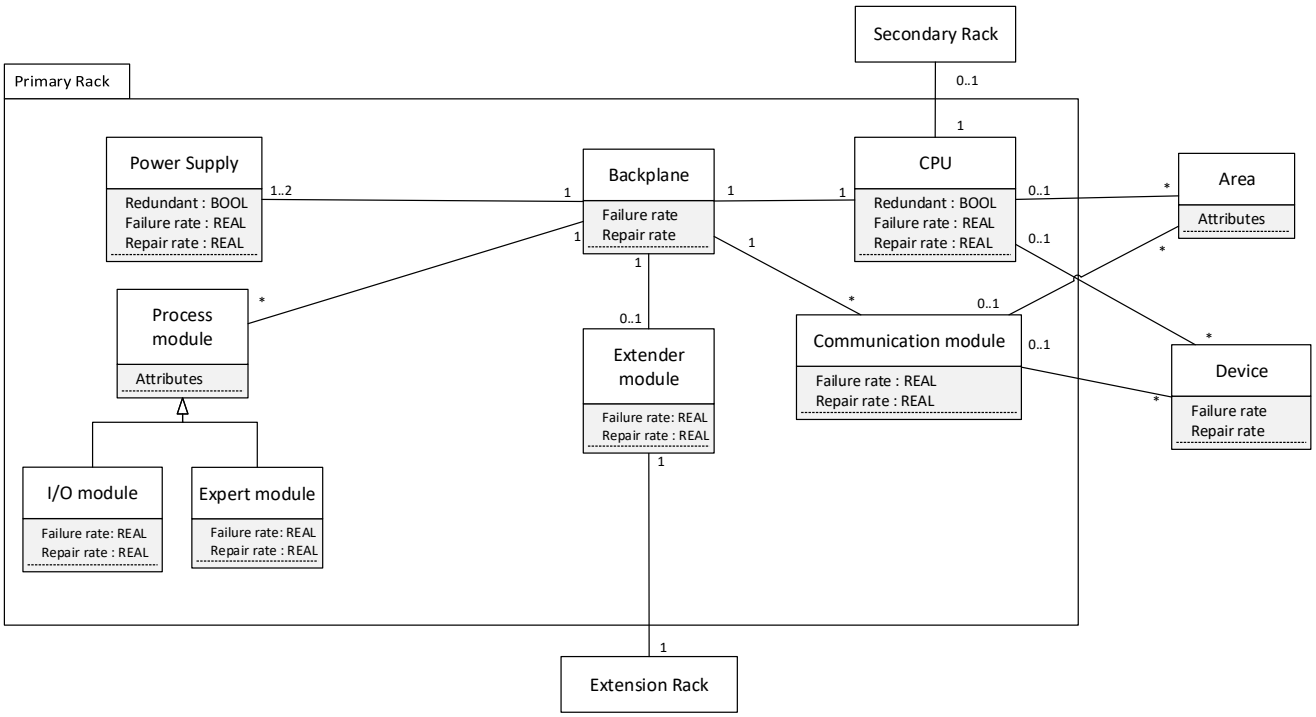



*Fig. 2. 10 : Exemple de diagramme de classe pour la description générale d'une architecture de contrôle-commande*

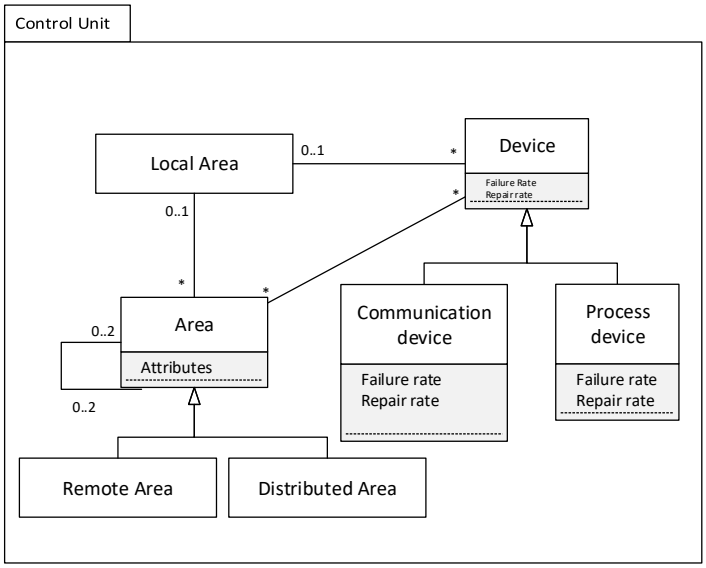



*Fig. 2. 11 : Diagrammes de classes non élémentaire dans notre étude*

Il est à noter que les composants qui sont répertoriés dans les diagrammes de classe sont essentiels pour la prise en compte des aspects de sûreté de fonctionnement de l'architecture de contrôle-commande. Nous avons écarté les composants qui n'ont pas d'impact sur les performances fiabilistes ou qui ne font pas partie des niveaux fonctionnels considérés (cf. Chapitre 1). Ainsi les diagrammes actuels ne sont pas une représentation fidèle de système physique mais une partie de ce dernier dans le cadre de notre étude.

D’un point de vue fiabiliste, ces diagrammes de classe vont nous permettre de (Boyer, Pétin, Brînzei, Camerini, & Ndiaye, 2019):

- Déterminer le niveau de redondance possible des composants grâce à la multiplicité qui est affiché sur l’association entre deux types de composants,
- Déterminer les ensembles de composants qui pourraient être impactés pas une défaillance de cause commune (DCC), par exemple de savoir tous les types de composants qui sont touchés par la panne du module d’alimentation,
- Déterminer les composants qui seront impactés par une défaillance sur un composant, par effet de cascade, ainsi que les localisations concernées, par exemple le processeur CPU entraine la dégradation des composants dont il a le contrôle.

#### *2.3.3.2. Deuxième entrée : diagramme d’états-transitions*

Nous venons de voir que les diagrammes de classe permettent de définir les spécifications techniques concernant la structure et les liens physiques entre les composants dans une architecture de contrôle-commande.

Nous devons également pouvoir être en mesure de décrire les différents états dans lequel le composant peut se placer avant de voir comment il interagit avec le reste des modules. Le diagramme d’états-transitions nous servira à décrire cela.

Le diagramme d’états-transitions représente le comportement interne du composant à l’aide d’un automate à états finis. Il permet de montrer les transitions possibles entre les états et les actions ainsi que les conditions pour réaliser le changement.

Ici, seuls les états ayant une importance pour la prise en compte des notions de sûreté de fonctionnement sont considérés. Ainsi des états comme l’état en attente d’activation *idle* d’un module pour exécuter une fonction de lecture d’un programme ne sont pas décrits car n’apportant pas d’information pour la suite de l’évaluation des performances fiabilistes.

Selon la famille du composant, que nous retrouvons dans les diagrammes de classes, le diagramme d’états-transitions, dans la Fig. 2. 12, sera adapté pour ne prendre que les états dans lequel le composant peut se mettre, ce modèle étant générique pour tous les types de modules d’automatisme rencontrés à Schneider Electric. Le principe reste néanmoins similaire pour d’autres cas d’applications, à savoir lister les états possibles.

Dans le cadre de notre travail, trois états principaux sont listés :

- L’état en marche *Powered* où le composant est alimenté électriquement et en mesure de fonctionner,
- L’état à l’arrêt *Unpowered* où le composant est éteint et n’est pas alimenté, principalement à cause d’une coupure au niveau de l’approvisionnement en électricité (sur la source d’alimentation ou sur le canal),
- L’état de défaillance *Failed* où le composant est tombé en panne à un instant donné et se met en réparation ou est remplacé si une pièce de rechange est disponible, pour redevenir opérationnel.

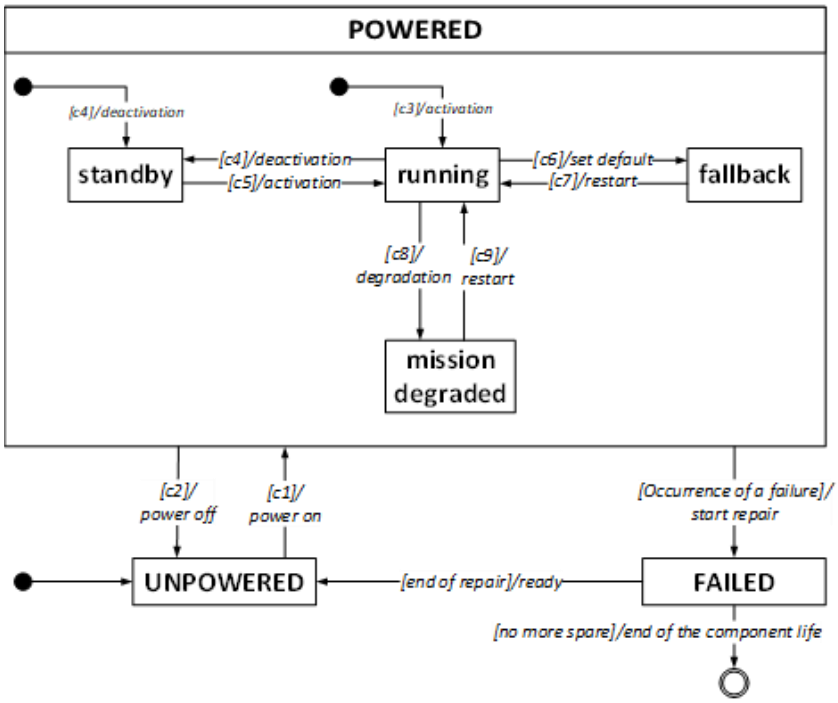



*Fig. 2. 12 : Diagramme d'états-transitions pour les états d'un composant dans une architecture de contrôle-commande*

En addition de ces états principaux, nous avons besoin de définir des sous-états qui permettent de décrire plus précisément la situation du composant et sa capacité à réaliser les missions qui lui sont attribuées ou le statut de la communication avec le reste du système. Ces sous-états se trouvent dans l'état principal *Powered* et sont :

- L'état de marche opérationnel *Running* où le composant fonctionne dans les conditions optimales et réalise toutes ses missions sans dégradation,
- L'état en redondance chaude *Standby* où le composant est en attente pour prendre le relais à la suite de la défaillance du composant principal,
- L'état en position de repli *Fallback* où le composant a perdu les instructions nécessaires pour réaliser ses fonctions correctement et se met dans un état par défaut pour fonctionner de façon minimale,
- L'état de dégradation des missions du composant *Mission Degraded* où le composant se retrouve à performer des missions qui ne respectent plus intégralement les besoins.

Les changements d'état sont primordiaux dans l'évaluation des performances fiabilistes. En effet, la transition vers un nouveau statut pour un module engendre des impacts sur le reste du système, et peut mener vers un dysfonctionnement ou vers une défaillance. Des incidents que nous souhaiterions minimiser pour améliorer la robustesse du système. Ces transitions sont conditionnées par deux sortes d'événements :

- Un événement interne, qui dépend du composant même. Il s'agit principalement de sa défaillance, le faisant passer de l'état *Powered* à l'état *Failed,* ou de la fin de sa réparation ou son remplacement, passant de l'état *Failed* à l'état *Unpowered*

- Un événement externe, dont l'origine comme son l'indique provient d'un autre composant de l'architecture impactant le composant en question. Par exemple, la communication vers le module est interrompue car un des composants assurant le canal est tombé en panne.

#### *2.3.3.3. Troisième entrée : diagrammes de séquence*

Les performances de sûreté de fonctionnement de l'architecture de contrôle-commande dépendent fortement de l'état de ses composants au cours du temps. C'est pourquoi, il est nécessaire, en plus de la structure du système et des liens physiques entre les composants, de définir les états dans lesquels ces derniers peuvent être.

Cependant, ce ne sont pas les seules spécifications techniques dont nous avons besoin. Il faut décrire les interactions entre les différents acteurs, comment s'agencent les événements et les changements d'état, quelle est l'origine de ces transitions. Pour définir ces interactions, nous avons utilisé les diagrammes de séquences, dernière catégorie de diagrammes UML utilisés pour notre étude.

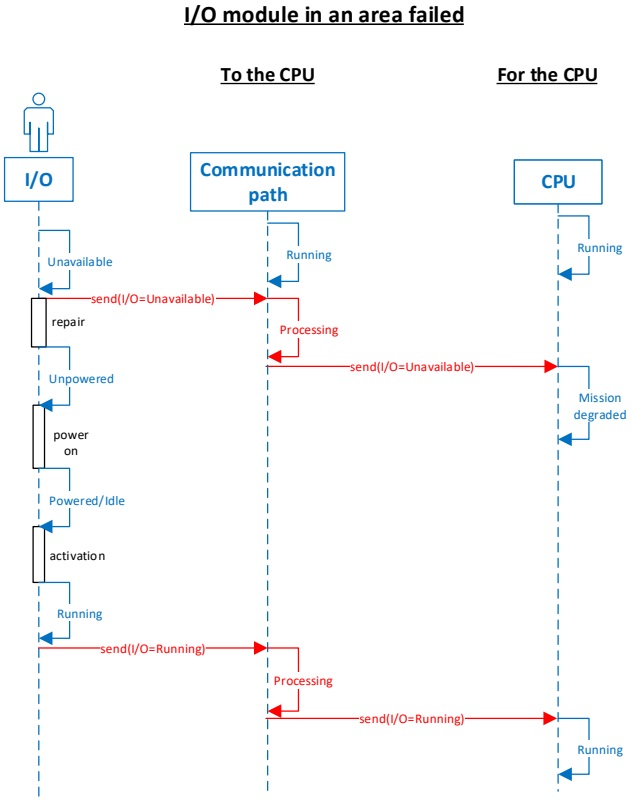



*Fig. 2. 13 : Diagramme de séquences dans le cas où un module d'entrée-sortie est défaillant*

Les diagrammes de séquence sont une représentation graphique de ces interactions entre les composants, les acteurs, selon un ordre chronologique avec la succession des événements tels qu'ils apparaissent au cours du temps, les messages (cf. Fig. 2. 13). Ils donnent la série de changements d'états pour les composants à partir d'un composant source qui se trouve à la tête du diagramme et conditionnent donc les transitions des modèles d'états de ces composants (*événements externes*).

A la différence des diagrammes de classes, les diagrammes de séquence se concentrent sur la dynamique des séquences dysfonctionnelles du système en identifiant les conséquences de la défaillance d'un composant sur le reste de l'architecture.

Les informations données par les diagrammes de séquence doivent être pris en compte dans l'évaluation des indicateurs fiabilistes étant donné que les interactions entre les composants peuvent diminuer ou dégrader les performances du système de contrôle-commande.

### 2.4. Conclusion

Ce chapitre a posé les bases de la méthode à rechercher pour la nouvelle évaluation en formalisant les besoins industriels auxquels nous sommes confrontés.

Nous avons répertorié en premier lieu les pratiques et méthodes d'évaluation utilisées actuellement pour en dégager des pistes d'amélioration qui font l'objet de ce travail de thèse autour de la proposition d'une démarche d'évaluation d'indicateurs SdF basée sur des modèles dynamiques mais ne nécessitant pas le recours à des experts incompatible avec les contraintes présentes en phase d'avant-vente.

Nous avons ensuite listé les données disponibles en entrée, à savoir une description informelle reprenant la liste des composants de l'architecture à évaluer ainsi que sa configuration et les liens physiques et de communication entre les modules. Cette description est complétée par des paramètres environnementaux permettant de donner des informations complémentaires jugées utiles pour la prise en compte des aspects de SdF d'un système de contrôle-commande. À cela s'ajoute des spécifications techniques sous la forme de diagrammes UML permettant de définir des règles pour tous types d'architectures, quel que soit l'application industrielle en œuvre :

- Définir la structure du système et les liens possibles entre les différents types de composants, avec les diagrammes de classe,
- Définir les états dans lequel un composant peut se mettre, avec les diagrammes d'états-transitions,
- Définir les interactions entre les composants et comment s'ordonne les événements dans le temps, avec les diagrammes de séquence.

C'est en partant de ces spécifications que nous allons maintenant voir quelles modèles d'évaluation dans la littérature scientifique permettent de répondre à ces spécifications et de mettre en place l'estimation la plus appropriée pour notre étude, avant d'expliquer comment le modèle choisi sera développé en milieu industriel.

# Chapitre 3
# État de l’art

## 3.1. Introduction

Les chapitres précédents ont permis de mettre en place le contexte de l’étude ainsi que les besoins industriels qui y sont associés. Nous devons maintenant déterminer quelle approche pour l’évaluation est la plus pertinente dans ce contexte.

Ce troisième chapitre dresse donc un état de l’art des différentes méthodes d’évaluation de performances de sûreté de fonctionnement (SdF) sur un système de type architecture de contrôle-commande selon trois axes :

- Modélisation des architectures,
- Formalisation des indicateurs,
- Méthode d’évaluation de ces indicateurs.

Pour chacune de ces approches, nous expliquerons succinctement leur fondement théorique et nous présenterons leurs apports et limites dans notre contexte d’étude. Une synthèse permettra de déterminer son adéquation avec nos besoins et de sélectionner le choix le formalisme utilisé pour la modélisation, les indicateurs retenus et la méthode d’évaluation à mettre en place.

## 3.2. Modélisation dysfonctionnelle d’une architecture de contrôle-commande

L’évaluation des performances SdF des architectures repose en premier lieu sur la modélisation de ces dernières. Nous avons besoin de modéliser leur comportement fonctionnel mais surtout dysfonctionnel. La modélisation permet de voir comment réagit le système face à un événement perturbateur, comme les défaillances des composants qui le constituent. Elle permet également d’avoir une première compréhension du système d’un point de vue SdF et de prendre en compte les aspects liés à cette science.

Les prochaines sous-sections seront consacrées à la recherche du formalisme de modélisation dysfonctionnelle le plus approprié dans notre étude.

### 3.2.1. Modèles booléens

Une première catégorie pour modéliser les architectures de contrôle-commande sont les formalismes basés sur une représentation booléenne des systèmes.

#### *3.2.1.1. Diagramme de fiabilité et arbre de défaillances*

Deux formalismes booléens, les arbres de défaillances et les diagrammes de fiabilité, ont déjà été abordés dans le chapitre 2 (sections 2.2.2 et 2.2.3) dans la mesure où ils font partie des solutions déjà mises en œuvre chez Schneider Electric.

Comme il a été dit précédemment, ces approches sont statiques et ne répondent que partiellement aux attentes exprimées au chapitre 2. Des extensions « dynamiques » à ces formalismes de représentation ont été introduits. En particulier, les arbres de défaillances dynamiques (Fussell, Aber, & Rahl, 1976) (Cepin & Mavko, 2002) (Ruijters & Stoelinga, 2015) (Gascard & Simeu-Abazi, 2018) ont été développés dans l'optique d'affiner les analyses de fiabilité des systèmes industriels. Comme pour la version statique, il s'agit toujours de déterminer les combinaisons d'événements qui entrainent l'évènement redouté, tel que la panne du système complet. La principale différence réside dans le fait que l'événement redouté n'est plus vu comme la résultante d'un ensemble d'événements de défaillance des composants mais comme une séquence ordonnée sur ces événements (Rauzy, 2011). Cela se traduit par l'utilisation de portes logiques qui ne se limiteront plus aux portes logiques classiques mais à des portes dynamiques permettant de représenter les changements dépendants du temps.

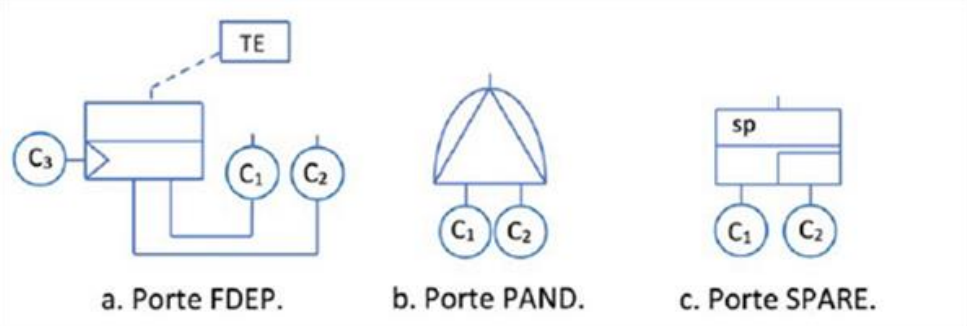



*Fig. 3. 1 : Portes dynamiques pour les AdD (Duroeulx, 2020)*

Les trois principales portes sont (cf. Fig. 3. 1) :

- La porte FDEP (Functional-DEPendency), où l'activation d'un événement appelé gâchette, située à gauche de la porte, permet d'activer les entrées de la porte logique (Dugan, Bavuso, & Boyd, 1990). Ceci se fait par exemple dans le cas d'une alimentation commune à deux composants, si l'alimentation est coupée (gâchette), les deux composants sont en panne (entrées de la porte), et on passe la porte FDEP.
- La porte PAND (Priority-AND), où l'ordre d'occurrence des défaillances est représenté (Fussell, Aber, & Rahl, 1976). La porte PAND s'active si tous les entrées s'activent dans un ordre précis, de la gauche vers la droite, suivant leur connexion à la porte.
- La porte SPARE (*de rechange* en français) qui permet de prendre en compte, comme son nom l'indique, les pièces de rechange pour les composants du système (Dugan, Bavuso, & Boyd, 1990). Le composant le plus à gauche est le composant principal, les autres composants sont à la suite, au niveau de l'entrée de la porte.

L'évaluation probabiliste des indicateurs de sûreté de fonctionnement d'un système à partir d'un modèle d'arbre de défaillances dynamique peut se faire soit par une méthode analytique (Duroeulx, 2020) soir par la simulation de Monte-Carlo (Gascard & Simeu-Abazi, 2018) (Simeu-Abazi, Gascard, & Sidqi, 2015).

Les AdD dynamiques peuvent modéliser les architectures de contrôle-commande en considérant les séquences de dépendances dysfonctionnelles entre composants. En revanche, ils ne permettent pas de représenter les modifications opérées au cours du temps sur ces séquences, par exemple suite à une défaillance et une reconfiguration de l'architecture, ni la prise en compte de paramètres fiabilistes variables au cours du temps ou les opérations de maintenance car ils sont essentiellement orientés vers l'étude des systèmes non-réparables. Ces approches ne seront donc pas retenues.

#### *3.2.1.2. Fonction de structure*

La fonction de structure est une expression booléenne permettant de modéliser le fonctionnement d'un système. En règle générale, il est possible de décrire sous la forme d'une fonction de structure, un diagramme de fiabilité ou un arbre de défaillances, ayant les mêmes bases logiques. La fonction de structure est simplement une représentation analytique de ces derniers (Kaufmann, Grouchko, & Cruon, 1975) (Cocozza-Thivent, 1997) (Rausand & Hoyland, 2003).

Etudier le système grâce à cette fonction permet de déterminer l'état du système à partir de l'état de ses éléments. Il est possible de se focaliser soit sur le fonctionnement, comme pour les diagrammes de fiabilité, soit sur le dysfonctionnement, avec les arbres de défaillances (Duroeulx, 2020) (Aubry & Brinzei, 2016) .

Considérons un système S constitué d'un ensemble de *n* composants (noté $C = \{c_1,\ldots,c_n\}$), un ensemble de variables booléennes $x_i \in \{0,1\}$ représentant l'état du composant $c_i$ (qui valent 1 si le composant fonctionne, et 0 sinon), un n-uplet $X = (x_1,\ldots,x_n)$ représentant l'état de l'ensemble des n composants du système S et *y* une variable booléenne représentant l'état du système global S (égale à 1 si le système fonctionne, à 0 sinon). La fonction de structure $\phi$ est la fonction qui à chaque valeur de X associe une valeur $y = \varphi(X) \in \{0,1\}$. Elle est d'ordre k si elle dépend de k variables, c'est-à-dire de k composants.

L'expression de la fonction se base sur des relations d'ordre entre les composants et peut être représentée par un diagramme de Hasse, comme dans la Fig. 3. 2, dans lequel un arc entre deux valeurs prises dans X, montre l'existence d'une telle relation (Aubry, et al., 2016). L'intérêt de ce diagramme et de la fonction de structure liée à ce dernier, est que les arcs du graphe peuvent être considérés comme l'arrivée d'une défaillance ou d'une réparation d'un composant lorsque l'on parcourt le graphe de haut en bas ou vice versa. Les traits pleins représentent les combinaisons d'état des composants où le système S fonctionne, les traits en pointillés sont les combinaisons d'état des composants où le système S ne fonctionne pas.

Pour un système avec n composants en série, l'expression de la fonction de structure est :

$$\varphi(X) = \varphi(x_1, \ldots, x_n) = y = x_1 * \ldots * x_n = \prod_{i=1}^{n} x_i \qquad (3.1)$$

Pour un système avec n composants en parallèle, la fonction de structure sera exprimée de la façon suivante :

$$\varphi(X) = \varphi(x_1, \ldots, x_n) = y = 1 - (1 - x_1) * \ldots * (1 - x_n) = 1 - \prod_{i=1}^{n} (1 - x_i) \qquad (3.2)$$

Dans le cadre de l'exemple représenté dans la Fig. 3. 2, en ne prenant que les n-uplets X qui aboutissent à la défaillance du système constitué ici de trois composants, après les simplifications que nous pouvons faire en se basant sur l'algèbre de Boole, nous avons la fonction de structure suivante :

$$\varphi(X) = \varphi(x_1, x_2, x_3) = y = [1 - (1 - x_1)(1 - x_2)][1 - (1 - x_1)(1 - x_3)] \qquad (3.3)$$

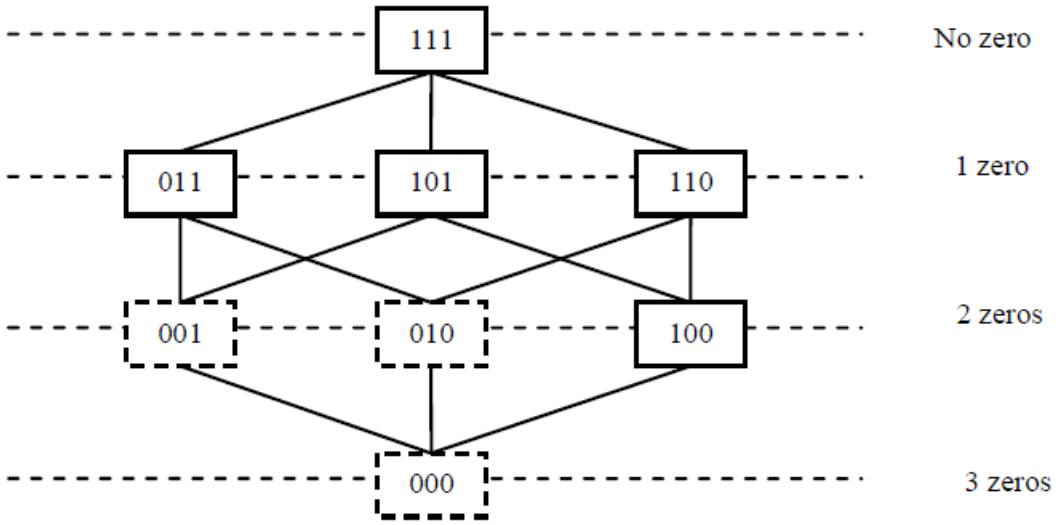



*Fig. 3. 2 : Diagramme de Hasse d'une fonction de structure pour un système avec trois composants (Aubry & Brinzei, 2016)*

Nous avons vu que, dans le cas des diagrammes de fiabilité et des arbres de défaillance, il n'était pas envisageable d'utiliser ces représentations parce qu'ils n'étaient pas adaptés non seulement pour des systèmes dynamiques comme les architectures de contrôle-commande que nous devons évaluer, mais également parce qu'ils ne sont pas adaptés à un contexte incertain d'avant-vente. Dans la mesure où la fonction de structure est simplement une représentation analytique de ces deux modèles, pour les mêmes raisons, elle ne sera pas parmi les choix possibles de modélisation des architectures de contrôle-commande pour notre étude.

### 3.2.2. Modélisation à états et à paramètres stochastiques

Dans cette section, nous allons nous intéresser aux modèles à états et à paramètres stochastiques, qui vont nous permettre de décrire une plus grande gamme de systèmes, mais également nous permettre de décrire des composants dont le comportement ne se limite pas à une simple combinaison binaire.

#### *3.2.2.1. Chaînes de Markov*

Une première représentation possible sont les chaînes de Markov (CdM) (Rubino & Sericola, 2014) (Cocozza-Thivent, 1997). Il s'agit d'une modélisation pour des systèmes à événements discrets permettant de modéliser des systèmes dynamiques incluant des réparations, des défaillances ou des reconfigurations qui surviennent dans le temps.

Les CdM sont un modèle d'états-transitions modélisant le comportement d'un système, de telle sorte que la connaissance de l'état des composants à un instant donné permet de déterminer les évolutions possibles du système. Dans ce cadre, le comportement dysfonctionnel du système est considéré comme un processus stochastique respectant la propriété de Markov : son évolution future ne dépend que de son état présent, et non de ses états passés. C'est ce que nous appelons plus communément d'un processus sans mémoire.

Ce processus est appelé chaîne de Markov à temps discret si l'intervalle temporel d'observation est sur un ensemble discret, et chaine de Markov à temps continu si l'intervalle est un ensemble continu (Ionescu, 2016). Généralement, et plus particulièrement dans le domaine de la SdF les chaînes de Markov à temps continu sont celles qui sont les plus largement utilisées.

Une CdM repose ainsi sur une suite de variables aléatoires qui permet de modéliser l'évolution dynamique d'un système à comportement stochastique. X(t) représente l'état discret du système à l'instant t.

Les transitions entre les différents états possibles sont stochastiques et dépendent des états d'entrée, ainsi que des taux de défaillance et de réparation caractérisant les composants dans le système. Dans la Fig. 3. 3, nous avons un système basique qui peut se mettre dans trois états possibles, par exemple un état de marche (état 1), un état de dégradation partielle (état 2), et un état de panne (état 3). Ici les transitions entre les états sont conditionnées par les taux de défaillance ou de réparation :

- $\lambda_{12}$ entre les états 1 et 2. Cela peut être un changement réalisé à la suite d'une défaillance sur un composant du système qui le rend partiellement dégradé. Le taux de défaillance $\lambda_{12}$ ici serait le taux de défaillance du composant dans le système qui amènerait le système vers un état de dégradation.
- $\lambda_{23}$ et $\lambda_{32}$ entre les états 2 et 3. $\lambda_{23}$ serait un taux de défaillance d'un composant critique par exemple qui ramènerait le système d'un état dégradé à un état de panne totale. $\lambda_{32}$ s'apparenterait ici à un taux de réparation qui amènerait le système d'un état défaillant total vers un état de marche partielle.
- $\lambda_{31}$ entre les états 3 et 1. Cela représente le cas où le système est totalement réparé, après avoir été en panne totale, et est remis en marche directement. C'est donc un taux de réparation.

Bien que ce type de modélisation soit plus riche pour décrire le comportement d'un système (réparable, subissant des reconfigurations durant son exploitation) que les formalismes de la section sur les modèles dysfonctionnels statiques, il n'est pas envisageable d'utiliser les CdM pour représenter les architectures de contrôle-commande que nous souhaitons évaluer, et ce pour les raisons suivantes :

- L'hypothèse où les CdM sont des processus sans mémoire implique que les phénomènes de vieillissement ne sont pas pris en compte. Dans le cas où les architectures de contrôle-commande possèdent des composants qui se dégradent au cours du temps, comme les modules dédiés à l'alimentation, il ne sera pas possible de les décrire à partir de ce formalisme.

- L'explosion combinatoire est un problème qui peut survenir rapidement. Les systèmes complexes comme les architectures de contrôle-commande ont bien souvent un grand nombre d'états possibles et il serait difficile de les lister un par un en avant-vente, en plus de déterminer les transitions possibles et les changements possibles entre ces états.

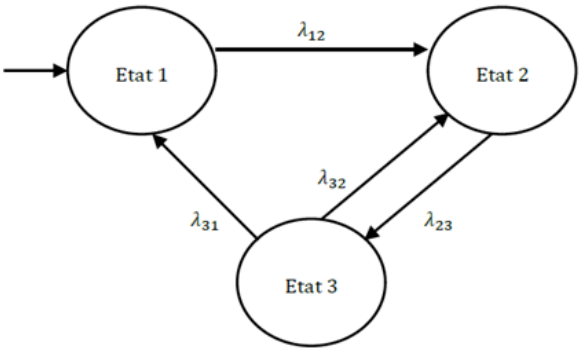



*Fig. 3. 3 : Chaine de Markov à temps continu d'un système pouvant avoir trois états (Ionescu, 2016)*

Du fait de ces limites, les CdM ne seront donc pas un choix dans la modélisation des architectures de contrôle-commande, ne pouvant répondre aux besoins formulés dans le chapitre précédent.

#### *3.2.2.2. Boolean Logic Driven Markov Process (BDMP)*

Les BDMP sont un formalisme qui allie les avantages des arbres de défaillances avec celles des chaînes de Markov (Piriou, 2015) (Bouissou, 2003) (Bouissou, 2008) (Piriou, Faure, & Lesage, 2017). Il permet de prendre en compte les aspects dynamiques du comportement du système et de conserver également une représentation graphique simple. A travers ce formalisme, il est possible de réaliser des analyses quantitatives sur des systèmes dynamiques réparables.

Un BDMP a pour base un arbre de défaillances dont les variables booléennes sont remplacées par des processus de Markov. Ces processus intègrent la représentation dynamique du comportement du système, et donne l'opportunité de considérer les défaillances et réparations survenant sur ce dernier. En complément de ces processus, des mécanismes de reconfiguration et commutation sont possibles grâce à un ensemble de gâchettes, comme décrit dans les arbres de défaillances dynamiques avec les portes FDEP.

Un BDMP est constitué des éléments dans la liste suivante :

- Un arbre de défaillances.
- Un événement top principal r, correspondant à l'événement redouté par analogie à un arbre de défaillances. Généralement, il s'agit de la panne du système global.
- Un ensemble de gâchette dans le modèle possible.
- Un ensemble de processus de Markov $P_i$ associés aux événements de base de l'arbre. Chaque processus est défini par deux catégories d'états : en marche et en panne.

Dans la Fig. 3. 4, nous retrouvons la structure logique d'un arbre de défaillances classique, complété d'une gâchette entre les portes G1 et G2, dont l'origine est la porte G1 et la cible la porte G2. L'événement top principal dépend de quatre processus de Markov, avec la gâchette qui joue un rôle dans l'activation de la défaillance des processus P3 et P4.

Avec les BDMP, nous conservons une modélisation dont la nature reste simple, grâce à sa représentation en arbres de défaillances. Nous sommes en mesure également de modéliser les composants de façon intrinsèque grâce aux processus de Markov, dont chaque modèle peut décrire un composant. Avec les mécanismes de gâchette, il est possible également de prendre en considération des mécanismes de commutation pour modéliser des reconfigurations dans le système à la suite d'une défaillance d'un composant, ou même de considérer des dépendances entre les composants, comme l'alimentation électrique d'un composant dépendant de l'état d'un autre.

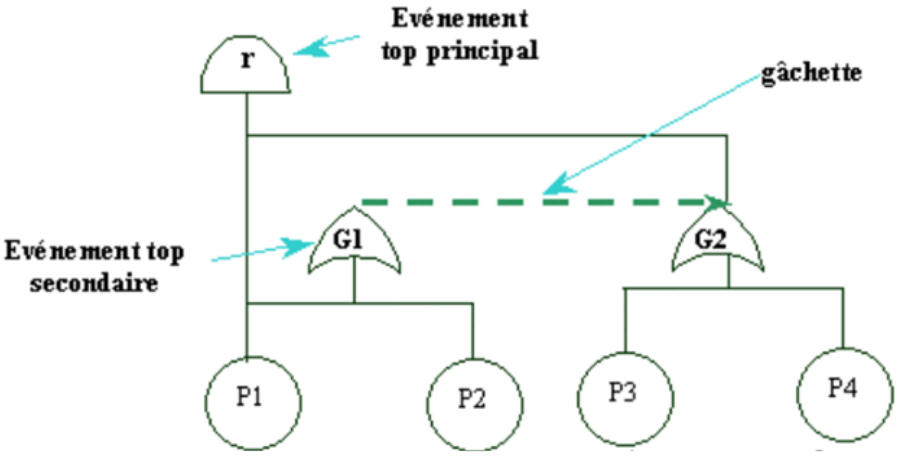



*Fig. 3. 4 : Exemple de BDMP (Bouissou, 2006)*

L'extension à un BDMP dit généralisé a été développé pour enrichir le modèle et apporter des précisions sur les gâchettes et leurs mécanismes de commutation ainsi que leurs effets sur la sûreté de fonctionnement des systèmes (Piriou, 2015). Il permet d'étendre le principe en prenant en compte tous les scénarios de reconfiguration possibles, en ne limitant plus les modes d'opération des composants à deux états. Désormais, il est possible de faire intervenir des composants ayant un nombre supérieur à deux de modes.

Le BDMP est un formalisme de modélisation qui peut être utilisé dans les cas suivants :

- Redondance passive. Un composant reste en mode passif tant que le composant principal est en marche. Lorsque le composant principal tombe en panne, une gâchette est activée et le composant de secours prend le relai à l'instant de défaillance du premier (**Erreur ! Source du renvoi introuvable.**). Il est également possible de définir un temps de basculement si le changement n'est pas réalisé de manière instantanée en posant des conditions et des portes logiques supplémentaires.
- Report de charge. Un composant voit sa charge augmenter quand un autre composant tombe en panne. Par exemple dans le cas où deux composants se

partagent la tâche d'alimenter électriquement d'autres composants du système, et fonctionnent chacun à 50%. Si l'un des deux composants tombe en panne, celui qui est encore en marche passe à une charge de 100%.

- Système multiphase. Dans le cas où un système possède plusieurs phases par rapport à son évolution au cours du temps. Les critères d'arrêt évoluent et changeront en fonction de l'état des composants et des événements étant arrivés sur le système.
- L'ordre des événements est à prendre en compte. Dans le cas où l'occurrence d'un événement $E_i$, selon sa nature et l'origine de ce dernier, il n'y aura pas les mêmes conséquences sur le système qui pourra se retrouver dans un état différent. Au lieu de passe dans un état de panne, il pourrait se mettre en état dégradé par exemple, si la défaillance a eu lieu sur composant $C_i$ plutôt que sur le composant $C_j$.
- Défaillance de cause commune (DCC). Il est possible de considérer ces types de défaillance en définissant un bloc séparé modélisant la panne en commun de plusieurs composants dépendants. La DCC entraîne, grâce à l'activation d'une gâchette, la défaillance simultanée des composants qui sont concernés par la panne.

Nous voyons à travers tous ces cas d'utilisation que les BDMP est un puissant formalisme de modélisation et permet de répondre à de nombreux problèmes de modélisation que nous pouvons rencontrer avec les systèmes dynamiques.

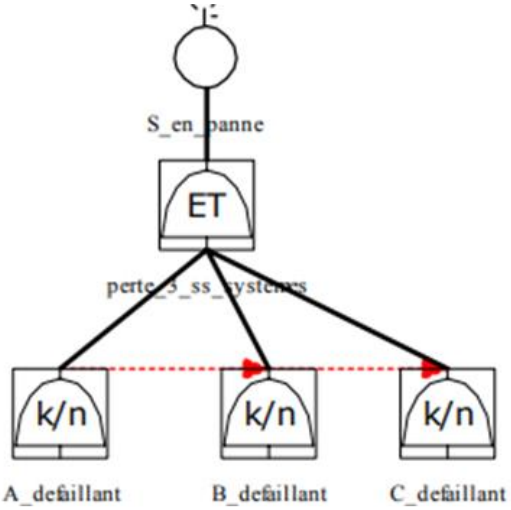



*Fig. 3. 5 : BDMP pour une redondance passive (Bouissou, 2006)*

Bien que ce formalisme puisse être appliqué pour décrire les systèmes du type des architectures de contrôle-commande, il nous paraît fastidieux de faire appel à cette modélisation dans notre cas d'étude. Le travail en amont à réaliser pour pouvoir décrire le système selon la liste des composants qui le constituent est trop important pour être applicable dans le contexte d'avant-vente.

Le choix ne serait pas judicieux, sachant que l'architecture et sa configuration ne sont pas connues d'avance et peuvent rapidement être modifiées pour se conformer aux demandes du client ce qui impose de reconstruire un nouvel BDMP à chaque modification. Dès lors, sachant les contraintes de temps et la nécessité de répondre rapidement au client, il ne sera pas envisageable d'utiliser les BDMP dans notre approche.

#### *3.2.2.3. Automates stochastiques hybrides*

Les automates stochastiques hybrides (ASH) peuvent être vus comme une fonction de structure étendue à un système dynamique (Perez Castaneda, 2009) (Babykina, Brînzei, Aubry, & Deleuze, 2016) (Broy, 2014) (Perez Castaneda, Aubry, & Brînzei, 2011) (Aubry & Brînzei, 2015). Il s'agit d'un automate composé d'un ensemble d'états discrets. Il est dit hybride car chacun de ces états est défini par un système d'équations qui sont continues et par un ensemble de transitions de sortie de ces états définis par des seuils sur les variables en continues. Enfin, il est stochastique car des événements aléatoires décrits par des lois de probabilités peuvent également être pris en compte lors du franchissement de transitions.

Un exemple d'ASH est représenté dans la Fig. 3. 6. À chaque état discret $X^i$, on associe une équation différentielle $\dot{x} = f_i(x,t)$ qui décrit l'évolution des variables continues $x$ dans l'état discret $X^i$. Un arc $A_i$ représentant une transition est défini par deux états discrets représentant l'état de départ et l'état cible de la transition, par un événement $e_i$, une condition de garde $G_i$ (représentant le franchissement d'un seuil par la variable continue lorsqu'il s'agit d'une transition déterministe) et une fonction de réinitialisation $R_i$ (représentant la fonction de réinitialisation des variables continues $x$ dans le nouvel état cible de la transition). Lorsque les transitions sont franchies sur occurrence des événements stochastiques, ces événements sont caractérisés par un taux d'occurrence (taux de défaillance ou de réparation). S'il est possible de quitter un état donné, sur occurrence du même événement, pour atteindre deux états différents (par exemple les états $X^3$ ou $X^4$ atteint depuis l'état $X^2$ sur occurrence de $e_2$), une distribution de probabilités discrètes est associée à ces transitions en conflit. Cette distribution peut, par exemple, modéliser la détection ou non de la défaillance d'un composant : l'événement $e_2$ représente cette défaillance et si cette défaillance est non-détectée (ce qui peut se produire avec une probabilité $p_2^3$) le système arrive dans l'état dangereux $X^3$. Si la défaillance est détectée (ce qui peut se produire avec une probabilité $1 - p_2^3$ ), le système arrive dans l'état $X^4$ où la réparation est effectuée et elle ramène le système dans l'état $X^1$.

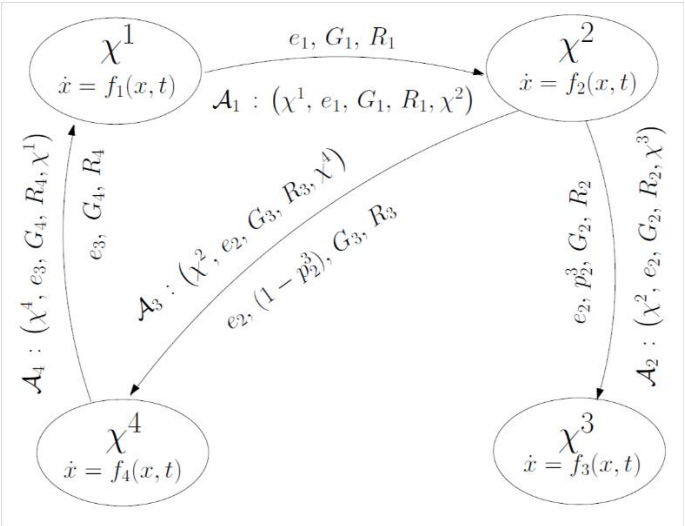



*Fig. 3. 6 : Exemple d'automate stochastique hybride (Babykina, Brînzei, Aubry, & Perez Castaneda, 2011)*

Les ASH sont utilisés pour résoudre les problématiques liées à la fiabilité dynamique que nous avons évoqué dans le chapitre 1. Le principe consiste à construire un ASH élémentaire pour chaque composant du système étudié. La principale difficulté réside dans l'étape de synchronisation des ASH élémentaires pour construire l'ASH global décrivant le système dans son intégralité. La synchronisation résulte d'une composition des ASH.

Néanmoins en phase d'avant-vente, pour proposer une architecture de contrôle-commande, les informations détaillées sur l'évolution des paramètres physiques du système ne seront pas nécessaires et, par conséquent, l'aspect hybride proposé par les ASH n'est pas utile. De plus, comme pour les BDMP, les ASH peuvent souffrir de l'explosion combinatoire du nombre d'états et de transitions lorsque le système modélisé est constitué d'un nombre élevé de composants, ce qui est souvent le cas des architectures de contrôle-commande. Ainsi, ces limites constituent un frein à l'utilisation des ASH dans notre contexte d'étude.

#### *3.2.2.4. Réseaux de Petri et leurs extensions*

La notion de réseaux de Petri (RdP) et de la plupart de ses extensions (temporisé, coloré, stochastique, etc.) a fait l'objet de nombreux travaux dans le cadre d'évaluation de performances, aussi bien dans la SdF que pour d'autres performances (temporelles, etc.) (Signoret, et al., 2013) (Aubry, Brînzei, & Mazouni, 2016) (Ndiaye, 2017) (Barger, 2003) (Huyen, Zamaï, & Ngoc, 2012). Ils peuvent être utilisés à la fois pour des analyses qualitatives (vérification de propriétés) ou pour des analyses quantitatives, comme l'évaluation de performances fonctionnelles ou de sûreté de fonctionnement.

Les RdP sont des graphes orientés composés de places représentant l'état du système modélisé, de transitions et d'arcs qui relient les places aux transitions et les transitions aux places (Fig. 3. 7).

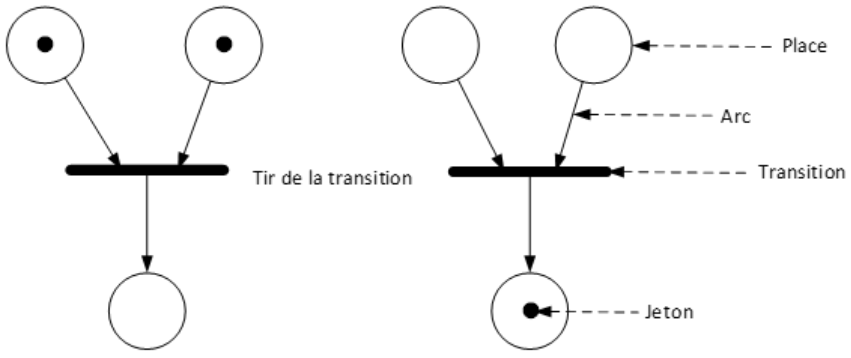



*Fig. 3. 7 : Exemple de RdP et du franchissement d'une transition dans ce réseau (Broy, 2014)*

Un réseau de Petri est un 6-uplet (P, T,F, Pré, Post, $M_0$) où :

- P définit l'ensemble des places.
- T définit l'ensemble des transitions.
- F définit l'ensemble des arcs (représentés par des flèches). Un arc ne peut pas connecter deux places ni deux transitions. Il ne lie que des paires de type place-transition ou transition-place, ce qui se traduit par $F \subseteq (P \times T) \cup (T \times P)$.

- $Pré \subseteq P \times T$ est l'ensemble des fonctions d'incidence avant qui donnent le nombre de jetons à retirer de chaque place en amont d'une transition lorsque celle-ci est franchie
- $Post \subseteq P \times T$ est l'ensemble des fonctions d'incidence arrière qui donnent le nombre de jetons à ajouter dans chaque place en aval d'une transition lorsque celle-ci est franchie.
- $M_0 : P \to \mathbb{N}$ représente le marquage initial du RdP, avec pour chaque place $p \in P$, nous avons à l'instant initial $n \in \mathbb{N}$ jetons.

Les RdP sont utilisés pour une modélisation locale de l'état des composants du système (l'état d'un composant est donné par la présence d'un jeton dans une place) et l'état du système global résulte de la synchronisation des comportements locaux et est donné par le marquage du réseau à un instant donné. Ainsi, l'étape de construction du modèle d'un système est plus facile que dans les cas des chaînes de Markov dans lesquelles on doit envisager directement à la modélisation tous les états du système global. Dans les réseaux de Petri, l'évolution du système est réalisée par le franchissement (ou tir) de transitions. Ces tirs se produisent lorsque toutes les places en amont de cette transition sont marquées. Ils correspondent, dans le RdP, à l'enlèvement des jetons de chacune des places situées en amont de la transition et à l'ajout des jetons dans chacune des places situées en aval de celle-ci. Le nouveau marquage du réseau décrit le nouvel état atteint par le système.

L'analyse d'un réseau de Petri peut s'effectuer de manière analytique à l'aide de sa formalisation dans une algèbre linéaire ou bien à l'aide de son graphe de marquage. Étudier ce graphe, dont les sommets sont les vecteurs de marquage et les arcs sont les transitions, permet d'exhiber des propriétés classiques telles que la bornitude, la vivacité, la réinitialisabilité ou encore le blocage. Si ces propriétés permettent, d'une certaine manière, d'évaluer qualitativement les performances d'un système modélisé sous la forme d'un RdP, elles demeurent insuffisantes pour procéder à une analyse quantitative des performances temporelles.

Les RdP ont ainsi fait l'objet de multiples extensions dont certaines que nous allons décrire ici: RdP temporisés, RdP colorés, RdP hiérarchiques et RdP stochastiques.

Les réseaux de Petri temporisés (*Timed Petri Nets* en anglais), comme dans la Fig. 3. 8, sont des RdP qui introduisent un temps à des places ou à des transitions (Wang, 1998) :

- Une durée de séjour dans une place pour un réseau P-temporisé (le temps est associé aux places). Lorsqu'un jeton est déposé dans une place temporisée, il reste indisponible dans celle-ci pour la durée associée à la place.
- Une durée de franchissement au niveau d'une transition pour un réseau T-temporisé (le temps est associé aux transitions). Le franchissement d'une transition temporisée implique que le nombre de jetons nécessaires dans les places situées en amont sont maintenus pendant la durée associée à la transition et ne peuvent être utilisés pour d'autres tirs. Quand la durée est écoulée, les jetons déposés dans les places en aval sont immédiatement disponibles pour le reste du réseau.

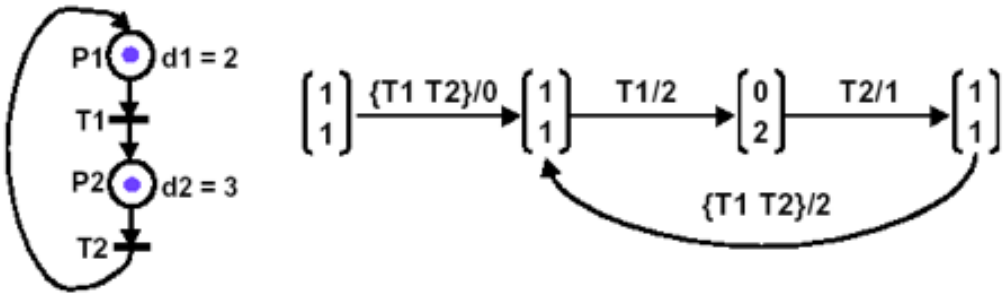



*Fig. 3. 8 : RdP P-temporisé avec son graphe de marquage (David & Alla, 1992)*

Les réseaux de Petri colorés (Peterson, 1980) (Jensen, 1981) présentés dans la Fig. 3. 9, sont des RdP dans laquelle les jetons dans les places du réseau contiennent une information qu'on appelle couleur. Cette couleur peut appartenir à un ensemble parmi des ensembles de type booléen, entier, réel, chaine de caractères. Elle peut être associée à d'autres couleurs pour former un ensemble de couleurs complexe construit comme le produit cartésien de ces couleurs élémentaires. Chaque place $P_i$ est associée à un ensemble de couleurs C et ne peut contenir que des jetons dont les couleurs appartiennent à cet ensemble. Chaque transition $T_j$ est également associée à un ensemble de couleurs C. Des fonctions d'incidence avant f et arrière g sont associés aux arcs respectivement entrant et sortant des transitions $T_j$ et admettent de :

- Définir les conditions de franchissement par rapport au nombre de jetons colorés devant être contenus dans les places d'entrée de la transition $T_j$,
- Définir le nombre et les couleurs des jetons qui vont être déposés dans les places en sortie de la transition $T_j$.

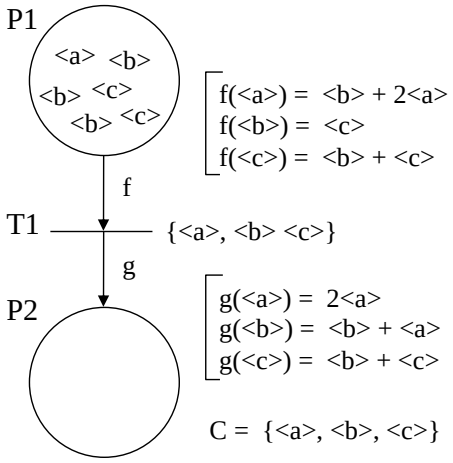



*Fig. 3. 9 : Exemple de RdP coloré (David & Alla, 1992)*

Les réseaux de Petri hiérarchiques (Jensen & Kristensen, 2009), comme dans la Fig. 3. 10, permet de proposer une structuration modulaire des RdP où des éléments du réseau, appelés transition de substitution, sont eux-mêmes composés d'un réseau de Petri. Ceci permet d'avoir une visualisation simplifiée par niveau d'un modèle RdP complexe en le décomposant d'un niveau global, comme un système entier, vers un niveau détaillé, pour la description des constituants du système.

Un RdP hiérarchique est un 11-uplet H = (P, T, F, $M_0$, Pré,Post,C,$T_{sub}$,$P_{port}$,$Type_{Port}$,$PT_{sub}$) où :

- (P, T, F, $M_0$, Pré,Post) est un RdP.
- C définit l'ensemble des couleurs présents dans le RdP.
- $T_{sub}$ définit l'ensemble des transitions de substitutions dans le RdP.
- $P_{port}$ définit l'ensemble des places connectées à une transition de substitution, avec $P_{port} \subseteq P$.
- $Type_{port}$ est un attribut qui définit le type du port de la place $p \in P_{port}$. Le type peut être de type entrée (IN) où p est en amont de la transition de substitution, de type sortie (OUT) où p en aval de la transition, ou de type entrée-sortie (IN/OUT) où p est à la fois en amont et en aval de la transition. Ainsi $Type_{port}: P_{port} \rightarrow \{IN, OUT, IN/OUT\}$.
- $PT_{sub}$ définit la fonction d'assignation des places $P_{port}$ à une transition de substitution $T_{sub}$.

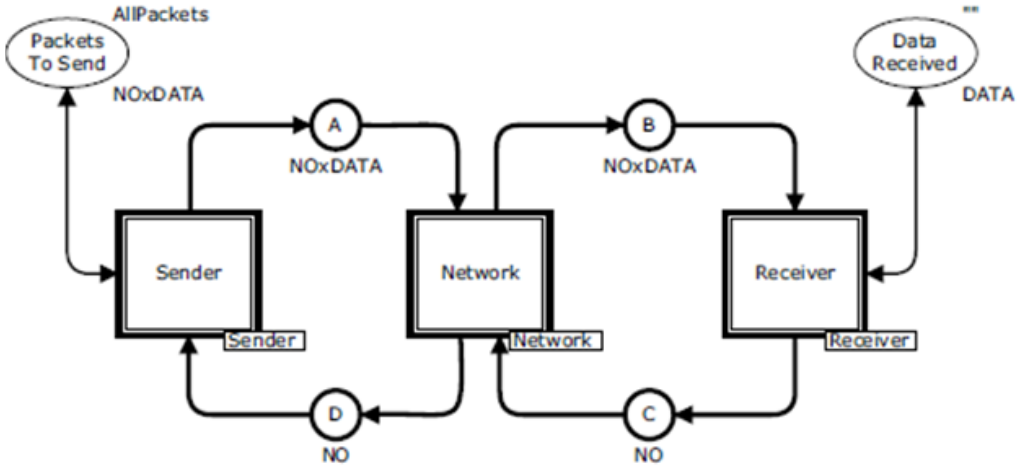



*Fig. 3. 10 : Exemple de RdP hiérarchique avec les transitions de substitution "Sender", "Network" et "Receiver" (Jensen & Kristensen, 2009)*

Les réseaux de Petri stochastiques (Marsan, Gianni Conte, & Balbo, 1984) (Aubry, Brînzei, & Mazouni, 2016), sont une extension des réseaux de Petri temporisés pour laquelle l'ensemble des trajectoires temporisées est muni d'une mesure de probabilité de telle sorte à ce que le couple (marquage, date d'entrée dans le marquage) soit un processus stochastique. D'un point de vue du formalisme, cela se traduit d'une part, par l'association, à toute transition, d'une variable temporelle aléatoire caractérisée par une distribution de probabilité, et d'autre part, par la définition d'une politique d'exécution permettant de définir le processus stochastique dans lesquels les temps de franchissement des transitions ou les durées de séjour dans les places sont générés par des variables aléatoires de distributions quelconques à support dans [0,∞[.

Le marquage d'un RdP stochastique évolue en fonction de ces variables aléatoires. Si pour un marquage donné, plusieurs transitions peuvent être validées, un tirage aléatoire de la durée associée à chaque transition est réalisé. Au bout d'un temps égal à la plus courte de ces durées calculées, la

transition correspondante à celle-ci est franchie, ce qui donne le nouveau marquage du RdP. La même procédure est de nouveau appliquée pour aboutir au marquage suivant du RdP.

La politique d'exécution comprend deux éléments principaux : la politique de sélection qui permet de définir la transition devant être tirée en cas de transitions simultanément sensibilisées depuis un marquage et la politique de mémorisation qui définit la prise en compte du passé, lors de l'entrée dans un marquage (avec réinitialisation, avec mémoire de la dernière période de validation d'une transition (enabling memory) ou avec mémoire de toutes les périodes de validation (age memory) même si les transitions sont désensibilisées.

La nature du processus stochastique dépendra donc de la politique d'exécution choisie. Si l'on considère une politique de réinitialisation, le processus sera markovien ou semi-markovien quel que soit la distribution de probabilités associée aux variables temporelles aléatoires. En revanche, si l'on considère une politique avec mémorisation de la dernière période ou de toutes les périodes (ce qui implique le choix d'un mode de sélection par compétition), cela dépendra des types de distributions. Dans ce contexte, plusieurs types de réseaux de Petri stochastiques devront être considérés :

- les réseaux SPN, pour Stochastic Petri Nets (Natkin, 1980), ne considèrent que des distributions de probabilités exponentielles ; avec une politique de compétition, le graphe de marquage de ce type de réseau est isomorphe avec une chaîne de Markov à temps continu sur laquelle des calculs de performances peuvent être réalisés ;
- les réseaux GSPN, pour Generalized Stochastic Petri Nets (Marsan, Balbo, Conte, Donatelli, & Franceschinis, 1998) introduisent en complément des transitions à durée distribuée exponentiellement, des transitions immédiates à durée nulle ; la technique de compétition est utilisée entre transitions temporisées et entre transitions temporisées et immédiates (avec une priorité pour les transitions immédiates) tandis que la présélection est utilisée entre transitions immédiates ; le processus stochastique associé aux états de durée non nulle est encore une chaîne de Markov à temps continu ;
- les réseaux ESPN, pour Extended Stochastic Petri Nets (Dugan, Trivedi, Geist, & Nicola, 1984) considèrent des distributions quelconques, mis à part pour les transitions concourantes pour lesquelles les distributions sont exponentielles ; le graphe de marquage des ESPN est semi-markovien ;
- les réseaux DSPN, pour Deterministic Stochastic Petri Nets (Marsan & Chiola, 1986), étendent encore le pouvoir de modélisation puisqu'ils considèrent à la fois des transitions immédiates, des transitions à durée exponentielle et des transitions à durée déterministe ; le processus stochastique est semi-markovien régénératif.

### 3.2.3. Formalisme retenu pour la description d'une architecture de contrôle-commande

Nous venons de répertorier un panel de formalismes de modélisation, du diagramme de fiabilité aux réseaux de Petri et leurs extensions. Il faut maintenant choisir le formalisme qui convient à notre étude et aux besoins industriels posés dans les chapitres précédents.

Pour cela, il faut dans un premier temps poser les critères qui vont nous permettre de sélectionner la modélisation adéquate. Une partie a déjà été explicitée dans le chapitre 2 et sera complétée ici par des considérations permettant de simplifier la construction du modèle.

La problématique de notre étude consiste à pouvoir évaluer les performances de sûreté de fonctionnement des architectures de contrôle-commande qui ont comme caractéristiques d'être des systèmes à la fois complexes et ayant un comportement dynamique. Cette évaluation se fait également durant la phase d'avant-vente, un contexte incertain avec une disponibilité des ressources humaines et temporelles limitées, que nous devons considérer dans le choix des formalismes de modélisation.

Les architectures de contrôle-commande ayant un comportement dynamique, il serait difficile d'utiliser une modélisation dysfonctionnelle statique du type diagramme de fiabilité, arbres de défaillances, ou tout modèle découlant d'une fonction de structure. Si des extensions dynamiques existent, pour autant, il n'est pas envisageable de les choisir car les limites en temps pour évaluer le système en avant-vente, ainsi que les connaissances limitées du système, rendent compliqué l'utilisation de ces modèles demandant une compréhension poussée de l'architecture à évaluer. Dans ces conditions, les modélisations dysfonctionnelles statiques ne peuvent être choisies.

Le choix se porte donc sur un formalisme parmi la sélection des modèles dysfonctionnels dynamique que nous avons vu dans la section 3.2.2. Au vu du contexte dans lequel l'évaluation doit être réalisée, il faut que le formalisme respecte les critères suivants :

- Être stochastique. Ce critère permet de considérer les événements survenant aléatoirement sur le système et qui sont liés à la sûreté de fonctionnement, tels que les pannes. Par extension, ceci implique également que le formalisme doit pouvoir prendre en compte le temps et son impact sur le comportement du système.
- Être hiérarchique. Un modèle respectant ce critère donne une vue d'ensemble du système modélisé plus simple et qui se rapproche de la structure des architectures de contrôle-commande de Schneider Electric.
- Être modulaire. Ainsi, il sera envisageable de construire des composants génériques dans le système que l'on pourra instancier rapidement.
- Être paramétrable. Ceci donne la possibilité de s'orienter vers la génération automatique du modèle à instancier et donc de réduire le temps de construction des modèles pour évaluer plusieurs architectures en phase d'avant-vente.

Compte tenu de toutes ces considérations, choisir les réseaux de Petri est un choix judicieux. Plus précisément, le formalisme retenu est celui des réseaux de Petri colorés et temporisés de Jensen (Jensen, 1994) (Jensen & Kristensen, 2009) qui permet notamment de prendre en compte les aspects hiérarchiques et modulaires. Il regroupe des extensions présentées précédemment tels que les RdP P-temporisés, colorés et hiérarchiques. Nous allons employer le terme CPN (pour Coloured Petri Nets) pour évoquer les RdP respectant ce formalisme (Mkhida, Barger, Thiriet, & Aubry, 2005).

Ce formalisme a servi de base au développement de l'outil CPN Tools (Jensen & Kristensen, 2009) en le complétant par :

- un couplage des réseaux CPN à un langage de modélisation fonctionnelle (ML) (Milner, Tofte, Harper, & MacQueen, 1997)
- l'introduction de variables aléatoires définies par une distribution de probabilité pour caractériser les durées de séjour dans les places,
- le recours à la simulation de Monte-Carlo dont nous allons voir la définition dans la section 3.4 notamment pour l'évaluation de performances, temporelles ou fiabilistes.

Néanmoins, il faut préciser que le formalisme CPN original défini par (Jensen & Kristensen, 2009) et implanté dans l'outil CPN Tools n'intègre pas les réseaux de Petri stochastiques, car le comportement formel des réseaux de Petri stochastiques n'est pas prévu dans la définition des réseaux de Petri colorés. Or disposer du comportement formel des réseaux de Petri stochastiques est indispensable dans les études de sûreté de fonctionnement qui doivent prendre en compte l'occurrence aléatoire des défaillances. Des travaux précédents effectués au laboratoire ont permis de proposer une approche de modélisation en réseaux de Petri colorés pour obtenir le comportement formel des réseaux de Petri stochastiques (Pinna, Babykina, Brînzei, & Pétin, 2013). Dans le domaine de la sûreté de fonctionnement, cette approche a permis la modélisation et l'évaluation probabiliste des systèmes de contrôle-commande numérique utilisés comme systèmes de protection pour les réacteurs nucléaires (Deleuze & Brînzei, 2015) ou pour l'étude et l'évaluation de l'impact des défaillances de cause commune (Deleuze, Brînzei, & Gérard, 2015).

Les places dans les RdP vont permettre de décrire principalement les différents états dans lequel les composants du système sont à un instant de la simulation, en plus de fournir des informations annexes pris en considération pour faire évoluer le comportement de l'architecture. Le franchissement des transitions qui relient ces places va être déterminé par les conditions pour passer d'un état à un autre, comme les interactions entre les composants ou l'occurrence d'un événement aléatoire. Ces conditions et ces états sont donnés, dans un cadre général, par les diagrammes UML que nous avons vu dans les chapitres précédents et que nous allons transcrire sous la forme d'un RdP (David, 2009).

Les différentes extensions donnent la possibilité d'étendre le scope et de modéliser de façon réaliste les architectures de contrôle-commande. La temporisation va permettre de considérer le temps dans l'évolution du système, avec les phénomènes de dégradation sur les composants ou encore l'arrivée d'événements aléatoires sur ceux-ci tels que les pannes, les mises en hors tension ou les réparations. La hiérarchisation du modèle, comme dit plus haut, donne une vue globale proche de la description des systèmes que nous allons évaluer, en construisant des modules sous la forme de transition de substitution pour chaque composant du système. La coloration permet le paramétrage du système au moment de l'instanciation mais également de véhiculer les informations inhérentes au système, permettant de faire évoluer ce dernier et de donner les conditions nécessaires pour franchir les transitions du RdP.

Un autre argument important concerne la génération la plus assistée possible des modèles dans le contexte de l'avant-vente. Les caractéristiques hiérarchiques et modulaires associées aux

possibilités de paramétrage et d'instanciation des modèles ouvrent la voie vers la construction d'une bibliothèque de modèles d'architectures que les intégrateurs pourront de manière transparente instancier aux architectures analysées. Plusieurs travaux attestent de la faisabilité de cette démarche vers la génération automatique des modèles d'évaluation SdF : depuis des diagrammes UML vers les RdP stochastiques (David, 2009), depuis une description informelle vers CPN (Ndiaye, 2017) ou depuis un schéma « bloc diagramme » vers les réseaux de Petri stochastiques (Chevalier, Vinuesa, Buchsbaum, Folleau, & Thomas, 2020).

## 3.3. Indicateurs pour l'évaluation des performances de SdF des architectures de contrôle-commande

L'évaluation des performances de SdF d'une architecture de contrôle-commande peut être réalisée à partir de deux types d'indicateurs de SdF possibles. Nous avons les indicateurs classiques de SdF et leurs métriques qui concentrent les quatre composantes de la SdF : Disponibilité, Fiabilité, Maintenabilité et Sécurité. Le deuxième type sont les facteurs d'importance permettant de quantifier la contribution des composants dans les performances du système à être fiable ou non face à un risque local ou global.

Dans les prochaines sous-sections, nous allons voir la description de ces différents indicateurs.

### 3.3.1. Indicateurs de SdF et leurs métriques

La sûreté de fonctionnement englobe quatre composantes : la fiabilité, la disponibilité, la maintenabilité et la sécurité que nous pouvons regrouper dans le sigle FMDS ou RAMS en anglais (pour *Reliability, Availability, Maintainability* et *Safety*) (Villemeur, 1988) (CEI, 2010) (Laprie J.-C. , 1995) (Laprie J.-C. , 2004) (Mkhida, Thiriet, & Aubry, 2008). Ces indicateurs sont complétés par des métriques qui permettent de quantifier ces indicateurs et de les corréler avec un temps de fonctionnement ou de dysfonctionnement du système.

Les prochaines sous-sections vont se concentrer sur la définition de ces indicateurs, de leur expression mathématique et des métriques associés à ceux-ci.

#### *3.3.1.1. Fiabilité*

Le premier indicateur SdF est la fiabilité. C'est l'aptitude d'une entité à réaliser une fonction requise dans des conditions données, pendant une période donnée, sans interruption. Cela se traduit au niveau de la fiabilité R à un instant t par la probabilité suivante :

$$R(t) = P\ [\text{entité non défaillante sur } [0, t]]$$

Ou pour une période donnée $[t_1, t_2]$ :

$$R(t) = P\ [\text{entité non défaillante sur } [t_1, t_2]]$$

Son complémentaire noté $\bar{R} = 1 - R(t)$, est appelé défiabilité et représente la probabilité de défaillance.

La loi de la fiabilité d'une entité va permettre de prédire la probabilité d'occurrence d'une panne sur ce dernier. Elle est basée sur un taux de défaillance λ suivant une distribution aléatoire. Le taux de défaillance peut s'apparenter à une probabilité conditionnelle de défaillance sur ]t,t+dt], en sachant que l'entité n'a pas eu de défaillance sur [0,t].

La relation entre la fiabilité et le taux de défaillance est définie par la formule suivante :

$$R(t) = exp\left[-\int_0^t \lambda(t)dt\right] \tag{3.4}$$

Selon la loi de probabilité utilisée, l'expression de la fiabilité peut être déduite rapidement du taux de défaillance. Dans le cadre de la loi exponentielle par exemple, le taux de défaillance est une valeur constante. La loi exponentielle est utilisée lorsque les défaillances sont indépendantes et imprévisibles, dont la génération suit un processus de Poisson. Pour rappel, les événements passés sur des entités suivant cette loi n'affectent pas l'évolution future de ces entités. La loi exponentielle est appliquée pour des composants électroniques en période utile de fonctionnement, la période qui correspond à celle où le rodage du composant est terminé et celle de vieillissement accéléré pas encore entamée (cf. Fig. 3. 11).

Dans ce cas, la fiabilité du composant se résume à :

$$R(t) = e^{-\lambda t}$$

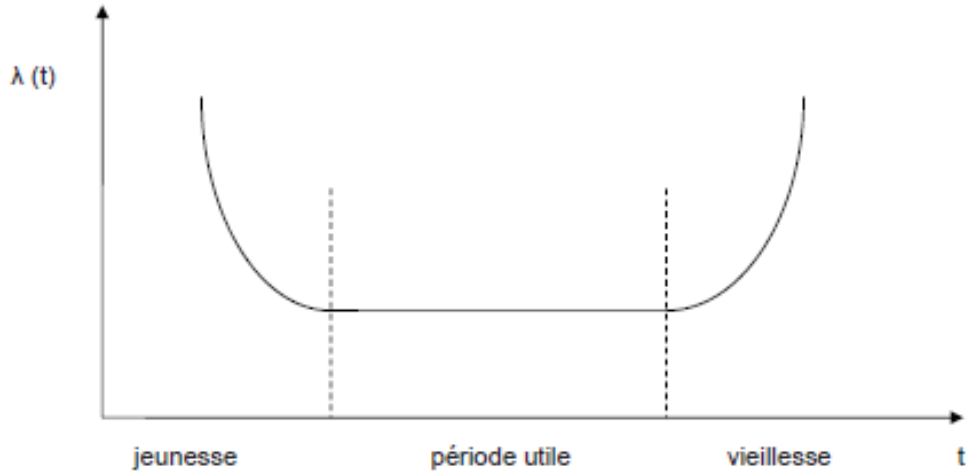



*Fig. 3. 11 : Courbe en baignoire du taux de défaillance*

En complément de la fiabilité, nous allons estimer le temps moyen de fonctionnement avant la panne de l'entité, Mean Time To Failure (MTTF). Cette métrique permet d'estimer le temps opérationnel du système avant que la première défaillance arrive, c'est-à-dire avant que le plan de maintenance ne commence à prendre de l'importance.

En équation mathématique, la MTTF se caractérise par la formule suivante :

$$MTTF = \int_0^\infty R(t)dt \tag{3.5}$$

Si le composant suit une loi exponentielle, alors la MTTF se résume à l'expression suivante :

$$MTTF = \int_0^\infty R(t)dt = \int_0^\infty e^{-\lambda t}dt = \frac{1}{\lambda}$$

#### *3.3.1.2. Disponibilité*

Le deuxième indicateur SdF est la disponibilité. C'est l'aptitude d'une entité à être en état de réaliser une fonction requise dans des conditions données, liées à l'environnement par exemple, et à un instant donné. Cela se traduit au niveau de la disponibilité instantanée A par la probabilité suivante :

$$A(t) = P \text{ [entité non défaillante à l'instant t]}$$

C'est une probabilité non conditionnée par l'histoire de l'entité, c'est-à-dire qui ne dépend pas des événements impactant l'entité avant l'instant t (réparations, arrêts, etc.). Son complémentaire s'appelle indisponibilité et est notée $\bar{A}$ :

$$\bar{A}(t) = 1\text{-}A(t)$$

Il est possible de calculer la disponibilité asymptotique à partir de la MTTF et du temps moyen de réparation, Mean Time To Repair (MTTR), que nous allons aborder dans la sous-section suivante. Elle est équivalente à la proportion du temps durant laquelle l'entité est en état de fonctionnement durant toute la période d'utilisation. Elle est définie par :

$$A(\infty) = \frac{MTTF}{MTTF + MTTR} \quad (3.6)$$

Une métrique que nous pouvons associer avec la disponibilité est le temps moyen de disponibilité, Mean Up Time (MUT) qui représente l'espérance mathématique de toutes les durées de fonctionnement. L'indisponibilité est caractérisée par le Mean Down Time (MDT) qui comprend la détection de la panne, la durée d'intervention, la durée de la réparation et la durée de la remise en service. Ici, nous faisons l'hypothèse que la MDT est égale à la durée de réparation MTTR, la logistique n'étant pas pris en compte dans l'évaluation en phase d'avant-vente, et les pièces de rechange étant disponibles à tout moment.

#### *3.3.1.3. Maintenabilité*

Le troisième indicateur SdF est la maintenabilité. C'est l'aptitude d'une entité à être maintenue ou rétablie dans un état où elle peut accomplir ses fonctions requises lorsque la maintenance est réalisée dans des conditions données avec des procédures et des moyens prescrits. Cela se traduit au niveau de la maintenabilité instantanée M par la probabilité suivante :

$$M(t) = P \text{ [entité réparée à t sachant qu'elle était défaillante à 0]}$$

ou

$$M([t_1, t_2]) = P[\text{entité défaillante à } t_1 \text{ et réparée à } t_2]$$

Comme dans le cas de la fiabilité, il est possible d'associer cet indicateur à une métrique comme le temps moyen de réparation, la MTTR. En équation mathématique, la MTTR se caractérise par la formule suivante :

$$MTTR = \int_0^\infty M(t)dt \tag{3.7}$$

Si le composant suit une loi exponentielle pour la réparation également, alors la MTTR se résume à l'expression suivante :

$$MTTR = \int_0^\infty M(t)dt = \int_0^\infty e^{-\mu t}dt = \frac{1}{\mu}$$

Avec μ le taux de réparation du composant.

#### *3.3.1.4. Sécurité*

Le dernier indicateur SdF est la sécurité. C'est l'aptitude d'une entité à éviter de faire apparaître des événements critiques ou catastrophiques pouvant affecter autant les personnes que les équipements ou encore l'environnement.

Les questions de sécurité sont primordiales dans des domaines industriels où les défaillances mettant en péril l'intégrité de l'installation ne sont pas permises, comme le monde de l'énergie nucléaire, des transports ou dans les domaines aéronautique et spatial. Dans les installations de production manufacturière, comme nous pouvons le rencontrer dans nos architectures de contrôle-commande, les questions sont plus relatives à la disponibilité.

#### *3.3.1.5. Indicateurs retenus pour l'évaluation des performances de SdF*

Les sous-sections précédentes ont permis de définir une liste d'indicateurs de sûreté de fonctionnement, ainsi que les métriques qui leur sont associés. Nous allons répertorier également un nombre de facteurs d'importance pour développer une analyse plus approfondie de l'impact des composants sur le comportement du système global, pour une architecture de contrôle-commande dans notre étude. Bien que tous ces indicateurs et ces facteurs d'importance sont valables à évaluer dans nos travaux de recherche, nous ne pourrons pas tous les considérer, et ceci est dû aux exigences et aux besoins industriels posés dans les chapitres précédents.

Dans le cadre des indicateurs et des métriques, les suivants sont retenus dans l'évaluation que nous allons mettre en place pour une architecture de contrôle-commande :

- La disponibilité et la MDT (ou temps d'indisponibilité)
- La MTTF (ou temps moyen de fonctionnement avant défaillance)
- La MTTR (ou temps moyen de réparation)

Bien qu'il soit potentiellement possible de les calculer, les métriques MTTF et MTTR ont été pris sans les indicateurs associés, car ils ne font pas partie des exigences établies à partir des besoins industriels. Dans les spécifications du projet en phase d'avant-vente, il n'est pas demandé d'estimer explicitement la fiabilité et la maintenabilité. En revanche, leurs métriques représentent une

demande récurrente, qui au final, permet de traduire pour un non-expert en sûreté de fonctionnement, ce que représenterait l'indicateur s'il était calculé, en donnant un ordre de grandeur temporel. La MTTF et la MTTR donnent une vision plus pratique des performances fiabilistes d'une architecture de contrôle-commande, combien de temps il est indisponible à cause d'une ou plusieurs défaillances, et combien de temps il faut pour le réparer.

Une exception est faite sur la MTTR qui ne sera pas un indicateur calculé mais un indicateur donné en entrée de l'évaluation. En effet, dans un contexte aussi incertain que l'avant-vente, le plan de maintenance n'est pas totalement défini et dépend fortement des résultats affichés pour le reste des indicateurs, notamment la disponibilité et la MDT. De ce fait, il est plus judicieux d'évaluer les performances fiabilistes d'une architecture, en jouant plusieurs scénarios, par exemple en donnant le nombre de pièces de rechange à disposition, et le temps moyen alloué pour réparer un composant.

Ainsi, il sera possible par la suite de savoir comment mettre en place la maintenance pour atteindre les résultats escomptés et respecter les exigences du projet industriel dont fait partie le système. En sachant la MTTR fixée au début et le nombre moyen de défaillances dans le système après évaluation, le système intégrateur, ou le client du projet, pourra dimensionner le stock de pièces de rechange et mettre en place la logistique adaptée (dont les ressources humaines, les opérateurs), afin d'atteindre la MTTR donnée lors de l'évaluation.

L'indicateur sur la sécurité du système et par extension son niveau SIL n'est pas estimé dans le cadre de nos travaux de recherche. En effet, le scope de notre étude se limite à des architectures de contrôle-commande, dont l'application n'est pas impliquée dans des domaines complexes tel que le nucléaire et dont la défaillance du système n'implique pas un danger imminent ou une catastrophe. Nous nous limitons à des applications où la défaillance du système entraîne une dégradation du procédé industriel (production manufacturière, traitement chimiques sans une gravité extrême).

Il sera néanmoins envisageable en perspectives de prendre en considération les systèmes de sécurité dont les architectures de contrôle-commande sont constitués de composants appliqués à la mise en place de fonctions de sécurité du procédé industriel, dans le but de calculer le niveau SIL du système. Dans notre étude, les composants de sécurité sont pris en compte dans un cadre purement applicatif. S'ils ont un comportement différent des autres composants, ils possèdent malgré tout une partie applicative dédiée au fonctionnement du procédé et une partie sécurité dédié à la protection du système et des personnes. Ici dans notre étude, nous considérons seulement la partie applicative pour les composants de sécurité.

### 3.3.2. Facteurs d'importance

Dans une étude liée à la sûreté de fonctionnement, il est important de pouvoir cibler les composants critiques du système, et qui mérite une grande attention dans la mise en place du plan de maintenance. Il faut pouvoir savoir quel composant réparer en priorité ou lequel renforcer afin d'améliorer les performances fiabilistes du système. C'est dans ce cadre que les facteurs d'importance ont été introduits.

Dans les prochaines sous-sections, nous allons définir les facteurs d'importance les plus classiques, leur expression et dans quel cas ils s'appliquent (Birnbaum, 1968) (Villemeur, 1988) (Duflot, 2007) (Rausand & Hoyland, 2003) (Kuo & Zhu, 2012) (Borgonovo, 2007). Un facteur d'importance définit une valeur pour chaque composant dans le système.

#### *3.3.2.1. Facteur de Birnbaum*

Le facteur de Birnbaum représente la variation de la fiabilité du système en fonction de la fiabilité du composant *i*. Il est exprimé comme la différence entre le risque du système d'être défaillant quand le composant *i* fonctionne et le risque du système à tomber en panne quand le composant *i* ne fonctionne pas. Sa définition mathématique est :

$$I^B(i|système\ défaillant\ \text{à}\ t) = R_{systeme}(1_i, R(t)) - R_{systeme}(0_i, R(t)) \qquad (3.8)$$

avec :

- $R_{systeme}(1_i, R(t))$ la fiabilité du système à l'instant t en supposant que le composant *i* fonctionne
- $R_{systeme}(0_i, R(t))$ la fiabilité du système à l'instant t en supposant que le composant *i* ne fonctionne pas

Le facteur de Birnbaum varie dans le temps, selon les événements qui sont arrivés sur le système ainsi que sur le composant *i* en question. Plus la valeur est élevée, plus la fiabilité du composant *i* a un impact sur le système, c'est-à-dire que sa défaillance entraîne rapidement une dégradation du système (Kvassay, Zaitseva, & Levashenko, 2017).

#### *3.3.2.2. Facteur d'importance critique*

Le facteur d'importance critique (*Criticality Importance* en anglais) est en relation avec le facteur de Birnbaum. Il représente la probabilité qu'un composant soit défaillant et critique pour le système sachant que le système global est défaillant. Un équivalent serait de dire qu'il s'agit de la probabilité que le composant *i* soit la source de la défaillance du système sachant que le système est défaillant. Sa définition mathématique est :

$$I^{CR}(i|système\ défaillant\ \text{à}\ t) = \frac{I^B(i|système\ défaillant\ \text{à}\ t)[1 - R_i(t)]}{1 - R_{systeme}(t)} \qquad (3.9)$$

avec :

- $R_{systeme}(t)$ la fiabilité du système à l'instant t
- $R_i(t)$ la fiabilité du composant *i* à l'instant t
- $I^B(i|système\ défaillant\ \text{à}\ t)$ le facteur Birnbaum du composant *i* sachant que le système est défaillant à l'instant

Le composant *i* est défini comme critique pour le système, si le système fonctionne si et seulement si le composant *i* fonctionne, en d'autres termes, si l'état du système dépend du composant *i*.

#### *3.3.2.3. Facteur d'amélioration potentielle*

Le facteur d'amélioration potentielle (*Improvement potential measure* en anglais) représente la capacité du composant dans un système à pouvoir améliorer la fiabilité de l'ensemble si les ressources de maintenance sont concentrées sur le composant pour réduire au maximum l'occurrence d'une défaillance sur celui-ci.

Il est calculé comme la différence entre la fiabilité du système sachant que le composant est opérationnel et la fiabilité du système sachant que le composant est parfait. Nous faisons donc une comparaison entre deux situations. Sa définition mathématique est :

$$I^{IP}(composant_i|\ t) = \ R_{systeme}\big(1_i, R(t)\big) - R(t) \tag{3.10}$$

avec :

- $R_{systeme}\big(1_i, R(t)\big)$ la fiabilité du système à l'instant t en supposant que le composant *i* est parfait
- R(t) la fiabilité du système à l'instant t
- $I^{IP}(composant_i|t)$ le facteur d'amélioration potentielle due au composant *i* à l'instant t

Une autre variante est le facteur d'amélioration potentielle crédible (*Credible improvement potential measure* en anglais) où la différence réside dans le fait de considérer le composant non pas dans un état parfait, état impossible à atteindre en réalité, mais dans un état atteignant un certain seuil de fiabilité du composant acceptable. De ce fait, nous observons son degré d'impact sur l'amélioration de la fiabilité du système si le composant atteint une fiabilité envisagée.

#### *3.3.2.4. Facteur d'augmentation du risque*

Le facteur d'augmentation du risque (*Risk Achievement Worth RAW* en anglais) mesure l'augmentation du risque sur le système à devenir défaillant quand le composant *i* est défaillant. C'est le ratio entre la défiabilité du système si le composant *i* est défaillant (ne fonctionne pas) et la défiabilité nominale du système. Sa définition mathématique est :

$$I^{RAW}(composant_i|t) = \frac{1 - R_{systeme}\big(0_i, R(t)\big)}{1 - R_{systeme}(t)} \tag{3.11}$$

avec :

- $R_{systeme}\big(0_i, R(t)\big)$ la fiabilité du système à l'instant t en supposant que le composant *i* est défaillant
- $R_{systeme}(t)$ la fiabilité du système à l'instant t
- $I^{RAW}(composant_i|t)$ le facteur d'augmentation du risque dû au composant *i* à l'instant t

Si le facteur d'augmentation du risque du composant se rapproche de la valeur égale à 1, alors cela signifie que la défaillance du composant a peu ou pas d'effet sur la défaillance du système.

#### *3.3.2.5. Facteur de diminution du risque*

Le facteur de diminution du risque (*Risk Reduction Worth RRW* en anglais) peut être vu comme le complémentaire du facteur d'augmentation du risque. Il mesure la diminution du risque sur le système à devenir défaillant quand le composant *i* fonctionne. Pareillement, il s'agit d'un ratio entre la défiabilité nominale du système et la défiabilité du système si le composant *i* est en marche (fonctionne). Sa définition mathématique est :

$$I^{RRW}(composant_i|t) = \frac{1 - R_{systeme}(t)}{1 - R_{systeme}(1_i, R(t))} \tag{3.12}$$

avec :

- $R_{systeme}(1_i, R(t))$ la fiabilité du système à l'instant t en supposant que le composant *i* fonctionne
- $R_{systeme}(t)$ la fiabilité du système à l'instant t
- $I^{RRW}(composant_i|t)$ le facteur de diminution du risque dû au composant *i* à l'instant t

Si la valeur du facteur d'augmentation du risque du composant se rapproche de 1, alors cela signifie qu'améliorer le fonctionnement du composant a peu ou pas d'effet sur la fiabilité du système.

#### *3.3.2.6. Facteur de Fussell-Vesely*

Le facteur de Fussell-Vesely représente la part du composant dans les coupes minimales du système pour être défaillant. Ici, une coupe est une combinaison de composants sans lien de causalité et avec des conditions suffisantes, telle que cet ensemble, si tous les composants sont défaillants, alors provoque la défaillance du système (Duflot, Bérenguer, Dieulle, & Vasseur, 2009) (Vatn, 1992). Une coupe est dite minimale si elle ne contient aucune autre coupe du système, c'est-à-dire que nous avons la plus petite séquence de composants provoquant la défaillance immédiate du système. Les coupes minimales permettent donc de décrire les conditions pour que le système soit défaillant de la façon la plus concise possible.

Ce facteur mesure l'importance d'un composant *i* en calculant la probabilité que le système soit défaillant à cause d'une coupe minimale qui contient le composant *i*. Sa définition mathématique est :

$$I^{FV}(composant_i|t) = \frac{\sum_{j=1}^{m_j} P(i\ dans\ coupe\ minimale\ j|t)}{1 - R_{systeme}(t)} \tag{3.13}$$

avec :

- $R_{systeme}(t)$ la fiabilité du système à l'instant t
- $m_j$ le nombre de coupes minimales dans le système
- $P(i \text{ dans coupe minimale } j|t)$ la probabilité que le composant *i* soit dans la liste des composants constituant la coupe minimale j
- $I^{FV}(composant_i|t)$ le facteur Fussell-Vesely du composant *i* à l'instant t

#### *3.3.2.7. Facteurs d'importance retenus pour l'évaluation des performances de SdF*

Concernant le choix au niveau des facteurs d'importance, nous avons sélectionné les suivants pour mesurer l'impact des composants sur l'architecture :

- Le facteur de Birnbaum
- Le facteur d'importance critique
- Le facteur d'amélioration potentielle

Le facteur de Birnbaum est important et il sera retenu pour son évaluation dans ce travail car il permet d'identifier les composants dont la variation de leur fiabilité provoque une variation importante sur la fiabilité du système. Ainsi, on obtient une indication sur les composants qu'il faut choisir parmi ses variantes les plus fiables ou si cela n'est pas possible, alors il faudra les redonder, afin d'empêcher qu'il y ait une chute importante de la fiabilité de l'architecture de contrôle-commande si la fiabilité des composants en question diminue.

Le facteur d'importance critique découle du facteur de Birnbaum. L'évaluation et l'analyse du facteur d'importance critique est intéressante pour la maintenance du système car il permet d'identifier les composants dont les réparations entraîneront la réparation du système avec la plus grande probabilité ; plus sa valeur est grande, plus la probabilité que le système soit réparé, en réparant ce composant, est grande. Nous aurons donc une indication sur les composants qu'il faudra réparer en priorité, une fois qu'ils sont défaillants, et ainsi prévoir une meilleure politique de maintenance corrective.

Le facteur d'amélioration potentielle sera retenu également car, dans le cas d'une stratégie de maintenance préventive, il permet d'identifier le composant sur lesquels il faudra concentrer les ressources de maintenance pour améliorer la fiabilité du système en réduisant au maximum la probabilité d'occurrence de sa défaillance.

Nous avons écarté de notre étude le facteur d'augmentation du risque, et son complémentaire le facteur de diminution du risques, à cause de leur approche. En effet, celui-ci calcule le risque que le système soit indisponible en partant de l'hypothèse que le composant est défaillant. Pour des composants avec un taux de défaillance tel que l'occurrence d'une panne est un événement rare durant la période d'exploitation de l'architecture de contrôle-commande, ce facteur risque de ne pas montrer l'importance du composant qui se verra le plus souvent en état de marche.

Le facteur de diminution du risque pourrait être calculé, mais comme le facteur d'importance critique étant déjà donné, cela donnerait une information redondante et des calculs supplémentaires dans l'évaluation. Dans un contexte d'avant-vente où l'évaluation des performances fiabilistes d'une architecture doit se faire le plus rapidement possible, par manque de ressources en temps et en réponse, il nous a paru judicieux de ne pas considérer les facteurs d'augmentation ou de diminution du risque.

Le facteur de Fussell-Vesely n'est pas calculé pour les mêmes raisons qui nous ont poussé à écarter les formalismes de modélisation de type Arbres de défaillance ou encore tout autre modèle basé sur la fonction de structure permettant de déterminer les coupes d'un système. En effet, le facteur de Fussell-Vesely se base sur les coupes minimales de l'architecture. Ceci implique d'avoir

une connaissance approfondie du système ainsi que des ressources en temps suffisantes pour entamer l'analyse du système et déterminer les séquences de composants défaillants amenant à l'indisponibilité du système.

Il ne sera pas envisageable de calculer ce facteur de Fussell-Vesely dans le contexte aussi incertain que l'avant-vente. Le système n'est pas fixe, et plusieurs architectures peuvent être candidats pour un même procédé industriel à faire tourner, ce qui impliquerait de calculer à chaque fois ce facteur en faisant une analyse approfondie sur les coupes minimales du système pour chaque composant. Le manque de ressources en temps, mais également en expertise, pour pouvoir répondre rapidement au client nous impose de ne pas estimer ce facteur.

C'est donc avec les indicateurs et les facteurs d'importance listés dans cette section 3.3.2.7 que nous allons pouvoir introduire les différentes méthodes d'évaluation dans la prochaine section, afin de voir quelle approche est la plus optimale pour obtenir ces indicateurs et ces facteurs.

## 3.4. Estimation des indicateurs de SdF et des facteurs d'importance

Dans les sections précédentes, nous avons sélectionné le formalisme de modélisation pour décrire une architecture de contrôle-commande et nous avons défini une liste d'indicateurs de SdF et de facteurs d'importance pour interpréter les performances fiabilistes et l'impact des équipements sur le système global.

Le dernier socle manquant est l'approche choisie pour calculer les performances de SdF de l'architecture à partir de l'évolution du comportement du système modélisé dans les CPN dont nous allons voir le développement dans le chapitre 4. Ici, il s'agit de déterminer comment nous allons pouvoir récupérer les indicateurs de SdF et les facteurs d'importance à partir du modèle CPN de l'architecture.

Deux grandes familles se dessinent pour la méthode d'évaluation : les approches basées sur des calculs analytiques et les approches par simulation. Les prochaines sections présentent ces deux types d'approches en soulignant les arguments qui conduiront au choix de l'une d'entre elle dans le cadre de notre étude.

### 3.4.1. Approches analytiques

La première approche possible pour calculer les performances de SdF des architectures de contrôle-commande est basée sur les calculs analytiques permettant l'obtention des valeurs numériques exactes pour les indicateurs et les facteurs d'importance qui seront calculés.

Certaines méthodes sont destinées à des formalismes de modèles statiques, d'autres pour des modèles dynamiques. Dans notre cas, comme nous l'avons vu précédemment, les méthodes qui sont dédiés aux modèles statiques ne sont pas envisageables dans nos travaux de recherche, étant donné que nous travaillons sur des systèmes avec un comportement dynamique et que nous ne pouvons pas faire abstraction de ce dernier.

Brièvement, pour les méthodes de calculs analytiques pour les systèmes modélisés de façon statique, nous pouvons citer le principe d'inclusion-exclusion (Narushima H. , 1982) (Narushima

H. , 1974). Ce principe permet de déterminer la probabilité d'occurrence d'un événement indésirable, comme la défaillance, à partir de ses coupes. La probabilité d'occurrence de l'événement est équivalente à la probabilité qu'au moins une coupe soit vraie, sachant que la probabilité pour une coupe correspond au produit des probabilités des événements élémentaires indépendants (défaillances) des composants dans la coupe.

Une autre méthode pour les modèles statiques se base sur le calcul différentiel booléen (Zaitseva, Levashenko, & Kostolny, 2015). Ces travaux ont permis d'élargir ce problème à des systèmes complexes et de grande taille, mais également à des système non cohérents (un système est cohérent s'il est défaillant quand ses composants sont défaillants et fonctionnent quand ses composants fonctionnent : une défaillance sur un composant ne peut pas entrainer le système à se remettre en marche, et la réparation d'un composant ne peut pas entrainer la défaillance du système).

Une dernière approche pour les méthodes statiques sont les diagrammes de décision binaire (Dutuit & Rauzy, 2005) qui sont une représentation graphiques de fonctions booléennes. La fonction de structure est décrite par ce diagramme au lieu d'une expression booléenne et permet également d'identifier les combinaisons possibles qui causent un événement indésirable, comme la défaillance du système.

Concernant les modèles dynamiques, il est possible d'avoir une approche analytique en se basant sur le principe des processus stochastiques, comme les CdM (Ionescu, 2016). Il s'agit d'identifier et d'évaluer la probabilité d'occurrence de séquences d'événements en régime stationnaire (ou asymptotique lorsque la distribution des probabilités d'états devient stationnaire lors de l'évolution du système) ou en régime transitoire, si le modèle dépend du temps.

En fonction des états à prendre en compte dans le système, il sera possible de calculer les indicateurs de SdF que nous souhaitons estimer comme la disponibilité et la fiabilité et leurs métriques ou les facteurs d'importance. Nous pouvons calculer l'espérance mathématique de ces indicateurs sur les états où le système est en fonctionnement, ou l'inverse pour les indicateurs afférant à la maintenabilité. L'estimation des probabilités d'occurrence des séquences d'événements est basée, sur la détermination d'un automate probabiliste. En régime asymptotique, ce dernier est déterminé grâce à une chaîne de Markov à temps discrets immergée dans un processus stochastique continu. En régime transitoire, c'est à partir d'une transformée de Laplace (en faisant la convolution des tous les temps de passage pour tous les chemins amenant à un état souhaité).

### 3.4.2. Approches par simulation

Une deuxième approche pour évaluer les indicateurs de SdF d'une architecture de contrôle-commande est celle se basant sur les méthodes de simulation, et plus exactement la simulation de Monte Carlo (Zio, 2013) (Chabot, Dutuit, & Rauzy, 2001) (Kim & Lee, 1992). Le principe de la simulation, au sens général, est d'utiliser une représentation abstraite d'un système (ou d'un problème) sous la forme d'un modèle, et d'observer son évolution sans à avoir faire fonctionner le système réel. Nous observons le comportement du système modélisé pour pouvoir observer

l'évolution d'une ou plusieurs valeurs inhérentes au système, dont nous avons besoin pour notre étude.

La simulation de Monte Carlo est une approche orientée vers la résolution statistique, fréquemment utilisée dans de nombreux domaines industriels, notamment lors de la conception et/ou la configuration d'un nouveau système, ou encore l'amélioration d'un système déjà existant. Elle permet d'obtenir des estimations statistiques sur des variables aléatoires de type indicateur ou paramètre du système modélisé. Elle est utilisée particulièrement pour des systèmes complexes ou à grande échelle dont l'analyse par approche analytique serait fastidieuse à mettre en œuvre, la taille du système étant trop grande pour déterminer tous les paramètres ou les évolutions possibles de ce dernier.

La simulation de Monte Carlo est une méthode numérique basée sur le tirage au sort de variables aléatoires. Les valeurs des variables que nous désirons estimer vont être obtenues grâce à leur espérance mathématique, évoluant selon un processus stochastique. La quantification de ces grandeurs recherchées, comme la disponibilité ou la probabilité d'apparition d'une défaillance dans le système, est basée sur l'étude d'un certain nombre de scénarios différents sur le système, permettant d'extraire des résultats statistiques.

La répétition de plusieurs scénarios appelés également expérience ou histoire et dont les paramètres diffèrent va permettre d'observer l'évolution comportementale du système simulé. La répétition des histoires permet entre autres de multiplier les cas d'évolution possibles pour le système, de ne pas manquer des cas particuliers pouvant arriver mais aussi d'intégrer les aspects stochastiques pouvant être présents dans le modèle et de considérer les distributions aléatoires de probabilité.

Nous simulons donc le comportement à la fois fonctionnel et dysfonctionnel du système en suivant son évolution sur une durée fixée, la durée d'exploitation du système par exemple. On réitère un grand nombre de fois l'expérience en partant à chaque fois dans les mêmes conditions initiales. Les données de l'histoire en cours sont conservées en mémoire du processus. Lorsque la série de simulations est terminée, nous exploitons l'ensemble des données accumulées par un traitement statistique post-simulation permettant de déterminer les estimations désirées sur les espérances mathématiques des variables aléatoires qui nous intéressent.

La simulation de Monte Carlo convient particulièrement bien aux études de sûreté de fonctionnement dans un cadre dynamique étant donné le caractère stochastique naturel du problème étudié. Cette méthode de résolution s'avère être peu sensible au problème de dimension des systèmes ou de la complexité de ceux-ci. Il est possible d'effectuer une simulation de Monte Carlo à partir d'un large panel de formalismes de modélisation dédiés à la description du comportement d'un système. Nous pouvons citer par exemple les automates d'états, les réseaux de Petri ou les arbres de défaillance que nous avons vu dans les sections précédentes, pour la résolution d'un problème d'estimation de performances de SdF.

Dans le cadre de la SdF, la simulation de Monte Carlo permet d'estimer les performances de SdF en simulant le comportement du système dans un modèle avec son processus actuel et les événements.

Les informations nécessaires pour lancer la simulation sont :

- Les taux de défaillance et de réparation de chaque composant et leurs différents modes de défaillance si nécessaire
- Une liste de valeurs des paramètres pour chaque composant nécessaires pour faire évoluer le modèle
- Le nombre d'histoires permettant de simuler plusieurs fois le comportement du système
- La durée de simulation d'une histoire, de l'ordre de la durée d'exploitation du système

Afin de voir apparaître un événement redouté rare comme l'apparition d'une défaillance un nombre suffisant de fois dans les histoires simulées, nous devons réaliser un grand nombre de simulations, ce qui implique des temps de calcul importants. Il est possible d'accélérer la simulation pour réduire ces temps de calcul grâce à certaines techniques (Dubi, 2000) (Niel & Craye, 2002). Elles se basent soit sur une diminution de la complexité du modèle, soit sur la réduction du nombre de scénarios à simuler, en favorisant l'apparition des événements rares. Cependant, ces techniques peuvent être difficiles à mettre en œuvre, selon les hypothèses à prendre en compte, qui peuvent être fortes et de nature à ne pas correspondre à la description réelle du système modélisé. La qualité des estimations se retrouve tâchée par cela et n'est pas forcément un bon révélateur des performances réelles. Il faut donc trouver un compromis pour choisir un nombre d'histoires suffisamment grand sans sacrifier un temps de calcul qui ne soit pas trop important.

Outre le fait de simuler suffisamment un grand nombre de fois le modèle, il faut également s'assurer que les valeurs estimées par la simulation de Monte Carlo soient cohérentes et se rapprochent de la valeur théorique. Pour cela, les intervalles de confiance vont nous permettre de jauger le niveau de précision des valeurs simulées par rapport à la vraie valeur de la variable aléatoire.

Un intervalle de confiance encadre une valeur réelle que nous cherchons à estimer à l'aide de mesures prises par un procédé aléatoire, dans le cadre d'une simulation. Cet intervalle permet de définir une marge d'erreur entre les résultats récupérés à partir de la simulation et la valeur réelle.

Un intervalle de confiance est associé à un niveau de confiance, en général sous la forme d'un pourcentage donné, par exemple un niveau de 95 %. Plus le niveau de confiance est élevé, plus l'intervalle autour de la valeur moyenne est réduit, donc plus précis, mais plus il peut être difficile de se retrouver dans l'intervalle si le nombre d'histoires n'est pas assez suffisant. Pour réduire le risque d'erreur, on peut élargir l'intervalle en baissant le niveau, mais la précision est moins importante et les écarts entre les valeurs estimées et la valeur réelle se creusent. Le niveau de confiance doit donc être choisi en équilibrant entre le besoin de précision et le nombre d'expériences réalisées.

Les intervalles de confiance sont souvent élaborés à partir d'un échantillon, c'est-à-dire une série de mesures indépendantes sur une population, notamment pour estimer des indicateurs statistiques comme la moyenne ou la variance. Mathématiquement, un intervalle de confiance est modélisé par un couple de variables aléatoires qui encadrent un paramètre réel.

La valeur obtenue $\bar{X}(t)$ est une moyenne empirique de la variable aléatoire du système que nous souhaitons estimer. Pour compléter cette valeur et avoir plus de précision sur le résultat, la variance et l'écart type permettent de s'assurer que la valeur estimée est proche de la valeur moyenne réelle. Ces deux valeurs permettront d'établir un intervalle de confiance.

Si $n_0$ est le nombre d'histoires de la simulation de Monte Carlo et $\bar{x}$ la moyenne empirique de la variable aléatoire x calculée, alors cette moyenne est calculée par :

$$\bar{x} = \frac{\sum_{i=1}^{n_0} x_i}{n_0} \tag{3.14}$$

avec $x_i$ la valeur de la variable aléatoire trouvée dans l'histoire *i*.

L'estimation de l'écart type $s_x$, et de la variance $s_x^2$ est donnée par :

$$s_x^2 = \frac{\sum_{i=1}^{n_0}(x_i - \bar{x})^2}{n_0 - 1} \tag{3.15}$$

$$s_x = \sqrt{s_x^2} \tag{3.16}$$

Il est alors possible d'obtenir un intervalle de confiance d'un niveau 1-α grâce à l'intervalle suivant :

$$\left[\bar{x} - t_\alpha \frac{s_x}{\sqrt{n_0}}, \bar{x} + t_\alpha \frac{s_x}{\sqrt{n_0}}\right] \tag{3.17}$$

où $t_\alpha$ est le paramètre associé au degré de confiance α pour un intervalle de confiance à 1-α.

### 3.4.3. Approche retenue pour estimer les indicateurs et les facteurs d'importance

Les sous-sections précédentes ont permis de définir deux approches pour évaluer les indicateurs SdF à partir du modèle décrivant l'architecture de contrôle-commande. La première approche se base sur un calcul analytique des indicateurs, la deuxième sur la simulation de Monte Carlo pour estimer une valeur statistique. Les deux approches peuvent être utilisées dans notre cas d'étude, avec plus ou moins de difficultés. Cependant, nous ne pourrons pas toutes les considérer, nous devons choisir une approche, en accord avec les contraintes et les besoins industriels posés précédemment.

Dans le cadre de nos travaux de recherche, nous avons sélectionné une approche par simulation de Monte Carlo que nous allons mettre en place en complément du modèle pour une architecture de contrôle-commande.

En effet, l'évaluation par la simulation est la plus approprié pour estimer des variables aléatoires comme les indicateurs de SdF des architectures de Schneider Electric, qui sont des systèmes complexes. La taille de ces derniers, ainsi que leur comportement dynamique et les liens de dépendance qui caractérisent l'ensemble des équipements sont un argument de poids dans le choix d'une approche par simulation.

Faire des calculs analytiques sur un système de grande taille peut vite entraîner une explosion combinatoire dans les possibilités qu'il serait difficile de maîtriser par de simples calculs sans simulation. Comme nous travaillons dans un contexte aussi incertain que la phase d'avant-vente, et parce que les ressources en temps et en opérateurs humains avec une expertise en SdF sont grandement limitées, il est plus judicieux de coupler le modèle descriptif de l'architecture de contrôle-commande avec une simulation de Monte Carlo, afin d'estimer les valeurs statistiques des indicateurs SdF et des facteurs d'importance que nous souhaitons évaluer. Les intervalles de confiance nous permettront de déterminer des valeurs proches de la valeur réelle et de ne pas perdre en précision grâce au niveau de confiance qui sera établie.

Ainsi, nous serons en mesure d'apporter une évaluation des performances de SdF qui allie à la fois qualité dans les résultats et rapidité dans les temps de calculs et l'affichage des résultats. C'est sur cette dernière approche que nous concluons ce chapitre sur l'état de l'art.

### 3.5. Conclusion

Ce chapitre avait pour but de voir les différentes approches qui nous sont proposées dans la littérature scientifique afin de pouvoir développer par la suite une méthode d'évaluation des performances de SdF des architectures de contrôle-commande.

Le premier choix à effectuer concerne le formalisme de modélisation du comportement fonctionnel et dysfonctionnel des architectures de contrôle-commande. Notre choix s'est porté sur les réseaux de Petri colorés et temporisés définis par Jensen, les CPN, qui permettent de modéliser des systèmes complexes de grande taille avec un comportement dynamique et sujet à des phénomènes aléatoires comme nous pouvons le rencontrer dans le domaine de la SdF.

Le deuxième choix concerne les indicateurs les plus pertinents pour répondre aux besoins des intégrateurs Schneider Electric en vue d'estimer les performances d'une architecture de contrôle-commande. Nous avons fait le choix de déterminer la disponibilité, la fiabilité et la maintenabilité et leurs métriques respectifs en tant qu'indicateurs SdF. Nous les complétons avec les facteurs d'importance suivants : le facteur de Birnbaum, le facteur d'importance critique, et le facteur d'amélioration potentielle, qui vont nous permettre de quantifier l'importance et l'impact des équipements de l'architecture sur le système global, afin d'aider dans la mise en place d'un plan de maintenance adéquat au contexte, en déterminant les parties critiques du système.

Enfin le dernier point concerne la méthode d'évaluation des indicateurs à partir du modèle et des indicateurs retenus. Nous avons choisi la simulation de Monte Carlo qui se prête bien à nos systèmes, notamment par le fait qu'il est fréquemment utilisé dans le cas de systèmes de grande taille et pour des problématiques alliant sûreté de fonctionnement et comportement fonctionnel et dysfonctionnel dynamique, donc des problèmes liés à la fiabilité dynamique.

Sur la base de ces choix argumentés, le chapitre suivant détaille le cadre formel de modélisation que nous avons proposé, la formalisation, au sein de ce cadre, des indicateurs retenus et enfin, les techniques d'évaluation déployées.

# Chapitre 4
# Modélisation et évaluation de la SdF d'une architecture de contrôle-commande

## 4.1. Introduction

Ce chapitre présente la méthode d'évaluation des performances de sûreté de fonctionnement pour une architecture de contrôle-commande dans un contexte dynamique et incertain. Celle-ci se basera sur les choix que nous avons fait en termes de modélisation dans le chapitre précédent. Une vue globale de la méthode, sera expliquée dans une section dédiée à la vue d'ensemble. Chaque section introduira les différentes parties de la méthode d'évaluation et comment sont construits les différents modèles sous-jacents. Nous verrons également comment les résultats sur les indicateurs d'intérêt sont récupérés et calculés, pour obtenir les performances que nous souhaitons afficher et montrer aux différentes personnes concernées par l'étude du système. Enfin, nous verrons comment sont calculés les facteurs d'importance dans notre évaluation et comment ils vont permettre d'aider dans l'interprétation des résultats.

## 4.2. Vue d'ensemble de l'évaluation

Avant de rentrer dans les détails de l'approche mise en place pour l'évaluation, une vue globale est présentée dans cette section. Cette vue d'ensemble est résumée dans la Fig. 4. 1. Elle comprend d'une part un cadre formel de modélisation des architectures permettant la construction des modèles servant de base à l'évaluation et, d'autre part, un processus d'évaluation des indicateurs basé sur de la simulation.

L'élaboration des modèles pour l'évaluation (partie gauche de la Fig. 4. 1) s'appuie sur :

- La définition d'une bibliothèque de modèles d'équipements associés aux différents constituants d'une architecture de contrôle-commande Schneider Electric ; par construction, ces modèles sont spécifiques pour chaque offreur de solutions d'automatismes.
- Des mécanismes d'instanciation et de paramétrage des modèles de la bibliothèque pour obtenir le modèle d'une architecture donnée. Si les paramètres des modèles peuvent être spécifiques à chaque offreur, le mécanisme de construction proposé qui s'appuie sur les couleurs des jetons du réseau de Petri est quant à lui générique.

La définition des modèles de la bibliothèque exploite les spécifications techniques des équipements d'automatismes présents dans l'architecture. Dans notre cas, celles-ci sont présentées dans un format UML et ont été présentées au chapitre 2. Les choix de modélisation effectués correspondent au savoir-faire de l'offreur de solution et des besoins qu'il exprime notamment vis à vis du niveau de finesse souhaité sur les modèles d'évaluation. Néanmoins, la section 4.4.1.

présente quelques guides de modélisation générique, notamment en ce qui concerne le modèle comportemental des équipements ainsi que l'intégration des caractéristiques dysfonctionnelles dans les modèles. La bibliothèque présentée reste bien entendu appliquée aux équipements Schneider (modules d'alimentation, processeur *CPU,* fond de panier d'alimentation et de communication *Backplane,* etc.) et doit pouvoir être enrichie au grès des évolutions technologiques.

La section 4.4.2. présente le processus d'élaboration des modèles d'architectures à partir des modèles d'équipements disponibles en bibliothèque.

Le principal point d'entrée de ce processus est la description informelle de l'architecture. Comme dit précédemment, cette description découle de la traduction des exigences techniques du client en une architecture de contrôle-commande qui est un ensemble constitué de composants matériels et logiciels. Cette description informelle comprend la liste des composants constituant le système, leur emplacement physique, s'ils sont locaux ou à distance de l'unité de décision, ainsi que les canaux de communication qui relient tous ces éléments. Toutes ces informations sont données à la fois sous la forme d'une vue graphique pour le client et le système intégrateur, et sous la forme textuelle qui sera repris en entrée du modèle d'évaluation, transparent aux utilisateurs de l'évaluation. Cette description est également enrichie par les spécifications techniques des produits (servant déjà à la définition de la bibliothèque) notamment pour identifier les chaines de dépendances dysfonctionnelles entre équipements.

La description informelle est complétée par la définition d'un contexte d'utilisation, propre au procédé industriel. Ce contexte donne les paramètres environnementaux nécessaires pour poser l'aspect fiabiliste du système. À l'issue de sa lecture, le modèle d'évaluation est en mesure de récupérer le taux de réparation $\mu$ pour chaque composant avec son temps moyen de réparation MTTR (Mean Time To Repair), et le taux de défaillance $\lambda$ avec son temps de fonctionnement avant une panne MTBF (Mean Time Between Failures). Les valeurs de ces deux taux pour chaque composant sont déterminées à partir de la température et de la charge de travail du composant, et d'un catalogue répertoriant les données MTBF.

Enfin, la liste des performances de sûreté que l'on souhaite évaluer fait également partie des éléments requis pour construire les modèles d'architectures. En effet, ces performances seront évaluées à l'aide d'observateurs sur les éléments des modèles (places, transitions…) qu'il sera nécessaire d'intégrer dans les modèles en prenant soin qu'ils ne soient pas intrusifs et ne remettent pas en cause l'évolution des modèles. Ils fourniront les données brutes sur le comportement de l'architecture (défaillance, réparation des composants…).

L'évaluation des performances de sûreté de fonctionnement est ensuite réalisée sur la base des modèles obtenus (partie droite de la Fig. 4. 1). Cette évaluation repose sur une évaluation statistique des indicateurs à l'aide de simulations de Monte-Carlo. La description des performances attendues utilisée pour définir les observateurs intégrés dans les modèles d'architectures sera également un élément d'entrée de cette phase de simulation. Elle comprend en effet les paramètres de la simulation (nombre d'histoires, critères d'arrêt, temps de simulation d'une histoire…). La section 4.5 décrit le processus d'évaluation et présente notamment des algorithmes permettant le calcul des indicateurs RAMS et des facteurs d'importance à partir des données brutes de simulation.

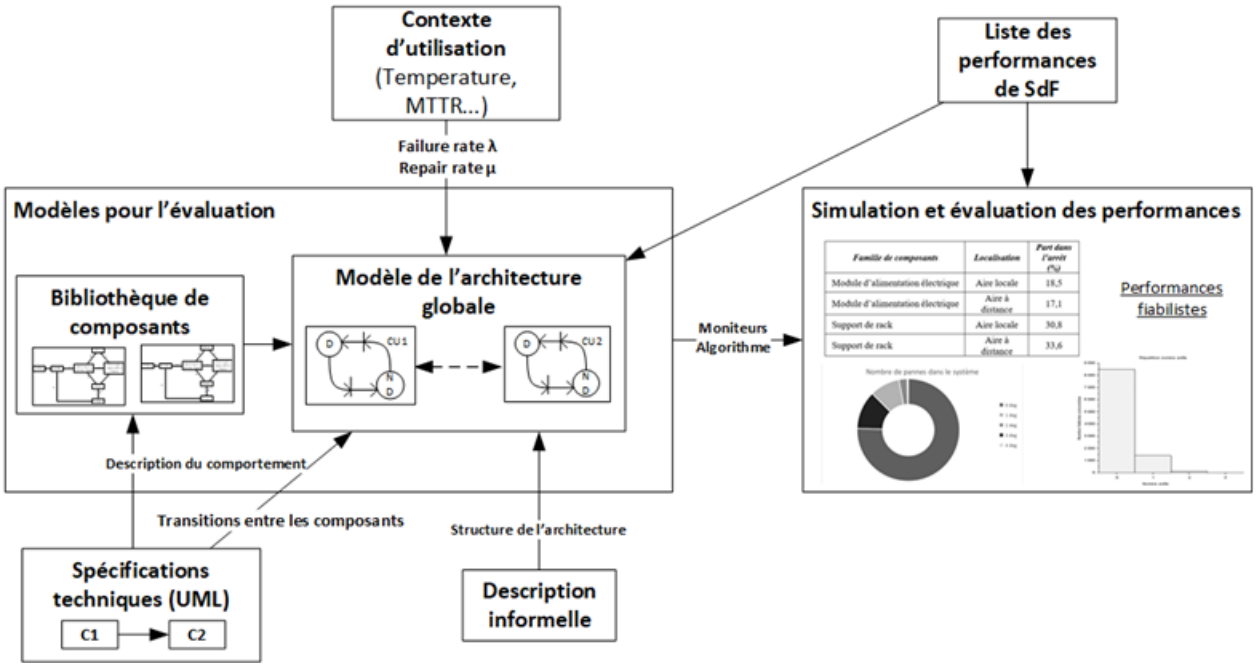



*Fig. 4. 1 : Vue d'ensemble de la méthode proposée*

L'ensemble du cadre de modélisation et d'évaluation proposé répond aux contraintes opérationnelles de mise en œuvre en phase d'avant-vente. L'élaboration d'une bibliothèque de modèles, eux-mêmes satisfaisant des critères de généricité, les mécanismes d'instanciation et de paramétrage permettant la construction automatique des modèles d'architectures, la proposition d'observateurs génériques utilisables ou pas selon les performances à évaluer contribuent à fournir aux intégrateurs des outils transparents et ne faisant pas appel à une expertise SdF déjà formalisée dans le cadre de modélisation et d'évaluation proposé.

### 4.3. Définition des couleurs dans un modèle CPN

Avant d'entrer en détails dans la construction des différentes parties du modèle d'évaluation pour une architecture de contrôle-commande, il est nécessaire de définir en premier lieu les notions de couleur, et par association des jetons colorés, dans un réseau de Petri stochastique dit coloré. En effet, ces derniers sont au cœur du modèle et vont pouvoir non seulement faire évoluer le système au cours du temps dans une simulation mais également dispenser des informations capitales sous la formes de données brutes utilisées dans le traitement post-simulation. Comprendre la nature des couleurs est important pour comprendre comment le modèle d'évaluation de l'architecture a été construit.

La définition des ensembles de couleurs dans le modèle va permettre de mettre en place deux principes utilisés dans cette étude. Le premier sera le principe de hiérarchisation des composants dans l'architecture. Chaque composant se verra attribuer un sous-modèle avec des règles propres à celui-ci. Les composants seront envoyés vers ce sous-modèle grâce aux couleurs qui vont donner son emplacement, sa connexion avec les autres composants et la famille à laquelle il appartient. Le deuxième principe se base sur le type de données nécessaires pour faire évoluer le système et pour

fournir les informations post-simulation, en complément de la temporisation du modèle constructeur.

En convention de notation, nous allons définir par la suite les ensembles de couleur par une lettre grecque majuscule, et les éléments (les couleurs) dans ces ensembles par une lettre grecque minuscule. Le marquage d'une place associée à un ensemble de couleur Φ sera défini, en suivant les notations usuelles de (Jensen, Kristensen, & Wells, 2007), par x`(ϕ) avec x le nombre de jetons dans la place et ϕ une couleur appartenant à l'ensemble Φ dans la place.

***Définition 1 : Couleur***

Soit $\Lambda_{i\_k}$ une couleur caractérisant un attribut k pour un composant (ou une donnée) *i*. Les éléments de cette couleur sont les variables $\lambda_{i\_k}$ qui peuvent prendre toutes les valeurs possibles définies au préalable, de type $F_k$, pour l'attribut k.

Par exemple, pour définir l'identifiant unique d'un composant, une couleur est créée, nommée $\Lambda_{composant\_identifiant}$ = "ID", dont les éléments $\lambda_{composant\_identifiant}$ peuvent prendre leur valeur dans l'ensemble des entiers $F_k = \mathbb{N}$, tous différents les uns des autres ($\lambda_{composant\ i\_identifiant} \neq \lambda_{composant\ j\_identifiant}$). Les identifiants uniques font partie du socle de l'instanciation que nous verrons dans les sections suivantes. L'identifiant unique permet sur un seul et unique modèle de composant générique d'instancier plusieurs composants de ce modèle tout en faisant le distinguo entre eux lors de la simulation.

Dans le logiciel CPN Tools, utilisé ici pour développer le modèle d'architecture pour l'évaluation, la primitive "Colset", faisant référence au langage Standard ML (SML, langage de programmation modulaire et fonctionnel) permettra l'association d'un type de données à une couleur (ML, 1998) (Paulson, 2010) (Milner, Tofte, Harper, & MacQueen, 1997). Nous verrons dans le chapitre dédié à l'application industrielle comment ce logiciel est appliqué dans le cadre de ces travaux de recherche, et du développement de la méthode d'évaluation.

***Définition 2 : Ensemble de couleurs associé à un composant***

Un ensemble de couleurs associé à un composant (ou une donnée) *i*, que nous notons $\Delta_i$, est défini par le produit cartésien des différentes couleurs $\Lambda_{i\_k}$ associées au composant (ou à la donnée). Cela se traduit par la formule suivante :

$$\Delta_i = \Lambda_{i\_k1} \times \Lambda_{i\_k2} \times \ldots \times \Lambda_{i\_kn}$$

Avec $k_n$ le nombre d'attributs utilisés pour caractériser le composant (ou la donnée) *i*. Les éléments constituant cet ensemble sont notés par les variables $\delta_i$ qui vont être présentes dans le modèle sous la forme de jetons qu'on appelle jetons colorés. Les jetons situés dans les places amont permettront de donner les conditions de validation du franchissement d'une transition et ce franchissement génèrera des jetons qui seront déposés dans les places aval en accord avec le comportement fonctionnel et dysfonctionnel décrit du système. Ces éléments sont constitués des valeurs possibles à partir des différents $F_{k1}$, $F_{k2}$, …, $F_{kn}$ du jeu de paramètres définissant un composant (ou une donnée), selon ce qui est retenu dans la couleur.

Dans le logiciel CPN Tools, la primitive "Record" du langage SML permettra de décrire ce genre d'ensemble de couleurs comme un produit cartésien d'un ensemble de couleurs dites élémentaires :

$$Colset\ \Delta_i = record\ \lambda_{i1} : \Lambda_{i1} *\ \lambda_{i2} : \Lambda_{i2} * \ldots *\ \lambda_{in} : \Lambda_{in}$$

Pour illustrer cette deuxième définition, prenons l'exemple d'un composant faisant parti de la famille des processeurs *CPU* (composant en charge de commander une unité de contrôle dans une architecture de contrôle-commande). Celui-ci possède plusieurs paramètres parmi lesquelles :

- son identifiant unique,
- son emplacement dans le système (sa localisation),
- sa famille (processeur),
- son état (en marche ou en panne par exemple),
- ses données fiabilistes (MTBF, MTTR et son niveau de dégradation).

Ces paramètres sont codés dans cinq couleurs qui sont définies comme étant de variables dont les valeurs sont choisies selon leur type notamment :

- un entier naturel unique pour la couleur se référant à l'identifiant
- un triplet d'identifiants uniques (entiers naturels) pour la couleur se référant à l'emplacement (un identifiant pour l'unité de contrôle, un pour la zone et un pour le support physique qui est le fond de panier du composant),
- une liste finie de familles pour la couleur se référant à la famille,
- une liste finie d'état dans lequel un composant peut se mettre pour la couleur se référant à l'état,
- un triplet comprenant un réel pour la MTBF, un réel pour la MTTR et un réel compris dans [0,1] pour le niveau de dégradation, pour la couleur se référant aux données fiabilistes.

Dans cet exemple, l'ensemble de couleurs définissant le processeur est donc :

$$Colset\ CPU = record\ \lambda_{CPUidentifiant} : \Lambda_{CPUidentifiant} *\ \lambda_{CPUemplacement} : \Lambda_{CPUemplacement} * \lambda_{CPUfamille} : \Lambda_{CPUfamille} * \lambda_{CPUétat} : \Lambda_{CPUétat} * \lambda_{CPUdata} : \Lambda_{CPUdata}$$

***Définition 3 : Ensemble de couleurs associé à un couple de composants***

L'ensemble de couleurs associé à un couple de composants *i* et j liés par un lien de type physique (alimentation électrique) ou fonctionnel (communication ou mission en lien) dans l'architecture, que nous notons $\Sigma_{ij}$, est défini par :

$$\Sigma_{ij} = \mathrm{I}_i \times \mathrm{I}_j \times P_{ij1} \times \ldots \times P_{ijn}$$

Avec $I_k$ la couleur caractérisant l'identifiant unique du composant k et $P_{ij}$ les attributs qui permettent de définir plus finement le lien existant entre les composants *i* et j. Les éléments de cet ensemble sont notés par les variables $\sigma_{ij}$ caractérisant les jetons colorés en charge de décrire ce lien entre composants dans le modèle. Cet ensemble de couleurs est utilisé notamment pour définir les liens de redondance entre deux composants ou encore les liens de communications entre un

composant maître et un composant esclave exécutant les ordres du premier ou bien envoyant des données pour le premier.

Dans le logiciel CPN Tools, nous faisons appel de nouveau à la primitive "Record" pour décrire cet ensemble de couleurs :

$$Colset\ \Sigma_{12} = record\ \iota_1 : I_1 *\ \iota_2 : I_2 * p_{121} : P_{121} * \ldots * p_{12n} : P_{12n}$$

Si nous prenons un exemple avec deux modules d'alimentation électriques redondants pour illustrer cette troisième définition, nous avons un couple d'identifiants uniques pour lier les deux modules et caractériser ce lien de redondance.

Dans ce cas, l'ensemble de couleurs définissant ce lien de redondance est :

$$Colset\ Module_{12} = record\ \iota_{id1} : \mathrm{I}_{identifiant} * \iota_{id2} : I_{identifiant} * p_{emplacement12} : P_{emplacement}$$

En plus de la définition des composants dans le système, ainsi que leur lien de dépendance, il existe également des ensembles de couleurs pour définir les événements arrivant sur le système, du type d'une défaillance, réparation ou mise sous/hors tension. Comme la caractérisation d'un composant dans la deuxième définition, un jeton coloré contiendra l'événement, sous la forme d'un produit cartésien de couleurs. Ces types de jetons feront office de message pour faire évoluer le système au cours du temps, mais également pour fournir des informations en données de sortie de la simulation de Monte Carlo.

Avec toutes ces définitions, il apparaît que les couleurs et les jetons colorés associés ont une place prépondérante dans le modèle et son évolution, et font partie intégrante du mécanisme d'évaluation pour récupérer les performances fiabilistes d'une architecture de contrôle-commande que nous souhaitons fournir au client et au système intégrateur.

### 4.4. Construction du modèle d'architecture

Cette section présente la construction des modèles pour l'évaluation de l'architecture. Comme dit précédemment, ce modèle se base sur l'assemblage de briques de composants, un ensemble de modèle RdP de composants, qui une fois instanciées forme l'architecture de contrôle-commande. Les prochaines sous-sections vont se focaliser sur la construction de ces modèles de composants, à partir des spécifications techniques, et ensuite sur la construction du système entier. Ces explications permettront d'introduire dans la prochaine section la mise en place de la simulation de Monte Carlo adaptée à ce modèle d'évaluation, ainsi que des observateurs de performance et du traitement post-simulation.

#### 4.4.1. Modèle d'évaluation pour un composant élémentaire

Le modèle d'architecture va se baser principalement sur la description informelle de l'architecture de contrôle-commande ainsi que sur les spécifications techniques décrites précédemment par les diagrammes UML. Les composants élémentaires sont également basés sur ces informations. Nous allons voir ici comment le modèle pour chaque composant est construit à partir de ces données d'entrée et du format utilisé pour la modélisation, à savoir les réseaux de Petri.

Ici, un composant élémentaire est un composant classé dans une catégorie unique de familles parmi une liste finie. Chaque composant élémentaire possède son propre modèle de réseau de Petri (RdP). Une famille est un ensemble d'équipements ayant la même mission et pouvant utiliser des fonctions analogues, mais dont les caractéristiques techniques peuvent varier sensiblement d'un équipement à un autre, notamment par rapport à leur taux de défaillance qui diffère selon la gamme du produit : deux composants dans une même famille n'ont pas forcément le même niveau de robustesse face à des défaillances ou des dégradations. Par exemple, nous pouvons mettre dans une seule famille nommée "CPU" l'ensemble de la gamme des processeurs disponibles chez Schneider Electric. Les caractéristiques techniques peuvent différer mais la mission principale reste la même, à savoir piloter et gérer une unité de contrôle de l'architecture de contrôle-commande.

Nous avons évoqué précédemment que chaque famille de composants se comporte différemment. Ces modèles de familles se basent sur le diagramme d'états-transitions. Par exemple, un processeur n'aura pas tout à fait les mêmes états qu'un module d'alimentation. Ici, nous décrivons le comportement attendu d'une famille en prenant en compte les aspects de sûreté de fonctionnement, les états concernant la partie purement applicative ne sont pas représentés. Nous pouvons classer les composants dans ces familles, que nous pouvons regrouper dans quatre catégories. Dans cette étude, les quatre grandes catégories sont :

- Les composants en charge de **l'alimentation.** Cette catégorie comprend la famille des modules d'alimentation "Power Supply" et la famille des supports d'alimentation des modules que sont les fonds de panier "Backplane".
- Les composants en charge du **contrôle**. Il s'agit de la famille des processeurs et co-processeurs "CPU", et les modules gérant les composants placés dans des supports déportés faisant office d'interface avec le processeur, appelés "CRA".
- Les composants en charge de la **communication.** Cette catégorie est constituée d'une famille pour les modules de communication présents sur des fonds de panier et d'une famille pour les modules de communication avec une alimentation indépendante tels que les switchs ou les gateway.
- Les composants correspondant aux **équipements terminaux**. Les trois familles comprises dans cette catégorie sont les "Devices", les modules d'entrée-sorties et experts pour une tâche spécifique dédié au procédé (comptage…) "Process module", et les îlots de module d'entrée-sortie "STB".

Tout comme le modèle global de l'architecture, le modèle d'un composant élémentaire aura des entrées et des sorties qu'il faudra pouvoir gérer pour le faire fonctionner correctement. En entrée de celui-ci, nous retrouvons la liste des composants faisant partis de la même famille, à partir de la description informelle de l'architecture de contrôle-commande et qui y seront instanciés. À l'interface du modèle d'un composant, nous aurons également la récupération des événements arrivant sur les autres composants du système pouvant impacter le composant en question, car il est lié à ceux-ci par des liens fonctionnels et dysfonctionnels, ainsi que l'envoi des événements étant arrivés sur ce dernier et ayant un impact sur les autres composants. En sortie de ce modèle, nous avons la liste des événements étant survenus sur le composant élémentaire, avec les informations nécessaires, sous la forme d'attributs, pour pouvoir réaliser l'estimation des

performances fiabilistes du système. La Fig. 4. 2 donne l'aperçu des données en entrée et en sortie du modèle RdP d'un composant élémentaire.

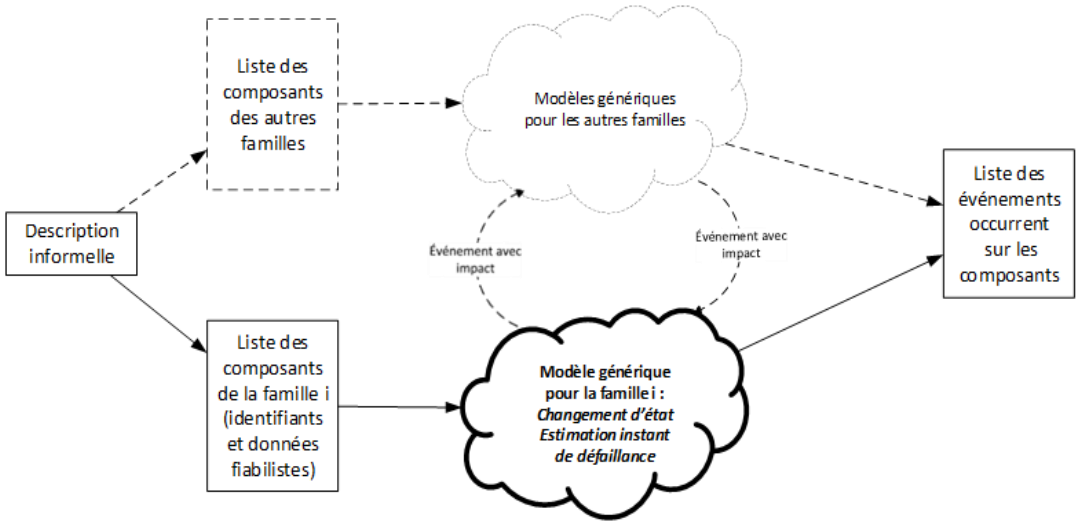



*Fig. 4. 2 : Entrées-sorties du modèle pour un composant*

Le modèle du composant est composé de trois parties réalisés grâce aux RdP, comme nous pouvons le voir dans la Fig. 4. 3. Il y a la gestion du changement d'état du composant, la gestion des modes de fonctionnement du composant, et les estimations réalisées autour du calcul du prochain instant de défaillance et de réparation du composant. Nous allons voir maintenant ces trois aspects du modèle.

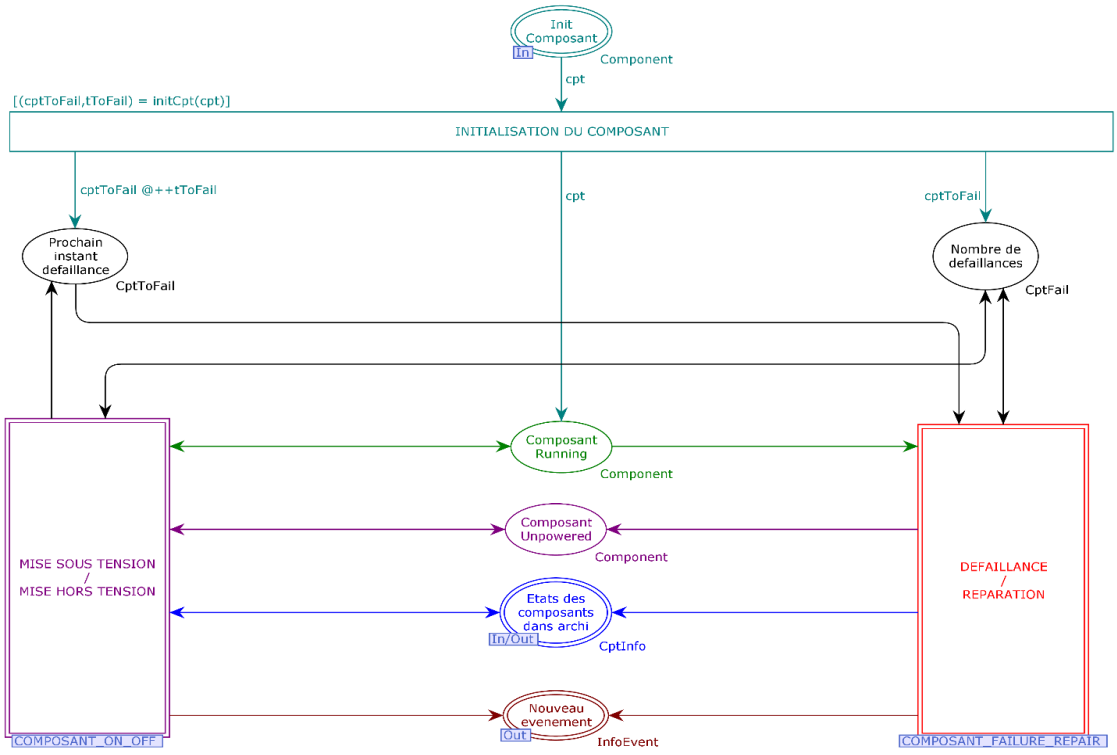



*Fig. 4. 3 : Modèle RdP d'un composant dans l'architecture de contrôle-commande*

Chaque composant constituant l'architecture est stocké dans un jeton coloré qui sera assigné dans le modèle de sa famille au moment de l'initialisation. Ce jeton comporte les informations qui sont nécessaires pour identifier le bon composant, grâce à l'identifiant unique. La construction de ce jeton lors de l'instanciation sera détaillée dans les sections suivantes.

#### *4.4.1.1. Changement d'état*

Le changement d'état d'un composant élémentaire intègre les conditions à la fois intrinsèques au composant ainsi que l'impact des autres composants dans le système, comme cela a été montré dans les diagrammes de séquences du chapitre 2. Ces changements donnent lieu à une transition du composant passant d'un état de marche à un état de panne ou de mise en hors tension par exemple, et vice versa, disponibles dans le diagramme d'état-transition. Les conditions qui sont propres à la famille du composant vont compléter le diagramme d'état qui restait général et valide pour tous types de composant.

Après une phase d'instanciation où tous les composants sont injectés dans le sous-modèle correspondant à leur famille, les conditions pour le franchissement des transitions déterminent dans quel cas, le composant doit changer d'état. Le composant peut se mettre dans plusieurs états qui sont (cf. Fig. 4. 3) :

- En marche *"Powered"/"Running"*. Il s'agit de l'état opérationnel dans lequel l'utilisateur du système souhaite que les composants se trouvent le plus souvent possible, et ce pendant toute la durée d'utilisation.
- En hors tension *"Unpowered"*, dont le changement se fait au sein d'un sous-modèle *"MISE SOUS TENSION/MISE HORS TENSION"*.
- En panne *"Composant Failed"*, un état géré à l'intérieur du sous-modèle *"DEFAILLANCE/REPARATION"* (Fig. 4. 4).

Les conditions dans lesquelles le composant peut changer d'état après un événement pouvant impacter potentiellement ses performances fiabilistes sont décrites dans plusieurs sous-modèles regroupés au sein d'une transition de substitution. Une transition de substitution est une notion qui s'appuie sur la structuration modulaire possible des RdP, les réseaux de Petri hiérarchiques, dont l'outil CPN Tools permet de construire. Les transitions de substitution, sont eux-mêmes constitués d'un réseau de Petri, dans lequel on peut retrouver, si besoin, d'autres transitions de substitution. Elles permettent donc de décomposer le modèle RdP en plusieurs niveaux, où chaque niveau est un sous-modèle RdP également, donnant une partie de la description du système.

Changement intrinsèque au composant

Les changements intrinsèques au composant concernent principalement ses défaillances et ses réparations, ainsi que sa remise en marche à la suite de sa propre réparation.

Prenons l'exemple de la transition de substitution ayant la charge de gérer les défaillances et les réparations d'un composant. La probabilité d'occurrence de la panne dépend de deux éléments : l'état du composant à un instant donné et l'estimation de son prochain instant de défaillance, suivant les conditions actuelles du système. La Fig. 4. 4 montre le sous-modèle RdP associé à la transition de substitution gérant les pannes et les réparations.

Pour que la transition *"DEFAILLANCE DU COMPOSANT"* soit valide et franchissable, les trois conditions suivantes doivent être remplies :

- Le composant doit être en état de marche précédemment, ce qui se caractérise par la présence d'un jeton coloré dans la place en amont *"Composant Running"*, selon la définition 2 vue dans la section 4.3. L'ensemble de couleurs dans le jeton doit avoir l'attribut correspond à l'état mis à *"Powered"/"Running"*.
- L'instant actuel de la simulation en cours doit correspondre à l'estimation faite de l'instant de la défaillance arrivant sur le composant. Pour cela, il faut que l'instant du jeton coloré temporisé dans la place *"Prochain instant défaillance"* correspondant au composant en question et l'instant de la simulation soit égaux (que l'horloge interne du simulateur soit égale au temps du jeton temporisé).
- L'estimation de l'instant de la défaillance réalisée doit être la dernière estimation qui a été faite. Cet attribut est comparé au numéro de la prédiction faite sauvegardé dans le jeton de la place *"Prochain instant défaillance"*, également un attribut dans le jeton coloré de cette place.

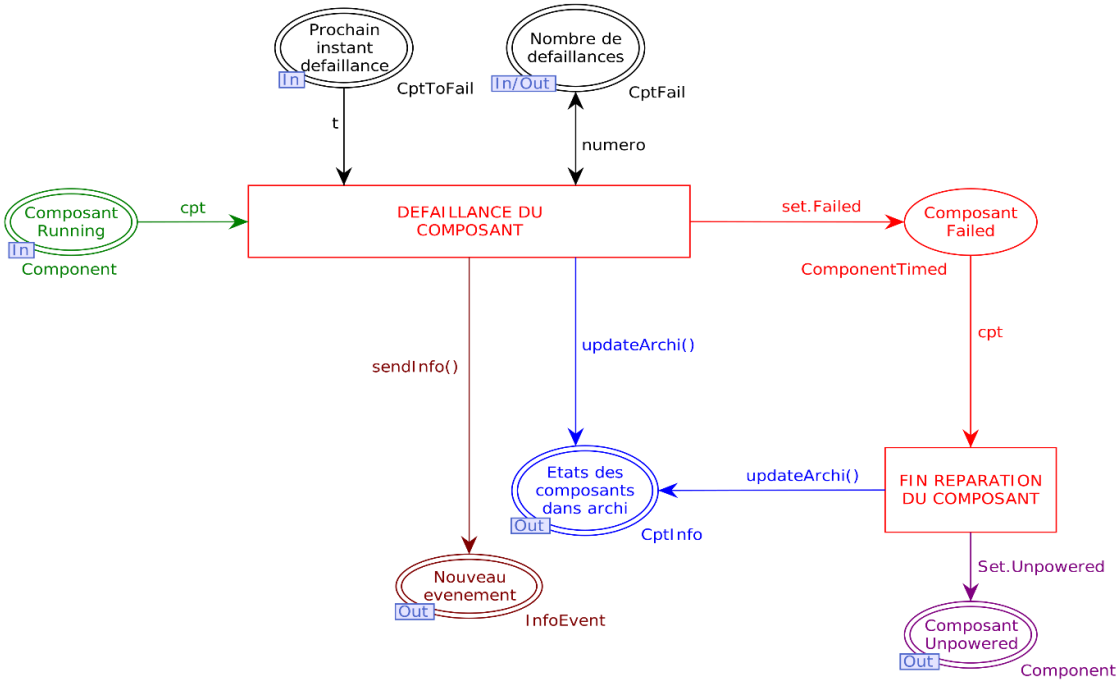



*Fig. 4. 4 : Transition de substitution pour gérer la défaillance et la réparation d'un composant*

Si toutes les conditions sont validées, alors le composant passe dans l'état de panne *"Composant Failed"* et se met en attente de sa réparation. Comme nous le voyons dans cet exemple, c'est un changement intrinsèque au composant élémentaire.

Changement externe dû au composant, lié à son environnement et à ses liens de dépendance

Il est à noter que le changement intrinsèque d'un composant élémentaire peut se répercuter sur les autres composants dans le système dû au fait des liens d'interdépendance entre composants. Ainsi, il faut dans le même temps informer les autres composants ayant un lien d'interdépendance

de la défaillance du composant. Cette action est réalisée à travers les jetons qui passent dans la place *"Etats des composants dans archi"* pour que les autres composants changent d'état s'ils sont concernés. Un cas d'illustration serait lorsqu'un module d'alimentation électrique est défaillant. Alors tous les composants qui sont alimentés par ce module changent d'état et sont mis en hors tension. Le passage de la transition *"DEFAILLANCE DU COMPOSANT"* dans le modèle du module d'alimentation va générer un jeton coloré qui va donner l'information que le module en question est tombé en panne. Ce jeton sera consommé par tous les composants touchés par la défaillance du module.

La place *"Etats des composants dans archi"* représente les échanges d'informations entre les équipements qui se basent sur les diagrammes de séquence. Lorsque la défaillance d'un composant $C_1$ arrive, un jeton coloré est envoyé au reste du système. Si un autre composant $C_2$ est impacté par $C_1$, alors il va changer d'état. Les conditions dans les transitions du modèle de la famille de $C_2$ sont écrites à partir de fonctions SML qui permettent de vérifier si ces dernières sont validées pour franchir la transition et changer l'état de $C_2$.

La définition de ce jeton coloré qui contient le changement survenu dans l'architecture globale et stocké dans la place *"Etats des composants dans archi"* donne toutes les informations de l'événement arrivé (défaillance, réparation, mise sous tension ou mise en hors tension), grâce à un ensemble de couleurs dont les attributs sont les suivants :

- L'identifiant unique du composant source de l'événement
- Son emplacement dans l'architecture (identifiant de son unité de contrôle, de son aire et de son support d'alimentation)
- La famille du composant source, pour vérifier si c'est un composant qui impacte l'état du composant qui va recevoir l'information
- L'état du composant source résultant du changement, pour déterminer quel type d'événement est arrivé
- Un booléen pour indiquer si le composant source a un autre composant qui prend le relais ou non, c'est-à-dire s'il est redondé
- Un booléen pour indiquer si l'événement est intrinsèque au composant source ou vient d'un autre composant en amont de celui considéré comme la source
- L'instant où est arrivé l'événement

Notification de changement d'état d'un composant élémentaire

En complément de ceci, les données découlant de l'événement arrivé sont également stockées dans un fichier externe qui servira pour le traitement post-simulation, grâce à la place *"Nouveau evenement"* : l'événement sera sauvegardé avec entre autres sa nature, le composant source et l'instant d'occurrence. Nous verrons dans une prochaine section, comment le fichier est utilisé pour réaliser le traitement post-simulation.

#### *4.4.1.2. Gestion des modes de fonctionnement*

Nous avons vu précédemment que tous les composants n'ont pas le même comportement, et que par conséquent, il faut les classifier par famille. Certaines familles, en plus des changements

d’état, peuvent avoir des modes particuliers dont le mécanisme sera décrit également à travers des transitions de substitution.

Pour comprendre rapidement cette distinction de modes, prenons l’exemple de la famille des processeurs et coprocesseur "CPU". Il est possible de décrire les différents processeurs existants à l’heure actuelle dans les architectures de contrôle-commande proposés par Schneider Electric en trois catégories (Fig. 4. 5) : les processeurs autonomes (dits "standalone"), les processeurs en redondance (dits "hot standby") et les processeurs de sécurité (dits "safety").


*Fig. 4. 5 : Catégories (et illustration) des processeurs existant au sein des produits proposés par Schneider Electric*

Les processeurs autonomes sont les processeurs où un module gère à lui seul une unité de contrôle pour une configuration (Fig. 4. 6). Il reçoit des données en permanence pour faire fonctionner la partie du procédé industriel dont la charge incombe à l’unité de contrôle-commande en question, et fait le lien avec la salle de contrôle-commande et toute la partie en charge de la supervision. Des ports de communication (au nombre de trois) permettent de réaliser tous ces échanges de données.

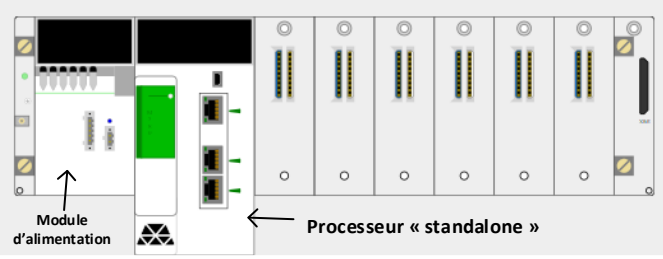



*Fig. 4. 6 : Configuration d’un processeur autonome*

Les processeurs redondants sont regroupés en deux modules identiques dans deux supports d’alimentation séparés (Fig. 4. 7), dont l’un est défini comme le primaire et l’autre comme le secondaire. Les deux processeurs sont en redondance chaude. Le module primaire met à jour l’intégralité des entrées-sorties en temps réel à chaque cycle et envoie les instructions et rapports au reste de l’architecture. Le secondaire, dans le même temps, met uniquement à jour ses variables à chaque cycle, afin de conserver le dernier état du système et de pouvoir prendre le relais à tout moment si le primaire devient hors service. À la différence du processeur basique, un processeur redondant possède un port de communication supplémentaire pour connecter le primaire et le secondaire, afin de surveiller l’état de l’autre module et de déterminer s’il faut transférer le primaire vers le secondaire ou non.

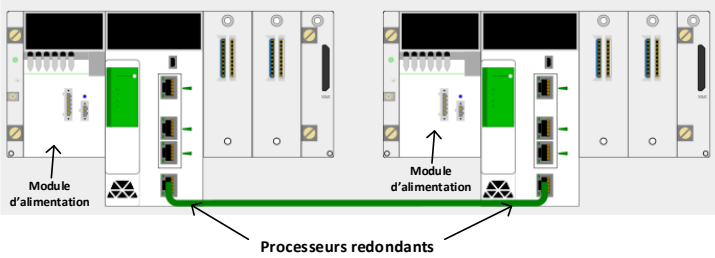



*Fig. 4. 7 : Configuration d'un ensemble de processeurs redondants*

Les processeurs safety sont un ensemble composé d'un processeur qui dédie un pan de sa mémoire à la partie strictement applicative faisant fonctionner le procédé industriel et un autre sur l'implémentation des fonctions de sécurité pour le système (sous la forme de SIF, ou fonctions instrumentées de sécurité), et d'un coprocesseur dédié uniquement à la partie sécurité du système (Fig. 4. 8). Les deux processeurs communiquent entre eux pour la partie sécurité, pour mettre en place un système de vote 2oo2 permettant d'éviter les fausses alertes principalement et de ne pas déclencher sans raison les mécanismes de sécurité. Étant donné que le coprocesseur n'a pas besoin de communiquer avec le reste de l'architecture, il ne possède pas de ports. La communication avec le processeur se fait à travers le support qui contient les deux modules, avec un canal consacré à la communication entre les composants dans le même support. Pour des raisons de visibilité, les processeurs safety ont un module de couleur différente, rouge par convention.

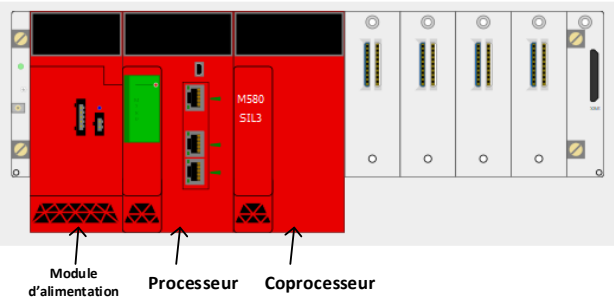



*Fig. 4. 8 : Configuration d'un ensemble de processeur et coprocesseur safety*

Si nous prenons la gestion du mode redondance chaude pour les processeurs redondants, la défaillance d'un processeur n'entraîne pas forcément la mise en hors service du deuxième processeur qui est en secours. Ce comportement est différent des processeurs safety dédié à la sécurité du procédé industriel. Dès lors que le processeur ou le coprocesseur est défaillant, l'ensemble est considéré comme hors service, ne pouvant plus assurer sa mission de protection, il n'est pas envisageable de continuer à faire fonctionner le procédé dans des conditions sûres. Dans ce cas le système, suivant comment il a été configuré, va se mettre dans une position de repli n'impliquant pas de danger certain, en attendant que l'ensemble des processeurs soit remis en service de nouveau.

Dans l'exemple donné dans la Fig. 4. 9, les processeurs ont deux modes supplémentaires en plus de la configuration basique : le mode "Stand*by Mode*" et le mode "*Safety Mode*", chacun étant décrit par une transition de substitution. Le mode "*Standby Mode*" correspond au cas où le processeur est redondé par un autre processeur dans l'unité de contrôle. Le mode "*Safety Mode*"

correspond au cas où le processeur est secondé par un coprocesseur afin de mettre en place un système de sécurité en plus d'un système purement de contrôle.

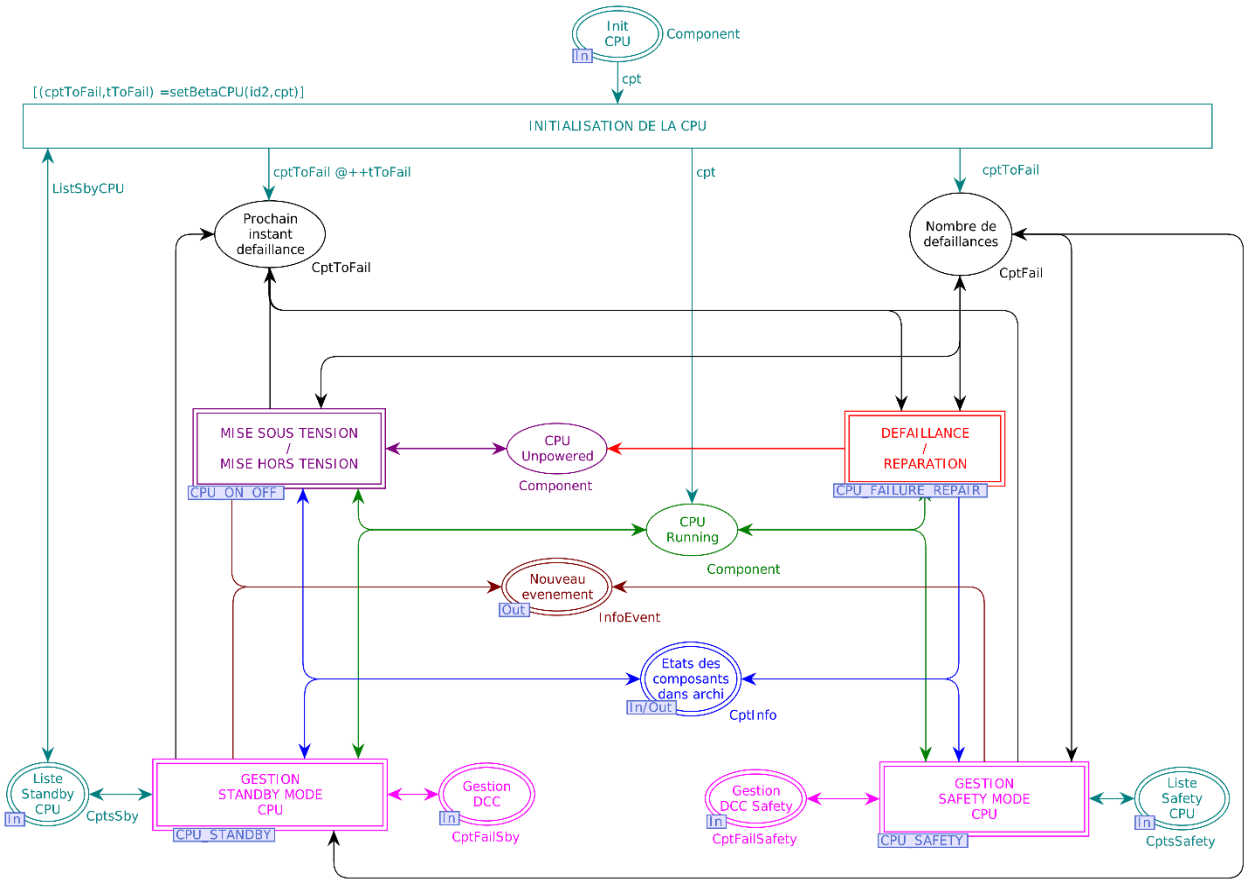



*Fig. 4. 9 : Modèle RdP du composant "CPU" de la famille Contrôle avec la prise en compte des modes particuliers*

Dans le cadre de la gestion de la redondance, se faisant dans une transition de substitution dont le détail est présenté dans la Fig. 4. 10, trois transitions de substitution d'un niveau hiérarchique inférieur vont permettre de prendre en compte les aspects liés à cette notion de redondance :

- La gestion des défaillances de cause commune, défaillance qui impacte simultanément les deux processeurs. Cette défaillance est la plus redoutée dans un système redondant car elle rend caduque ce principe de renfort au niveau du composant. Elle met en réparation les deux processeurs au même instant.
- La gestion des défaillances indépendantes qui impacte un seul composant à la fois parmi le couple. Si une défaillance de ce type arrive sur l'un des composants redondés, alors une vérification est faite. Si c'est le composant primaire qui est touché, dans ce cas un basculement se fait sur le deuxième. Si c'est le composant secondaire, aucune discontinuité est relevée et le secondaire est uniquement mis en réparation.
- La gestion du calcul du prochain instant de défaillance de cause commune de l'ensemble de processeurs redondants. Ce modèle est en charge de l'estimation de la prochaine défaillance de cause commune et injecte l'instant estimé dans le modèle se chargeant de la gestion de ces défaillances.

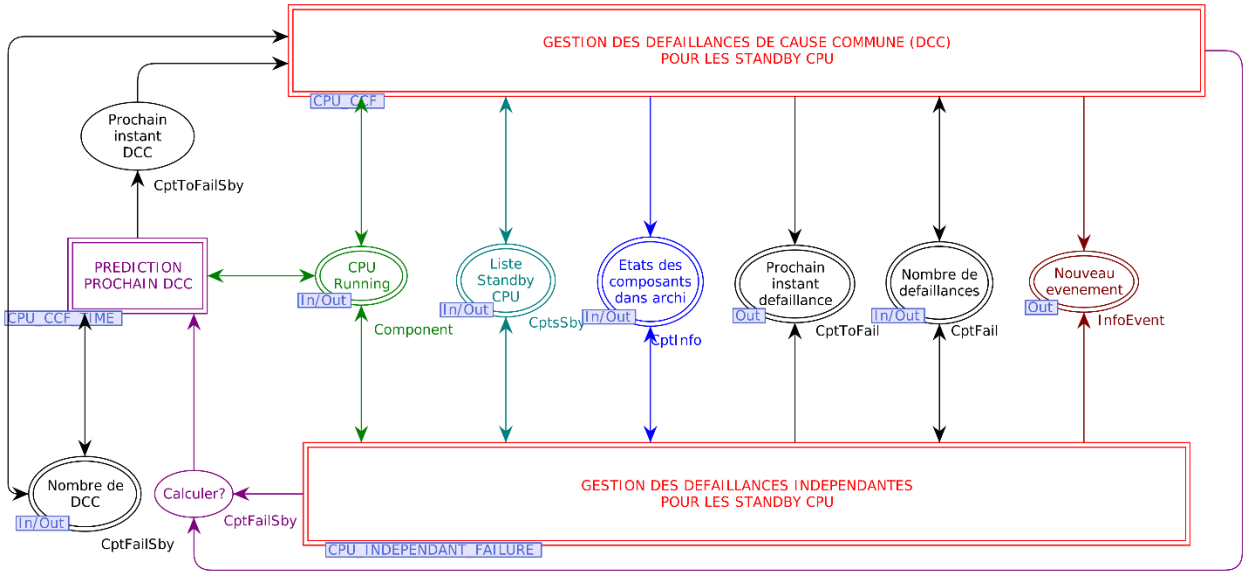



*Fig. 4. 10 : Modèle RdP pour la gestion de la redondance d'un processeur "CPU"*

Dans le cadre de la gestion du mode "Safety" pour un processeur, les règles de fonctionnement pour le couple processeur/coprocesseur sont différentes de celui du couple processeur primaire / processeur secondaire pour un processeur redondant. Si le processeur ou le coprocesseur tombe en panne, c'est l'ensemble du système qui est défaillant, que ce soit pour la partie qui gère le procédé ou la partie dédié à la sécurité des fonctions. Ceci est géré dans la transition de substitution concernée par ce mode dans la Fig. 4. 9.

#### *4.4.1.3. Estimation du prochain instant de défaillance*

Si la majeure partie du modèle d'évaluation d'un composant élémentaire concerne la description du comportement du composant, une partie est dédiée au calcul du prochain instant de défaillance intrinsèque au composant. Ce calcul est réalisé en fonction de son taux de défaillance λ, récupéré à partir de la MTBF du composant. Par exemple, pour des constituants électroniques que l'architecture de contrôle-commande peut contenir, nous pouvons calculer le taux de défaillance du composant à partir de sa MTBF par :

$$\lambda = \frac{1}{MTBF} \tag{4.1}$$

Le calcul du prochain instant de défaillance d'un composant va donc dépendre de son taux de défaillance principalement. Pour les composants électroniques qui suivent la loi exponentielle (équation ci-dessus), étant sans mémoire, il suffit à l'entrée dans un nouvel état de considérer un nouveau taux de défaillance correspondant au changement.

Dans le cas de composants qui vieillissent dans le temps, des composants mécaniques par exemple, le calcul se basera également sur le niveau de dégradation déjà présent sur le module, ainsi que de l'état des autres composants et qui ont un impact sur lui (Rieker, Zeiler, & Bertsche,

2015) (Boyer, Brînzei, Do, & Pétin, 2017) (Belkacem, Simeu-Abazi, Gascard, & Messaoud, 2017). Pour comprendre comment ce calcul est réalisé prenons un exemple simple.

Considérons un système constitué de deux composants nommés $C_1$ et $C_2$. Nous prenons pour hypothèse que le deuxième composant $C_2$ change de mode de dégradation dès l'instant où le premier composant $C_1$ est défaillant. Dans ce cas de figure, le modèle dynamique régi pour décrire le comportement de $C_2$ dépend à la fois de son propre taux de défaillance mais également de l'état de $C_1$.

Nous pouvons imaginer par exemple un système mécanique composé de deux pompes fonctionnant en même temps pour délivrer chacune 50% du débit requis d'un fluide se déversant dans un réservoir. Lorsqu'une des deux pompes tombe en panne, la deuxième pompe se met à plein régime afin de fournir 100% du débit. Nous voyons dans cet exemple, que le deuxième composant va se dégrader plus rapidement quand il sera à pleine charge, et donc l'estimation de son prochain instant de défaillance doit être mis à jour au moment du changement d'état pour respecter ce nouveau mode de dégradation.

Pour réestimer le prochain instant de défaillance, l'estimation se base sur l'âge (plus communément niveau de dégradation) du composant à l'instant t de basculement. L'âge peut être vu également comme la probabilité de défaillance achevée F(t) à t, où F est la fonction de répartition calculée à partir de la loi de probabilité, définissant le profil de vieillissement du composant, telle que la loi de Weibull, ou la loi Normale. Nous calculons alors l'âge du composant à l'instant t grâce à la fonction de répartition et nous réajustons l'estimation de la prochaine panne à partir de ce niveau de dégradation lors des changements d'état. Nous considérons ensuite que la défaillance intervient lorsque le composant atteint un certain seuil pour son âge et nous déduisons l'instant de défaillance par rapport à ce seuil en déterminant le temps associé à ce seuil et suivant le nouveau profil de vieillissement du composant.

Rappelons ici que l'âge d'un composant à un instant t donné, dans un état *i* et s'il n'y a pas eu de changement d'état entre temps, est défini par :

$$Age_i\,(t) = F\left(t - t_{origine}\right) \tag{4.2}$$

avec :

- $t_{origine}$ l'instant où le composant s'est mis dans l'état *i*
- F la fonction de répartition décrivant le profil de dégradation du composant dans l'état *i* (Weibull, etc.)
- $Age_i$ (t) le niveau de dégradation du composant dans l'état *i* à l'instant t

Nous considérons que pour chaque état, le composant suit une loi usuelle dont les paramètres sont modifiés pour chaque nouvel état, ce qui engendre une mise à jour de la probabilité de défaillance achevée F(t) au moment du changement d'état (cf. Fig. 4. 11). Il faut donc mettre à jour les données à cet instant afin de recouper cette probabilité et de calculer le prochain instant de défaillance.

Lorsqu'il y a changement d'état à l'instant t, nous calculons l'âge atteint du composant à l'entrée du nouvel état :

$$Age_{entrée} = F\left(t - t_{origine}\right) \tag{4.3}$$

avec :

- $t_{origine}$ l'instant où le composant s'était mis dans l'état précédant le changement
- F la fonction de répartition du profil de dégradation de l'état précédent
- $Age_{entrée}$ (t) le niveau de dégradation du composant dans l'état *i* à l'instant t

A partir de cet âge, nous pouvons alors déterminer l'instant où le composant va sortir de cet état $t_{exit}$, et tomber en panne s'il n'y a pas eu de changement d'état entre temps, grâce à l'équation suivante :

$$\xi = 1 - \frac{1 - F\left(t_{exit} - t_{origine}\right)}{1 - Age_{entrée}} \tag{4.4}$$

avec :

- *ξ tirée selon la loi uniforme dans l'intervalle [0,1]*
- $t_{origine}$ l'instant où le composant s'était mis dans l'état précédant le changement
- F la fonction de répartition du profil de dégradation de l'état précédent
- $Age_{entrée}$ (t) le niveau de dégradation du composant dans l'état courant *i* à l'instant t

L'instant $t_{exit}$ déduit de cette équation est le nouvel instant de prochaine défaillance du composant à partir de l'état courant *i*, qui a été réactualisé suite à un changement de son état.

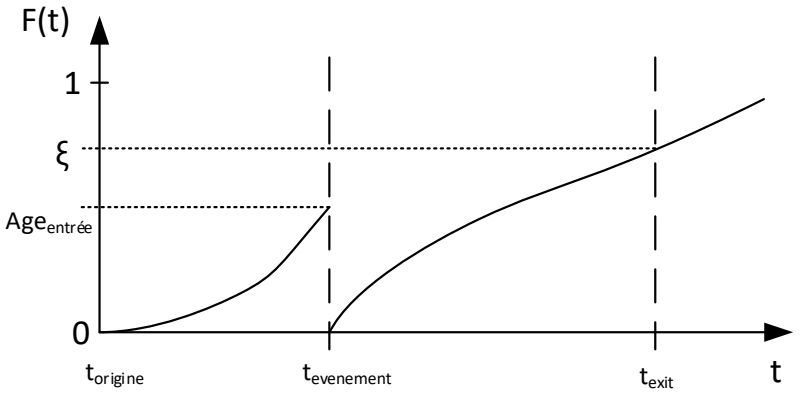



*Fig. 4. 11 : Probabilité de défaillance achevée et mise à jour de sa courbe après un changement d'état*

Si nous reprenons le système de deux composants que nous avons décrits au début de cette explication, si $C_1$ tombe en panne, nous avons $C_2$ qui prend le relai et passe dans un état de sursollicitation. Par conséquent ce dernier se dégrade plus rapidement, car sa sollicitation est plus importante. Le système est défaillant lorsque les deux composants sont en panne.

Sur la figure ci-après (Fig. 4. 12), nous pouvons observer l'évolution de l'âge de C2 en fonction du temps. Au début, la progression est lente, tant que $C_1$ est encore opérationnel, mais s'accélère à partir du moment où $C_1$ tombe en panne. Nous avons la superposition de deux modes pour $C_2$ : un régime nominal (dans l'exemple de la pompe, 50% du débit) ou partiellement chargé, et un régime maximal (pour la pompe, 100% du débit) ou entièrement chargé. C'est ce changement que nous avons modélisé et pris en compte dans nos calculs et qui illustre parfaitement la complexité de la fiabilité dite dynamique.

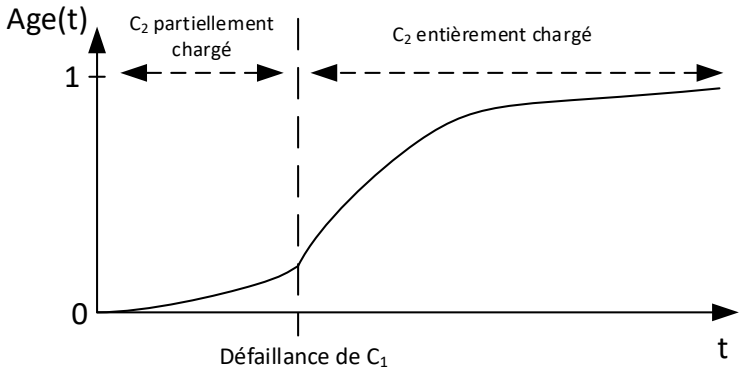



*Fig. 4. 12 : Évolution du niveau de dégradation du composant $C_2$ changeant d'état à la suite de la défaillance du composant $C_1$*

### 4.4.2. Modèle d'évaluation de l'architecture globale

Les sous-modèles RdP constituant les différents composants qui peuvent composer une architecture de contrôle-commande ont été définis dans les sous-sections précédentes. Ces modèles sont des briques qui vont constituer le modèle de l'architecture dans son intégralité.

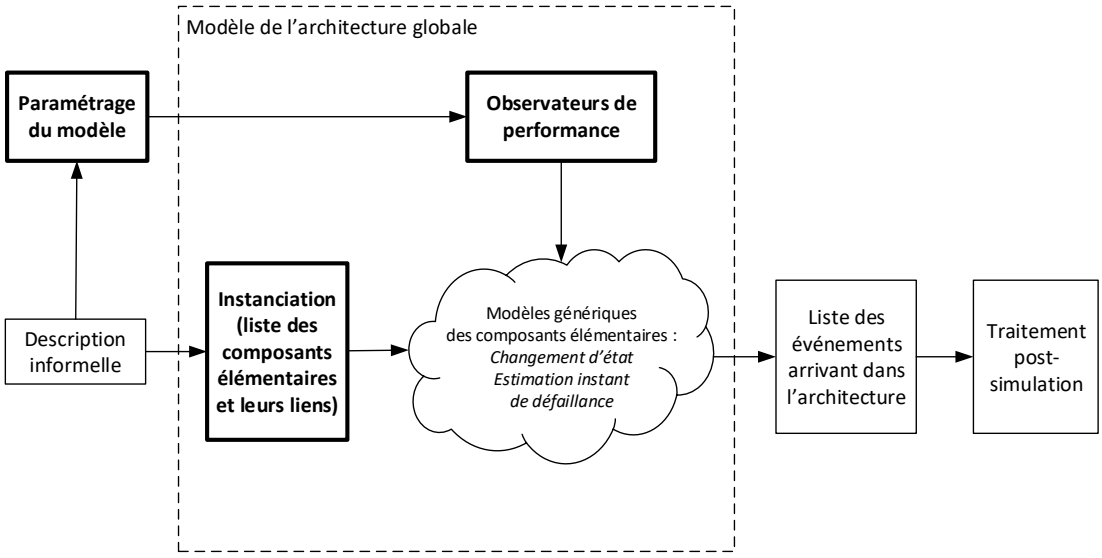



*Fig. 4. 13 : Entrées-sorties du modèle pour l'architecture globale*

Le modèle de l'architecture globale (Fig. 4. 13) se base principalement sur la structure qui est définie dans les diagrammes de classe vues dans le chapitre 2, dans la partie spécifications techniques d'une architecture de contrôle-commande.

Dans ce modèle, chaque transition de substitution correspond à un composant élémentaire possible dans une architecture de contrôle-commande, séparés entre les quatre grandes catégories évoquées précédemment (Fig. 4. 14).

Le modèle de l'architecture globale (Fig. 4. 13) est composé de trois éléments toujours réalisées grâce aux RdP. Les trois éléments sont l'instanciation de la configuration du système dans le modèle, le paramétrage du modèle pour respecter les critères de simulation que nous souhaitons mettre en place, et la mise en place des observateurs de performances de l'architecture pour l'analyse post-simulation. Ces trois éléments font l'objet des trois prochaines sous-sections.

#### *4.4.2.1. Instanciation du modèle*

Dans le modèle RdP de l'architecture (Fig. 4. 14), l'instanciation est réalisée par des fonctions qui seront exécutés après le franchissement d'une transition associée à l'initialisation des composants de l'architecture dans le modèle, selon leur catégorie et leur famille, et qui projetteront des jetons dans les places en aval pour chaque composant.

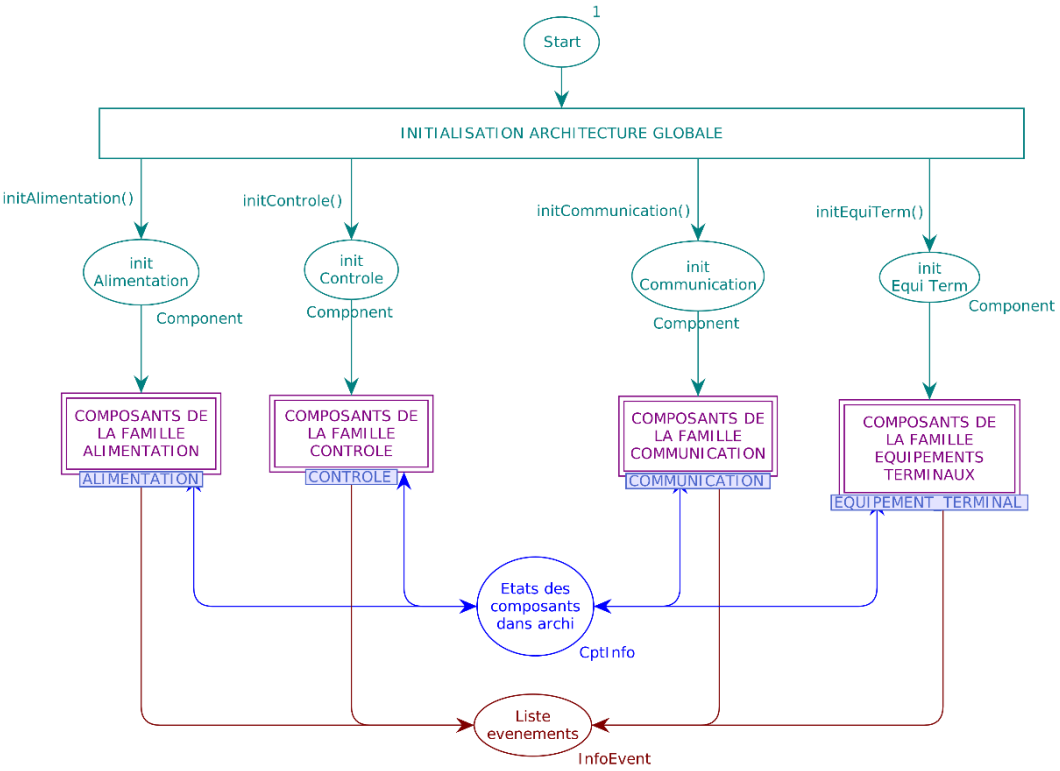



*Fig. 4. 14 : Modèle RdP de l'architecture globale*

L'instanciation permettra de définir le nombre de composants auxquels les transitions de substitutions devront faire appel. Ce type de transition n'étant lié qu'à un modèle unique de niveau hiérarchique inférieur, l'instanciation de n équipements d'un même type de composant élémentaire

*i* sera réalisé à l'aide de n jetons, chacun étant associé à une instance de composant. Le paramétrage consiste à associer, à chaque jeton coloré, l'ensemble des informations relatives aux paramètres de fonctionnement et de dysfonctionnement d'un équipement.

Le marquage initial du modèle d'architecture globale est constitué d'un jeton unique non coloré présent dans la place nommée "Start" et qui constitue le point d'entrée. L'instanciation et le paramétrage du réseau de Petri (RdP) correspondant à l'architecture, dont on souhaite évaluer les performances, repose sur :

- la définition d'un ensemble de couleurs caractérisant les composants élémentaires ainsi que leurs paramètres et les liens de dépendance,
- un ensemble de fonctions d'incidence arrière permettant, depuis le franchissement de la toute première transition située en aval de la place "Start" d'effectuer les traitements nécessaires à la génération des couleurs pour la création de jetons définissant les paramètres des composants de l'architecture de contrôle-commande.

Définition des fonctions d'incidence arrière

L'affectation des couleurs caractérisant l'architecture et ses composants élémentaires, ainsi que leurs paramètres, dépend de la topologie de l'architecture dont on souhaite évaluer les performances de sûreté de fonctionnement. Cette structure et les informations s'y afférant sont connues grâce à la description informelle importée depuis un outil dédié à l'élaboration d'offre commerciale (cet outil fera l'objet du chapitre dédié à l'application industrielle de la méthode d'évaluation). Ceci permet d'avoir les paramètres suivants :

- les variables $\delta_i$ caractérisant la déclaration des composants qui peuvent constituer l'architecture de contrôle-commande à simuler,
- le nombre d'instances de chaque composant de type *i* dans la famille j présents dans l'architecture, défini par la fonction $init_{(type\ Composant=i)}$ récupérant la liste entière des composants de type *i*,
- les valeurs des variables $\sigma_{ij}$ caractérisant la déclaration des liens de dépendance entre les composants de l'architecture. Elles sont fournies avec une liste ordonnée de jeux de paramètres $p_{ij} = (p_{ij1}, p_{ij2}, \ldots, p_{ijn})$ associés à chaque couple de composants liés entre eux.

Pour une variable $\delta_i$ décrivant un composant élémentaire, un ensemble de couleurs avec les attributs suivants a été défini :

- *Un identifiant unique* pour le composant $\lambda_{identifiant}$, parmi l'ensemble des naturels $\mathbb{N}$.
- *L'emplacement du composant* dans le système $\lambda_{emplacement}$, parmi un triplet d'identifiants caractérisant l'unité de contrôle, l'aire et le support d'alimentation où se trouve le composant.
- *La famille du composant* parmi les grandes familles répertoriées précédemment $\lambda_{famille}$, parmi une liste finie de familles possibles.

- *L'état du composant à l'instant t* (à l'initialisation, le composant est en marche) $\lambda_{état}$, parmi une liste finie d'états possibles pour un composant dans une architecture de contrôle-commande.
- *Les données fiabilistes du composant nécessaires pour calculer le prochain instant de défaillance, le prochain instant de réparation et le niveau de dégradation actuel* du composant $\lambda_{donnéesRAMS}$. Les valeurs sont la MTBF, le temps de réparation moyen MTTR et l'âge du composant, qui peuvent prendre respectivement leurs valeurs parmi l'ensemble des réels $\mathbb{R}$ pour la MTBF et la MTTR et sur l'intervalle [0,1] pour l'âge.

Ainsi, la variable $\delta_i$ découle de l'ensemble de couleurs $\Delta_i$ suivant :

$$\text{Colset } \Delta_i = record\ \lambda_{identifiant} : \Lambda_{identifiant} * \lambda_{emplacement} : \Lambda_{emplacement} * \lambda_{famille} : \Lambda_{famille} * \lambda_{état} : \Lambda_{état} * \lambda_{donnéesRAMS} : \Lambda_{donnéesRAMS}$$

La fonction $init_{(type\ Composant=i)}$ est définie de la manière suivante :

$$init_{typeComposant=i}() = n`(composants\ de\ type\ i) \quad (4.5)$$

où la fonction génère autant de jetons n qu'il y a de composants de type $i$.

Avec cette fonction, nous sommes en mesure de placer les composants élémentaires dans leurs modèles respectifs. Elle sera exécutée une seule fois au premier franchissement de la transition d'initialisation (Fig. 4. 15). La lecture de la variable $\delta_i$ caractérisant un composant et de son attribut $\lambda_{famille}$, pour la famille du composant, par la fonction $init_{(type\ Composant=i)}$ permettra de positionner les composants dans le bon modèle.

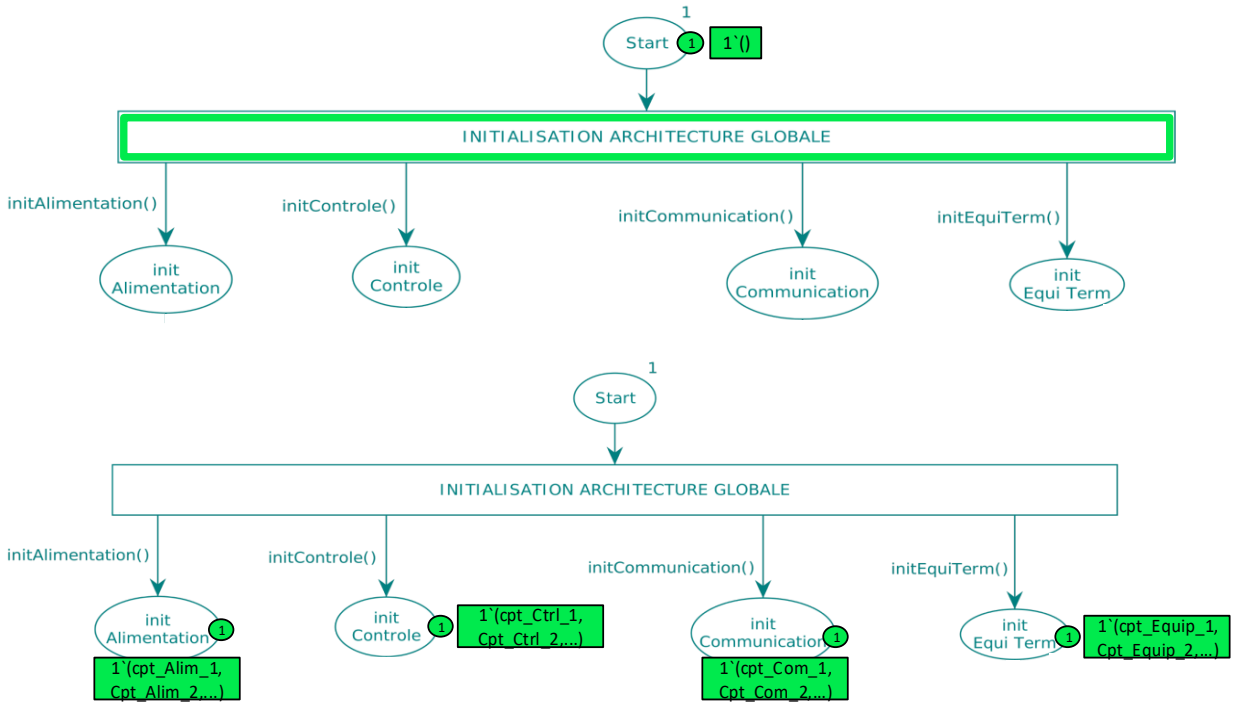



*Fig. 4. 15 : Première étape de l'instanciation du modèle : placement des composants à l'entrée du modèle correspondant à la famille du composant*

Pour les variables $\sigma_{ij}$ décrivant les liens de dépendance entre les composants élémentaires, et donc la structure de l'architecture globale, plusieurs ensembles de couleurs ont été définis, dépendant de la nature du lien. Néanmoins, ces couleurs suivent un pattern similaire en prenant les attributs généraux suivants :

- *Les identifiants uniques du couple de composants liés entre eux*. On définit ainsi avec un couple unique le lien entre un composant $C_1$ et un composant $C_2$.
- *La nature du lien*. Indiquer si c'est un lien de redondance, un lien entre deux composants safety pour la partie sécurité, ou un lien fonctionnel entre deux composants. Les liens physiques se référant à l'alimentation sont déjà fixés dans le modèle RdP des composants élémentaires, ces liens étant définis dans les diagrammes UML cités dans le chapitre 2.

Une fois le composant placé dans le modèle correspondant à sa famille, l'estimation du prochain instant de défaillance intrinsèque de chaque composant, grâce à l'attribut $\lambda_{\text{donnéesRAMS}}$ decrivant les données fiabilistes peut être réalisée. Ceci se fait grâce à la lecture des valeurs dans le champ de l'attribut $\lambda_{\text{donnéesRAMS}}$, à partir d'une fonction SML qui va calculer cet instant. Cette fonction reprend les calculs détaillés dans la section 4.4.1.3. Pour un composant *i* de la famille j, elle aura la forme suivante :

$$setNextFailureTime\left(composant_{ij}\right) = F\left(MTBF\ composant_{ij}\ , Age\ composant_{ij}\right) \qquad (4.6)$$

avec :

- F est une fonction ML à partir des calculs d'estimation expliquée dans la section 4.4.1.3
- MTBF une variable récupérée dans les données fiabilistes du composant *i* dans la famille j.
- Age, représentant le niveau de dégradation du composant, étant une variable récupérée dans les données fiabilistes du composant *i* dans la famille j.

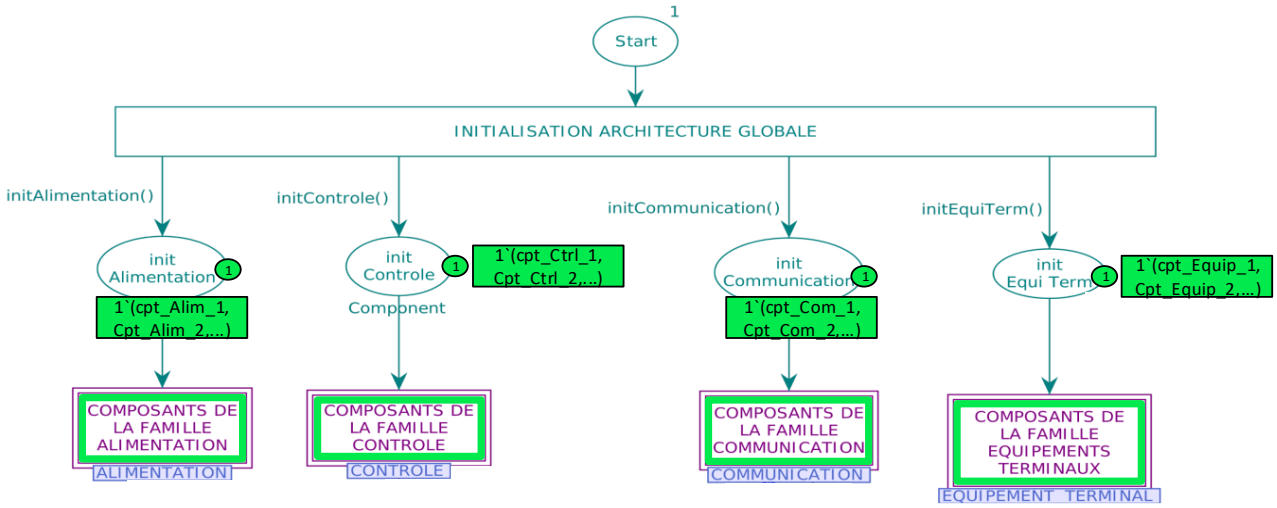



*Fig. 4. 16 : Deuxième étape de l'instanciation du modèle : mise en place des composants dans le modèle de leur famille, en état de marche avec leur prochain instant de défaillance*

Ce temps estimé sera pris en compte dans la temporisation du jeton coloré en charge de la gestion des défaillances du composant, après le franchissement de la transition qui assigne les

composants dans le modèle de leur famille (Fig. 4. 16). Ce jeton est en complément du jeton coloré qui sera en charge de gérer l'état du composant.

#### *4.4.2.2. Paramétrage du modèle*

En addition de l'instanciation de la structure de l'architecture et des différents composants, nous définissons les paramètres de la simulation de Monte Carlo que nous allons mettre en place au niveau du modèle d'architecture. Ce paramétrage se fait en parallèle de la définition de la description informelle de l'architecture de contrôle-commande. Il s'agit des paramètres suivants :

- Le nombre de fois qu'on va simuler le comportement de l'architecture dans le temps, à partir du modèle RdP. C'est le nombre d'histoires à envisager.
- La durée de simulation d'une histoire, c'est-à-dire sur combien d'années nous souhaitons simuler le système.
- La température des différentes parties du système.
- Le niveau de mission attendu pour les composants dans une partie de l'architecture. Si les composants doivent fonctionner en continu sans interruption ou par intermittence (c'est-à-dire sollicité une partie de la journée, 8h par exemple) pour faire tourner le procédé industriel.
- Le temps de réparation moyen MTTR de chaque composant.

Ces paramètres permettent de préciser le contexte d'utilisation de l'architecture. La température et le niveau de mission des composants permettent de sélectionner les données fiabilistes corrects (MTBF et niveau de dégradation) parmi une liste de valeurs possibles, suivant les conditions d'utilisation du composant.

À l'issue de ce paramétrage et de l'instanciation du système à simuler dans le modèle d'architecture globale, nous serons en mesure de créer les observateurs de performance de ce système.

#### *4.4.2.3. Observateurs des performances de SdF*

Nous savons dès le départ quelles performances nous souhaitons récupérer et les critères de simulation que nous souhaitons performer. A partir de la description de ces performances qui doivent être évaluées et de ces critères, nous pouvons mettre en place des observateurs qui seront générés automatiquement au moment de la simulation. Au cours de celle-ci, le réseau peut contenir et générer nombre d'informations quantitatives sur la performance et l'évolution d'un système, telles que les changements d'état. Ces informations peuvent être extraites du modèle en examinant les marquages des places ou les tirs des transitions puis enregistrées à l'aide de moniteurs. Leur insertion est définie dans le modèle de l'architecture, aux emplacements pertinents du modèle. Dans le formalisme défini par (Jensen, Kristensen, & Wells, 2007), ces observateurs de performance sont appelés « moniteurs ». Ils se basent sur deux principes :

- La définition d'une structure générique de moniteurs pour chaque famille de performances et leur intégration sur des places et transitions dédiées dans les modèles d'équipements ou de l'architecture globale,

- Le développement d'un algorithme permettant de traiter le relevé automatique des valeurs statistiques observées lors de la simulation et de leur traitement en fonction des performances à évaluer.

Un moniteur permet de collecter des données brutes relatives aux événements d'intérêt considérés lors de la simulation. Il constitue un mécanisme utilisé pour observer, inspecter, ou contrôler une simulation d'un réseau de Petri coloré et temporisé. De nombreux moniteurs différents peuvent être définis pour un réseau de Petri donné.

Les moniteurs peuvent inspecter à la fois les marquages des places et les tirs de transitions lors d'une simulation. La collecte des informations peut s'effectuer sur occurrence d'un événement particulier (tir d'une transition par exemple). Sur la base des informations collectées, les moniteurs peuvent également effectuer des traitements ayant un intérêt vis à vis de la performance à évaluer.

Par exemple, un collecteur de données qui mesure la longueur d'une file d'attente de paquets peut calculer à la fois la longueur moyenne de la file d'attente ainsi que la longueur maximale de la file d'attente.

Parmi les traitements réalisables par un moniteur, nous pouvons citer, à titre d'exemple :

- Arrêter une simulation sous certaines conditions,
- Compter le nombre de fois qu'une transition se produit,
- Mettre à jour un fichier lors du franchissement d'une transition avec une variable liée à une valeur spécifique à observer,
- Calculer le nombre moyen de jetons dans une place.

Dans le logiciel CPN Tools utilisés pour le déploiement de nos modèles RdP, des moniteurs natifs dont les fonctions ont été prédéfinies, sont déjà implémentés :

- **Breakpoint monitor** : sont utilisés pour arrêter une simulation,
- **Marking size** : enregistre le marquage d'une place,
- **Count transition** : compte le nombre de tir d'une transition,
- **List length** : enregistre la taille d'une liste lorsque le jeton fait appel à un ensemble de couleurs défini par une liste,
- **Generic data collector** : traitement spécifique défini par le modélisateur.

Si les moniteurs prédéfinis ne suffisent pas, il est possible d'utiliser des moniteurs définis par l'utilisateur dont les conditions dépendent des besoins de l'utilisateur :

**- User define monitor** qui permet de définir des fonctions dédiées aux post-traitements comme par exemple l'enregistrement personnalisé des observations dans des fichiers. L'utilisateur définit lui-même les conditions de création du fichier de relevé (son nom, son format l'extension), le prédicat pour déclencher un relevé statistique, les données à relever et à intégrer dans le fichier et l'action à faire sur le fichier à la fin de la simulation. Il peut être utilisé pour des applications autres que l'évaluation de performances, par exemple pour faire communiquer CPNTools avec un autre logiciel en temps réel pendant la simulation.

Dans tous les cas, un moniteur aura ce mode de fonctionnement propre à tous les types possibles (Ndiaye, 2017):

*Algorithme de mise en œuvre d'un moniteur*

**Répéter jusque** fin de simulation
    Vérifier si les conditions décrites dans le prédicat du moniteur sont remplies
    **Si les conditions sont remplies**
        **Alors**
            Effectuer un ou plusieurs traitements à partir des informations observées (informations contenues dans la marquage et/ou les éléments de liaison)
            Effectuer un enregistrement
        **Sinon** ne rien faire
    **Fin_Si**
**Fin_Répéter**

Dans le cadre de cette étude, les types de moniteurs utilisés sont un Breakpoint pour arrêter la simulation à la durée choisie par l'utilisateur, et un User define monitor. Ce moniteur User define monitor est défini par un 5-uplet de fonctions (init, predicat, observer, action, stop) où :

- **Init**, est la fonction permettant l'exécution d'une action au premier événement sur le moniteur.
- **Predicat** est une fonction qui fixe les conditions dans lesquelles une observation doit être réalisée (ce qui conditionne notamment les instants où la valeur surveillée doit être relevée),
- **Observer** est une fonction qui permet d'effectuer la relevée de la valeur désirée,
- **Action** est une fonction permettant d'assigner une valeur observée vers un objet, un fichier de simulation,
- **Stop** est la fonction définissant l'action à réaliser à la fin de la simulation.

Les données relevées par le moniteur sont ensuite utilisées pour calculer des statistiques et produire les différents indicateurs de performances.

Le moniteur que nous allons utiliser ici va permettre d'enregistrer tous les événements survenus sur l'architecture pendant la durée de simulation, et ceci pour chaque histoire. Ces événements vous nous permettre de calculer par la suite les indicateurs de sûreté de fonctionnement qui nous intéressent et que nous verrons dans les sections suivantes. Nous aurons alors une trace du comportement du système à travers ces données, sauvegardées sous la forme d'un fichier qui sera traité en post-simulation.

Le moniteur en question sera associé à une transition où les jetons colorés relatés aux notifications survenant à la suite d'un événement au niveau de l'architecture vont être consommés. Ce moniteur aura la structure suivante :

- **Init : création du fichier moniteur** (Création du fichier pour les relevés statistiques du moniteur lors du démarrage de la simulation (action réalisée une seule et unique fois). Puis ouverture de celui-ci pour l'écriture des relevés.)

- **Predicat** : **conditions pour activer le moniteur** (Une conditions possible : l'instant est inférieur au temps de simulation défini par l'utilisateur.)
- **Observer** : **récupération des données à écrire dans le fichier moniteur** (Récupération des valeurs caractérisant l'événement arrivé au moment où le moniteur est activé)
- **Action : écriture de ces données dans le fichier moniteur** (Écriture des valeurs citées dans l'Observer)
- **Stop : fermeture du fichier moniteur** (Fermeture du fichier de moniteur lorsque le moniteur break point survient marquant la fin de la simulation)

## 4.5. Simulation et évaluation des performances de SdF

### 4.5.1. Mise en place de la simulation

L'évaluation des performances de l'architecture se base sur la simulation de Monte Carlo et sur l'observation du comportement du système au cours du temps. Le mécanisme se fait en trois temps (Fig. 4. 17) :

- La mise en place des observateurs de performances, qui vont nous permettre de récupérer les informations essentielles pour estimer les indicateurs fiabilistes voulus,
- Le lancement de la simulation de Monte Carlo, en respectant les paramètres de simulation voulus,
- Le traitement des données à l'issue de la simulation, à partir des observateurs, pour la récupération des performances.

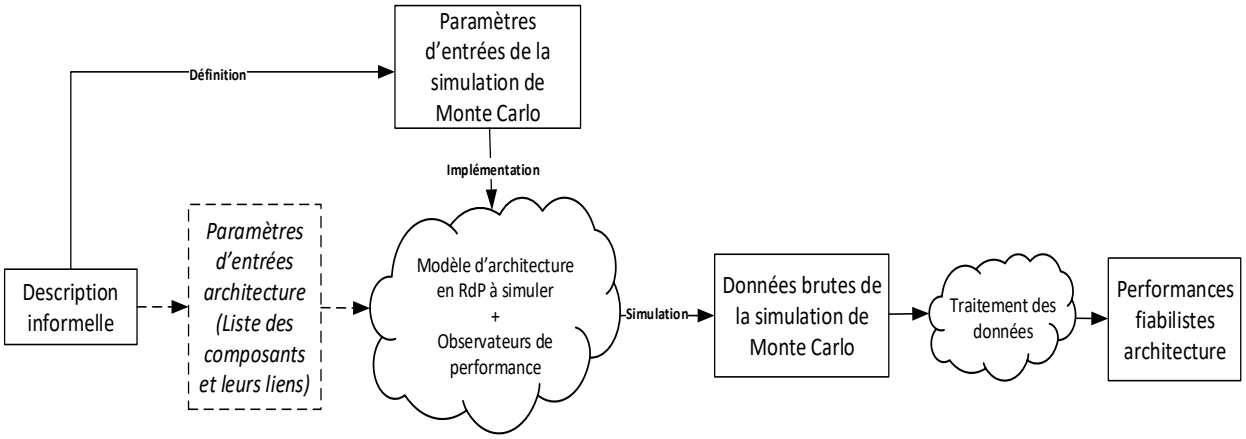



*Fig. 4. 17 : Vue d'ensemble de la simulation et de l'évaluation des performances de l'architecture à partir de son modèle RdP*

Cette évaluation a été construite en parallèle de la construction du modèle d'architecture. En plus de la définition de la structure du système, avec les modèles de chaque famille et les liens pour le système global, nous définissons les observateurs de performance et les paramètres de simulation de Monte Carlo.

Enfin, l'intervalle de confiance souhaité par les ingénieurs en charge de l'élaboration des offres d'architectures sera pris en compte pour déterminer les paramètres d'arrêt de la simulation et le nombre d'histoires à simuler.

Par ailleurs, les simulations de Monte-Carlo sont basées sur la simulation d'un ensemble d'histoires, chacune d'entre elles ayant une durée bornée à l'aide d'un critère d'arrêt équivalent à la durée de simulation donné en entrée du modèle. Compte tenu du fait que les histoires sont indépendantes et identiquement distribuées, les résultats obtenus pour chacune d'entre elles seront différents.

### 4.5.2. Évaluation des performances de SdF de l'architecture

Le moniteur développé pour la simulation de Monte Carlo sur le modèle d'évaluation permet de créer un fichier pour chaque histoire répertoriant tous les événements arrivés sur l'architecture de contrôle-commande. Cette liste d'événements, pour qu'elle soit exploitable à l'évaluation des performances fiabilistes du système, se présente avec les attributs suivant pour chaque événement :

- L'instant de l'événement.
- La localisation de l'événement, à quel niveau de l'architecture il est arrivé.
- Le composant source de l'événement, celui qui est défaillant ou réparé ou mis en hors tension. Son identifiant unique et sa famille sont donnés, ainsi que son état à la suite de cet événement.
- Un booléen indiquant si l'événement a un impact important sur la dégradation d'une aire de l'architecture de contrôle-commande. Il est à *vrai* si l'événement dégrade la mission de l'emplacement et à *faux* si ce n'est pas le cas.

À partir de cette liste, nous allons pouvoir estimer les performances de sûreté de fonctionnement demandées par le client ou par le système intégrateur (listées dans le chapitre 1). L'algorithme post-simulation permettant, à partir des données brutes de la simulation sous la forme de la liste d'événements, de calculer ces performances est détaillé juste après. Ici, l'algorithme est divisé par indicateur fiabiliste, afin de visualiser les informations nécessaires pour chaque indicateur.

#### *4.5.2.1. Disponibilité*

La disponibilité est un indicateur permettant d'observer à un instant donné t, la capacité du système à être opérationnel pour réaliser dans ses conditions nominales les missions qui lui sont assignées. Ceci implique que pour que le système soit disponible à un instant t, il faut en principe que tous les composants élémentaires soient fonctionnels à cet instant. Pour des composants en redondance, il faut qu'au moins un des composants soit fonctionnel à t.

Dans le cadre d'une simulation de Monte Carlo et de son approche statistique, estimer cette probabilité du système d'être opérationnel à l'instant t peut se traduire par la formule suivante :

$$A_{moy}(t) = 1 - \frac{n_{def}^{sys}(t)}{n_0} \qquad (4.7)$$

avec :

- $n_0$ le nombre total d'histoires réalisées pour la simulation de Monte Carlo
- $n_{\text{def}}^{\text{sys}}(t)$ le nombre d'histoires où le système est défaillant (non disponible) à l'instant t
- $A_{moy}(t)$ la disponibilité du système à l'instant t

La valeur obtenue $A_{moy}(t)$ est une moyenne empirique de la disponibilité du système. Pour compléter cette valeur et avoir plus de précision sur le résultat, la variance et l'écart type permettent de s'assurer que la valeur estimée est proche de la valeur moyenne réelle. Ces deux valeurs permettront d'établir un intervalle de confiance dont les formules ont été vues dans la section 3.4.2.

Pour rappel, si $n_0$ est le nombre d'histoires et $\bar{x}$ la moyenne empirique de l'indicateur x calculée, alors :

$$\bar{x} = \frac{\sum_{i=1}^{n_0} x_i}{n_0} \tag{4.8}$$

avec $x_i$ la valeur de l'indicateur dans l'histoire *i*

L'estimation de l'écart type $s_x$, et de la variance $s_x^2$ est donnée par :

$$s_x^2 = \frac{\sum_{i=1}^{n_0}(x_i - \bar{x})^2}{n_0 - 1} \tag{4.9}$$

$$s_x = \sqrt{s_x^2} \tag{4.10}$$

L'intervalle de confiance (fixé à 99%) est donné par :

$$\left[\bar{x} - t_\alpha \frac{s_x}{\sqrt{n_0}}, \bar{x} + t_\alpha \frac{s_x}{\sqrt{n_0}}\right] \tag{4.11}$$

où $t_\alpha$ est le paramètre associé au degré de confiance $\alpha$=0,01 pour un intervalle de confiance à 99%.

Les trois équations précédentes ainsi que l'intervalle de confiance sont valables pour tous les indicateurs de SdF calculés ici.

En termes d'algorithme, un algorithme post-simulation est mis en place, pour estimer la disponibilité de l'architecture de contrôle-commande, ou d'une de ses zones :

---

*Algorithme comptant le nombre de défaillances (pour la disponibilité)*

---

T = durée de simulation d'une histoire
dt = pas de temps pour estimer la défaillance *(1 h)*
$n_0$ = nombre d'histoires réalisées pour la simulation de Monte Carlo
$n_{def}$ = vecteur de longueur $\frac{T}{dt} + 1$ comptabilisant le nombre d'histoires où le système est défaillant à un instant t (correspondant à l'indice dans le vecteur)
$\text{ListeEvent}_i$ = liste des événements dans l'histoire *i*

$t_{actuel}$ = pas de temps de l'algorithme
$t_{evenement}$ = instant d'occurrence de l'événement qui est lu dans le fichier de simulation
$t_{réparation}$ = instant de réparation correspondant à l'événement Défaillance lu dans le fichier de simulation
**Pour $n_i$ allant de 1 à $n_0$** *(pour chaque histoire)*
Lire le fichier de données de simulation de $n_i$ avec la liste des événements étant arrivés sur le système $ListeEvent_i$
**Pour $t_{actuel}$ allant de 0 à T par pas dt**
**Si $t_{actuel}$ = $t_{evenement}$ ET evenement = Défaillance ET le système est dégradé**
**Alors**
Lire $ListeEvent_i$ jusqu'à ce que evenement = Réparation et déterminer $t_{reparation}$
$n_{def}$ (de $t_{evenement}$ à $t_{reparation}$) = $n_{def}$ (de $t_{evenement}$ à $t_{reparation}$) +1
**Sinon** ne rien faire
**Fin_Si**
**Fin_Pour**
**Fin_Pour**
⇨ Obtention d'un vecteur comptant le nombre d'histoires avec une défaillance à t $n_{def}$

---

À la fin de l'exécution de l'algorithme, une fois que le vecteur contenant le nombre d'histoires avec une défaillance à t est calculé, nous obtenons la disponibilité à chaque pas de temps, l'écart-type, la variance et l'intervalle de confiance en utilisant les équations de (4.7) à (4.11).

#### *4.5.2.2. Fiabilité*

La fiabilité est un indicateur permettant d'observer à un instant donné t, la probabilité du système de ne pas avoir eu de défaillances sur ses composants qui l'empêche d'assurer sa mission, depuis le début de son opération jusqu'à t. Ceci implique qu'il n'y ait pas eu préalablement de défaillances critiques sur les équipements jusqu'à l'instant t.

Ainsi, pour estimer la fiabilité de l'architecture de contrôle-commande, ou d'une de ses zones, nous réalisons l'algorithme suivant.

Dans le cadre d'une simulation de Monte Carlo et de son approche statistique, estimer cette probabilité du système d'avoir été opérationnel jusqu'à l'instant t peut se traduire par la formule suivante :

$$R_{moy}(t) = 1 - \frac{n_{1def}^{sys}(t)}{n_0} \qquad (4.12)$$

avec :

- $n_0$ le nombre total d'histoires réalisées pour la simulation de Monte Carlo
- $n_{1\text{def}}^{\text{sys}}(t)$ le nombre d'histoires où le système a eu sa première défaillance avant l'instant t (où le système n'a pas été opérationnel jusqu'à t)
- $R_{moy}(t)$ la fiabilité du système à l'instant t (moyenne empirique)

Le calcul de la MTTF, le temps moyen de fonctionnement avant la première défaillance, est obtenu en faisant la moyenne des instants de première défaillance sur toutes les histoires.

$$MTTF = \frac{\sum_{i=1}^{n_0} t_{1def}(i)}{n_0} \tag{4.13}$$

En termes d'algorithme, pour estimer la fiabilité de l'architecture de contrôle-commande, ou d'une de ses aires, nous réalisons les étapes suivantes.

---

*Algorithme comptant le nombre de premières défaillances (pour la fiabilité)*

---

T = durée de simulation d'une histoire
dt = pas de temps pour estimer la défaillance *(1 h)*
$n_0$ = nombre d'histoires réalisées pour la simulation de Monte Carlo
$n_{1def}$ = vecteur de longueur $\frac{T}{dt} + 1$ comptabilisant le nombre d'histoires où le système a eu une défaillance avant l'instant t (correspondant à l'indice dans le vecteur)
$t_{1def}$ = vecteur de longueur $n_0$ sauvegardant l'instant de la première défaillance du système dans l'histoire *i* (correspondant à l'indice dans le vecteur)
aucuneDefaillance = booléen initialisé à "vrai" et indiquant si l'événement est la première défaillance du système
$ListeEvent_i$ = liste des événements dans l'histoire *i*
$t_{actuel}$ = pas de temps de l'algorithme
$t_{evenement}$ = instant d'occurrence de l'événement qui est lu dans le fichier de simulation
**Pour $n_i$ allant de 1 à $n_0$** *(pour chaque histoire)*
    Lire le fichier de données de simulation de $n_i$ avec la liste des événements étant arrivés sur le système $ListeEvent_i$
    premiereDefaillance = "vrai"
    **Pour $t_{actuel}$ allant de 0 à T par pas dt**
        **Si $t_{actuel}$ = $t_{evenement}$ ET evenement = Défaillance ET le système est dégradé ET aucuneDefaillance = "vrai"**
            **Alors**
                $n_{1def}$ (de $t_{evenement}$ à T) = $n_{def}$ (de $t_{evenement}$ à T) +1
                $t_{1def}(n_i)$ = $t_{evenement}$
                aucuneDefaillance = "faux"
            **Sinon** ne rien faire
        **Fin_Si**
    **Fin_Pour**
**Fin_Pour**
⇨ Obtention d'un vecteur comptant le nombre d'histoires avec une première défaillance avant t $n_{1def}$

---

À la fin de l'exécution de l'algorithme, une fois que le vecteur contenant le nombre d'histoires avec une première défaillance à t est calculé, nous obtenons la fiabilité à chaque pas de temps, l'écart-type, la variance et l'intervalle de confiance en utilisant les équations de (4.8) à (4.13).

#### *4.5.2.3. Nombre de défaillances*

Cet indicateur donne le nombre de défaillances de l'architecture, ou de la zone, pendant toute la durée de simulation, correspondant généralement à la durée d'exploitation du système. Il permettra de déterminer le nombre de pièces de rechange minimum nécessaire pour ne pas mettre à l'arrêt le système. Ceci implique de compter toutes les défaillances jusqu'à l'instant de fin de simulation.

En termes d'équation, ceci reviendrait à formaliser cet indicateur par la formule suivante :

$$\overline{N}(T) = \frac{\Sigma_{i=1}^{n_0} N_i(T)}{n_0} \tag{4.14}$$

avec :

- $n_0$ le nombre total d'histoires réalisées pour la simulation de Monte Carlo
- T la durée de simulation
- $N_i(T)$ le nombre de défaillances du système dans une histoire *i* sur T
- $\overline{N}(T)$ le nombre de défaillances du système sur T (moyenne empirique)

Ainsi, pour estimer le nombre de défaillances de l'architecture de contrôle-commande, ou d'une de ses zones, nous réalisons l'algorithme suivant.

---

*Algorithme d'évaluation du nombre de défaillances*

---

T = durée de simulation d'une histoire
$n_0$ = nombre d'histoires réalisées pour la simulation de Monte Carlo
$n_{def}$ = vecteur de longueur $n_0$ comptabilisant le nombre de défaillances du système dans l'histoire *i* (correspondant à l'indice dans le vecteur)
$ListeEvent_i$ = liste des événements dans l'histoire *i*
**Pour $n_i$ allant de 1 à $n_0$** *(pour chaque histoire)*
    Lire le fichier de données de simulation de $n_i$ avec la liste des événements étant arrivés sur le système $ListeEvent_i$
    **Si evenement = Défaillance**
        **Alors**
            $n_{def}(i) = n_{def}(i) + 1$
        **Sinon** ne rien faire
    **Fin_Si**
**Fin_Pour**
⇨ Obtention d'un vecteur comptant le nombre de défaillances pour chaque histoire *i* $n_{def}$

---

À la fin de l'exécution de l'algorithme, une fois que le vecteur contenant le nombre de défaillances pour chaque histoire est calculé, nous obtenons la moyenne en utilisant l'équation (4.14).

#### *4.5.2.4. Temps d'indisponibilité*

Cet indicateur donne la durée de temps pendant laquelle l'architecture, ou la zone, est indisponible et ne peut réaliser son rôle de faire fonctionner un procédé industriel et ce, pendant toute la durée de simulation. Ceci implique de compter toutes les défaillances jusqu'à l'instant de fin de simulation et de durée cumulée des pannes avant que la réparation ne soit terminée.

En termes d'équation, ceci reviendrait à formaliser cet indicateur par la formule suivante :

$$\overline{T_{indisponible}}(T) = \frac{\sum_{i=1}^{n_0} T_{indispo_i}(T)}{n_0} \quad (4.15)$$

avec :

- $n_0$ le nombre total d'histoires réalisées pour la simulation de Monte Carlo
- T la durée de simulation
- $T_{indispo\_i}(T)$ le temps d'indisponibilité du système dans une histoire *i* sur T
- $\overline{T_{indisponible}}(T)$ le temps d'indisponibilité du système sur T (moyenne empirique)

Ainsi, pour estimer le temps d'indisponibilité de l'architecture de contrôle-commande, ou une de ses zones, nous réalisons l'algorithme suivant.

---

*Algorithme d'évaluation du temps d'indisponibilité*

---

T = durée de simulation d'une histoire
$n_0$ = nombre d'histoires réalisées pour la simulation de Monte Carlo
$t_{def}$ = vecteur de longueur $n_0$ comptabilisant le temps d'indisponibilité du système dans l'histoire *i* (correspondant à l'indice dans le vecteur)

**Pour $n_i$ allant de 1 à $n_0$** *(pour chaque histoire)*
Lire le fichier de données de simulation de $n_i$ avec la liste des événements étant arrivé sur le système $ListeEvent_i$
**Si evenement = Défaillance ET le système est dégradé**
**Alors**
Sauvegarder $t_{evenement}$
Lire $ListeEvent_i$ jusqu'à ce que evenement = Réparation et déterminer $t_{reparation}$
$t_{def}(i) = t_{def}(i) + (t_{reparation} - t_{evenement})$
**Sinon** ne rien faire
**Fin_Si**
**Fin_Pour**

⇨ Obtention d'un vecteur comptant le temps d'indisponibilité pour chaque histoire *i* $t_{def}$

---

À la fin de l'exécution de l'algorithme, une fois que le vecteur contenant le temps d'indisponibilité pour chaque histoire est calculé, nous obtenons la moyenne en utilisant l'équation (4.15).

### 4.5.3. Estimation des facteurs d'importance

Au-delà des performances classiques de sûreté de fonctionnement estimées dans la section précédente, des critères basés sur des facteurs d'importance jouent un rôle également dans l'interprétation des résultats. Ils permettent d'approfondir l'analyse du comportement dysfonctionnel de l'architecture de contrôle-commande. Dans ce sens, ils apportent un complément d'informations qui doit aider le système intégrateur à prendre des décisions pour proposer la meilleure architecture.

D'un point de vue statistique, chaque histoire dans une simulation de Monte Carlo présente un scénario différent : les événements aléatoires qui surviennent sur le système ne sont pas les mêmes entre deux histoires. Aussi, l'objectif est d'identifier quel est le composant le plus important dans un système de contrôle-commande, à tout instant de son fonctionnement, en observant statistiquement l'impact qu'il a eu sur le système.

Le but est de comparer ces facteurs d'importance et d'identifier les composant les plus critiques, c'est-à-dire le composant dont la défaillance a le plus d'impact sur la probabilité de l'architecture à être en mode dégradé, par rapport à la situation normale. Estimer l'importance dynamique d'un groupe de composants apportera également quelques informations.

#### *4.5.3.1. Facteur de Birnbaum et d'importance critique*

Cet indicateur donne l'impact de chaque composant sur la dégradation de l'architecture, ou l'aire, pendant toute la durée de simulation. Ceci implique de compter toutes les défaillances pour chaque composant de l'architecture, ou de l'aire, jusqu'à l'instant de fin de simulation.

Il est calculé à partir du facteur de Birnbaum et du facteur d'importance critique, appelé *Criticality Importance* en anglais, que nous avons introduit dans le chapitre 3. Comme il a été dit dans le chapitre précédent, le facteur de Birnbaum représente la variation de la fiabilité du système en fonction de la fiabilité du composant *i*. Plus il est élevé, plus le composant a un impact sur la fiabilité du système. Pour rappel, sa définition mathématique est :

$$I^B(i|système\ défaillant\ \text{à}\ t) = R_{systeme}(1_i, R(t)) - R_{systeme}(0_i, R(t)) \qquad (4.16)$$

avec :

- $R_{systeme}(1_i, R(t))$ la fiabilité du système à l'instant t en supposant que le composant *i* fonctionne
- $R_{systeme}(0_i, R(t))$ la fiabilité du système à l'instant t en supposant que le composant *i* ne fonctionne pas

Le facteur d'importance critique découle du facteur de Birnbaum et représente la probabilité qu'un composant soit défaillant et critique pour le système sachant que le système global est défaillant :

$$I^{CR}(i|système\ défaillant\ \text{à}\ t) = \frac{I^B(i|système\ défaillant\ \text{à}\ t)[1 - R_i(t)]}{1 - R_{systeme}(t)} \qquad (4.17)$$

avec :

- $R_{systeme}(t)$ la fiabilité du système à l'instant t
- $R_i(t)$ la fiabilité du composant *i* à l'instant t
- $I^B(i|système\ défaillant\ \text{à}\ t)$ le facteur Birnbaum du composant *i* sachant que le système est défaillant à l'instant

Dans le cadre d'une simulation de Monte Carlo et de son approche statistique, estimer ces facteurs sur toute la durée de simulation pour chaque composant dans le système peut être effectué en se basant sur les nombres différents d'histoires concernées par une défaillance sur le système ou sur le composant *i*. Nous allons détailler ici comment les deux facteurs sont calculés.

Concernant le facteur de Birnbaum, en reprenant l'équation (4.16), nous devons calculer les deux types de fiabilité pris en compte dans l'expression.

La fiabilité du système à l'instant t en supposant que le composant *i* fonctionne est :

$$R_{systeme}\left(1_i, R(t)\right) = 1 - \frac{n_{1def}^{sys}(t) - n_{1def}^{sys}(t|i)}{n_0 - n_{1def}^{sys}(t|i)} \qquad (4.18)$$

avec :

- $n_{1def}^{sys}(t)$ le nombre d'histoires où le système a eu sa première défaillance avant l'instant t
- $n_{1def}^{sys}(t|i)$ le nombre d'histoires où le système a eu sa première défaillance avant l'instant t avec le composant *i* qui est la source de la défaillance
- $n_0$ le nombre total d'histoires réalisées pour la simulation de Monte Carlo

La fiabilité du système à l'instant t en supposant que le composant *i* ne fonctionne pas est :

$$R_{systeme}\left(0_i . R(t)\right) = 1 - \frac{n_{1def}^{sys}(t|t_{0i} < t)}{n(t|t_{0i} < t)} \qquad (4.19)$$

avec :

- $t_{0i}$ l'instant où le composant *i* a eu sa première défaillance
- $n_{1def}^{sys}(t|t_{0i} < t)$ le nombre d'histoires où le système a eu sa première défaillance avant à l'instant t avec le composant *i* qui a été défaillant avant t
- $n(t|t_{0i} < t)$ le nombre d'histoires où le composant *i* a été défaillant avant l'instant t, que le système soit défaillant ou non

Ainsi, le facteur de Birnbaum peut être donnée par la formule suivante :

$$I^B(i|système\ défaillant\ \text{à}\ t) = 1 - \frac{n_{1def}^{sys}(t) - n_{1def}^{sys}(t|i)}{n_0 - n_{1def}^{sys}(t|i)} - \left[1 - \frac{n_{1def}^{sys}(t|t_{0i} < t)}{n(t|t_{0i} < t)}\right] \qquad (4.20)$$

Selon l'expression du facteur d'importance critique donnée par équation (4.17)., il peut être obtenu à partir de l'expression du facteur de Birnbaum donnée dans l'équation (4.20) en considérant également deux autres fiabilités données ci-dessous.

La fiabilité du système à l'instant t reprend l'expression de l'équation (4.12), à savoir :

$$R_{systeme}(t) = 1 - \frac{n_{1def}^{sys}(t)}{n_0} \tag{4.21}$$

avec :

- $n_{1def}^{sys}(t)$ le nombre d'histoires où le système a eu sa première défaillance avant l'instant t
- $n_0$ le nombre total d'histoires réalisées pour la simulation de Monte Carlo

La fiabilité du composant *i* à l'instant t est :

$$R_i(t) = 1 - \frac{n_{1def}^{i}(t)}{n_0} \tag{4.22}$$

avec :

- $n_{1def}^{i}(t)$ le nombre d'histoires où le composant *i* a eu sa première défaillance avant l'instant t
- $n_0$ le nombre total d'histoires réalisées pour la simulation de Monte Carlo

Ainsi, le facteur d'importance critique peut être donnée par la formule suivante :

$$I^{CR}(i|système\ défaillant\ \text{à}\ t) = \frac{\left[1 - \frac{n_{1def}^{sys}(t) - n_{1def}^{sys}(t|i)}{n_0 - n_{1def}^{sys}(t|i)} - \left[1 - \frac{n_{1def}^{sys}(t|t_{0i} < t)}{n(t|t_{0i} < t)}\right]\right] \times n_{1def}^{i}(t)}{n_{1def}^{sys}(t)} \tag{4.23}$$

*Exemple :*

Considérons un système simple constitué de trois composants suivant une loi exponentielle avec un taux de défaillance respectif de $\lambda_1$, $\lambda_2$, et $\lambda_3$, avec un composant en série et les deux autres en parallèle comme dans la Fig. 4. 18.

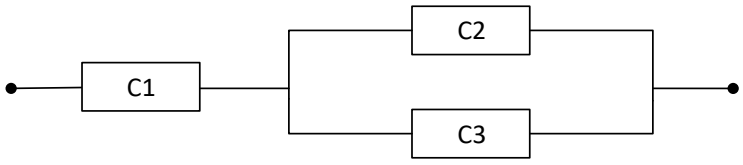



*Fig. 4. 18 : Exemple de système*

**Valeurs numériques : $\boldsymbol{\lambda_1 = 10^{-7}\ h^{-1}}$ ; $\boldsymbol{\lambda_2 = 2.10^{-5}\ h^{-1}}$ ; $\boldsymbol{\lambda_3 = 10^{-5}\ h^{-1}}$**

Calculons dans un premier temps, les valeurs théoriques de chaque facteur d'importance, Birnbaum et critique, pour chaque composant. Nous fixons l'instant à t=10 ans.

Les fiabilités des composants et du système sont :

- $R_1(t) = e^{-\lambda_1 t}$ ce qui donne $R_1(t = 10\ ans) = e^{-10^{-7} \times 10 \times 8760} \approx 0{,}991$
- $R_2(t) = e^{-\lambda_2 t}$ ce qui donne $R_2(t = 10\ ans) = e^{-2.10^{-5} \times 10 \times 8760} \approx 0{,}173$
- $R_3(t) = e^{-\lambda_3 t}$ ce qui donne $R_3(t = 10\ ans) = e^{-10^{-5} \times 10 \times 8760} \approx 0{,}416$
- $R_{systeme}(t) = R_1(t) \times [1 - (1 - R_2(t))(1 - R_3(t))]$
  ce qui donne $R_{systeme}(t = 10\ ans) \approx 0{,}512$

Les facteurs de Birnbaum pour chaque composant vont se baser sur ces fiabilités ainsi que sur l'hypothèse si le composant fonctionne ou non, et la structure du système.

Pour le composant 1 :

$$R_{systeme}\left(1_1, R_{systeme}(t)\right) = 1 - (1 - R_2(t))(1 - R_3(t))$$

$$R_{systeme}\left(0_1, R_{systeme}(t)\right) = 0$$

$$\Rightarrow\ I^B(1|\text{système défaillant à t}) = 1 - \left(1 - R_2(t)\right)\left(1 - R_3(t)\right)$$

Ce qui donne $\boldsymbol{I^B(1|\text{système défaillant à } t = 10\ ans) \approx 0{,}517}$

Pour le composant 2 :

$$R_{systeme}\left(1_2, R_{systeme}(t)\right) = R_1(t)$$

$$R_{systeme}\left(0_2, R_{systeme}(t)\right) = R_1(t) \times R_3(t)$$

$$\Rightarrow\ I^B(2|\text{système défaillant à t}) = R_1(t)\left(1 - R_3(t)\right)$$

Ce qui donne $\boldsymbol{I^B(2|\text{système défaillant à } t = 10\ ans) \approx 0{,}578}$

Pour le composant 3 :

$$R_{systeme}\left(1_3, R_{systeme}(t)\right) = R_1(t)$$

$$R_{systeme}\left(0_3, R_{systeme}(t)\right) = R_1(t) \times R_2(t)$$

$$\Rightarrow\ I^B(3|\text{système défaillant à t}) = R_1(t)\left(1 - R_2(t)\right)$$

Ce qui donne $\boldsymbol{I^B(3|\text{système défaillant à } t = 10\ ans) \approx 0{,}819}$

Pour rappel, plus le facteur de Birnbaum du composant *i* est élevé, plus le composant en question a un impact sur la fiabilité du système. Ici nous voyons que le composant 3 est le plus critique. Même si la structure du système devrait donner une plus grande importance pour le composant 1 mis en série (coupe d'ordre 1) par rapport à la structure en parallèle des composants

2 et 3 (coupe d'ordre 2), ces derniers ont un plus grand impact en raison de leur fiabilité trop faible pour que la structure parallèle soit pertinente à utiliser.

Les facteurs d'importance critique se basent sur celui de Birnbaum en complément des fiabilités déjà calculées.

Pour le composant 1 :

$$\mathrm{I^{CR}}(1|\text{système défaillant à t}) = \frac{\mathrm{I^{B}}(1|\text{système défaillant à t})(1 - R_1(t))}{1 - R_{systeme}(t)}$$

$$\Rightarrow \; I^{CR}(1|\text{système défaillant à t}) = \frac{[1-(1-R_2(t))(1-R_3(t))]\times(1-R_1(t))}{1-R_1(t)\times[1-(1-R_2(t))(1-R_3(t))]}$$

Ce qui donne $\boldsymbol{I^{CR}(1|\text{système défaillant à } t = 10 \text{ ans}) \approx 9,53.10^{-3}}$

Pour le composant 2 :

$$\mathrm{I^{CR}}(2|\text{système défaillant à t}) = \frac{\mathrm{I^{B}}(2|\text{système défaillant à t})(1 - R_2(t))}{1 - R_{systeme}(t)}$$

$$\Rightarrow \; I^{CR}(2|\text{système défaillant à t}) = \frac{R_1(t)(1-R_3(t))\times(1-R_2(t))}{1-R_1(t)\times[1-(1-R_2(t))(1-R_3(t))]}$$

Ce qui donne $\boldsymbol{I^{CR}(2|\text{système défaillant à } t = 10 \text{ ans}) \approx 0,979}$

Pour le composant 3 :

$$\mathrm{I^{CR}}(3|\text{système défaillant à t}) = \frac{\mathrm{I^{B}}(3|\text{système défaillant à t})(1 - R_3(t))}{1 - R_{systeme}(t)}$$

$$\Rightarrow \; I^{CR}(3|\text{système défaillant à t}) = \frac{R_1(t)(1-R_2(t))\times(1-R_3(t))}{1-R_1(t)\times[1-(1-R_2(t))(1-R_3(t))]}$$

Ce qui donne $\boldsymbol{I^{CR}(3|\text{système défaillant à } t = 10 \text{ ans}) \approx 0,981}$

De même que pour le facteur de Birnbaum, le facteur d'importance critique montre ici que les composants 2 et 3 sont largement plus impactant sur la fiabilité du système que le composant 1 en série. Ici, nous voyons clairement l'impact des composants mis en évidence par la valeur du facteur d'importance.

Maintenant, calculons les valeurs trouvées à partir d'une simulation de Monte Carlo sur le système choisi en exemple, avec le modèle développé et T = 10 ans. Nous obtenons alors pour le système :

- $\mathrm{n}_{1\mathrm{def}}^{\mathrm{sys}}(\mathrm{T}) = 2433$
- $\mathrm{n}_0 = 5000$

Pour chaque composant, avec $i = \{1,2,3\}$, nous avons :

- $\mathrm{n}_{1\mathrm{def}}^{\mathrm{sys}}(\mathrm{T}|1) = 32$ ; $\mathrm{n}_{1\mathrm{def}}^{\mathrm{sys}}(T|2) = 1022$ ; $\mathrm{n}_{1\mathrm{def}}^{\mathrm{sys}}(T|3) = 1379$

- $n_{1def}^{sys}(T|t_{01} < T) = 32$ ; $n_{1def}^{sys}(T|t_{02} < T) = 1375$ ; $n_{1def}^{sys}(T|t_{03} < T) = 990$
- $n(T|t_{01} < T) = 32$ ; $n(T|t_{02} < T) = 1474$ ; $n(T|t_{03} < T) = 891$
- $n_{1def}^{1}(T) = 32$ ; $n_{1def}^{2}(T) = 4098$ ; $n_{1def}^{3}(T) = 2891$

Ainsi, les facteurs de Birnbaum et d'importance critique pour chaque composant *i* sont :

⇨ $\mathbf{I^B(1|système\ défaillant\ à\ T) = 0{,}517}$ ; $\mathbf{I^{CR}(1|système\ défaillant\ à\ T) = 6{,}79.10^{-3}}$

⇨ $\mathbf{I^B(2|système\ défaillant\ à\ T) = 0{,}578}$ ; $\mathbf{I^{CR}(2|système\ défaillant\ à\ T) = 0{,}974}$

⇨ $\mathbf{I^B(3|système\ défaillant\ à\ T) = 0{,}820}$ ; $\mathbf{I^{CR}(3|système\ défaillant\ à\ T) = 0{,}974}$

Nous observons donc ici, que ce soit pour le cas théorique ou pour le cas pratique, que les composants ayant le plus d'impact sur le système sont les composants 2 et 3 en parallèle, en raison du faible niveau de fiabilité de ces derniers qui ne permet pas de compenser la structure en parallèle.

Dans le cadre d'une simulation de Monte Carlo, pour estimer l'impact de chaque composant sur le système global, ou une de ses aires, nous réalisons l'algorithme suivant.

---

*Algorithme d'évaluation de l'impact d'un composant j*

---

T = durée de simulation d'une histoire
dt = pas de temps pour estimer la défaillance *(1 h)*
$n_0$ = nombre d'histoires réalisées pour la simulation de Monte Carlo
$n_{sys1def}$ = nombre de premières défaillance du système initialisé à 0
$n_{j1def}$ = nombre de premières défaillance du composant j initialisé à 0
$n_{j\_sys1def}$ = nombre de premières défaillance du système initialisé à 0 où le composant j est la source de la défaillance du système
$n_{j1def\_avant\_sys1def}$ = nombre de premières défaillance du système initialisé à 0 où le composant j a été défaillant avant
$attenteDefaillance_{sys}$ = booléen initialisé à "vrai" et indiquant si nous sommes encore en attente d'une défaillance du système
$attenteDefaillance_j$ = booléen initialisé à "vrai" et indiquant si nous sommes encore en attente d'une défaillance du composant j

**Pour $n_i$ allant de 1 à $n_0$** *(pour chaque histoire)*

Lire le fichier de données de simulation de $n_i$ avec la liste des événements étant arrivés sur le système $ListeEvent_i$
$attenteDefaillance_{sys}$ = "vrai" ; $attenteDefaillance_j$ = "vrai"
**Si evenement = Défaillance**
**Alors**
**Si la source est le composant j**
**Alors**
**Si $attenteDefaillance_j$ = "vrai"**
**Alors**
$n_{j1def} = n_{j1def} + 1$ ; $attenteDefaillance_j$ = "faux"
**Fin_Si**
**Fin_Si**

**Si le système est dégradé ET attenteDefaillance$_{sys}$ = "vrai"**
**Alors**
$n_{sys1def} = n_{sys1def} + 1$ ; attenteDefaillance$_{sys}$ = "faux"
**Si la source est le composant j**
**Alors**
$n_{j\_sys1def} = n_{j\_sys1def} + 1$
**Fin_Si**
**Si attenteDefaillance$_j$ = "faux"**
**Alors**
$n_{j1def\_avant\_sys1def} = n_{j1def\_avant\_sys1def} + 1$
**Fin_Si**
**Fin_Si**
**Fin_Si**
**Fin_Pour**

⇨ Obtention des vecteurs comptant les nombre d'histoires avec des premières défaillances $n_{sys1def}$, $n_{j1def}$, $n_{j\_sys1def}$, $n_{j1def\_avant\_sys1def}$

---

#### *4.5.3.2. Facteur d'amélioration potentielle de la fiabilité*

Cet indicateur donne l'impact de chaque composant sur la fiabilité de l'architecture, ou de la zone, pendant toute la durée de simulation, comment améliorer la fiabilité. Ceci implique de compter toutes les défaillances pour chaque composant de l'architecture, ou de l'aire, jusqu'à l'instant de fin de simulation.

Il se base sur le facteur d'importance d'amélioration potentielle, appelé *Improvement potential measure* en anglais. Comme il a été dit dans le chapitre précédent, ce facteur est la différence de la fiabilité du système sachant que le composant est opérationnel et la fiabilité du système sachant que le composant est parfait (comparaison des deux situations) :

$$I^{IP}(composant_i|t) = R_{systeme}(1_i, R(t)) - R_{systeme}(t) \qquad (4.24)$$

avec :

- $R_{systeme}(1_i, R(t))$ la fiabilité du système à l'instant t en supposant que le composant *i* est parfait
- $R_{systeme}(t)$ la fiabilité du système à l'instant t
- $I^{IP}(composant_i|t)$ le facteur du composant *i* à l'instant t

Dans le cadre d'une simulation de Monte Carlo et de son approche statistique, estimer ce facteur sur toute la durée de simulation pour chaque composant dans le système peut se traduire par la formule suivante :

$$I^{IP}(i|\,T) = R_{systeme}(1_i, R(T)) - R_{systeme}(T)$$

$$I^{IP}(i|T) = \left[1 - \frac{n_{1def}^{sys}(t) - n_{1def}^{sys}(t|i) - n_{1def}^{sys}(t|t_{0i} < t)}{n_0 - n_{1def}^{sys}(t|i) - n_{1def}^{sys}(t|t_{0i} < t)}\right] - \left[1 - \frac{n_{1def}^{sys}(T)}{n_0}\right] \quad (4.25)$$

avec :

- $n_0$ le nombre total d'histoires réalisées pour la simulation de Monte Carlo
- T la durée de simulation pour une histoire
- $n_{1def}^{sys}(T)$ le nombre d'histoires où il y a eu au moins une défaillance du système sur T
- $n_{1def}^{sys}(T|i)$ le nombre d'histoires où il y a eu au moins une défaillance du système sur T où le composant était la source de la première défaillance du système
- $n_{1def}^{sys}(t|t_{0i} < t)$ le nombre d'histoires où le système a eu sa première défaillance avant à l'instant t avec le composant *i* qui a été défaillant avant t

*Exemple :*

Reprenons l'exemple dans la section 4.5.3.1.

**Valeurs numériques (Rappel) : $\boldsymbol{\lambda_1 = 10^{-7}\ h^{-1}}$ ; $\boldsymbol{\lambda_2 = 2.10^{-5}\ h^{-1}}$ ; $\boldsymbol{\lambda_3 = 10^{-5}\ h^{-1}}$**

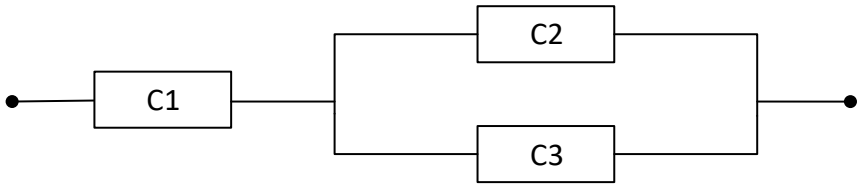



*Fig. 4. 19 : Exemple de système (Rappel)*

Comme précédemment, calculons d'abord les valeurs théoriques du facteur d'importance d'amélioration potentielle pour chaque composant. Pour rappel, nous fixons l'instant à t=10 ans.

La fiabilité du système est : $R_{systeme}(t) = R_1(t) \times [1 - (1 - R_2(t))(1 - R_3(t))]$ ce qui donne $R_{systeme}(t = 10\ ans) \approx 0{,}512$

Le facteur d'importance d'amélioration potentielle pour chaque composant va se baser sur cette fiabilité ainsi que sur la fiabilité du système quand le composant est supposé parfait, et la structure du système.

Pour le composant 1 :

$$R_{systeme}\left(1_1, R_{systeme}(t)\right) = 1 - (1 - R_2(t))(1 - R_3(t))$$

⇨ $I^{IP}(1|\text{système défaillant à t}) = 1 - R_1(t) - \left(1 - R_1(t)\right)\left(1 - R_2(t)\right)\left(1 - R_3(t)\right)$

Ce qui donne $\boldsymbol{I^{IP}(1|\text{système défaillant à t} = 10\ \text{ans}) \approx 0{,}005}$

Pour le composant 2 :

$$R_{systeme}\left(1_2, R_{systeme}(t)\right) = R_1(t)$$

$$\Rightarrow \quad I^{IP}(2|\text{système défaillant à t}) = R_1(t)\big(1 - R_2(t)\big)\big(1 - R_3(t)\big)$$

Ce qui donne $\boldsymbol{I^{IP}(2|\text{système défaillant à } t = 10 \text{ ans}) \approx 0,479}$

Pour le composant 3 :

$$R_{systeme}\left(1_3, R_{systeme}(t)\right) = R_1(t)$$

$$\Rightarrow \quad I^{IP}(3|\text{système défaillant à t}) = R_1(t)\big(1 - R_2(t)\big)\big(1 - R_3(t)\big)$$

Ce qui donne $\boldsymbol{I^{IP}(3|\text{système défaillant à } t = 10 \text{ ans}) \approx 0,479}$

Ici nous voyons que pour améliorer la fiabilité du système, il faut travailler sur les composants 2 et 3, ce qui conforte les observations faites précédemment, où nous avons vu que les composants 2 et 3 étaient les plus critiques et que leur faible fiabilité annulait les bénéfices de la structure parallèle. Si on améliorait ces composants en priorité, on pourrait augmenter significativement la fiabilité de l'ensemble du système et le rendre plus robuste à l'arrivée de défaillances.

Maintenant, calculons les valeurs trouvées à partir d'une simulation de Monte Carlo sur le même système, toujours avec le modèle développé.

Nous avons pour le système :

- T = 10 ans
- $n_0 = 5000$
- $n_{1def}^{sys}(T) = 2433$

Pour chaque composant, avec *i* = {1,2,3}, nous avons :

- $n_{1def}^{sys}(T|1) = 32$ ; $n_{1def}^{sys}(T|2) = 1022$ ; $n_{1def}^{sys}(T|3) = 1379$
- $n_{1def}^{sys}(T|t_{01} < T) = 32$ ; $n_{1def}^{sys}(T|t_{02} < T) = 1375$ ; $n_{1def}^{sys}(T|t_{03} < T) = 990$

Ainsi le facteur d'importance d'amélioration potentielle pour chaque composant *i* = {1,2,3} est :

$$\Rightarrow \quad \mathbf{I^{IP}(1|système défaillant à T) = 0,0066}$$
$$\Rightarrow \quad \mathbf{I^{IP}(2|système défaillant à T) = 0,473}$$
$$\Rightarrow \quad \mathbf{I^{IP}(3|système défaillant à T) = 0,462}$$

Nous observons donc ici, que ce soit pour le cas théorique ou pour le cas pratique, que les composants sur lesquels nous devons travailler pour améliorer la fiabilité du système sont les composants 2 et 3.

Ainsi, pour estimer l'impact de chaque composant sur le système global, ou une de ses aires, nous réalisons l'algorithme suivant.

T = durée de simulation d'une histoire
dt = pas de temps pour estimer la défaillance *(1 h)*
$n_0$ = nombre d'histoires réalisées pour la simulation de Monte Carlo
$n_{sys1def}$ = nombre de premières défaillance du système initialisé à 0
$n_{j\_sys1def}$ = nombre de premières défaillance du système initialisé à 0 où le composant j est la source de la défaillance du système
$n_{j1def\_avant\_sys1def}$ = nombre de premières défaillance du système initialisé à 0 où le composant j a été défaillant avant
$attenteDefaillance_{sys}$ = booléen initialisé à "vrai" et indiquant si nous sommes encore en attente d'une défaillance du système
$attenteDefaillance_j$ = booléen initialisé à "vrai" et indiquant si nous sommes encore en attente d'une défaillance du composant j

**Pour $n_i$ allant de 1 à $n_0$** *(pour chaque histoire)*

Lire le fichier de données de simulation de $n_i$ avec la liste des événements étant arrivé sur le système $ListeEvent_i$
$attenteDefaillance_{sys}$ = "vrai" ; $attenteDefaillance_j$ = "vrai"
**Si evenement = Défaillance**
**Alors**
**Si la source est le composant j**
**Alors**
**Si $attenteDefaillance_j$ = "vrai"**
**Alors**
$attenteDefaillance_j$ = "faux"
**Fin_Si**
**Fin_Si**
**Si le système est dégradé ET $attenteDefaillance_{sys}$ = "vrai"**
**Alors**
$n_{sys1def} = n_{sys1def} + 1$ ; $attenteDefaillance_{sys}$ = "faux"
**Si la source est le composant j**
**Alors**
$n_{j\_sys1def} = n_{j\_sys1def} + 1$
**Fin_Si**
**Si $attenteDefaillance_j$ = "faux"**
**Alors**
$n_{j1def\_avant\_sys1def} = n_{j1def\_avant\_sys1def} + 1$
**Fin_Si**
**Fin_Si**
**Fin_Si**
**Fin_Pour**

⇨ Obtention des vecteurs comptant les nombre d'histoires avec des premières défaillances $n_{sys1def}$, $n_{j\_sys1def}$, $n_{j1def\_avant\_sys1def}$

## 4.6. Conclusion

Ce chapitre a présenté le développement de la méthode d'évaluation des performances de sûreté de fonctionnement pour une architecture de contrôle-commande. Nous avons détaillé depuis les entrées du modèle jusqu'aux sorties attendues de celle-ci, comment le modèle a été construit et sous quelles formes se présente les paramètres, les données ainsi que la structure du modèle. Nous avons eu une vue macroscopique de la méthode proposée. Puis nous nous sommes penchés sur les différentes briques qui constituent le modèle d'évaluation RdP de l'architecture, avant de s'intéresser au modèle global. En complément du modèle en lui-même, nous avons défini l'instanciation et le paramétrage du modèle, ainsi que la création d'observateurs de performance pour nous permettre de récupérer les données de la simulation de Monte Carlo. Enfin, nous avons vu comment, à partir des données brutes de la simulation, les performances de sûreté de fonctionnement ont été estimées. Avec cette conclusion, nous avons terminé la présentation concernant l'explication de la méthode d'évaluation des performances fiabilistes développée et qui sera déployée au sein de Schneider Electric. Le chapitre suivant va se focaliser sur l'application industrielle de cette méthode.

# Chapitre 5
# Application industrielle

## 5.1. Introduction

Dans un premier temps, l'objectif de ce chapitre est de présenter le développement d'un prototype qui implémente la méthode de modélisation et d'évaluation des performances de SdF proposée au chapitre précédent. Le développement de cet outil se base, comme pour l'outil développé par Moulaye Ndiaye (Ndiaye, 2017), sur CPN Tools[1] permettant l'édition et la simulation de réseaux de Petri Colorés et temporisés. Il est interfacé avec dans l'outil d'avant-vente de Schneider-Electric appelé EcoStruxure Plant Builder[2] (ECX Builder). Compte tenu de la durée des périodes de simulation et du nombre d'histoires simulées, très largement supérieur à ceux présents dans l'évaluation des performances temporelles réalisée dans ces travaux précédents, un effort a été porté sur l'optimisation du temps de simulation notamment via une amélioration sur le formalisme d'échange de données entre l'interface graphique et le simulateur.

Dans un deuxième temps, un cas d'étude permet d'illustrer la mise en œuvre de notre proposition sur une architecture de contrôle-commande d'une cimenterie. Cette application permet de mettre en évidence la faisabilité de notre démarche et sa pertinence pour évaluer les performances de sûreté de fonctionnement de différentes architectures.

## 5.2. Développement d'un prototype pour l'évaluation des performances de SdF

Le prototype d'outil pour l'évaluation de performances fiabilistes se base sur l'architecture fonctionnelle décrite dans la Fig. 5. 1 qui comprend trois entités distinctes :

- L'interface Graphique est le point d'entrée de la simulation. Elle permet de configurer l'architecture en sélectionnant les différents composants, de définir les conditions d'utilisation et enfin les paramètres de simulation. Cette interface aussi permet l'affichage des résultats obtenus après simulation.
- Un module de convertisseur qui fait le lien entre l'interface graphique et le simulateur CPN (Jensen & Kristensen, 2009). Sa mission principale est de convertir dans le sens interface graphique vers simulateur et vice-versa de fichiers basés sur des modèles de données.
- Le simulateur CPN qui regroupe un compilateur de Standard ML (ML, 1998) avec le modèle unique que nous avons vu dans la section 4.4.2. Sa mission est de simuler l'architecture selon la méthodologie définie dans le chapitre 4 et de renvoyer des données brutes de simulation.

[1] CPN Tools est développé et distribué par Eindhoven University of Technology

[2] ECX Builder est un outil de cotation de matériel développé par Schneider-Electric

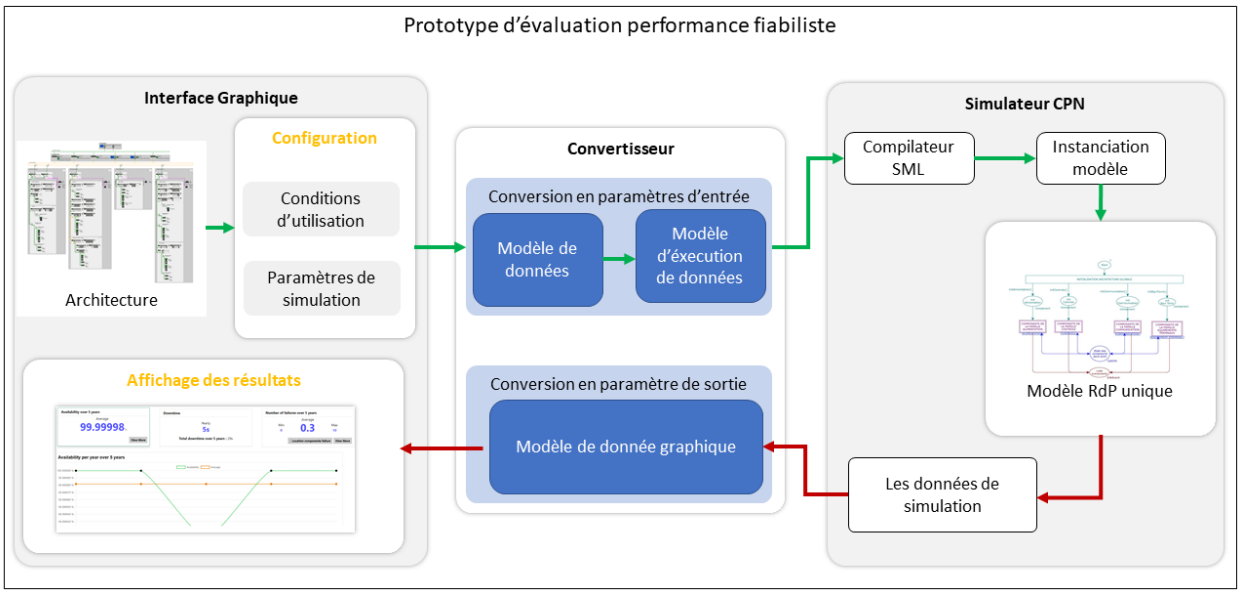



*Fig. 5. 1 : Architecture fonctionnelle du prototype*

Ainsi le développement de ce prototype se repose donc sur les éléments suivants :

- L'outil CPN Tools pour la construction du modèle unique ainsi que pour la simulation de celui-ci à travers son simulateur CPN
- Un ensemble de modules pour la conversion des modèles de données
- L'interface graphique de l'outil pour la configuration de l'architecture ainsi que des paramètres de simulation.

Afin d'expliciter le prototype, nous allons décrire dans les sections suivantes comment se fait le passage de la configuration de l'architecture vers le lancement de la simulation. Ensuite de la fin de la simulation à l'affichage des résultats.

### 5.2.1. De la configuration de l'architecture à la simulation

#### *5.2.1.1. L'interface de configuration*

L'interface de configuration de l'architecture est le point de départ de la simulation. L'architecture devant être configurée est fournie par l'outil ECX Builder. En effet cet outil permet la sélection des différents composants qu'une architecture de C/C et de définir aussi leurs liens que cela soit topologique ou physique. Ainsi dans le cadre de ce prototype nous définissons une architecture en amont avec l'outil ECX Builder et nous l'exportons dans un fichier spécifique.

Afin de lire le fichier ECX Builder une interface complémentaire a été développée; cette interface a deux missions. La première est d'importer le fichier d'architecture définie avec ECX Builder et de l'afficher dans une interface graphique. La deuxième est de fournir un ensemble de données de configuration à cette architecture. Comme nous l'avons vu dans les sections 4.4.2.2 et 4.4.2.3 dans le chapitre 4, afin de faire une simulation en vue d'évaluer les performances fiabilistes, il nous faut définir les conditions dans lesquelles les composants de l'architecture vont fonctionner.

Ces conditions, dans le cadre spécifique de Schneider Electric, sont la température de fonctionnement ainsi que le niveau de demande (section 4.4.2.2). Elles permettent d'adapter les paramètres fiabilistes de chaque composant en fonction du contexte d'utilisation, ici le temps moyen de fonctionnement avant défaillance, la MTBF. A cela s'ajoute la possibilité de configurer le MTTR pour chaque composant de l'architecture. Les autres données nécessaires et requises dans l'interface de configuration concernent les paramètres de simulation : durée de simulation ainsi que le nombre de réplications à effectuer (section 4.4.2.2).

Le prototype intègre donc une interface (Fig. 5. 2) permettant de présenter la configuration de l'architecture générée par le ECX Builder et définir l'ensemble des données de configuration (conditions d'utilisation, paramètres fiabilistes, durée de simulation…).

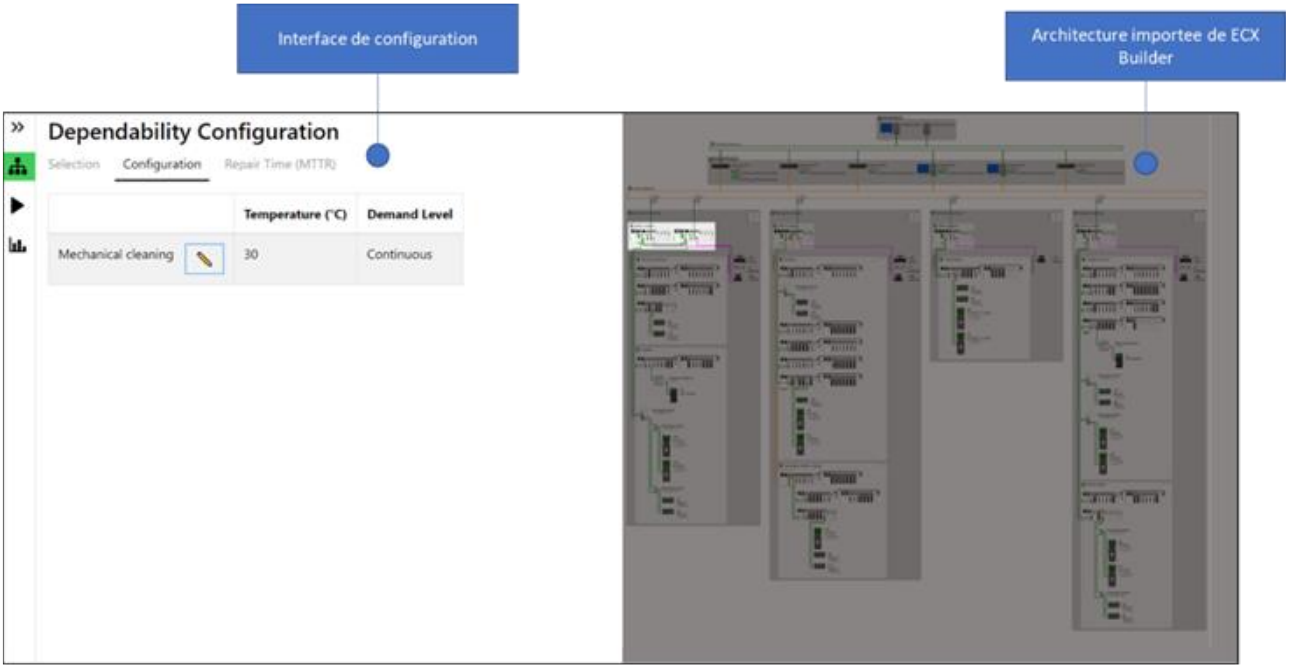



*Fig. 5. 2 : Interface de configuration*

#### *5.2.1.2. Le module de conversion*

Une fois que la configuration de l'architecture a été réalisée grâce à l'interface de configuration et que les conditions d'utilisation et de simulation définies, le prototype fait une conversion de ces données en un modèle de données. En effet, comme le décrit la Fig. 5. 3 un module de conversion a été réalisé dans le prototype.

Ce module de conversion et de génération permet la transcription des informations liées à l'architecture, et aux conditions d'utilisation et de simulation en générant un fichier au format *.xml* (Bray, Paoli, & Sperberg-McQueen, 1997). Ce format a été choisi pour deux raisons la première est que le format d'échange interne de l'outil ECX Builder est basé sur ce formalisme. Donc à terme, si une intégration de ce prototype doit être réalisée avec ECX Builder cela pourra se faire sans un effort supplémentaire de réadaptation. La deuxième raison est que c'est un format qui permet de faire une analyse aisée des données en les structurant sous la forme hiérarchique mais

surtout que c'est un formalisme que tous types de programmes ou langages informatiques sont capable de lire et de traiter ses données. En effet CPN Tools est basé sur le langage fonctionnel alors que ECX Builder quant à lui est basé sur un langage orienté objet. Or comme nous l'avons décrit dans le chapitre 4 notre méthode est basée sur de la modélisation de RdP avec l'outil CPN Tools et que l'instanciation et la paramétrisation des modèles se font à travers des fonctions (sections 4.4.2.1 et 4.4.2.2). Donc afin de créer ces fonctions pour instancier et paramétrer le modèle à partir de l'architecture, nous avons créé un algorithme qui analyse le fichier *.xml* afin de générer un fichier en *.sml*. Le format *.sml* est l'extension de fichier que le compilateur SML peut lire et exécuter (ML, 1998). Ce fichier *.sml* contient l'ensemble des fonctions d'instanciations et de paramétrisation dont les valeurs ont été mises à jour en fonction des composants de l'architecture ainsi que les conditions d'utilisation. Sur ce fichier est présent aussi le temps relatif à la durée de la simulation ainsi que le nombre de réplication à effectuer. Ce fichier sera envoyé vers le simulateur CPN.

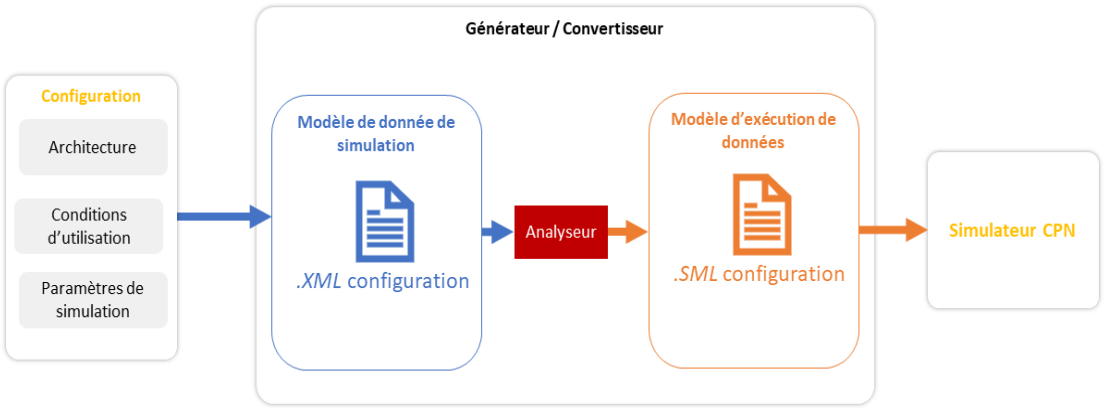



*Fig. 5. 3 : Architecture fonctionnelle du générateur et convertisseur de fichier*

#### *5.2.1.3. Le serveur de simulation*

Le Serveur de simulation CPN appelé aussi *CPN server* est le point central de l'outil d'estimation des performances fiabilistes.

Ses missions sont les suivantes :

- Intégrer le modèle unique qui regroupe l'ensemble des composants génériques.
- A partir du fichier *.sml* de paramétriser et d'instancier le modèle unique de RdP en fonction de l'architecture et de sa configuration.
- Exécuter la simulation de Monte-Carlo afin d'estimer les performances fiabilistes.
- Renvoyer les relevés statistiques des moniteurs afin que les résultats soient traités.

Le serveur CPN est une entité de l'outil CPN Tools qui a été extrait de son environnement d'origine. En effet CPN Tools est composé de deux entités distinctes, le premier est le serveur de simulation CPN. Le deuxième est un environnement de développement sous forme d'interface graphique appelé *Design CPN* (Jensen & Kristensen, 2009). Cette interface permet la modélisation en RdP ainsi que l'intégration des différents moniteurs. C'est avec cette interface que le modèle

unique de RdP a été développé, où les fonctions d'incidences arrière, les variables ainsi que les moniteurs ont été définis.

Modèle unique de RdP

Un modèle unique de RdP a été développé lors de ce prototypage. Comme l'indique la Fig. 5. 4, ce modèle représente le modèle de RdP de haut niveau regroupant l'ensemble des sous modèles de RdP des composants. Ces sous-modèles de composant ont été modélisés selon les besoins exprimés par Schneider Electric.

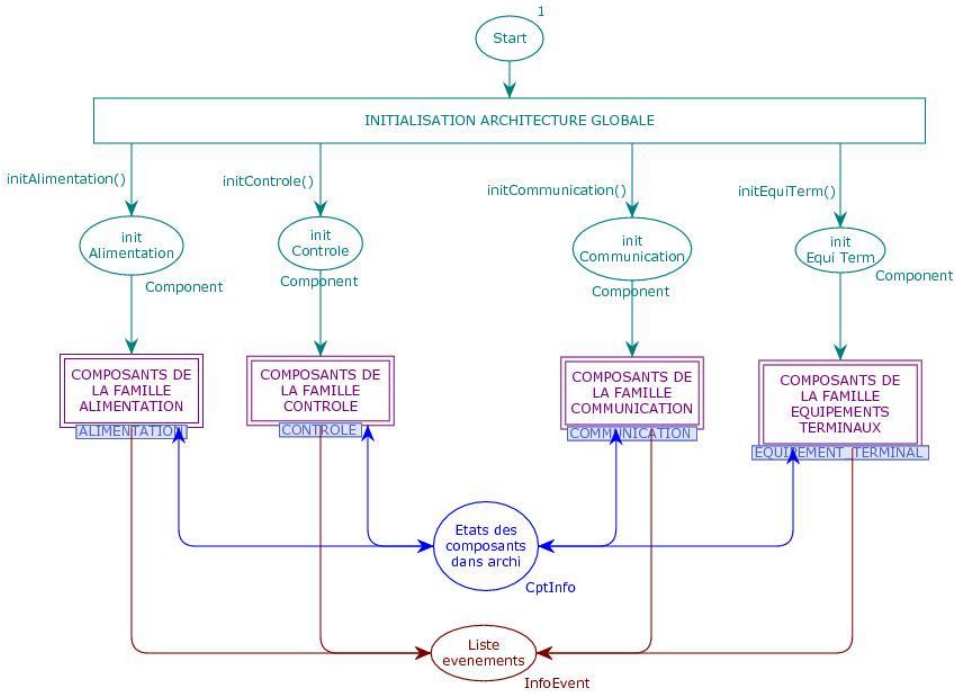



*Fig. 5. 4 : Modèle unique de RdP*

Ce modèle unique de RdP se compose de 4 grandes catégories répartissant les familles de composants qui sont les plus courantes dans une architecture de C/C.

Ces quatre grandes catégories familles se décrivent à travers :

- La catégorie *Alimentation* qui regroupe un modèle de RdP représentant le comportement des alimentations et les fonds de panier que cela soit pour les processeurs, ou les E/S locales ou déportées.
- La catégorie *Contrôle* qui regroupe un modèle de RdP représentant l'ensemble des composants de control plus précisément les processeurs, et les interfaces dans les supports E/S déportés etc…
- La catégorie *Communication* qui englobe un modèle de RdP représentant l'ensemble des modules de communications d'une architecture de C/C que ceux-ci soient locaux ou déportés.
- La catégorie *Equipement terminaux* qui permet regroupe un modèle de RdP représentant le comportement d'équipements terminaux (variateur de vitesse, gestion de l'énergie etc.).

Il est important de remarquer que chaque modèle de famille de composant est englobé au sein d'une transition de substitution (section 4.4.2). Ce choix est lié au fait que d'une part cela permet d'avoir une lisibilité du modèle, mais aussi d'ajouter rapidement et facilement d'autres familles de composant en cas d'évolution de l'outil sans altérer le modèle unique. Ce rajout se fera en développant de manière séparée le modèle de la nouvelle famille, ensuite de le mettre dans une transition de substitution et enfin une fois validé de l'associer au modèle unique.

Il est aussi important de rappeler que le modèle unique de RdP contient la définition des différents ensembles de couleur définis dans le chapitre 4. Ces ensembles de couleurs sont associés à des fonctions d'incidence arrière sur les arcs sortant de la transition nommée '*initialisation_architecture_globale'*. Ces fonctions nommées ici *initAlimentation(), initControle, initCommunication() et initEquiTerm()* ont pour mission d'instancier et de paramétriser le modèle de RdP. Cette instanciation et paramétrisation se fera par la mise à jour des valeurs de ces fonctions à partir fichier *.sml* renvoyer par le convertisseur. Ainsi, une fois que les fonctions d'incidence arrière ont été mise à jour, le franchissement de la transition *'initialisation_architecture_globale'* entraine l'exécution de ces fonctions et le placement dans les places en avales les jetons correspondant aux composants de chaque famille.

Entre chaque transition de substitution représentant une famille de composant, la place nommée '*Etats_des_composants_dans_l'archi'* permet de faire le lien de dépendant entre les familles de composants. Enfin comme d'écrit dans le chapitre 4 dans la section 4.4.2.3, la place *'Liste_Event'* est associé à un moniteur qui récupère l'ensemble des évènements relevés durant la simulation.

Gestion de la simulation par le serveur CPN

Le serveur de simulation CPN a été développé à partir du langage fonctionnel Standard ML (SML) (ML, 1998) par Jensen (Jensen & Kristensen, 2009). En pratique, une fois que le modèle de RdP a été développé pour exécuter celui-ci avec le server CPN une compilation et une transcription doit être réalisées. Comme l'indique la Fig. 5. 5, une fois qu'un modèle a été créé, la simulation de celui-ci passe par une compilation. Celle-ci est faite par un encodeur intégré au simulateur CPN qui, récupère le fichier généré par l'interface graphique design CPN au format *.cpn* pour ensuite générer en interne un fichier binaire exécutable. Celui-ci est ensuite exécuter grâce au compilateur SML et la simulation est alors lancée.

Si les fonctions (paramètres) du modèle doivent changer, il n'est pas nécessaire de recompiler complétement le modèle. Il suffit juste d'assigner les nouvelles valeurs de ces fonctions et d'exécuter à nouveau le modèle. Cette assignation peut se faire de manière manuelle c'est-à-dire écrire chaque valeur des fonctions sur une console et ensuite l'exécuter ou de manière automatique à travers un fichier *.sml* regroupant l'ensemble des fonctions à mettre à jour et ensuite exécuter celui-ci.

Comme l'indique la Fig. 5. 6 nous avons fait une réadaptation du principe de fonctionnement de CPN Tools pour notre prototype.

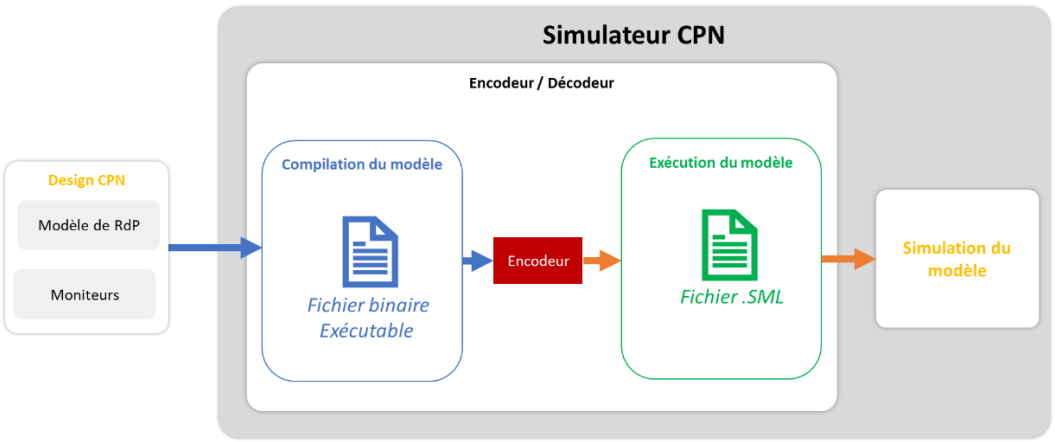



*Fig. 5. 5 : Modèle fonctionnel de CPN Tools*

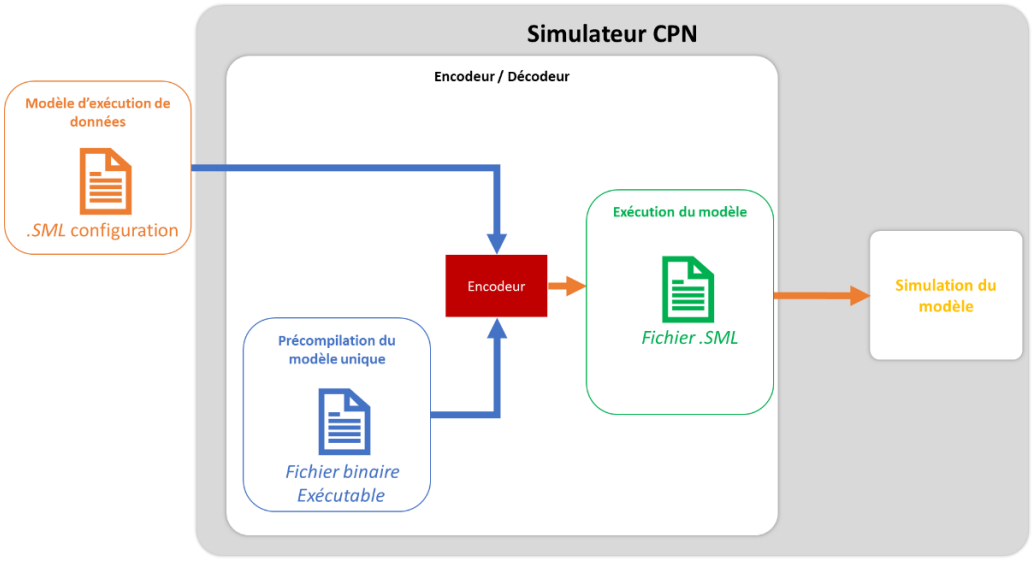



*Fig. 5. 6 : Modèle fonctionnel du prototype pour la simulation*

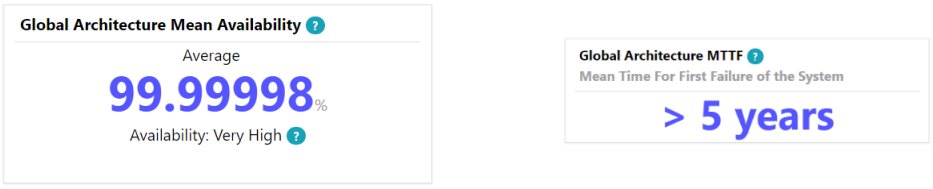
En effet, en se basant sur ce fonctionnement de CPN Tools, nous avons dès lors dans le cadre du prototype précompilé le modèle unique de RdP pour générer le fichier binaire exécutable. Puis nous avons extrait le simulateur dans son environnement de base en n'y intégrant le modèle unique de RdP précompilé. Ensuite pour la mise à jour des fonctions d'instanciation nous allons récupérer puis exécuter automatiquement le fichier *.sml* généré par le convertisseur (cf. section 5.2.1.2). Ce fichier associé au modèle pré compilé sont exécutés par le compilateur SML et la simulation est lancée. A la fin de la simulation le fichier de simulation (cf. section 4.5.1) est renvoyé par simulateur pour traitement des données.

### 5.2.2. De la simulation à l'affichage des résultats

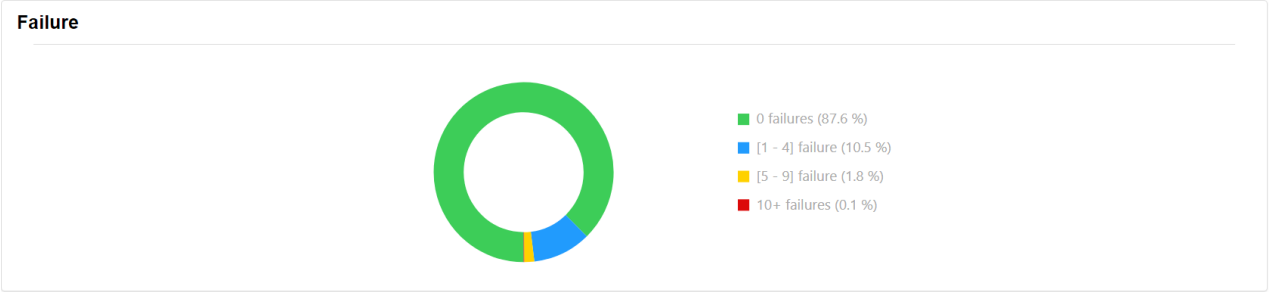
La fin de la simulation induit à la création d'un fichier au format *.xml* qui concatène les fichiers dans lequel l'ensemble des évènements notés pour chaque histoire durant la simulation sont répertoriés (cf. section 4.4.2.3). Contrairement à aux travaux de Moulaye Ndiaye (Ndiaye, 2017) où un fichier par réplication est créé, dans notre cas un seul et unique fichier l'est. Ce choix est lié

au nombre de réplications que nous faisons lors de la simulation. En effet, lors d'une campagne d'estimation des performances fiabiliste, plusieurs milliers de réplications sont réalisées. Créer un fichier *.xml* pour chaque réplication entrainerait de facto la création de plusieurs milliers de fichiers de très petite taille et demanderait donc un temps de traitement conséquent. Pour éviter ce problème, un seul et unique fichier est généré ; il regroupe les relevés statistiques de l'ensemble des réplications tout en conservant le référencement des données à chacune des histoires à d'identifiants uniques.

Le fichier .xml créé par le simulateur est converti, en deux étapes, vers un modèle de donnée graphique, comme l'indique la Fig. 5. 3 par le convertisseur. La première étape traite l'ensemble des données brutes stockées dans le fichier .xml pour les convertir en valeurs de performances sur la base des formules définies dans le chapitre 4. La deuxième étape récupère l'ensemble des valeurs de performances calculées et produit un ensemble de résultats statistiques présentés sous forme textuelle, de courbes, de graphes etc. dont la Fig. 5. 7 donne un exemple.




(a) Disponibilité de l'architecture globale et sa MTTF




(b) Un diagramme montrant le nombre de défaillances d'une partie de l'architecture


(c) Disponibilité d'une partie de l'architecture durant le temps de simulation.

*Fig. 5. 7 : Exemple d'affichage de résultat*

Ainsi grâce à ce prototype nous avons pu montrer à l'aide de CPN Tools et de la méthodologie proposée dans le chapitre 4 nous pouvons évaluer les performances fiabilistes d'une architecture de C/C durant la phase d'avant-vente. D'autant plus grâce à l'interface graphique propriétaire développée ainsi que le mécanisme de communication avec le serveur CPN, l'utilisateur peut se départir complétement de la complexité d'utilisation des RdP. En fait il ne saura même que c'est ce formalisme qui a été utilisé car complétement caché. Dès lors il définira son architecture comme il a l'habitude de faire et obtiendra ses résultats de manière rapide et claire.

## 5.3. Application sur un cas d'étude

### 5.3.1. Introduction du cas d'étude

Les applications de procédés industriels demandent de plus en plus des systèmes à haute disponibilité. Pour déterminer de la pertinence d'implémenter un tel système, un utilisateur final doit répondre à plusieurs questions :

- Quel est le niveau de sécurité nécessaire au niveau du matériel et également des hommes ?
- Quel est le caractère critique du processus ? Ce point concerne principalement les procédés qui impliquent une réaction mécanique ou chimique.
- Quelle est la criticité environnementale ?

Si nous prenons l'exemple d'un four à très haute température d'une cimenterie, un arrêt brutal peut le détruire et causer des dégâts importants matériels et parfois humain. En conséquence l'usine devra être arrêter pendant plusieurs mois pour réinstaller un nouveau four. Un arrêt programmé d'un four pour maintenance devra toujours suivre une séquence très précise de refroidissement et un arrêt progressif.

Un autre exemple est le traitement biologique d'une station d'épuration qui ne doit pas être arrêté très longtemps et fréquemment au risque de perdre l'efficacité de ce traitement. Nous pouvons prendre un dernier exemple qui peut entrainer des grands dommages au niveau des personnes, est le système de ventilation d'un tunnel. Le système de détection de fumée doit toujours être opérationnel ainsi que la ventilation du tunnel.

Par conséquent le niveau de disponibilité d'un système d'automatisme est une mesure qui permet de le qualifier et de vérifier si le système répond à la criticité de l'application. Le niveau de disponibilité est de plus très souvent spécifié dans le cahier des charges d'un nouveau projet par l'utilisateur final.

Le taux de fiabilité d'un système et son niveau de maintenabilité sont également des indicateurs qui permettent de consolider l'architecture d'automatisme proposé et qui sont très utiles pour prévoir un plan de maintenance adapté.

Le taux de disponibilité d'un système industriel est généralement classifié en sept niveaux. Le nombre de neuf du pourcentage de disponibilité est utilisé comme indicateur. Les sept niveaux sont listés dans le tableau suivant.

| Classe | Niveau de disponibilité | **Disponibilité (%)** | Temps d'arrêt | Nombre de Neuf |
|---|---|---|---|---|
| 1 | Non Géré | **90** | 36,5 jours | 1-neuf |
| 2 | Géré | **99** | 3,65 jours | 2-neufs |
| 3 | Bien géré | **99,9** | 8,76 heures | 3-neufs |
| 4 | Tolérant | **99,99** | 52,6 minutes | 4-neufs |
| 5 | Haute disponibilité | **99,999** | 5,26 minutes | 5-neufs |
| 6 | Très Haute disponibilité | **99,9999** | 30 secondes | 6-neufs |
| 7 | Ultra Haute disponibilité | **99,99999** | 3 secondes | 7-neufs |

*Tableau 5.1 : Niveau de disponibilité selon le nombre de neufs*

Le modèle et son application industrielle qui ont été présentés dans les chapitres précédents sont un outil très efficace d'aide à la décision lors de la définition d'une architecture d'automatisme. Un intégrateur système pourra rapidement, sans être un expert en disponibilité, obtenir des réponses sur la disponibilité et la fiabilité de son architecture en utilisant cet outil de simulation. Grace à ses résultats il sera capable de proposer l'architecture qui correspond le mieux aux exigences de son client.

Nous allons démontrer l'efficacité de cet outil industriel à travers un cas d'utilisation qui correspond à la conception d'une cimenterie.

### 5.3.2. Cas d'étude d'un projet de cimenterie

Le procédé d'une cimenterie comprend plusieurs étapes en partant de l'extraction des matériaux dans une carrière jusqu'à la mise en sac du ciment.

L'étape la plus critique est le four de la cimenterie. Le four rotatif est constitué d'un tube en tôle d'acier, revêtu de briques réfractaires, la température peut atteindre jusqu'à plus de 1000°C. Pour contrôler ce procédé, un système tolérant et si possible à haute disponibilité est nécessaire (cf. Tableau 5.1).

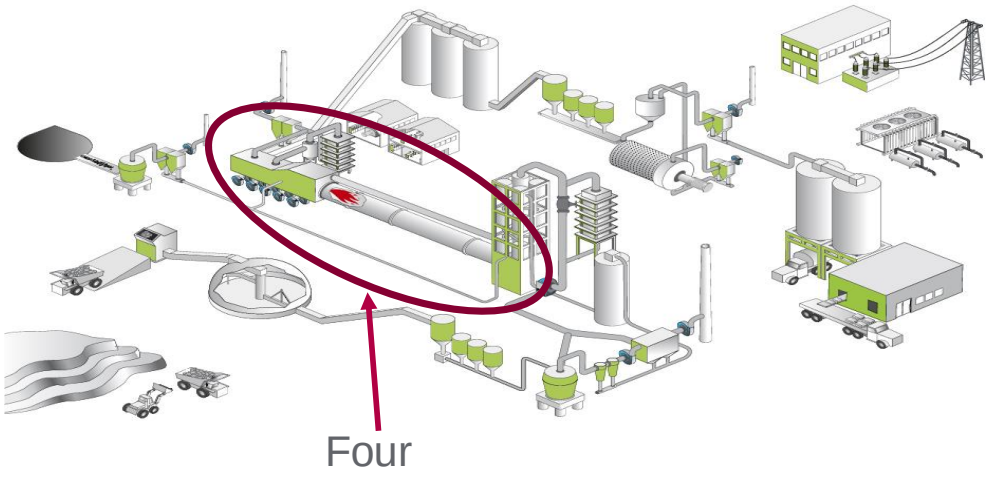



*Fig. 5. 8 : Usine de cimenterie*

Pour ce projet l'utilisateur final de la cimenterie exige au moins un système tolérant (4-neufs) pour l'automatisation du four mais un système avec une haute disponibilité (5-neufs) serait très apprécié. Le niveau de disponibilité dépend du type d'architecture d'automatisme proposé, principalement lié au niveau de redondance sélectionné. Le plan de maintenance est également un paramètre important pour réduire le MTTR des produits les plus critiques.

Trois types d'architectures candidats pour le projet en question peuvent être analysées rapidement par l'intégrateur système pour vérifier celle qui peut répondre le mieux aux attentes de son client en considérant également le cout de la solution sélectionnée :

- L'Architecture A dispose d'un automate autonome qui contrôle l'ensemble de l'unité de contrôle du four. Plusieurs zones regroupent des Entrées/Sorties (E/S) distantes qui pilotent le procédé. Ces E/S sont alimentées par des alimentations non redondées.
- L'Architecture B a un automate redondant pour contrôler comme dans le cas A l'ensemble des E/S distantes du four.
- L'Architecture C possède comme l'architecture B un automate redondant mais également des alimentations redondées pour alimenter toutes les E/S ainsi que l'API.

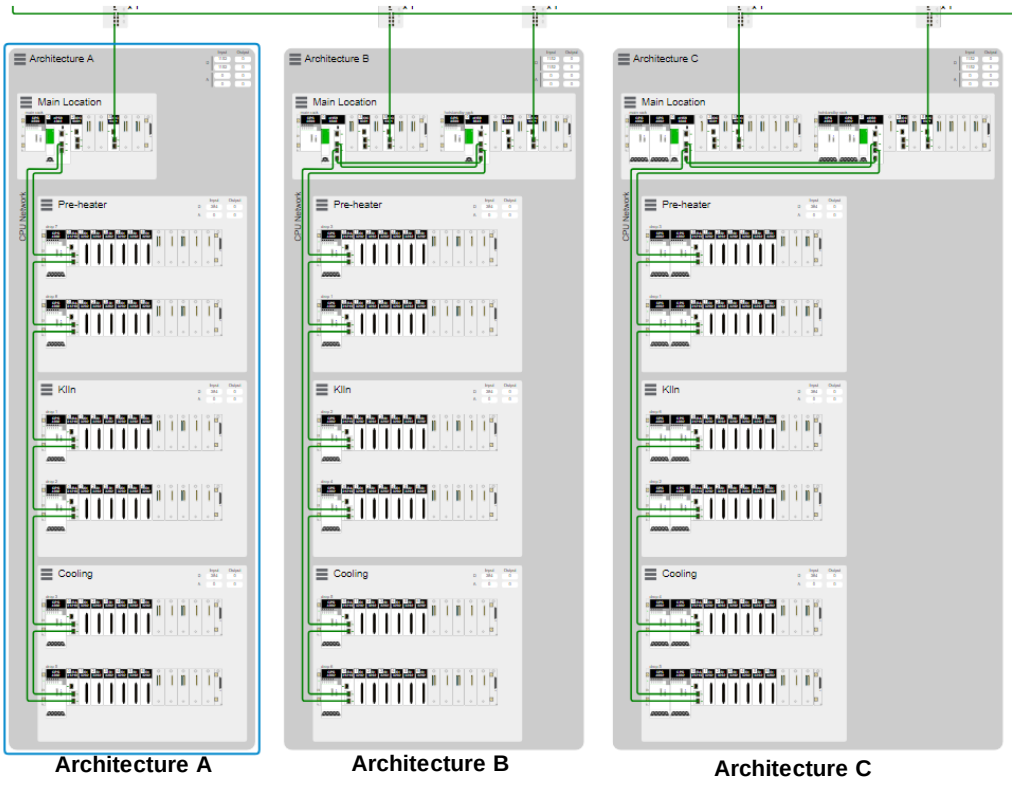



*Fig. 5. 9 : Les trois architectures à analyser*

L'outil décrit précédemment permet de simuler le comportement de chaque architecture sur une période de 30 années pour obtenir le niveau de disponibilité et de fiabilité de chacune de ces architectures. L'analyse des résultats permettra soit de prendre directement une décision soit

d'affiner une ou plusieurs architectures proposées. Ces résultats sont également une excellente base pour préparer un plan de maintenance pertinent.

Dans le cas de cette simulation, un temps de réparation de 8h (MTTR) a été définie pour l'ensemble des produits. Les résultats suivants sont obtenus :

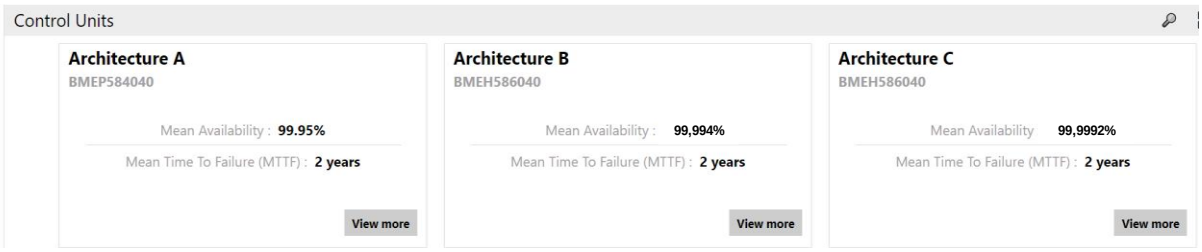



*Fig. 5. 10 : Résultats de l'outil de simulation*

Concernant l'exactitude des résultats, les intervalles de confiance que nous avons fixés à 99% donnent pour chaque architecture, les chiffres suivants :

| Partie | Disponibilité | IC 99% | MTTF | IC 99% |
|---|---|---|---|---|
| Architecture A | 99,95% | $\pm 5.10^{-5}$ % | 2 ans | ±14 jours |
| Architecture B | 99,994% | $\pm 6.10^{-6}$ % | 2 ans | ±19 jours |
| Architecture C | 99,9992% | $\pm 1,4.10^{-7}$ % | 2 ans | ±16 jours |

*Tableau 5.2 : Intervalles de confiance à 99% pour la Disponibilité et la MTTF*

Au vu des résultats apportés par les intervalles de confiance, nous pouvons affirmer que les conditions de simulations permettent des performances fiabilistes proches des chiffres moyens théoriques.

Les architectures B et C répondent aux exigences du cahier des charges qui demandait une architecture au moins Tolérante (4-neufs). L'architecture A peut déjà être éliminée de la sélection pour ce projet.

Pour une question de cout total du projet, l'architecture B est sélectionnée. Son niveau de disponibilité peut être néanmoins encore amélioré par le système intégrateur en travaillant sur deux axes :

- Une étude plus approfondie sur les modules proposés pour analyser si d'autres modules ayant un meilleur MTBF peuvent être proposés
- La définition d'un plan de maintenance efficace.

La simulation utilisant le modèle décrit précédemment dans le document délivre également des informations sur la fiabilité du système pendant la période de simulation qui est dans ce cas de 30 ans.

Le système basé sur l'architecture B (Fig. 5. 11) aura un temps d'arrêt estimé à 3 heures par an et donc 90 heures pendant les 30 années. Le nombre de fautes est de 16 en moyenne sur les 30 ans de la simulation.

Si nous prenons les intervalles de confiance à 99% pour ces valeurs, nous avons les chiffres suivants :

| Nom de l'indicateur | Valeur | IC 99% |
|---|---|---|
| Temps d'arrêt | 3 heures/an | ±8 minutes/an |
| Nombre de fautes | 16,7 | ±0,5 |

*Tableau 5.3 : Intervalles de confiance à 99% pour le temps d'arrêt et le nombre de fautes*

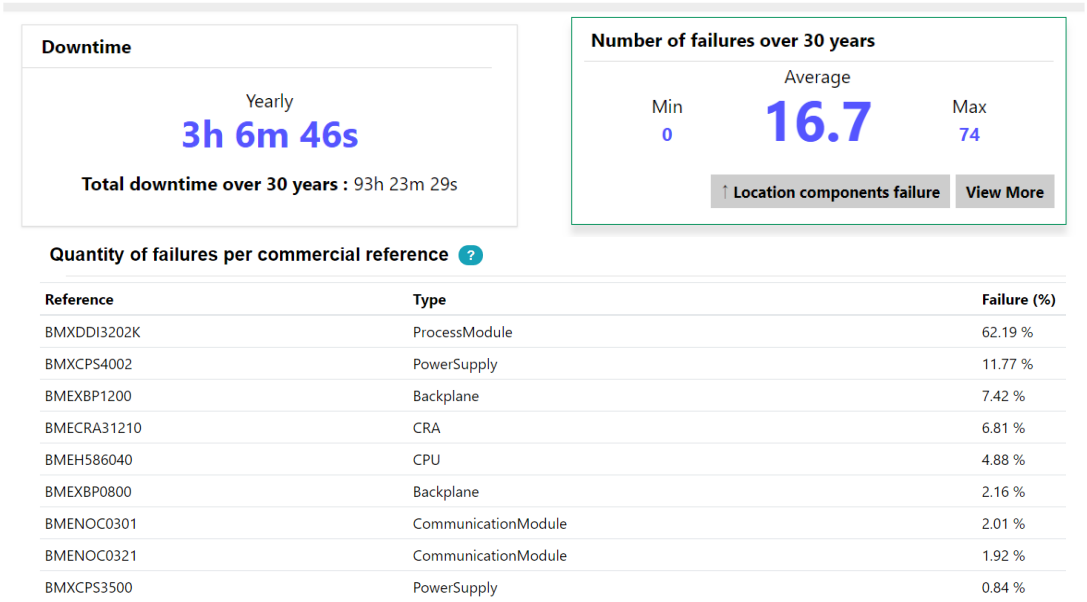



*Fig. 5. 11 : Temps d'arrêt et nombre de fautes estimés*

La simulation montre clairement que le module qui a le plus de fautes pendant cette période de temps est un module d'entrée de 32 voies, le BMXDDI3202K. Ce module représente environ 62% de fautes.

Le même constat peut être obtenu en se penchant sur le calcul des facteurs d'importance des composants constituant l'architecture. Pour des raisons de visibilité, nous avons calculé ici seulement les facteurs d'importance (Birnbaum, Importance critique et Amélioration potentielle) des cinq premiers composants étant tombés en panne que nous voyons dans la liste de la Fig. 5. 11. Le reste des composants ont des valeurs négligeables. Nous avons calculé ces facteurs à l'instant t=30 ans.

Ainsi, pour les cinq composants concernés, nous avons les chiffres suivants :

| Composant | Facteur de Birnbaum | Facteur d'importance critique | Facteur d'amélioration potentielle |
|---|---|---|---|
| BMXDDI3202K | 0,02 | 0,02 | 0,02 |
| BMXCPS4002 | $1{,}13.10^{-6}$ | $5{,}48.10^{-7}$ | $4{,}13.10^{-7}$ |
| BMEXBP1200 | $2{,}08.10^{-6}$ | $1{,}50.10^{-7}$ | $1{,}36.10^{-6}$ |
| BMECRA31210 | $1{,}80.10^{-6}$ | $1{,}22.10^{-6}$ | $1{,}08.10^{-6}$ |
| BMEH586040 | $1{,}33.10^{-6}$ | $7{,}54.10^{-7}$ | $6{,}20.10^{-7}$ |

*Tableau 5.4 : Facteurs d'importance des composants les plus critiques de l'architecture B*

Nous observons nettement que le module BMXDDI3202K est le plus problématique et que si nous améliorons la maintenance au niveau de ce module, la fiabilité de l'ensemble de l'architecture B sera grandement amélioré.

Le système intégrateur peut donc proposer une évolution de cette architecture (Fig. 5. 12) en intégrant un nouveau module d'entrées de 32 voies disposant d'un meilleur MTBF (BMXDDI3232) analyser le gain apporté.


*Fig. 5. 12 : Architecture B raffinée*

Une nouvelle simulation est activée pendant une durée de 30 ans pour analyser les gains apportés par cette nouvelle architecture.

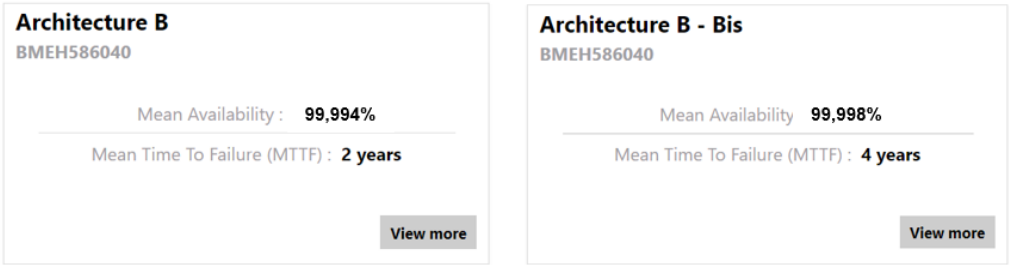



*Fig. 5. 13 : Résultats avec la simulation de la nouvelle architecture B*

Le résultat de la simulation montre clairement un gain au niveau de la disponibilité de l'architecture qui passe d'un taux de 99,994% à un taux de 99,998% (± $5.10^{-6}$ % avec un intervalle de confiance à 99%). La MTTF de l'architecture est également améliorée passant de 2 ans à 4 ans (± 21 jours avec un intervalle de confiance à 99%).

Ceci donne la possibilité d'atteindre donc plus facilement les seuils exigés dans le projet. Le gain le plus important se trouve d'ailleurs sur la fiabilité, comme indiqué dans la figure précédente, où lorsqu'on augmente la MTBF d'un composant, on diminue les chances de l'architecture à

tomber en panne en retardant l'apparition d'une défaillance. Ainsi, le déploiement des ressources dédiées à la maintenance du système peut être repoussé dans le temps et le stock de pièce de rechange à considérer peut être diminué. De même pour la fiabilité un gain important peut-être constaté comme indiqué dans la figure suivante.

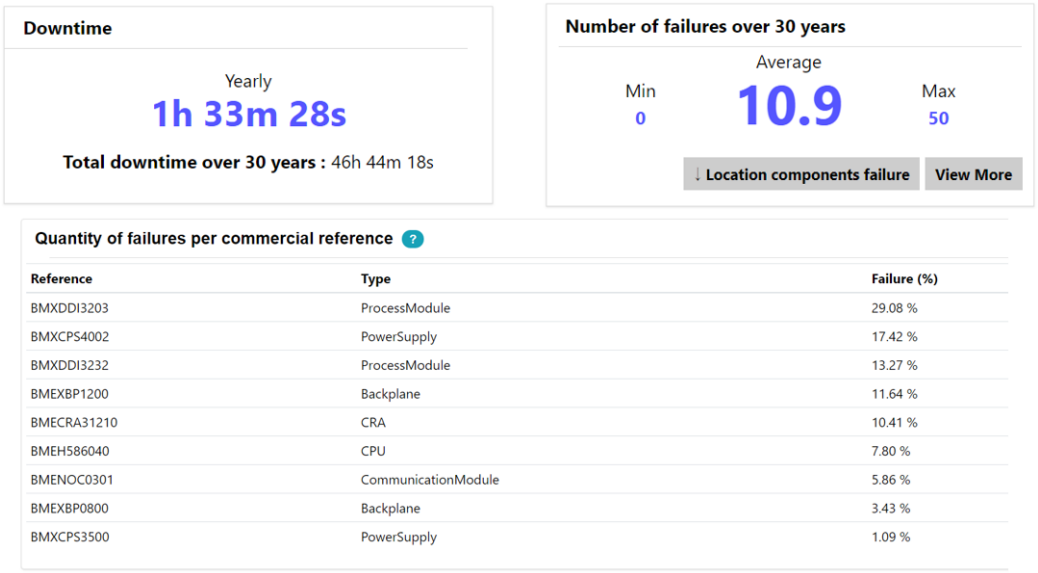



*Fig. 5. 14 : Nouveau temps d'arrêt et nombre de fautes estimés*

Le temps d'arrêt par an a été réduit de 3 heures à 1 heure 30 (± 8 minutes avec un intervalle de confiance à 99%), de même le nombre fautes a clairement diminué passant de 16,7 à 10,9 (± 0,5 avec un intervalle de confiance à 99%). Le nouveau module représente maintenant environ 29% des cas de fautes au lieu de plus de 60%.

Comme énoncé précédemment, un plan de maintenance peut également participer à l'amélioration de la disponibilité. En prévoyant un stock de pièces de rechange pour le module BMXDDI3232, le temps de réparation passe de 8 heures à 1 heure. Ce nouveau MTTR appliqué à ce module est utilisé pour effectuer une nouvelle simulation de cette architecture.

Le résultat de la nouvelle simulation montre un gain important sur la disponibilité de l'architecture B Bis qui devient maintenant un système à haute disponibilité (5-neufs) et qui répond pleinement aux exigences du client. Ce gain est également très intéressant sur le temps d'arrêt du système et le nombre de pannes estimé.

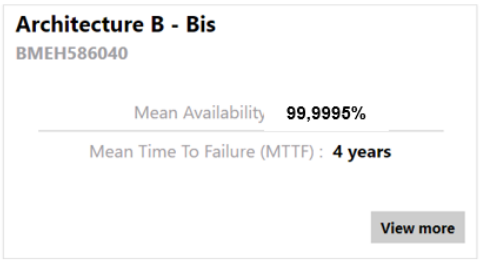



*Fig. 5. 15 : Disponibilité avec un MTTR à 1h*

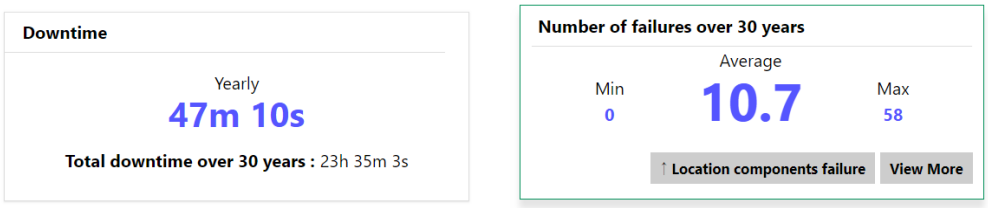



*Fig. 5. 16 : Temps d'arrêt et nombre de fautes estimés avec un MTTR à 1h*

Un temps d'arrêt de 47 minutes par an est estimé par la simulation ce qui représente une nouvelle amélioration par rapport à la précédente simulation et surtout une très nette amélioration en comparaison à la première estimation qui était de plus de 3 heures d'arrêt.

En conclusion, à l'aide de l'outil de simulation et du modèle associé, l'intégrateur système a été capable d'évaluer plusieurs scénarios d'architecture et de proposer à l'utilisateur final une architecture à haute disponibilité pour l'automatisation du four de la cimenterie en minimisant les couts.

### 5.4. Conclusion

Ce chapitre a présenté le prototype d'évaluation des performances fiabiliste des architectures de contrôle-commande en nous basant sur notre méthodologie explicitée dans le chapitre 4 de ce document.

L'utilisation de ce prototype sur un cas d'étude industriel fourni par Schneider Electric a permis de :

- Démontrer la faisabilité de notre méthodologie et de valider par extenso les modèles CPN intégrés au prototype.
- Mettre en évidence l'intérêt principal de notre proposition dont l'un des objectifs est la mise à disposition pour les forces de vente d'un outil leurs permettant d'évaluer les performances fiabilistes de leurs architectures. Ceci en limitant l'effort et les ressources consentis par les forces de vente pour évaluer une architecture puisque celui-ci est limité à la description sur une interface graphique de l'architectures et de ses paramètres,

En définitive, les résultats obtenus confortent les principes de notre proposition et constituent une base scientifique solide et pertinent pour la mise en production de notre prototype.

# Conclusion et perspectives

Dans le cadre de cette thèse, l'enjeu principal de nos travaux de recherche était de fournir une démarche outillée pour l'évaluation des performances de sûreté de fonctionnement (SdF) des architectures de contrôle-commande de Schneider Electric en phase d'avant-vente. Cette évaluation est capitale dans la course pour gagner des parts de marché ou des appels d'offres, en assurant de proposer des systèmes fiables et robustes face à l'occurrence de pannes sur les équipements de l'architecture, et en apportant des arguments techniques pertinents justifiant la configuration et les choix portés sur l'architecture proposée par les système intégrateurs (SI) pour un projet industriel en particulier.

C'est dans ce cadre que Schneider Electric a désiré se doter d'un outil d'aide à la décision reprenant cette évaluation des performances de SdF. La mise en place de cette évaluation s'était articulée autour de trois grands axes :

1- Une définition formelle d'une architecture de contrôle-commande prenant en compte toutes les notions en rapport avec la SdF et la description du comportement fonctionnel et dysfonctionnel du système en phase d'avant-vente.
2- La définition d'une liste d'indicateurs SdF pertinente dans le contexte où l'évaluation sera réalisée, pour interpréter correctement les performances fiabilistes de l'architecture.
3- La mise en place de l'évaluation des performances de SdF d'une architecture de contrôle-commande à partir de sa définition formelle et des indicateurs SdF sélectionnés.

En complément de ces trois axes, l'évaluation devait respecter plusieurs critères. Elle devait être réalisée simplement par les forces de ventes de Schneider Electric, les SI, donc exiger une expertise moindre dans le domaine de la SdF. Elle devait être rapide et réutilisable pour évaluer toutes les architectures possibles pour un même projet. Enfin, elle devait prendre en compte le contexte dynamique dans lequel évolue le système ainsi que le contexte incertain que nous retrouvons dans la phase d'avant-vente.

Nous avons pu répondre à ces questions et considérer les contraintes énoncées dans chaque chapitre de ce mémoire. Ces réponses ont été le fruit d'une réflexion portée sur les besoins industriels de Schneider Electric et d'une recherche approfondie dans ce que la littérature scientifique pouvait nous apporter.

Le formalisme de modélisation pour décrire les architectures de contrôle-commande est défini par les réseaux de Petri stochastiques colorés et temporisés (CPN pour *Coloured Petri Nets*). Ces CPN permettent en effet de prendre en compte le comportement dynamique du système, mais également les aspects stochastiques propres à la SdF comme la description des défaillances sur les équipements qui est un événement aléatoire. La prise en compte du temps, la description modulaire et hiérarchique du système, en adéquation avec les architectures de Schneider Electric et ce qui est donné dans la description informelle, et qui sont proposées par les CPN, nous ont conforté dans le choix de ces derniers pour modéliser les architectures de contrôle-commande.

Une bibliothèque de modèles de familles de composants sous la forme de modules hiérarchisés dans le modèle CPN du système a été développée. La composition de ces modèles permet de former le modèle constructeur décrivant une topologie type qui est ensuite instanciée et paramétrée en fonction de la description informelle de l'architecture.

La liste des indicateurs SdF retenus a été constituée dans le respect des besoins industriels réclamés par nos SI et par les retours d'expérience sur les exigences que nous pouvons rencontrer dans les spécifications d'un projet industriel présenté en phase d'avant-vente.

Nous avons donc retenu trois composantes parmi les quatre grands aspects de la SdF (Disponibilité, Fiabilité, Maintenabilité, Sécurité), à savoir la disponibilité, la fiabilité et la maintenabilité et leurs métriques liés au temps moyen de fonctionnement du système, son temps moyen de réparation ou encore son temps moyen d'indisponibilité.

Nous complétons cette énumération avec les facteurs d'importance qui permettent de montrer l'impact des composants de l'architecture sur les performances de SdF du système global. Trois facteurs ont été retenus parmi une liste. Le facteur de Birnbaum et le facteur d'importance critique, dont le deuxième découle directement du premier. Ces deux facteurs ont permis de quantifier la criticité des composants sur le système, à quel point un tel composant peut potentiellement dégrader la fiabilité du système, par rapport à ses caractéristiques fiabilistes (sa fiabilité intrinsèque principalement) et sa place dans la structure du système. Le dernier facteur choisi a été le facteur d'amélioration potentielle qui quantifie l'impact du composant sur la fiabilité du système en estimant l'augmentation potentielle de cette fiabilité si nous améliorons le composant pour le rapprocher d'un composant dit parfait, c'est-à-dire ne connaissant aucune panne au cours de sa durée de vie.

Le dernier point s'est porté sur l'approche pour la mise en place de l'évaluation à partir du formalisme de modélisation choisi et les indicateurs de SdF retenus. Nous avons sélectionné une approche par simulation, la simulation de Monte Carlo. Cette approche se fait grâce à la génération d'observateurs dans le modèle pour récupérer les données de simulation en vue de récupérer les performances fiabilistes de l'architecture de contrôle-commande.

Le développement d'un prototype implémentant cette méthode d'évaluation, et son application sur un cas d'étude, un projet de cimenterie, permettant d'illustrer son intérêt dans un cadre industriel que nous pouvons rencontrer en avant-vente, ont permis de mettre en évidence notre contribution pour les ingénieurs en avant-vente de Schneider Electric.

En outre, il est important de noter que les calculs inhérents à l'évaluation des performances SdF d'une architecture est réalisée de manière totalement transparente pour les utilisateurs de l'outil d'aide à la décision. Une interface graphique ergonomique permet à ces derniers d'intégrer la description informelle de l'architecture, de définir les paramètres d'entrée d'utilisation connus, et les paramètres de simulation, le nombre d'histoires et le temps de simulation pour une histoire, généralement la durée d'exploitation envisagée pour l'architecture.

Ces informations sont automatiquement appelées dans le prototype grâce aux fonctions d'instanciation et de paramétrage du modèle constructeur pour lancer la simulation de Monte-Carlo sur le modèle et fournir aux SI une estimation statistique des performances de SdF de l'architecture.

À la suite de ces travaux de recherche, plusieurs perspectives sont envisagées, toujours dans une optique de continuité pour pouvoir évaluer les performances de SdF des architectures de contrôle-commande que propose Schneider Electric en phase d'avant-vente.

La première perspective à court terme concerne l'industrialisation complète de la méthode d'évaluation en vue de répondre aux besoins industriels de Schneider Electric. Cela implique, entre autres :

- un enrichissement et une consolidation des bibliothèques de composants, voire l'ajout de nouveaux composants ou la prise en compte de nouvelles caractéristiques d'un composant déjà existant,
- un travail d'optimisation des temps de calculs lors des simulations de Monte Carlo : le modèle constructeur peut être adapté pour privilégier des options de modélisation moins couteuses en temps de calcul tout en conservant sa sémantique, notamment via une analyse du code des fonctions ML portées par les transitions,
- un travail d'optimisation de la collecte des données de simulation à partir des observateurs afin de simplifier le processus de traitement post-simulation en portant une réflexion sur la qualité des informations récupérées et des moniteurs utilisés,
- une phase de validation sur un panel d'architectures issues de projets industriels préalablement développés par Schneider Electric sera nécessaire pour augmenter la précision des résultats et vérifier de la pertinence des résultats affichés à l'utilisateur de l'outil d'aide à la décision,
- la recueil d'un solide retour d'expérience relatif aux phénomènes de de dégradation et de vieillissement des équipements ; en effet, faute de données suffisantes, les profils usuels de dégradation, tels que le processus Gamma, pourtant intégrés dans nos modèles n'ont pu être déployés. pour les appliquer dans le calcul du taux de défaillance sont déjà intégrés dans le modèle.

La deuxième famille de perspectives concerne l'enrichissement du périmètre de notre proposition.

La prise en compte des aspects de sécurité fonctionnelle et des équipements dédiés aux fonctions de mise en sécurité du procédé industriel devrait permettre d'intégrer le calcul d'un niveau de sécurité de l'architecture de contrôle-commande grâce aux classes SIL (*Safety Integrity Level*. L'évaluation des performances fiabilistes des architectures ne serait donc plus limitée à des systèmes fonctionnels de contrôle-commande mais élargie aux systèmes de sécurité.

Par ailleurs, le calcul des facteurs d'importance en régime dynamique, c'est-à-dire en fonction du temps et non plus à la fin de la durée d'exploitation du système, ouvrirait des perspectives en termes notamment de maintenance. En effet, les valeurs des facteurs d'importance variant dans le temps, la criticité apportée par chacun des équipements constituant l'architecture évolue également, et pourrait guider l'adaptation du stock des pièces de rechange par exemple, et par extension du plan de maintenance.

Enfin, la dernière perspective concerne la prise en compte des fonctions d'automatismes supportés par les architectures de contrôle-commande. En effet, dans certains cas en avant-vente,

les clients peuvent demander les performances de SdF de l'architecture proposée, non pas sur la topologie du système mais sur la disponibilité d'une fonction de fonction de contrôle répartie sur plusieurs équipements.

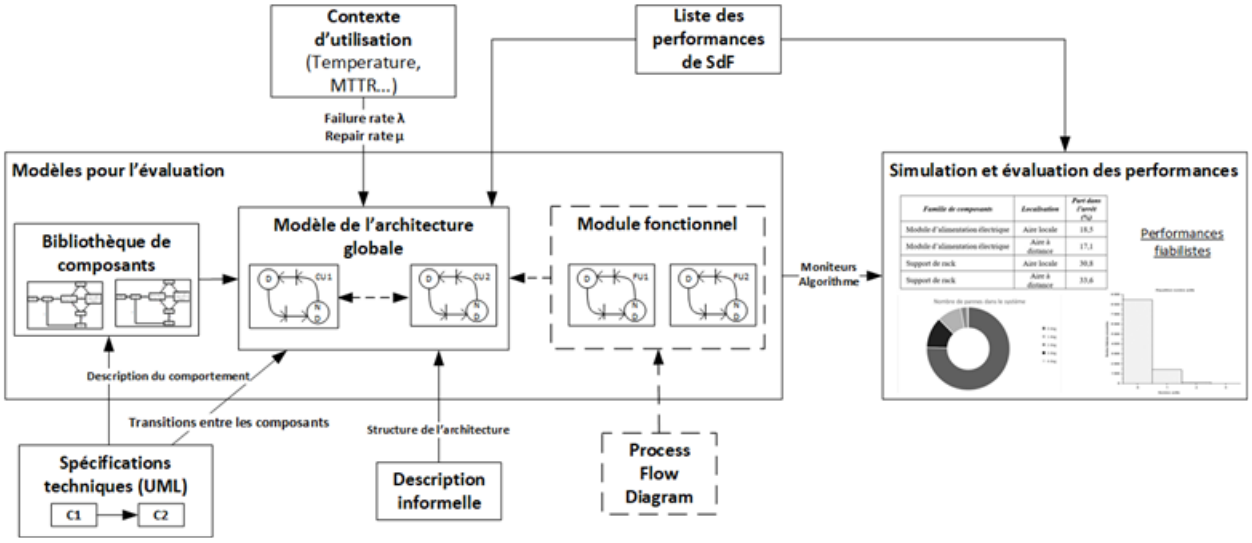



*Fig. C. 1 : Perspectives sur la méthode d'évaluation des performances de SdF d'une architecture de contrôle-commande*

Cela exigerait probablement d'enrichir le cadre de modélisation proposé au chapitre 4 par un ensemble de modèles à visée fonctionnelle pouvant être définis à partir d'un description de type PFD (*Process Flow Diagram*). L'évaluation des performances s'opérerait dès lors sur un couplage entre le modèle d'architecture du chapitre 4 et ces nouveaux modèles (cf. Fig. C. 1).

# Références bibliographiques


Alur, R., & Dill, D. (1994). A theory of timed automata. *Theoretical computer science, 126*(2), 183-235.

Aubry, J., & Brînzei, N. (2015). *Systems Dependability Assessment : Modeling with graphs and finite state automata.* Wiley-ISTE.

Aubry, J., & Brinzei, N. (2016). Calcul direct de la fiabilité d'un système par son graphe ordonné. *20e Congrès de maitrise des risques et de surete de fonctionnement.* Saint-Malo, France.

Aubry, J.-F., Brînzei, N., & Mazouni, M.-H. (2016). *Systems Dependability Assessment : Benefits of Petri Net Models.* ISTE Ltd and John Wiley & Sons Inc.

Babykina, G., Brînzei, N., Aubry, J.-F., & Deleuze, G. (2016). Modeling and simulation of a controlled steam generator in the context of dynamic reliability using a Stochastic Hybrid Automation. *Reliability Engineering & System Safety*, 115-136.

Babykina, G., Brînzei, N., Aubry, J.-F., & Perez Castaneda, G. (2011). Reliability assessment for complex systems operating in dynamic environment. *Annual Conference of the European Safety and Reliability Association, ESREL 2011*. Troyes.

Barger, P. (2003). *Evaluation et validation de la fiabilité et de la disponibilité des systèmes d'automatisation à intelligence distribuée, en phase dynamique*. Nancy: Thèse de l'Université Henri Poincaré.

Behrends, E. (2000). *Introduction to Markov chains* (Vol. 228). Braunschweig/Wiesbaden: Vieweg.

Belkacem, L., Simeu-Abazi, Z., Gascard, E., & Messaoud, H. (2017). Diagnostic and prognostic of hybrid dynamic systems: Modeling and RUL evaluation for two maintenance policies. *Reliability Engineering & System Safety, 164*, 98-109.

Bertsche, B. (2008). *Reliability in Automotive and Mechanical Engineering.* Berlin: Springer-Verlag Berlin Heidelberg.

Birnbaum, Z. (1968). *On the Importance of Different Components in a Multicomponent System.*

Birnbaum, Z., Esary, J., & Saunders, S. (1961). Multi-component systems and Structures and Their Reliability. *Technometrics, 3:1*, pp. 55-77.

Birolini, A. (2004). *Reliability Engineering : Theory and Practice.* Springer-Verlag Berlin Heidelberg.

Borgonovo, E. (2007). A new uncertainty importance measure. *Reliability Engineering & System Safety, 92*, 771-784.

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

Bouissou, M. (2006). Détermination efficace de scenarii minimaux de défaillance pour des systèmes séquentiels.

Bouissou, M. (2008). BDMP (Boolean logic Driven Markov Processes) ® as an alternative to Event Trees. *17th European Safety and Reliability Conference,ESREL 2008.* Valence, Espagne.

Boyer, G., Brînzei, N., Camerini, J., Ndiaye, M., & Pétin, J.-F. (2020). Dynamic Dependability Assessment of Industrial Control Systems using colored stochastic Petri nets. *30th European Safety and Reliability Conference and 15th Probabilitic Safety Assessment and Management Conference, ESREL 2020 PSAM 15.* Venise, Italie.

Boyer, G., Brînzei, N., Camerini, J., Ndiaye, M., & Pétin, J.-F. (2020). Évaluation dynamique d'indicateurs de sûreté de fonctionnement pour des architectures de contrôle-commande par le biais de réseaux de Petri colorés stochastiques. *22e Congrès de Maîtrise des Risques et Sûreté de Fonctionnement*, (pp. 111-119). Le Havre.

Boyer, G., Brînzei, N., Do, P., & Pétin, J.-F. (2017). Reliability modelling and assessment by joint consideration of Petri nets and gamma deterioration processes. *2017 2nd International Conference on System Reliability and Safety (ICSRS)* (pp. 57-61). Milan, Italie: IEEE.

Boyer, G., Pétin, J.-F., Brînzei, N., Camerini, J., & Ndiaye, M. (2019). Toward generation of dependability assessment models for industrial control system. *International Conference on Information and Digital Technologies (IDT)*, (pp. 50-59). Zilina, Slovaquie.

Bray, T., Paoli, J., & Sperberg-McQueen, C. (1997). Extensible Markup Language (XML). *World Wide Web J.*, 27-66. Récupéré sur W3C Recommendation: https://www.w3.org/TR/REC-xml/

Broy, P. (2014). *Evaluation de la sûreté de systèmes dynamiques hybrides complexes. Application aux systèmes hydrauliques*. Troyes: Thèse de l'Université de Troyes.

Cassandras, C., & Lafortune, S. (2009). *Introduction to discrete event systems.* Springer Science & Business Media.

CEI. (2010). CEI 61508 : Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité. Genève: Commission Electrotechnique Internationale (CEI).

Cepin, M. (2011). *Reliability Block Diagram. In : Assessment of Power System Reliability.* Londres: Springer.

Cepin, M., & Mavko, B. (2002). A dynamic fault tree. *Reliability Engineering & System Safety, 75*(1), 83-91.

Chabot, J.-L., Dutuit, Y., & Rauzy, A. (2001). De l'usage de la simulation de Monte-Carlo couplée aux réseaux de Petri en sûreté de fonctionnement. *3e Conférence Francophone de Modelisation et simulation "Conception, Analyse et Gestion des systèmes industriels" MOSIM'01*. Troyes.

Chevalier, M., Vinuesa, C., Buchsbaum, L., Folleau, C., & Thomas, P. (2020). Vers un transcripteur automatique de réseaux électriques industriels en réseaux de Petri. *22e Congrès de maîtrise des risques et surete de fonctionnement*, (pp. 453-458). Le Havre.

Cocozza-Thivent, C. (1997). *Processus stochastiques et fiabilité des systèmes.* Berlin: Springer-Verlag.

Cressent, R., David, P., Idasiak, V., & Kratz, F. (2013). Designing the database for a reliability aware Model-Based System Engineering process. *Reliability Engineering & System Safety, 111*, 171-182.

David, P. (2009). *Contribution à l'analyse de sûreté de fonctionnement des systèmes complexes en phase de conception : application à l'évaluation des missions d'un réseau de capteurs de présence humaine*. Orléans: Thèse de l'Université d'Orléans.

David, P., Idasiak, V., & Kratz, F. (2008). Towards a better interaction between design and dependability analysis: FMEA derived from UML/SysML models. *ESREL 2008 and 17th SRA-EUROPE annual conference*, (pp. 8-17). Valence, Espagne.

David, P., Idasiak, V., & Kratz, F. (2010). Reliability study of complex physical systems using SysML. *Reliability Engineering & System Safety, 95*, 431-450.

David, R., & Alla, H. (1992). *Du Grafcet aux réseaux de Petri.* Hermès.

Deleuze, G., & Brînzei, N. (2015). Modeling digital I&C systems for PRA with coloured Petri nets. *9th International Conference on Nuclear Plant Instrumentation, Control & Human-Machine Interface Technologies (NPIC & HMIT 2015).* Charlotte, North Carolina, USA.

Deleuze, G., Brînzei, N., & Gérard, L. (2015). Common cause failure parameters estimation with coloured Petri nets. *PSA'15 International Topical Meeting on Probabilistic Safety Assessment and Analysis.* Sun Valley, Idaho, USA.

Dubi, A. (2000). *Monte Carlo Applications in systems engineering.* Londres: John Wiley & Sons.

Duflot, N. (2007). *Les mesures d'importances fiabilistes issues des études probabilistes de sûreté nucléaire : contrôle des incertitudes et nouvelles applications pour l'aide à la décision*. Troyes: Thèse de l'Université de Troyes.

Duflot, N., Bérenguer, C., Dieulle, L., & Vasseur, D. (2009). A min cut-set-wise truncation procedure for importance measures computation in probabilistic safety assessment. *Reliability Engineering & System Safety, 94, Issue 11*, 1827-1837.

Dugan, J., Bavuso, S., & Boyd, M. (1990). Fault trees and sequence dependencies. *Annual Proceedings on Reliability and Maintainability Symposium* (pp. 286-293). Los Angeles: IEEE.

Dugan, J., Trivedi, K., Geist, R., & Nicola, V. (1984). *Extended Stochastic Petri Nets : Applications and Analysis.* Wisconsin: Wisconsin Univ-Madison Motor Behavior.

Duroeulx, M. (2020). *Évaluation de la fiabilité des systèmes modélisés par arbres de défaillances grâce aux techniques de satisfabilité*. Nancy: Thèse de l'Université de Lorraine.

Dutuit, Y., & Rauzy, A. (2005). Approximate estimation of system reliability via fault trees. *Reliability Engineering & System Safety*, 163-172.

Fehling, R. (1991). A concept of hierarchical Petri nets with building blocks. *International Conference on Application and Theory of Petri Nets* (pp. 148-168). Berlin, Heidelberg: Springer.

Fussell, J., Aber, E., & Rahl, R. (1976). On the Quantitative Analysis of Priority-AND Failure Logic. *IEEE Transactions on Reliability*(p 324-326), pp. 324-326.

Gascard, E., & Simeu-Abazi, Z. (2018). Quantitative Analysis of Dynamic Fault Trees by means of Monte Carlo Simulations: Event-Driven Simulation Approach. *Reliability Engineering & System Safety, 180*, 487-504.

Huyen, C., Zamaï, E., & Ngoc, Y. (2012). Modeling the distribution grid by Petri Nets for reconfiguration the power system. *Advanced Materials Research*, 2561-2565.

Ionescu, D.-R. (2016). *Évaluation quantitative de séquences d'événements en sûreté de fonctionnement à l'aide de la théorie des langages probabilistes*. Nancy: Thèse de l'Université de Lorraine.

Jensen, K. (1981). Coloured Petri nets and the invariant-method. *Theoretical computer science, 14*(3), 317-336.

Jensen, K. (1994). An introduction to the Theoritical Aspects of Coloured Petri Nets. *Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems)*, (pp. 230-272).

Jensen, K., & Kristensen, L. (2009). *Coloured Petri Nets : Modelling and Validation of Concurrent Systems.* Berlin: Springer-Verlag Berlin Heidelberg.

Jensen, K., & Kristensen, L. (2015). Colored Petri nets : a graphical language for formal modeling and validation of concurrent systems. *Communications of the ACM, 58*(6), 61-70.

Jensen, K., Kristensen, L., & Wells, L. (2007). Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, (pp. 213-254).

Johnsson, C. (2004). How and where can it be applied. *ISA Expo*, (pp. 1-10). Houston.

Kaufmann, A., Grouchko, K., & Cruon, R. (1975). *Evaluation de la fiabilité des systèmes cohérents et non cohérents par la théorie des graphes.* Paris: Editions Masson et Cie.

Kim, C., & Lee, H. (1992). A Monte Carlo simulation algorithm for finding MTBF. *IEEE Transactions on Reliability*, 193-195.

Kuo, W., & Zhu, X. (2012). *Importance Measures in Reliability, Risk, and Optimization : Principles and Applications.* John Wiley & Sons.

Kuo, W., & Zhu, X. (2012). Some Recent Advances on Importance Measures in Reliability. *IEEE Transactions on Reliability, 61*(2), 344-360.

Kvassay, M., Zaitseva, E., & Levashenko, V. (2017). Importance analysis of multi-state systems based on tools of logical differential calculus. *Reliability Engineering and System Safety, 165*, pp. 302-316.

Laprie, J. (1992). *Dependability : Basic concepts and terminology.* Vienne, Autriche: Springer.

Laprie, J.-C. (1995). *Guide de la Sûreté de fonctionnement.* Toulouse: Editions Cépaduès.

Laprie, J.-C. (2004). Sûreté de fonctionnement des systèmes : concepts de base et terminologie. *Revue de l'Electricité et de l'Electronique (REE)*, 95-105.

Limnios, N. (2005). *Arbres de défaillances* (éd. 2e édition revue et augmentée). Paris: Hermes science publ. Lavoisier.

Marsan, A., Gianni Conte, M., & Balbo, G. (1984). A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems (TOCS), 2*(2), 93-122.

Marsan, M., & Chiola, G. (1986). On Petri nets with deterministic and exponentially distributed firing times. *European Workshop on Applications and Theory in Petri Nets*, (pp. 132-145).

Marsan, M., Balbo, G., Conte, G., Donatelli, S., & Franceschinis, G. (1998). Modelling with generalized stochastic Petri nets. *ACM SIGMETRICS performance evaluation review, 26*(2), 61-70.

Milner, R., Tofte, M., Harper, R., & MacQueen, D. (1997). *The Definition of Standard ML.* Cambridge, MA, USA: The MIT Press.

Mkhida, A., Barger, P., Thiriet, J.-M., & Aubry, J.-F. (2005). Influence of the control strategy choice on the safety level of a distributed control system. *The European Safety and Reliability Conference, ESREL 2005.* Gdansk-Gdynia, Pologne.

Mkhida, A., Thiriet, J.-M., & Aubry, J.-F. (2008). Toward an Intelligent Distributed Safety Instrumented Systems: Dependability Evaluation. *17th IFAC World Congress (IFAC WC 2008)*, (pp. 3586-3591). Seoul, Corée du Sud.

ML, S. (1998). *Standard ML '97*. Consulté le 2020, sur http://www.smlnj.org/sml97.html

Muller, P.-A., & Gaertner, N. (2000). *Modélisation objet avec UML.* Paris: Eyrolles.

Narushima, H. (1974). Principle of inclusion-exclusion on semilattices. *Journal of Combinatorial Theory*, 196-203.

Narushima, H. (1982). Principle of inclusion-exclusion on partially ordered sets. *Discrete Mathematics*, pp. 243-250.

Natkin, S. (1980). *Stochastic Petri Nets and their application for the evaluation of computer systems*. Paris: Thèse de docteur ingénieur CNAM.

Ndiaye, M. (2017). *Évaluation de performance d'architecture de contrôle-commande en réseau dans un contexte incertain d'avant-vente*. Nancy: Thèse de l'Université de Lorraine.

Niel, E., & Craye, E. (2002). *Maitrise des risques et surete de fonctionnement des systèmes de production. Productique : information, commande, communication.* Lavoisier.

Object Management Group. (2015). *About the Unified Modeling Language specification version 2.5.* Récupéré sur https://www.omg.org/spec/UML/2.5

Pagès, A., & Gondran, M. (1980). *Fiabilité des systèmes.* Paris: Eyrolles.

Papadopoulos, C., & Yeung, H. (2001). Uncertainty estimation and Monte Carlo simulation method. *Flow Measurement and Instrumentation, 12*(4), 291-298.

Paulson, L. (2010). *ML for the Working Programmer.* Cambridge: Cambridge University Press.

Perez Castaneda, G. (2009). *Évaluation par simulation de la sûreté de fonctionnement de systèmes en contexte dynamique hybride*. Nancy: Thèse de l'Institut National Polytechnique de Lorraine (INPL).

Perez Castaneda, G., Aubry, J.-F., & Brînzei, N. (2011). Stochastic hybrid automata model for dynamic reliability assessment. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 28-41.

Peterson, J. (1980). A note on colored Petri nets. *Inf. Process. Lett., 11*(1), 40-43.

Pinna, B., Babykina, G., Brînzei, N., & Pétin, J.-F. (2013). Using Coloured Petri nets for integrated reliability and safety evaluations. *4th IFAC Workshop on Dependable Control of Discrete Systems DCDS'13*, (pp. 19-24). York, Royaume-Uni.

Piriou, P.-Y. (2015). *Contribution à l'analyse de sûreté de fonctionnement basée sur les modèles des systèmes dynamiques, réparables et reconfigurables*. Paris: Thèse de l'Université de Paris-Saclay.

Piriou, P.-Y., Faure, J.-M., & Lesage, J.-J. (2017). Generalized Boolean logic Driven Markov Processes: a powerful modeling framework for Model-Based Safety Analysis of dynamic repairable and reconfigurable systems. *Reliability Engineering & System Safety, 163*, 57-68.

Rausand, M., & Hoyland, A. (2003). *System reliability theory : models, statistical methods, and applications.*

Rauzy, A. (1993). New algorithms for fault trees analysis. *Reliability Engineering & System Safety*, 203-211.

Rauzy, A. (2011). Sequence Algebra, Sequence Decision Diagrams and Dynamic Fault Trees. *Reliability Engineering & System Safety, 96*(7), 785-792.

Rieker, T., Zeiler, P., & Bertsche, B. (2015). Reliability Modelling of a Hybrid Car Drive System with advanced Petri Nets considering dependencies and aging. *European Safety and Reliability Conference ESREL 2015*, (pp. 1689-1697). Zürich, Suisse.

Rubino, G., & Sericola, B. (2014). *Markov chains and dependability theory*. Cambridge University Press.

Ruijters, E., & Stoelinga, M. (2015). Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review, 15-16*, 29-62.

Schneider Electric. (2018). How can I select Ecostruxure Plant Hybrid reference architectures. Carros: System Technical Note.

Schneider Electric. (2019). How can I increase the availability of a control system. System Technical Note.

Scholten. (2007). *Integrating ISA-88 and ISA-95.* Rosmalen: Ordina ISA-95 & MES competence centre.

Signoret, J.-P., Dutuit, Y., Cacheux, P.-J., Folleau, C., Collas, S., & Thomas, P. (2013). Make your Petri nets understandable : Reliability Block Diagrams driven Petri nets. *Reliability Engineering & System Safety, 113*, 61-75.

Simeu-Abazi, Z., Gascard, E., & Sidqi, Y. (2015). Quantitative analysis of Dynamic Fault Tree by probabilistic approach. *25th European Safety and Reliability Conference, ESREL 2015.* Zürich, Suisse.

Suiphon, B., Simeu-Abazi, Z., & Gascard, E. (2014). Premier pas vers le diagnostic de défaillances par exploitation d'un modèle SysML. *19e Congrès de Maitrise des Risques et Sûreté de Fonctionnement.* Dijon.

Thiriet, J.-M., Conrard, B., & Robert, M. (2005). Distributed system dependability evaluation: a case study on a pilot thermal process. *Reliability Engineering and System Safety, 88*, 109-119.

Vatn, J. (1992). Finding minimal cut sets in a fault tree. *Reliability Engineering & System Safety, 36, Issue 1*, 59-62.

Verma, A. K., Ajit, S., & Karanki, D. (2010). *Reliability and Safety Engineering.* Londres: Springer-Verlag London.

Villemeur, A. (1988). *Sûreté de fonctionnement des systèmes industriels : fiabilité, facteurs humains, informatisation.* Paris: Eyrolles.

Wang, J. (1998). *Timed Petri Nets : Theory and application.* New York: Springer Science & Business Media.

Zaitseva, E., Levashenko, V., & Kostolny, J. (2015). Application of logical differential calculus and binary decision diagram in importance analysis. *Eksploatacja i Niezawodnosc - Maintenance and Reliability*, pp. 379-388.

Zio, E. (2009). Reliability engineering: Old problems and new challenges. *Reliability Engineering & System Safety, 94*, 125-141.

Zio, E. (2013). *The Monte Carlo Simulation Method for System Reliability and Risk Analysis.* Londres: Springer-Verlag.

# Résumé/Abstract

Ce mémoire, réalisé dans le cadre d'une convention CIFRE avec la société Schneider Electric, porte sur l'évaluation sur l'évaluation d'indicateurs de sûreté de fonctionnement des architectures de contrôle-commande. Le problème industriel posé est un problème d'aide au dimensionnement des architectures en réponse à des exigences de performances. Les problèmes scientifiques sont relatifs, d'une part, à la génération automatique des modèles support à l'évaluation pour un ensemble d'architectures dans la mesure où leur construction manuelle ne permet pas de faire face à la multiplicité des architectures dans des délais contraints, et, d'autre part, à la prise en compte des phénomènes de dégradation dans les modèles dysfonctionnels des architectures. Le choix du formalisme de modélisation s'est porté sur les réseaux de Petri colorés et temporisés (CPN) qui permet notamment la hiérarchisation des modèles et la prise en compte de phénomènes temporisés dont les durées peuvent suivre des distributions de probabilité. Notre contribution se présente sous la forme d'une méthodologie permettant la construction automatique d'un modèle CPN d'architecture par instanciation d'un ensemble de composants élémentaires et de la définition d'observateurs génériques couvrant les besoins exprimés et qui permettent de déterminer les indicateurs de sûreté de fonctionnement tels que la fiabilité, la disponibilité du système, son MTTF, le nombre de défaillances mais aussi des facteurs d'importance permettant d'orienter les choix des concepteurs en phase d'élaboration en réponse à un appel d'offre.

**Mots-clés** : évaluation de performances, architecture de contrôle-commande, réseaux de Petri colorés stochastiques, sûreté de fonctionnement, facteurs d'importance

This PhD dissertation, supported by a CIFRE agreement with the Schneider Electric company, deals with the dependability performances assessment of industrial control system (ICS) architectures. The industrial challenge is to provide a tool able to guide on the sizing of ICS architectures in terms of performances. The scientific challenge in another hand is double. Firstly, we must generate automatically the models which are the core of the performance evaluation for a set of ICS architecture, since their manual modelling is challenging based on the wide variety of architecture to model and the limited amount of time and resources. Secondly, we must consider the degradation phenomena in those architectures. The selected modelling formalism is the coloured and timed Petri nets (CPN) which allows to represent model hierarchically, to handle random time delays which help on characterizing the components lifetime. Our contribution is on the development of a method for automatically generate a CPN model of ICS by instantiating a set of elementary components and defining generic observers, covering dependability indicators such as system reliability, availability, MTTF, number of failures and importance measures, and allowing to guide the choices of system integrators during pre-sales stage.

**Keywords**: performances assessment, industrial control system, coloured stochastic Petri Nets, dependability, importance measures