N° d'ordre :

Thèse

présentée devant
l'UNIVERSITÉ DE BRETAGNE SUD

pour obtenir le titre de

Docteur

spécialité : *Automatique et Informatique Industrielle*

Proposition d'une approche haut niveau pour la conception, l'analyse et l'implantation des systèmes reconfigurables

par
Florent Frizon de Lamotte

le 23/11/2006 devant la commission d'Examen

Composition du jury

Rapporteurs

M. E. CRAYE	Professeur à l'Ecole Centrale de Lille
M. Y. TRINQUET	Professeur à l'IUT de Nantes

Examinateurs

M. P. BERRUET	Maître de Conférences à l'Université de Bretagne Sud (Co-Encadrant)
M. J.-L. PHILIPPE	Professeur à l'Université de Bretagne Sud (Directeur de thèse)
M. A. ROSSI	Maître de Conférences à l'Université de Bretagne Sud
M. E. ZAMAI	Maître de Conférences à l'Institut National Polytechnique de Grenoble

Thèse préparée au Laboratoire d'Electronique des Systèmes Temps Réel
LESTER CNRS FRE 2734 – Université de Bretagne Sud

Remerciements

Mes remerciements vont tout d'abord à mes encadrants. Pascal tout d'abord, avec qui j'ai eu le plaisir de travailler au jour le jour. Jean-Luc ensuite qui a été présent aux bons moments, mis le holà quand on s'embourbait et m'a permis d'appliquer mes travaux a l'électronique. André, lui, a posé des questions pertinentes dès son arrivée dans l'équipe. Il a probablement été le plus difficile à convaincre, mais la rigueur des travaux a d'autant plus été améliorée. Je n'oublie pas non plus Thierry, avec qui j'ai commencé ces travaux et qui a donné l'impulsion initiale.

J'étends mes remerciements à tout le laboratoire et en particulier à l'équipe "Systèmes Reconfigurables". Grâce à tous, j'ai passé trois années formidables. Une mention spéciale pour Aïssam et Jean-Louis qui ont débuté leur thèse en même temps que moi sur des sujets complémentaires et avec lesquels ce fût un plaisir de travailler et de discuter.

Les membres de mon jury ont particulièrement contribué à la réussite de cette thèse. Je remercie donc Messieurs Etienne Craye et Yvon Trinquet qui m'ont fait l'honneur et le plaisir d'être mes rapporteurs. Ainsi que Monsieur Eric Zamaï qui en plus de figurer en tant qu'examinateur, m'offre un post-doctorat dans les montagnes.

D'autres personnes m'ont permis de me lancer dans cette thèse, il s'agit de mes encadrants de DEA, Jérôme et David, avec lesquels j'aurais aimé pouvoir continuer la route.

Les travaux présentés dans cette thèse n'auraient vraisemblablement pas eu la même contenance s'ils n'avaient pas été outillés. Je remercie donc l'équipe ATLAS du LINA et de L'INRIA pour m'avoir permis d'utiliser l'outil ATL depuis les premières versions. Je remercie plus particulièrement Frédéric qui est pour moi plus qu'un ami et qui m'a formé à cette technologie, nous permettant d'entamer une collaboration malgré nos parcours différents.

Je souhaite, pour finir, exprimer une pensée pour Yann Bozec, qui a été un formidable camarade pendant près de six mois et dont je ne m'explique pas le geste. Sois certain, Yann, que tes travaux ne resteront pas vains.

Table des matières

Abréviations

CIM Computer Integrated Manufacturing
DSL Domain Specific Language
GAO Graphe d'Accessibilité Opérationnelle
IOA Input Output Association
OCL Object Constraint Language
QdS Qualité de Service
RMS Reconfigurable Manufacturing Systems
SAP Système Automatisé de Production
SFC Sequential Function Chart [IEC, 2003]
ST Structured Text [IEC, 2003]
UML Unified Modeling Language

Introduction générale

"Any customer can have a car painted any colour that he wants so long as it is black."

(H. Ford, 1909)

Le choix unique de la couleur noire pour la célèbre *Ford T* de Henry Ford était imposé par l'impossibilité, pour la chaîne de production dédiée à la fabrication de cette voiture, de s'adapter pour permettre la peinture des voitures d'une couleur différente du noir. L'évolution des systèmes de production a heureusement permis d'offrir des alternatives en adaptant la production aux choix des clients, augmentant ainsi leur satisfaction.

La reconfiguration nous entoure dans notre quotidien. Au sein d'une maison, il est fréquent de modifier l'utilisation des pièces. Un bureau peut par exemple être transformé, de façon temporaire, en chambre d'amis, pour accueillir une connaissance le plus chaleureusement possible. Si plus d'amis sont invités alors le canapé du salon peut aussi être utilisé. Des modifications à plus long terme sont aussi envisageables afin d'adapter l'usage de la maison à des nouveaux besoins ou de nouvelles envies. Des murs peuvent alors être ajoutés ou supprimés sous condition pour créer de nouvelles pièces. De nouveaux meubles peuvent être achetés pour permettre un meilleur rangement. L'équipement domestique subit aussi des modifications et évolue au gré des besoins. Les chaînes hi-fi offrent généralement la possibilité d'ajouter ou de remplacer une platine pour lire un nouveau support disponible sur le marché. Le passage du 33 tours au disque compact en est un exemple. L'amplificateur peut être remplacé par une version plus récente supportant des fonctions "home-cinema" par exemple, ces nouvelles fonctions apportent une meilleure expérience audiovisuelle à l'utilisateur.

Du côté de l'informatique personnelle, la configuration d'un ordinateur individuel peut être totalement remodelée. Le processeur peut être remplacé, tout comme le disque dur. De nouveaux périphériques comme une imprimante ou un scanner peuvent être ajoutés. De même, des cartes d'accélération graphique permettent à l'utilisateur d'augmenter le réalisme du dernier jeu à la mode. De manière générale, la reconfiguration d'un PC peut concerner à la fois le matériel et le logiciel, qui doit s'adapter aux nouvelles configurations matérielles ou tout simplement être mis à jour, ce qui constitue en soi une reconfiguration de l'utilisation qui est faite de l'ordinateur.

Les robots constitués d'atomes pouvant s'assembler de manières différentes au gré des contraintes imposées par leur mission et leur environnement constituent le "Graal" des systèmes reconfigurables[1]. La structure physique du système ainsi que son organisation logique (communication entre les atomes) peut être entièrement modifiée. De tels robots s'adaptent parfaitement au terrain sur lequel ils évoluent et à la mission qu'ils accomplissent. Une configuration en pont peut par exemple être adoptée pour franchir un obstacle. De plus, la coopération entre les atomes doit permettre, alors que chaque atome dispose de ressources de calcul limitées, d'obtenir une organisation de ces ressources permettant d'assurer le fonctionnement du robot.

[1] Voir par exemple `http://www-valoria.univ-ubs.fr/Dominique.Duhaut/maam/`

La reconfiguration apparaît donc comme un moyen de faire évoluer en douceur la constitution d'un système. Elle permet de satisfaire au mieux les besoins en terme de qualité de service, synonyme dans le cas d'une maison de qualité de vie et de convivialité, pour un coût acceptable.

La reconfiguration telle que nous la traitons dans ce mémoire est une reconfiguration automatique des divers éléments composant un système dont les entités peuvent remplir de multiples fonctions et dont les liens (flux, dépendance) entre ces celles-ci sont modifiables dynamiquement.

L'étude de la reconfiguration dans la conception de tels systèmes connaît actuellement un intérêt grandissant quel que soit le domaine considéré (contrôle, électronique, informatique, production). La prise en compte de la capacité de reconfiguration du système permet d'étendre considérablement son cadre d'utilisation. En effet, la reconfiguration permet de modifier l'utilisation des éléments constituant le système, afin de l'adapter à la demande exprimée en terme de Qualité de Service (QdS) ainsi qu'aux aléas. La reconfiguration permet aussi de trouver un compromis entre les performances attendues du système qui nécessitent une spécialisation de ses éléments, et le coût de ce même système qui peut diminuer en choisissant des ressources généralistes, bien que moins performantes pour un besoin donné. En électronique, les circuits de type FPGA (Field Programmable Gate Array) peuvent ainsi être reconfigurés au cours du fonctionnement du système et répondre ponctuellement, mais de manière optimisée, à une mission. Un circuit dédié ne pourrait être utilisé que pour répondre à cette mission et un microprocesseur, qui peut réaliser tous les traitements, pourrait ne pas convenir en raison de ses performances ou de sa consommation.

Deux domaines nous ont plus particulièrement intéressés dans le cadre des travaux réalisés pendant cette thèse. Il s'agit du domaine des Systèmes Automatisés de Production (SAP) et de celui des systèmes électroniques peuvant par exemple servir de support à une architecture de contrôle/commande. Ces deux domaines disposent de la caractéristique commune de manipuler des flux. Dans le premier cas il s'agit d'un flux de produits qui doit être géré alors que dans le second il s'agit d'un flux d'informations.

Dans le cadre des systèmes de production, deux axes sont principalement considérés pour la reconfiguration. Le premier concerne l'utilisation de moyens de production évolutifs, permettant de s'adapter aux demandes du marché à moindre coût en remplaçant un équipement ou un module d'un équipement. Le deuxième axe vise à utiliser au mieux les flexibilités physiques offertes par le système pour réaliser une mission. Il ne s'agit alors pas de faire évoluer les moyens dont dispose le système mais de profiter au mieux de ceux-ci pour continuer à fournir un service après une défaillance ou bien pour s'adapter à une grande variété de gammes de produits réalisées sur le système.

Les systèmes électroniques sont traditionnellement conçus pour réaliser un service particulier. Avec l'évolution des technologies, la conception de systèmes embarqués offrant de multiples fonctionnalités aux utilisateurs et respectant un grand nombre de standards peut aujourd'hui être envisagée. Dans le cadre des systèmes embarqués, un compromis doit être fait entre consommation, performance et coût du système. L'utilisation de la reconfiguration permet de garantir des performances optimales en utilisant des accélérateurs matériels, tout en limitant la surface de silicium occupée par le circuit puisque la même surface peut être utilisée pour implémenter des accélérateurs matériels différents en fonction des besoins. Plusieurs versions des tâches mettant en œuvre une implémentation matérielle ou logicielle et proposant des niveaux de qualité de service différents sont alors proposées et peuvent être combinées pour constituer une configuration adéquate. La reconfiguration en électronique est rendue possible grâce à dès circuits reprogrammables dynamiquement tels que les FPGA.

A un niveau de granularité plus élevé, sur des systèmes automatisés comme par exemple le démonstrateur ferroviaire présenté au chapitre 6, l'utilisation d'une électronique répartie sur plusieurs nœuds, reliés entre eux par des bus ou des réseaux est courante. L'architecture de commande est alors organisée de manière hiérarchique ou distribuée. L'utilisation d'une architecture électronique reconfigurable pour la conduite de tels systèmes constitue un avantage dans la mesure où les nœuds constituant l'électronique de commande ou les réseaux reliant ceux-ci sont susceptibles de tomber en panne, au même titre que le procédé. Un langage commun aux différents domaines entrant en jeu au sein des systèmes reconfigurables apporterait une généricité permettant d'aborder sous un même point de vue la reconfiguration dans chacun d'entre eux.

Contexte

Les travaux présentés dans ce mémoire se sont déroulés pendant trois ans au sein de l'équipe Systèmes Reconfigurables du Laboratoire d'Electronique des Systèmes TEmps Réel (LESTER) de l'Université de Bretagne Sud.

Ils ont été initiés dans le cadre de la convergence entre les deux projets conduits par l'équipe Systèmes Reconfigurables. Ces projets traitent d'une part de la reconfiguration des systèmes électroniques et d'autre part de la reconfiguration des systèmes de production. Ce rapprochement est né du constat que pour gérer la complexité atteinte par les systèmes électroniques, une approche "système" est désormais nécessaire et qu'au niveau des systèmes de production, la complexité et les exigences portent à intégrer la notion de composant, bien connue des électroniciens et de descendre au niveau de l'électronique pour peu que l'on s'intéresse aux problématiques relevant de l'implantation.

Ces travaux s'inscrivent dans la continuité de ceux réalisés par J.S. Mouchard [Mouchard, 2002] et sont complétés par les thèses d'A. Belabbas et J.L. Lallican qui se sont déroulées en parallèle. La première traite de l'utilisation de la reconfiguration dans un contexte d'assistance technique au handicap. La seconde étend la démarche de conception par composants introduite dans la thèse de J.S. Mouchard, les activités d'implantation qui y sont définies constituent l'articulation avec les travaux présentés dans cette thèse.

Problématique

L'occurrence d'aléas au sein d'un système complexe peut avoir des conséquences aussi bien anecdotiques que catastrophiques. Ces aléas peuvent indifféremment être des défaillances internes au système que des modifications des contraintes extérieures qui lui sont appliquées. La redondance est souvent utilisée pour faire face aux aléas, mais elle est coûteuse et parfois inefficace pour des incidents imprévus. L'utilisation de la reconfiguration nécessite la conception d'une architecture flexible et tolérante, qui n'est pas définie pour réaliser un traitement de manière unique. La reconfiguration et la flexibilité sont donc indissociables puisque l'utilisation des flexibilités du système passe par les mécanismes de reconfiguration.

Ce constat est valable pour une grande variété de systèmes. Au sein des systèmes de production manufacturière, par exemple, l'utilisation de machines flexibles permet de réaliser des pièces différentes après modification du programme. Pour les systèmes électroniques, l'approche de l'échéance de la fin de vie de la batterie peut conduire à une révision à la baisse des fonctionnalités offertes afin d'augmenter la durée de vie du système.

La volonté d'exploiter les capacités d'un système en terme de reconfiguration nécessite une prise en compte de ces capacités tout au long du développement. De plus, on remarque que des caractéristiques propres aux systèmes reconfigurables sont présentes quel que soit le domaine (électronique, systèmes de production, informatique). Ces caractéristiques propres doivent être décrites par un langage commun de haut-niveau, sur lequel se base l'activité de conception et qui sert de point de départ aux activités d'analyse et d'implantation du système reconfigurable.

Les différentes attentes du concepteur concernant les capacités du système dans une configuration donnée peuvent être d'abord estimées par la définition de métriques sur la description. Ensuite, si des outils ad hoc sont disponibles, des analyses plus fines peuvent être lancées.

Objectifs

L'objectif de cette thèse est de proposer une approche système permettant l'étude et la mise en œuvre des systèmes reconfigurables, pouvant s'appliquer à la fois aux systèmes de production et aux systèmes électroniques. A cette fin, la proposition d'un langage de haut niveau doit être assez générique pour permettre la représentation de ces systèmes provenant de différents domaines. A partir de ce langage, différentes analyses ayant trait à la performance, au coût, à la flexibilité, à la robustesse ou à la cohérence seront proposées, pour aider le concepteur et l'utilisateur à tirer le meilleur parti de la reconfiguration.

L'implantation et l'exploitation des systèmes ainsi modélisés constitue un second objectif. La réalisation de cet objectif passe par la proposition de passerelles avec les outils industriels permettant l'implantation du programme de commande. Ces outils sont généralement basés sur des normes industrielles telles que l'IEC 61131-3 pour les langages automates ou ISO C pour le langage C.

La réalisation d'outils d'analyse pour illustrer l'approche ainsi que d'une plate-forme de démonstration constitue un autre objectif. Il nous semble en effet nécessaire d'évaluer les propositions qui seront faites à travers des exemples réalistes et au moins sur un système réel.

Plan

Ce document est organisé en trois parties. Chacune de ces trois parties est composée de deux chapitres.

La première partie présente la problématique des systèmes reconfigurables et répond à la question "Pourquoi une représentation de haut niveau ?". La variété des systèmes reconfigurables existants ainsi que les points communs entre ces différents systèmes font l'objet du premier chapitre qui se termine par des définitions associées au concept de reconfiguration. Le chapitre 2 présente les bases de la définition d'un langage haut niveau pour la description des systèmes reconfigurables dans les domaines des systèmes de production et de l'électronique. Des approches utilisées dans les différents domaines sont décrites ainsi que des langages de haut niveau. Ce chapitre introduit aussi les concepts liés à l'ingénierie des modèles qui a une place importante dans les travaux.

La deuxième partie de ce mémoire répond à la question "Quelle représentation de haut niveau adopter ?". Le chapitre 3 introduit le langage de représentation proposé. Ce langage, appelé DeSyRe pour Description des Systèmes Reconfigurables, est présenté sous plusieurs formes : une description informelle, une formalisation mathématique et une modélisation utilisant l'ingénierie dirigée par les modèles. Deux exemples, un en électronique et un dans le domaine des

systèmes de production sont ensuite présentés et montrent comment DeSyRe peut être utilisé dans les deux domaines. Le chapitre 4 décrit l'utilisation de ce langage pour les activités de conception. Les besoins des concepteurs y sont d'abord présentés et un cadre d'analyse est défini pour y apporter des réponses. Les métriques proposées à cette fin sont ensuite définies, puis illustrées sur des exemples.

La dernière partie répond à la question "Comment exploiter les capacités de reconfiguration du système ?". Le chapitre 5 présente le fonctionnement d'un système reconfigurable. Au cours de cette description, la place de la configuration du système est clairement située par rapport aux fonctions du contrôle/commande. Le processus de reconfiguration, plaçant le système dans la configuration souhaitée est ensuite décrit. Finalement, l'activité d'implantation à partir de l'architecture, de la configuration, et du modèle de haut-niveau est présentée. Le chapitre 6 décrit la mise en place d'un système reconfigurable, pour le "démonstrateur ferroviaire" développé au cours de cette thèse. Les différents concepts, décrits dans les chapitres précédents sont alors déployés sur une plate-forme réelle.

Première partie

Problématique des systèmes reconfigurables

Chapitre 1

Présentation de systèmes reconfigurables

La notion de reconfiguration s'est vue, depuis de début des années 1990, déclinée de manière variée dans différents domaines pour lesquels elle est utilisée. En effet, ces différents domaines se sont appropriés le terme reconfiguration, de l'électronique aux systèmes de production manufacturière. De fait, la reconfiguration à été traitée à des niveaux d'abstraction différents et dans différents types de systèmes.

L'objectif de ce chapitre est de caractériser ce qu'est un système reconfigurable. Il n'existe, en effet, pas de définition précise et universelle caractérisant ce type de systèmes. Ce chapitre vise donc à définir à quoi correspondent pour nous, les termes *configuration* et *reconfiguration* à partir de l'utilisation qui est faite de ceux-ci dans la littérature.

A cet effet, la première section de ce chapitre présente des exemples de systèmes reconfigurables appartenant à différents domaines afin d'avoir un aperçu du spectre recouvert par cette catégorie de systèmes. La deuxième partie identifie des points communs entre les différents systèmes étudiés dans le but de ressortir des paramètres caractérisant un système reconfigurable. Les définitions concernant les notions de configuration, de reconfiguration et de système reconfigurable sont ensuite introduites dans une dernière partie.

1.1 Exemples de systèmes reconfigurables

La notion de reconfiguration est utilisée dans des domaines aussi différents que l'automatique, l'électronique, l'informatique, les communications ou les systèmes de production manufacturiers. L'utilisation qui est faite de la reconfiguration est détaillée ici pour différents domaines. Un exemple est donné afin d'illustrer comment est utilisé ce concept.

1.1.1 Contrôle reconfigurable

Le contrôle reconfigurable consiste à utiliser les redondances au niveau des moyens de contrôle ou de mesure pour pallier la défaillance ou l'utilisation dans des conditions sous-optimales de l'un d'entre eux. Plusieurs exemples de systèmes utilisant du contrôle reconfigurable sont présentés dans la littérature [Wills and et al., 2001; Crow and Rushby, 1991]. Au sein de ces applications, c'est le système de contrôle qui est considéré comme un système reconfigurable.

La mise en œuvre de tels systèmes peut être réalisée avec des systèmes d'une complexité plus ou moins grande. L'élément reconfigurable peut être placé dans la chaîne de contrôle, comme c'est le cas dans l'exemple présenté ensuite. Les paramètres de cet élément pouvant être ajustés dynamiquement. Une architecture électronique et informatique complexe, comme

celle présentée dans [Wills and et al., 2001] peut aussi être utilisée. Celle-ci s'occupe alors de la gestion des équipements et de l'algorithme de contrôle. L'utilisation d'une telle architecture est plus lourde mais permet de facilement interfacer de nouveaux éléments à la chaîne de commande, de plus, différents outils de simulation ou de conception sont directement inclus dans la plate-forme.

Présentation d'un exemple

L'exemple introduit ici est le système de contrôle de l'avion spatial X-33 présenté dans [Cotting and Burken, 2001] et représenté à la figure 1.1. Ce système permet de modifier la commande des volets afin de permettre un vol correct de l'avion malgré la défaillance de l'un d'eux. Le mélangeur reconfigurable de l'avion X-33 est le premier système de commande reconfigurable embarqué dans un avion.

FIG. 1.1 – L'avion X-33

Description du système reconfigurable

Le système reconfigurable présent dans l'avion est le mélangeur. Comme le montre la figure 1.2, le mélangeur reconfigurable se place entre les trois commandes provenant du pilote (roll/pitch/yaw) placées sur la gauche et les différent volets que l'on retrouve sur la droite. Des gains, représentés par des boîtes sont affectés à la liaison aux commandes puis additionnés ou soustraits afin d'obtenir les ordres envoyés au volet. Ainsi, le mélangeur interprète les ordres donnés par le pilote et envoie une commande aux différents volets conforme à la configuration actuelle.

Anatomie d'une configuration

Une configuration du mélangeur correspond aux gains affectés à chaque commande afin de piloter les sorties.

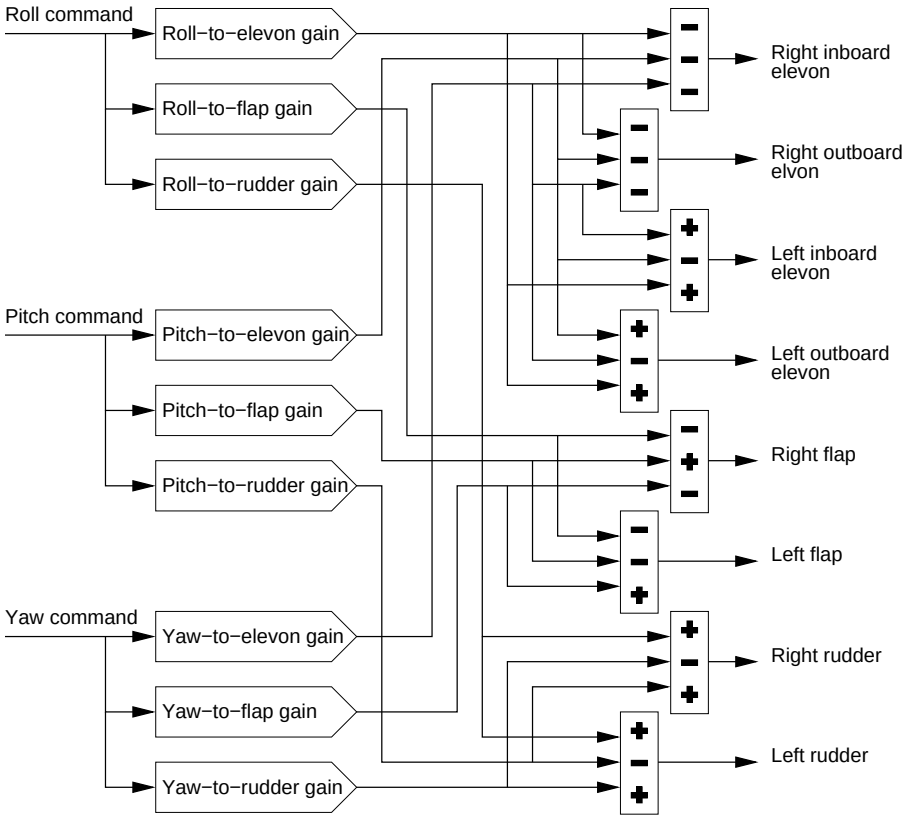


FIG. 1.2 – Le “reconfigurable mixer” [Cotting and Burken, 2001]

Objectif de la reconfiguration

L’objectif de la reconfiguration est, dans ce cas de l’avion X-33, le prolongement du fonctionnement du système malgré la défaillance d’un de ses éléments.

Mise en œuvre de la reconfiguration

La reconfiguration du système de contrôle de l’avion est réalisée en changeant les paramètres du mélangeur. Les actionneurs utilisés fournissent une information d’erreur s’ils sont bloqués ainsi que leur position. Connaissant l’état de ces différents actionneurs les paramètres du mélangeur sont modifiés conformément à une table dont les valeurs sont déterminées hors-ligne.

1.1.2 Systèmes de communication

La reconfiguration a aussi été utilisée pour optimiser le fonctionnement des systèmes de communication multimédia. Dans le cas, par exemple, d’une liaison radio au cours de laquelle les émetteurs et récepteurs peuvent se déplacer, il peut s’avérer nécessaire de changer ou de modifier les paramètres de certains éléments de la chaîne de communication afin de garantir une qualité de service (QdS) [Jin and Nahrstedt, 2004] acceptable.

Présentation d’un exemple

Le framework DJINN, présenté dans [Mitchell et al., 1998] permet de mettre en place des systèmes de communication pour gérer des applications multi-média. Un tel système pourrait

par exemple être utilisé dans un studio mobile pour une émission télévisée en direct.

Identification du système reconfigurable

Le framework DJINN permet de bâtir des systèmes de communication reconfigurables. Toute application développée à partir de ce framework peut donc être considérée comme étant un système reconfigurable.

La modélisation du système est réalisée à l'aide de composants. Trois catégories de composants sont distinguées. Les *model components* permettent de modéliser l'application et ses contraintes. Les *peer components* sont des composants actifs produisant, traitant ou consommant des données. La structuration du système est portée par les *composite components* qui peuvent encapsuler des composants des deux types précédents, les composants de haut niveau ainsi élaborés servent de base à la construction des applications. La figure 1.3 présente un exemple d'application décrite à l'aide de ces composants.

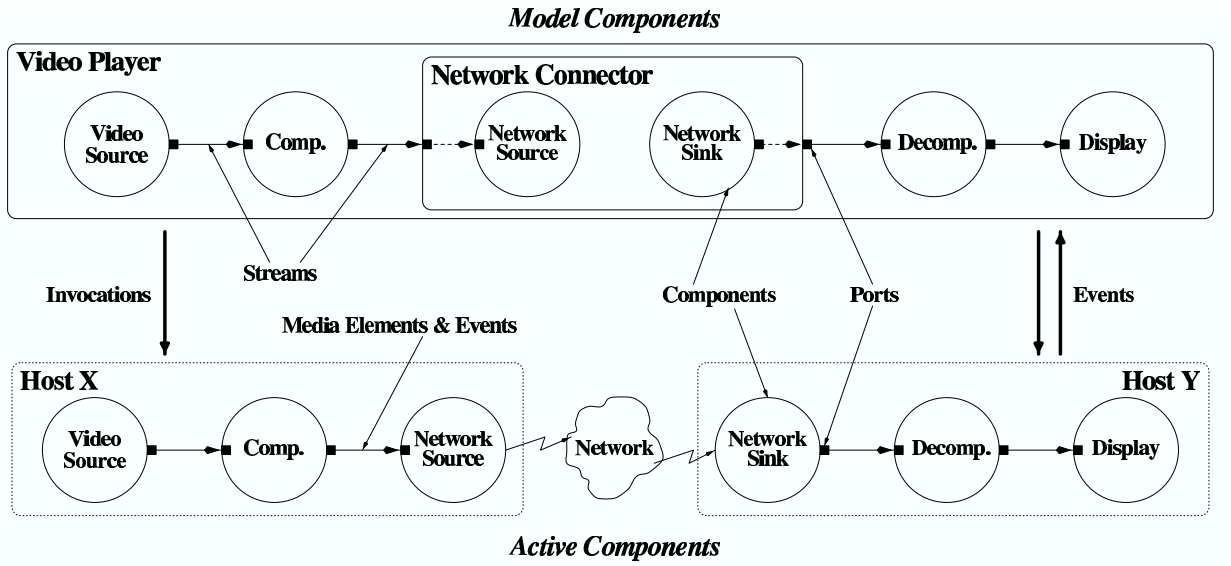


FIG. 1.3 – Les composants utilisés dans DJINN [Mitchell et al., 1998]

Anatomie d'une configuration

Une configuration est décrite par un modèle du système réalisé avec le framework DJINN. Le modèle de l'application lie les différents composants constituant la chaîne de diffusion. L'élément principal caractérisant une configuration par rapport à une autre est le chemin des données au sein du système.

Objectif de la reconfiguration

L'objectif de la reconfiguration, dans le cas des systèmes de communication, est de garantir la qualité de service de la communication alors que les capacités du système évoluent suivant la disponibilité d'un protocole dans un lieu géographique donné.

D'autre part, le processus de reconfiguration lui-même ne doit pas dégrader la QdS pendant sa phase active (passage d'une configuration à l'autre).

Mise en œuvre de la reconfiguration

La reconfiguration du système est réalisée en remplaçant les composants. Il s'agit donc d'une modification de l'architecture du système.

Pour cela, le passage d'une configuration à l'autre doit se dérouler en deux étapes soumises à des contraintes de qualité et des contraintes temporelles comme le montre la figure 1.4. Ce schéma présente les différentes étapes de la reconfiguration, le timing qui doit être suivi ainsi que la détérioration de la qualité subie pendant le changement. Ainsi pendant la première phase, les nouveaux composants sont mis en place, une partie des ressources est nécessaire pour leur construction ce qui peut entraîner une chute de la QdS. Une phase d'intégration pendant laquelle les nouveaux composants sont progressivement allumés est ensuite réalisée, ce qui entraîne une nouvelle dégradation de la QdS, en effet l'ancienne configuration et la nouvelle cœxistent. Une fois la nouvelle configuration atteinte, le système retrouve le niveau de QdS qu'il avait avant la reconfiguration.

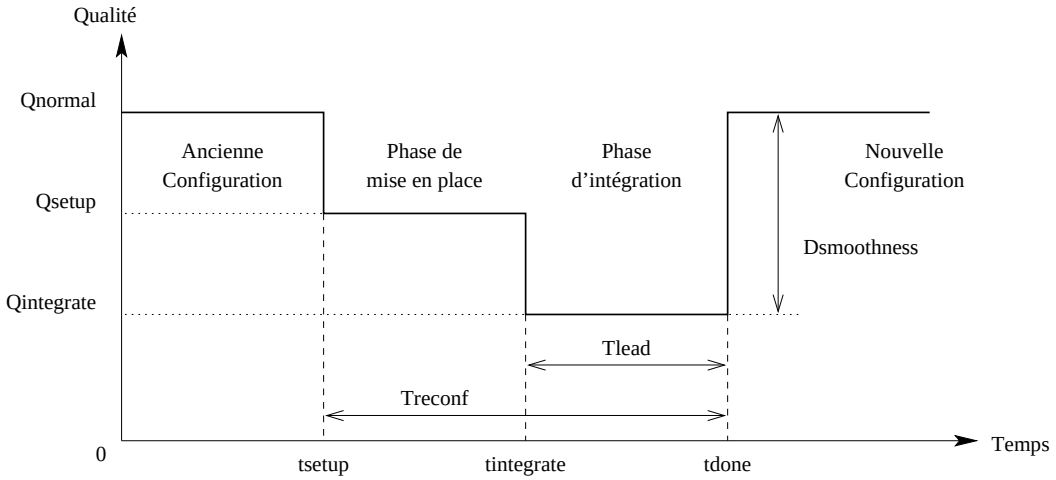


FIG. 1.4 – Exemple de reconfiguration [Mitchell et al., 1998]

1.1.3 Systèmes électroniques

La reconfiguration au sein des systèmes électroniques à été rendue possible par l'apparition des circuits reprogrammables que sont les FPGA (Field Programmable Gate Array) introduits par la société Xilinx [Xilinx, 2006] au milieu des années 1980. Un FPGA peut être décrit comme une multitude de cellules élémentaires (un FPGA peut être composé de plusieurs millions de ces cellules) fournissant quelques fonctions logiques ainsi qu'un peu de mémoire, ces cellules sont connectées entre elles par un réseau d'interconnexions programmable formant une grille. L'utilisation de ces FPGA permet de disposer d'unités de calcul dont les fonctionnalités peuvent changer suivant les besoins au cours du fonctionnement du système.

La reconfiguration se retrouve à plusieurs niveaux de granularité. Pour un grain fin, une configuration est décrite comme une organisation des portes logiques du système alors qu'à un grain plus élevé, des opérateurs et les flux de données sont manipulés.

Présentation d'un exemple

L'exemple présenté est issu du projet Epicure [Auguin et al., 2003] et concerne la mise en place d'unités de traitement reconfigurables (RPU : Reconfigurable Processing Unit). Ces unités

de traitement peuvent, suivant les besoins, réaliser différentes fonctions dédiées. Les fonctions exécutées correspondent à une partie d'un traitement réalisé par le système complet.

Identification du système reconfigurable

Le système reconfigurable est le RPU ainsi que son interface avec le processeur générique (interface ICURE).

Anatomie d'une configuration

Une configuration correspond aux fonctions utilisées dans les RPU à un instant donné. Si le système complet et les fonctions qu'il réalise sont considérés, il s'agit d'une reconfiguration gros-grain. Par contre, si les configurations sont décrites sous la forme d'un agencement de portes logiques élémentaires, la description d'une configuration est réalisée à un grain fin.

Objectif de la reconfiguration

La reconfiguration dans le cas d'une telle architecture vise à améliorer les performances du système en utilisant des accélérateurs matériels (plus performants pour une tâche donnée que les processeurs généralistes) tout en limitant la surface silicium occupée par ces accélérateurs puisqu'ils sont élaborés à partir des mêmes cellules matérielles.

Mise en œuvre de la reconfiguration

La reconfiguration est gérée par un contrôleur de reconfiguration qui charge les fonctions matérielles dans l'unité reconfigurable lorsque le système en a besoin. Le chargement d'une nouvelle fonction dans le processeur reconfigurable est demandé par le processeur, quand il en a besoin.

1.1.4 Informatique

La notion de reconfiguration est couramment utilisée dans les systèmes informatiques complexes. Les deux aspects d'un système informatique (logiciel ou matériel) sont reconfigurables. Le concept de configuration se retrouve en effet au niveau du logiciel, lorsque celui-ci est construit de manière modulaire. Le système informatique complet constitue lui-même un système reconfigurable dans lequel les aspects de reconfiguration matériels et logiciels sont enchevêtrés. Au niveau matériel, la mise sous-tension ou hors-tension d'une ressource de calcul peut par exemple être envisagée.

Présentation d'un exemple

Un exemple de système informatique utilisant la reconfiguration est donné dans [Welch and Purtilo, 1997]. Dans cet exemple sont traitées à la fois la reconfiguration de l'architecture matérielle du système et celle de l'architecture logicielle. Il s'agit d'un environnement de simulation des champs de bataille pour l'armée américaine utilisant un système informatique distribué sur plusieurs ordinateurs. L'environnement de simulation appelé CVE (Collaborative Virtual Environment) est construit de manière modulaire, une partie des modules sont appelés VE (Virtual Environment), les autres sont des utilitaires liés à la simulation. [Welch and Purtilo, 1997] présente un module utilitaire appelé Bullpen qui permet de gérer la reconfiguration du CVE.

Identification du système reconfigurable

Le système reconfigurable est le CVE, dont les différents modules peuvent être chargés, déchargés ou déplacés au cours du fonctionnement du système. Le module Bullpen gérant la reconfiguration fait aussi partie du système reconfigurable.

Anatomie d'une configuration

Dans le cadre de cet exemple, une configuration correspond au découpage de l'application sur les différentes machines disponibles ainsi que le lien entre ces machines. Plusieurs découpages sont possibles, tous les bâtiments peuvent par exemple être placés sur un même serveur, les terrains sur un deuxième et les objets mobiles sur un troisième serveur. Mais le découpage de l'application peut tout aussi bien être réalisé suivant le lieu géographique des éléments.

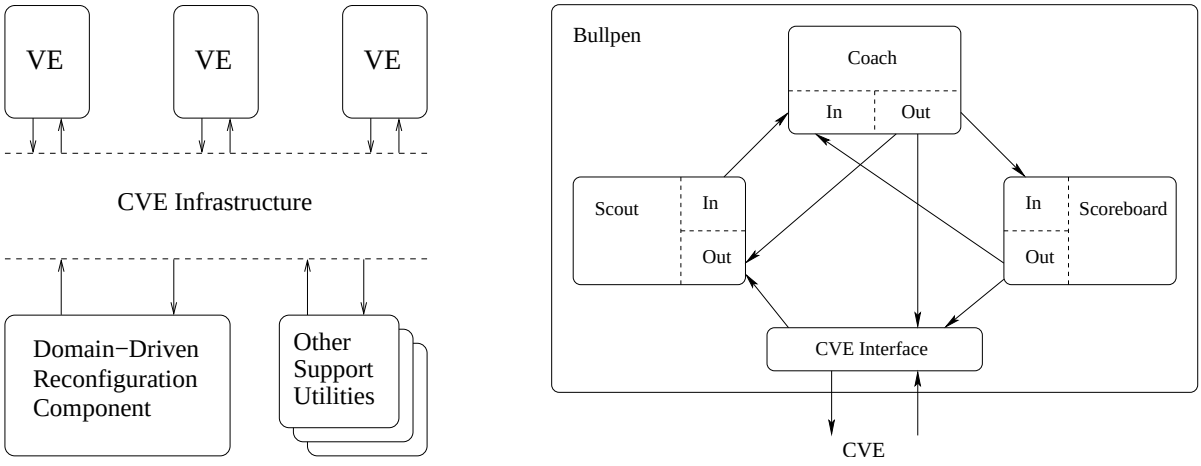
Objectif de la reconfiguration

Deux scénarios nécessitant une reconfiguration sont proposés dans [Welch and Purtilo, 1997]. Le premier survient lorsqu'une machine tombe en panne. Il est alors nécessaire que cette panne aie le moins d'effet possible sur la poursuite de la simulation.

Une deuxième utilisation de la reconfiguration permet d'assurer l'équilibre de la charge des différentes machines pendant l'évolution de la simulation, il peut en effet être intéressant de déplacer le code simulant un pont sur une machine dédiée lorsqu'un bataillon se dirige vers ce pont.

Mise en œuvre de la reconfiguration

L'environnement de simulation, présenté à la figure 1.5 est découpé de manière modulaire. L'infrastructure supporte la connexion de différents composants qui peuvent aussi bien être des environnements virtuels que des utilitaires assurant la gestion du système.



(a) L'environnement de simulation. (b) L'architecture de Bullpen.

FIG. 1.5 – Bullpen [Welch and Purtilo, 1997]

Le module Bullpen gérant la reconfiguration, est branché sur l'infrastructure du simulateur. Ce composant est constitué de trois agents : le Scout, le Coach et le Scoreboard. Le Scout est chargé de surveiller le système et tient à jour le Scoreboard qui reproduit l'état de ce même

système. Le Coach prend les décisions de reconfiguration, il décide de la nouvelle configuration à mettre en place ainsi que de la façon de l'obtenir.

1.1.5 Systèmes automatisés de production (SAP)

Les systèmes de production doivent faire face à des changements rapides de gammes pour suivre la demande. De constitution complexe, ils doivent aussi pouvoir continuer de fonctionner malgré la défaillance d'un élément.

Les systèmes automatisés de production peuvent-être considérés à différents niveaux de granularité, un système automatisé de production étant composé d'autres systèmes (électronique, mécanique). Le niveau de granularité généralement considéré correspond aux différentes machines ainsi qu'aux moyens de transport entre ces machines. Différent niveaux de reconfiguration peuvent alors être considérés à leur tour. Ces niveaux vont de la réorganisation des opérations déjà utilisées à la modification complète de l'architecture du système qui correspond à l'ajout, le remplacement, la suppression ou la modification d'une ressource.

Plusieurs propositions d'utilisation de la reconfiguration dans les SAP sont faites. Les systèmes manufacturiers virtuels (VCMS) [Saad, 2003] ou dynamiques (DCMS) [Drolet et al., 1996] proposent une évolution du concept de cellules qui traditionnellement regroupent et coordonnent des ressources réalisant des traitements sur un même repoduit. Des cellules virtuelles sont constituées et les transports entre ou au sein de ces cellules sont assurés par des véhicules autoguidés (AGV : Auto Guided Vehicule) ce qui permet de modifier le routage des produits et ainsi de reconfigurer les cellules.

Le concept des RMS (Reconfigurable Manufacturing Systems) [Mehrabi et al., 2002] défini au sein de l'ERC/RMS de l'Université du Michigan vise à construire des systèmes de production de manière modulaire afin de faciliter leur évolution. Un exemple de RMS est présenté dans [Moyne et al., 2004]. L'ERC/RMS distingue plusieurs niveaux de description : système, machine et module.

Les systèmes multi-agents ont récemment attiré beaucoup d'attention pour la mise en œuvre de la reconfiguration des systèmes de production. Parmi les architectures proposées, on retrouve les systèmes biologiques [Ueda, 1992], fractals [Ryu et al., 2003] ou encore holoniques [Xu et al., 2002].

Présentation d'un exemple

L'exemple proposé à été introduit dans [Mouchard, 2002; Belabbas and Berruet, 2004]. Il s'agit d'un convoyeur industriel en anneau autour duquel sont placés cinq postes sur lesquels peuvent-être réalisées différentes fonctions correspondant à des traitements sur les produits. Le terme convoyeur sera utilisé par la suite pour désigner à la fois le convoyeur, les cinq postes et les produits qui circulent dessus.

Identification du système reconfigurable

Le convoyeur est en lui-même un système flexible sur lequel la reconfiguration peut être appliqué. Il est en effet possible de modifier le routage des produits entre les différents postes pour réaliser le traitement souhaité. Il est aussi possible de changer le traitement effectué sur le convoyeur. En effet, la demande soumise au convoyeur, en est effectuée en terme de quantité de produits finis d'un type donné, avec des dates de réalisation pour chacun de ces produits, cette demande n'est donc pas ordonnancée préalablement.

Anatomie d'une configuration

Une configuration du convoyeur décrit les produits manipulés, les traitements effectués dessus, l'utilisation des différents postes et le routage des produits entre les différents postes.

Objectif de la reconfiguration

L'objectif de la reconfiguration est de satisfaire les demandes malgré l'apparition d'aléas sur le système. Ici, un aléa peut aussi bien désigner une évolution de la demande ou la défaillance d'un élément du système.

Mise en œuvre de la reconfiguration

Les différents niveaux de reconfiguration considérés dans l'application du convoyeur sont les suivants : une reconfiguration mineure consiste à utiliser différemment les opérations mises en œuvre actuellement. Lors d'une reconfiguration significative, toutes les opérations réalisables sur les postes actuellement utilisés ou sur le convoyeur peuvent être utilisées. Pour une reconfiguration majeure, toutes les potentialités du système sont mobilisables.

Le changement de configuration se traduit par la mise à jour des gammes opératoires du système et des grafcets gérant les postes interprétant ces gammes opératoires. Le passage d'une configuration à une autre peut nécessiter une phase transitoire. En effet, l'état des produits présents sur le système n'est pas nécessairement conforme à la nouvelle configuration.

1.1.6 Systèmes mécaniques

Des cas d'utilisation de la reconfiguration apparaissent aussi au sein de systèmes mécaniques. De tels systèmes sont conçus de manière modulaire, les modules pouvant être modifiés ou réorganisés [Gueganno and Duhaut, 2005; Kamimura et al., 2001; Kotay and Rus, 1999].

Un exemple extrême d'un tel système est présenté dans les paragraphes suivants.

Présentation d'un exemple

L'exemple de système mécanique permettant d'illustrer la reconfiguration est le robot autoconfigurable présenté dans [Kamimura et al., 2001]. Ce robot est constitué d'un agencement de modules identiques interconnectés par un système d'électro-aimants. Ce robot est représenté à la figure 1.6 montrant plusieurs étapes conduisant à une configuration "robot marcheur".

Identification du système reconfigurable

Dans le cas du robot, le système reconfigurable est constitué à la fois du robot, mais aussi de l'ordinateur qui envoie sa configuration. En effet, si l'ordinateur n'est plus disponible, alors le robot ne pourra pas être reconfiguré.

Anatomie d'une configuration

Une configuration de ce système est un agencement des différents blocs qui permet le déplacement du robot dans un contexte particulier.

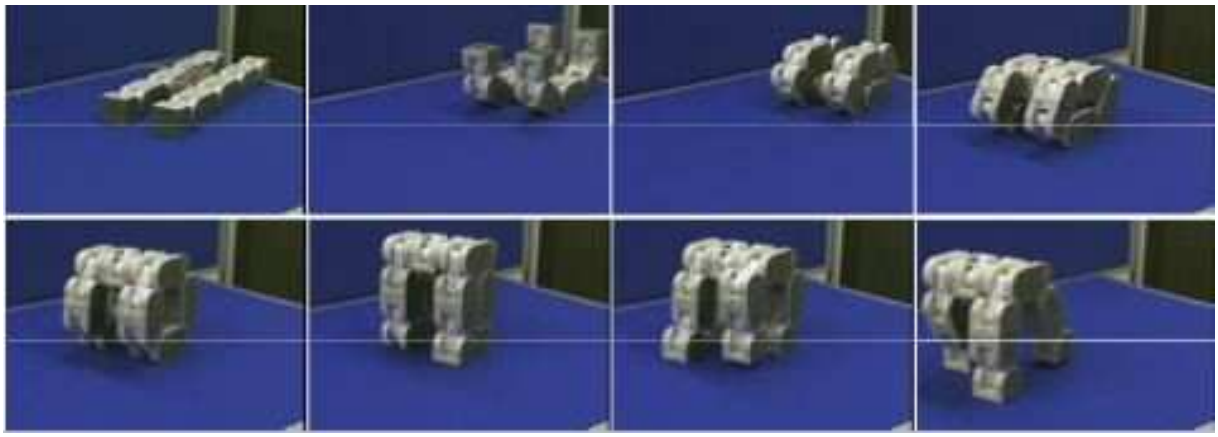


FIG. 1.6 – Metamorphose d'un robot auto-reconfigurable [Kamimura et al., 2001]

Objectif de la reconfiguration

La reconfiguration permet de changer le mode de déplacement du robot suivant l'environnement dans lequel il évolue. Le robot peut ainsi passer d'une configuration dans laquelle il se déplace comme une chenille, propice au déplacement sur du sable, à une nouvelle configuration dans laquelle il marche à quatre pattes et qui lui permet de franchir aisément des obstacles.

Mise en œuvre de la reconfiguration

La construction ainsi que le choix de la nouvelle configuration sont réalisés sur un poste de travail. Une fois la nouvelle configuration choisie, cet ordinateur envoie des ordres aux différents blocs constituant le robot qui se déplacent et prennent une nouvelle position. La reconfiguration se déroule ainsi en plusieurs étapes comme le montre la figure 1.6.

1.2 Points communs entre ces systèmes

Les exemples présentés dans la section 1.1 proviennent de domaines d'application différents. Néanmoins, ces exemples mettent tous en place des mécanismes de reconfiguration. Une liste des caractéristiques communes à ces systèmes tant au niveau de la description que de la mise en œuvre de la reconfiguration est dressée dans cette section.

1.2.1 Invariants dans la description

Un certain nombre de caractéristiques récurrentes dans la description des systèmes utilisant la reconfiguration peuvent tout d'abord être remarquées. Ces systèmes sont généralement définis à un niveau de granularité propre en tenant compte des autres niveaux. Ils possèdent tous un ou plusieurs objectifs de fonctionnement. Différentes configurations peuvent aussi être distinguées pour un système, ainsi qu'une partie fixe, généralement appelée architecture.

Gestion de la granularité

Dans chaque domaine, la description du système considéré se fait à des niveaux de granularité différents. A un niveau de granularité choisi, la description du système se présente généralement sous la forme de composants et de leurs interactions.

Lorsque la description est réalisée au grain le plus fin, comme c'est le cas pour les robots, le niveau de description est atomique. Les composants de base de la configuration ne peuvent pas être divisés.

Au niveau de granularité le plus élevé, un seul composant représentant le système. C'est le cas du mélangeur reconfigurable utilisé sur l'avion X-33 [Cotting and Burken, 2001] qui se présente sous la forme d'un seul composant dont les paramètres peuvent être modifiés. Le RPU présenté dans le cadre du projet Epicure [Auguin et al., 2003] est aussi un exemple de système gros-grain, en effet, l'utilisateur ne dispose que du choix de la fonction réalisée par le RPU.

Bien sûr, il existe une grande variété de granularités intermédiaires. Dans les cas de DJINN [Mitchell et al., 1998], Bullpen [Welch and Purtilo, 1997] ou le Convoyeur, des descriptions plus fines ou plus grossières peuvent aisément être imaginées.

L'objectif du système

Les systèmes étudiés possèdent tous un objectif qui justifie l'utilisation d'un système reconfigurable.

Le premier objectif que l'on rencontre est l'adaptation à la demande. Le système doit être capable de répondre aux besoins pour lesquels il a été mis en place. Si ces besoins ne sont pas satisfaits, il faut trouver une nouvelle configuration pour continuer à répondre à la demande ou pour répondre à une nouvelle demande. Cet objectif est présent au niveau de la plupart des exemples, que ce soit pour le Convoyeur, le framework DJINN, Bullpen, l'avion X-33, ou les robots.

Le second objectif est un objectif de qualité. Il n'est pas toujours suffisant de répondre à la demande, encore faut-il y répondre en suivant les contraintes fixées, c'est l'objectif du framework DJINN et du Bullpen.

La sûreté de fonctionnement est aussi un objectif auquel la reconfiguration fournit une réponse. C'est la motivation de l'utilisation d'un mixer reconfigurable sur l'avion X-33. C'est aussi un des objectifs de l'utilisation de la reconfiguration dans le cas du système de convoyage de colis.

Les exemples présentés montrent aussi que la reconfiguration peut être utilisée pour optimiser l'utilisation des ressources du système. Ce besoin se retrouve dans le cadre du projet Epicure et pour les robots reconfigurables. L'objectif est de réaliser des fonctions, de manière performante avec un minimum de ressources. La reconfiguration permet donc de faire un compromis entre performance et consommation (pris dans un sens général).

Les différents objectifs donnés aux systèmes peuvent être résumés sous la forme de contraintes de Qualité de Service (QdS), où pour chaque mission du système, les contraintes de qualité, de sûreté de fonctionnement, de performance et de consommation peuvent être précisées.

La configuration

La notion de configuration n'est pas toujours explicite, c'est pourquoi nous nous sommes efforcés de faire ressortir cet aspect de la description des différents systèmes présentés. En effet, après l'opération de reconfiguration, le système n'est plus le même qu'avant.

La complexité de la configuration est liée au niveau de granularité auquel le système est décrit. Pour le mélangeur reconfigurable, décrit à un très gros grain, la configuration consiste en un ensemble de neuf paramètres. Pour le RPU de la plate-forme Epicure, une configuration est principalement décrite par le choix de la fonction exécutée. Pour le robot, décrit à un grain très fin, celle-ci consiste en une description de l'assemblage des briques élémentaires. La description de la configuration d'un système décrit à un grain intermédiaire est généralement constituée

d'une description de l'interaction entre les composants ainsi que d'un paramétrage de ceux-ci. C'est le cas pour la plate-forme DJINN, Bullpen et le Convoyeur, que nous avions identifiés comme étant décrits à un grain moyen.

Les notions d'objectif et de configuration sont aussi liées. En effet une configuration est mise en place pour répondre à l'objectif qui a été fixé pour le système. Une configuration peut donc être interprétée comme une traduction de la définition de l'objectif du système en une organisation de ses éléments.

L'architecture du système

La notion de configuration est relative à un ensemble d'éléments qui peuvent être paramétrés et interconnectés. Ces éléments sont les constituants de base du système et constituent son architecture.

Une architecture peut être identifiée dans tous les exemples présentés. Pour le mélangeur reconfigurable de l'avion X-33, il s'agit de la structure du mélangeur qui relie les entrées aux sorties. Pour la plate-forme DJINN, il s'agit de la description des composants qui sont connectés pour constituer l'application. Pour Epicure, l'architecture correspond à l'architecture électronique du système. Pour Bullpen, il s'agit du CVE. Pour le Convoyeur c'est la description de celui-ci. Enfin pour le robot, c'est le nombre d'éléments disponibles ainsi que leur constitution.

1.2.2 Invariants de la reconfiguration

Certains points communs à tous les systèmes reconfigurables peuvent être relevés dans le cadre de la reconfiguration. Il s'agit du déclenchement de la reconfiguration, de la recherche d'une nouvelle configuration et de la mise en œuvre de la configuration. L'impact de l'application d'une nouvelle configuration au système nécessite généralement de passer par une phase de transition.

Déclenchement de la reconfiguration

Dans les exemples présentés, la reconfiguration du système est lancée lorsqu'il apparaît que la configuration courante ne permet plus de remplir les objectifs de QdS qui lui ont été imposés.

La reconfiguration de l'avion X-33 se déroule lorsque l'un des actionneurs des volets se bloque. La configuration des éléments de la plate-forme DJINN est modifiée lorsque la QdS fixée pour l'application n'est plus respectée à cause par exemple d'une détérioration du canal de communication. Dans le cadre d'Epicure, le RPU est reconfiguré lorsqu'une fonction matérielle nécessaire n'est pas chargée. Dans le cadre du système Bullpen, une entité appelée Scout est chargée de tenir à jour l'état du système, un autre agent appelé Coach décide de lancer une reconfiguration quand le réalisme de la simulation est en jeu. Pour le Convoyeur, c'est le non-respect des contraintes de QdS qui entraînent la reconfiguration, si la configuration actuelle nécessite la réparation d'une ressource pour poursuivre la production, et que cette réparation entraîne des retards par rapport aux objectifs fixés, il est nécessaire d'envisager une configuration. Enfin dans le cas du robot, une reconfiguration est lancée quand la configuration actuelle n'est plus adaptée à l'environnement, la configuration "chenille" ne permet pas, par exemple de franchir un obstacle alors que la configuration "quatre pattes" le permet.

Recherche d'une configuration

Quel que soit le système étudié, il faut alors trouver une nouvelle configuration (ou décider de conserver la configuration courante). La nouvelle configuration peut alors être construite, c'est le cas pour le framework DJINN ou Bullpen. La nouvelle configuration peut aussi être choisie dans un ensemble de configurations générées hors-ligne ou conçues, c'est ce qui est fait pour les robots reconfigurables, le mélangeur de l'avion X-33 ou le RPU du projet Epicure. Une solution mélangeant les deux aspects peut aussi être envisagée, c'est ce qui est fait sur le Convoyeur où les configurations en fonctionnement nominal sont définies hors-ligne alors qu'après l'apparition d'une défaillance une nouvelle configuration est construite.

La construction des configurations hors-ligne pour un choix en-ligne permet d'utiliser des méthodes nécessitant une puissance de calcul importante sur des systèmes dans lesquels les ressources dédiées au choix de la configuration sont trop limitées. La construction de la nouvelle configuration permet de profiter au maximum des capacités de réaction offertes par le système.

Plusieurs critères peuvent guider le choix d'une configuration. Dans le cadre de la plate-forme DJINN, intervient non-seulement la QdS fournie par la nouvelle configuration, mais aussi celle que l'on aura pendant la reconfiguration. Cette notion de coût de la reconfiguration se retrouve dans le cadre du Convoyeur pour lequel un ensemble de *niveaux de reconfiguration* ont été définis.

Les niveaux de reconfiguration ont été définis dans le cadre des systèmes automatisés de production [Berruet, 1998] pour classifier les reconfigurations suivant la complexité de mise en œuvre. Trois niveaux ont été identifiés pour les systèmes de production. Le premier consiste à reparamétrer la commande active. Pour le second niveau, seules les ressources déjà en tension sont utilisées. Le troisième niveau permet de prendre en considération toutes les ressources.

Les configurations peuvent aussi être distinguées par rapport à la modification ou non des objectifs du système. En effet, un aléa peut, par exemple, empêcher la réalisation des objectifs du système. Si aucune configuration ne permet de pallier ce manque, une solution consiste à choisir un objectif moins prioritaire mais réalisable pendant que la partie défaillante est réparée. Ce dilemme entre changement de configuration et changement d'objectif se retrouve, par exemple, dans le cadre du Convoyeur.

La phase de transition

La nécessité de gérer la transition depuis l'ancienne configuration vers la nouvelle se retrouve aussi dans des systèmes appartenant aux différents domaines présentés.

Les conditions de cette transition diffèrent en fonction des contraintes imposées au système. Si les contraintes le permettent, le système peut par exemple être arrêté complètement. C'est ce qui est fait dans le cadre du robot où celui-ci arrête sa progression, se reconfigure, puis continue son chemin.

Dans le cas de la plate-forme DJINN, par contre, où des documents multi-média sont transférés sur un réseau, il faut que la transition soit la moins brutale possible. Des contraintes de QdS sont fixées pour les phases de reconfiguration et doivent être respectées.

Dans le cas des systèmes manufacturiers comme le Convoyeur, la gestion des produits dans le système au moment de la reconfiguration constitue un problème majeur qui peut être traité par la gestion des modes [Dangoumau, 2000].

Pour Bullpen, l'avion X-33 et Epicure, les reconfigurations sont considérées comme étant atomiques. Néanmoins, dans le cas de Bullpen, il est spécifié que la simulation n'est pas réaliste pendant la reconfiguration qui dure une fraction de secondes. Dans le cadre d'Epicure, le RPU ne peut pas être utilisé pendant sa reconfiguration.

1.2.3 Enchevêtrement de systèmes reconfigurables

Si les systèmes reconfigurables peuvent généralement être considérés à des niveaux de granularité différents, des systèmes de natures différentes peuvent se retrouver conjointement à un même niveau de granularité.

Un tel enchevêtrement apparaît dans le cas des SAP. Celui-ci fait intervenir des systèmes électroniques, informatiques, mécaniques ainsi que de la gestion de l'énergie.

En effet, la partie commande du SAP est constituée de systèmes électroniques pour le contrôle commande ainsi que pour la communication. Ces systèmes électroniques constituent le support d'un système informatique. La partie opérative du SAP est un système mécanique, relié au système électronique par les interfaces des actionneurs et des capteurs. Enfin, le fonctionnement du système nécessite de l'énergie.

Ces différents sous-systèmes composant un SAP peuvent tous être des systèmes reconfigurables. Il semble alors tout à fait envisageable d'utiliser une même notation pour décrire les capacités de reconfiguration des différents systèmes métiers constituant le système complet.

1.3 Proposition de définitions autour des Systèmes Reconfigurables

A partir d'un ensemble identifié de caractéristiques communes aux systèmes reconfigurables utilisés dans des domaines différents, nous introduisons des définitions se rapportant aux systèmes reconfigurables, leur composition et leur reconfiguration.

1.3.1 Système Reconfigurable

Un Système Reconfigurable est un système disposant d'une architecture constituée des entités composant le système ainsi que d'une configuration spécifiant comment ces entités sont utilisées afin de répondre à un objectif. Un système reconfigurable dispose d'un mécanisme permettant de choisir une nouvelle configuration et de la mettre en place dans le cadre du processus de reconfiguration.

1.3.2 Architecture d'un système reconfigurable

L'architecture d'un système reconfigurable peut être définie comme l'ensemble des entités composant le système ainsi que les interactions potentielles entre elles (implicitement ou explicitement décrites).

1.3.3 Configuration d'un système reconfigurable

La configuration d'un système reconfigurable décrit l'utilisation des éléments de l'architecture afin de répondre aux objectifs fixés au système sous la forme de contraintes de QdS.

1.3.4 Reconfiguration des systèmes reconfigurables

La reconfiguration d'un système reconfigurable consiste à modifier la configuration du système. Cette reconfiguration s'effectue dans le cadre du processus de reconfiguration. La reconfiguration peut être menée en plusieurs étapes si l'état courant du système (position des produits

et mode des ressources pour un système de production) n'est pas admissible par la nouvelle configuration.

1.3.5 Le processus de reconfiguration

Le processus de reconfiguration d'un système reconfigurable démarre lorsque l'objectif fixé pour la configuration actuelle est atteint ou lorsque la configuration actuelle du système ne satisfait plus ces mêmes objectifs. Ce processus se déroule en plusieurs phases dès lors que la décision de lancer le processus de reconfiguration a été prise. Tout d'abord, une nouvelle configuration répondant aux objectifs fixés est recherchée. Un cheminement doit être déterminé pour aboutir à cette configuration, il est nécessaire de placer le système dans un état acceptable pour la nouvelle configuration. Puis la reconfiguration est mise en œuvre pour aboutir au nouveau fonctionnement répondant aux objectifs fixés en terme de QdS.

1.4 Conclusion

Bien que provenant de domaines différents, des caractéristiques semblables se retrouvent dans les différents systèmes reconfigurables présentés dans ce chapitre. Serait-il alors possible de considérer la reconfigurabilité comme une caractéristique d'un système et définir un aspect reconfiguration dans sa description ?

Concernant la proposition d'un langage pour la description du système, la prise en compte de l'enchevêtrement des systèmes reconfigurables rend nécessaire le développement conjoint de différentes parties du système faisant intervenir des experts provenant de domaines très différents. Une représentation commune de l'aspect reconfigurable du système permettrait à ces personnes de réfléchir ensemble à la mise en œuvre de la reconfiguration de manière globale et ainsi de tirer le meilleur parti des potentialités offertes par le système.

Plusieurs approches de la reconfiguration sont proposées dans la littérature par rapport au moment où les configurations sont obtenues. Les approches de type "réactives pures" se basent sur un ensemble de configurations déterminées avant le fonctionnement du système, ces approches garantissent un temps de réaction rapide mais seuls les utilisations prévues du système sont prises en compte, c'est le type d'approche adopté au chapitre 6 pour le démonstrateur ferroviaire. D'autres approches proposent une construction de la nouvelle configuration pendant le fonctionnent du système et peuvent être qualifiées de "réactives", la recherche d'une nouvelle configuration nécessite des ressources de calcul dédiées et n'est généralement pas réalisée en un temps déterministe.

Avant de proposer une notation commune, permettant de décrire l'aspect "reconfigurable" d'un système, il est nécessaire de faire un tour d'horizon des notations actuellement utilisées ainsi que les outils à notre disposition pour établir la description d'un système. L'objectif du chapitre suivant est aussi de trouver des outils adaptés à la mise en place d'une approche pertinente de conception des systèmes reconfigurables. Cette approche doit permettre la modélisation et l'analyse des aspects communs aux systèmes reconfigurables tout en autorisant une adaptation de la modélisation afin de représenter des concepts propres à un domaine particulier en vue d'analyses spécifiques et/ou d'une mise en œuvre.

Chapitre 2

Vers une représentation de haut niveau

Le chapitre précédent a mis en évidence que les différents types de systèmes reconfigurables partageaient un ensemble de caractéristiques communes, à la fois dans leur description et dans leur fonctionnement.

Deux grandes familles de systèmes reconfigurables sont plus particulièrement ciblées dans le cadre de ces travaux. Il s'agit des systèmes électroniques et des systèmes automatisés de production. L'objectif de ce chapitre est de rappeler les approches couramment utilisées pour l'étude de ces systèmes, de présenter des langages de description de haut niveau puis d'introduire l'ingénierie des modèles comme une base pour la définition d'un langage de description des systèmes reconfigurables.

La première partie de ce chapitre se focalise sur les approches de la littérature en électronique et pour les SAP. Ensuite, des langages de haut-niveau, utilisés dans ces mêmes domaines sont décrits. Finalement l'approche DSL est introduite dans une troisième partie.

2.1 Approches classiques pour la conception des systèmes de production et électroniques

Les systèmes de production comme les systèmes électroniques ont fait l'objet de très nombreuses études concernant les méthodologies de conception. Un état de l'art de celles-ci est dressé dans cette section.

2.1.1 La conception des systèmes de production

Dans le cas des SAP, la grande majorité des méthodes de conceptions s'intègrent dans le cadre du CIM (Computer Integrated Manufacturing) présenté dans un premier temps. La conception de la commande des systèmes de production fait l'objet d'études au sein des laboratoires participant au GT INCOS (INgénierie de la COmmande et de la Supervision) du CNRS. Des approches ont été proposées au LAGIS [Toguyeni, 2001], au LAI [Rezg, 1996; Khatab, 2000], au CRAN [Gouyon, 2004] au LAAS [Zamaï, 1997] et au LAG [Mendez, 2002; Henry, 2005]. Parmi ces approches, celles du LAGIS et du LAG seront particulièrement développées dans deux sections. La dernière section présente les travaux pré-existants au niveau du LESTER.

Le concept CIM

Le CIM (Computer Integrated Manufacturing) [Hannan, 1996] est un concept d'intégration des services de l'entreprise développé au milieu des années 1970 et toujours d'actualité. L'ob-

jectif du CIM est de réaliser une intégration verticale des différents services en profitant des avancées dans le domaine des systèmes d'information. Cette décomposition a progressivement remplacé la traditionnelle décomposition horizontale qui séparait les différents corps de métiers.

Les premiers éléments à être intégrés furent les aspects production qui comprennent les processus de fabrication, la planification de ces processus ainsi que la gestion de production et le contrôle des approvisionnements.

Conjointement au concept de CIM, un ensemble d'approches permettent d'intégrer les informations présentes dans les différents secteurs de l'entreprise. Parmi ces approches, nous nous intéressons aux méthodologies de type exécutif qui correspondent aux travaux réalisés dans le cadre du GT INCOS. Ces méthodologies s'intéressent au système opérationnel. Les points de vue intégrés sont la commande, le contrôle, la maintenance et la gestion du système.

La décomposition du système conduit à la pyramide CIM, représentée à la figure 2.1. De par l'intégration verticale chaque point de vue (commande, contrôle, maintenance, gestion) est représenté à chaque niveau.

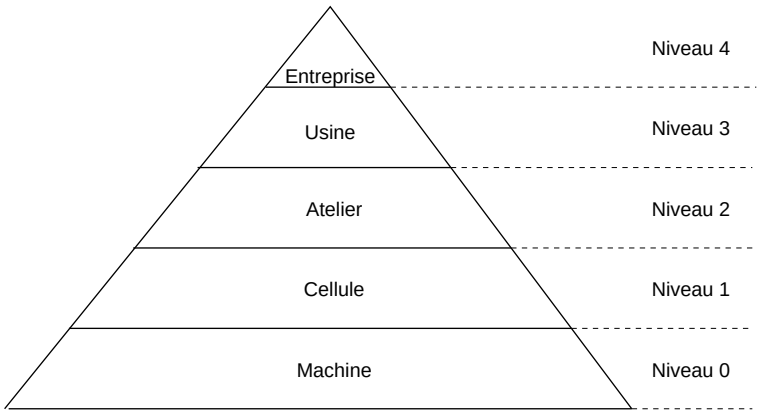


FIG. 2.1 – Pyramide CIM

Cette pyramide présente les différents niveaux de décomposition d'une entreprise. Le choix d'un ou plusieurs niveaux de cette pyramide détermine le grain de la description du système. Ces niveaux vont du niveau "entreprise" dans lequel les préoccupations sont principalement financières, jusqu'au niveau "machine" s'occupant du fonctionnement d'une machine en particulier.

Les niveaux du CIM qui nous intéressent plus particulièrement sont les niveaux 1 et 2, respectivement "cellule" et "atelier". Ces niveaux gèrent la coordination entre les machines afin de réaliser les traitements souhaités. Ce sont aussi les niveaux visés par les approches présentées dans les paragraphes suivants.

Nous avons aussi ciblé le niveau 0 du CIM dans le cadre de l'implantation de la commande telle qu'elle est réalisée au chapitre 6.

L'approche du LAGIS

Le LAGIS a développé une méthodologie complète appelée CASPAIM [Craye, 1994] (Conception Assistée des Systèmes de Production Automatisée de l'Industrie Manufacturière) correspondant aux niveaux 1 et 2 du CIM. Cette méthodologie définit une architecture de commande pour les systèmes de production mettant en place ses différentes fonctions.

Un système de production est décomposé en deux parties : une partie logique et une partie physique. La partie logique du système permet de décrire les produits fabriqués par le système

de production. Ceux-ci sont décrits par leur gamme logique qui représente les différentes fonctions que le système doit réaliser afin d'obtenir le produit fini à partir de son état brut. La partie physique décrit les différentes ressources de production (transformation, transport, stockage ...).

La conception du système est réalisée en quatre phases représentées sur la figure 2.2. La phase de spécification permet de décrire les parties physiques et logiques du système ainsi que les informations liées à la supervision (stratégie, gestion des modes). Vient ensuite la phase d'analyse et de conception au cours de laquelle le modèle du système de coordination et de contrôle est obtenu, ce modèle est constitué des gammes opératoires (GO) et est étendu pour prendre en compte le contrôle des ressources. Les GO constituent une description des opérations réalisées (usinages, transferts, stockage) ainsi que leur enchaînement. La troisième phase est une phase de validation du système complet. Finalement vient la phase d'implantation qui permet de mettre en œuvre les GO obtenues sur l'architecture de contrôle/commande présentée à la figure 2.3.

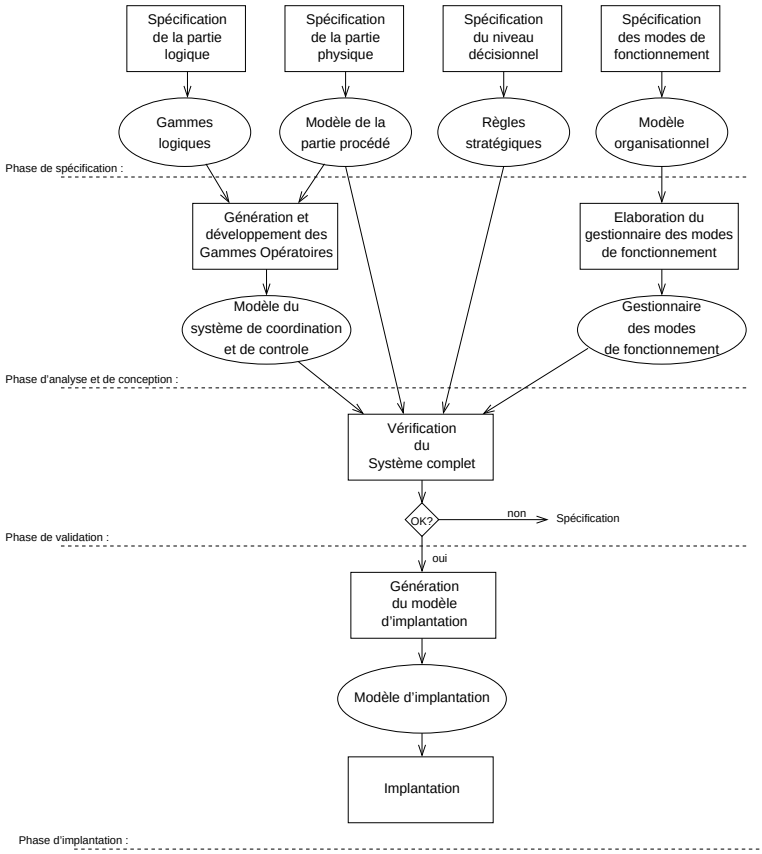


FIG. 2.2 – Démarche de conception CASPAIM simplifiée

L'architecture de commande CASPAIM intègre les différentes fonctions d'un système de production. La fonction "Commande" est liée à la "Partie Opérative" au travers du module de "Surveillance" qui peut effectuer un filtrage en cas de défaillance de la partie opérative ou de la partie commande. La "Commande" est aussi reliée à la "Supervision" qui à travers les modules "Pilotage" et "Gestion des modes" assure une évolution sûre du système. La fonction "Planification Ordonnancement" est réalisée hors-ligne et permet de déterminer un plan de production. Enfin, la fonction "Maintenance" se charge en coordination avec la "Supervision" de mettre en œuvre le plan de maintenance.

Deux opportunités pour la reconfiguration apparaissent au niveau de la fonction "Supervision". La *reconfiguration matérielle* fait appel à la gestion des modes pour déterminer l'évolu-

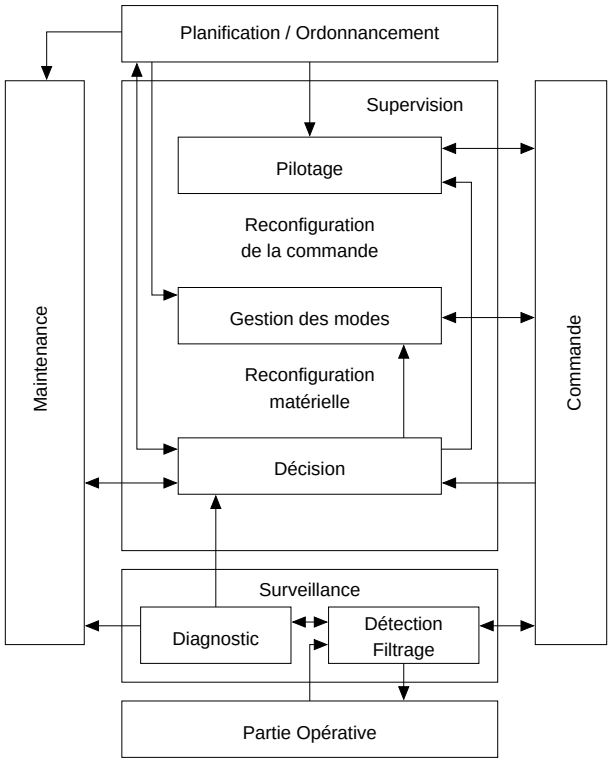


FIG. 2.3 – Architecture de contrôle commande CASPAIM

tion de la partie opérative alors que la *reconfiguration logicielle* revient à modifier l'action du "Pilotage" sur la commande.

L'approche du LAG

Les travaux de l'équipe Conduite et Organisation des Systèmes de Production (COSP) du LAG prennent leur source dans les travaux effectués au LAAS pour la conception de systèmes de contrôle/commande sûrs de fonctionnement.

L'architecture de commande proposée est une architecture hiérarchique. Les nœuds de cette hiérarchie sont semblables et appelés module CERBERE, ils sont décrits à la figure 2.4. Le CERBERE intègre les fonctions de surveillance/supervision au même niveau que les autres. La gestion des communications est distribuée dans des blocs de routage présents dans chaque module. Ces blocs orientent les informations vers les fonctions correspondantes du bloc de traitement ou vers les autres niveaux.

La composition du CERBERE garantit la réactivité lors de la réaction à une défaillance, ces points ont principalement été traités dans les travaux d'Eric Zamaï [Zamaï, 1997] et Hector Mendez [Mendez, 2002].

La commande en fonctionnement normal du système est obtenue en ligne à l'aide d'un algorithme de synthèse décrit dans la thèse de Sébastien Henry [Henry, 2005]. La synthèse de la loi de commande est réalisée à partir d'un modèle de la partie opérative, elle produit un réseau de Petri permettant de commander le système considéré. Les réseaux de Petri obtenus représentent le fonctionnement normal du système, tout écart par rapport à cette commande est donc traité par une autre fonction du système. Le suivi du bon déroulement des opérations est alors primordial. Cet activité est mise en œuvre dans le cadre d'une fonction de suivi actuellement développée dans le cadre des travaux d'Eric Deschamps [Deschamps et al., 2006]. La reconfiguration du système passe alors par l'application de l'algorithme de synthèse de commande sur le modèle,

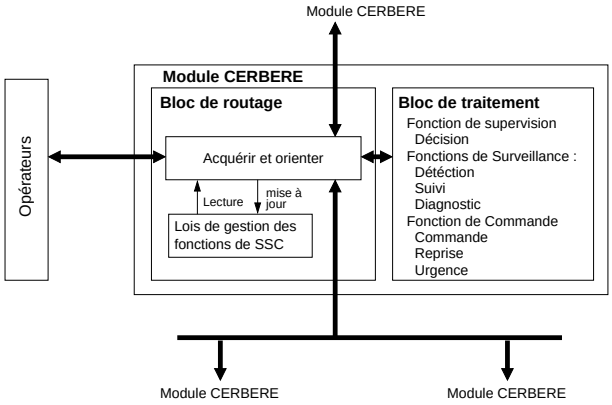


FIG. 2.4 – Structure interne d'un module CERBERE [Mendez, 2002]

mis à jour par la fonction de suivi.

L'approche du LESTER

Les travaux développés dans la thèse de Jean-Sébastien Mouchard [Mouchard, 2002] ont introduit la notion de composant, tel que représentés à la figure 2.5, pour la construction d'un système transitique. La notion de composant permet d'augmenter les possibilités de réutilisation de programmes déjà validés lors de la conception d'un nouveau système.

Un composant est défini comme un *élément modélisant une partie du système transitique en incluant différents modèles pour décrire cette partie du système. Chacun de ces modèles est inclus dans une «vue» différente (opérative, commande, graphique, contrainte, commande, surveillance/supervision).*

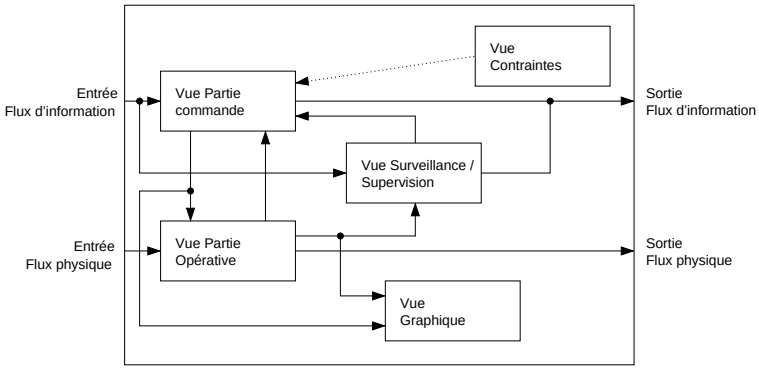


FIG. 2.5 – Schéma interne d'un composant [Mouchard, 2002]

La vue "Partie opérative" décrit le comportement des parties mécaniques, pneumatiques et électriques de l'élément modélisé. Cette vue est nécessaire en phase de conception, elle permet de prendre en compte la partie opérative dans la vérification du système. Dans le cadre de cette vérification, la vue "Graphique" permet, lors des simulations, de n'afficher que les paramètres pertinents de la partie opérative à l'utilisateur. La vue "Contrainte" exprime les contraintes d'utilisation d'un composant et permet de vérifier leur bonne utilisation. La vue "Commande" décrit les traitements permettant de réaliser les fonctions du composant. Finalement la vue "Surveillance / Supervision" intervient lors de l'exploitation et décrit les mécanismes de surveillance pour la détection d'aléas, ainsi que les techniques de correction après leur occurrence. La conception de la commande du système complet est donc distribuée au sein des différents

composants. Chaque composant exporte ses vues afin de constituer des composants de niveau supérieur.

Le cycle de conception proposé est constitué de quatre étapes. La conception débute par les spéfications du système. Une modélisation de celui-ci est ensuite réalisée, conformément aux spécifications. Ensuite vient une étape de prototypage qui se termine par une simulation à l'aide du logiciel SimSED. Le cycle se termine par une intégration de la commande sur un ou plusieurs automates industriels. Les activités de vérification et de validation sont réalisées parallèlement aux activités de prototypage et d'intégration.

L'approche proposée lors de la phase de modélisation est réalisée en deux phases. La première phase de modélisation est descendante. Elle débute par la décomposition de l'application en macro-éléments. Ces éléments sont alors décomposés en éléments simples. Une deuxième phase, ascendante débute alors. Des composants sont choisis dans une bibliothèque pour représenter les éléments simples. Si aucun élément ne convient, il est créé et enrichit la bibliothèque. Les composants élémentaires sont ensuite agrégés pour retrouver les macro-éléments définis précédemment. Le dernier niveau d'agrégation permet d'obtenir le modèle du système complet.

2.1.2 La conception des System on Chip (SoC) en électronique

La conception des circuits électroniques soulève plusieurs problèmes. En effet, les possibilités offertes par les technologies actuelles sont sans cesse en évolution. D'un autre côté, les besoins des utilisateurs sont de plus en plus difficiles à satisfaire. Il en résulte des systèmes complexes, nécessitant des méthodologies de conceptions adaptées. Le flot décrit sur la figure 2.6 et issu de [ITRS, 2005] reprend les différentes étapes du développement d'un système électronique.

La phase de spécification consiste à déterminer les exigences (cahier des charges) à partir des besoins du client.

A partir des exigences, une analyse fonctionnelle du système est suivie de la conception d'une architecture électronique. La recherche de performance nécessite l'utilisation d'entités dédiées au sein d'un circuit. A côté de processeurs génériques réalisant les tâches courantes, prennent place des *accélérateurs* et des *co-processeurs*. Néanmoins, l'utilisation de ces unités de calcul dédiées a un coût en surface silicium, il est donc nécessaire pour chaque fonction, de choisir entre une implémentation logicielle ou matérielle. Ce choix nécessite une étape d'exploration des différentes solutions envisageables. L'exploration des différentes solutions permet d'aboutir à des spécifications conjointes pour le logiciel et une autre spécification pour le matériel.

S'ensuit le développement de l'architecture matérielle et du logiciel. Pour le matériel une première étape consiste à décomposer l'architecture en blocs, ces blocs appelés IP (Intellectual Property) peuvent être décrits à plusieurs niveaux suivant le grain utilisé. A un niveau comportemental, l'algorithme est décrit alors qu'au niveau RTL (Resistor Transfer Logic) l'IP doit être utilisée telle qu'elle. La conception de l'architecture matérielle nécessite des phases de modélisation et de vérification.

Les IP doivent ensuite être interconnectées pour réaliser l'architecture, des interfaces sont générées pour adapter les moyens de communication (bus, NOC (réseau sur puce), point-à-point). Dans le cas d'IP comportementales, il est aussi nécessaire de synthétiser le code RTL de l'IP. Une fois les modèles RTL de l'application obtenus, il faut les placer sur le circuit dans une phase de conception physique.

Le développement du logiciel s'effectue en parallèle, à partir des spécifications logicielles du système. La conception conjointe de l'application logicielle et de l'architecture matérielle sur

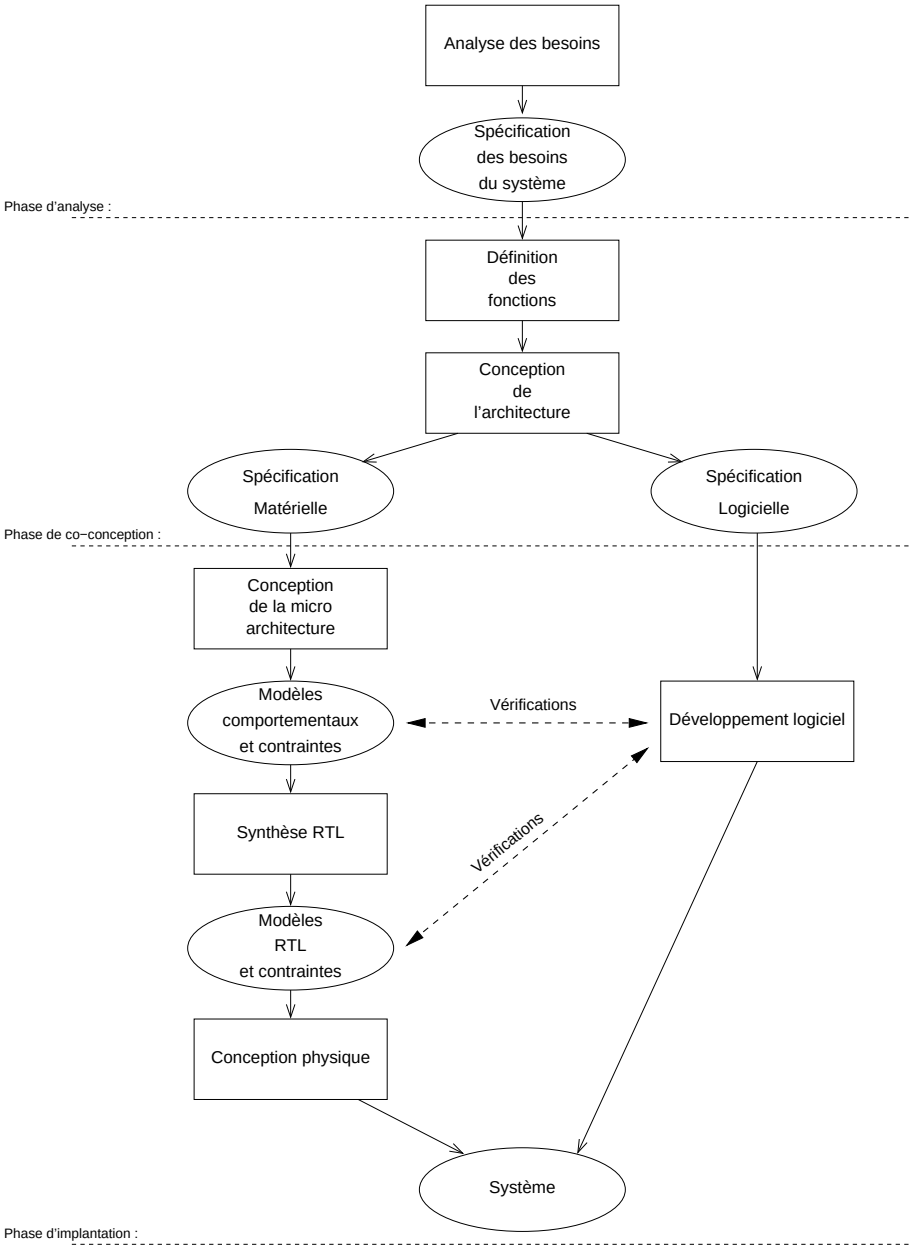


FIG. 2.6 – Flot de conception des systèmes électroniques

laquelle elle fonctionne pose de nombreux problèmes. Il peut par exemple s'avérer nécessaire de mettre au point un logiciel pour lequel la plate-forme matérielle n'existe pas entièrement, il en est de même pour le test du matériel. Des techniques de prototypage ont donc été développées, elles permettent par exemple de simuler l'exécution du logiciel sur la modélisation haut-niveau de l'architecture [Calvez, 1990].

Le LESTER apporte des solutions pour la conception des systèmes électroniques entrant dans le cadre présenté précédemment. L'exploration de l'espace de conception est réalisé par le logiciel Design-Trotter [le Moullec, 2003]. L'outil développé dans le cadre de A3S (Adéquation Algorithme Architecture Système) couvre les phases de spécification et de conception. L'équipe IP traite de la conception et de la réutilisation des blocs IP (propriété intellectuelle) [Coussy, 2003]. L'outil GAUT [Martin et al., 1993] permet de réaliser la synthèse de ces IP à partir d'une description algorithmique.

En ce qui concerne le positionnement de noms travaux dans le flot de conception électronique, ils se placent au niveau de la phase de co-conception. Le modèle de haut-niveau qui sera présenté au chapitre 3 permet de décrire à la fois les spécifications logicielles et matérielles, spécifications prenant en compte la possibilité de faire évoluer la configuration du système.

2.1.3 Bilan sur les approches classiques

Ainsi, différentes approches pour l'étude de systèmes ont été présentées à la fois pour les SAP et dans le cadre de la conception de systèmes électroniques. La section suivante traite des langages actuellement utilisés comme support pour ces méthodes.

2.2 Langages de représentation de haut niveau

L'étude des systèmes passe par l'utilisation de langages adaptés à leur description. Certains de ces langages sont spécifiques et ne sont utilisés que dans un domaine d'application bien particulier. D'autres langages très génériques n'ont pas toujours une sémantique clairement fixée mais peuvent être utilisés par de nombreux acteurs. L'objectif de cette section est de présenter quelques langages de représentation de haut niveau.

2.2.1 Les modèles utilisés pour la conception des systèmes de production

La conception des systèmes de production passe par l'utilisation de nombreux modèles. Ces modèles servent de support à l'approche suivie. Deux grandes familles d'approches peuvent être caractérisées :

La première famille d'approche regroupe les approches formelles. Ces approches sont basées sur des langages formels qui peuvent soit représenter le système par ses états et les transitions entre ces états. Les modèles utilisés sont des automates, des réseaux de Petri. Dans le cadre des approches formelles, l'utilisation des systèmes logiques permet de réaliser des preuves par réécritures. L'utilisation de modèles formels permet en outre de vérifier et valider le système. Ils fournissent aussi des opérations de synthèse permettant d'obtenir un contrôleur pour le système [Ramadge and Wonham, 1987; Achour and Rezg, 2004].

La deuxième famille d'approches repose sur la description du système. La conception se fait par raffinements successifs jusqu'à obtenir le modèle correspondant au code de commande du système. Deux catégories peuvent alors être distinguées. Certaines approches utilisent des modèles homogènes, qui sont raffinés tout au long de la conception. C'est le cas par exemple de la méthode CASPAIM qui utilise des réseaux de Petri. D'autres méthodes sont fondées sur

l'utilisation de modèles différents pour les différentes phases de la conception. Des outils tels que la transformation de modèles rendent possible la construction d'un cadre pour l'utilisation de ces modèles.

2.2.2 Langages de haut niveau en électronique

Des langages de haut niveau sont aussi utilisés dans la conception électronique, à des fins de modélisation et de vérification. Les langages de description d'architecture tels que LISA [Hoffmann and Meyr, 2002] permettent de décrire l'architecture de processeurs disposant d'un jeu d'instructions spécifique. D'autres outils comme Esterel studio [Esterel Technologies, 2006] permettent la validation de systèmes et sont utilisés en électronique.

Le langage LISA consiste en un ensemble de modèles décrivant les différents aspects du processeur. LISA repose sur un modèle des mémoires, un modèle des ressources, un modèle comportemental, un modèle du jeu d'instructions, un modèle temporel ainsi que le modèle de la micro-architecture.

Les modèles sont utilisés lors des différentes phases de conception. Certains modèles comme le modèle comportemental sont utilisés uniquement lors de la simulation et sont des abstractions d'autres modèles. La table 2.1 indique comment sont utilisés les différents modèles pour l'obtention des outils de conception du système.

	modèle des mémoires	modèle des ressources	modèle comportemental	modèle du jeu d'instructions	modèle temporel	modèle de μ-architecture
Compilateur HLL	allocation des registres	ordonnancement des instructions	-	sélection des instructions	ordonnancement des instructions	regroupement des opérations
assembleur	-	-	-	translation des instructions	-	-
éditeur de liens	allocation mémoire	-	-	-	-	-
désassembleur	-	-	-	désassemblage des instructions	-	-
simulateur	simulation du stockage	-	simulation des opérations	décodage des instructions	ordonnancement des opérations	-
débuggeur	configuration de l'affichage	profiling	-	-	-	-
générateur HDL	structure de base	résolution des conflits	-	décodage des instructions	ordonnancement des opérations	groupement des opérations
intégration	accessibilité, contrôlabilité	-	-	-	-	-

TAB. 2.1 – Utilisation des différents modèles au sein du langage LISA [Hoffmann and Meyr, 2002]

D'autres langages de haut-niveau sont utilisés pour la description et la vérification des circuits électroniques. Dans le cadre du projet Epicure [Auguin et al., 2003], le langage Safe State Machines (SSM) [André, 1996] fourni par l'outil Esterel Studio est utilisé pour modéliser le contrôle de l'application. L'utilisation de ce langage permet d'effectuer une vérification formelle de la partie du système modélisé, de la simuler pour finalement réaliser une implantation.

2.2.3 UML : Le langage "universel"

Le langage UML (Unified Modeling language) [OMG, 2003a] est un langage de modélisation introduit comme support pour le développement logiciel. C'est un langage graphique, constitué de 9 diagrammes pour la version 1. Le langage UML, propose le diagramme de classe, qui permet de définir des relations entre les entités composant le système. Les diagrammes état/-transition et d'activité sont des machines à états permettant d'exprimer le comportement des entités décrites. Les diagrammes de séquence permettent de décrire des scénarios en ordonnançant des messages entre entités sur une échelle temporelle. Les diagrammes de cas d'utilisation

servent de support à l'analyse fonctionnelle. Les diagrammes de déploiement permettent de partitionner l'application sur une architecture matérielle.

La sémantique d'UML est semi-formelle, ce qui signifie qu'une partie de la définition de la notation est formellement décrite par des diagrammes de classe et des contraintes écrites en OCL (Object Constraint Language) [OMG, 2003b] mais qu'une partie de la description est réalisée en langage naturel (en anglais) et est sujette à de multiples interprétations.

La grande force d'UML vient justement de la sémantique semi-formelle sur laquelle il se base. La liberté laissée au concepteur est très intéressante pendant les phases de conception, encore faut-il que tous les intervenants soient d'accord sur les significations utilisées.

Une caractéristique intéressante d'UML repose sur les mécanismes d'extension qu'il fournit. Ces mécanismes permettent d'étendre la notation en fournissant ses propres stéréotypes. Un stéréotype est une étiquette qui peut être placée sur les éléments UML afin de préciser ou même changer leur sens. Ainsi, il est possible de se mettre d'accord sur une sémantique particulière, décrite dans un profil et contraignant l'utilisation d'une notation précise. Il est ainsi plus aisé de proposer des générateurs de code reposant sur ces profils et dont le comportement sera déterministe.

Un exemple de profil UML proposé pour les applications de radio logicielles est le profil A3S développé au LESTER. Ce profil étend la notation UML pour permettre la description des composants matériels constituant un système électronique. La définition des algorithmes utilisés est contrainte par une redéfinition des diagrammes d'activité. La nouvelle notation proposée, basée sur UML est alors supportée par la grande variété d'outils conformes à la norme. La définition précise du profil permet l'utilisation d'outils de vérification tels que XAPAT [Rouxel et al., 2005] ou encore d'outils de génération de code.

UML a réussi à s'imposer comme la notation de référence dans le domaine du génie logiciel, principalement grâce à son acceptation par les différents acteurs du développement qui facilite grandement la communication entre eux. UML a souffert des nombreuses incohérences qu'il portait ainsi que des nombreuses interprétations que peuvent avoir les mêmes notations. Les mécanismes d'extension permettent de résoudre ce problème mais rendent le langage ainsi défini uniquement utilisable par des experts, UML perd alors son caractère générique.

Néanmoins, plusieurs travaux se basent sur UML pour la conception aussi bien des systèmes de production que des systèmes électroniques. Pour les systèmes de production, [Tsai and Sato, 2004] présente une utilisation d'UML pour générer, à partir du même modèle UML, le plan de production et le code de contrôle/commande. En électronique, [Beierlein et al., 2004] présente un flot de conception d'architectures électroniques reconfigurables utilisant UML.

Au vu de la popularité d'UML et de la grande variété des utilisations qui en est fait, il semble nécessaire, dans le cadre de l'élaboration d'un langage de haut-niveau de proposer au moins des passerelles entre le langage défini et UML. Ces passerelles peuvent être définies à l'aide d'un profil UML.

2.2.4 Les langages de configuration

Toujours dans le domaine informatique sont apparus, au début des années 1980 [Medvidovic and Taylor, 2000; Kramer, 1990], les langages de description d'architecture (ADL) ou langages de configuration. L'objectif de ces langages est de permettre la programmation d'un logiciel en se focalisant sur la structure du système. La construction du système se fait en définissant des composants de base, qui sont ensuite agencés afin de mettre en place une architecture.

Certains ADL reposent sur une notation formelle, ils permettent alors la vérification de l'architecture avant de procéder à la mise en œuvre du logiciel puis à son déploiement.

La deuxième génération des langages de configuration a introduit la notion de reconfiguration dynamique. La structure du logiciel peut alors évoluer au cours de son fonctionnement.

Le langage CLARA (Configuration LAnguage for Real-time Applications) illustre parfaitement les concepts des langages de configuration. Ce langage a été développé spécialement pour prendre en compte les caractéristiques des applications Temps-Réel [Durand, 1998]. Il permet la description d'entités logicielles encapsulées dans des composants appelés "activités" ainsi que de leurs interactions. La figure 2.7 présente un exemple de description d'une architecture logicielle à l'aide de CLARA. Les activités y sont représentées sous la forme de boites. Ces activités arborent différents ports par lesquels peuvent transiter des signaux (triangles blancs) ou des données (triangles noirs). Le protocole utilisé pour le lien est représenté sur les liens entre ces différents ports. Une contrainte de temps représentée sous la forme d'un arc entre les deux ports concernés a aussi été proposée sur le schéma.

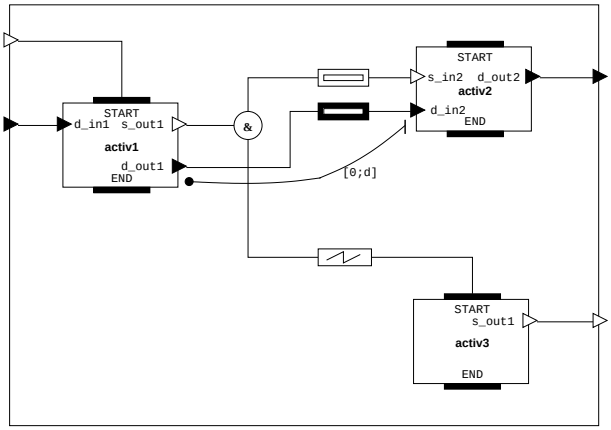


FIG. 2.7 – Exemple de description d'une architecture logicielle avec CLARA [Faucou et al., 2001]

Petit à petit, le monde des ADL s'est rapproché de celui d'UML. Des profils sont alors apparus afin de permettre la manipulation des concepts introduits avec les langages de configuration dans le cadre d'UML. Un profil UML a par exemple été défini pour le langage AADL [SAE, 2004].

Les langages de configuration prennent en compte cette notion de configuration dont nous aurons besoin pour la description des systèmes reconfigurables.

2.3 Introduction à l'ingénierie dirigée par les modèles

L'évolution de l'informatique a récemment vu la naissance d'une évolution des approches orientées objet pour aboutir à des approches orientées modèles. L'idée est de disposer de plusieurs modèles différents du système tout au long de sa conception.

L'approche MDA (Model Driven Architecture) est une initiative de l'OMG (Object Management Group) allant dans ce sens. Mais la plupart des approches basées sur le MDA n'utilisent généralement qu'un seul langage : UML, avec lequel plusieurs modèles sont manipulés tout au long de la conception.

Le concept de modèle est introduit à la première section. Suivi par une deuxième section définissant les transformations. La notion de DSL est ensuite décrite préalablement à l'introduction de la plate-forme AMMA, utilisée dans le cadre de ces travaux.

2.3.1 Organisation des modèles

L'ingénierie dirigée par les modèles introduit une hiérarchisation des modèles manipulés. Un *modèle* correspond à l'abstraction d'un système. La figure 2.8 présente la relation entre un système et un modèle telle qu'elle est décrite dans [Bézivin, 2005]. On peut lire sur cette figure : "un système est représenté par un modèle".

De plus, un modèle nécessite une définition lui procurant une sémantique. Cette définition est fournie par son *méta-modèle*. Un méta-modèle est un modèle définissant un modèle du niveau inférieur, la relation entre un modèle et son méta-modèle est une notion de conformance, on dit qu'un modèle est "conforme à" son méta-modèle.

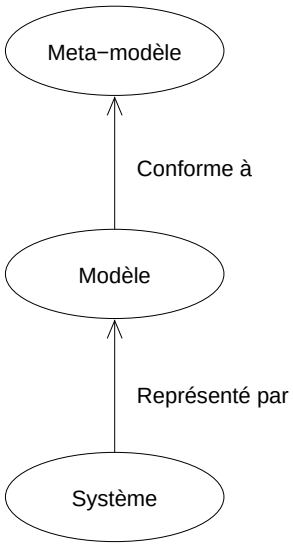


FIG. 2.8 – Notions de base de l'ingénierie des modèles

Un méta-modèle est aussi un modèle, il doit donc lui-aussi disposer d'un méta-modèle. Dans la majorité des cas, il s'agit du modèle le plus élevé de la hiérarchie qui se définit lui-même, il s'appelle le *méta-méta-modèle*.

Prenons un exemple concret pour illustrer les notions de système, modèle, méta-modèle et méta-méta-modèle, par exemple un vérin double-effet (ce vérin dispose donc d'une commande pour faire sortir la tige ainsi que d'une commande pour la faire rentrer). Le système considéré est donc notre vérin. Nous pouvons faire un modèle de ce système qui en représentera un aspect, par exemple un modèle des différentes positions que peut occuper ce vérin. Il faut alors choisir le méta-modèle adapté à cette représentation, les différents états du vérin peuvent être représentés par un automate par exemple. La description du méta-modèle se fait en utilisant le MOF (Méta Object Facility) [OMG, 2002] qui a été défini par l'OMG. L'exemple est illustré à la figure 2.9 qui permet de montrer la hiérarchie des modèles.

2.3.2 La notion de transformation

La définition de transformations entre les modèles est un concept clé au sein de l'ingénierie des modèles.

Les transformations peuvent avoir des objectifs divers. Une transformation peut par exemple être définie pour effectuer le raffinement du modèle de description d'un système lors du passage d'une étape de sa conception à une autre, c'est ce qui est fait dans la plupart des approches

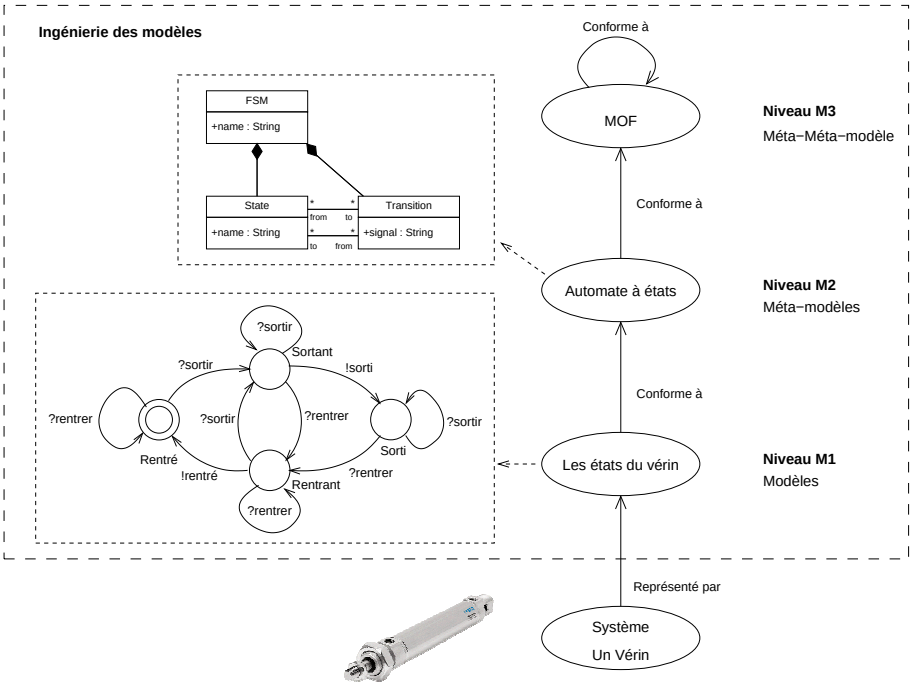


FIG. 2.9 – Exemple de modélisation d'un vérin

MDA où les différents modèles manipulés sont conformes au méta-modèle UML. Une transformation peut aussi permettre la vérification d'un modèle, c'est ce qui est réalisé dans [Bézivin and Jouault, 2005] où des contraintes sont vérifiées sur un modèle saisi. La sémantique d'exécution d'un modèle peut aussi être définie par une transformation de celui-ci vers un modèle exécutable.

Au niveau de la hiérarchie des modèles, comme le montre la figure 2.10, une transformation est généralement définie au niveau des méta-modèles, c'est-à-dire au niveau M2. Elle est exécutée sur des modèles : niveau M1. Une transformation peut généralement être considérée comme une opération atomique qui à partir d'un ensemble de modèles d'entrée, produit un ensemble de modèles de sortie.

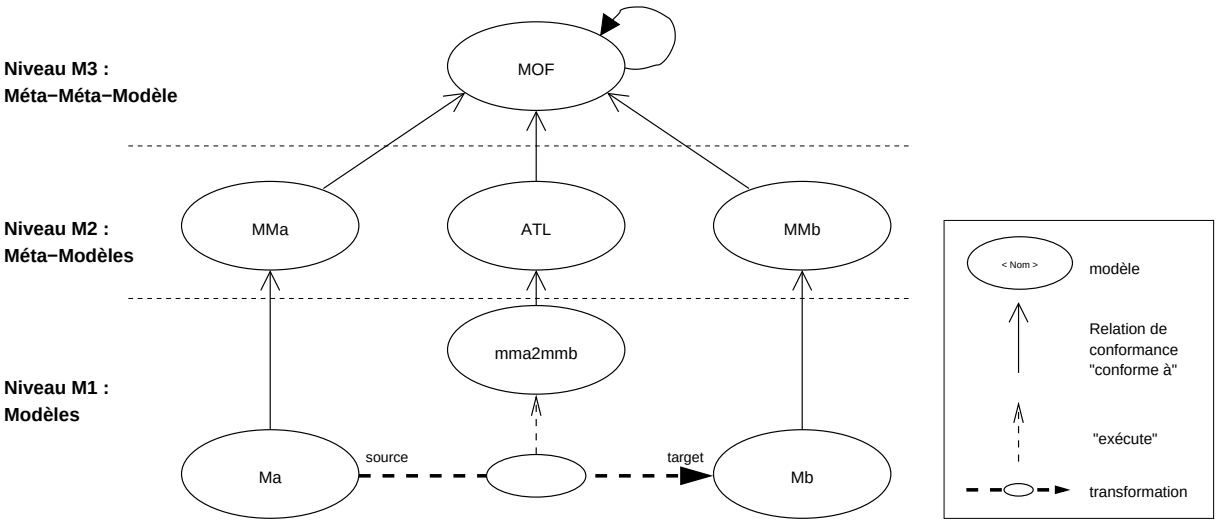


FIG. 2.10 – Hierarchisation des modèles et transformation

Des transformations peuvent aussi s'appliquer entre des modèles de niveau différents. C'est

ce qui est fait pour l'édition des méta-modèles. Ceux-ci sont d'abord décrits avec des modèles au niveau M1 pour être ensuite transformés en un modèle du niveau M2 par une transformation qui est définie entre le méta-modèle du modèle source (niveau M2) et le méta-méta-modèle (niveau M3) qui est le méta-modèle du modèle de destination.

2.3.3 La définition d'un DSL

Un DSL (Domain Specific Language) [Jouault et al., 2006a], est un langage défini pour un besoin spécifique dans un domaine particulier. Les DSL disposent généralement d'outils permettant de les manipuler. Plusieurs DSL peuvent être associés dans un cadre.

Les DSL peuvent être développés en utilisant différents moyens techniques. Les grammaires EBNF ont été d'abord utilisées. Récemment, l'utilisation des documents XML (eXtensible Markup Language) a permis la naissance d'une grande diversité de DSL. Les documents XML sont des fichiers décrivant des structures de données à l'aide d'une syntaxe utilisant des balises, les balises utilisées sont définies dans un schéma. La création de parsers pour charger les documents est assez simple, de plus un ensemble d'outils tels que le langage XSLT (eXtended Stylesheet Language Transformation) permettent de manipuler les documents XML. Des profils UML peuvent aussi être un moyen de définir des DSL, cela correspond à l'approche MDA proposée par l'OMG.

L'utilisation de l'ingénierie des modèles pour la définition des DSL permet de définir clairement le domaine, à l'aide du méta-modèle. La représentation du modèle (graphique ou textuelle) est réalisée à l'aide de transformations. Ces transformations étant généralement faites vers des formats extérieurs à la technologie de l'ingénierie des modèles en utilisant des injecteurs et des extracteurs.

2.3.4 Présentation d'une plate-forme pour l'ingénierie des modèles

Un exemple de plate-forme basée sur l'ingénierie des modèles et permettant la définition de DSL est la plate-forme AMMA [Bézivin et al., 2005]. Cette plate-forme a été développée au LINA (Laboratoire d'Informatique de Nantes Atlantique) par l'équipe ATLAS.

La plate-forme AMMA dont les composants sont présentés à la figure 2.11 est architecturée autour du langage de transformation de modèles générique ATL. Une syntaxe textuelle peut être associée à un modèle en utilisant TCS (Textual Concrete Syntax) [Jouault et al., 2006b] qui permet la génération d'injecteurs et d'extracteurs de et vers des documents respectant une grammaire EBNF (Extended Backus-Naur form). Les méta-modèles sont définis à l'aide de KM3 (Kernel M3) [Jouault and Bézivin, 2006]. KM3 est un DSL très simple, disposant d'une syntaxe concrète textuelle définie à l'aide de TCS ainsi que d'une transformation permettant d'obtenir méta-modèle à partir d'un modèle KM3.

Un DSL correspond donc, dans le cadre de la plate-forme AMMA à un méta-modèle, définissant les éléments du modèle et nommé DDM (Domain Definition Model), d'une syntaxe associée, définie à l'aide de TCS ou d'une autre transformation (pour une représentation graphique par exemple). Une ou plusieurs sémantiques d'exécution peuvent aussi être associées au DSL par une transformation vers un modèle exécutable.

Un DSL peut par exemple être défini pour les machines à état finis. Ce DSL serait composé du méta-modèle, définissant ce qu'est une FSM et que l'on a déjà présenté sur 2.9. La syntaxe concrète d'une FSM est définie par une transformation depuis et vers un autre DSL pour les dessins vectoriels. La sémantique d'exécution peut être définie par une transformation vers un programme informatique écrit en langage C simulant l'automate.

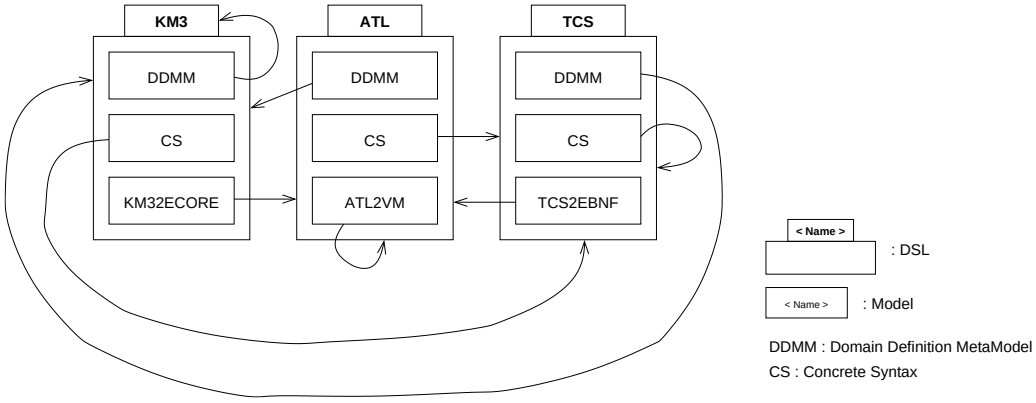


FIG. 2.11 – Les DSLs composant le cœur de la plate-forme AMMA

Le langage de transformation utilisé dans le cadre de la plate-forme AMMA est le langage ATL (Atlas Transformation Language) [Jouault and Kurtev, 2005]. Ce langage est implanté au travers d'un prototype fonctionnel. La définition d'une transformation se fait au travers d'un ensemble de règles. Les règles sont décomposées en deux motifs. Le motif *source* permet de définir sur quels éléments des modèles d'entrée la règle sera appliquée. Le motif *cible* de la règle décrit les éléments produits par la règle. Les expressions permettant de naviguer le modèle d'entrée sont écrites à l'aide du langage OCL [OMG, 2003b], un mécanisme de "helpers" permet d'associer une expression à une classe d'un méta-modèle. Ces expressions sont ensuite utilisées comme des méthodes dans les règles.

Le listing 2.1 présente un extrait d'une transformation de FSM vers un graphe d'état. Les deux premières lignes constituent l'en-tête de la transformation, celle-ci donne le nom du module ainsi que les modèles et méta-modèles en entrée et en sortie de la transformation. Un helper est défini à la ligne 4, il s'applique aux états des FSM et permet de déterminer si cet état est marqué ou non. Une règle est définie de la ligne 6 à la ligne 15. Cette règle s'applique sur les états de la FSM qui ne sont pas marqués et créé une place pour chacun d'entre eux.

Listing 2.1 – Exemple de règle ATL et de helper provenant de la transformation "FSM vers graphe d'état"

```
1  module FSM2PN;
2  create OUT : PN from IN : FSM;
3
4  helper context FSM!State def : notMarked : Boolean = not s.oclIsKindOf
       (FSM!MarkedState);
5
6  rule State2Place {
7    from
8      s : FSM!State (s.notMarked)
9    to
10     p : PN!Place (
11       name <- s.name,
12       net <- s.fsm,
13       initialMarking <- 0
14     )
15 }
```

La définition de nos DSL se fera à l'aide de la plate-forme AMMA.

2.4 Conclusion

Ce chapitre a présenté des solutions pour la représentation “haut-niveau” des systèmes dans les domaines de l’électronique et des systèmes automatisés de production. Les différentes caractéristiques de la conception des systèmes visés ont d’abord été abordées. Nous avons ensuite vu un ensemble de représentations de haut-niveau existantes telles que UML et les langages d’architecture. Pour finir, nous avons présenté une avancée récente dans le domaine du génie logiciel constituée par l’ingénierie dirigée par les modèles ainsi que l’approche DSL.

Les différentes approches de modélisation et d’exploitation proposées à la fois pour les systèmes de production et les systèmes électroniques permet de mettre en exergue les caractéristiques importantes de ces systèmes permettant de gérer leur complexité. Les concepts décrits et les préocupations qui vont avec devront être intégrés dans le modèle de haut niveau qui sera proposé et utilisé dans les chapitres suivants.

Parmi les représentations proposées, l’utilisation de langages de description d’architecture (ADL) est spécifique au domaine considéré (informatique ou système électronique) mais propose des concepts importants comme les notions de composant, ou celles d’architecture et de configuration. Le langage UML, quant-à-lui est trop vaste et sa définition peu précise rend son exploitation complexe pour nos besoins. UML est néanmoins un standard reconnu notamment en informatique et il paraît évident que la proposition de passerelles vers ce langages pour la représentation de nos systèmes est intéressante. Les modèles utilisées en automatique se limitent quant-à-eux à un aspect du système, son comportement par exemple. Ils ne sont pas adaptés à l’approche que nous souhaitons mettre en œuvre et dans laquelle nous souhaitons faire ressortir les différents aspects, mais se révéleront cruciaux lors des phases d’analyse.

L’utilisation de l’approche DSL en conjonction avec l’ingénierie des modèles telle qu’elle est préconisée par l’équipe ATLAS et implémentée dans la plate-forme AMMA constitue la solution la plus intéressante et donc celle que nous retiendrons. Ces technologies nous permettront de développer un ensemble de langages, utilisés lors des différentes phases de conception et adaptés à la description des systèmes sous l’angle de l’aspect reconfiguration. De plus, l’utilisation d’injecteurs et d’extracteurs permettra d’utiliser les modèles obtenus par transformation au sein d’outils pré-existants.

La deuxième partie de ce mémoire s’attache donc à la définition d’un langage de haut-niveau pour la description des systèmes reconfigurables, se basant sur les concepts présentés dans les deux premiers chapitre en utilisant une approche DSL, c’est-à-dire un ensemble de modèles métiers reposant sur une sémantique propre et entre lesquels des passerelles sont définies par l’intermédiaire de transformations de modèles. Le langage proposé devra être assez générique pour représenter à la fois les systèmes manufacturiers et électroniques. Il devra en outre proposer une distinction claire entre l’architecture et la configuration du système. Finalement un ensemble d’outils devra permettre son utilisation, aussi bien pour des analyses, pratiquées à l’aide d’outils académique, que pour son implantation à l’aide d’outils et de standards de l’industrie.

Deuxième partie

Une représentation de haut niveau pour les systèmes reconfigurables

Chapitre 3

Langage de description d'un système reconfigurable

Le chapitre 1 a introduit des caractéristiques communes aux différents systèmes reconfigurables étudiés dans la littérature. Ce chapitre propose une représentation de ces systèmes, dans les domaines des systèmes de production et de l'électronique. Cette représentation s'inspire des différents travaux présentés au chapitre 2. L'approche retenue pour la description est l'approche DSL présentée dans la section 2.3. Le langage proposé a été baptisé DeSyRe (Description de Systèmes Reconfigurables).

La première section de ce chapitre décrit l'organisation générale du modèle. Des descriptions formelles des éléments constituant l'architecture et les configurations sont introduites dans une deuxième section. La définition informatique du langage est ensuite présentée, suivie par la définition de la représentation visuelle associée. Finalement, deux systèmes provenant des domaines manufacturier et électronique sont modélisés à l'aide du langage défini afin d'illustrer son utilisation.

3.1 Principe de modélisation

Le langage proposé pour la description des systèmes reconfigurables se divise selon deux axes [Lamotte et al., 2005a] distingués sur la figure 3.1. Le premier axe, horizontal, sépare l'architecture du système de ses configurations pour respecter la distinction qui a été faite entre ces deux concepts dans le chapitre 1. Cette décomposition permet de définir deux modèles distincts : le premier définit les concepts propres à l'architecture et le second ceux qui correspondent à la configuration. La séparation de la description en deux modèles permet de faire correspondre à une même architecture, un ensemble de configurations différentes. La liaison entre ces modèles est réalisée en faisant référence aux éléments de l'architecture depuis les configurations.

L'axe vertical de décomposition du système est l'axe logique/physique, déjà distingué dans la méthodologie CASPAIM [Craye, 1994]. L'axe vertical sépare les aspects "produit", constituant la partie logique du système, des aspects "machine" et "transport" appartenant à la partie physique. Cette séparation apparaît à la fois au niveau de l'architecture et de la configuration.

Les *opérations* occupent un rôle central dans le modèle. En reprenant la définition donnée dans [Berruet, 1998], "Une opération est une fonction réalisée sur un produit par une ressource compte tenu de sa situation spatiale au sein du système". Ce sont donc les opérations qui sont au centre de la décomposition du système et qui relient la partie physique décrivant les ressources à la partie logique définissant les fonctions. Le lien entre l'architecture et la configuration est

aussi réalisé au travers des opérations définies dans l'architecture et sélectionnées dans la configuration.

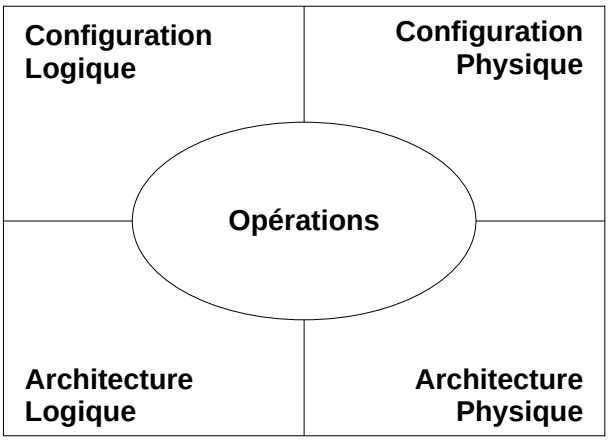


FIG. 3.1 – Organisation du modèle

3.1.1 Description de l'architecture

L'architecture constitue le socle du système reconfigurable. Celle-ci définit les éléments qui constituent le système ainsi que les moyens de communication entre ces éléments.

L'architecture est découpée verticalement, conformément à la figure 3.1, séparant la partie logique de la partie physique du système.

La partie logique décrit les *produits* d'un SAP ou les données d'un système électroniques. La notion de produit est donc un point central de cette partie. Un produit est un élément qui est transformé et/ou transporté sur le système. Il peut être introduit sur le système par son entrée, il est ensuite utilisé comme entrée des *séquences de fonctions*. Les séquences de fonctions décrivent les traitements subis par un produit en organisant les différentes *fonctions* qui lui sont appliquées. Un ensemble de nouveaux produits est obtenu à la fin de la séquence.

La partie physique de l'architecture décrit l'organisation du système en ressources. Une *ressource* est un élément du système réalisant des opérations. Le type de ces ressources est défini par rapport aux opérations qu'elles réalisent. La typologie des opérations proposée est adaptée de [Berruet, 1998] et présentée à la figure 3.2. Une *opération de transfert* réalise le déplacement d'un lieu à un autre par l'intermédiaire d'une *ressource de transport*. Une *opération de stockage* correspond au stockage d'un produit en attendant son utilisation. Quant aux *opérations de routage*, elles correspondent au déplacement d'un produit entre deux lieux caractéristiques d'une même ressource, ces lieux caractéristiques sont représentés par des ports, associés aux ressources. Deux types de ressources sont distinguées au niveau du modèle : les *ressources de transport* et les *ressources stationnaires* qui peuvent aussi bien réaliser des opérations de traitement, de stockage ou de routage.

La possibilité de réaliser le transfert unitaire d'un produits entre deux ressources stationnaires est modélisé par une *connexion*. Une connexion est reliée aux différentes ressources de transport qui peuvent la réaliser.

Les opérations définies au niveau de l'architecture ne sont que potentielles. En effet, lors de la définition de l'architecture, elles ne sont pas encore utilisées et indiquent qu'une ressource donnée peut réaliser cette opération qui implémente une fonction définie dans l'architecture logique. Les opérations potentielles relient la partie logique et la partie physique de l'architecture. Parmi toutes les opérations distinguées sur la figure 3.2, nous n'en avons explicité que

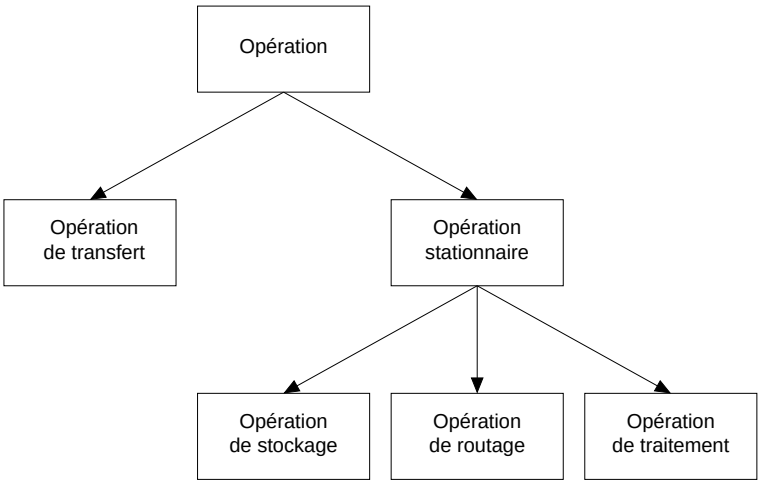


FIG. 3.2 – Typologie des opérations

deux types pour la définition des opérations potentielles : les opérations de stockage et de traitement. Les opérations représentant un transfert de produit sont implicitement définies par les connexions.

Un exemple de système décrit par le langage de description de l'architecture est donné à la figure 3.3. Sur cette figure sont représentés les différents éléments composant l'architecture physique, les ressources de transport (`R` et `Cv`), les ressources stationnaires (`IN`, `OUT`, `M1`, `M2`) ainsi que les différentes connexions (flèches). Le robot peut réaliser toutes les connexions alors que le convoyeur peut uniquement effectuer les transferts de `M1` à `M2`. L'entrée et la sortie du système sont représentées par les tampons `IN` et `OUT` qui sont modélisés comme des ressources de stockage. L'architecture logique est constituée de trois produits : `P1`, `P2` et `P3`. Deux séquences de fonction permettent d'obtenir ces produits en réalisant les fonctions `F1`, `F2` et `F3`. Chaque machine peut réaliser deux opérations de traitement. `F1` et `F2` peuvent être implémentées par les opérations `O1` et `O3`, sur `M1`. `F1` et `F3` peuvent être implémentées par `O2` et `O4` sur `M2`. Quant aux opérations de stockages, `ST1` permet le stockage de `P1` sur `IN`. `ST2` et `ST3` permettent quant à elles de stocker `P2` et `P3` sur `OUT`. Ce système est utilisé comme illustration pour la représentation informatique introduite à la section 3.3 et visuelle présentée à la section 3.4

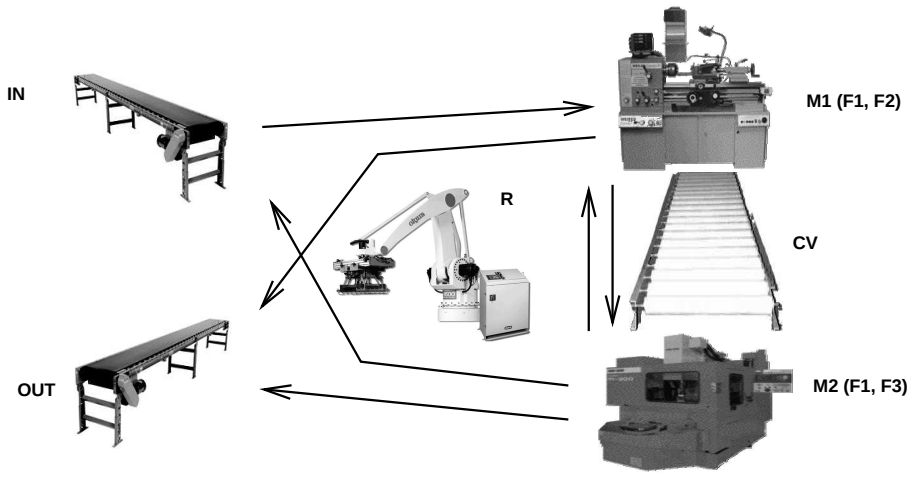


FIG. 3.3 – Description informelle de l'architecture "2Machines"

3.1.2 Description de la configuration

La configuration d'un système reconfigurable représente la partie du système qui évolue au gré des aléas, des modifications de l'environnement ou des modification des objectifs fixés pour le système.

La configuration est, tout comme l'architecture, décomposée selon l'axe logique/physique. La configuration logique reprend les fonctions utilisées en les instanciant. La configuration physique décrit les différents transferts utilisés ainsi que les ressources utilisées.

L'un des objectifs de la configuration physique est de décrire des chemins que peuvent emprunter les produits pour aller d'une machine à une autre. Ces chemins sont appelés *séquences de transferts*. Dans l'exemple des deux machines, dont l'architecture a été présentée à la figure 3.3, pour effectuer le transfert d'un produit de `IN` à `M2`, on peut utiliser directement le robot `R`, ce qui correspond à l'utilisation de la connexion `INM2`. Si maintenant la connexion `INM2` n'est plus disponible (un objet bloque le robot et il ne peut plus accéder à `M2` par exemple) il faut utiliser un autre chemin. Une seconde possibilité pour effectuer le transfert de `IN` vers `M2` est de déposer le produit en `M1` à l'aide du robot puis de le transférer en `M2` à l'aide du convoyeur, ce transfert nécessite l'utilisation de deux connexions.

Un autre objectif de la configuration est de décrire l'enchaînement des opérations pour réaliser des séquences de fonctions. Cet enchaînement appelé *séquence d'opérations* peut être rapproché des gammes opératoires définies dans le cadre du projet CASPAIM. Pour cela, deux types d'opérations sont définies : les opérations de transfert réalisent une séquence de transfert définie dans la configuration physique alors que les opérations stationnaires réalisent une opération potentielle définie dans l'architecture.

3.1.3 Bilan sur la description d'un système reconfigurable

Cette section a décrit de manière informelle les différents concepts mis en œuvre dans le langage pour la description des systèmes reconfigurables. La séparation entre l'architecture et la configuration constitue le premier concept mis en avant dans le modèle, tout comme la séparation entre les aspects physiques et logique. Le concept d'opération est central dans le modèle. Les concepts introduits dans cette section sont décrits formellement dans la section suivante.

3.2 Définition formelle d'un système reconfigurable

La définition informelle du langage de description des systèmes reconfigurable a permis d'introduire les concepts relatifs à ceux-ci.

Une définition formelle du langage permet de fixer les concepts manipulés dans le langage et d'expliciter les relations entre les éléments. Cette définition formelle de la description permettra au chapitre 4 de définir les analyses réalisées sur le modèle d'un système.

Cette définition est réalisée en définissant différents ensembles ainsi que des applications entre ces ensembles. Certains aspects de la description sont simplifiés, la notion de port n'a par exemple pas été prise en compte dans la formalisation, ce qui permet de conserver des définitions relativement claires tout en exprimant les concepts principaux des langages.

3.2.1 Définition formelle de l'architecture

$\mathcal{A} : \{\mathcal{A}p, \mathcal{A}l, \mathcal{O}, loc, stock\}$ est un quintuplet représentant l'architecture. $\mathcal{A}p$ désigne l'architecture physique, $\mathcal{A}l$, l'architecture logique, $\mathcal{O}$ l'ensemble des opérations et $loc : \mathcal{O} \times R \times F \rightarrow$

$\{0,1\}$ l'application décrivant où les fonctions peuvent être réalisées et quelles opérations les implémentent. $stock : \mathcal{O} \times R \times P \rightarrow \{0,1\}$ est une application décrivant où les opérations de stockage peuvent être réalisées et quels produits sont stockés.

$\mathcal{A}p$ est l'architecture physique, décrite par le couple $\{R, con\}$ où R représente l'ensemble des ressources de l'architecture et $con : R \times R \times R \rightarrow \{0,1\}$ l'application connexions. Une connexion relie deux ressources entre elles par l'intermédiaire d'une troisième ressource (une ressource de transport) qui réalise le transfert.

$\mathcal{A}l$ est l'architecture logique représentée par le sextuplet $\{F, G, P, pre, real, cons\}$ où F est l'ensemble des fonctions, G est l'ensemble des séquences de fonction et P l'ensemble des types de produits. $pre : G \times F \times F \rightarrow \{0,1\}$ est la relation de précédence entre les fonctions au sein d'une séquence de fonctions. $real : G \times P \rightarrow \{0,1\}$ définit l'affectation d'une ou plusieurs séquences de fonction à tout type de produit. De façon duale, $cons : G \times P \rightarrow \{0,1\}$ est la relation de consommation associant les produits consommés par une séquence de fonction à celle-ci.

3.2.2 Définition formelle de la configuration

$\mathcal{C} : (\mathcal{C}l, \mathcal{C}p, Os, opre, otran, oimp)$ représente une configuration, constituée par la configuration logique, la configuration physique, les séquences d'opérations ainsi qu'une relation de précédence entre les opérations dans une séquence : $opre$.

Le couple $\mathcal{C}l : (I, imp)$ décrit la configuration logique composée de l'ensemble I des instances des fonctions assurées par la configuration ainsi que de l'application $imp : I \times F \rightarrow \{0,1\}$ prenant la valeur 1 lorsque I instancie F.

$\mathcal{C}p : (T, S, trans, spare)$ est un quadruplet correspondant à l'architecture logique, dans laquelle T représente les séquences de transfert et S l'ensemble des ressources réservées pour la configuration mais non-utilisées. L'application $trans : T \times R \times R \times R \rightarrow \{0,1\}$ permet d'indiquer que le triplet de ressources représentant une connexion réalisée par une ressource de transfert participe à la réalisation de la séquence de transfert T. L'application $spare : S \times R \rightarrow \{0,1\}$ permet de dire quelles ressources de l'architecture sont réservées.

L'ensemble Os représente les différentes séquences d'opérations. L'application $otran : O \times T \rightarrow \{0,1\}$ associe une opération à une séquence de transfert. Quant-à l'association $opre : Os \times O \times O \rightarrow \{0,1\}$, c'est une relation de précédence entre deux opérations dans le contexte d'une séquence d'opérations. Finalement l'association $oimp : O \times I \rightarrow \{0,1\}$ associe une opération à une instance de fonction, indiquant que cette instance de fonction est précisément implémentée par l'opération.

3.2.3 Bilan sur la formalisation de la description

La description des systèmes reconfigurables a été formalisée. Pour pouvoir être utilisée par les outils d'ingénierie des modèles, la définition de différents DSL est nécessaire (pour l'architecture et pour la configuration). Cette définition est réalisée dans la section suivante.

3.3 Description informatique du système

L'utilisation de la description des systèmes reconfigurables par les outils d'ingénierie des modèles passe par la définition de DSL (Domain Specific Language) décrivant les modèles. Un DSL est défini par un méta-modèle et une syntaxe concrète. Des méta-modèles pour l'architecture et pour les configurations sont donc décrits dans cette section.

Les méta-modèles décrivent des concepts sous la forme d'un diagramme de classe, en établissant les relations entre ces différents concepts représentés par des classes. Il s'agit d'une implémentation des définitions mathématiques données précédemment.

3.3.1 DSL pour la description de l'architecture

Le méta-modèle de l'architecture est donné à la figure 3.4, les éléments sont explicités ensuite.

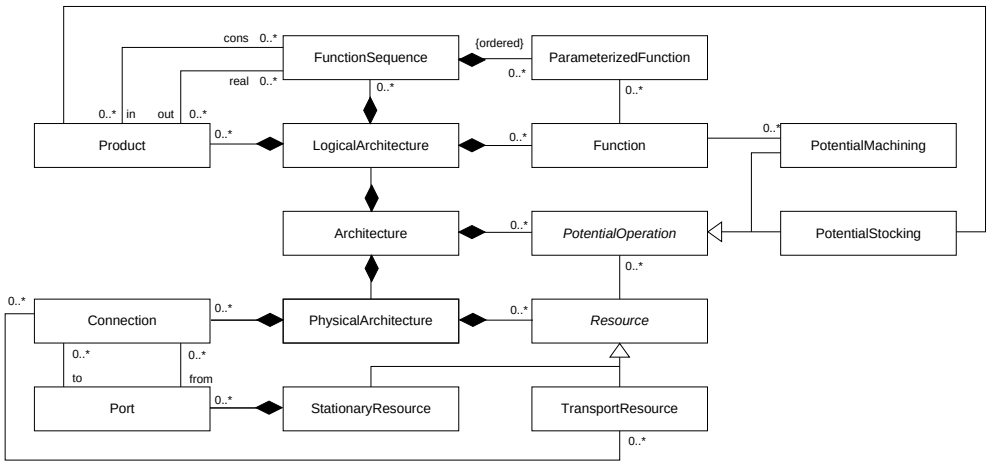


FIG. 3.4 – Méta-modèle de l'architecture

Les concepts définis dans la description formelle se retrouvent dans le méta-modèle présenté sur la figure 3.4. La classe `Architecture` représente l'architecture et possède trois relations, vers l'architecture logique et l'architecture physique représentées par les classes `LogicalArchitecture` et `PhysicalArchitecture`, ainsi qu'avec les opérations potentielles. L'architecture logique est composée des types de produits représentés par la classe `Product`, des séquences de fonctions représentées par la classe `FunctionSequence` et des fonctions représentées par la classe `Function`. La classe `ParameterizedFunction` permet d'associer des paramètres aux fonctions dès lors qu'elles sont dans une séquence.

Pour ce qui est de l'architecture physique, les deux types de ressources considérés (de transport et stationnaire) sont représentés par les classes `TransportResource` et `StationaryResource`. Les connexions sont représentées par la classe `Connection`. Au niveau du méta-modèle, les connexions relient les ressources par l'intermédiaire de leurs ports. La notion de port, représentée par la classe `Port` a été introduite afin de représenter des lieux caractéristiques pour les transferts au niveau des ressources stationnaires.

Les opérations potentielles, représentées par la classe `PotentialOperation` sont liées aux ressources sur lesquelles elles sont réalisées. Deux types d'opérations potentielles sont distinguées dans la modélisations. Il s'agit des opérations potentielles de traitement, liées aux fonctions et représentées par la classe `PotentialMachining`, les opérations potentielles de stockage sont liées à un produit et représentées par la classe `PotentialStocking`.

En plus du méta-modèle, une syntaxe concrète textuelle a été associée à l'architecture en utilisant l'outil TCS. Cette TCS complète la définition du DSL qui peut alors être saisi puis utilisé par des transformations de modèles. Ainsi l'exemple d'architecture donné à la figure 3.3 peut être représenté textuellement par la définition donnée en annexe A.1.1.

3.3.2 Définition d'un DSL pour la configuration

La description d'une configuration est réalisée par un deuxième DSL. Ici encore, il s'agit principalement de donner un méta-modèle pour la description de la configuration. Ce méta-modèle est représenté à la figure 3.5. On y retrouve les classes `LogicalConfiguration` et `PhysicalConfiguration` représentant les deux aspects de la configuration, ainsi que la classe `Operation` représentant les opérations qui sont organisées dans des `OperationSequence`.

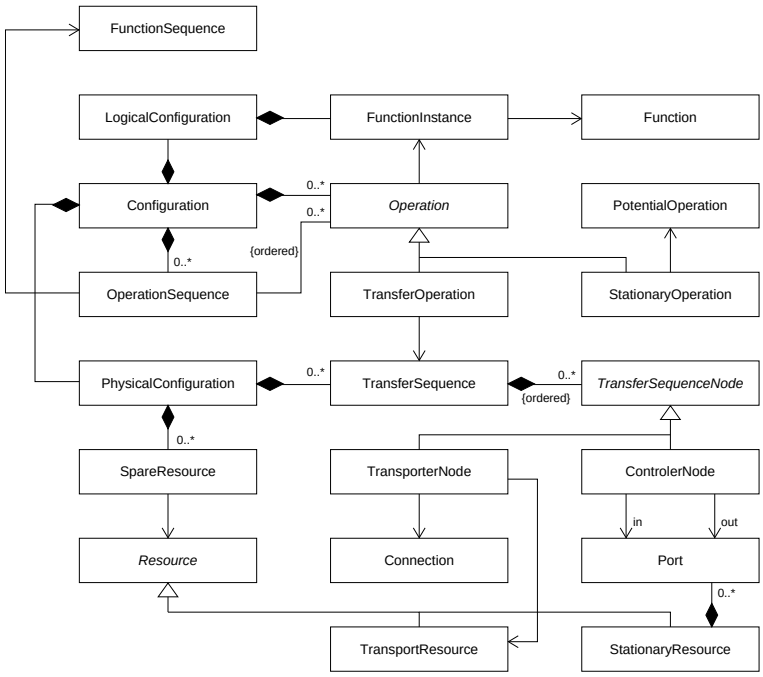


FIG. 3.5 – Meta-modèle pour la configuration

Au niveau de la configuration logique, une `FunctionInstance` est reliée à une `Function` (importée de l'architecture). Les `FunctionInstance` sont liées aux `Operation` qui les implémentent.

En ce qui concerne la configuration physique, une `SpareResource` correspond à une ressource réservée par la configuration mais non utilisée. La configuration physique est aussi composée des séquences de transfert représentées par la classe `TransferSequence`. Une séquence de transfert est composée de différents nœuds de la classe `TransferSequenceNode`. Deux types de nœuds sont distingués à ce niveau. Les nœuds de contrôle (`ControlerNode` et les nœuds de transport (`TransporterNode`. Les nœuds de contrôle décrivent un transfert au niveau d'une ressource stationnaire alors qu'un nœud de transport représente une connexion réalisée par une ressource de transport.

Comme pour l'architecture, une syntaxe concrète textuelle TCS a été associée à la configuration, un exemple de configuration pour l'exemple des deux machines peut être retrouvé en annexe.

3.3.3 Bilan sur la description informatique

Un langage a été défini pour représenter les systèmes reconfigurables, ce langage est constitué de deux DSL. Le premier permet de décrire une architecture, le second une configuration.

La séparation des deux langages permet de décrire, dans des modèles séparés, plusieurs configurations pour une même architecture.

Le langage de description nécessite néanmoins des moyens de représentation. Une première représentation, textuelle a été associée à l'aide de l'outil TCS. Une représentation visuelle est proposée à la section suivante.

3.4 Modèle de présentation visuel pour la description des systèmes reconfigurables

Parmi les différents types de modèles utilisés lors de la description d'un système, le *modèle de description*, représente le système à l'aide de concepts propres au domaine étudié, alors qu'un *modèle de présentation* est défini en terme d'éléments graphiques pour une description visuelle ou par une grammaire dans le cas d'une description textuelle et permet la visualisation, la saisie ou la modification du modèle de description.

La représentation textuelle pour le langage de description des systèmes reconfigurables a été définie à l'aide de l'outil TCS. Des exemples pour l'exemple des deux machines ont été fournis en annexe. Une représentation visuelle a aussi été définie. Notre choix s'est porté sur l'utilisation d'UML pour la définition de la représentation visuelle.

La représentation UML de la description permet sa saisie graphique pour peu que l'on dispose d'un éditeur UML. Cela rend aussi le modèle plus lisible. L'utilisation d'UML pour la saisie graphique répond à deux objectifs. Le premier objectif est de faciliter la prise en main du modèle par l'utilisateur. Le second objectif est de pouvoir s'intégrer dans une démarche de conception dans laquelle les étapes d'analyse des besoins se feraient en UML, un seul outil serait alors utilisé pour toute la conception. De plus UML étant un des piliers de l'approche MDA, ce langage est totalement intégré aux outils d'ingénierie des modèles que nous souhaitons pouvoir utiliser.

Une fois les diagrammes saisis, les modèles de l'architecture et des configurations sont obtenus par transformation de modèle. Celle-ci redonne alors leur sens aux éléments décrits graphiquement.

La description du système est principalement réalisée à partir des diagrammes de classe et d'activité UML. Des stéréotypes sont définis pour spécialiser les éléments de ces diagrammes afin de représenter les éléments du modèle de l'architecture ou des configurations. Le choix du diagramme de classes pour la représentation des éléments s'explique par le rôle central qu'ont les classes en UML, des références aux classes peuvent être faites depuis la grande majorité des éléments (typage, associations ...).

3.4.1 Organisation du modèle de présentation

Le modèle UML suit une organisation particulière représentée à la figure 3.6. L'architecture est placée dans un package portant le stéréotype «architecture». Ce package est lui-même décomposé en deux autres packages portant les stéréotypes «logicalarchitecture» et «physicalarchitecture» qui contiennent les représentations des architectures logiques et physiques. Les opérations potentielles sont directement définies au niveau de l'architecture (dans le package stéréotypé «architecture»).

Les configurations sont décrites dans le même modèle que l'architecture. Chaque configuration est définie dans son propre package, lui même décomposé en deux sous-packages pour

FIG. 3.6 – Organisation du modèle UML

la configuration physique et la configuration logique. Dans l'exemple présenté à la figure 3.6, deux configurations sont définies, la première s'appelle "standard" et la seconde "degrade".

Le lien entre le modèle de présentation visuel et le modèle de description est réalisé à l'aide de transformations de modèles. La figure 3.7 décrit les liens entre les différents modèles de description et de présentation pour la saisie et la modification de la description d'un système reconfigurable.

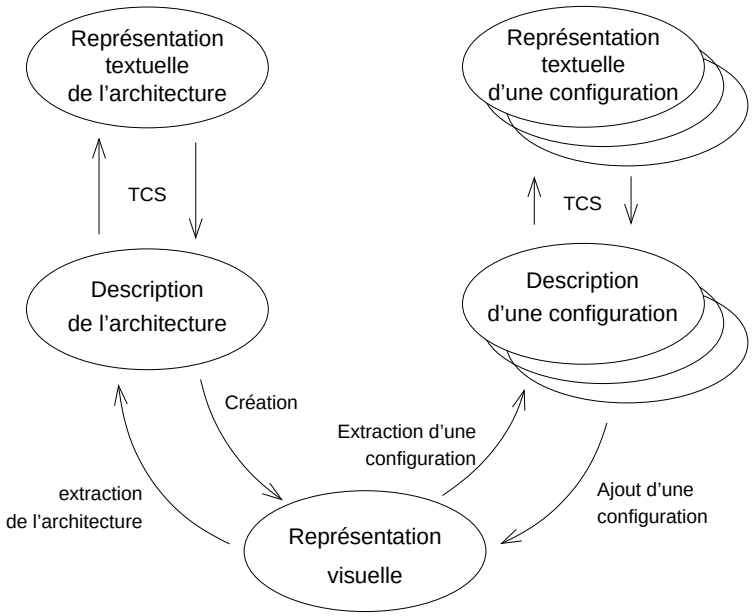


FIG. 3.7 – Gestion des modèles pour la description des systèmes reconfigurables

Deux transformations sont utilisées pour obtenir le modèle de description de l'architecture à partir de la représentation visuelle et vice-versa. Une autre transformation permet la sélection et l'extraction individuelle des configurations qui sont dans le modèle. Une quatrième transformation permet d'ajouter une configuration au modèle de présentation. Les liens avec les représentations textuelles sont aussi représentés.

3.4.2 Représentation visuelle de l'architecture

Représentation visuelle de l'architecture physique

L'architecture physique est décrite par un diagramme de classes UML. Les ressources sont représentées par des classes, celles portant le stéréotype «stationary» sont des ressources stationnaires, celles portant le stéréotype «transporter» sont des ressources de transport. Les ports associés aux ressources stationnaires sont représentés à l'aide d'une interface portant le stéréotype «port» et reliée avec la ressource par une composition. Les connexions sont aussi décrites par une interface portant le stéréotype «connection». Une association est dirigée du port d'entrée vers la connexion, une autre est dirigée de la connexion vers le port de sortie. Les ressources de transport sont reliées aux connexions par l'intermédiaire d'une dépendance (représentée par un trait discontinu dont l'extrémité porte une flèche pleine).

La figure 3.8 présente la représentation UML de l'architecture physique de l'exemple des deux machines. Les quatre ressources stationnaires "IN", "M1", "M2" et "OUT" sont représentées, ainsi que les deux ressources de transport "R" et "Cv" et les connexions liant toutes ces ressources. La connexion "M1M2" est reliée aux deux ressources de transport qui peuvent toutes les deux réaliser un transfert de "M1" vers "M2".

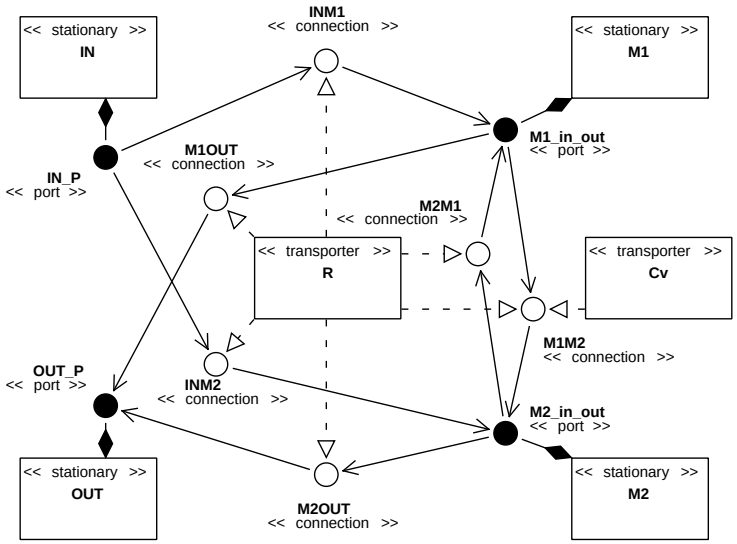


FIG. 3.8 – Représentation UML de l'architecture physique de l'exemple des deux machines

Représentation visuelle de l'architecture logique

L'architecture logique du système reconfigurable est représentée par un diagramme de classe définissant les éléments la constituant. Les séquences de fonctions étant ensuite décrites par des diagrammes d'activité, reprenant les autres éléments de l'architecture logique.

Au niveau du diagramme de classe de l'architecture logique, on retrouve les fonctions, définies par une classe portant le stéréotype «function». Ce diagramme permet aussi de définir les produits, représentés par des classes stéréotypées par «product». Les séquences de fonction sont elles aussi représentées par une classe stéréotypée «sequence».

La représentation UML de l'architecture logique de l'exemple des deux machines est donnée à la figure 3.9. Les trois fonctions plus une fonction de transfert générique, ainsi que les séquences de fonctions et les produits y sont décrits.

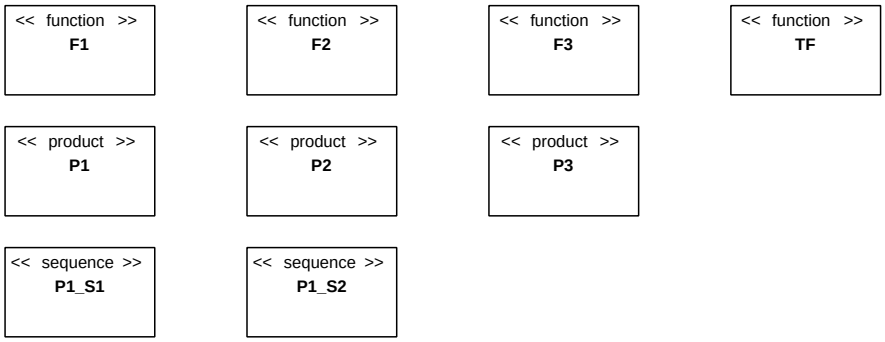


FIG. 3.9 – Représentation UML de l'architecture logique de l'exemple des deux machines

Les séquences de fonctions sont décrites par un diagramme d'activité qui leur est attaché. Ce diagramme d'activité est composé d'un ensemble d'activités chaînées, portant le stéréotype «function», chacune de ces activités est reliée à un `ObjectFlowState` UML, représenté par un rectangle et stéréotypé «function» et dont le type est la fonction réalisée. Les produits en entrée et en sortie de la séquence de fonction sont représentés par un object flow state portant le stéréotype «product» et dont le type est le produit représenté. Les produits en entrée sont reliés à la première activité de la séquence, et les produits en sortie à la dernière.

La figure 3.10 présente la description de la séquence de fonction `P1_S1` qui apparaît dans la figure 3.9. Cette séquence consomme un produit de type `P1`, réalise les fonctions `F1`, `F2` et `F3` pour aboutir aux produits `P2` et `P3`. Comme deux produits ont été obtenus à la fin de la séquence, on peut supposer que la séquence conduit à un désassemblage.

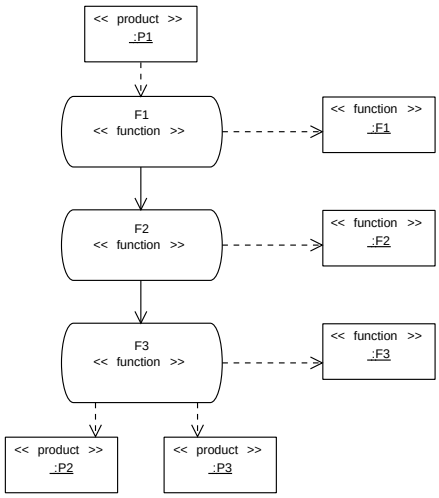


FIG. 3.10 – Représentation UML de la séquence P1_S1

Représentation visuelle des opérations

Les opérations potentielles sont définies dans un diagramme de classe au niveau du package "architecture". Elles sont représentées sous la forme d'une classe arborant le stéréotype «potentialoperation» ou «potentialstocking» suivant qu'il s'agit d'une opération de traitement ou une opération de stockage. Les opérations de traitement associent une ressource à une fonction, ces

deux éléments sont représentés à l'aide d'un attribut ayant respectivement pour nom "resource" et "function" et dont le type est l'élément correspondant. Pour les opérations de stockage, reliant une ressource à un produit, le même mécanisme est utilisé, avec les noms "resource" et "product".

Les opérations potentielles définies dans le cadre de l'exemple des deux machines ont été représentées à la figure 3.11. On retrouve quatre opérations potentielles ainsi que trois opérations de stockage pour les trois produits.

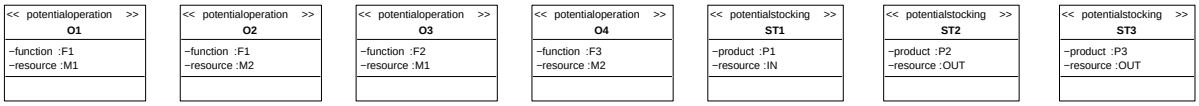


FIG. 3.11 – Les opérations potentielles définies dans l'exemple des deux machines

3.4.3 Représentation visuelle de la configuration

Représentation de la configuration logique

La configuration logique réside dans un package portant le stéréotype «logicalconfiguration». Un diagramme de classe permet de définir les différentes instances de fonctions décrites avec des classes portant le stéréotype «functioninstance».

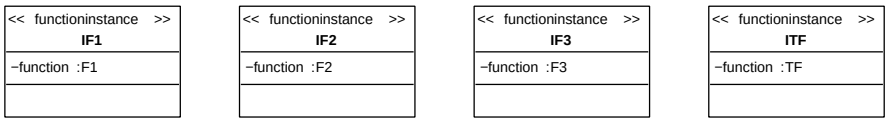


FIG. 3.12 – Diagramme de configuration logique

Représentation de la configuration physique

La configuration physique contient la définition des séquences de transfert ainsi que des ressources réservées. Ces définitions sont contenues dans un package portant le stéréotype «physicalconfiguration».

Une séquence de transfert est définie par une classe portant le stéréotype «transfer». La description de la séquence de transfert se fait dans un diagramme d'activité associé à cette classe. Chaque nœud de la séquence est représenté par une activité portant le stéréotype «connection» pour les nœuds de transfert et «stationary» pour les nœuds de contrôle. A chaque type de nœud sont associés des éléments de l'architecture par l'intermédiaire d'`ObjectFlowState`. Aux nœuds de transfert sont associés une connexion ainsi qu'une ressource de transfert représentée par un `ObjectFlowState` portant les stéréotype «connection» ou «transporter». Aux nœuds de contrôle sont associés le port source et le port destination du contrôle, les deux `ObjectInState` représentant ces deux ports étant respectivement stéréotypés «inport» et «outport».

La définition des ressources réservées se fait à l'aide d'une classe stéréotypée «spare» contentant un attribut nommé "resource" dont le type est la ressource en provenance de l'architecture.

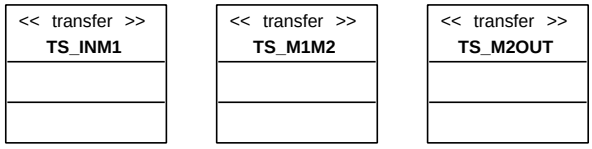


FIG. 3.13 – Diagramme de configuration physique

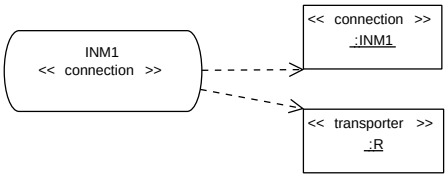


FIG. 3.14 – Description d'une séquence de transfert

Opérations et séquences d'opérations

Les opérations réalisées dans la configuration ainsi que les séquences d'opérations sont décrites dans un diagramme de classe défini au niveau du package de la configuration.

Les opérations de transfert sont définies par une classe stéréotypée «transferop» portant un attribut nommé "transfer" pointant vers la séquence de transfert et un autre nommé "function" pointant vers l'instance de fonction correspondant.

Les opérations stationnaires sont définies par une classe stéréotypée «stationaryop» portant un attribut nommé "potop" pointant vers l'opération potentielle ainsi qu'un attribut nommé "function" pointant vers l'instance de fonction associée.

Un exemple de diagramme définissant les opérations utilisées dans la configuration est donné à la figure 3.15

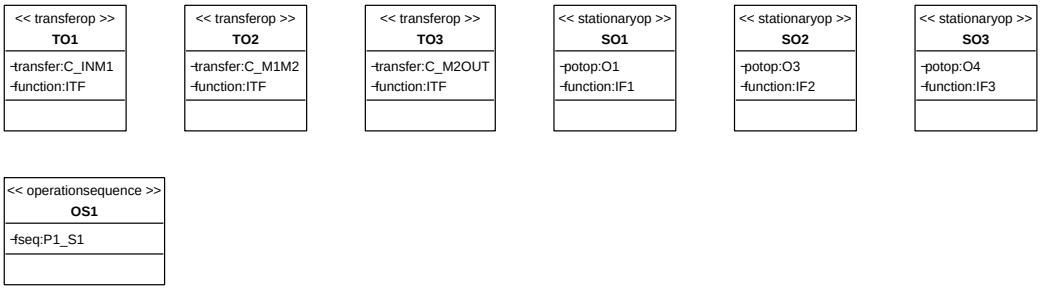


FIG. 3.15 – Diagramme de description des opérations

Les séquences d'opérations sont décrites par une classe stéréotypée «operationsequence» portant un attribut nommé "fseq" dont le type est la séquence de fonctions implémentée. La définition de la séquence est réalisée dans un diagramme d'activité associé à la classe la décrivant. Ce diagramme d'activité contient des nœuds représentant les opérations et portant le stéréotype «operation», ces nœuds pointent vers l'opération réalisée grâce à un `ObjectFlowState` stéréotypé «operation» et dont le type est cette opération. Un exemple de séquence, définie pour la séquence d'opération `OS1` est présenté à la figure 3.16.

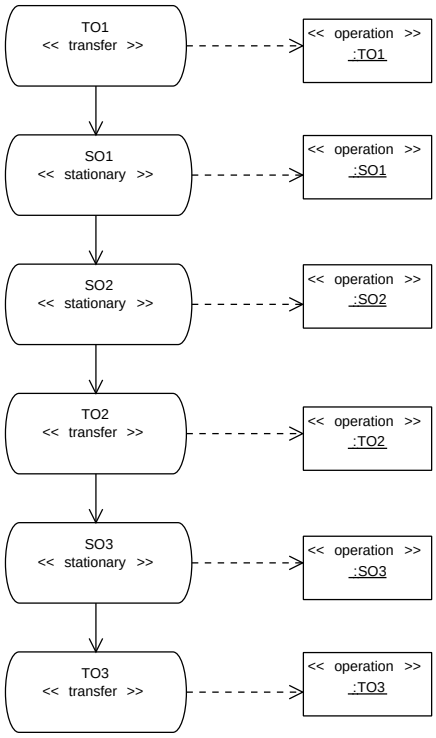


FIG. 3.16 – Description d'une séquence d'opérations

3.4.4 Bilan sur la représentation visuelle

La représentation visuelle facilite la prise en main et la lecture du langage de description. Certains aspects de la représentation visuelle sont assez lourds. Pour cette raison, cette représentation est associée par la suite à la représentation textuelle pour décrire le système de la manière la plus efficace possible.

Les deux représentations sont utilisées dans la section suivante pour décrire des systèmes reconfigurables provenant des SAP et des systèmes électroniques.

3.5 Projection de la description sur les domaines métier

Le modèle de description de haut niveau vient d'être décrit. Comme il a été défini pour permettre la représentation des systèmes électroniques et des systèmes de production, cette partie illustre le modèle sur un exemple complet dans chacune des deux disciplines afin de mettre en évidence des correspondances sémantiques entre le langage, les SAP et les systèmes électroniques.

Pour chacun des deux exemples, l'architecture du système sera présentée ainsi que deux configurations.

3.5.1 Projection sur les systèmes de production

Le premier type de système que nous avons modélisé était un système de production. La modélisation d'un tel système est présentée dans cette section. Après une rapide présentation de l'exemple, l'architecture du système est présentée puis deux configurations possibles pour cette architecture.

Présentation de l'exemple

L'exemple de système de production retenu est un convoyeur industriel, introduit dans [Berruet, 1998] et représenté à la figure 3.17. Il est constitué de 6 zones Z1-6. Trois ressources d'usinage et deux tampons (IN et OUT) sont accessibles via un convoyeur (Cv) et quatre robots, les trois premiers R11-13 chargent les machines M1-3 alors que R14 est dédié au chargement et au déchargement du convoyeur, il permet d'accéder à la zone Z1 ainsi qu'aux zones Z2 et Z4, cette faculté permet au système de continuer à fournir des services lorsque le convoyeur tombe en panne.

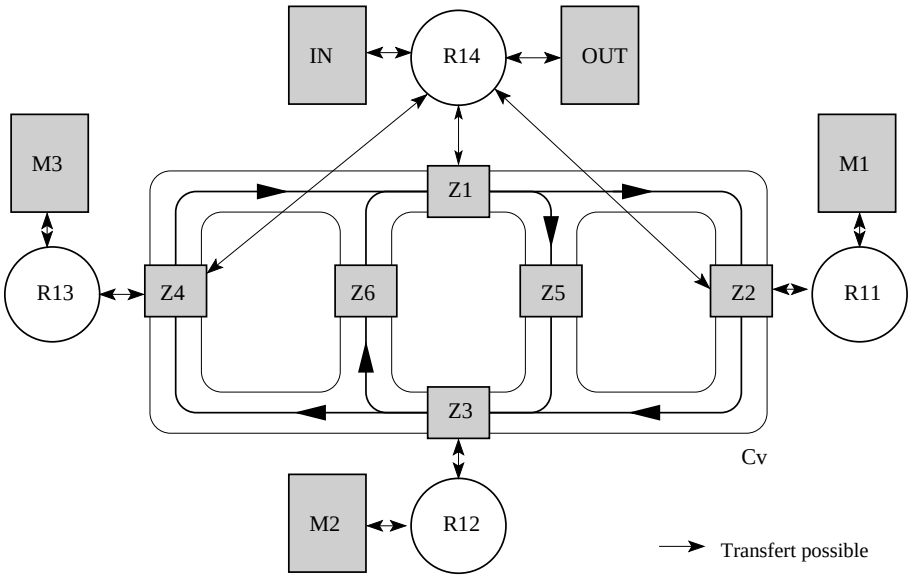


FIG. 3.17 – Description informelle de l'architecture du convoyeur

Trois fonctions d'usinage ont été définies sur cette architecture, F1, F2 et F3. Elles interviennent dans la réalisation de trois produits, décrits par les gammes logiques S1 (F1, F2, F3), S2 (F1, F2), S3 (F1, F3). Ces gammes logiques sont réalisées à partir du même produit P0.

Toutes les machines ne peuvent réaliser toutes les fonctions. La fonction F1 est implémentée par deux opérations, réalisées sur M1 et M2. F2 est implémentée par une opération réalisée sur M3 alors que F3 est implémentée par trois opérations sur les trois machines.

IN et OUT sont respectivement les buffers d'entrée et de sortie. P0 entre par IN où il peut être stocké, les produits finis sont évacués sur OUT, qui fournit aussi une capacité de stockage.

Description de l'architecture

L'architecture logique est décrite dans un premier temps pour illustrer les capacités du système. Les trois fonctions d'usinage (F1, F2, F3) ainsi qu'une fonction de transport générique (TF) sont définies. Les séquences S1, S2 et S3 sont aussi définies ainsi que les produits qu'elles permettent d'obtenir. La définition de l'architecture est donnée sous sa forme textuelle au listing 3.1.

Listing 3.1 – Description textuelle de l'architecture logique du convoyeur

```
1 logical architecture {
2   function TF;
3   function F1;
4   function F2;
5   function F3;
```

```
6
7     product P0;
8     product P1;
9     product P2;
10    product P3;
11
12    sequence S1 [P0 -> P1] : F1, F2, F3;
13    sequence S2 [P0 -> P2] : F1, F2;
14    sequence S3 [P0 -> P3] : F1, F3;
15  }
```

L'architecture physique du système, sur laquelle sont réalisés les services proposés par l'architecture logique est ensuite décrite. Il est nécessaire de choisir à quel niveau d'abstraction cette description est faite. Les différents sous-systèmes sont modélisés à des niveaux de granularité différents suivant leur importance. Au niveau des ressources de transport, deux types de ressources sont décrites : le convoyeur et les robots. Le convoyeur étant la partie centrale du système, nous avons choisi de décomposer sa description en zones (le convoyeur aurait pu être modélisé comme une boîte noire). Chaque robot est représenté par les transferts qu'il peut réaliser. Au niveau des ressources stationnaires, leur constitution n'est pas décrite, une ressource stationnaire correspond à chaque machine, il en est de même pour les deux buffers.

La représentation UML de l'architecture physique du convoyeur est donnée à la figure 3.18. Chaque zone a été représentée par une ressource stationnaire possédant deux ports, des connexions ont été établies entre ces ports conformément aux transferts que le convoyeur peut réaliser. Le convoyeur a été modélisé par une ressource de transport pouvant réaliser les différentes connexions entre les zones. Chaque machine a été modélisée par une ressource stationnaire, un port représentant le lieu où sont pris et déposés les produits. Afin de représenter le chargement et le déchargement des machines par les robots, des connexions ont été établies entre la zone permettant d'accéder à la machine et la machine elle-même, une connexion a été utilisée pour représenter chaque sens dans lequel le transfert peut avoir lieu. Une ressource de transport a été utilisée pour représenter le robot, elle permet de réaliser les deux connexions. Pour les buffers d'entrée/sortie, nous avons procédé de la même manière que pour les machines, mais cette fois-ci le robot de chargement pouvant réaliser des transferts vers d'autres zones, nous avons alors utilisé les connexions réalisées par le convoyeur ainsi que des nouvelles connexions que celui-ci ne pouvait pas traiter.

Le concept d'opération des systèmes de production, dans le cas des opérations d'usinage et de stockage, est traduit directement au niveau de la modélisation par l'utilisation d'opérations potentielles dans l'architecture. Ainsi, six opérations potentielles d'usinage sont définies. Ainsi que quatre stockages potentiels. Ces opérations sont définies au listing 3.2.

Listing 3.2 – Description textuelle des opérations

```
1  operation OpM1_F3 (F3,M1);
2  operation OpM2_F3 (F3,M2);
3  operation OpM3_F2 (F2,M3);
4  operation OpM2_F1 (F1,M2);
5  operation OpM1_F1 (F1,M1);
6  operation OpM3_F3 (F3,M3);
7
8  stocking OpIN_P0 (P0, IN);
9  stocking OpOUT_P1 (P1, OUT);
10 stocking OpOUT_P2 (P2, OUT;)
11 stocking OpOUT_P3 (P3, OUT);
```

Ainsi, l'architecture du convoyeur a été complètement décrite à l'aide du modèle proposé dans ce chapitre. Un choix de modélisation est nécessaire quant à la granularité à laquelle cette modélisation est réalisée. On remarque aussi que certains attributs qui pourraient être associés aux éléments du modèle (capacités, temps de traitement) ne l'ont pas été. Ces attributs pour-

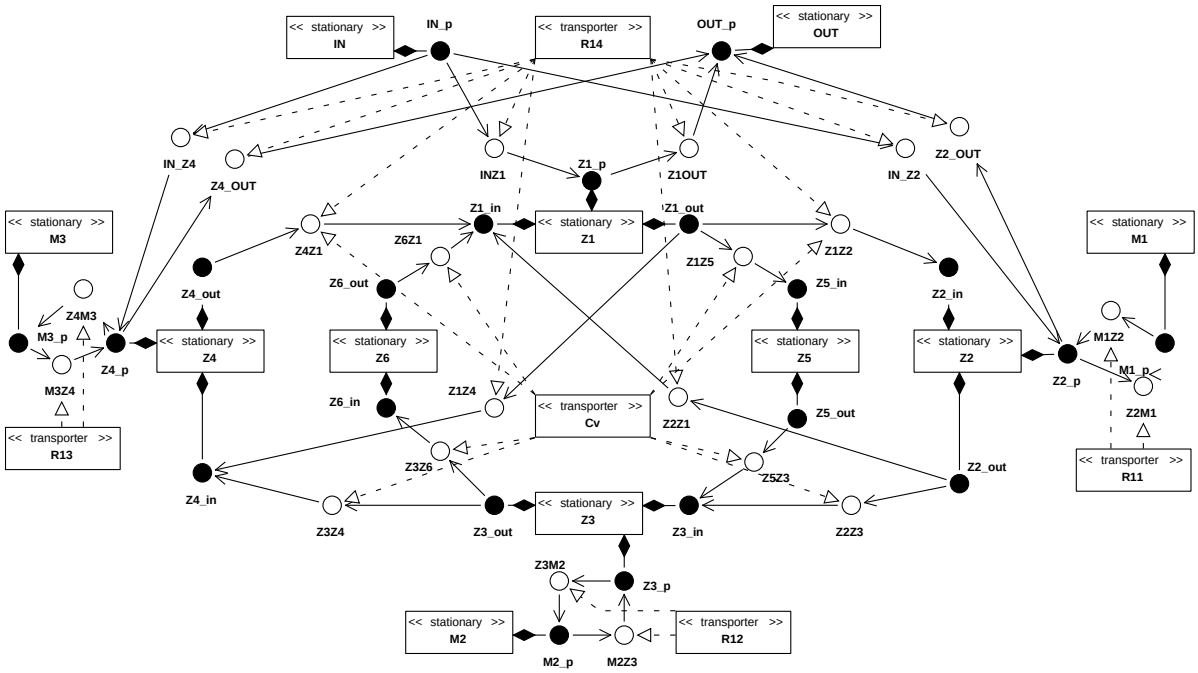


FIG. 3.18 – Représentation UML de l'architecture physique du convoyeur

raient faire l'objet d'une extension de la modélisation spécifique au domaine considéré, ici les systèmes de production.

Description d'une première configuration pour le convoyeur

La première configuration du convoyeur, CV1, permet la réalisation du produit P1. Cette configuration utilise les trois machines afin de minimiser le temps de cycle.

La première étape dans la conception d'une configuration consiste à décrire la configuration logique retenue. La configuration logique décrit les différentes instances de fonction qui sont utilisées dans la configuration, pour la configuration CV1, les quatre fonctions définies dans l'architecture doivent être réalisées. Leur définition est donnée sur le listing 3.3.

Listing 3.3 – Partie logique de la configuration CV1

```
1 logical configuration {
2   instance IFT of FT;
3   instance IF1 of F1;
4   instance IF2 of F2;
5   instance IF3 of F3;
6 }
```

Vient ensuite le choix des machines sur lesquelles sont réalisées les opérations implémentant les fonctions définies. Ayant choisi d'effectuer une opération par machine, la première contrainte dans ce choix concerne la réalisation de la fonction F2 qui ne peut être implémentée que par OpM3_F2. Le choix de l'emplacement des deux autres fonctions est ensuite réalisé. Pour limiter les transferts, il semble intéressant d'utiliser les opérations OpM2_F1 et OpM1_F3.

Les séquences de transfert permettant de réaliser la séquence de fonctions sont ensuite définies. Cette définition est proposée au listing 3.4. La première séquence effectue un transfert de IN vers M2. Le second transfert fait transiter les produits de M2 à M3. La dernière opération stationnaire étant réalisée sur M1, une autre séquence de transfert est définie pour aller de M3 à M1. Finalement on atteint la sortie en réalisant un transfert de M1 à OUT.

Listing 3.4 – Partie physique de la configuration CV1

```
physical configuration {
  transfer sequence TS_INM2 {
    transfer INZ1 using R14;
    control from Z1.Z1_p to Z1.Z1_out;
    transfer Z1Z5 using Cv;
    control from Z5.Z5_in to Z5.Z5_out;
    transfer Z5Z3 using Cv;
    control from Z3.Z3_in to Z3.Z3_p;
    transfer Z3M2 using R12;
  }
  transfer sequence TS_M2M3 {
    transfer M2Z3 using R12;
    control from Z3.Z3_p to Z3.Z3_out;
    transfer Z3Z4 using Cv;
    control from Z4.Z4_in to Z4.Z4_p;
    transfer Z4M3 using R13;
  }
  transfer sequence TS_M3M1 {
    transfer M3Z4 using R13;
    control from Z4.Z4_p to Z4.Z4_out;
    transfer Z4Z1 using Cv;
    control from Z1.Z1_in to Z1.Z1_out;
    transfer Z1Z2 using Cv;
    control from Z2.Z2_in to Z2.Z2_p;
    transfer Z2M1 using R11;
  }
  transfer sequence TS_M1OUT {
    transfer M1Z2 using R11;
    transfer Z2OUT using R14;
  }
}
```

Les opérations et de la séquence `S1`, sont finalement définies et présentées sur le listing 3.5.

Listing 3.5 – Opérations définies dans CV1

```
stationary operation S_F1 : IF1 using OpM2_F1;
stationary operation S_F2 : IF2 using OpM3_F2;
stationary operation S_F3 : IF3 using OpM1_F3;

transfer operation T_INM2 : IFT using TS_INM2;
transfer operation T_M2M3 : IFT using TS_M2M3;
transfer operation T_M3M1 : IFT using TS_M3M1;
transfer operation T_M1OUT : IFT using TS_M1OUT;

operation sequence OS_S1 realize S1 : T_INM2, S_F1, T_M2M3, S_F2,
    T_M3M1, S_F3, T_M1OUT;
```

Description d'une seconde configuration pour le convoyeur

La seconde configuration nommée CV2, permet la réalisation de la même séquence que la première, sans utiliser le convoyeur. La configuration logique est donc identique et est présentée au listing 3.6.

Listing 3.6 – Partie logique de la configuration CV2

```
logical configuration {
  instance IFT of FT;
  instance IF1 of F1;
  instance IF2 of F2;
  instance IF3 of F3;
}
```

Le convoyeur n'étant pas utilisé, La machine M2 n'est plus accessible, il faut utiliser les opérations disponibles sur les autres machines : OpM1_F1, OpM3_F2 et OpM3_F3. Les transferts

se font entre IN et M1 puis entre M1 et M3 et finalement entre M3 et OUT, tous ces transferts sont réalisés en utilisant les différents robots, leur définition est précisée sur le listing 3.7.

Listing 3.7 – Partie physique de la configuration CV2

```
physical configuration {
  transfer sequence TS_INM1 {
    transfer INZ2 using R14;
    transfer Z2M1 using R11;
  }
  transfer sequence TS_M1M3 {
    transfer M1Z2 using R11;
    transfer Z2Z1 using R14;
    transfer Z1Z4 using R14;
    transfer Z4M3 using R13;
  }
  transfer sequence TS_M3OUT {
    transfer M3Z4 using R13;
    transfer Z4OUT using R14;
  }
}
```

Vient ensuite la définition des opérations et de la séquence `S1`, présentées sur le listing 3.8.

Listing 3.8 – Opérations définies dans CV2

```
stationary operation S_F1 : IF1 using OpM1_F1;
stationary operation S_F2 : IF2 using OpM3_F2;
stationary operation S_F3 : IF3 using OpM3_F3;

transfer operation T_INM1 : IFT using TS_INM1;
transfer operation T_M1M3 : IFT using TS_M1M3;
transfer operation T_M3OUT : IFT using TS_M3OUT;

operation sequence OS_S1 realize S1 : T_INM1, S_F1, T_M1M3, S_F2, S_F3
    , T_M3OUT;
```

3.5.2 Projection sur les systèmes électroniques

Un point fort du langage proposé est sa généricité. Cette section montre comment il peut être utilisé pour représenter un système électronique.

Pour ce type de système, les produits représentent les données traitées. Les fonctions sont les traitements réalisés. Les ressources stationnaires représentent des nœuds de traitement ou des mémoires et les ressources de transport les liens entre ces ressources. On peut, comme c'était le cas pour les SAP, modéliser des systèmes de communication plus complexes en utilisant une conjonction de ressources stationnaires et de ressources de transport.

Présentation du système modélisé

Le système modélisé est un système de traitement vidéo composé de quatre cartes électroniques et représenté à la figure 3.19. Les quatre cartes réalisent les fonctions suivantes : la première carte se charge de l'acquisition vidéo et transmet les images à la seconde qui effectue des traitements réalisés en temps réel. Ces traitements permettent d'identifier des images nécessitant un traitement particulier. Ces images sont alors stockées sur la troisième carte et traitées sur la quatrième à l'aide d'algorithmes plus complexes, ce dernier traitement aboutit à une image annotée et éventuellement à une alarme. Les quatre cartes peuvent communiquer entre elles de deux manières différentes. La première manière est un bus partagé par les quatre cartes. La seconde est une liaison dédiée entre les deux premières cartes. Cette liaison haut-débit permet de faire un transfert rapide entre les deux cartes.

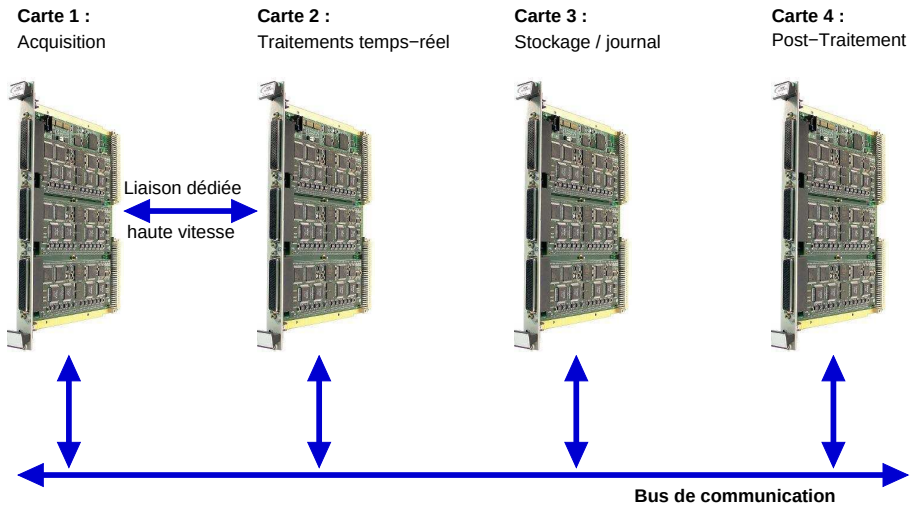


FIG. 3.19 – Description informelle du système électronique

Plusieurs configurations peuvent être envisagées pour ce système. Dans une première configuration : Eon1, les quatre cartes sont utilisées. La communication entre la carte d'acquisition et la carte de traitement temps-réel se fait en utilisant la liaison dédiée. Les autres communications se font à travers le Bus. La configuation Eon2 n'utilise que les deux premières cartes, seul le traitement temps-réel est alors réalisé.

Différents éléments du système peuvent tomber en panne. Une panne de la carte de traitement temps-réel entraîne son remplacement par la carte de post-traitement, bien sûr, la cadence sera moindre vu qu'aucune liaison dédiée n'a été établie entre la carte d'acquisition et la carte de post-traitement. Le bus peut aussi tomber en panne, dans ce cas aucun post-traitement n'est réalisé, aucun historique n'est conservé non plus, les deux premières cartes fonctionnent seules.

Architecture

L'architecture logique est décrite dans un premier temps. Il faut identifier les données manipulées qui constitueront les produits de l'architecture. Le produit "image brute", "image traitée", "image annotée", "alarme" sont alors définis. Il faut aussi identifier les différentes fonctions qui peuvent être réalisées : "acquisition", "traitement grossier", "traitement fin". Ces fonctions sont ensuite organisées en séquences.

Une première séquence est chargée de l'acquisition de l'image et de son traitement grossier afin d'obtenir l'image traitée. Une seconde séquence partant de l'image traitée réalise un traitement fin pour obtenir une image annotée (insérée dans l'historique) ainsi qu'une alarme. L'architecture logique ainsi obtenue est présentée sur lisitng 3.9

Listing 3.9 – Description textuelle de l'architecture logique du système électronique

```
1  logical architecture {
2    function transfert;
3    function acquisition;
4    function traitement_grossier;
5    function traitement_fin;
6
7    product image_brute;
8    product image_traitee;
9    product image_annotee;
10   product alarme;
11
12   sequence choix_image [image_brute -> image_traitee] : acquisition,
       traitement_grossier;
```

```
13    sequence annotation [image_traitee -> image_annotee, alarme] :
          traitement_fin;
14  }
```

Le modèle ne permet pas de décrire le contrôle dans la définition des séquences de fonctions. Ainsi, seule une partie des images traitées est conservée afin d'être analysée plus finement mais le choix de ces images n'apparaît pas dans le modèle.

Au niveau de l'architecture physique, sa représentation visuelle est fournie à la figure 3.20. La description de l'exemple suggère que chaque carte peut être considérée comme une ressource de traitement. La liaison dédiée peut être modélisée par une simple connexion bi-directionnelle entre les deux ressources concernées. Pour ce qui est du bus, plusieurs modélisations sont possibles. Nous avons choisi de modéliser le bus comme une ligne de transfert bi-directionnelle constituée de quatre nœuds en liaison directe avec chacune des cartes.

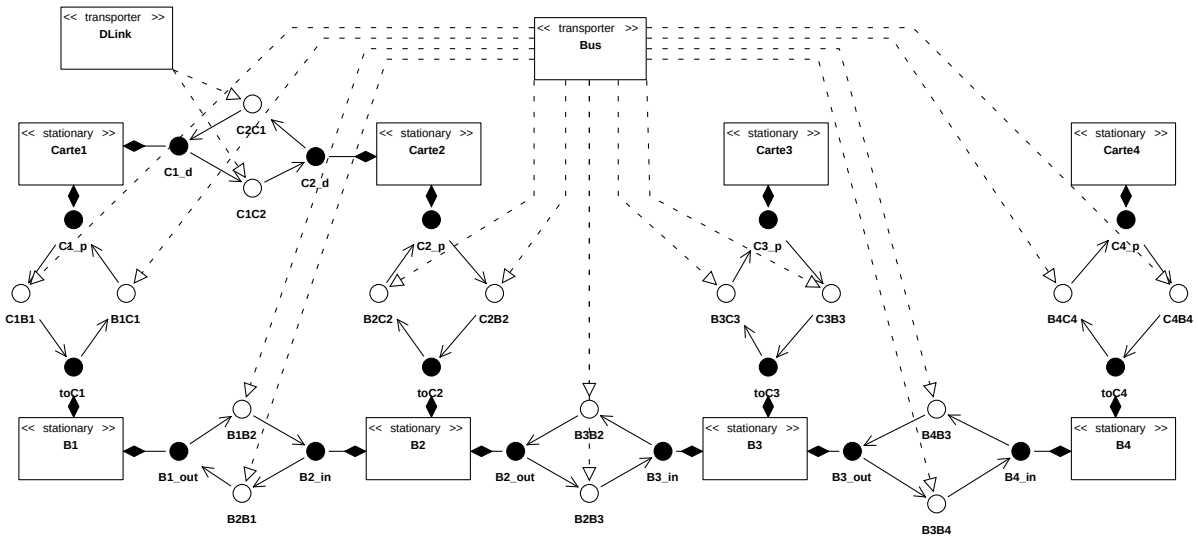


FIG. 3.20 – Représentation visuelle de l'architecture physique du système électronique

Au niveau des opérations potentielles, les capacités de chacune des cartes sont décrites. Les définitions des différentes opérations sont données au listing 3.10.

Listing 3.10 – Opérations potentielles définies pour le système électronique

```
1  operation C1Acq (acquisition, Carte1);
2  operation C2trf (traitement_fin, Carte2);
3  operation C2trg (traitement_grossier, Carte2);
4  operation C4trf (traitement_fin, Carte4);
5  operation C4trg (traitement_grossier Carte4);
6
7  stocking C1Brute (image_brute, Carte1);
8  stocking C2Traitee (image_traitee, Carte2);
9  stocking C3Traitee (image_traitee, Carte3);
10 stocking C3Annotee (image_annotee, Carte3);
11 stocking C3Alarme (alarme, Carte3);
12 stocking C4Traitee (image_traitee, Carte4);
```

Une première configuration pour le système électronique

La configuration Eon1 définie pour le système de traitement d'images utilise au maximum les potentialités du système pour obtenir une QdS maximale.

Les deux traitements définis sont alors réalisés. La configuration logique est donc constituée des instances de fonction pour chaque fonction de l'architecture logique. Elle est présentée au listing 3.11.

Listing 3.11 – Partie logique de la configuration Eon1

```
logical configuration {
  instance I_transfert of transfert;
  instance I_acquisition of acquisition;
  instance I_traitement_grossier of traitement_grossier;
  instance I_traitement_fin of traitement_fin;
}
```

La carte 1 est utilisée pour acquérir les images à l'aide des caméras. La carte 2 effectue des traitements grossiers afin de choisir des images qui présentent un intérêt. Finalement, la carte 4 effectue un traitement plus fin de l'image afin d'identifier des points d'intérêt et de l'annoter. Les opérations potentielles mises en œuvre seront donc `C1Acq`, `C2trf` et `C4trq`.

Il faut donc mettre en place des connecteurs permettant de réaliser les transferts de données entre les différentes cartes. Un premier transfert est nécessaire entre la carte 1 et la carte 2, on utilisera pour cela la liaison dédiée. Le second transfert entre la carte 2 et la carte 4 se fera en utilisant le bus. Il faut aussi définir une séquence de transfert entre la carte 4 et la carte 3 pour stocker les résultats en mémoire. La définition des différents transferts est réalisée dans la configuration physique représentée au listing 3.12.

Listing 3.12 – Partie physique de la configuration Eon1

```
physical configuration {
  transfer sequence TS_C1C2 {
    transfer C1C2 using DLink;
  }
  transfer sequence TS_C2C4 {
    transfer C2B2 using Bus;
    control from B2.toC2 to B2.B2_out;
    transfer B2B3 using Bus;
    control from B3.B3_in to B3.B3_out;
    transfer B3B4 using Bus;
    control from B4.B4_in to B4.toC4;
    transfer B4C4 using Bus;
  }
  transfer sequence TS_C4C3 {
    transfer C4B4 using Bus;
    control from B4.toC4 to B4.B4_in;
    transfer B4B3 using Bus;
    control from B3.B3_out to B3.toC3;
    transfer B3C3 using Bus;
  }
}
```

Les opérations sont ensuite être définies ainsi que les deux séquences d'opérations. Ces dernières respectent les séquences de fonctions définies dans l'architecture. La définition textuelle de ces éléments est proposée au listing 3.13.

Listing 3.13 – Opérations définies dans la configuration Eon1

```
stationary operation S_acquisition : I_acquisition using C1Acq;
stationary operation S_traitement_grossier : I_traitement_grossier
   using C2trg;
stationary operation S_traitement_fin : I_traitement_fin using C4trf;

transfer operation T_C1C2 : I_transfert using TS_C1C2;
transfer operation T_C2C4 : I_transfert using TS_C2C4;
transfer operation T_C4C3 : I_transfert using TS_C4C3;

operation sequence OS_choix_image realize choix_image : S_acquisition,
    T_C1C2, S_traitement_grossier;
```

```
10 operation sequence OS_annotation realize annotation : T_C2C4,
       S_traitement_fin, T_C4C3;
```

Une deuxième configuration pour le système électronique

La configuration “Eon2” est une configuration minimaliste du système électronique dans laquelle seules les deux premières cartes sont utilisées. Dans cette configuration, la phase de post-traitement n’est plus réalisée, diminuant la QdS du système mais simplifiant sont fonctionnement.

La configuration logique ne définit alors des instances que pour les fonctions `acquisition` et `traitement_grossier`. Ces définitions sont donnée au listing 3.14.

Listing 3.14 – Partie logique de la configuration Eon2

```
1 logical configuration {
2   instance I_transfert of transfert;
3   instance I_acquisition of acquisition;
4   instance I_traitement_grossier of traitement_grossier;
5 }
```

La configuration physique ne définit alors que de la séquence de transfert `TS_C1C2`, comme le montre le listing 3.15

Listing 3.15 – Partie physique de la configuration Eon2

```
1 physical configuration {
2   transfer sequence TS_C1C2 {
3     transfer C1C2 using DLink;
4   }
5 }
```

La définition des opérations proposée au listing 3.16 reflète les modifications apportées par rapport à la configuration Eon1.

Listing 3.16 – Opérations définies dans la configuration Eon2

```
1 stationary operation S_acquisition : I_acquisition using C1Acq;
2 stationary operation S_traitement_grossier : I_traitement_grossier
     using C2trg;
3
4 transfer operation T_C1C2 : I_transfert using TS_C1C2;
5
6 operation sequence OS_choix_image realize choix_image :
7         S_acquisition, T_C1C2, S_traitement_grossier;
```

3.5.3 Bilan sur les projections

Le langage DeSyRe permet la modélisation de SAP ainsi que de systèmes électroniques. Ce langage utilise volontairement un vocabulaire le plus neutre possible. Le tableau 3.1 établit une analogie entre les termes employés dans le modèle, et les termes correspondant dans les deux exemples. Cette analogie est principalement valable au niveau de granularité employé dans les exemples qui correspond à un niveau “système”.

Certaines notions ne sont pas définies dans le modèle. Il est ainsi impossible de décrire un algorithme puisque aucune structure de contrôle n’est disponible. La description se fait donc nécessairement à un niveau supérieur, dans lequel les contrôles font partie des fonctions ou délégués à une entité réalisant l’enchaînement des différentes séquences d’opérations définies dans l’architecture en choisissant les produits à traiter ainsi que les ressources chargées de ces traitements.

Element du modèle	Traduction SAP	Tradution électronique
Système logique	Définition des produits	Application
Fonction	Fonction	Traitement élémentaire
Produit	Pièce/Colis	Donnée
Séquence de fonctions	Gamme Logique	Processus
Système physique	Cellule	Architecture Matérielle
Ressource stationnaire	Machine	Opérateur/processeur
Ressource stationnaire	Rayonnage	Mémoire
Ressource de transport	Convoyeur/Robot/Véhicule Autoguidé	Réseau/Bus
Connexion	Transfert potentiel	Liaison
Opération	Opération	Tâche
Séquence d'opérations	Gamme Opératoire	Graphe de tâche

TAB. 3.1 – Correspondances sémantiques entre le modèle, les SAP et les systèmes électroniques

3.6 Conclusion sur le langage de description

Ce chapitre a présenté un langage de description pour les systèmes reconfigurables. Ce langage a été défini de manière informelle, dans un premier temps pour introduire les concepts manipulés. Une définition formelle a ensuite permis d'expliciter les relations entre les différents éléments de modélisation. Pour finir deux DSL ont été définis pour utiliser la description dans le cadre de l'ingénierie des modèles, une représentation visuelle a alors été proposée.

Le langage proposé pour la description des systèmes reconfigurables a été défini en séparant clairement l'architecture de la configuration. Cette séparation permet de définir plusieurs configurations pour une même architecture.

Le langage a ensuite été utilisé pour modéliser à la fois des systèmes électroniques et des systèmes de production. Ainsi, un rapprochement a été réalisé entre les deux domaines, confirmant l'intuition formulée au chapitre 1, des correspondances sémantiques ont pu être réalisées. Cette versatilité de la description laisse envisager l'utilisation du langage DeSyRe pour la description du procédé et de l'électronique de la partie commande d'un système de production.

Le chapitre suivant présente l'utilisation de ce langage pour l'analyse du système reconfigurable afin d'aider à la conception d'architecture tolérante et de diriger la reconfiguration.

Chapitre 4

Analyse d'un système reconfigurable à partir de sa représentation de haut niveau

Un langage a été défini pour décrire les systèmes reconfigurables. Ce langage a été utilisé pour modéliser des systèmes reconfigurables aussi bien dans les domaines de l'électronique que des SAP.

La complexité inhérente à ces systèmes, est décuplée par leur caractère reconfigurable qui empêche d'appréhender toutes les facettes du système sans l'aide d'outils de décision. Ces outils peuvent aussi bien permettre de qualifier une architecture en terme de reconfigurabilité que de choisir une configuration.

Le langage DeSyRe est utilisé comme point d'entrée pour l'analyse des systèmes reconfigurables. Pour cela, un cadre d'analyse utilisant l'ingénierie des modèles est défini. Ce cadre est ensuite utilisé pour analyser les propriétés du modèle de l'architecture et de la configuration.

La première section présente les besoins en terme d'analyses réalisées sur les modèles. Dans une deuxième section, un cadre d'analyse générique utilisant l'ingénierie des modèles est présenté. Les analyses mises en œuvre sont ensuite présentées dans la troisième section. Finalement, des résultats d'analyse sont donnés sur les exemples présentés au chapitre précédent.

4.1 Motivation des analyses mises en œuvre

Cette section introduit les différentes analyses qui seront ensuite pratiquées sur la description du système reconfigurable. Ces différentes analyses sont déclinées sur l'architecture et sur la configuration.

Les besoins qui ont été retenus dans le cadre de l'évaluation des systèmes reconfigurables sont l'évaluation du coût du système, de ses performances, de sa flexibilité, de sa robustesse ainsi que des besoins de cohérence. Chaque besoin est évalué à la fois sur l'architecture et sur la configuration.

4.1.1 Analyses de coût

Le coût d'une solution correspond à un besoin fort de la part de l'utilisateur, et se rapporte généralement à un coût financier. Associé à d'autres évaluations, il permet de privilégier une solution par rapport à une autre.

Besoins

Différents coûts peuvent être envisagés. En électronique, on considère principalement un coût en surface de silicium et un coût énergétique. Dans le cas des systèmes de production, les mêmes coûts peuvent être envisagés, un coût technologique correspondant à la complexité des ressources utilisées peut aussi être considéré.

Dans le cadre de l'architecture, c'est principalement le critère de surface qui peut être évalué. La consommation statique en énergie qui peut aussi être un critère important est principalement liée au nombre de ressources utilisées.

Pour les configurations, la surface occupée par celle-ci peut être évaluée de même que la consommation énergétique statique, la détermination de la consommation dynamique nécessitant de savoir comment est utilisée la configuration. Il faut aussi prendre en considération le coût de la mise en place de la configuration considérée.

Moyens

Afin d'évaluer la surface prise par l'architecture, la métrique mise en œuvre est une métrique structurelle consistant à compter un *nombre d'éléments*. Il s'agit, pour chaque type d'élément (ressource, connexion) de compter combien d'éléments sont utilisés dans l'architecture. Cette métrique est aussi utilisée pour la configuration.

Le coût de la reconfiguration, quant à lui, peut être décomposé en deux parties. Il faut tout d'abord évaluer le changement du mode des ressources pour qu'il corresponde à celui de la configuration. Ce coût peut prendre en considération la mise sous tension d'une ressource suivie d'une phase de préparation et également la remise en marche d'une ressource tombée en panne. La deuxième partie du coût de reconfiguration est constituée par la modification de l'état des produits dans le système pour qu'il corresponde à un état accepté par la configuration. On parle alors de la mise en cohérence de l'état des produits et de la configuration.

4.1.2 Analyses de performance

Les critères de performance considérés concernent la réalisation des différents produits. En effet, les systèmes considérés sont conçus pour fournir des services sous un ensemble de contraintes parmi lesquelles apparaissent la quantité de produits réalisés ainsi que la qualité de leur réalisation.

Besoins

Pour les systèmes électroniques, la performance souhaitée peut s'exprimer en terme de qualité de service, elle est alors mesurée au niveau de l'application et correspond à la satisfaction de l'utilisateur. Pour un traitement vidéo, les paramètres de QdS sont la résolution de l'image, sa netteté et la fluidité de la vidéo. Au niveau du système, on ne peut qu'évaluer ces critères par des mesures réalisées pendant le fonctionnement du système sur les résultats intermédiaires des différents traitements. Pendant la conception, il faut estimer la quantité d'opérations nécessaires et calculer la durée de ces calculs. Au niveau des SAP, les mêmes préoccupations apparaissent, le critère utilisé pour mesurer la performance est le temps de cycle. L'objectif est de déterminer à quelle cadence sortent les produits en mettant en œuvre les traitements nécessaires à l'obtention de la QdS souhaitée.

Moyens

La mesure d'un temps de cycle, dans le cas général, nécessite la réalisation d'un ordonnancement, permettant de gérer, entre autre, les différents partages de ressources. Un moyen générique pour mesurer le temps de cycle du système dans une configuration donnée consiste donc à réaliser l'ordonnancement des opérations et ainsi mesurer le temps de cycle par simulation.

Lors de la conception de l'architecture, l'utilisation des différentes ressources n'est pas nécessairement connue. Cette information est déterminée lors de la création d'une configuration pour l'architecture. Il faut alors évaluer les capacités de cette architecture. L'architecture physique peut fournir une information quant aux capacités du système pour réaliser des traitements. Un critère pertinent est alors le degré de parallélisation des traitements. Cette *parallélisation potentielle* peut être décomposée en une parallélisation pour les traitements (fonction, séquence de fonction, produit) ou pour les transferts. L'architecture logique permet de quantifier la quantité de traitements nécessaires pour réaliser une séquence de fonctions ou un produit.

La configuration détermine plus précisément comment peut être utilisée l'architecture (mais ne contient pas l'ordonnancement des opérations). Ainsi, les capacités réellement mises en œuvre par le système sont connues et peuvent être évaluées. La détermination de la durée d'une séquence d'opération est alors possible. En réalisant l'ordonnancement des opérations, la détermination du temps de cycle est possible.

4.1.3 Analyse de la flexibilité du système

La flexibilité caractérise la capacité du système à évoluer pour faire face à un aléa.

Besoins

Plusieurs types d'évolutions sont possibles. On peut faire évoluer les objectifs de production (pour un système électronique il peut s'agir des fonctionnalités proposées à l'utilisateur), on parle alors d'une flexibilité de produits. On peut aussi faire évoluer les moyens, ce qui correspond à une flexibilité de gamme. L'utilisation des moyens de transport pour réaliser une même séquence peut également évoluer. On a alors à faire à une flexibilité des moyens de transport.

Moyens

La flexibilité de produits s'évalue par rapport au nombre de types de produits qui peuvent être réalisés sur le système. Plus ce nombre est élevé, plus le choix du type de production est grand.

La flexibilité des gammes correspond aux alternatives de séquences de fonctions proposées pour réaliser un produit donné. Une métrique permettant d'évaluer cette flexibilité pour une architecture consiste à faire le rapport du nombre de séquences de fonctions sur le nombre de produits.

La flexibilité de transport caractérise les possibilités de transferts pour réaliser une gamme donnée. Les différents transferts possibles peuvent alors être énumérés pour une séquence de fonctions donnée. Nous avons choisi d'évaluer les capacités du système de transport en établissant un ratio entre le nombre de connexions et le nombre de ressources stationnaires dans le système.

4.1.4 Analyse de la robustesse du système

La robustesse du système caractérise sa capacité à continuer un traitement suite à des perturbations. Cette caractéristique découle en partie de la flexibilité.

Besoins

Au niveau de l'architecture, il s'agit de déterminer tout d'abord, les caractéristiques des éléments du système en terme de sûreté de fonctionnement (MTBF[1], MTTF[2]). On peut aussi vouloir évaluer, indépendamment de la sûreté de fonctionnement du système, la possibilité d'avoir un blocage dû à une mauvaise gestion des ressources. L'impact de la perte d'un élément sur le fonctionnement est ensuite envisagé.

L'analyse de l'impact de la perte d'un élément est un critère prépondérant par rapport à la problématique de la reconfiguration. Il s'agit, en phase de conception, de déterminer les capacités de reconfiguration de l'architecture. En phase d'exploitation, cette analyse permet de déterminer si les objectifs de productions peuvent être poursuivis ou s'il faut les changer.

La notion de niveau de reconfiguration peut intervenir dans le cadre de la robustesse d'une configuration. Il s'agit alors de savoir si en n'autorisant qu'un niveau de reconfiguration 1.1.5, le système peut ou non continuer à fonctionner.

Moyens

La mise en œuvre des analyses de sûreté de fonctionnement peut passer par l'obtention de modèles de SdF à partir du modèle de l'architecture ou de la configuration. Des outils analytiques ou procédant par simulation permettent ensuite de calculer automatiquement les différents paramètres.

En ce qui concerne l'identification de blocages potentiels, il s'agit de vérifier que les évolutions possibles du système ne conduisent pas à un interblocage. Des configurations susceptibles de se bloquer peuvent exister. Des mécanismes d'évitement ou de résolution de ces blocages doivent alors être mis en place.

Les travaux sur la criticité [Berruet, 1998] constituent un point de départ pour les analyses réalisées vis à vis de la détermination de l'impact de la défaillance d'un élément sur le système dans le cas d'un SAP. La criticité a alors été introduite par la définition 1.

Définition 1 *Un élément (opération / ressource) critique est un élément dont la panne rends inutilisable(s) une ou plusieurs fonction(s) de transformation du SFPM [Berruet et al., 1997]*

La disponibilité d'une opération est donnée par la définition 2

Définition 2 *Une opération est disponible si elle est accessible depuis l'entrée du système et si la sortie de celui-ci peut être atteinte depuis celle-ci.*

L'intérêt de ces travaux est qu'ils considèrent non-seulement les éléments directement indisponibles, mais aussi les éléments indirectement rendus indisponibles car non accessibles. Par contre, ces travaux considèrent que le mode de défaillance des ressources est de type "fail-silent", c'est-à-dire que celles-ci n'influent pas sur le fonctionnement des autres ressources lorsqu'elles tombent en panne.

[1]Mean Time Between Failures : Durée moyenne entre les fautes

[2]Mean Time To Failure : Durée moyenne avant la première faute

4.1.5 Analyses de cohérence

La cohérence de deux éléments caractérise l'accord entre deux modèles d'un système ou entre un modèle et le système qu'il représente.

Besoin

Pendant la conception de l'architecture, il peut être nécessaire de vérifier que l'architecture physique et les opérations potentielles définies permettent de réaliser les séquences de fonctions définies dans l'architecture logique. De même, la configuration doit utiliser les éléments de l'architecture dans le cadre qui leur a été défini.

Pendant le fonctionnement du système, celui-ci évolue, et les différents modèles qui le représentent que sont son architecture, sa configuration ainsi que les modèles opérationnels, utilisés au cours du fonctionnement du système et qui traduisent l'état des ressources ou l'état des produits dans le système peuvent perdre leur cohérence.

Moyen

Plusieurs évaluations de cohérence sont mises en œuvre à partir du langage DeSyRe. Une analyse de l'adéquation entre la partie logique et la partie physique de l'architecture peut tout d'abord être réalisée pour vérifier la faisabilité des séquences de fonctions sur le système. La vérification de la cohérence des configurations avec l'architecture peut ensuite être réalisée. Il s'agit de vérifier que les éléments utilisés dans une configuration (ressource, connexion, fonction, séquence de fonctions) sont bien définis dans l'architecture. Il faut aussi vérifier que ces éléments sont utilisés correctement. Dans le cas d'une séquence d'opérations définie dans la configuration, ces opérations doivent respecter l'ordre des fonctions défini dans la séquence de fonctions réalisée par cette séquence d'opérations.

Pendant le fonctionnement du système, la cohérence entre le modèle de configuration et les modèles opérationnels peut aussi être vérifiée. La configuration est cohérente par rapport au mode des ressources si les ressources utilisées dans la configuration peuvent l'être (elles doivent être disponibles et en état de marche). Par rapport à l'état des produits, celui-ci doit rester admissible dans la configuration, c'est à dire que la configuration doit être en mesure de faire évoluer correctement l'état des produits.

4.1.6 Bilan sur les analyses

Différentes analyses ont été introduites pour répondre aux attentes des utilisateurs des systèmes que nous visons.

Les analyses définies dans cette section sont appliquées à des phases différentes de la vie du système. Les analyses sur l'architecture sont généralement appliquées lors de la conception du système, pour vérifier que l'on pourra profiter de ses capacités en terme de reconfiguration. Il s'agit donc principalement d'une recherche de l'adéquation entre la partie physique et la partie logique de l'architecture. Les analyses appliquées sur la configuration permettent principalement de choisir ou de construire une configuration. La mise en place et l'utilisation de ces dernières dépendent fortement de l'application.

Les différentes analyses présentées à cette section sont mises en œuvre dans les section 4.3 et 4.4 au sein du cadre défini à la section suivante.

4.2 Cadre d'analyse générique

Un langage pour la description des systèmes reconfigurables a été défini et différentes analyses ont été identifiées pour cette description.

La mise en œuvre de l'analyse de l'architecture et de la configuration nécessite souvent le recours à des outils extérieurs comme par exemple les réseaux de Petri [Murata, 1989] pour des analyses d'interblocage. Les résultats sont alors fournis sous la forme d'un rapport qu'il faut analyser et intégrer dans un rapport plus général sur le système.

Le *cadre d'analyse* [Lamotte et al., 2005c] qui suit répond à cet objectif. Celui-ci met en œuvre les différentes étapes identifiées et associe un modèle à chacune de ces étapes. Celui-ci est présenté à la figure 4.1. Il englobe l'activité de saisie du modèle, son analyse proprement dite et l'intégration de ces résultats au sein de la description du système.

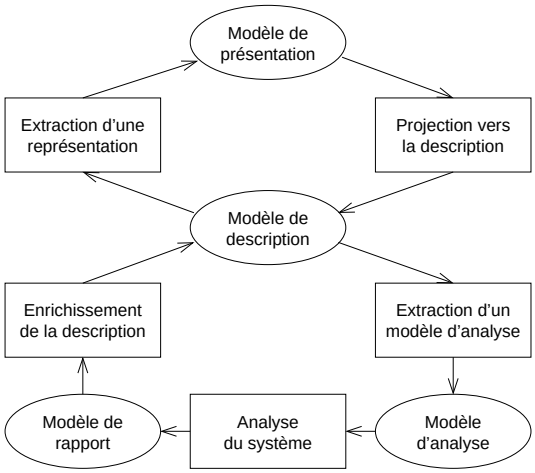


FIG. 4.1 – Cadre pour l'analyse des systèmes reconfigurables

Quatre types de modèles, représentés par des boites ellipsoïdales composent ce cadre. Le modèle de description présenté à la section 4.2.1 est le pivot de l'analyse, il regroupe les éléments caractéristiques du système. Un ou plusieurs modèles de présentation décrit à la section 4.2.2 lui sont associés afin de permettre sa saisie, sa visualisation ou sa modification. Les modèles d'analyse présentés à la section 4.2.3 ont été introduits comme interface entre la description du système et les outils d'analyse internes ou externes. Ces modèle doivent être conforme à ce qu'attend l'outil d'analyse sélectionné. Finalement, le modèle de rapport 4.2.4 permet de récupérer les résultats d'analyse afin d'enrichir la description du système.

Les boites rectangulaires représentent les étapes qui permettent le passage d'un modèle d'un type à un autre.

Le cadre d'analyse ayant été introduit, les différents modèles constituant ce cadre sont introduits dans les sections suivantes, de même que les relations entre ces modèles.

4.2.1 Les modèles de description

Un modèle de description permet de définir les éléments du système dans une sémantique proche de celui-ci. Dans notre étude, il s'agit de la description d'un système reconfigurable définie à la section 3.3.

La description du système peut être réalisée en utilisant plusieurs modèles. Ce sera le cas pour nous lorsque nous analyserons les configurations qui sont intrinsèquement liées à l'architecture du système.

4.2.2 Les modèles de présentation

Les modèles de présentation servent d'interface entre l'utilisateur et le modèle de description. Ils sont associés à des outils spécifiques permettant de manipuler le modèle, il peut s'agir d'un outil de dessin auquel cas le modèle de présentation est défini en termes de figures. Un éditeur de texte peut aussi être utilisé pour saisir un modèle de présentation, le modèle associé est alors un arbre syntaxique obtenu à partir du document.

Certains modèles de présentation ont pour objectif unique de présenter une vue du système, l'exemple le plus parlant d'un tel modèle serait une courbe obtenue à partir des valeurs inscrites dans les éléments du modèle.

Dans le contexte de la modélisation des systèmes reconfigurables deux modèles de présentation ont déjà été présentés. Le premier est une représentation textuelle de l'architecture et de la configuration. Le second est une représentation visuelle utilisant un outil UML, présenté à la section 3.4.

4.2.3 Les modèles d'analyse

Les modèles d'analyse font l'interface entre les outils d'analyse et le modèle de description. En effet, il peut exister un fossé sémantique entre le langage utilisé pour décrire le système et celui qu'utilise l'outil d'analyse. Dans le cas de l'architecture par exemple, si on veut réaliser une analyse de vivacité à l'aide d'un réseau de Petri, il faut faire une traduction de l'architecture en places et en transitions.

Un modèle d'analyse est donc obtenu automatiquement à partir d'un modèle de description par transformation de modèles. Il est ensuite mis au format demandé par l'outil afin d'être traité par celui-ci.

4.2.4 Le rapport d'analyse

Le rapport d'analyse est construit à partir des résultats donnés par le logiciel utilisé pour réaliser l'analyse. Ce rapport est réalisé en associant les résultats au modèle de description. Un méta-modèle générique de rapport a été défini et est présenté à la figure 4.2, il permet d'associer un ou plusieurs résultats à chaque élément du modèle de description.

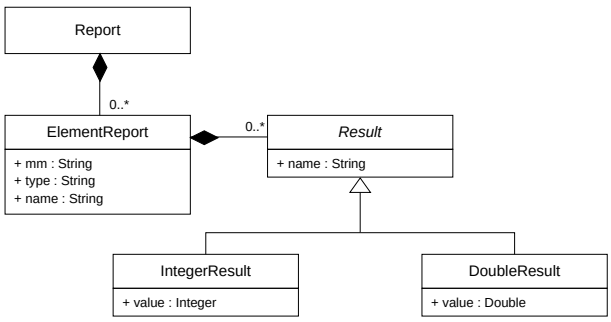


FIG. 4.2 – Méta-modèle du rapport d'analyse

L'obtention d'un modèle de rapport d'analyse à partir des résultats nécessite une transformation qui interprète les résultats fournis par l'outil d'analyse et fait l'association entre ceux-ci et les éléments du modèle de description du système.

L'utilisation du méta-modèle de rapport présenté à la figure 4.2 permet d'ajouter incrémentalement les résultats d'analyse en juxtaposant les rapports. L'utilisation d'un modèle annexe pour

stocker les résultats permet d'éviter de surcharger le modèle de description du système tout en adjoignant les résultats d'analyse au niveau de ce même modèle sous la forme d'un modèle annexe relié au modèle de description du système par l'intermédiaire du nom des éléments.

Un modèle de présentation particulier, sous la forme d'un document imprimable peut ensuite être obtenu à partir du modèle de description associé aux différents rapports d'analyse.

4.2.5 Bilan sur le cadre d'analyse

La définition d'un cadre générique pour l'analyse des systèmes représentés par un ou plusieurs modèles permet de clarifier le processus de conception itératif, guidé par l'évaluation d'un ensemble de métriques obtenues sur le modèle. Différentes analyses peuvent alors être appliquées sur le même modèle de description et compléter celui-ci.

La visualisation du modèle de description auquel ont été associés les résultats d'analyse par l'intermédiaire d'un modèle de présentation adapté permet l'amélioration du modèle du système qui pourra alors être soumis à de nouvelles analyses.

Le cadre présenté dans cette section est utilisé dans les deux sections suivantes pour mettre en œuvre les analyses définies pour les systèmes reconfigurables.

4.3 Analyses mises en œuvre sur l'architecture

Le cadre d'analyse développé dans la section 4.2 est utilisé dans cette section ainsi que dans la suivante, pour mettre en œuvre les différentes métriques identifiées à la section 4.1.

Tout d'abord, la notion de contexte d'analyse est définie. Cette notion permet de travailler sur des sous-ensembles de l'architecture afin d'appliquer, sur la configuration, les métriques définies ici dans le cadre de l'architecture. Ensuite, les différentes analyses réalisées sur l'architecture sont présentées.

4.3.1 Notion de contexte

La définition des différentes analyses nécessite l'introduction de la notion de contexte pour le calcul à la fois sur l'architecture et sur la configuration des différentes métriques.

Le contexte, noté C, correspond à une sous partie de l'architecture sur laquelle les mesures sont relevées. Par nature, le contexte est homogène à une architecture. Dans le cas d'une analyse de l'architecture on a $C = \mathcal{A}$.

4.3.2 Obtention du nombre d'éléments d'un type donné

Cette première métrique a pour objectif d'évaluer la taille de l'architecture. On peut ainsi connaître le nombre de ressources mises en œuvre, le nombre de fonctions ou de produits distincts qui peuvent être réalisés.

Bien sûr, les résultats de ces métriques dépendent de la granularité à laquelle le concepteur s'est placé pour cette modélisation. Une modélisation dans laquelle le fonctionnement des ressources est explicitement représenté contient naturellement plus d'éléments qu'une modélisation du système ne décrivant que les relations entre ces ressources.

Définition

L'utilisation d'analyses structurelles sur l'architecture a été motivée à la section 4.1.1 pour l'évaluation du coût de l'architecture. Ces analyses sont réalisées en deux temps, dans un premier temps, il s'agit d'énumérer les ressources stationnaires, les ressources de transport et les connexions de l'architecture physique. Dans l'architecture logique, on dénombre les fonctions, les produits et les séquences. Dans un second temps les résultats du dénombrement sont utilisés pour obtenir d'autres indicateurs introduits aux section 4.3.3 et 4.1.3.

La métrique *nombre d'éléments*, basée sur la définition formelle de l'architecture présentée en 3.2.1, est représentée par la fonction $N_{x,C}$ où x correspond au type d'élément et C est le contexte considéré. $N_{x,C}$ est défini par l'équation 4.1[3] où x peut prendre les valeurs données dans le tableau 4.1 suivant le type d'élément considéré.

$$N_{x,C} = |x| \tag{4.1}$$

Nom	Ensemble	définition
Ressources	R	R
Ressources de transport	Tr	$\{tr \in R \mid \exists src, dst \in R \times R, con(src, dst, tr) = 1\}$
Ressources stationnaires	St	$\{r \in R \mid r \notin Tr\}$
Connexions	Cx	$\{(src, dst) \in R \times R \mid \exists tr, con(src, dst, tr) = 1\}$
Fonctions	F	F
Séquences de Fonctions	FS	G
Produit	P	P
Produit consommé	Pc	$\{p \in P \mid \exists g \in G, cons(g, p) = 1\}$
Produit réalisé	Pr	$\{p \in P \mid \exists g \in G, real(g, p) = 1\}$
Opérations potentielles	O	O

TAB. 4.1 – Liste des ensembles que peut désigner x dans la définition de N

Mise en œuvre

L'application du calcul du nombre d'éléments se fait dans le cadre d'analyse proposé à la section 4.2, en utilisant uniquement la transformation de modèles qui dispose, grâce au langage OCL de constructions permettant facilement d'obtenir les informations souhaitées.

Le modèle d'analyse utilisé pour le calcul est le modèle de description. L'obtention du modèle d'analyse se fait donc directement.

Le rapport est obtenu grâce à une règle prenant en entrée l'architecture et produisant un résultat pour chacun des types d'éléments considéré. Une partie de cette règle est présentée au listing 4.1.

Listing 4.1 – Extrait de la règle utilisée pour l'analyse structurelle de l'architecture

```
1 rule Architecture2Report {
2   from
3     a : Architecture!Architecture
4   to
5     r : Report!Report (
6       name <- a.name + '_' + 'report',
7       elements <- Sequence{ar}
8     ),
```

[3] $|x|$ désigne le cardinal de x et non sa valeur absolue

```
9       ar : Report!ElementReport (
10        mm <- 'Architecture',
11        type <- 'Architecture',
12        name <- a.name,
13        results <- Sequence {sn}
14      ),
15      sn : Report!IntegerResult (
16        name <- 'stationaryResourceNumber',
17        value <- Architecture!StationaryResource.allInstances()->size()
18      )
19  }
```

La règle présentée créé trois éléments à partir de l'architecture : un rapport constituant la racine du modèle (lignes 4 à 8) ainsi qu'un rapport rattaché à l'architecture (lignes 9 à 14) contenant un résultat correspondant au nombre de ressources stationnaires de cette architecture (lignes 15 à 18). Ce résultat est obtenu en calculant la taille de la séquence composée de toutes les ressources stationnaires de l'architecture.

Exemple

Sur l'exemple des deux machines, introduit au chapitre 3 : $N_{R,\mathcal{A}} = 6$, $N_{Cx,\mathcal{A}} = 6$, $N_{F,\mathcal{A}} = 4$, $N_{FS,\mathcal{A}} = 2$, $N_{P,\mathcal{A}} = 3$, $N_{O,\mathcal{A}} = 4$

4.3.3 Parallélisme potentiel des transferts

Le *parallélisme potentiel des transferts* est une deuxième analyse structurelle effectuée sur l'architecture. Il s'agit aussi d'une analyse de performance puisque en augmentant le parallélisme des transferts on peut limiter l'attente avant d'effectuer un transfert.

Définition

Deux métriques complémentaires sont utilisées. La première est décrite par l'équation 4.2 et correspond au ratio du nombre de ressources de transport sur le nombre de ressources stationnaires. Le sens de cette métrique est limité par la modélisation du système retenue. En effet aucune capacité n'est affectée aux ressources. Une ressource de transport ne réalise donc qu'un transfert à la fois, ce qui n'est pas le cas dans le cadre de la modélisation d'un convoyeur telle qu'elle est réalisée sur l'exemple 3.5.1. On peut en effet représenter le convoyeur comme une ressource unique ou comme un ensemble de ressources indépendantes.

$$TP_C = \frac{N_{Tr,C}}{N_{St,C}} \tag{4.2}$$

Une deuxième métrique permet d'évaluer le parallélisme potentiel des transferts et complète ainsi la première métrique. Cette métrique est définie par l'équation 4.3, où l'on calcule le rapport du nombre de connexions sur le nombre de ressources stationnaires.

$$CS_C = \frac{N_{Cx,C}}{N_{St,C}} \tag{4.3}$$

Mise en œuvre

Ces métriques peuvent être calculées de deux manières différentes. Se basant sur différents $N_{x,C}$, il est possible de les appliquer en utilisant comme modèle d'analyse, la description de l'architecture associée au rapport obtenu lors du calcul des nombres d'éléments. En se basant

sur les valeurs calculées précédemment, on peut alors raffiner le rapport et y ajouter les parallélismes potentiels des transferts.

Une deuxième méthode consiste à réaliser toutes les analyses structurelles lors de la même transformation. Les règles calculant les deux métriques définies dans cette section sont alors ajoutées à celles permettant d'obtenir les nombres d'éléments. Cette façon de faire est moins modulaire, mais plus simple à mettre en œuvre.

Exemples

Sur l'exemple des deux machines, dans lequel on trouve quatre ressources stationnaires et deux ressources de transport, le calcul du parallélisme potentiel donne $TP_{\mathcal{A}} = 0.5$. L'architecture physique est aussi composée de six connexions donc $CS_{\mathcal{A}} = 1.5$.

4.3.4 Parallélisme potentiel de traitement

Au niveau de l'architecture logique, d'autres analyses structurelles permettent de mettre en évidence les parallélismes potentiels. Il s'agit du *parallélisme potentiel des traitements*, indiquant a quel degré peuvent être parallélisée une fonction, une séquence de fonctions ou encore la réalisation d'un produit.

Définition

Le parallélisme potentiel de traitement peut être défini pour chacune des différentes entités qui composent l'architecture logique. L'équation 4.4 définit ce parallélisme dans le cas des fonctions. $PP_{f,C}$ désigne alors le nombre de ressources pouvant réaliser simultanément une opération implémentant la fonction f dans le contexte C.

$$PP_{f,C} = |\{r \in R | \exists o \in \mathcal{O}, loc(o, r, f)\}| \tag{4.4}$$

Dans le cas des séquences de fonction, la définition est donnée par l'équation 4.5. On recherche le nombre de ressources distinctes pouvant réaliser simultanément des opérations implémentant une fonction de la même séquence de fonctions g.

$$\begin{aligned} PP_{g,C} &= |\{r \in R \mid \exists f1, f2, o \in F \times F \times \mathcal{O}, \\ &\quad (pre(g, f1, f2)) \wedge (loc(o, r, f1) \vee loc(o, r, f2))\}| \end{aligned} \tag{4.5}$$

En ce qui concerne les produits, on retrouve une définition proche de celle des séquences de fonctions. Cette fois-ci on considère toutes le ressources pouvant réaliser des opérations implémentant une fonction appartenant à une séquence pouvant réaliser le produit.

$$\begin{aligned} PP_{p,C} &= |\{r \in R \mid \exists g, f1, f2, o \in G \times F \times F \times \mathcal{O}, \\ &\quad (real(g, p) \wedge pre(g, f1, f2)) \wedge (loc(o, r, f1) \vee loc(o, r, f2))\}| \end{aligned} \tag{4.6}$$

Mise en œuvre

Tout comme pour les autres analyses structurelles, la mise en œuvre de cette métrique se fait par transformation de modèle à partir du modèle de description.

Exemples

La fonction $F1$ est implémentée par des opérations réalisées sur $M1$ et $M2$ donc $PP_{F1,\mathcal{A}} = 2$. Les deux autres fonctions ont un parallélisme potentiel de traitement de 1 puisqu'elles ne sont implémentées que par une opération respectivement réalisées sur $M1$ pour $F2$ et $M2$ pour $F3$.

En ce qui concerne les séquences de fonction et les produits, la séquence $P1\_S1$ utilisant toutes les fonctions, son parallélisme potentiel de traitement est de 2. Le parallélisme potentiel du produit $P1$ est aussi égal à 2.

4.3.5 Flexibilité des produits

L'objectif de cette métrique est de quantifier le nombre de séquences de fonctions alternatives qui peuvent être mises en œuvre pour réaliser un produit donné.

Définition

La *flexibilité des produits* dans un contexte C : FP_C, est définie par l'équation 4.7. Il s'agit du ratio entre le nombre de séquences et le nombre de produits réalisés.

$$FP_C = \frac{N_{FS,C}}{N_{Pr,C}} \tag{4.7}$$

Mise en œuvre

La mesure de la flexibilité des produits s'effectue en procédant de la même manière qu'avec les potentialités des transferts.

Exemples

L'exemple des deux machines décrit une seule séquence de fonctions permettant de réaliser un produit, $FP_{\mathcal{A}} = 1$.

4.3.6 Détermination du degré de criticité des éléments de l'architecture

L'analyse de la tolérance à été introduite à la section 4.1.4 comme un moyen de quantifier un aspect relevant de la sûreté de fonctionnement d'un système.

L'objectif est de mesurer la flexibilité d'un système face à l'indisponibilité d'un sous-ensemble de ses ressources de traitement ou de transport.

La définition et la mise en œuvre de la criticité sont abordées dans les sections suivantes. Les définitions sont proposées dans un premier temps. La deuxième section introduit la mise en œuvre des calculs associés. Le modèle d'analyse permettant d'obtenir les résultats est ensuite décrit, suivi, dans une autre section, par la description de la transformation permettant d'obtenir le modèle d'analyse à partir du modèle de description de l'architecture. Finalement, l'interprétation des résultats d'accessibilité est présentée.

Définitions

La criticité a été introduite dans [Berruet et al., 1997] par la définition 1. Cette définition introduit les concepts d'opérations et de ressources critiques. Elle a ensuite été étendue aux

ensembles d'opérations [Lamotte et al., 2005b]. Une ressource ou une opération peuvent en effet être considérées comme un ensemble d'opérations, il en est de même pour les fonctions.

En notant $\mathcal{O}$ l'ensemble des opérations de l'architecture (transformation, traitement ou stockage) et R l'ensemble des ressources, une opération étant réalisée sur une seule ressource, une ressource $r \in R$ définit une partition de $\mathcal{O}$.

O un sous-ensemble de $\mathcal{O}$, l'application $Acc : O \times O \rightarrow \{0, 1\}$ est appelée relation d'accessibilité et indique si deux opérations sont directement accessible c'est à dire que l'on peut réaliser la seconde après réalisation de la première.

$C : \{O, Acc\}$ est un couple représentant le contexte considéré.

T est l'ensemble des éléments cibles, par rapport auxquels la criticité des éléments est obtenus. Cet ensemble est constitué de sous-ensembles de O appellés *t-éléments*.

La définition de la criticité est alors donnée à la définition 3.

Définition 3 *$e \subset O$ est critique par rapport à T dans le contexte C si la perte de toutes les opérations de e conduit à la disparition d'un ou plusieurs t-élément de T.*

La fonction *impact* : $Imp_{C,T}(e) : O \rightarrow T$ renvoie les t-elements rendus indisponibles par le perte d'un ensemble d'opérations $e \subseteq O$.

Le *degré de criticité*, défini par la définition 4 et l'équation 4.8, permet de quantifier le nombre de t-éléments perdus suite à la disparition d'un élément de l'architecture représenté par un ensemble d'opérations. Dans le cas où tous les t-éléments sont rendus indisponibles, le degré de criticité de l'élément considéré est égal à 1, si tous les t-éléments restent disponibles le degré de criticité est nul.

Définition 4 *Le degré de criticité $d_{C,T}(e)$ d'un ensemble d'opérations e dans un contexte C par rapport à T est le ratio des t-éléments rendus indisponibles par la perte de e sur le nombre d'éléments de T.*

$$d_{C,T}(e) = \frac{|Imp_{C,T}(e)|}{|T|} \tag{4.8}$$

A partir de ces définitions, il est alors possible de calculer la criticité d'une opération ($e = o, o \in O$), d'un ensemble d'opérations, d'une ressource ($e = r, r \subseteq O$) ou d'un ensemble de ressources ($e = \bigcup_i (r_i), r_i \subseteq O$). Les résultats de criticité peuvent être obtenus par rapport à un ensemble T qui peut correspondre à un ensemble d'opérations de l'architecture si chaque t-élément ne comporte qu'une opération. Ils peuvent aussi correspondre à des fonctions de l'architecture si chaque t-élément est constitué des opérations implémentant cette fonction. L'utilisation de t-éléments composés de toutes les permutations d'opérations permettant de réaliser une séquence de fonction permet d'obtenir des résultats de criticité par rapport à des gammes logiques.

La mise en œuvre du calcul de criticité est présentée dans la section suivante.

Mise en œuvre du calcul de criticité

La figure 4.3 présente la mise en œuvre du cadre d'analyse pour le calcul de la criticité des éléments. Il s'agit des grandes étapes menant à la détermination des résultats de criticité.

Le modèle d'analyse utilisé pour réaliser les calculs est le graphe d'accessibilité opérationnelle (GAO) décrit par le méta-modèle de la figure 4.4. Ce modèle est obtenu automatiquement à partir de la modélisation de l'architecture selon les règles énoncées à la page 91.

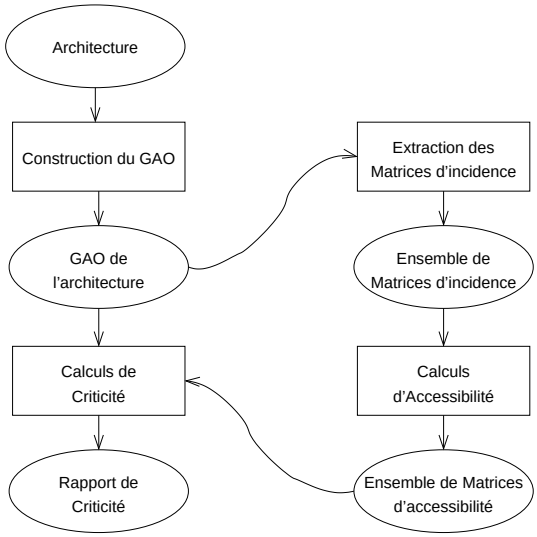


FIG. 4.3 – Obtention du rapport de criticité

La criticité d'un élément est calculée en supprimant les opérations correspondant à cet élément du graphe et en calculant la fermeture transitive de la matrice d'incidence associée (indiquant les relations de précédence entre les nœuds). Pour chaque élément considéré, il faut donc extraire la matrice d'incidence correspondante et calculer la matrice d'accessibilité (obtenue par le calcul de la fermeture transitive) à partir de celle-ci. Les calculs d'accessibilité sont réalisés en une seule fois pour tous les éléments, les matrices sont ensuite utilisées pour déterminer la disponibilité des t-éléments. Les degrés de criticité sont ensuite associés aux éléments du modèle sous la forme d'un modèle de rapport.

Les différents modèles et étapes composant le flot présenté à la figure 4.3 sont présentés dans les sections suivantes.

Le graphe d'accessibilité opérationnelle

Les définitions de la criticité présentées précédemment mettent en œuvre une structure de graphe permettant de déterminer la disponibilité d'un t-élément. Cette structure, appelée contexte correspond au graphe d'accessibilité opérationnelle (GAO) défini dans [Berruet, 1998]. Celui-ci correspond à une réduction du contexte utilisé comme modèle d'analyse pour réaliser les calculs d'accessibilité.

Le GAO est un graphe orienté, constitué de sommets correspondant à des opérations réalisées dans le système et dont les arrêtes représentent la relation d'accessibilité entre ces opérations.

Dans un souci de simplification, les sommets représentent des opérations composées. La composition des opérations se fait à l'aide de connecteurs de type "et" et de type "ou". L'utilisation d'opérations composées permet d'indiquer que plusieurs opérations sont nécessairement exécutées lors de la réalisation de l'opération de niveau supérieur dans le cas d'un "et". Dans le cas d'un "ou", au moins l'une des opérations du niveau inférieur est réalisée.

La figure 4.4 décrit le méta-modèle du GAO. Les sommets sont représentés par l'élément `Node` (nœud), les arrêtes par des liens (`Link`) entre ces nœuds. Un nœud du GAO contient une `Operation`. Cette opération est soit une opération agrégée (`AggregateOperation`), elle contient alors à son tour d'autres opérations et peut prendre le type `AndAggregate-Operation` ou `OrAggregateOperation`. Les opérations élémentaires, représentées par l'élé-

ment `ElementaryOperation` sont liées à une fonction (`Function`) et une ressource (`Resource`).

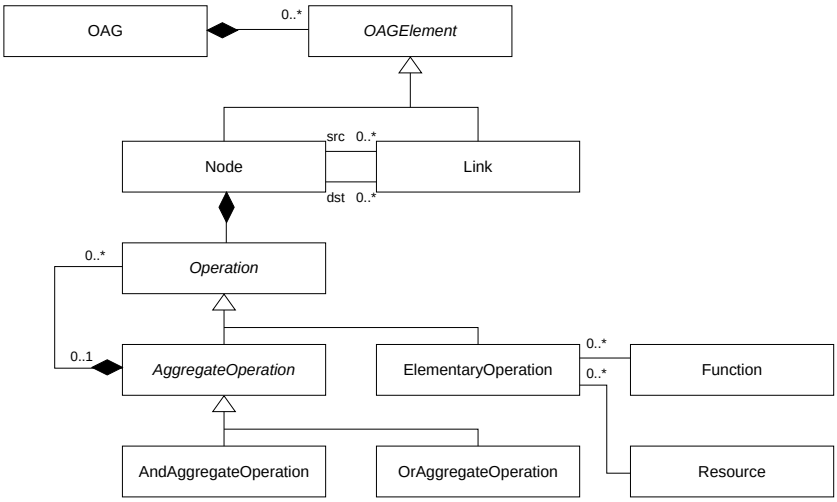


FIG. 4.4 – Méta-modèle du graphe d'accessibilité opérationnelle

Obtention du GAO à partir du modèle de l'architecture

Pour l'analyse de criticité sur l'architecture, le contexte considéré est l'architecture elle-même. Toutes les opérations de l'architecture, qu'elles soient implicites comme les opérations de transfert ou explicites comme c'est le cas pour les opérations potentielles de traitement ou de stockage, sont représentées dans le GAO.

Au niveau de la transformation de modèles, un nœud du GAO est créé pour chaque ressource stationnaire de l'architecture. Ce nœud contient une opération de stockage actif indiquant que si la ressource ne peut plus supporter de pièce, aucun traitement ne pourra être réalisé sur un produit. Cette opération de stockage actif est reliée par un connecteur "et" à un ensemble d'opérations représentant les actions réalisées sur la ressource et qui sont reliées par un connecteur "ou". Ces opérations sont : une opération de "non-traitement" indiquant que l'on peut ne rien faire, une opération pour chaque opération potentielle (de traitement ou de stockage) pouvant être réalisée sur la ressource.

Un nœud est aussi créé pour chaque connexion, les différentes opérations de transfert correspondantes y sont reliées avec un connecteur "ou", chacune de ces opérations de transfert correspond à la réalisation de la connexion par une ressource de transport.

Le résultat de la transformation de l'architecture de l'exemple présenté à la section 3.3.1 et décrit textuellement en annexe A est donné à la figure 4.5. Chaque nœud du GAO est représenté par une boîte rectangulaire dans laquelle l'opération qu'il contient est décrite. Les liens sont représentés par des flèches.

Détermination du degré de criticité à partir du calcul de criticité

Le GAO, une fois obtenu pour une architecture, permet après avoir effectué les calculs d'accessibilité, de déterminer la possibilité d'accéder à une opération à partir d'une autre en réalisant plusieurs autres opérations.

La disponibilité d'un ensemble d'opération conformément à la définition 2 est obtenue en vérifiant l'accessibilité de cette opération depuis l'entrée ainsi que l'accessibilité de la sortie du système depuis cette opération.

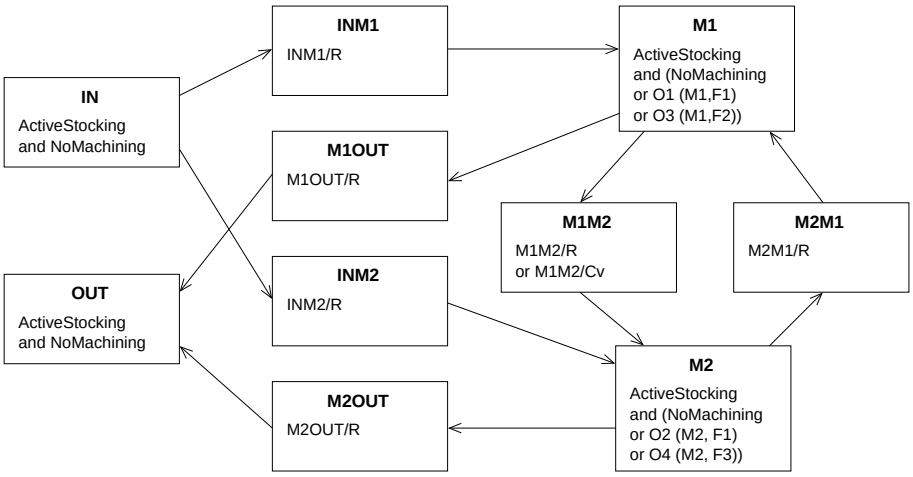


FIG. 4.5 – Graphe d'accessibilité opérationnelle pour l'exemple des deux machines

Lorsque l'accessibilité par rapport à des ensembles d'opérations (opérations, fonctions) doit être déterminé, il est possible de procéder directement conformément aux définitions, en déterminant la disponibilité des opérations appartenant à ces ensembles.

Dans le cas des séquences de fonctions, nous avons choisi de ne pas appliquer directement la définition en utilisant un algorithme qui semble plus adapté à l'utilisation de la structure manipulée. Cet algorithme présenté sur le listing 4.2 permet de déterminer s'il existe une séquence d'opérations implémentant, dans l'ordre, les fonctions de la séquence.

Listing 4.2 – Détermination de la criticité par rapport à une séquence de fonctions

```
1  cur_nodes = IN;
2  cur_func = function_sequence.first;
3
4  while cur_func != {}
5    next_nodes = cur_func.operations.nodes;
6    cur_nodes = next_nodes->select(e |
7                  cur_nodes->exists (n |
8                  accessible (n, e)));
9    cur_func = cur_func.next;
10 end while;
11
12 if cur_nodes != {} then 'la sequence de fonctions est disponible';
```

L'algorithme utilise trois variables. `cur_nodes` est une liste contenant l'ensemble des nœuds actuellement visités, il est donc initialisé à la valeur '{IN}' au début. `cur_func` contient la fonction que l'on souhaite réaliser et est initialisée avec la première fonction de la séquence. La troisième variable, `next_nodes` est utilisée pour construire le prochain`cur_nodes`. Cette construction se fait en sélectionnant, parmi les nœuds des opérations implémentant la fonction considérée, ceux qui sont accessibles à partir de l'un des nœuds de `cur_nodes`.

L'algorithme se termine quand toutes les fonctions de la séquence ont été visitées. Le résultat est donné par la taille de `cur_nodes`. Si cette liste est vide, cela signifie qu'à une itération de l'algorithme, aucun nœud réalisant l'une des opérations implémentant la fonction considérée n'était disponible.

La détermination de la disponibilité d'un produit se fait à partir de la disponibilité des séquences de fonctions le réalisant. Un produit est disponible si au moins une séquence de fonctions le réalisant est disponible.

Exemples de résultats

En considérant le contexte de l'architecture de l'exemple des deux machines, la ressource $M2$ réalise les opérations $O2$ et $O4$. La criticité de $M2$ par rapport aux fonctions d'usinage : $F1$, $F2$, $F3$ est obtenue en construisant l'ensemble des éléments cibles $T = \{\{O1, O2\}, \{O3\}, \{O4\}\}$. La perte de $M2$ entraîne la disparition de $O2$ et de $O4$. D'après le GAO, $M1$ reste disponible donc aucune autre indisponibilité n'est induite par la perte de $M2$. Seuls $O2$ et $O4$ sont alors enlevés des t-éléments, ce qui conduit à la construction du nouvel ensemble : $\{\{O1\}, \{O3\}, \{\varnothing\}\}$ dans lequel sur les trois ensembles contenus à l'origine, celui correspondant à $F3$ est vide ($F3$ n'est plus implémentée par aucune opération). Le degré de criticité de $M2$ par rapport aux fonctions de traitement dans le contexte de l'architecture est donc : $d_{C,T}(M2) = 1/3$ ce qui signifie que la perte de l'élément $M2$ entraîne la disparition d'un tiers des fonctions du systèmes.

L'application du cadre d'analyse permet d'obtenir automatiquement les résultats pour toutes les ressources et opérations du système par rapport aux fonctions et aux séquences de fonctions. Les résultats obtenus pour les opérations de traitement de l'architecture sont présentés dans le tableau 4.2. En ce qui concerne les ressources, leurs degrés de criticité sont donnés dans le tableau 4.3.

Opération	$d_{\mathcal{A},F}$	$d_{\mathcal{A},FS}$
O1	0	0
O2	0	0
O3	0.33	1
O4	0.33	1

TAB. 4.2 – Degrés de criticité des opérations dans l'exemple des deux machines

Au niveau des opérations, $F1$ est implémentée à la fois par $O1$ et $O2$, ces deux dernières opérations ne sont donc pas critiques. En ce qui concerne $O3$ et $O4$ elles sont critiques par rapport à la fonction qu'elles implémentent et empêchent donc la réalisation de la séquence de fonctions $P1\_S1$. En ce qui concerne les opérations de transfert, non représentées dans le tableau, leur degré de criticité est nul. Le GAO permet en effet de se rendre compte que la suppression d'un nœud de transfert n'empêche l'accès à aucun nœud de traitement.

Ressource	$d_{\mathcal{A},F}$	$d_{\mathcal{A},FS}$
M1	0.33	1
M2	0.33	1
IN	1	1
OUT	1	1
R	1	1
Cv	0	0

TAB. 4.3 – Degrés de criticité des ressources dans l'exemple des deux machines

L'analyse de la criticité des ressources sur un exemple tel que les deux machines donne des résultats prévisibles. Les ressources IN et OUT sont les uniques points d'entrée et de sortie du système. Il est donc logique qu'elles soient critiques. Le robot s'occupe du chargement et déchargement des ressources, c'est pourquoi il est critique. L'opération de transfert réalisée par le convoyeur peut être réalisée par le robot il n'est donc pas critique. Finalement, concernant les ressources de traitement M1 et M2, elles réalisent toutes deux une opération critique et entraînent la perte d'une fonction d'usinage qui rend la séquence de fonctions $P1\_S1$ irréalisable.

4.3.7 Critères globaux de criticité obtenus à partir de la criticité des éléments

Les résultats de criticité obtenus sur chacun des éléments de l'architecture peuvent ensuite être utilisés pour déterminer des critères globaux. Ces critères sont le nombre d'éléments critiques, le degré de criticité moyen de ces éléments ainsi que le degré de criticité maximal. Utilisés correctement, ces critères permettent d'avoir un aperçu global de la tolérance de l'architecture.

Définition des critères globaux

Les critères globaux sont obtenus sur un ensemble d'éléments noté $E = \{e_1, e_2, e_3, ...e_i\}$ avec $e_i \subseteq O$. Les e_i sont appelés e-éléments. E est défini sous l'hypothèse que les défaillances des éléments qui le constituent sont équiprobables.

Afin de simplifier la définition des critères, la fonction $CE_{C,T}(E)$ est définie, elle renvoie les éléments de E, critiques par rapport à T dans le contexte C.

Le premier critère global introduit est appelé *indice de criticité* et se note $CI_{C,T}(E)$. Il est introduit par l'équation 4.9 et renvoie la proportion d'éléments critiques. Lorsque sa valeur est égale à 1, cela signifie que tous les e-éléments (éléments de E) sont critiques.

$$CI_{C,T}(E) = \frac{|CE_{C,T}(E)|}{|E|} \tag{4.9}$$

Le second critère se nomme *degré de criticité moyen des éléments critiques* et se note $\overline{CR_{C,T}(E)}$. On l'obtient en appliquant la formule donnée à l'équation 4.10. Une valeur de 0.5 signifie que la disparition d'un élément critique entraînerait en moyenne l'indisponibilité de la moitié des éléments.

$$\overline{CR_{C,T}(E)} = \frac{\sum_{e \in CE_{C,T}(E)} (d_{C,T}(e))}{|CE_{C,T}(E)|} \tag{4.10}$$

Le dernier critère est le *degré de criticité maximal* noté $CR^{\max}_{C,T}(E)$ et défini par l'équation 4.11. Si ce critère prend la valeur 1, cela signifie que l'indisponibilité d'un élément de E peut rendre tout le système indisponible.

$$CR^{\max}_{C,T}(E) = \max_{e \in CE_{C,T}(E)} (d_{C,T}(e)) \tag{4.11}$$

Mise en œuvre du calcul des critères globaux

L'obtention des critères globaux de criticité sur l'architecture se base sur la criticité des éléments considérés. Ce calcul se fait donc en utilisant le rapport de criticité obtenu à la section 4.3.6. La lecture de ce rapport permet d'obtenir $CE_{C,T}(E)$, nécessaire à la détermination des différents critères globaux.

La description du système utilisée pour le calcul des critères globaux de criticité est le modèle de l'architecture auquel sont adjoints les résultats de criticité obtenus précédemment. La description du système constitue le modèle d'analyse de celui-ci et les critères globaux résultent d'une transformation de modèles.

Exemple

Dans le cadre de l'exemple des deux machines, pour un ensemble E constitué des deux machines $M1$ et $M2$ et en considérant comme ensemble T les fonctions de traitement, l'indice de criticité $CI_{C,T}(M1, M2) = 1$ indique que les deux ressources $M1$ et $M2$ sont critiques. La criticité moyenne $\overline{CR_{C,T}}(M1, M2) = 0.33$ indique qu'en moyenne, la perte d'une ressource de traitement entraîne la disparition d'un tiers des fonctions de traitement. Pour la criticité maximale $CR^{\max}_{C,T} = 0.33$, elle permet de savoir qu'aucune des deux ressources considérées ne peut empêcher la réalisation d'une fonction (bien que ce soit le cas pour les séquences de fonctions).

4.3.8 Bilan sur l'analyse de l'architecture

Le cadre d'analyse a été appliqué afin d'obtenir différentes métriques caractérisant l'architecture. Le travail est automatique grâce aux transformations de modèles qui ont été mises en œuvre.

L'utilisation des différentes analyses permet de caractériser une architecture. Des mesures quantitatives sur le modèle permettent ainsi de connaître la taille de l'architecture, la dispersion des différentes fonctions (caractérisée par le parallélisme potentiel des traitements) ainsi que la tolérance de l'architecture. Dans le cas des deux machines, on sait que si on perd l'une des ressources de traitement, aucune configuration ne permettra de réaliser la séquence de fonction définie.

La section suivante présente les analyses mises en œuvre sur les configurations. Si les analyses définies pour l'architecture sont alors utilisées, de nouvelles analyses, propres aux configurations sont introduites.

4.4 Analyses mises en œuvre sur les configurations

Cette section traite de l'analyse des configurations, dans le cadre défini à la section 4.2.

Dans un premier temps, l'application à la configuration des analyses présentées sur l'architecture est introduite. Ensuite sont présentées des analyses spécifiques à la configuration qui sont le calcul du temps de réalisation d'une séquence d'opération ainsi que des analyses de cohérence entre la configuration et l'état des produits ainsi qu'entre la configuration et le mode des ressources.

4.4.1 Application des analyses définies sur l'architecture à la configuration

La notion de contexte a été introduite à la section 4.3.1, lors de la définition des analyses pour l'architecture. Un contexte est homogène à une architecture et correspond à un sous-ensemble de celle-ci, cette notion de contexte est utilisée afin d'appliquer, à la configuration, les différentes analyses définies sur l'architecture.

En effet, on peut faire correspondre un contexte à une configuration, en sélectionnant pour ce contexte et suivant les mesures que l'on souhaite réaliser, une partie de l'architecture sur laquelle on souhaite obtenir des résultats.

Construction du contexte

L'application des analyses définies sur l'architecture nécessite la construction du contexte sur lequel elles vont être appliquées.

Pour chaque configuration, on peut considérer un contexte regroupant les éléments de l'architecture effectivement utilisés dans la configuration. Pour évaluer la capacité de reconfiguration du système dans cette configuration, la définition de nouveaux contextes est envisageable. La construction d'un contexte peut par exemple retenir toutes les opérations potentielles ainsi que les connexions pouvant être réalisées par des ressources présentes dans la configuration. Le contexte ainsi construit permet alors d'évaluer les capacités de reconfiguration dans le cas de reconfigurations significatives. Une autre construction de contexte, correspondant aux reconfigurations majeures, reprend tous les éléments de l'architecture physique en ne restreignant que l'architecture logique.

La construction d'un contexte en prévision d'une reconfiguration se rapproche donc du concept de niveaux de reconfiguration présenté à la section 1.2.2. Les construction de contexte évoquées précédemment correspondent en effet aux trois niveaux définis dans [Berruet, 1998].

Ainsi, pour une configuration Cf, trois contextes peuvent être définis en considérant les niveaux de configuration. Cf_{min} ne contient que des éléments définis dans la configuration. Cf_{sig} y ajoute toutes les opérations potentielles ou connexions réalisées par des ressources présentes dans la configuration. Cf_{maj} ajoute à Cf_{min} l'intégralité de l'architecture physique.

Application des analyses structurelles définies sur l'architecture

Les analyses structurelles définies sur l'architecture (nombres d'éléments, parallélisme des transferts, parallélisme de traitement) peuvent être appliquées à la configuration. Le calcul des analyses structurelles à partir de Cf_{min} indique les caractéristiques structurelles de la configuration.

La configuration introduit de nouveaux éléments par rapport à l'architecture. Ces éléments sont les séquences de transfert et les séquences d'opérations. Ils peuvent être structurellement analysés.

Application aux calculs de criticité

Dans le cas de l'analyse de la criticité des éléments de la configuration ainsi que du calcul des critères globaux associés, l'application de la notion de contexte permet d'avoir des résultats sur la capacité d'évolution du système dans une configuration donnée. Suivant le niveau de reconfiguration considéré, on peut ainsi considérer l'utilisation des opérations actuellement utilisées, l'utilisation des opérations des ressources utilisées ou encore toutes les opérations du système. La question *"Le système peut-il continuer à fonctionner après une défaillance en n'autorisant que les reconfigurations significatives ?"* peut alors être traitée.

Deux méthodes donnant des résultats identiques ont été appliquées pour obtenir les résultats de criticité dans un contexte. La première consiste à construire un GAO correspondant au contexte considéré [Lamotte et al., 2006], un exemple de GAO obtenu pour la configuration de l'exemple des deux machines (section 3.4.3) est présenté à la figure 4.6, seules les opérations effectivement utilisées ont été conservées. La seconde méthode consiste, pour chaque élément, à calculer son degré de criticité dans l'architecture en considérant un autre élément constitué de l'élément original et de toutes les opérations hors contexte. Ces deux méthodes permettent d'utiliser les outils développés pour l'architecture en les modifiant légèrement.

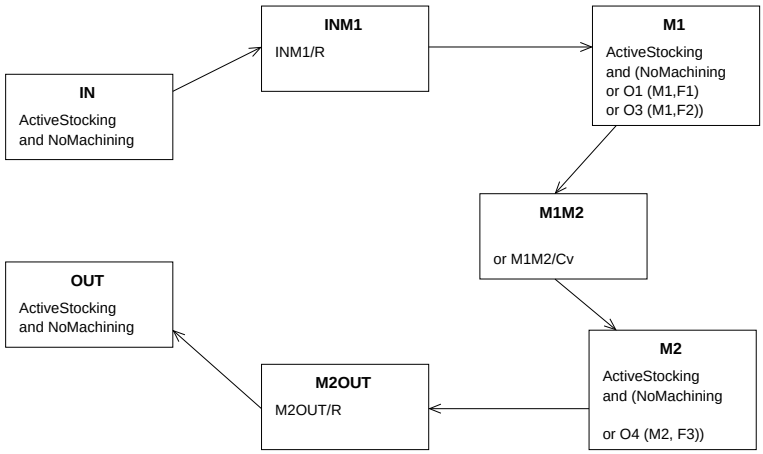


FIG. 4.6 – GAO pour la configuration nominale de l'exemple des deux machines en vue d'une reconfiguration mineure

Dans le cas de la configuration définie pour l'exemple des deux machines dont le GAO est proposé à la figure 4.6, toutes les opérations sont critiques. En construisant un contexte prenant en compte les reconfigurations significatives, c'est-à-dire en ajoutant toutes les opérations des ressources de la configuration dans le contexte. Les ressources étant toutes utilisées, les résultats sont identique à ceux obtenus pour l'architecture (le GAO est alors le même).

4.4.2 Analyse de la cohérence entre l'architecture et la configuration

La configuration et l'architecture sont définies dans deux modèles différents. La vérification de la cohérence entre une configuration et l'architecture pour laquelle elle a été définie est nécessaire afin d'éviter des erreurs de saisie lors de la définition des modèles ou encore de vérifier que l'utilisation d'une ancienne configuration avec une architecture qui aurait subi des modification est toujours possible.

Définition de la cohérence d'une configuration vis à vis d'une architecture

Une configuration est cohérente par rapport à son architecture si chaque élément défini dans la configuration utilise correctement les éléments de l'architecture sur lesquels il est basé.

Au niveau logique, les fonctions instanciées dans la configuration logique doivent avoir été définies dans l'architecture logique. Au niveau physique, les ressources utilisées pour réaliser les transferts doivent avoir été définies ainsi que les connexions. Les séquences de fonctions réalisées par les séquences d'opérations doivent exister dans l'architecture. De plus, l'ordre des opérations dans les séquences d'opérations doit respecter l'ordre défini dans les séquences de fonctions.

Mise en œuvre

La mise en œuvre de l'analyse de cohérence est réalisée à l'aide d'une transformation de modèle. La description du système constituée des modèles de configuration et d'architecture sont utilisées comme modèles d'analyse.

Les règles écrites pour réaliser la vérification sont inspirées de [Bézivin and Jouault, 2005] mais en mettant en jeu deux modèles différents en entrée. La majeure partie des contraintes

vérifiées concernent la définition correcte dans l'architecture d'un élément utilisé dans la configuration, il est donc important de ne pas considérer que les éléments existent dans l'architecture lors de l'écriture des règles de vérification.

4.4.3 Calcul du temps de réalisation d'une séquence d'opérations

Les séquences d'opération déterminent exactement comment sont enchaînées les opérations. Une information quantitative concernant la *durée de réalisation* de celle-ci peut donc être obtenue. Comme les séquences d'opérations sont réalisées en concurrence et que certaines ressources peuvent être partagées, la durée calculée n'est qu'une borne minimale.

Définition

La durée δ_s de réalisation de la séquence s est donnée par la somme des durées des opérations composant cette séquence.

Mise en œuvre

Dans le langage de description, les éléments de l'architecture et de la configuration ne disposent pas d'attributs permettant d'attribuer une durée aux opérations, ces attributs peuvent néanmoins être mentionnés dans un modèle annexe, ajoutant les attributs nécessaires et correspondant au domaine. Lorsque aucune information n'est ajoutée, chaque opération élémentaire du GAO associé à l'architecture a une durée égale à 1.

Le calcul de la durée d'une séquence d'opérations est réalisé à partir des descriptions de l'architecture et de la configuration, utilisées comme modèles d'analyse par une transformation de modèle qui transforme chaque séquence d'opérations de l'architecture en un résultat du rapport d'analyse. Chaque étape d'une séquence de transfert a une durée de 1, ce qui permet de calculer la durée d'une opération de transfert. La durée des opérations de transfert est ajoutée à celle des opérations stationnaires qui ont aussi pour valeur 1.

Une transformation spécifique peut aussi être écrite en utilisant des durées spécifiées dans un modèle annexe aux modèles de l'architecture et de la configuration.

Exemples

Dans le cadre de la configuration proposée pour l'exemple des deux machines, la durée de la séquence d'opérations est égale à $\delta_{OS1} = 6$.

4.4.4 Analyse de la cohérence entre une configuration et l'état du système

Le maintien de la cohérence entre une configuration et le mode des ressources est nécessaire. Il en est de même entre la configuration et l'état des produits dans le système.

Lorsqu'une configuration n'est plus cohérente avec le mode des ressources, après par exemple la panne de l'une d'entre elles, il est nécessaire de procéder à une reconfiguration. De même, si la cohérence avec l'état des produits n'est plus assurée, il faut reconfigurer pour tenir compte de l'état réel du système.

Lors d'une reconfiguration, le mode des ressources est modifié. Parmi les modifications apportées, la plus simple consiste à mettre sous tension ou hors-tension la ressource. On peut aussi envisager une réparation, qui sera plus coûteuse. Par rapport a l'état des produits dans le système, la reconfiguration doit rendre cet état compatible avec la configuration choisie. Changer

de configuration entraîne donc un coût supplémentaire qui constitue un élément à prendre en compte lors du choix de la nouvelle configuration.

Définition

La configuration est cohérente avec le mode des ressources si les ressources auxquelles la configuration fait référence sont disponibles.

Une deuxième cohérence doit être respectée. Il s'agit de la cohérence entre la configuration et l'état des produits dans le système. La configuration est cohérente par rapport à l'état des produits dans le système si cet état est *admissible* par la configuration. C'est à dire que le système peut faire évoluer de manière correcte cet état.

Le coût d'une reconfiguration prend en compte le coût des différents changements de mode qui ont lieu pendant la reconfiguration ainsi que celui de la mise en conformité de l'état des produits dans le système avec la nouvelle configuration. Cette mise en conformité peut entraîner le passage par des configurations intermédiaires comme nous le verrons au chapitre 5.

Mise en œuvre

Ces métriques sont utilisées en fonctionnement et dépendent de l'application. Elles nécessitent la mise en place de modèles représentant respectivement le mode des ressources et l'état des produits dans le système. Leur mise en œuvre est dépendante du système étudié, et en particulier de son caractère observable et des contraintes imposées au processus de reconfiguration.

4.4.5 Bilan sur l'analyse de la configuration

Les différentes analyses mises en œuvre sur la configuration ont été présentées dans cette section. La notion de contexte a été utilisée pour adapter les analyses effectuées sur l'architecture.

De nouvelles analyses, sont ajoutées par rapport aux analyses de l'architecture. Ces analyses prennent leur place au sein du processus de reconfiguration.

La section suivante présente l'utilisation des résultats d'analyse sur différents exemples. Des topologies usuelles ainsi que les deux exemples présentés au chapitre 3 sont analysées dans la section suivante.

4.5 Résultats d'analyse sur des exemples

Les analyses présentées tout au long de ce chapitre ont été instrumentées. Elles ont donc pu être appliquées sur une grande variété d'exemples.

Ces analyses ont tout d'abord été appliquées afin de comparer différentes topologies standard pour la construction d'architectures physiques dans le but de les comparer. Les résultats de cette analyse sont donnés dans un premier temps. Ensuite, les exemples introduits dans le chapitre 3 sont analysés.

4.5.1 Résultats d'analyse sur des architectures physiques usuelles

Pour permettre la validation des analyses développées sur l'architecture, nous avons décidé de comparer différentes topologies qui peuvent être employées comme patrons pour la concep-

tion d'une architecture physique. Ces topologies sont issues du monde des réseaux dans lequel elles sont utilisées.

Présentation des topologies

Liaison point-à-point La liaison point-à-point, représentée visuellement à la figure 4.7, constitue la liaison la plus simple entre deux ressources. Il s'agit d'un lien direct entre elles, réalisé par une ressource de transport. Ce type de lien constitue la base de toutes les topologies.

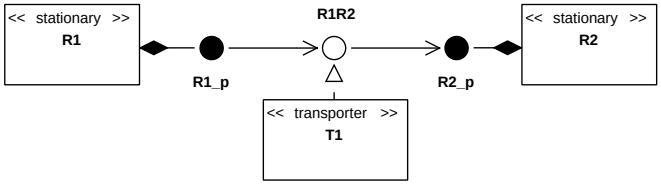


FIG. 4.7 – Représentation visuelle d'une liaison point-à-point

La ligne de transfert Une ligne de transfert est constituée de plusieurs ressources stationnaires reliées par des liaisons point-à-point. Les liens entre les différentes ressources peuvent être bi-directionnels, la ligne peut aussi être bouclée si on relie la dernière ressource de la ligne à la première. L'utilisation d'une ou plusieurs ressources de transport pour réaliser les connexions entre les ressources stationnaires est envisageable. Une ligne uni-directionnelle et sans rebouclage constituée de trois ressources stationnaires est présentée à la figure 4.8.

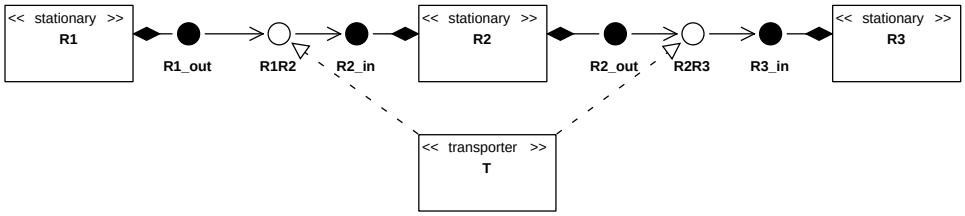


FIG. 4.8 – Représentation d'une ligne de transfert

Le Bus Un bus est constitué d'une ligne de transfert dont les nœuds sont des ressources stationnaires de routage reliées par une liaison point-à-point aux ressources stationnaires de traitement. La perte d'une ressource de traitement n'induit plus nécessairement la perte du système. La figure 4.9 présente un exemple de bus bi-directionnel composé de trois ressources stationnaires.

Les bus sont couramment utilisés en électronique, un bus à été utilisé pour représenter le système électronique défini à la section 3.5.2. C'est aussi une manière de modéliser un convoyeur dans le cas des systèmes de production.

L'étoile Dans la topologie en étoile, telle que représentée à la figure 4.10, les ressources de traitement sont reliées à une ressource centrale par des liaisons point-à-point. Cette topologie est utilisée comme une abstraction pour d'autres topologies en encapsulant les moyens de communication dans la ressource centrale.

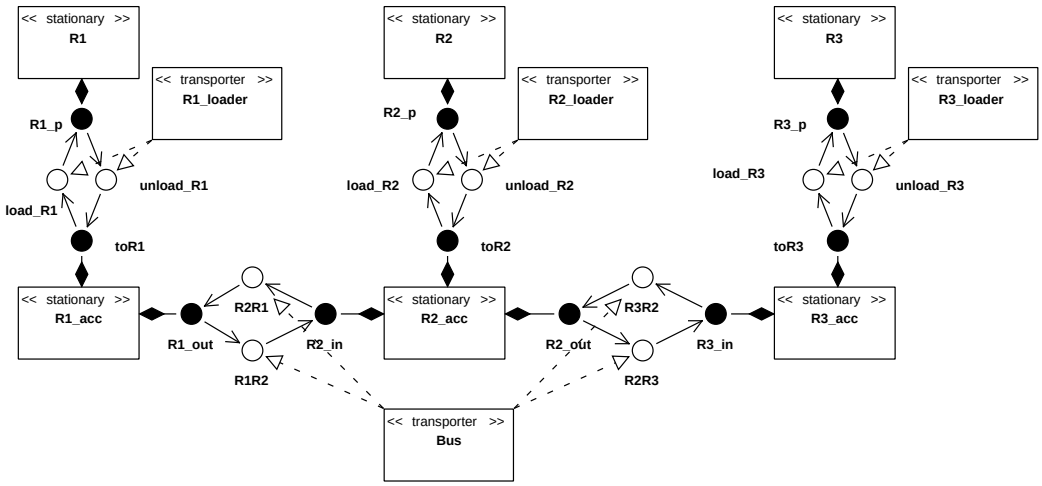


FIG. 4.9 – Représentation d'un bus

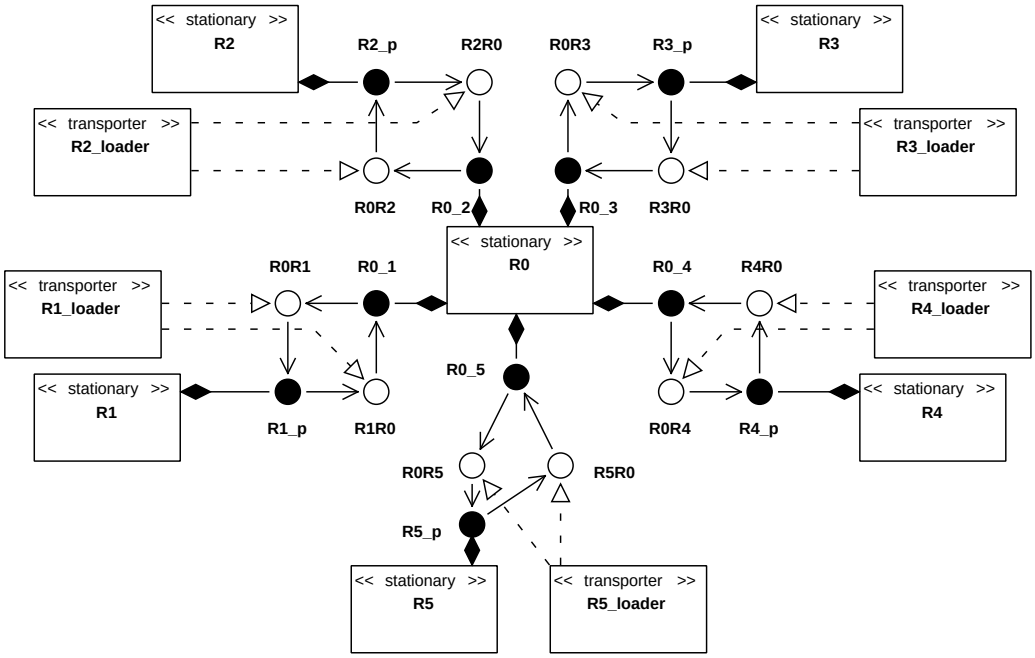


FIG. 4.10 – Représentation d'une étoile

Le mesh Dans une topologie mesh, les ressources sont toutes interconnectées. Un exemple d'une telle topologie est fourni sur la figure 4.11.

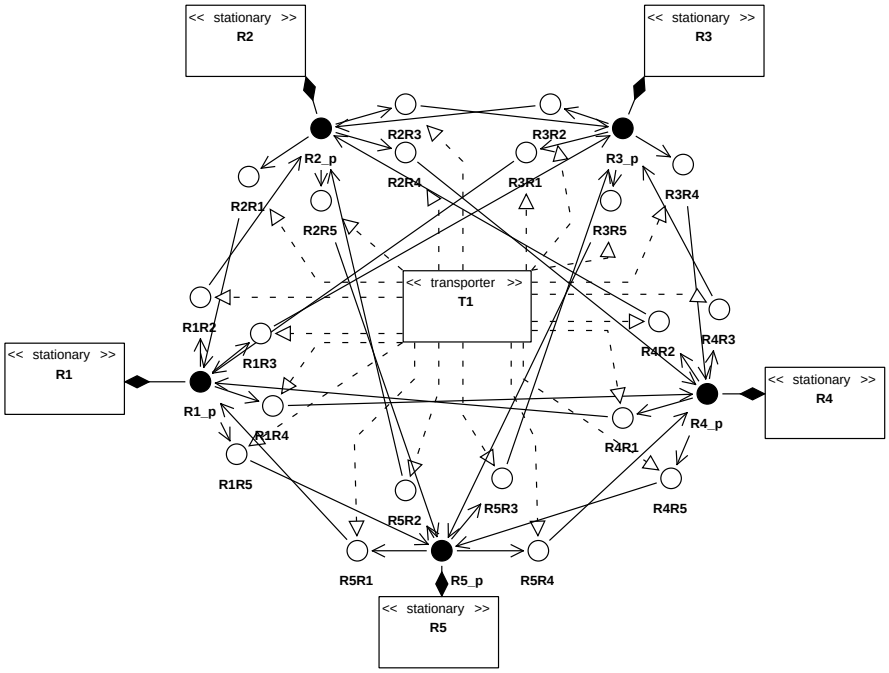


FIG. 4.11 – Représentation d'un mesh

La grille La grille est une version dégénérée de mesh. Les ressources stationnaires sont placées sur un quadrillage, chacune d'entre elle est reliée à ses plus proches voisines par une liaison point-à-point. Comme pour la ligne, les ressources de la grille peuvent être utilisées comme vecteur pour le transport vers les ressources de traitement, nous nommerons gridbus cette variante de la grille.

Normalisation des architectures Il est nécessaire de normaliser les différentes topologies qui viennent d'être introduites avant de pouvoir les comparer. Au cours des analyses, chaque architecture sera composée de neuf ressources de traitement, ce qui permet d'avoir une grille de taille raisonnable.

Le nommage des différentes architectures suit des règles précises. Le nom de celle-ci débute par son type (ligne, bus ...), ensuite, le nombre de ressources de traitement qu'elle possède est indiqué suivi d'un 'R'. Il est ensuite fait de même pour le nombre de ressources de transport qui est suivi d'un 'T'. Si l'architecture est bouclée, son nom est post-fixé par 'l', si elle est bi-directionnelle il est post-fixé d'un b.

Dix topologies sont ainsi comparées. Cinq d'entre elles sont des lignes, la première est uni-directionnelle et dispose d'une ressource de transport (ligne9R1T), une deuxième ligne dispose de quatre ressources de transport pour chaque connexion (ligne9R8T), la troisième est bi-directionnelle (ligne9R1Tb), la quatrième est bouclée (ligne9R1Tbl) tout comme la dernière qui par contre est uni-directionnelle (ligne9R1Tl). Un bus est étudié, il dispose de liens bi-directionnels et est bouclé (bus9R1Tbl). Une étoile (etoile9R9T), un mesh (mesh9R1T), une grille (grille9R1T) et une gridbus (gridbus9R1T) sont aussi étudiés.

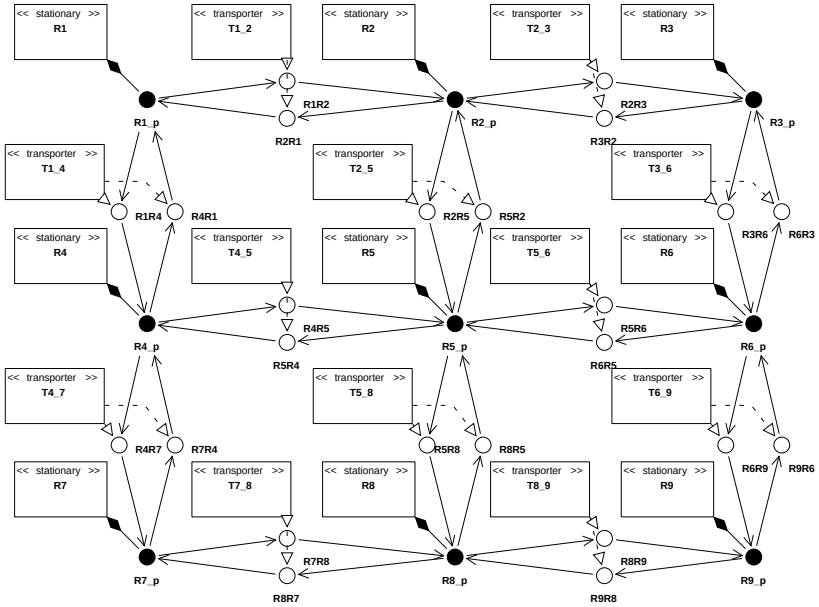


FIG. 4.12 – Représentation d'une grille

Il faut aussi normaliser l'architecture logique et les opérations. Chaque architecture décrit neuf fonctions de traitement $F\{1-9\}$ déployées sur les ressource de la façon suivante : chaque ressource Ri réalise l'opération Oi implémentant la fonction Fi de même que l'opération $O(9+i)$ réalisant la fonction $F(10-i)$. La ressource $R5$ est une exception puisqu'elle ne réalise qu'une opération : $O5$. Il n'y a donc pas d'opération $O14$.

Résultats d'analyse

Analyse structurelle Les critères retenus pour l'analyse des différentes architectures sont le nombre des ressources stationnaires, le nombre des ressources de transport, le nombre de connexions ainsi que les deux métriques indiquant le parallélisme potentiel des transferts. Les résultats d'analyses sont donnés dans le tableaux 4.4

A	$N_{St,A}$	$N_{Tr,A}$	$N_{Cx,A}$	CS_A	TP_A
ligne9R1T	11	1	10	0.91	0.09
ligne9R1Tb	11	1	18	1.64	0.09
ligne9R1Tbl	11	1	20	1.82	0.09
ligne9R1Tl	11	1	11	1	0.09
ligne9R8T	11	10	10	0.91	0.91
bus9R1T	20	10	38	1.9	0.5
etoile9R9T	12	10	20	1.67	0.83
grille9R1T	11	1	26	2.36	0.09
gridbus9R1T	20	10	44	2.2	0.5
mesh9R1T	11	1	90	8.18	0.09

TAB. 4.4 – Résultats des analyses structurelles sur les différentes topologies

Une seule architecture possède moins de connexions que de ressources stationnaires, il s'agit de ligne9R1T avec une valeur de 0.91.

L'architecture mesh est la plus coûteuse en terme de connexions, 90 y sont définies. Il en résulte une très forte capacité de parallélisation des transferts chaque ressource stationnaire dispose de plus de 8 connexions.

Analyse de sûreté de fonctionnement Les résultats de l'analyse de tolérance sur les différentes architectures sont donnés dans le tableau 4.5. Les critères globaux sélectionnés sont les indices de criticité des opérations de traitement (MOps), des opérations de transfert (XferOps) des ressources de traitement (MRes) et des ressources de transfert (XferRes). Le nombre d'éléments critiques ($CE_{C,T}$) pour chaque résultat est donné entre parenthèses.

Architecture	MOps	XferOps	MRes	XferRes
ligne9R1T	0.06 (1)	1 (10)	1 (9)	1 (1)
ligne9R1Tb	0.06 (1)	0.56 (10)	1 (9)	1 (1)
ligne9R1Tbl	0.06 (1)	0.1 (2)	0.33 (3)	1 (1)
ligne9R1Tl	0.06 (1)	0.9 (10)	1 (9)	1 (1)
ligne9R8T	0.06 (1)	1 (10)	1 (9)	1 (8)
bus9R1T	0.06 (1)	0.11 (4)	0.11 (1)	0.2 (2)
etoile9R9T	0.06 (1)	0.2 (4)	0.11 (1)	0.2 (2)
grille9R1T	0.06 (1)	0.08 (2)	0.33 (3)	1 (1)
gridbus9R1T	0.06 (1)	0.09 (4)	0.11 (1)	0.2 (2)
mesh9R1T	0.06 (1)	0 (0)	0.11 (1)	1 (1)

TAB. 4.5 – Résultats de criticité par rapport aux fonctions pour différentes topologies d'architecture

La criticité par rapport aux fonctions des opérations de traitement ne dépend pas de l'architecture considérée. Il n'y a, à chaque fois, qu'une seule opération critique : $O5$ qui est la seule opération implémentant $F5$, les autres fonctions sont toutes implémentées par deux opérations différentes.

En ce qui concerne la classification des architectures selon le critère de tolérance, les lignes sont hautement critiques. Le bouclage ainsi que l'utilisation de liaisons bi-directionnelles permet d'améliorer beaucoup les résultats (le résultat concernant les opérations de transfert descend à 0.1). L'utilisation d'un bus réduit la criticité des ressources de traitement. L'architecture la plus tolérante est le mesh, il s'agit aussi de la plus complexe structurellement. Le gridbus semble être un compromis intéressant.

4.5.2 Résultats d'analyse sur l'exemple en production

L'architecture et la configuration du convoyeur présenté à la section 3.5.1 sont maintenant analysées.

Analyse de l'architecture du convoyeur

Analyse structurelle Le convoyeur est composé de onze ressources stationnaires, de cinq ressources de transport et de vingt-deux connexions. $TP_A = 0.45$ et $CS_A = 2$ ce qui est proche du résultat obtenu pour un bus.

En ce qui concerne la partie logique, le parallélisme potentiel des trois séquences est égal à trois, ce qui signifie que chacune des trois séquences peut profiter des trois ressources pour paralléliser leur exécution.

Analyse de tolérance L'analyse de la tolérance de l'architecture du convoyeur nécessite la construction de son GAO qui est donné à la figure 4.13.

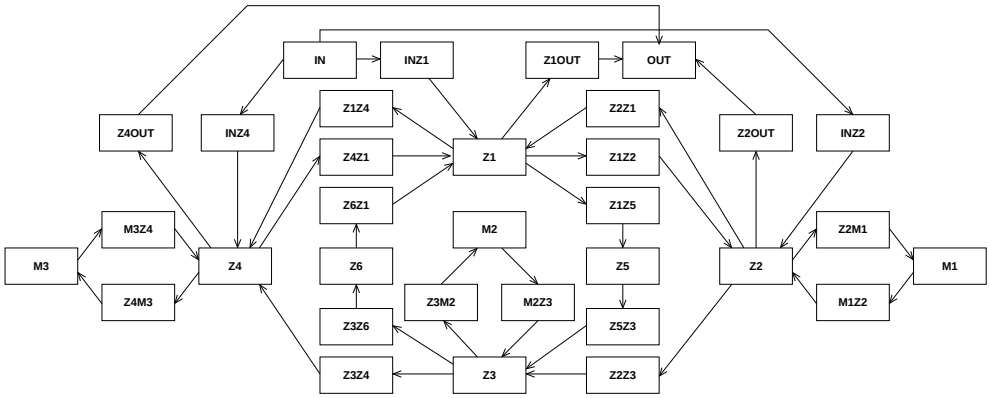


FIG. 4.13 – GAO du convoyeur

L'analyse de criticité indique que la ressource $M3$ est critique, en effet c'est la seule à pouvoir réaliser $F2$. Les ressources de transport critiques sont $R13$ et $R14$. $R13$ est le seul robot capable de charger $M3$ et $R14$ fait rentrer et sortir les produits du système.

Une seule opération de traitement est critique : $O_{F2,M3}$. Les stockages actifs sur $M3$ et $Z4$ sont aussi critiques puisqu'ils empêchent la réalisation de $O_{F2,M3}$. Deux opérations de transfert sont critiques, ce sont celles réalisées par $R13$.

D'après les critères globaux, le degré de criticité maximum des opérations de traitement et de transfert par rapport aux séquences de fonctions est égal à 0.66 ce qui signifie qu'il y a toujours au moins une séquence de fonctions qui peut être réalisée sur le système suite à la perte d'une opération. En ce qui concerne les ressources, le degré de criticité maximum par rapport aux séquences de fonctions est égal à 1. En effet la perte de $R14$ empêche la réalisation de n'importe quelle séquence de fonction.

Analyse et comparaison des configurations du convoyeur

La configuration CV1 mets en jeu une grande partie des ressources du système alors que le convoyeur est hors-service dans la configuration CV2, ce qui modifie la façon d'utiliser les ressources. Les deux configurations sont donc analysées et comparées.

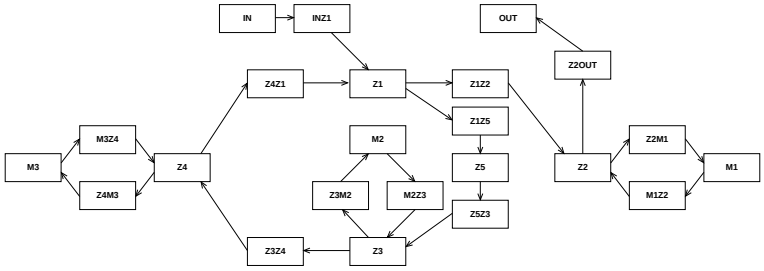
Analyse structurelle Le tableau 4.6 récapitule les résultats de l'analyse structurelle des deux configurations et les compare avec ceux obtenus pour l'architecture.

Métrique	$C = \mathcal{A}$	$C =$ CV1	$C =$ CV2
$N_{C,St}$	11	10	7
$N_{C,Tr}$	5	5	3
$N_{C,Cx}$	22	13	8
TP_C	0.45	0.5	0.42
CS_C	2	1.3	1.14
$PP_C(S1)$	3	3	2
$\delta_C(S1)$	-	24	11

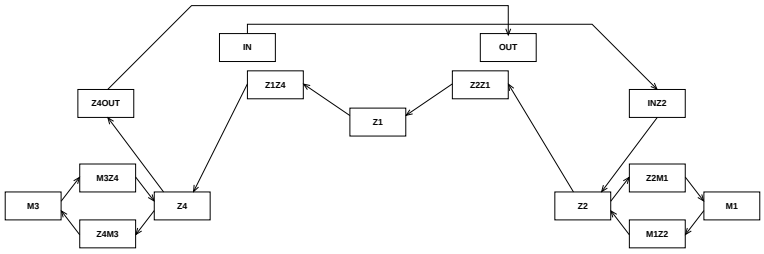
TAB. 4.6 – Résultats de l'analyse structurelle des deux configurations du convoyeur

Le nombre d'éléments utilisés dans CV2 est bien inférieur à celui de CV1 qui n'utilise déjà pas toutes les ressources. Le nombre de connexions par ressource stationnaire reflète bien le peu de choix de transferts restants dans la configuration dégradée dont les résultats à ce niveau se rapprochent de ceux d'une ligne de transfert. En ce qui concerne le parallélisme potentiel de traitement, une ressource de traitement n'est pas utilisée pour la réalisation de la séquence de fonctions. Néanmoins, la durée de réalisation de cette séquence est bien plus courte sur CV2.

Analyse de tolérance L'analyse de la tolérance des configurations se fait dans un contexte particulier. Envisageons dans un premier temps la possibilité d'effectuer une reconfiguration mineure, c'est à dire en utilisant les opérations déjà mises en œuvre par la configuration courante. Les GAO obtenus pour les deux configurations sont donnés aux figures 4.14. Ces deux GAO ne laissent apparaître que les opérations effectivement utilisées dans la configuration considérée.



(a) GAO pour CV1



(b) GAO pour CV2

FIG. 4.14 – GAO des configurations du convoyeur en vue d'une reconfiguration mineure

Dans ce contexte où seules des reconfigurations mineures sont envisagées. Toutes les opérations ainsi que les ressources sont critiques, à la fois par rapport aux fonctions et par rapport à la séquence de fonctions $S1$ réalisée dans la configuration, les degrés de criticité moyens des opérations et des ressources par rapport aux séquences de fonction ont pour valeur 100% ce qui signifie que la perte de n'importe quelle opération ou ressource empêche la poursuite de la production.

Le contexte peut alors être étendu. Cette extension peut être réalisée soit au niveau logique, soit au niveau physique. Au niveau logique, l'extension consiste à ajouter des fonctions, produits et séquences de fonctions en conservant la même configuration physique. Au niveau physique il s'agit d'ajouter des capacités de traitement au niveau des ressources ce qui a pour effet d'ajouter des opérations au GAO.

L'ajout de séquences de fonctions au contexte diminue le degré de criticité de certains éléments, tous ces éléments restent critiques mais le nombre de séquences de fonctions disponible après leur perte est plus important, certaines séquences qui ont été ajoutées peuvent toujours être réalisées. Pour la configuration nominale, le degré de criticité moyen des opérations par rapport aux séquences de fonctions passe à 89%, ce qui signifie que la disparition d'une opération critique entraîne, en moyenne, la disparition 89% des séquences de fonction. Pour les ressources,

le degré de criticité moyen passe à 91%. Pour la configuration CV2 seule le degré de criticité moyen des opérations est diminué, celui-ci passe 96%.

L'ajout des opérations réalisées par les ressources de la configuration (reconfiguration significative) diminue le nombre d'éléments critiques. Pour la configuration CV1, parmi les 26 opérations qui étaient critiques en considérant une reconfiguration mineures, 7 le sont encore si une reconfiguration significative est envisagée (le GAO est maintenant constitué de 38 opérations). Dans le cas de la configuration CV2, les résultats passent de 18 opérations critiques à 15 (sur 25) toutes les ressources restant critiques.

La configuration CV1 est donc plus tolérante que la configuration CV2. Il est toutefois nécessaire, pour profiter de cette tolérance, de considérer la mise en œuvre de reconfigurations significatives pour permettre la poursuite de la production après défaillance.

4.5.3 Résultats d'analyse sur l'exemple en électronique

Cette section traite de l'analyse de l'exemple électronique modélisé à la section 3.5.2. Comme pour le convoyeur, les résultats sur l'architecture ainsi que sur les deux configurations proposées sont présentés.

Analyse de l'architecture du système de traitement

Il est tout d'abord nécessaire de spécifier l'entrée et la sortie du système (ressources IN et OUT) avant de réaliser les analyses. L'entrée du système dans le système de vision correspond aux caméras, branchées sur la carte d'acquisition. La sortie doit rester accessible si le bus tombe en panne, mais ne peut pas être reliée uniquement à l'une des cartes qui peut aussi tomber en panne. La sortie est donc branchée à la fois sur le bus, où elle est accessible depuis n'importe quelle carte, et sur la première carte qui dans certaines configurations fonctionne uniquement avec la carte d'acquisition.

Analyse structurelle L'architecture du système est composée de dix ressources stationnaires, quatre ressources de transport ainsi que de dix-neuf connexions. Au niveau logique, quatre fonctions peuvent y être réalisées, formant deux séquences permettant d'obtenir quatre produits.

Les valeurs associées au parallélisme potentiel de transfert sont : $CS_A = 1.9$ et $TP_A = 0.4$, ces résultats correspondent à ce que l'on peut attendre d'une architecture de type bus.

Au niveau logique, le parallélisme potentiel de la séquence "choix_image" est de 3, celui de la séquence "annotation" est de 2.

Analyse de tolérance Les calculs de criticité sont réalisés sur le GAO de la figure 4.15 obtenu à partir de l'architecture de l'exemple électronique.

Quatre ressources sont critiques : IN, OUT, la Carte1 ainsi que CLink, la liaison entre IN et la carte 1, ce qui entraîne un taux de criticité de $CI_A(R) = 0.28$. La perte de l'une de ces trois ressources empêche le fonctionnement du système. Ces résultats sont confirmés sur les opérations.

Analyse des configurations

Tout comme pour le convoyeur. Les deux configurations définies à la section 3.5.2 pour le système électronique sont analysées et comparées.

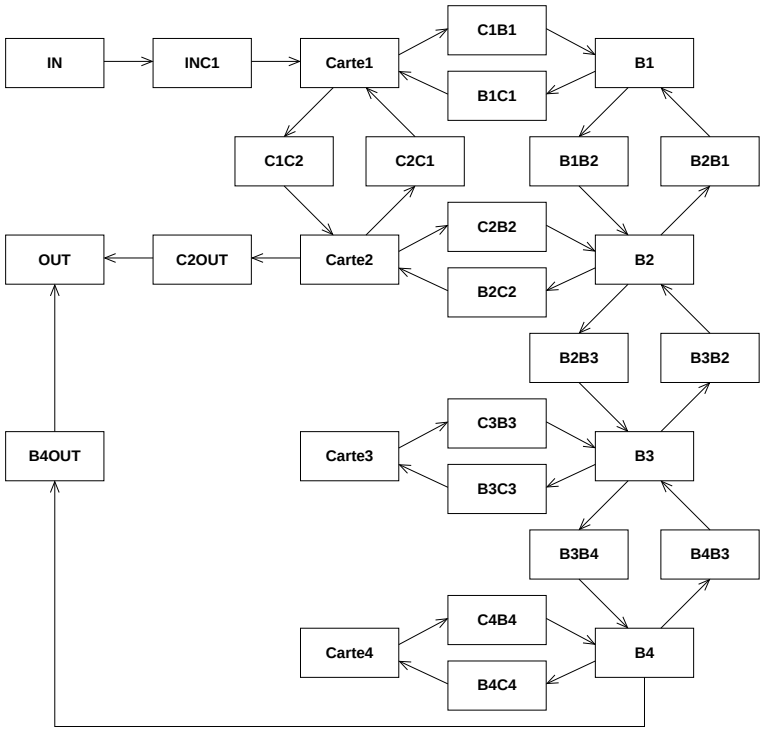


FIG. 4.15 – GAO pour l'architecture du système de traitement d'images

Analyse structurelle Les résultats de l'analyse structurelle des configurations de l'exemple électronique sont donnés au tableau 4.7.

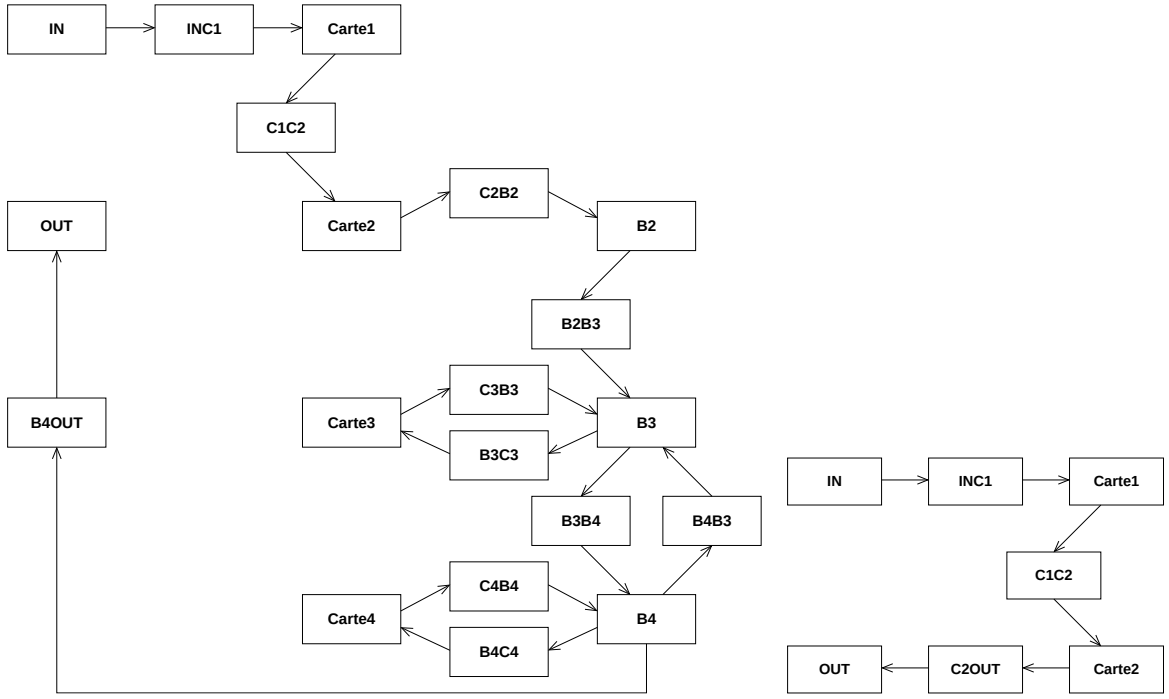
Métrique	$C = \mathcal{A}$	$C =$ Eon1	$C =$ Eon2
$N_{C,St}$	10	9	4
$N_{C,Tr}$	4	3	3
$N_{C,Cx}$	19	11	3
TP_C	0.4	0.33	0.75
CS_C	1.9	1.22	0.75
$PP_C(choix)$	3	2	2
$PP_C(annot)$	2	1	-
$\delta_C(choix)$	-	3	3
$\delta_C(annot)$	-	13	-

TAB. 4.7 – Résultats de l'analyse structurelle des deux configurations du système de traitement d'images

La configuration Eon1 utilise une grande partie des éléments de l'architecture, contrairement à la configuration Eon2 qui se contente des deux premières cartes. Le nombre de connexions par ressource stationnaire de cette dernière configuration est très faible, il correspond à la ligne de transfert équivalente.

Analyse de tolérance Les GAO obtenus pour les deux configurations en vue d'une reconfiguration mineure sont présentés aux figures 4.16.

Alors que les éléments de la configuration Eon2 sont hautement critiques par rapport à la séquence de fonctions qu'elle réalise, les résultats de la configuration Eon1 sont meilleurs. Les opérations de traitement sont toutes critiques mais uniquement par rapport à l'une des deux



(a) GAO pour la configuration Eon1 (b) GAO pour la configuration Eon2

FIG. 4.16 – GAO pour les configurations du système de traitement d'images en vue d'une reconfiguration mineure

séquences de fonctions (chaque opération n'est utilisée que pour la réalisation d'une séquence). 72% des opérations de transfert sont critiques. Au final 91% des ressources sont critiques.

Pour les deux configurations prises en compte, s'il n'est pas possible d'étendre le contexte à d'autres séquences de fonction, il est en revanche tout à fait envisageable d'effectuer une reconfiguration significative de la partie physique du système. Dans le cas de la configuration Eon2, la prise en compte des opérations qui peuvent être réalisées par les ressources en service diminue artificiellement l'indice de criticité des opérations, le nombre d'opérations critiques ne diminue pas (il y en a neuf) mais une opération de transfert non critique a été ajoutée (le transfert de la carte 2 à la carte 1). Ce résultat montre qu'il est nécessaire de toujours regarder le nombre d'opérations critiques lors de l'extension du contexte.

Pour la configuration Eon1 en revanche, la prise en compte des reconfigurations significatives améliore les résultats d'analyse de criticité. Le nombre d'opérations critiques passe de 19 sur 23 à 14 sur 28. Si on complète le bus en ajoutant la ressource B1 les résultats passent à 11 opérations critiques sur 33.

Les résultats de tolérance permettent donc de confirmer que la configuration Eon1 est beaucoup plus tolérante que la configuration Eon2.

4.5.4 Bilan sur les exemples

L'application des analyses sur les différents exemples présentés (architectures usuelles, exemple de SAP et exemple de système électronique) montre comment celles-ci peuvent-être utilisées en phase de conception, à la fois sur des architecture que des configurations.

L'application des analyses sur des configurations a montré qu'il était possible de mettre en évidence les différences entre deux configurations. Les résultats mettent clairement en avant l'intérêt de la notion de contexte, rapportés aux niveaux de reconfiguration lors de l'analyse de

criticité.

4.6 Conclusion

Ce chapitre a introduit un ensemble d'outils pour l'analyse des systèmes reconfigurables. Ces outils portent à la fois sur l'architecture, sur laquelle ils permettent d'évaluer les capacités de reconfiguration et sur les configurations pour lesquelles ils ont été présentés comme des outils d'aide à la conception (hors-ligne) et au choix ou à la construction (en ligne).

Un cadre d'analyse utilisant les outils de transformation de modèles et permettant d'obtenir automatiquement les résultats a été introduit. Les différentes analyses présentées ont été implémentées dans ce cadre.

Finalement un ensemble d'exemples d'analyses, tant sur des architectures que des configurations a été construit pour illustrer l'utilisation des différents critères définis.

La partie suivante traite de la mise en œuvre d'un système reconfigurable. Les travaux présentés dans ce chapitre seront alors utilisés pour assurer la gestion des différentes configurations.

Troisième partie

Exploitation d’un système reconfigurable

Chapitre 5

Fonctionnement du système reconfigurable

Tout système reconfigurable est conçu pour remplir un ensemble de missions qui lui sont confiées. La réalisation de ces missions constitue son exploitation.

Le langage DeSyRe, défini dans le chapitre 3 permet la description des systèmes reconfigurables et la représentation, à un haut-niveau, du concept de configuration, indépendamment du domaine considéré. Dans ce chapitre, un positionnement du modèle d'une configuration par rapport à la représentation des architectures de contrôle/commande de référence dans le domaine des systèmes de productions est proposé.

Une configuration du système est une organisation temporaire qui prend fin soit parce que les objectifs fixés sont atteints, soit parce que les contraintes de QdS appliquées au système ne sont plus respectées. Un processus de reconfiguration se met alors en place. Celui-ci conduit à la mise en place d'une nouvelle configuration permettant soit de répondre à de nouveaux objectifs, si les objectifs précédents ont été remplis ou ne peuvent plus être remplis, soit d'assurer une continuité du service avec une nouvelle organisation du système. Le choix d'une configuration repose alors en partie sur les analyses présentées au chapitre précédent.

Le fonctionnement du système dans une configuration donnée requiert la génération d'un programme de contrôle/commande qui doit être déployé sur le système. Les étapes du processus de génération de code permettant d'aboutir à un système fonctionnel ainsi que la place du langage DeSyRe dans ce processus sont donc étudiés dans ce chapitre.

Trois sections composent donc ce chapitre. La première présente les fonctions du contrôle/commande, organisées autour du concept de configuration. La seconde section traite du processus de reconfiguration. La dernière section présente un flot d'implantation. Ce flot peut non seulement permettre la mise en place l'architecture de commande, mais aussi celle du code associé à une configuration.

5.1 Description des fonctions du contrôle/commande pour un système reconfigurable

La gestion d'un système de production est généralement effectuée par le biais d'une architecture de contrôle/commande. Cette architecture a pour vocation d'identifier et de coordonner les différentes activités utiles à la mise en œuvre des fonctionnalités attendues sur le système.

La figure 5.1 présente une évolution de l'architecture présentée dans [Berruet et al., 2002] issue des travaux du LAGIS. Notre proposition place la configuration au centre du fonctionnement du système. La fonction maintenance n'a pas été reportée sur ce schéma pour se concentrer sur le fonctionnement du système.

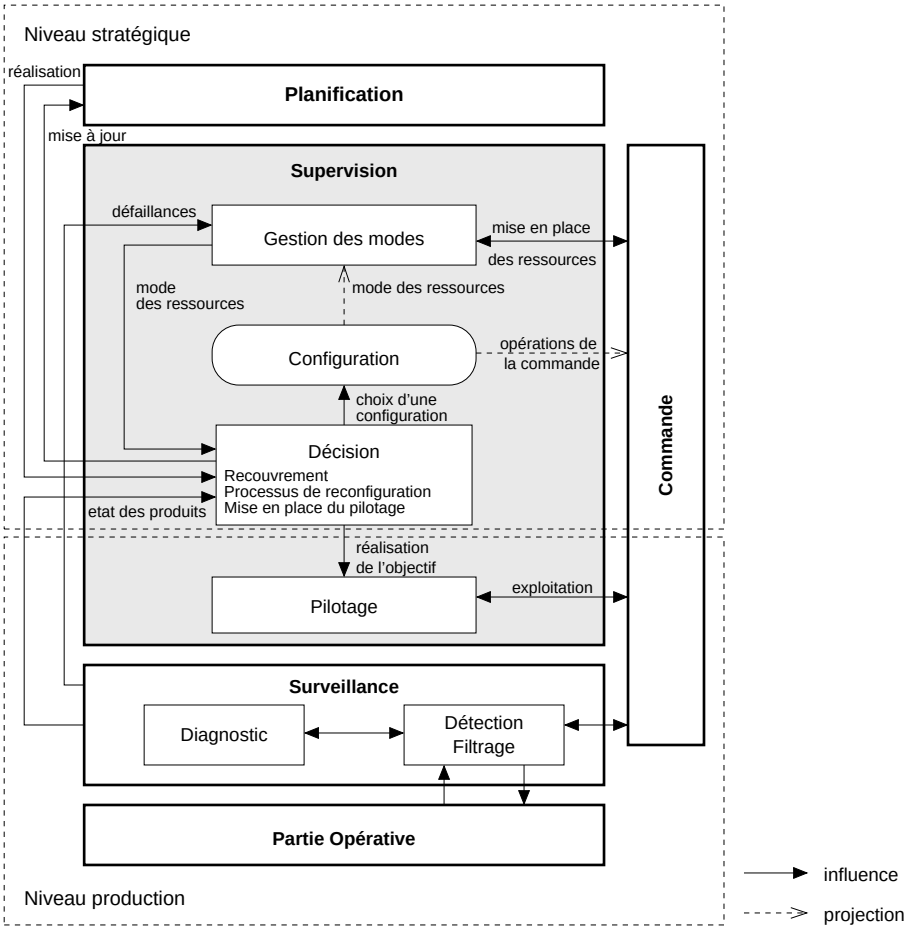


FIG. 5.1 – Proposition d'architecture de contrôle/commande pour un système reconfigurable

Parmi les différentes fonctions qui entrent en jeu, on distingue tout d'abord la *commande*, qui constitue l'interface avec les ressources. Le module de *pilotage* détermine la commande à appliquer pour réaliser les objectifs fixés par la *décision* en suivant la *planification*. Le module de *gestion des modes* s'occupe de l'utilisation des ressources. Le module de *surveillance* quant à lui, tient à jour l'état des produits dans le système et informe d'une défaillance.

Deux niveaux de fonctionnement peuvent alors être distingués pour le système. Le niveau production réalise les objectifs fixés par le niveau stratégique. Dans le cas d'un système de production, la réalisation des produits est à la charge du niveau production. C'est le niveau stratégique qui décide quel produit est réalisé et comment il est réalisé.

La suite de cette section décrit différents modules et objets du contrôle/commande que sont la commande, le pilotage, la configuration, la gestion des modes, la planification, la décision et la surveillance.

5.1.1 Commande et Pilotage

La *commande* réalise l'interface avec le procédé. Elle expose au pilotage les différentes opérations qui peuvent être réalisées sur le système.

Dans un système reconfigurable, la commande est relative à la configuration courante. En effet, les opérations qui peuvent être réalisées sur le système sont définies pour cette configuration, les ordres envoyés aux ressources ne peuvent donc pas sortir du cadre défini par cette configuration.

Le *pilotage* retourne un ordonnancement des opérations proposées par la configuration. Cet ordonnancement peut être plus ou moins complet. Il peut ne fixer que les dates de lancement des séquences d'opérations. Il peut aussi contrôler plus étroitement l'exécution de chaque opération élémentaire. Le niveau d'intervention du pilotage sur la commande dépend de la nature et de la conception du système.

La configuration suivant sa composition, peut aussi offrir au pilotage des degrés de liberté plus ou moins importants. Une configuration dans laquelle est définie une seule séquence d'opérations offre très peu de degrés de liberté au pilotage. En revanche, si la configuration définit plusieurs alternatives et si les séquences de transfert ne sont pas explicitées. Le rôle décisionnel du pilotage s'en trouve augmenté.

5.1.2 La configuration

La configuration tient une place importante dans le fonctionnement du système. Elle détermine les moyens mis en œuvre pour réaliser un objectif ainsi que les opérations réalisées par la commande.

Le rapprochement entre le pilotage et les configurations conduit naturellement à l'établissement d'une typologie des différentes configurations, permettant d'identifier leurs caractéristiques concernant le type de pilotage à leur associer.

Les configurations peuvent d'abord être classées suivant le nombre de séquences d'opérations différentes qui peuvent être mises en œuvre sur le système. Ces séquences d'opérations peuvent implémenter une seule séquence de fonctions ou plusieurs. Les configurations exposant une seule séquence d'opération sont appelées *configurations minimales* dès lors que seules les opérations nécessaires à la réalisation de cette séquence sont utilisées. Par opposition, une *configuration maximale* met en œuvre toutes les séquences de fonctions définies dans l'architecture et utilise toutes les ressources du système.

Même si une configuration n'est pas minimale, les différentes séquences qu'elle propose peuvent être totalement indépendantes, on parle alors de *configuration partitionnée*, quand les différentes séquences d'opérations d'une même configuration sont réalisées sur des ressources différentes.

Lorsqu'une configuration n'est pas partitionnée, il peut devenir nécessaire d'arbitrer l'accès aux ressources partagées. Ce besoin peut aussi apparaître lorsqu'une même séquence utilise plusieurs fois la même ressource et que plusieurs d'entre-elles sont mises en œuvre simultanément. Si aucun arbitrage n'est utilisé, l'utilisation d'une configuration peut conduire au blocage, c'est pourquoi la notion de configuration immunisée contre les blocages est introduite sous le nom de *configuration n-deadlock-safe* où *n* représente le nombre de produits pouvant être placés en concurrence avant qu'un blocage ne devienne possible.

5.1.3 La gestion des modes

Un *mode* est un ensemble d'états caractérisant une ressource ou un ensemble de ressources selon un point de vue [Dangoumau, 2000]. Les modes des ressource peuvent être organisés en *familles de modes*, permettant de hiérarchiser les différents points de vue. Les modes et les familles de modes sont définis pour un système considéré et peuvent faire l'objet d'une conception particulière [Hamani, 2005]. La figure 5.2 décrit les modes couramment utilisés pour la famille de modes "Production" d'un SAP.

L'évolution des modes des différentes ressources nécessite des mécanismes garantissant une cohérence dans cette évolution. Ce mécanisme est appelé *gestion des modes* [Bois, 1991].

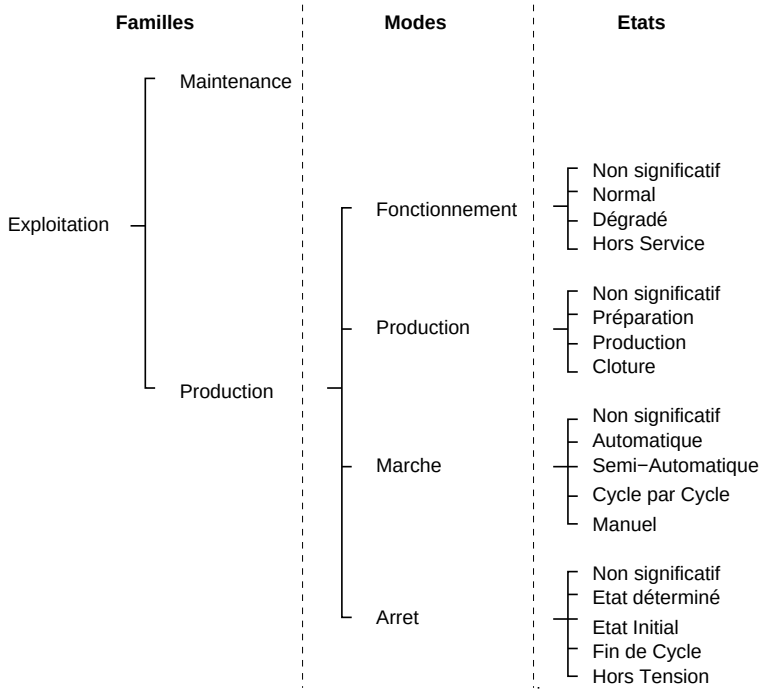


FIG. 5.2 – Famille de mode “Production” d’un SAP [Dangoumau, 2000]

Le mode des ressources est amené à changer au cours d’une reconfiguration. Le coût du changement de ces modes est alors évalué et participe au choix de la nouvelle configuration. Le module de gestion des modes dispose des informations nécessaires à cette évaluation.

De même, le changement de mode des ressources pendant le fonctionnement du système suite à une défaillance ou à une préemption de cette ressource nécessite une remise en cause de la configuration.

5.1.4 La planification

Lorsque des objectifs vastes sont donnés à un système, il s’avère souvent nécessaire de décomposer ces objectifs pour étaler leur réalisation dans le temps. Une *planification* est alors réalisée pour atteindre progressivement les objectifs fixés.

La planification introduit donc un deuxième niveau d’ordonnancement (différent du pilotage car sur un horizon à long terme), les différents objectifs considérés pouvant nécessiter une reconfiguration du système. Cet ordonnancement conduit à planifier des reconfigurations qui interviennent dans le cadre du fonctionnement normal du système. Dans ce cas, la reconfiguration n’est pas considérée comme un traitement d’exception par un opérateur humain. Elle est néanmoins perçue comme une action modifiant le procédé.

5.1.5 La décision

Le caractère incertain de l’environnement dans lequel évolue un système reconfigurable est tel qu’un certain nombre de choix concernant le fonctionnement du système doivent être pris en ligne. Ces choix concernent plus particulièrement la réaction face à une défaillance. Il faut alors décider des mesures à prendre et les mettre en œuvre, soit en agissant par l’intermédiaire du pilotage uniquement (sans reconfiguration), soit en reconfigurant le système. Le traitement d’une défaillance peut remettre en cause la planification.

Le module de *décision* doit faire face à ces choix. Ce module fait partie intégrante de la supervision du système. C'est lui qui dirige le processus de reconfiguration décrit dans la section suivante.

Les décisions sont prises en se basant sur des modèles rendant compte de l'état du système (état des produits et mode des ressources). Suivant les objectifs et contraintes, la planification initiale peut être remise en cause si les conditions l'exigent.

5.1.6 La surveillance du système

Un module de *surveillance* du système à un rôle informationnel et est nécessaire pour suivre le fonctionnement du système et réagir dès l'apparition d'une défaillance.

Parmi les rôles de la surveillance, le suivi des produits permet de reconstituer l'*état des produits dans le système* qui constitue, avec le mode des ressources l'état du système. L'état de chaque produit est constitué de plusieurs informations, la première est l'avancement dans la séquence de fonctions qui lui est appliquée à l'instant courant. La seconde information est sa localisation dans le système. Suivant les méthodes et les modèles utilisés, l'état des produits est représenté de manière plus ou moins fidèle.

Le deuxième rôle de la surveillance consiste à déterminer si le fonctionnement du système ne correspond pas au fonctionnement souhaité. La détection doit identifier le non respect des contraintes de QdS fixées pour la configuration courante. Ce non respect peut être entraîné par la défaillance d'un élément du système ou une modification de l'environnement. Il se traduit par une violation des exigences rapportée par des indicateurs.

Différents indicateurs peuvent être utilisés pour la détection. Ces indicateurs dépendent de la grandeur mesurée ainsi que de la manière de mettre en place le système. Placés sur le procédés, ils prennent généralement la forme de capteurs, couplés à des traitements. Mais ces indicateurs peuvent aussi être lus sur un modèle, reconstitué à partir d'autres indicateurs placés à un niveau inférieur.

Pour une analyse de qualité, il s'agit d'effectuer des mesures sur le produit pour déterminer si les contraintes de qualité sont respectées. Ces mesures peuvent être réalisées aussi bien sur le produit fini que sur un produit semi-fini. Un produit ne respectant pas les critères établis peut être rejeté ou déclassé, les contraintes de QdS peuvent alors prendre la forme d'un taux de rejet ou d'un taux de déclassement.

Une mesure de débit ou de vitesse est souvent l'indicateur permettant de qualifier la productivité d'un système.

Les mesures de QdS sont envoyées au module de décision qui utilise ces informations pour lancer ou non des procédures de recouvrement pouvant prendre la forme d'une reconfiguration.

En ce qui concerne la sûreté de fonctionnement, les défaillances qui apparaissent au niveau des ressources ont généralement des conséquences à la fois sur la qualité et sur la performance du système. La détection de défaillances passe par la surveillance du procédé au niveau événementiel (par l'intermédiaire des indicateurs).

Une fois qu'un comportement anormal est détecté, si la sécurité ou l'environnement sont menacés, un traitement d'urgence est mis en œuvre [Berruet et al., 2003] pour sécuriser le système avant de lancer un diagnostic dont le résultat est pris en compte par la décision pour déterminer si une reconfiguration est nécessaire.

5.1.7 Bilan sur le fonctionnement du système

Les différentes fonctions permettant de mettre en œuvre un système reconfigurable ont été présentées. Parmi ces fonctions, une partie participe au fonctionnement en production du système, l'autre partie traite la stratégie et met en place des configurations permettant d'atteindre les objectifs fixés.

La configuration tient une place importante dans le fonctionnement du système. Elle doit permettre de répondre à un objectif fixé par la planification. Son choix impacte sur les services que fournit la commande au pilotage, il détermine aussi le mode des différentes ressources qui sont utilisées.

La section suivante explique comment sont utilisées les fonctions de la supervision ainsi que la planification dans le cadre du processus de reconfiguration que nous proposons.

5.2 Le processus de reconfiguration

Cette section introduit les mécanismes mis en œuvre lors du changement de configuration. Cette phase du fonctionnement du système, lui permettant de réaliser un nouvel objectif ou de pallier une défaillance est appelée processus de reconfiguration. Il consiste à choisir une nouvelle configuration et à la mettre en place.

Après avoir décrit les étapes composant le processus de reconfiguration, ainsi que le rôle de la gestion des modes, les différentes étapes de son déroulement sont décrites.

5.2.1 Description générale du processus de reconfiguration

Le processus de reconfiguration est décrit à la figure 5.3. Il peut être déclenché suite à une défaillance, celle-ci est diagnostiquée par la surveillance et le résultat fourni à la décision. Le déclenchement peut aussi être souhaité par la planification, lorsque l'objectif est atteint.

Connaissant l'état du système, fourni par la surveillance et la gestion des modes, le choix de la nouvelle configuration peut débuter. Ce choix tient compte des objectifs fixés pour le système sous forme de contraintes de QdS, du mode des ressources et de l'état des produits dans le système. Le choix d'une nouvelle configuration peut nécessiter un changement de la planification. Il peut aussi arriver qu'aucune configuration ne convienne mieux que la configuration courante, le processus de reconfiguration est alors arrêté et le système continue son fonctionnement dans la configuration courante.

Si une nouvelle configuration a été choisie, elle doit être mise en place. Cette mise en place peut nécessiter le passage par des configurations intermédiaires de positionnement permettant d'accorder l'état des produits avec cette configuration.

5.2.2 Rôle de la gestion des modes

Dans le cadre de la reconfiguration, la gestion des modes joue plusieurs rôles. Elle donne tout d'abord une information sur l'utilisation des différentes ressources. Cette information permet de vérifier la cohérence entre la configuration et le mode actuel des ressources. En cas d'incohérence la gestion des modes permet d'identifier le coût de l'évolution du mode des ressources permettant de se conformer à la configuration et ainsi d'évaluer une partie du coût de la reconfiguration, ce coût est calculé par rapport aux étapes nécessaires pour placer toutes les ressources impliquées dans des modes compatibles avec la configuration choisie.

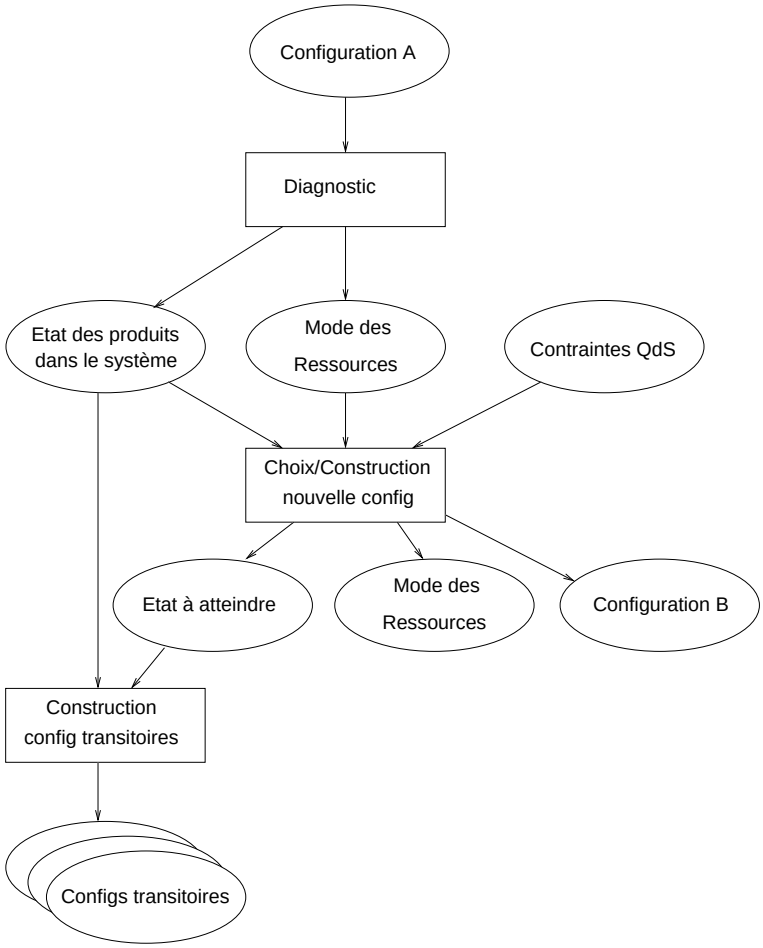


FIG. 5.3 – Description du processus de reconfiguration

La gestion des modes a ensuite la charge de faire respecter les contraintes associées aux modes des ressources et de leurs opérations pendant la reconfiguration. Cette charge correspond au rôle premier de la gestion des modes tel qu'il est présenté dans [Bois, 1991].

5.2.3 Choix/Construction d'une configuration

Le choix d'une nouvelle configuration constitue l'étape centrale du processus de reconfiguration. Cette nouvelle configuration peut avoir été déterminée préalablement, le choix est alors réalisé parmi un ensemble de configurations. La nouvelle configuration peut aussi être construite pour répondre aux besoins du système.

Le choix d'une configuration relève d'une décision multi-critères. Les critères pris en compte sont le coût de fonctionnement de la configuration, son coût de mise en place, ses performances. Ce choix peut être fait en ligne ou hors ligne, un tel choix repose sur l'analyse des configurations présentée au chapitre 4. Une analyse hors-ligne permet de classer les différentes configurations pour répondre à diverses utilisations pressenties du système. Une analyse en-ligne peut la compléter en prenant en considération le coût de reconfiguration qui dépend de l'état courant du système.

La construction automatique d'une configuration nécessite d'utiliser des techniques avancées. De plus ces techniques sont très dépendantes de l'application. Plusieurs méthodes plus ou moins automatisables peuvent être envisagées. Une première méthode complètement automatisable consiste à générer les séquences d'opérations pour réaliser les séquences de fonctions

visées. Une méthode par construction est aussi possible, il s'agit de composer différentes configurations en utilisant des configurations minimales préalablement définies, cette technique met à la fois en œuvre un choix et une construction de la configuration. Une troisième méthode consiste à aménager la configuration courante.

Le choix et la construction de la configuration peuvent aussi être utilisés conjointement. On peut alors utiliser une bibliothèque de configurations, dans laquelle une nouvelle configuration est choisie. Comme tous les cas de pannes ne peuvent pas nécessairement être identifiés avant le fonctionnement, si aucune configuration n'est compatible avec l'état du système, la construction d'une nouvelle configuration peut être envisagée. Cette configuration vient alors enrichir la bibliothèque et permettra une réaction plus rapide face à un nouvel aléa similaire.

Le choix ou la construction d'une configuration peut être validé par l'utilisation de la simulation [Berruet and Kindler, 2006]. Le fonctionnement du système dans sa nouvelle configuration est alors éprouvé avant d'entrer en fonctionnement réel.

5.2.4 Application de la configuration choisie

Une fois qu'une nouvelle configuration est choisie pour le système, celle-ci doit être mise en place. La gestion des modes est chargée de placer les ressources dans un mode compatible avec la configuration choisie. Mais le mode des ressources seul ne constitue pas tout l'état du système, celui-ci comprend aussi l'état des différents produits dans le système.

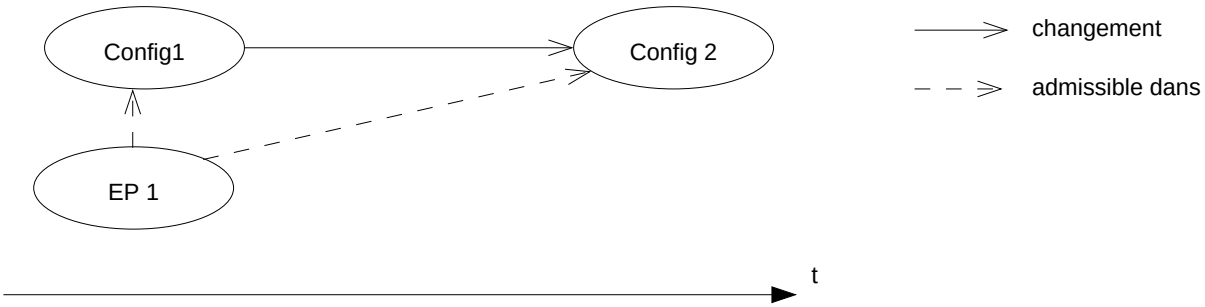
Avant de mettre en œuvre une nouvelle configuration, il faut en effet s'assurer que celle-ci permette au système de gérer les produits à partir de l'état où ils se trouvent au moment de la reconfiguration. Introduisons la notion d'admissibilité d'un état des produits dans une configuration par la définition 5.

Définition 5 *Un état des produits est admissible dans une configuration si cette dernière permet de faire évoluer correctement cet état dans le cadre de son fonctionnement.*

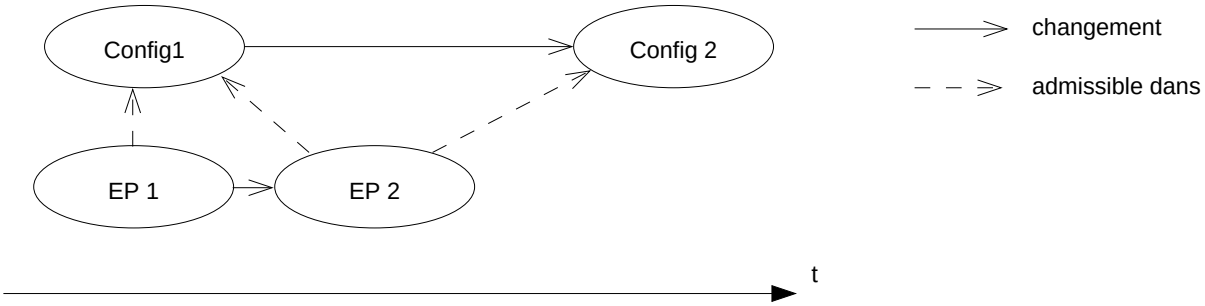
Le passage d'une configuration à l'autre par l'intermédiaire de la gestion des modes doit se faire en respectant cette notion d'admissibilité. Il est donc nécessaire, pour changer de configuration, que l'état courant des produits soit admissible à la fois dans la configuration source et dans la configuration destination. Si aucun état des produits accessible depuis la configuration source ne correspond à un état admissible de la configuration destination, alors il faut passer par une configuration transitoire qui partage un état admissible avec la configuration source et un état admissible avec la configuration destination.

La figure 5.4 illustre trois cas qui peuvent se présenter lors de la transition. Sur la figure 5.4(a), l'état des produits initial (EP 1) est admissible dans les deux configurations (config 1 et config 2), la transition se fait alors sans configuration intermédiaire. Dans la figure 5.4(b), il existe un état des produits commun entre les deux configurations, on choisit d'atteindre cet état par l'intermédiaire du pilotage. Tout en restant dans la première configuration, on fait évoluer le système pour que l'état des produits soit admissible dans les deux configurations, puis on change de configuration. Sur la figure 5.4(c) l'adoption d'une configuration de transition est indispensable. Le système évolue alors vers un état des produits admissible dans cette configuration (config T), il adopte cette configuration intermédiaire pour pouvoir atteindre un état des produits admissible dans la configuration cible pour pouvoir adopter cette configuration.

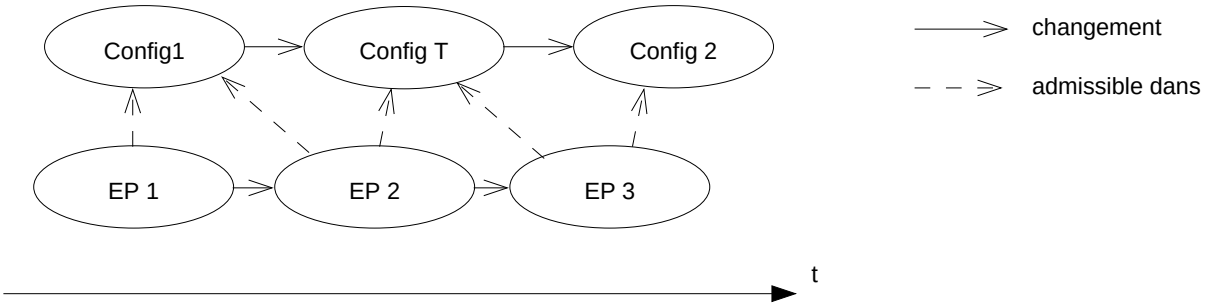
Le coût de reconfiguration doit prendre en compte le passage éventuel par une configuration de transition. Cette configuration intermédiaire est choisie en fonction de l'état courant des produits, décrit par un modèle afin d'être intégré au cadre d'analyse.



(a) L'état des produits initial est admissible dans les deux configurations



(b) Il existe un état des produits commun entre les deux configurations



(c) Passage par une configuration de transition

FIG. 5.4 – Les différentes transitions possibles entre deux configurations suivant l'admissibilité de l'état des produits dans la configuration destination

Construire une configuration de transition est généralement difficile. Un cas particulier consiste à démarrer la nouvelle configuration sans aucun produit, la configuration de transition sert alors à vider le système de tous les produits en cours de réalisation.

5.2.5 Bilan sur le processus de reconfiguration

Le processus de reconfiguration donne tout son sens à l'utilisation d'un système reconfigurable. Dans notre approche, la reconfiguration ne constitue qu'une étape de transition dans la vie du système, permettant au système de remplir au mieux les missions qui lui sont confiées.

Le processus de reconfiguration que nous proposons est composé de deux étapes principales. La première consiste à choisir une configuration, ou à défaut, à la construire. La seconde est consacrée à la mise en place de cette configuration en faisant éventuellement intervenir le pilotage et la gestion des modes afin de mettre le système et les produits dans un état compatible avec elle.

La section suivante traite de la génération du code de contrôle/commande pour l'architecture et les configurations d'un système reconfigurable.

5.3 Implantation du système reconfigurable

Le fonctionnement du système reconfigurable nécessite l'implantation de l'architecture de commande lui permettant de fonctionner, ainsi que de la commande elle-même, exécutée sur le système.

5.3.1 Besoins

L'implantation de la commande d'un système reconfigurable, doit permettre de réaliser les opérations et les séquences d'opérations, décrites dans la configuration, sur le procédé.

Le procédé est la partie du système à contrôler. Sa description transparaît de manière grossière dans l'architecture physique par une description des différentes ressources ainsi que des liens entre elles.

Le fonctionnement du système peut requérir la mise en place d'une architecture de commande. Cette architecture est déployée sur les divers équipements constituant l'électronique de commande et permet son exploitation. Elle fournit les différents modules présentés au chapitre 5 (commande, pilotage, gestion des modes, surveillance).

Une étape d'implantation est aussi nécessaire pour projeter la configuration sur la commande. Cette projection peut se faire par la génération d'un code spécifique chargé sur les cibles composant l'électronique de commande. Elle peut aussi consister à générer des données utilisées par la commande pour réaliser les séquences d'opérations.

Il faut donc proposer un cadre permettant aussi bien la mise en place d'une architecture de commande adaptée au système que la projection des séquences d'opérations définies dans la configuration sur l'architecture de commande préalablement obtenue. L'implantation de mécanismes permettant la reconfiguration est particulièrement importante dans le cadre des systèmes reconfigurables.

5.3.2 Présentation de l'activité d'implantation

D'une manière générale, l'activité d'implantation est décomposée en deux étapes représentées sur la figure 5.5.

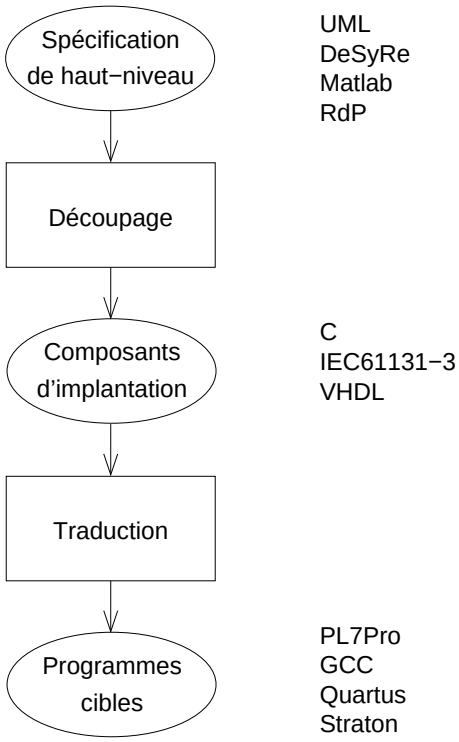


FIG. 5.5 – Flot d'implantation d'un système reconfigurable

L'implantation du système est généralement réalisée à partir d'une spécification de celui-ci. Cette spécification décrit le fonctionnement attendu du système et les moyens mis en œuvre. Cette spécification peut être écrite dans un langage tel qu'UML [OMG, 2003a], avec des réseaux de Petri [Valette, 2002] ou encore en utilisant Matlab [Mathworks, 2006]. Le langage DeSyRe, défini dans cette thèse constitue aussi un langage de spécification, nous l'utiliserons comme point d'entrée pour les activités d'implantation dans le cadre de l'application présentée au chapitre 6.

A partir de la spécification du système, une étape de découpage permet d'identifier des composants qui constitueront le code implanté, ces composants sont choisis dans une bibliothèque métier. Les langages utilisés dans ces composants correspondent aux langages utilisés sur les nœuds de l'architecture électronique de commande sur lesquels ils sont implantés. Il s'agit donc de langages de la norme IEC61131-3 [IEC, 2003], du C [Kernighan and Ritchie, 1990] ou du VHDL [IEEE/DASC/VASG, 2002]. Dans cette étape, les normes sont respectées afin de garantir une portabilité maximale, idéalement indépendante de la cible.

Finalement, une adaptation à la cible des différents codes est apportée pour supporter les contraintes spécifiques aux outils utilisés, les formats d'échange, les fonctionnalités non supportées ou encore les optimisations. Ainsi, si le code est écrit dans le langage SFC de la norme IEC61131-3, son utilisation dans un automate Schneider-Télémécanique de la famille TSX nécessite une traduction vers le langage G7 utilisé par le langage PL7-Pro [Telemecanique, 2003b].

5.3.3 La place du modèle de description

Le langage DeSyRe, défini pour la description des systèmes reconfigurables a été principalement employé pour réaliser des analyses sur ces systèmes. Ce langage a été défini et utilisé dans cette thèse dans le cadre de l'analyse des systèmes reconfigurables.

Ce cadre d'analyse peut être étendu à l'implantation. Une telle extension permet d'utiliser

des analyses menées sur le système pour identifier les composants logiciels. L'intérêt du langage de description de haut niveau pour aboutir à un code de contrôle commande du système est multiple. L'utilisation d'un même modèle pour les différentes activités de la conception permet de garantir une cohérence entre les résultats des analyses effectuées alors que le système n'existait pas encore et les résultats obtenus lors de son fonctionnement. En cas d'erreur dans la modélisation du système, et dans le pire des scénarios, cette erreur sera décelée lors de l'implantation et remettra en cause les analyses effectuées en amont.

Le modèle utilisé lors de l'analyse nécessite néanmoins un enrichissement préalablement à la génération de commande. En effet, certains aspects du système sont ignorés lors des analyses qui portent principalement sur les caractéristiques dépendantes du procédé. Les notions de contrôleurs sur lesquels le code est exécuté ainsi que les problèmes de partitionnement, de communication et de synchronisation n'ont pas été pris en compte et doivent être abordés pendant l'implantation du système.

La proposition d'extension du cadre d'analyse pour l'implantation consiste donc à considérer un découpage de la description du système reconfigurable en différents composants. Cette extension est présentée à la figure 5.6

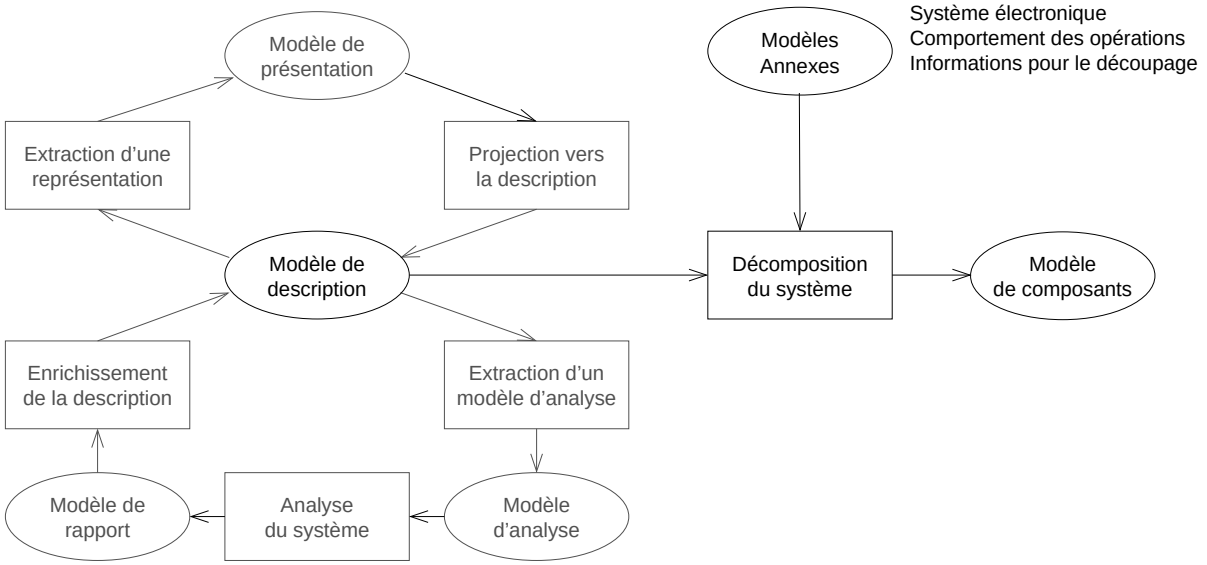


FIG. 5.6 – Extension du cadre d'analyse pour l'implantation

5.3.4 Découpage en composants pour l'implantation

La décomposition du système en architecture et configuration est un des principes de base du langage DeSyRe. Ce découpage offre deux choix lors de l'implantation. Le premier entraîne l'implémentation du système complet, composé à la fois de l'architecture et de sa configuration. Ce choix conduit à une nouvelle implémentation du code complet pour chaque configuration. Le deuxième choix conduit à la génération d'un code de contrôle commande pour l'architecture, disposant de mécanismes permettant la reconfiguration.

Le découpage du système peut ensuite être réalisé suivant différents critères, dépendant de l'application. Dans le cas du démonstrateur ferroviaire présenté au chapitre 6, le modèle de l'application a d'abord été découpé suivant l'axe physique. Un découpage suivant l'axe logique a aussi été proposé.

Un *découpage physique* est réalisé par rapport aux ressources et leurs opérations. Des composants d'implantation sont alors utilisés en respectant le découpage réalisé. Ces composants

sont chargés de réaliser les opérations élémentaires réalisées sur les ressources. Dans le cas du convoyeur décrit à la section 3.5.1, un tel découpage permet d'obtenir un code de contrôle/commande associé à chaque ressource et décrivant comment réaliser les produits. Les ressources collaborent alors pour aboutir à cette réalisation. Ce découpage sera appliqué à la section 6.3.4 pour la génération du code de contrôle/commande associé à l'architecture du circuit.

Un *découpage suivant l'aspect logique* peut aussi être réalisé. Un composant d'implantation est alors associé à chaque produit. Celui-ci prend en charge sa réalisation sur le système en pilotant les différentes ressources. Appliqué à l'exemple du convoyeur décrit à la section 3.5.1, un tel découpage conduirait à l'exécution d'un programme pour réaliser chaque produit individuellement, négociant l'utilisation des ressources et les pilotant. C'est le découpage utilisé pour l'exemple présenté au chapitre suivant, à la section 6.3.3 ou un grafcet est généré pour chaque séquence d'opérations de la configuration.

Une conjonction des deux découpages est possible. Un découpage en composants de la partie physique du système fournit alors une plate-forme d'exécution pour le code généré pour les produits qui n'a alors plus à se charger de la commande des ressources.

5.3.5 Modèles annexes pour l'implantation

Le modèle de description d'un système reconfigurable seul ne permet pas d'aboutir automatiquement à un système utilisable. Le langage DeSyRe ne décrit le système qu'en termes de ressources et de connexions. L'adjonction d'informations est donc nécessaire pour permettre l'implantation.

Tout d'abord, l'implantation nécessite des informations liées à l'électronique de commande. Ces informations décrivent l'architecture de ce système électronique ainsi que ses liens avec le procédé. Les différents nœuds composant l'électronique de commande doivent donc être décrits ainsi que les moyens de communication. Les entrées/sorties doivent aussi être liées aux opérations réalisées sur les ressources. Cette description de l'électronique de commande peut être réalisée à l'aide du langage DeSyRe, on aboutit alors à un système reconfigurable dont l'électronique de commande est elle-même reconfigurable. Pour l'application du train, présentée au chapitre 6 le lien est fait entre les entrées/sorties de l'automate et les éléments du modèles.

Il est aussi nécessaire de décrire le fonctionnement des ressources ou des opérations réalisées sur ces ressources. Cette description sert de base à la génération de commande. Dans le cadre de nos expérimentations sur le train, cette description a été intégrée directement dans les règles de génération de code (règles qui constituent en elle-même des modèles). Dans le cadre de l'approche par composants proposée dans [Mouchard, 2002], le fonctionnement des ressources est décrit dans une bibliothèque de composants sur étagère.

5.3.6 Bilan sur l'implantation

Le flot d'implantation présenté répond à deux besoins. Le premier consiste à réaliser l'architecture de commande du système, ce code est généralement obtenu à partir de l'architecture physique du système. Le deuxième besoin consiste à obtenir la commande du système à partir de la configuration.

Le cadre d'implantation présenté dans cette section a été utilisé avec succès pour obtenir le programme de commande du démonstrateur ferroviaire. Cette utilisation est présentée au chapitre 4.

La recherche et l'application de nouveaux découpages du modèle DeSyRe pour l'implantation constitue une perspective intéressante.

5.4 Conclusion

Le fonctionnement d'un système reconfigurable a été présenté dans ce chapitre. Une nouvelle architecture de contrôle/commande a été proposée, prenant en considération la configuration. Des répercutions ont été ensuite propagées sur les différentes fonctions de cette architecture.

Le processus de reconfiguration, a ensuite été développé dans le cadre de l'architecture de contrôle/commande présentée. Les étapes constituant ce processus ont été décrites, ces étapes sont le choix d'une configuration et la transition vers cette dernière.

Pour compléter ce chapitre sur la mise en œuvre des systèmes reconfigurables, l'activité d'implantation a été présentée. Celle-ci prend la forme d'un flot permettant, à partir de la spécification du système et de ses configurations, d'obtenir le code de contrôle/commande permettant l'évolution du système reconfigurable.

Les éléments présentés dans ce chapitre sont mis en œuvre sur l'exemple de système reconfigurable que constitue le démonstrateur ferroviaire présenté au chapitre suivant.

Chapitre 6

Application à l'exemple du démonstrateur ferroviaire

Les différents travaux effectués au cours de cette thèse ont été appliqués sur une plate-forme de démonstration mise en place parallèlement afin de montrer l'applicabilité de l'approche proposée. Le système étudié est un système de transport ferroviaire. La plate-forme est constituée d'un circuit ferroviaire et d'une architecture de commande distribuée. Cette plate-forme est globalement représentative des systèmes mécatroniques, constitués d'un procédé dont le fonctionnement est épaulé par une électronique de commande.

Une première section présente la plate-forme. La modélisation de celle-ci à l'aide du langage défini dans ce mémoire est ensuite réalisée dans une seconde section, les résultats de l'analyse du circuit sont alors fournis. La troisième section présente la génération d'une commande reconfigurable pour le démonstrateur selon la démarche proposée à la section 5.3. La dernière section traite enfin de son exploitation.

6.1 Présentation de la plate-forme

Le système étudié est composé de deux sous-systèmes qui sont le circuit de train d'un côté (partie opérative) et l'électronique de commande (partie commande) de ce circuit de l'autre. Ces deux sous-parties disposent de capacités de reconfiguration et peuvent être étudiées en tant que systèmes reconfigurables. Cette étude est réalisée pour le circuit dans le cadre de ce chapitre.

6.1.1 Description de la partie opérative

La partie opérative du système ferroviaire est constituée à la fois du circuit du train et des locomotives qui circulent sur ce circuit. Ces deux parties sont présentées dans les sections suivantes.

Objectifs du circuit

L'objectif du circuit est de retrouver des problématiques proches de celles des systèmes de production sur une maquette pouvant être déployée sur une table.

Le démonstrateur reproduisant le fonctionnement d'un système de production, la réalisation de plusieurs traitements est proposée en des lieux caractéristiques du circuit. Des séquences de fonctions peuvent alors être définies pour réaliser ces différents traitements.

Description du circuit

Le circuit du train est constitué de rails et d'aiguillages interconnectés. Trois types d'aiguillages sont utilisés et se distinguent par le nombre de branches qu'ils fournissent (trois ou quatre). Afin d'observer le passage de trains à certains points du circuit, des capteurs de type "barrage" sont placés et s'activent lorsqu'un train coupe le faisceau lumineux entre l'émetteur et le récepteur.

Le placement des capteurs délimite des tronçons. Chaque tronçon correspond à une portion du circuit, constituée uniquement de rails et délimitée par deux capteurs. Les capteurs permettent de savoir si un train entre ou sort d'un tronçon.

La figure 6.1 décrit la partie du circuit qui a été modélisée. Les aiguillages y sont délimités par des ellipses en pointillés, et le nom des différents éléments constituant le circuit est donné. Les noms des tronçons sont de la forme "Tx", leurs deux extrémités sont différenciées par la lettre "a" ou "b". Les aiguillages portent un nom de la forme "Ax", C1 correspond à une séparation entre deux tronçons. Le nom des différentes fonctions réalisées sur le circuit est indiqué sur les tronçons correspondants.

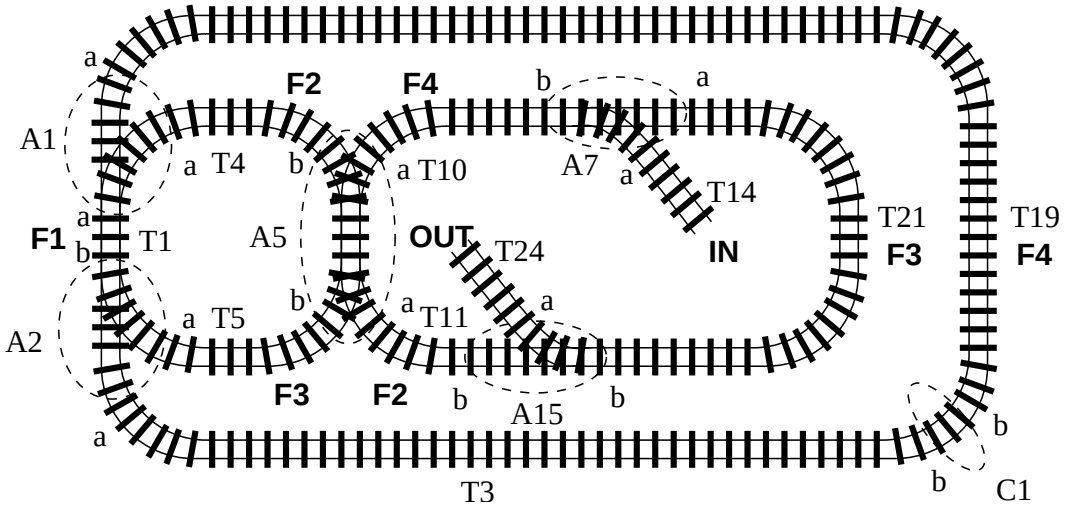


FIG. 6.1 – Description du circuit étudié

Description des locomotives

Les locomotives sont électrifiées et reçoivent leur énergie par les rails. De plus, elles sont équipées d'un système de décodage permettant d'interpréter des commandes qui leur sont envoyées par courants porteurs. Chaque locomotive dispose d'une adresse et interprète les ordres qui lui sont destinés (sens, vitesse, allumage des feux).

6.1.2 Description de l'architecture de commande

L'architecture de commande du démonstrateur ferroviaire a été conçue pour offrir des capacités de reconfiguration afin de pouvoir étudier, dans le futur, l'interaction entre le procédé reconfigurable ainsi qu'une électronique de commande reconfigurable.

L'architecture de commande complète, proposant différentes redondances est présentée dans une première section. Les éléments utilisés dans le cadre de l'étude réalisée dans ce document, sont présentés dans une deuxième section.

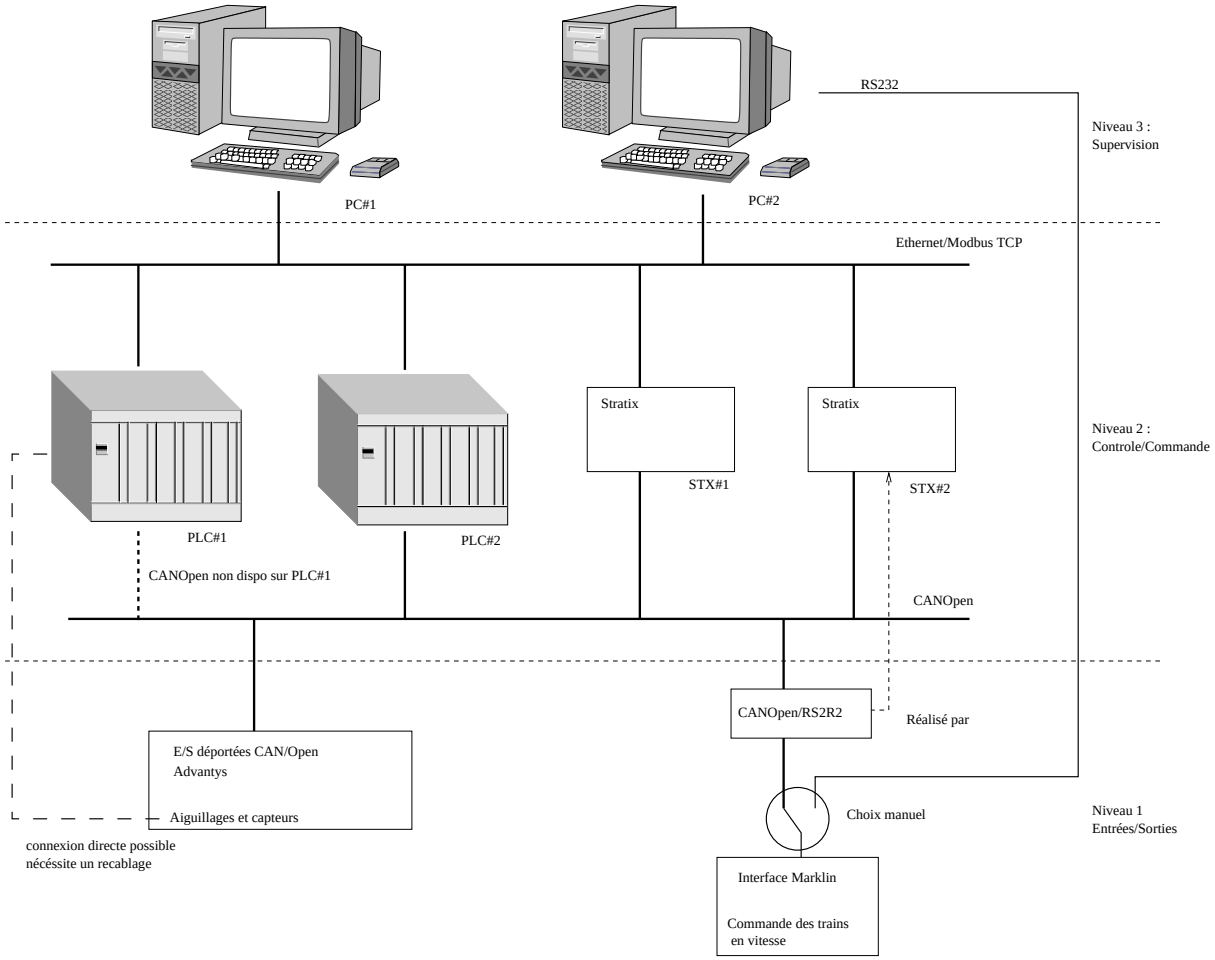


FIG. 6.2 – Architecture de commande du train

Architecture de commande complète

L'architecture de commande du train, représentée à la figure 6.2, se décompose en trois niveaux. Au niveau le plus bas, on retrouve les entrées/sorties déportées, liées aux capteurs et aux aiguillages, ainsi que le boîtier réalisant l'interface avec les trains. Le second niveau est constitué de nœuds de contrôle/commande, parmi ceux-ci se trouvent des automates programmables TSX Premium [Telemecanique, 2003a] ainsi que des cartes FPGA Stratix [Altera, 2006] qui peuvent se substituer aux automates programmables ou travailler avec eux. Le niveau le plus élevé est composé de deux PC, chargés de la supervision du système ainsi que de l'interface avec l'utilisateur.

Le lien entre les différents niveaux s'effectue à travers deux réseaux. La liaison entre les niveaux 1 et 2 est réalisée par un bus CANOpen alors que les équipements des niveaux 2 et 3 communiquent par le protocole modbus sur une liaison ethernet.

L'interface avec les trains étant propriétaire et disposant uniquement d'une liaison série RS232, un adaptateur CANOpen/RS232 est nécessaire, cet adaptateur est réalisé sur l'une des cartes Stratix, agissant alors sur le bus CAN en tant que maître et esclave, l'utilisation d'un PC est aussi possible pour piloter les trains.

Restrictions pour l'exemple

Pour l'étude du circuit, seul un PC et un automate sont utilisés. De plus, une partie du circuit ayant été câblée directement sur les entrées/sorties de l'automate (avant la mise en place du bus CANOpen), une partie des E/S est cablée sur l'automate, l'autre est connectée au bus CANOpen. De même, le pilotage des trains se fait au travers du PC sur lequel un programme est chargé de récupérer les commandes envoyées aux trains via Modbus et de les transmettre à l'interface. L'ensemble des logiciels utilisés pour mettre en œuvre l'application est présenté en annexe D.

6.2 Modélisation et analyse de la partie opérative du démonstrateur ferroviaire

La partie opérative du démonstrateur ferroviaire, présentée dans la section précédente, peut être modélisée à l'aide du langage défini à la section 3. Cette modélisation permet d'utiliser le cadre d'analyse et d'appliquer les différentes métriques qui ont été définies pour la caractérisation des systèmes reconfigurables.

6.2.1 L'architecture de la partie opérative du démonstrateur

L'architecture décrit les différents éléments constituant le circuit ainsi que les traitements qui y sont disponibles.

Architecture physique

Au niveau de la partie opérative, on peut distinguer deux types d'entités : les tronçons et les aiguillages.

Les tronçons correspondent à des lieux sur lesquels des opérations peuvent être réalisées, ils sont donc représentés comme des ressources stationnaires, deux ports permettent de distinguer les deux extrémités du tronçon.

Le passage d'un tronçon à un autre correspond au franchissement d'un aiguillage (en considérant une séparation simple entre deux tronçons comme un cas limite d'aiguillage à deux branches). Chaque transfert réalisable par l'aiguillage est modélisé en utilisant une connexion reliant les ports des ressources représentant les tronçons. Une ressource de transport représente chaque aiguillage et est reliée à chacune des connexions qu'elle peut réaliser.

L'architecture physique du circuit ainsi constituée est représentée à la figure 6.3.

Architecture logique

Le rôle des locomotives sur le circuit est proche de celui de palettes sur un convoyeur qui sont parfois assimilées à des produits composés de la palette et du produit transporté. La modélisation de nos locomotives s'inspire donc de ces modélisations. Les locomotives sont représentées par des produits qui sont déplacés sur le système et qui subissent des transformations sur les tronçons (chargement de la locomotive, déchargement, traitement sur le produit). Différents produits représentent donc la même locomotive, suivant son contenu. L'architecture logique complète, utilisée pour le démonstrateur ferroviaire est proposée sur le listing 6.1.

Quatre fonctions de traitement sont définies dans l'architecture logique. Il s'agit des fonctions F1, F2, F3 et F4. On retrouve aussi la fonction de transfert générique FT. Quatre autres

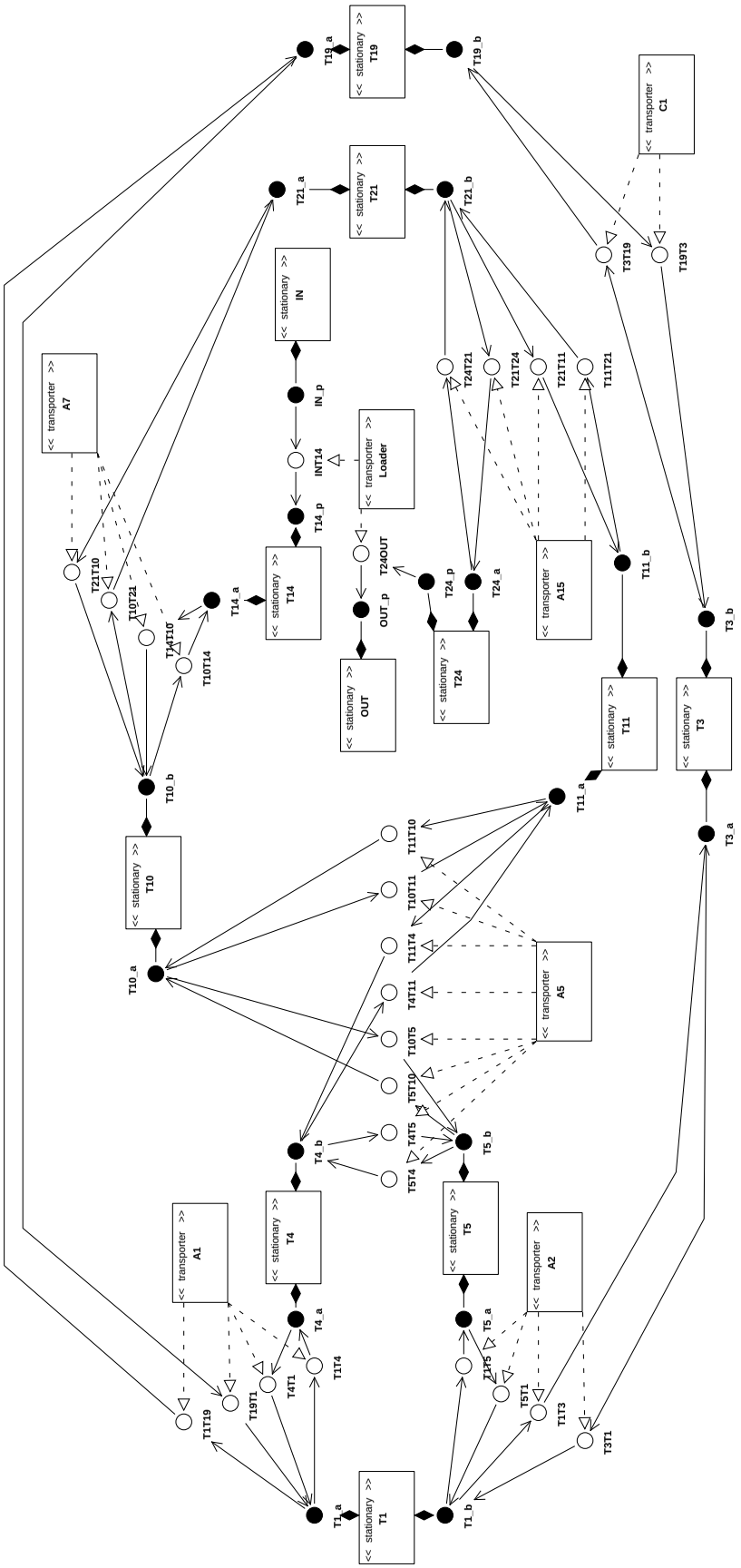


FIG. 6.3 – Architecture physique du circuit de train

Listing 6.1 – architecture logique du circuit de train

```
1  logical architecture {
2    function F1;
3    function F2;
4    function F3;
5    function F4;
6    function FT;
7    function ChargerP1;
8    function DechargerP1;
9    function MettreTrainSurReseau;
10   function SortirTrainDuReseau;
11   function ResetTrain;
12
13   product Produit1Brut;
14   product Produit1Fini;
15   product Train1Vide;
16   product Train1P1Brut;
17   product Train1P1Fini;
18
19   sequence P1_Chargement[Produit1Brut, Train1Vide->Train1P1Brut] :
        ChargerP1;
20   sequence P1_Dechargement[Train1P1Fini->Train1Vide, Produit1Fini] :
        DechargerP1;
21   sequence P1_Traitement1[Train1Vide, Produit1Brut->Produit1Fini,
        Train1Vide] : ChargerP1, F1, F2, DechargerP1;
22   sequence P1_Traitement1_mono[Train1P1Brut->Train1P1Fini] : F1, F2;
23   sequence P1_Traitement2[Produit1Brut, Train1Vide->Produit1Fini,
        Train1Vide] : ChargerP1, F2, F3, F4, DechargerP1;
24   sequence P1_Traitement2_mono[Train1P1Brut->Train1P1Fini] : F2, F3,
        F4;
25   sequence EntrerTrain1[Train1Vide->Train1Vide] : MettreTrainSurReseau
        ;
26   sequence SortirTrain1[Train1Vide->Train1Vide] : SortirTrainDuReseau;
27   sequence MoveTrain1[Train1Vide->Train1Vide] : ResetTrain;
28 }
```

fonctions ont été définies et concernent la gestion des trains. Les fonctions `ChargerP1` et `DechargerP1` correspondent comme leur nom l'indique au chargement et au déchargement du produit `P1` sur le train qui conduit à la constitution d'un nouveau produit représentant la locomotive. Nous avons choisi de faire apparaître explicitement ces fonctions, tout comme la mise en place du train sur le réseau ainsi que sa sortie du réseau représentées par les fonctions `MettreTrainSurReseau` et `SortirTrainDuReseau`. La fonction `ResetTrain` est définie pour permettre l'exécution de la séquence `MoveTrain1` qui réalise une phase de préparation du train.

Deux types de produits (`Produit1Brut` et `Produit1Fini`) correspondent aux pièces lorsqu'elles ne sont pas sur les trains. Trois autres produits représentent les locomotives dans un état déchargé ou bien chargé par ces pièces (`Train1Vide`, `Train1P1Brut`, `Train1P1Fini`).

Quant aux séquences de fonctions, deux traitements sont considérés : `Traitement1` et `Traitement2`. Chacun de ces deux traitements est décrit par une séquences de fonctions (lignes 21 et 23). Pour chacun de ces deux traitements, une deuxième séquence de fonctions a été définie sans les chargements (lignes 22 et 24), les séquences de chargement et déchargement sont alors réalisées séparément et décrites aux lignes 19 et 20. La séquence `MoveTrain1`, décrite à la ligne 27, effectue une phase de préparation du train. Et les séquences `EntrerTrain1` et `SortirTrain1`, décrites aux lignes 25 et 26, correspondent respectivement à une phase de préparation du train depuis l'entrée du système ainsi qu'une phase de sortie du système.

Définition des opérations

La définition des opérations potentielles est décrite au listing 6.2. Ces définitions permettent de décrire les emplacements où sont réalisées les fonctions ainsi que les lieux de stockage des produits.

Listing 6.2 – Opérations potentielles du démonstrateur ferroviaire

```
operation F1T1(F1, T1);
operation F2T4(F2, T4);
operation F2T11(F2, T11);
operation F3T5(F3, T5);
operation F3T21(F3, T21);
operation F4T10(F4, T10);
operation F4T19(F4, T19);
operation MettreTrainSurReseau(MettreTrainSurReseau, IN);
operation SortirTrainDuReseau(SortirTrainDuReseau, OUT);
operation ResetTrain_T4(ResetTrain, T4);
operation ResetTrain_T10(ResetTrain, T10);
operation Charger_T5(ChargerP1, T5);
operation Charger_T11(ChargerP1, T11);
operation Decharger_T4(DechargerP1, T4);
operation Decharger_T10(DechargerP1, T10);
stocking Train1IN(Train1Vide, IN);
stocking Train1OUT(Train1Vide, OUT);
stocking Train1Vide_T4(Train1Vide, T4);
stocking Train1Vide_T10(Train1Vide, T10);
stocking Train1Vide_T5(Train1Vide, T5);
stocking Train1Vide_T11(Train1Vide, T11);
stocking Train1P1Brut_T5(Train1P1Brut, T5);
stocking Train1P1Brut_T11(Train1P1Brut, T11);
stocking Train1P1Fini_T4(Train1P1Fini, T4);
stocking Train1P1Fini_T10(Train1P1Fini, T10);
stocking Produit1Brut_T5(Produit1Brut, T5);
stocking Produit1Brut_T11(Produit1Brut, T11);
stocking Produit1Fini_T4(Produit1Fini, T4);
stocking Produit1Fini_T10(Produit1Fini, T10);
```

6.2.2 Analyse de l'architecture

Les résultats de l'analyse de l'architecture du démonstrateur sont maintenant présentés. Les objectifs de ces analyses sont, d'évaluer la structure du circuit et sa capacité à se reconfigurer. Les résultats de l'analyse structurelle sont donnés dans un premier temps, suivis par l'analyse de la tolérance de cette architecture. Les résultats sont obtenus en mettant en œuvre le cadre d'analyse décrit au chapitre 4.

Analyse structurelle

L'architecture physique du circuit est composée de 12 ressources stationnaires, 7 ressources de transport et 28 connexions. Le parallélisme potentiel des transport a pour valeur $TP_A = 0.58$, dans le cas du circuit, cela indique qu'il y a pratiquement deux fois plus de tronçons que d'aiguillages. Le nombre de connexions par ressource stationnaire, quant à lui est égal à $CS_A = 2.33$. Ce résultat est équivalent à celui de la grille analysée à la section 4.5.1, une grande variété de transferts est donc proposée. L'analyse de l'architecture montre ainsi que deux transferts sont possibles à partir de `T19`, ce nombre peut monter jusqu'à quatre pour un tronçon tel que `T1`.

Pour l'architecture logique, constituée de dix fonctions utilisées par neuf séquences définissant cinq produits, les parallélismes des traitements ont pour valeur $PP_{A,P1\_Traitement1} = 5$ et

$PP_{A,P1\_Traitement2} = 6$. Les deux séquences d'opérations peuvent donc être assez facilement parallélisées sur l'architecture physique du circuit.

Tolérance de l'architecture

Le GAO du circuit est présenté à la figure 6.4. Les résultats de criticité sont relatifs aux séquences de fonctions.

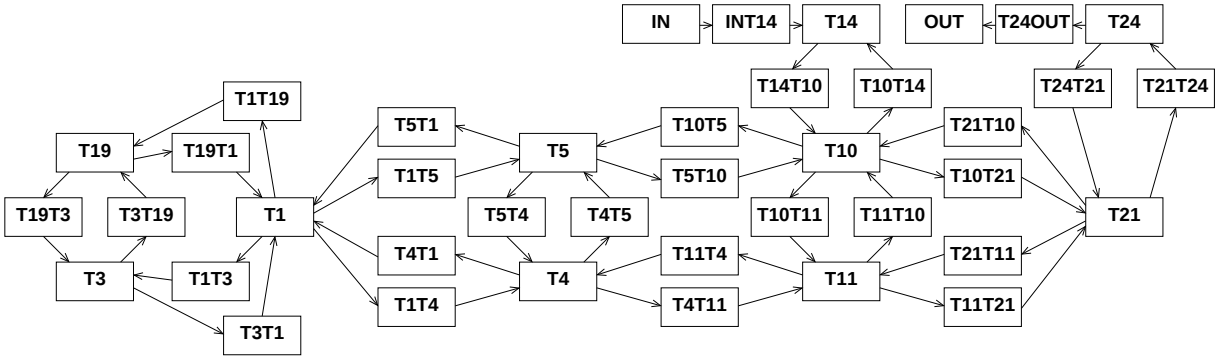


FIG. 6.4 – GAO pour l'architecture du circuit

Les critères globaux de criticité indiquent que 25% des opérations (14 sur 55) sont critiques. En ce qui concerne les ressources, 58% d'entre elles sont critiques.

Remarque : si on considère des entrées et sorties du système au niveau des ressources de stockage (lieux où s'arrêtent les trains pendant le fonctionnement du système), les résultats obtenus sont bien meilleurs. Ainsi, l'indice de criticité des opérations tombe à 12% et celui des ressources à 16%, de plus aucune ressource de transport n'est critique.

Bilan sur l'analyse de l'architecture du circuit

La modélisation du circuit à l'aide du langage de description des systèmes reconfigurables a permis d'évaluer les potentialités de l'architecture du circuit. Il ressort des analyses que celui-ci offre de bonnes capacités de parallélisation, aussi bien en terme de transferts que de traitements. La tolérance de l'architecture est très bonne, même si le placement de l'entrée et de la sortie du système n'est pas optimal par rapport aux mesures de criticité. L'utilisation de stocks intermédiaires résout ce problème.

6.2.3 Une configuration du démonstrateur

Pour le circuit ferroviaire, une configuration permet de décrire les lieux de traitement des produits ainsi que les routes empruntées pour atteindre ces lieux. Un exemple de configuration pour le démonstrateur, décrite à l'aide du langage DeSyRe, est donné sur le listing 6.3. Cette configuration définit la réalisation de la séquence de fonctions `Traitement2` sur la boucle intérieure. Les autres configurations développées pour le démonstrateur sont décrites en annexe B

Listing 6.3 – Configuration pour le démonstrateur ferroviaire

```
1 -- F2F3F4 sur la boucle interieure uniquement
2 configuration l5c3 {
3   logical configuration {
4     instance IF2 of F2;
5     instance IF3 of F3;
```

```
6       instance IF4 of F4;
7       instance IFT of FT;
8       instance I_MettreTrainSurReseau of MettreTrainSurReseau;
9       instance I_SortirTrainDuReseau of SortirTrainDuReseau;
10      instance I_ChargerP1 of ChargerP1;
11      instance I_DechargerP1 of DechargerP1;
12      instance I_ResetTrain of ResetTrain;
13    }
14
15    physical configuration {
16      transfer sequence TS_INT11 {
17        transfer INT14 using Loader;
18        control from T14.T14_p to T14.T14_a;
19        transfer T14T10 using A7;
20        control from T10.T10_b to T10.T10_a;
21        transfer T10T11 using A5;
22      }
23      transfer sequence TS_T11T21 {
24        transfer T11T21 using A15;
25      }
26      transfer sequence TS_T21T10 {
27        transfer T21T10 using A7;
28      }
29      transfer sequence TS_T10T11 {
30        transfer T10T11 using A5;
31      }
32      transfer sequence TS_T10OUT {
33        transfer T10T11 using A5;
34        control from T11.T11_a to T11.T11_b;
35        transfer T11T21 using A15;
36        control from T21.T21_b to T21.T21_b;
37        transfer T21T24 using A15;
38        control from T24.T24_a to T24.T24_p;
39        transfer T24OUT using Loader;
40      }
41    }
42
43    transfer operation TO_INT11 : IFT using TS_INT11;
44    transfer operation TO_T11T21 : IFT using TS_T11T21;
45    transfer operation TO_T21T10 : IFT using TS_T21T10;
46    transfer operation TO_T10T11 : IFT using TS_T10T11;
47    transfer operation TO_T10OUT : IFT using TS_T10OUT;
48
49    stationary operation SO_MettreTrainSurReseau :
          I_MettreTrainSurReseau using MettreTrainSurReseau;
50    stationary operation SO_SortirTrainDuReseau : I_SortirTrainDuReseau
          using SortirTrainDuReseau;
51
52    stationary operation SO_F4T10 : IF4 using F4T10;
53    stationary operation SO_F2T11 : IF2 using F2T11;
54    stationary operation SO_F3T21 : IF3 using F3T21;
55    stationary operation SO_Charger_T11 : I_ChargerP1 using Charger_T11;
56    stationary operation SO_Decharger_T10 : I_DechargerP1 using
          Decharger_T10;
57    stationary operation SO_ResetTrain_T10 : I_ResetTrain using
          ResetTrain_T10;
58
59    operation sequence OS_MTR2 realize EntrerTrain1 :
          SO_MettreTrainSurReseau, TO_INT11;
60    operation sequence OS_STR2 realize SortirTrain1 : TO_T10OUT,
          SO_SortirTrainDuReseau;
61    operation sequence OS_MoveTrain1_2 realize MoveTrain1 :
          SO_ResetTrain_T10, TO_T10T11;
62    operation sequence OS_P1_Traitement2_2 realize P1_Traitement2 :
          SO_Charger_T11, SO_F2T11, TO_T11T21, SO_F3T21, TO_T21T10,
          SO_F4T10, SO_Decharger_T10;
63  }
```

La définition de la configuration débute par l'instanciation des différentes fonctions dans la configuration logique réalisée aux lignes 3 à 13. Les différentes séquences de transfert permettant d'effectuer les déplacements sur la boucle intérieure sont alors décrites dans la configuration physique (lignes 15 à 41). Vient ensuite la définition des différentes opérations, les opérations de transfert, tout d'abord (lignes 43 à 47), puis les opérations stationnaires (lignes 49 à 57). Dans le cadre de cette configuration, la fonction `F2` est réalisée sur le tronçon `T11`, la fonction `F3` sur le tronçon `T21` et la fonction `F4` sur `T10`.

Quatre séquences d'opérations sont définies dans le cadre de cette configuration. Les trois premières décrites aux lignes 59, 60 et 61 gèrent la mise en place et le dégagement du train, ces séquences d'opérations réalisent les séquences de fonction `EntrerTrain1`, `SortirTrain1` et `MoveTrain1`. La dernière séquence d'opérations, décrite à la ligne 62, réalise la séquence de fonctions `P1_Traitement2` en utilisant les opérations stationnaires de traitement définies précédemment.

6.2.4 Résultats d'analyse sur la configuration

La configuration présentée pour le circuit est maintenant analysée, cette analyse permet d'identifier les caractéristiques d'une configuration pour le circuit. Les résultats de l'analyse structurelle sont d'abord présentés, précédant les résultats de l'analyse de tolérance.

Analyses structurelles

La configuration considérée utilise 7 ressources stationnaires et 4 ressources de transport sur les 12 ressources stationnaires et les 7 ressources de transport offertes par l'architecture. En ce qui concerne les connexions, sur 28 connexions définies dans l'architecture, 7 seulement sont utilisées dans la configuration. Cette configuration n'utilise donc pas l'architecture au maximum de ses capacités, par contre, le coût de son fonctionnement est plus faible que celui d'une configuration exploitant toute l'architecture si on considère que ce coût est lié au nombre de ressources utilisées.

Le nombre de connexions par ressource stationnaire a maintenant pour valeur $CS_C = 1$, la topologie de la configuration considérée est en effet proche de celle d'une ligne de transfert. En ce qui concerne le parallélisme de transfert potentiel, il est proche de celui de l'architecture soit $TP_C = 0.57$.

Seule la séquence de fonctions `Traitement2` est réalisée dans cette configuration. Sa durée d'exécution est de $\delta_{C,Traitement2} = 7$ et son parallélisme potentiel de $PP_{C,Traitement2} = 3$ alors que l'architecture permet une parallélisation de 6. Cette différence correspond au fait que seule une séquence d'opérations a été définie pour réaliser la séquence de fonction. En définissant une deuxième séquence d'opérations utilisant d'autres ressources du circuit, le parallélisme des séquences serait augmenté.

En résumé, cette configuration utilise un peu plus de la moitié des ressources offertes par l'architecture, une seule séquence de fonctions est réalisée. On retrouve donc pour la variété des connexions, des résultats proches d'une ligne, cette remarque est d'ailleurs corroborée par les analyses à partir du GAO présenté à la figure 6.5.

Tolérance de la configuration

Le graphe d'accessibilité opérationnelle de la configuration considérée est présenté à la figure 6.5. Deux contextes différents ont été considérés pour sa construction. Les éléments en

noirs correspondent à la prise en compte des reconfigurations mineures uniquement. L'ajout des éléments en gris permet de considérer les reconfigurations significatives.

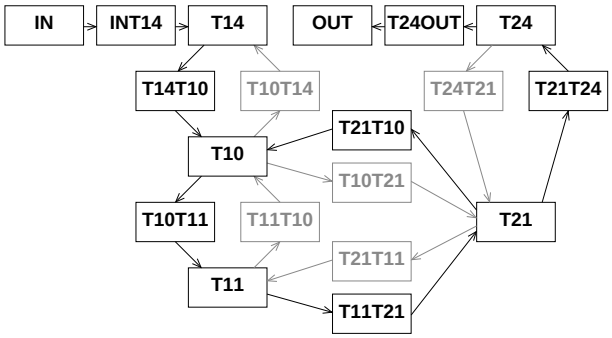


FIG. 6.5 – GAO de la configuration

La prise en compte des reconfigurations mineures uniquement donne des résultats de criticité très mauvais. En effet, toutes les opérations utilisées dans la configuration sont critiques.

L'extension du contexte en prenant en compte les reconfigurations significatives n'améliore pas énormément les résultats. 19 opérations sur 27 sont alors critiques dont 15 opérations de traitement, mais seules 22 opérations présentes dans le GAO sont utilisées et peuvent donc être critiques. En ce qui concerne les ressources, sur les 11 ressources utilisées, 10 sont critiques.

Bilan sur l'analyse de la configuration

L'utilisation de la boucle interne du circuit ferroviaire semble intéressante pour la réalisation de la séquence de fonctions `Traitement2`, les résultats en terme de performance sont corrects et le coût de fonctionnement faible. Les résultats de tolérance obtenus ne sont pas exceptionnels, les éléments de la boucle interne étant très critiques.

6.2.5 Bilan sur la modélisation du circuit

La modélisation du circuit de train en tant que système reconfigurable a permis l'application des analyses définies pour ces systèmes. Ainsi, le circuit dispose de capacités intéressantes pour mettre en œuvre la reconfiguration. Une configuration a ensuite été présentée et analysée.

La section suivante évalue la possibilité d'obtenir automatiquement le code de contrôle/-commande du circuit à partir de sa description haut niveau. Deux approches différentes sont alors utilisées et comparées.

6.3 Génération du code de contrôle/commande du circuit

Le chapitre 5 a décrit un cadre d'implantation pour l'architecture de commande d'un système reconfigurable. Ce cadre d'implantation est utilisé, dans cette section pour générer une partie de cette architecture de commande, à savoir le code de contrôle/commande, en prenant en compte les spécificités de l'architecture électronique utilisée.

6.3.1 Spécificités de la plate-forme

Le code de contrôle/commande est localisé sur un seul automate programmable. Cet automate est de la famille TSX Premium, commercialisée par Schneider Electric, il est programmé

à l'aide de l'outil PL7-Pro [Telemecanique, 2003b] qui offre la possibilité d'utiliser les langages de la norme IEC61131-3 [IEC, 2003] et qui permet d'importer un programme généré sous la forme d'un fichier textuel.

Au niveau du comportement des ressources, les constituants du circuit sont de deux types : tronçon ou aiguillage. De plus ceux-ci sont représentés par des ressources de type différent : ressource stationnaire pour les tronçons, ressource de transport pour les aiguillages. Le comportement des différentes ressources est directement décrit dans les transformations de modèles utilisées pour la génération de code.

Les entrées/sorties de l'automate pilotent les aiguillages et récupèrent les informations des capteurs. Une partie de ces entrées/sorties sont directement câblées sur l'automate, les autres sont connectées à un module déporté sur un bus CAN. La commande des locomotives est lue dans l'automate par un programme sur le PC, ce dernier envoie alors les ordres aux différentes locomotives. Pour pouvoir générer correctement le code de contrôle commande du circuit, il est nécessaire de relier ces entrée/sorties aux éléments du modèle, c'est ce qui est réalisé dans la section suivante.

6.3.2 Lien entre le modèle et la plate-forme matérielle

Les liens entre les entrées/sorties physiques de l'automate, les éléments du circuit (trains, aiguillages, capteurs) et le modèle de description haut niveau de ce circuit sont nécessaires à l'implantation doivent être définis. Ces liens dépendent de l'application et des technologies employées, c'est pourquoi ils ne sont pas définis dans le langage DeSyRe mais dans un modèle annexe, qui lui est rattaché par des références à ses éléments.

Des capteurs sont utilisés pour détecter le passage d'un train. Ils sont disposés aux extrémités des tronçons. Un tronçon est modélisé par une ressource stationnaire disposant de deux ports représentant ses extrémités. Chaque port peut donc être associé à une entrée de l'automate.

De même, la position d'un aiguillage est déterminée par le transfert que l'on souhaite réaliser. Au niveau du modèle, ce transfert est modélisé par une connexion. Chaque connexion peut donc être reliée à la commande de l'aiguillage qui lui correspond. Un aiguillage est un actionneur bi-stable, c'est-à-dire qu'il dispose de deux commandes séparées, l'une le positionne à gauche et l'autre à droite. Chaque commande est donc reliée à une sortie de l'automate, à laquelle un ensemble de connexions est associé.

En ce qui concerne les ordres envoyés aux locomotives, ils sont écrits dans la mémoire de l'automate par le code de contrôle/commande et lus par le PC pour être envoyés à l'interface. Chaque donnée associée aux trains (vitesse, sens, feux) est spécifiée dans la mémoire de l'automate. Une adresse de base est définie pour chacun des paramètres.

Un langage a été défini pour lier les entrées/sorties de l'automate aux éléments du modèle. Ce langage a été appelé IOA (Input Outputs Association) et est décrit par le méta-modèle représenté figure 6.6.

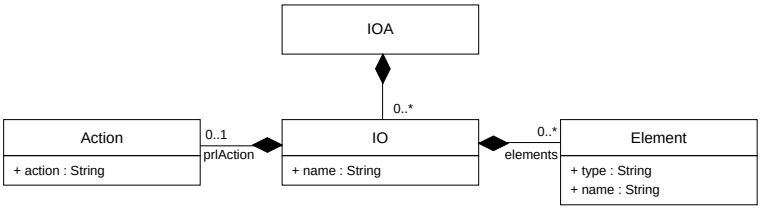


FIG. 6.6 – Méta-modèle pour le langage IOA

Un modèle IOA est constitué d'un ensemble d'entrée/sorties : `IO`, ces `IO` sont reliées à un ensemble d'éléments du modèle représentés par leur type et leur nom. Une action peut être associée à une `IO`, cette action est exécutée dans le traitement préliminaire du SFC et permet de réaliser l'acquisition de la valeur avant son utilisation.

Une syntaxe concrète textuelle, associée au méta-modèle IOA, permet de saisir un modèle IOA. Quelques exemples en provenance de l'IOA du circuit de train sont présentés sur le listing 6.4. La première `IO` est définie à la ligne 1, elle est associée à un port de l'architecture. La seconde, définie à la ligne 3 correspond à un capteur connecté à une entrée/sortie sur le bus CAN, et pour lequel une acquisition (copie en mémoire) est nécessaire. De plus, deux ports sont associés à la même entrée (le même capteur est utilisé pour délimiter les tronçons 3 et 19). La troisième `IO` est définie à la ligne 8, il s'agit d'une sortie, reliée à deux connexions qui lorsqu'elles sont réalisées, activent cette sortie. Pour les différentes commandes des trains, un exemple est donné à la ligne 13, les vitesses des différents trains sont enregistrées à partir de l'adresse "`%MW0`".

Listing 6.4 – Exemple de définitions d'un modèle IOA

```
1  io '%I2.2' { Port 'T1.T1_a' ; }  -- Capteur
2
3  io '%M3' ['%M3 := %MW20:X3;'] { -- Cas particulier pour le capteur C1
4    Port 'T19.T19_b';
5    Port 'T3.T3_b';
6  }
7
8  io '%Q3.1' { -- Une position de l'aiguillage A1 (droite)
9    Connection 'T1T4';
10   Connection 'T4T1';
11 }
12
13 io '%MW0' { Train 'speed'; } -- Vitesses des trains
```

Le langage IOA est nécessaire pour relier les concepts manipulés dans le langage DeSyRe, aux éléments physiques réalisant le lien entre le code de commande et la plate-forme que sont les entrées/sorties de l'automate. Ce lien ayant été formalisé, la génération automatique du code de contrôle/commande pour la plate-forme du démonstrateur est traitée dans la section suivante.

6.3.3 Génération du code de contrôle/commande à partir d'une configuration

La description de la configuration laisse envisager une première manière d'obtenir un code de contrôle/commande pour le train. Il s'agit de traduire les différentes séquences définies dans la configuration (séquences de transfert et séquences d'opérations) en SFC (Sequential Function Chart) et d'effectuer les actions décrites par les séquences.

Cette première approche a été mise en oeuvre sur la plate-forme du train, elle est présentée dans cette section. Dans un premier temps, le principe de génération du code est présenté. Les mécanismes permettant de lancer les différents SFC sont ensuite introduits. Finalement, les techniques de reconfigurations envisageables sont abordées.

Principe de génération

L'objectif de la première approche est de générer des SFC à partir des séquences d'opérations de la configuration. Les différentes étapes de cette génération sont décrites à la figure 6.7. Les modèles de départ sont la configuration, l'architecture ainsi que leur association avec les entrées/sorties de l'automate. A partir de ces trois modèles, un premier modèle de grafcet et

un modèle ST (Texte Structuré) décrivant les traitements postérieurs sont obtenus. Ces deux modèles sont ensuite utilisés par une troisième transformation qui génère un programme pour PL7-Pro.

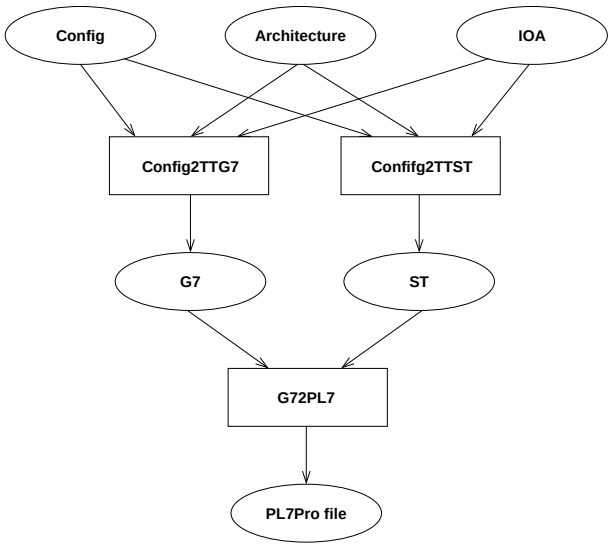


FIG. 6.7 – Génération de l'application à partir d'une configuration

La génération d'un modèle de grafcet à partir de la configuration comporte sept règles. La première créée l'élément racine du modèle grafcet, à partir de l'élément racine de la configuration. Les trois règles suivantes permettent d'obtenir un grafcet à partir d'une séquence de transfert. Deux de ces règles sont chargées de créer des étapes et des transitions à partir des différents noeuds de la séquence (`TransporterNode` et `ControlerNode`). La troisième règle réalise l'initialisation de la séquence et connecte les différentes étapes et transitions obtenues à partir des différents noeuds composant la séquence. Les trois dernières règles de la transformation permettent d'obtenir un grafcet pour chaque séquence d'opérations. Le principe est le même que pour les séquences de transfert, deux règles se chargent des deux types de nœuds composant la séquence, la troisième étant chargée de l'initialiser et d'ordonner les traitements.

Le lancement des séquences de transfert se fait par l'utilisation d'un 'bit' partagé par la séquence de transfert et les opérations de transfert qui la constituent. La séquence est démarrée par la mise à 1 de ce bit à l'entrée dans l'étape correspondant à l'opération de transfert. Ce bit est positionné à zéro par la séquence dès qu'elle est terminée.

La génération des différents grafcet est illustrée à la figure 6.8 qui présente à la fois le point de vue utilisateur du grafcet et le point de vue de l'automate qui est obtenu en utilisant l'association des entrées/sorties.

En ce qui concerne les traitements postérieurs, ils concernent l'activation des différents aiguillages et sont générés par la transformation "Config2TTST". A chaque `IO` associée à un élément `Connection` correspond une instruction chargée de mettre cette `IO` à 1 lorsque la connexion correspondante est utilisée (des connexions sont utilisées aux étapes 1 et 3 du grafcet présenté à la figure 6.8.

Reconfiguration

Le code de contrôle/commande envoyé à l'automate est généré à partir de la configuration. Une reconfiguration nécessite donc de générer à nouveau ce code de contrôle/commande.

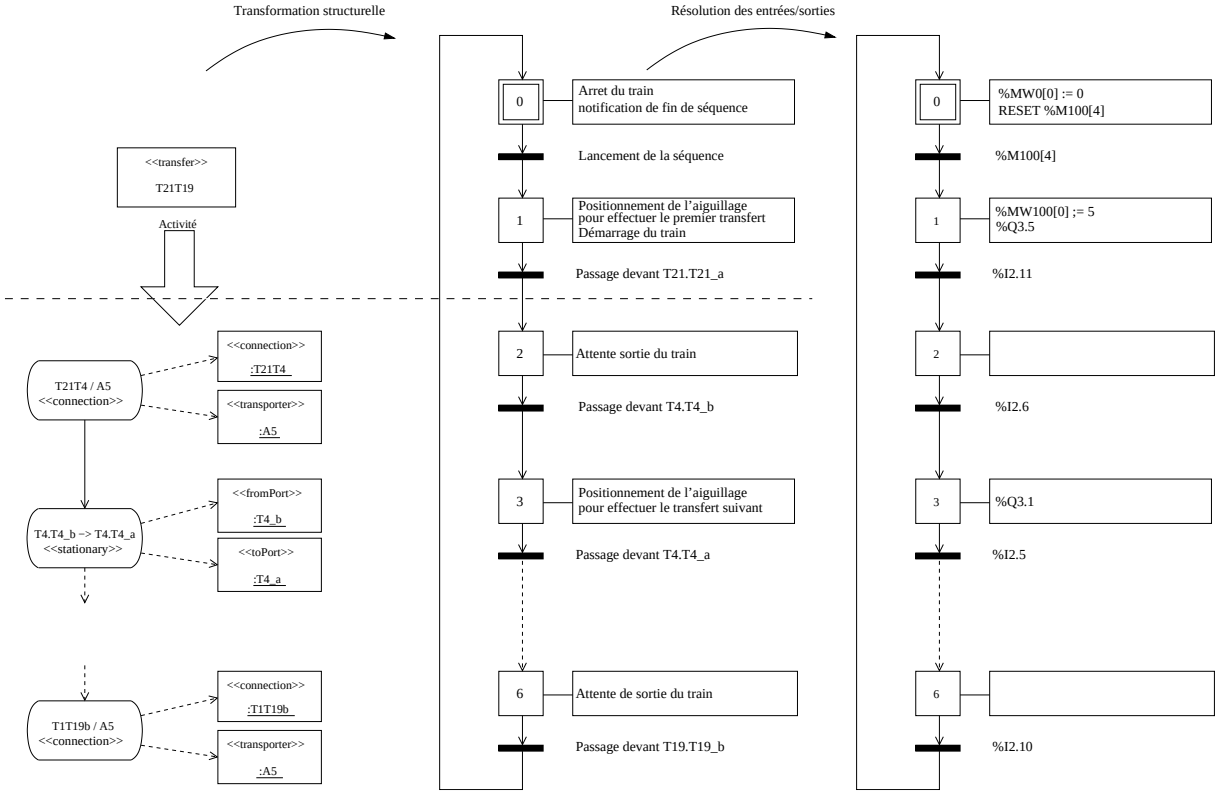


FIG. 6.8 – Transformation d'une séquence de transfert en grafcet

Le changement de configuration, avec cette méthode nécessite alors l'envoi d'un nouveau programme à l'automate et donc son arrêt. Cette étape est réalisée manuellement.

Lancement des séquences d'opérations

Le lancement des séquences d'opérations sur le train est réalisé à partir du PC. Pour permettre le lancement de ces séquences, chacune d'entre elles est associée à un bit de la mémoire de l'automate. La mise à 1 de ce bit provoque le lancement de la séquence.

Un programme appelé "ActionLauncher" écrit en Java et fonctionnant sur le PC est chargé de mettre ce bit à 1 en utilisant le protocole Modbus-TCP pour écrire dans la mémoire de l'automate. Afin de faire correspondre une séquence d'opérations à un bit de l'automate, les correspondances sont décrites dans un fichier obtenu automatiquement à partir du modèle IOA et du modèle de configuration.

Bilan sur la génération du code de contrôle/commande à partir de la configuration

Grâce à la transformation de modèles, le code de contrôle/commande est généré automatiquement à partir de la modélisation DeSyRe de la configuration et permet de faire fonctionner correctement les locomotives sur le circuit de train. Cette approche nécessite de remplacer le programme de l'automate lors d'une reconfiguration ce qui conduit à son arrêt. De plus, la gestion simultanée de plusieurs trains n'est pas prise en compte dans l'approche proposée.

Nous avons donc cherché une nouvelle approche permettant de pallier les défauts de cette première méthode qui, bien que parfaitement fonctionnelle, n'est pas totalement satisfaisante. Cette nouvelle approche est présentée à la section suivante.

6.3.4 Génération de l'application à partir de l'architecture

La deuxième mise en œuvre du cadre d'implantation utilise le modèle de l'architecture pour la génération du programme de l'automate. Des SFC sont associés à chaque tronçon et aiguillage. L'évolution des locomotives est déterminée par l'interprétation, par ces SFC, de tables de routages générées à partir de la configuration choisie. Une reconfiguration consiste donc à générer de nouvelles tables de routage et à les envoyer dans l'automate où les SFC pourront les interpréter.

Principe de fonctionnement

Les différents programmes de contrôle/commande sont obtenus à partir de l'architecture. Les deux types de composants correspondent à deux types de ressources (tronçons, aiguillages). Les composants associés aux tronçons sont chargés de mettre en œuvre les opérations de traitement réalisées sur ceux-ci, et de donner les ordres de déplacement aux trains pour qu'ils arrivent au tronçon suivant (changement de sens/vitesse). Les composants correspondant aux aiguillages, quant à eux, coordonnent les transferts entre tronçons et positionnent l'aiguillage afin de les réaliser.

Le fonctionnement des composants nécessite des informations en provenance de la configuration. Ces informations permettent, dans le cas d'un transfert, de connaître la direction que doit prendre le train lorsqu'il est sur un tronçon, tandis que pour les aiguillages il s'agit de pouvoir déterminer sa position.

L'utilisation de tables de routage pour chaque composant, générées à partir de la configuration a été retenue. Les tables de routage sont couramment utilisées au niveau des routeurs dans des réseaux de communication, elles sont présentes au niveau du routeur et associent une direction à la destination spécifiée sur le paquet qui est routé, ainsi même si le routeur ne connaît pas le destinataire, il sait lequel de ses voisins pourra acheminer le paquet à destination [Feldmann and Rexford, 2000].

Dans le cas des composants utilisés, les tables de routages associent la séquence de transfert et le port d'entrée du train à un port de sortie. Dans le cas des tronçons, la direction du train est déterminée à partir du port d'entrée et du port de sortie ; s'ils sont identiques alors le train doit changer de direction pour ressortir par où il est rentré, s'ils sont différents, le train continue dans la même direction et sort de l'autre côté du tronçon. Dans le cas des aiguillages, la connaissance du port d'entrée et du port de sortie permet de déterminer la connexion utilisée pour effectuer le transfert et ainsi positionner l'aiguillage en conséquence.

L'utilisation de composants associés aux éléments de l'architecture permet de simplifier la gestion des conflits entre ces différents éléments lorsque plusieurs trains sont sur le réseau. Il s'agit d'éviter qu'un train ne se déplace vers un tronçon occupé par un autre train, ou que deux trains empruntent le même aiguillage. Ce sont les composants associés aux aiguillages qui sont chargés de faire l'intermédiaire entre les composants associés aux tronçons comme le montre la figure 6.9. Avant de donner l'ordre au train de se déplacer, le tronçon source (`t1`) effectue une requête auprès de l'aiguillage (`a1`) pour réaliser un transfert. Celui-ci vérifie que le tronçon d'arrivée (`t2`) est disponible et en informe `t1` qui lance le transfert tout en positionnant correctement l'aiguillage. Si le tronçon n'est pas disponible, `a1` attend qu'il le devienne avant d'informer `t1`. Une fois la locomotive arrivé sur `t2`, c'est ce dernier qui en prend les commandes.

Ce mécanisme de réservation permet d'éviter les collisions entre les trains mais n'évite pas d'éventuels blocages qui pourraient apparaître si par exemple deux trains se trouvent de chaque côté d'un aiguillage et veulent occuper le tronçon de l'autre train. Des mécanismes au niveau de la supervision doivent donc être mis en œuvre pour éviter de telles situations.

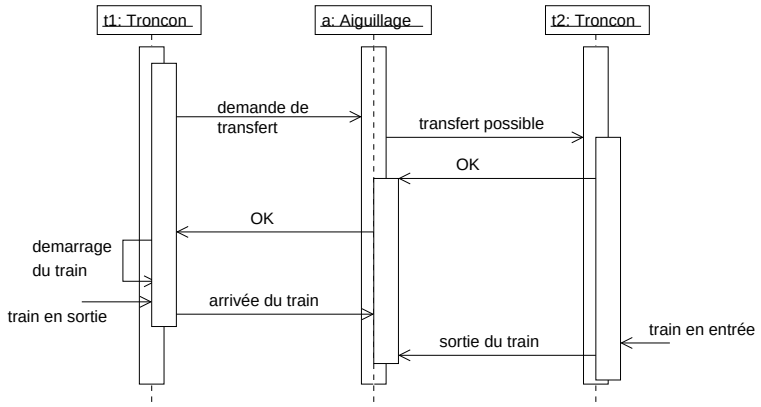


FIG. 6.9 – Coopération entre les composant aiguillage et tronçon

La section suivante traite du code généré pour mettre en œuvre les principes évoqués dans cette section, ainsi que des méthodes utilisées pour obtenir ce code.

Mise en œuvre

Le code de contrôle/commande associé aux ressources (tronçons et aiguillages) est implémentée à l'aide de langages de la norme IEC61131-3 que sont le SFC et le ST ainsi que les blocs fonctionnels programmés en ST interprétant les tables de routage. La constitution d'un composant est présentée à la figure 6.10, qui établit le lien entre chaque composant et le code de contrôle/commande qu'il contribue à constituer.

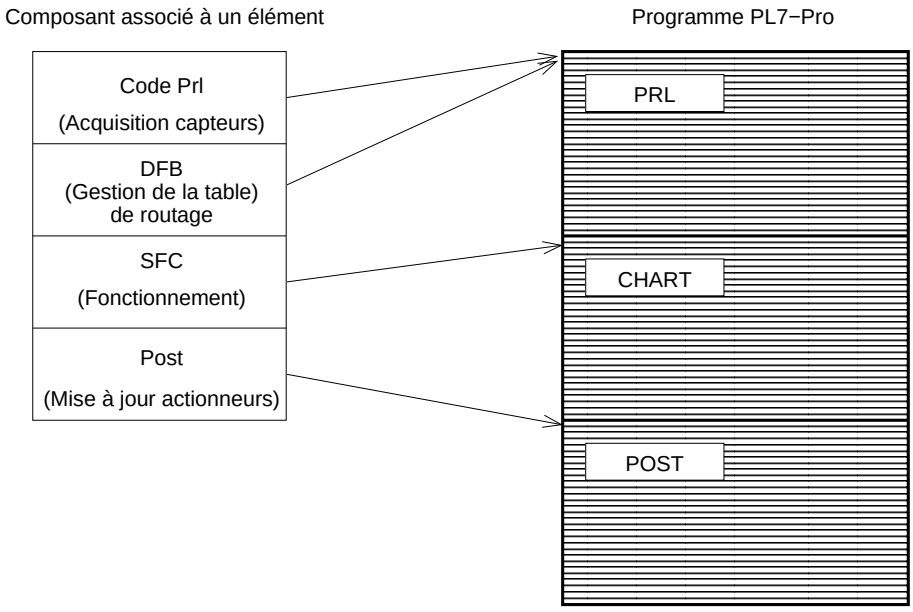


FIG. 6.10 – Composition d'un composant associé à un élément du circuit

La communication entre les différents composants s'effectue par l'intermédiaire des différentes étapes des SFC ainsi qu'au travers des entrées/sorties des blocs fonctionnels. Le code de contrôle/commande ainsi que la description de leur fonctionnement sont présentés en annexe C.

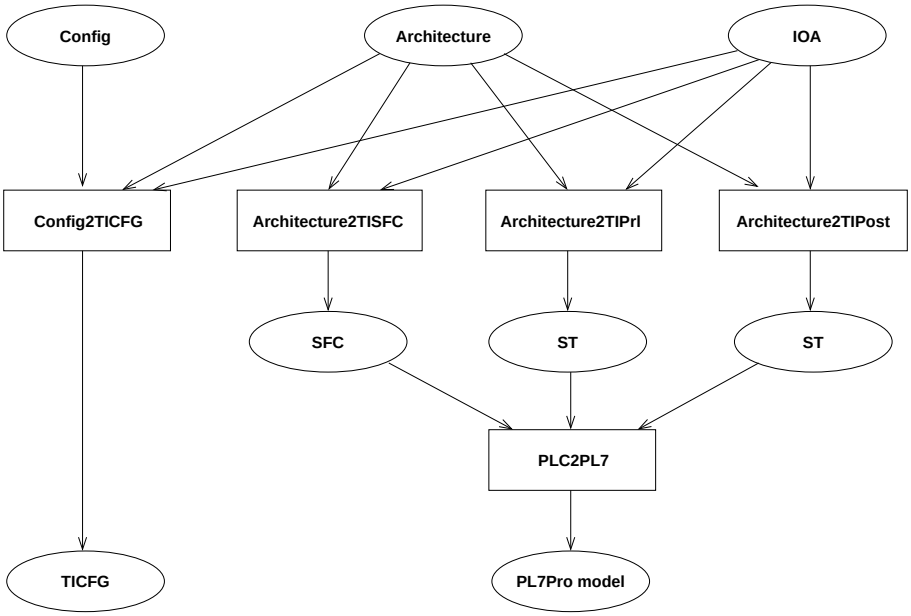


FIG. 6.11 – Génération des SFC à partir de l'architecture

Génération de la commande

La transformation de modèle est utilisée pour l'implantation de la commande. Celle-ci est réalisée conformément à la figure 6.11.

Ainsi, la génération du code de contrôle/commande est réalisée uniquement à partir du modèle de l'architecture et de son association aux entrées/sorties. Trois transformations différentes sont appliquées. La première génère un modèle contenant les SFC alors que les deux autres sont chargés des traitements préliminaires et postérieurs écrits en ST.

La première transformation : "Architecture2TISFC" est constituée de quatre règles. La première règle génère la section contenant les SFC à partir de l'architecture. Les deux règles suivantes génèrent les SFC pour les ressources (une règle par type de ressource). La quatrième règle ajoute des étapes et des transitions au SFC de la ressource stationnaire reliée à l'entrée du réseau pour permettre l'entrée du train sur le circuit.

La seconde transformation : "Architecture2TIPrl" est constituée de quatre règles. La première génère la section ST correspondant au traitement préliminaire. Les deux suivantes génèrent les séquences ST activant les blocs fonctionnels associés aux ressources. La dernière règle génère une action associée à certaines `IO` (acquisition) et spécifiées dans le modèle IOA.

La troisième transformation : "Architecture2TIPost" est constituée de deux règles. La première génère la section ST correspondant aux traitements postérieurs. La seconde règle génère les séquences ST correspondant au positionnement des aiguillages à partir des `IO` correspondantes dans le modèle IOA.

Les modèles alors générés sont conformes aux langages de la norme IEC-61131-3 [IEC, 2003]. Une nouvelle transformation est nécessaire afin d'obtenir un programme pour PL7-Pro. Un méta-modèle a alors été défini pour représenter les concepts utilisés dans les programmes PL7-Pro [Telemecanique, 2003b].

La configuration de l'application, sous la forme d'un ensemble de séquences d'opérations et de tables de routage est obtenue par une autre transformation de modèles "Config2TICFG", utilisant cette fois-ci le modèle de configuration. Cette transformation est constituée de trois règles. La première crée la racine du modèle de configuration. Les deux autres règles permettent de générer des tables de routage pour chaque ressource de l'architecture.

Reconfiguration

La reconfiguration du circuit de train est réalisée en envoyant une nouvelle configuration dans la mémoire et en mettant à jour les adresses des tables de routage parmi les données associées aux composants. Actuellement, un arrêt des trains dans une position déterminée est nécessaire.

Lancement des séquences

Ce nouveau cadre de génération de la commande est plus abouti que le premier. Il autorise en effet l'utilisation de plusieurs trains en parallèle et dispose de mécanismes facilitant la reconfiguration.

Un nouveau programme a été réalisé pour gérer le circuit en reprenant les principes du programme réalisé précédemment et en ajoutant plusieurs fonctionnalités. Ainsi la possibilité de contrôler plusieurs trains a été ajoutée, ainsi que le chargement d'une nouvelle configuration au sein de l'interface graphique et son téléchargement dans l'automate. Une capture d'écran de cette interface est donnée à la figure 6.12.

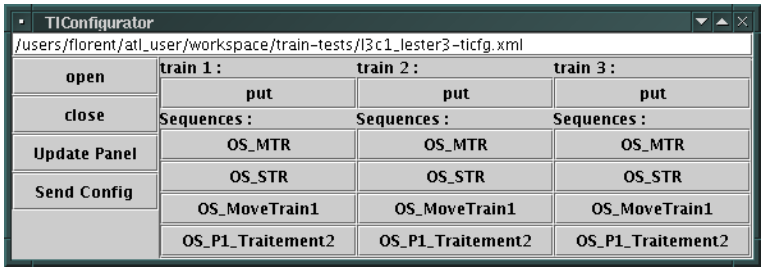


FIG. 6.12 – Interface de commande et de reconfiguration des trains

Les différentes commandes qui peuvent être envoyées aux trains apparaissent sur les trois colonnes de droite. Ces commandes sont générées à partir du fichier de configuration chargé, les ordres sont directement envoyés à l'automate via la liaison modbus-TCP.

Sur la gauche, une série de boutons concerne la gestion de l'automate. Les deux premiers permettent d'établir ou de stopper la communication avec celui-ci. Le troisième charge une configuration dont le chemin est donné dans la zone de texte présente en haut de la fenêtre, les boutons de commande du train sont alors mis à jour. Le dernier bouton permet d'envoyer la configuration chargée à l'automate.

Bilan sur la génération de code à partir de l'architecture

Cette deuxième approche pour la génération du code de contrôle/commande s'est révélée fructueuse. Elle a tout d'abord permis d'ajouter des fonctionnalités manquantes dans la première technique. La gestion de plusieurs trains a ainsi pu être éprouvée. Concernant la reconfiguration, celle-ci est réalisé directement à partir de l'interface de commande, aucune intervention par l'intermédiaire du logiciel PL7-Pro est nécessaire, l'automate ne doit donc pas être arrêté.

De plus, la séparation architecture/configuration est clairement visible dans le processus d'implantation et de fonctionnement. Les mécanismes de reconfiguration actuellement implémentés permettent d'utiliser la plate-forme pour évaluer des scénarios de reconfiguration.

6.3.5 Bilan sur la génération du code de contrôle/commande

Deux flots de génération de code ont été expertisés. Le premier permet d'obtenir, à partir d'une configuration, des SFC gérant les déplacements du train. Le second flot permet d'obtenir un programme qui interprète une configuration donnée sous la forme de séquences d'opérations ordonnant les opérations complétées par des tables de routages qui permettent l'acheminement des trains aux lieux caractéristiques souhaités.

Au point de vue de la complexité, la seconde approche est plus coûteuse. La taille du code de transformation est plus important (700 lignes contre 350) et le temps de développement nécessaire a été plus élevé (4 semaines contre 2).

La seconde approche possède néanmoins des apports indéniables. Elle intègre une gestion des ressources qui n'a pas été mise en place dans la première. La reconfiguration est aussi plus facile à mettre en œuvre lorsque le programme de l'automate ne doit pas être changé.

La seconde approche a été préférée pour l'étude de la mise en œuvre de la reconfiguration sur le démonstrateur ferroviaire qui est présentée dans la section suivante.

6.4 Fonctionnement du système

Le démonstrateur ferroviaire a été modélisé à l'aide du langage de description des systèmes reconfigurables. Cette description a permis à la fois l'évaluation des capacités du circuit et la génération du code de contrôle/commande permettant de le faire fonctionner.

Cette section traite du fonctionnement du système. Tout d'abord, la signification de la reconfiguration dans le contexte du démonstrateur est évoquée. Des outils sont ensuite mis en place pour aider au choix d'une configuration et la lancer.

6.4.1 Projection des concepts de reconfiguration dans le cadre du circuit

Avant d'être utilisés, les concepts présentés au chapitre 5 doivent être projetés sur l'application. C'est l'objet de cette section. L'utilisation de la gestion des modes est d'abord abordée, le concept de niveaux de reconfiguration est ensuite projeté sur l'application. Pour finir, le processus de reconfiguration mis en place dans le contexte du circuit de train est présenté.

Gestion des modes

Des modes peuvent être définis pour les ressources constituant le circuit de train. En associant ces modes au mécanisme de décision, la cohérence entre le mode des ressources et la configuration choisie peut être déterminée.

Différents modes ont été associés aux ressources. Dans la famille de mode "production", trois modes sont considérés : les modes d'arrêt (ON/OFF), des modes de marche (normal/-dégradé/hors-service) ainsi qu'un mode d'utilisation (disponible/non-disponible). Un méta-modèle a été défini pour associer un mode à une ressource.

La cohérence entre la configuration et le mode des ressources est déterminée par rapport aux familles des modes de marche et d'utilisation selon la règle suivante : "une ressource utilisée dans la configuration ne doit-être ni hors-service, ni non-disponible". Une nouvelle analyse, basée sur le cadre présenté au chapitre 4 a alors été définie. Une transformation permet de déterminer si la configuration est cohérente avec le mode des ressources, cette cohérence est déterminée par la routine OCL présenté sur le listing 6.5.

Listing 6.5 – Helper pour vérifier la cohérence du mode d'une ressource

```
1 helper context Architecture!Resource def : violateMode : Boolean =
2     self.isUsed and (self.mode.workingMode = #down or self.mode.
        usageMode = #unavailable);
```

Le mode des différentes ressources est renseigné manuellement à l'aide d'une interface homme-machine ajoutée au programme de gestion du train et dont une capture d'écran est fournie à la figure 6.13. Cette interface est couplée à la transformation de modèles et permet de déterminer si le mode des ressources est cohérent avec une configuration. Elle permet aussi de mettre à jour le mode d'arrêt des ressources après reconfiguration.

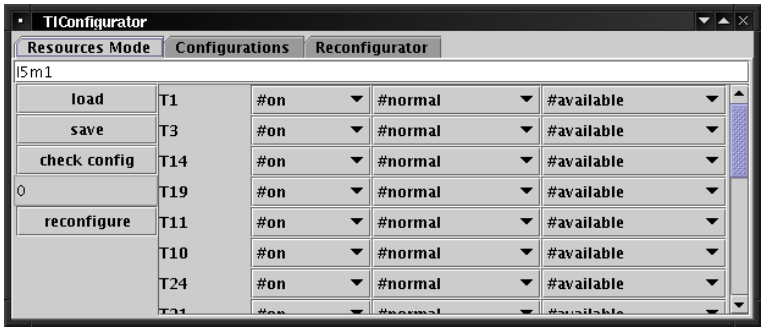


FIG. 6.13 – Interface de gestion des modes

Processus de reconfiguration

L'adaptation du processus de reconfiguration à l'exemple du train est représentée par l'organigramme de la figure 6.14.

L'application de gestion de la plate-forme est totalement intégrée dans cet organigramme. Le premier panneau concernant la gestion des modes est d'abord utilisé pour mettre à jour manuellement le mode des ressources après défaillance et vérifier la cohérence entre le mode actuel et la configuration. Si le mode ne correspond pas, le choix d'une nouvelle configuration (conformément à l'étude menée à la section 6.4.2) est réalisé. La vérification de la cohérence entre l'état des produits et la configuration ainsi que la construction d'une configuration de transition sont effectués manuellement. Le panneau 3 permet de transférer la nouvelle configuration et de commander la plate-forme.

Bilan

Les différentes étapes du processus de reconfiguration ont été présentées. L'adaptation des concepts dans le cadre du démonstrateur limite le fonctionnement des différents modules du contrôle/commande. Ainsi les modules de surveillance et de pilotage ne sont pas encore implantés et le module de gestion des modes est limité à un recensement manuel du mode des différentes ressources.

Le démonstrateur offre tout de même des perspectives très intéressantes pour l'étude du processus de reconfiguration dans le cadre de la figure 6.14. Au sein de ce processus, le choix d'une configuration constitue une étape très importante, présentée dans la section suivante.

6.4.2 Choix d'une configuration

Le choix d'une nouvelle configuration est réalisé par rapport à un ensemble de critères qui dépendent de l'application ainsi que des objectifs qui lui sont assignés. Il est donc nécessaire

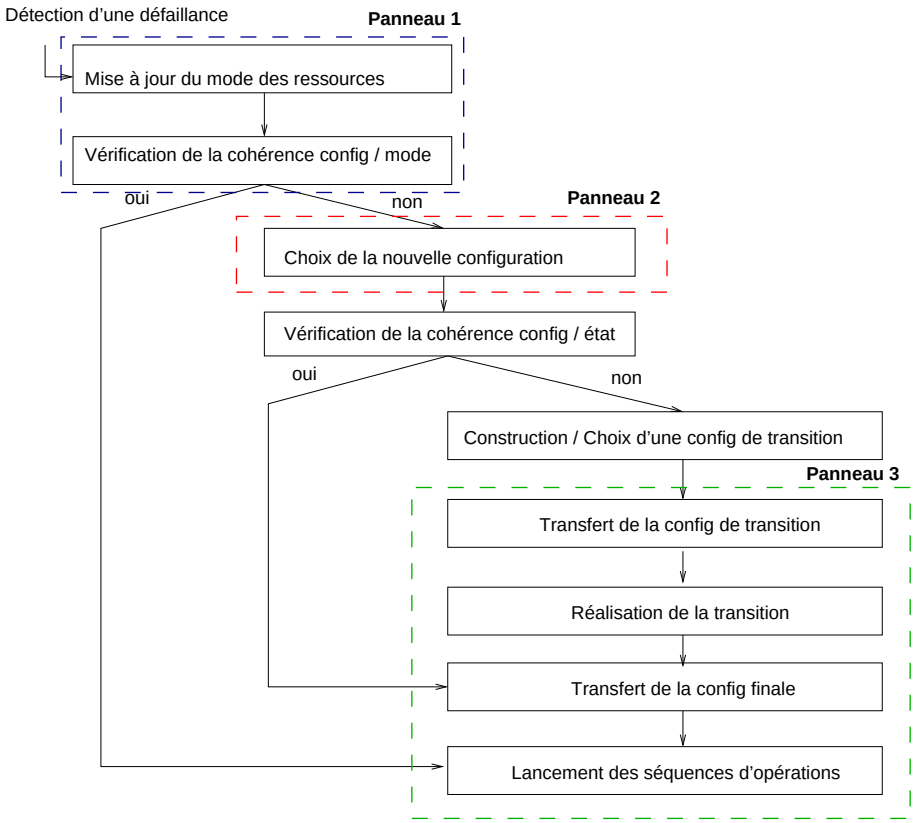


FIG. 6.14 – Scénario de reconfiguration

de définir les critères pertinents dans le cas du démonstrateur ferroviaire et de proposer une méthode de sélection multi-critères pour choisir une nouvelle configuration.

Nombre de ressources

Les critères structurels constituent un premier ensemble de métriques parmi lesquelles des critères pour le choix d'une configuration sont choisis.

Parmi les critères structurels, le nombre de ressources utilisées par la configuration permet d'évaluer la taille de la configuration et donc son coût global. Ce critère constitue le premier indicateur utilisé pour le choix d'une configuration.

Critère de tolérance

La criticité apparaît comme un critère pertinent lors du choix d'une configuration. Elle permet de quantifier la tolérance du système face aux défaillances de ses éléments.

Le nombre de ressources critiques par rapport aux séquences de fonctions en vue d'une reconfiguration significative est considéré comme un critère pertinent. Il permet de limiter le coût inhérent à la mise en œuvre d'une nouvelle reconfiguration en cas de panne.

Durée de traitement

La durée de réalisation des séquences de fonctions que l'on souhaite réaliser doit aussi être prise en compte, elle permet de caractériser la productivité de la configuration.

Les résultats de cette métrique ne prennent pas en compte le fonctionnement de la configuration de manière globale, les séquences sont évaluées séparément et le parallélisme n'est pas traité. Pour cela, une étude d'ordonnancement doit être envisagée.

Immunité contre les blocages

Sur le démonstrateur, les différentes commandes sont lancées manuellement. L'utilisateur peut provoquer des situations de blocage si les commandes envoyées aux trains conduisent à une situation dans laquelle deux trains se retrouvent de chaque côté d'un aiguillage et souhaitent prendre la place l'un de l'autre.

Les situations de blocages n'apparaissent pas dans toutes les configurations, un moyen de les éviter consiste alors à choisir une configuration dans laquelle les blocages ne sont pas possibles. On peut déterminer l'absence de blocage sur une configuration en la transformant en un réseau de Petri [Murata, 1989] sur lequel une analyse de vivacité est réalisée à l'aide de l'outil TINA [Berthomieu et al., 2003]. La vivacité du réseau dépend du marquage initial qui correspond au nombre de locomotives simultanément sur le réseau. Pour un nombre de train `n`, si le réseau de Petri est vivant, alors jusqu'à `n` trains peuvent se trouver présents sur le circuit sans qu'aucun blocage ne puisse survenir.

La construction du réseau de Petri est réalisée à partir des séquences d'opérations. Chaque séquence d'opérations est transformée en une séquence `(place->transition)*->place` où chaque place représente les différents tronçons empruntés lors de la réalisation de la séquence. Chaque transition de cette séquence prend le jeton du tronçon sur lequel la locomotive se dirige et libère celui du tronçon d'origine. Deux transitions sont ajoutées, la première appelée "start" représente le démarrage de la séquence, alors que la seconde : "end" en modélise la fin. Ces transitions sont liées à la séquence et aux places représentant l'attente d'une nouvelle gamme dans un tronçon, générées pour chaque tronçon sur lequel le train peut attendre un nouvel ordre.

Le réseau de Petri présenté à la figure 6.15 a été obtenu à partir de la configuration l5c3 présentée au listing 6.3. La séquence qui apparaît sur la droite de la figure correspond à la séquence d'opérations `OS_P1_Traitement2`. A la gauche de cette séquence, on retrouve la séquence permettant la réinitialisation du train. Les séquences réalisant la mise en place et la sortie du train apparaissent horizontalement en haut et en bas de la figure. Deux jetons composent le marquage initial de la place `IN_wait`. Le réseau de Petri étant vivant, cela signifie que pour deux trains, la configuration ne conduit pas à une situation de blocage. L'ajout d'un jeton dans la place `IN_wait` rend le réseau non-vivant. En effet, le circuit peut se retrouver dans une situation de blocage si les trois trains sont sur `T10`, `T11` et `T21` et souhaitent aller sur les tronçon `T11`, `T21` et `T10` respectivement.

Cohérence avec les modes

La section 6.4.1 a introduit la gestion des modes dans le cadre du circuit de train. Le mode des tronçons et des ressources est pris en compte lors du choix d'une nouvelle configuration.

Tout d'abord, il faut vérifier que la configuration peut être mise en place en déterminant si elle est cohérente avec le mode actuel des ressources. Cette vérification constitue le premier résultat concernant la gestion des modes.

Dans un deuxième temps, le coût du changement de mode des ressources pendant la reconfiguration doit être évalué. Ce coût participe au coût global de la reconfiguration. Son évaluation consiste à déterminer combien de ressources doivent changer de mode d'arrêt (ON/OFF).

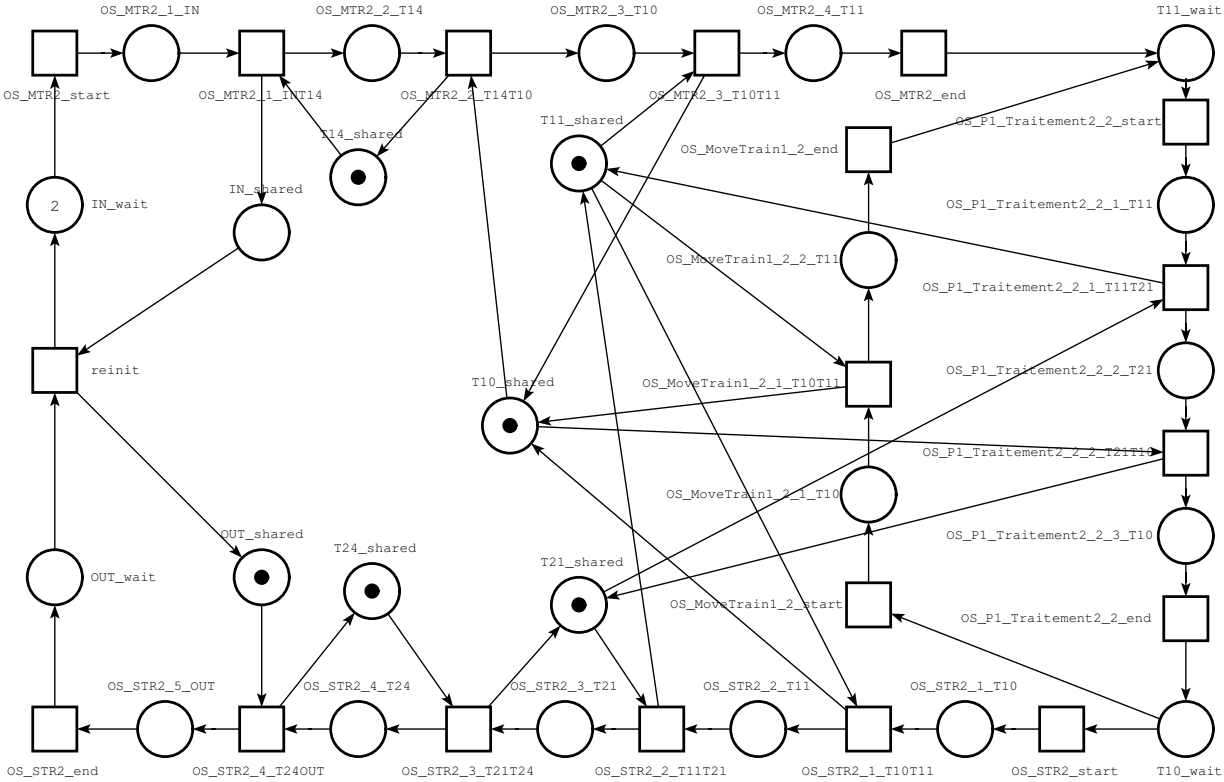


FIG. 6.15 – Réseau de Petri obtenu à partir de la configuration l5c3

Nécessité d'une configuration transitoire

La deuxième partie du coût de la reconfiguration concerne la modification de l'état des produits pour qu'il soit admissible dans la configuration testée.

Cette information est obtenue de manière approchée en comparant la configuration actuelle avec la configuration testée. Si une ressource sur laquelle se termine une séquence d'opérations de l'ancienne configuration correspond à une ressource sur laquelle débute une séquence d'opérations de la nouvelle configuration, alors l'utilisation de ces deux séquences peut permettre de réaliser la transition sans passer par une configuration de transition.

Compilation des différents critères pour choisir une configuration

Les différents critères ont été retenus pour permettre le choix d'une nouvelle configuration. Dans le cas de l'application du train, la décision finale est à la charge de l'utilisateur mais les différentes configurations définies pour le train sont classées.

Ce classement se fait par rapport à la somme pondérée des résultats donnés par les différents critères proposés. Les coefficients attribués à chacun des critères permettent d'adapter le choix de la nouvelle configuration aux préférences de l'utilisateur.

Le deuxième panneau de l'interface graphique de gestion du train, représenté à la figure 6.16 permet le choix d'une configuration. Celui-ci propose un tableau présentant dans chaque colonne un résultat d'analyse. Les différents poids affectés aux critères sont paramétrables. Lors de l'appui sur le bouton "Update Mode", les critères concernant la reconfiguration sont calculés par rapport aux modes des ressources spécifiés dans le premier panneau ainsi que la configuration choisie. Les configurations sont alors triées conformément au résultat de la somme pondérée, la première d'entre elles est proposée.

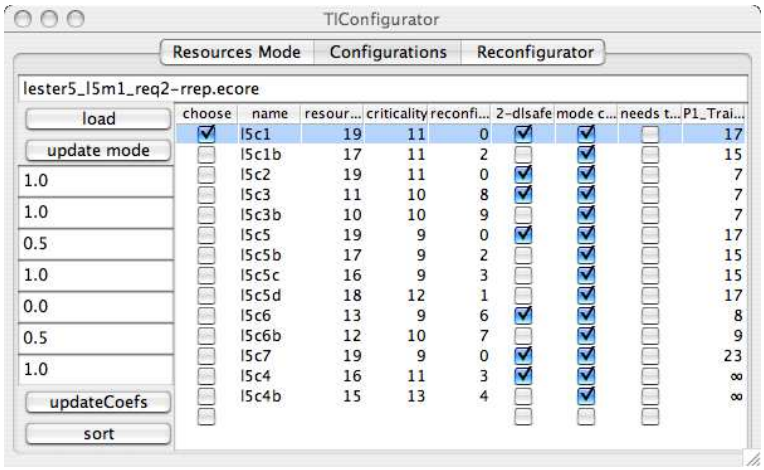


FIG. 6.16 – Panneau de sélection de la configuration

Conclusion

L'utilisation conjointe des différents critères définis dans cette section permet de choisir une configuration dans l'ensemble des configurations pré-établies. Le choix d'une configuration parmi celles qui sont proposées conduit à sa mise en place. Cette mise en place est décrite dans la section suivante.

6.4.3 Mise en place et utilisation d'une configuration

Une fois la configuration choisie, elle doit être mise en place sur le circuit. Cette mise en place est actuellement effectuée manuellement.

Le changement de configuration nécessite que les trains se trouvent dans une position admissible dans la nouvelle configuration. Deux cas peuvent alors se présenter suivant la nécessité ou non de passer par une configuration transitoire. Si la transition peut se faire sans configuration transitoire, les différents trains doivent être déplacés pour se trouver dans une position admissible dans la nouvelle configuration. Sinon, une configuration transitoire est mise en place pour initialiser les positions des trains.

L'envoi d'une nouvelle configuration (d'exploitation ou transitoire) à l'automate est réalisé par l'intermédiaire de l'interface graphique qui permet de choisir la nouvelle configuration et de la charger ensuite dans la mémoire de l'automate.

Une fois la nouvelle configuration mise en place, elle peut être exploitée. L'ensemble des séquences opératoires pouvant être utilisées par l'utilisateur est présenté, celui-ci peut alors choisir les traitements qu'il souhaite effectuer, réalisant les tâches incombant au module de pilotage.

6.4.4 Exemple de reconfiguration

L'exemple de reconfiguration considéré a été déployé avec succès sur le démonstrateur ferroviaire, il est illustré à la figure 6.17. Au début du scénario, le train circule conformément à la configuration `l5c5` qui définit comment le train utilise la grande boucle extérieure pour réaliser le traitement `Traitement2`.

Une défaillance apparaît alors sur le tronçon `T3` qui ne peut plus être utilisé. L'opérateur joue alors le rôle du diagnostic qui permet de mettre à jour le mode des ressources. Il utilise ensuite le programme pour choisir une nouvelle configuration. La configuration proposée lorsque

l'on interdit le passage par une configuration de transition (en fixant le poids de la métrique correspondante à 100 par exemple) est la configuration `l5c5b` qui est une version dégradée de `l5c5` dans laquelle la locomotive fait un demi-tour sur `T19` pour ne pas utiliser `T3`. L'opérateur demande alors la mise à jour de la commande conformément à la configuration ainsi que le mode des différentes ressources.

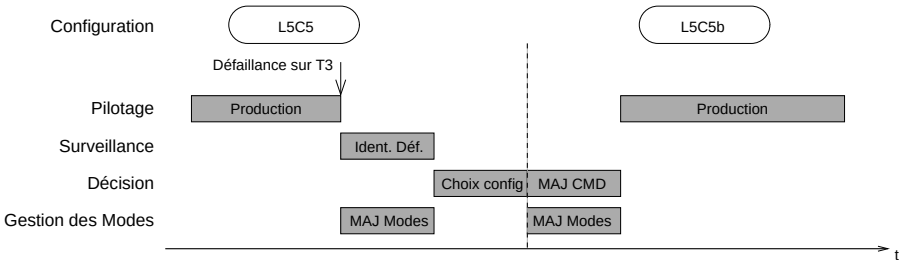


FIG. 6.17 – Scénario de reconfiguration du circuit

L'autorisation de recourir à une configuration de transition modifie le résultat du choix, en effet, la configuration `l5c3`, utilisant la boucle intérieure est plus performante et n'utilise pas `T3`, de plus, contrairement à `l5c5b`, `l5c3` est non-blocante avec deux locomotives. Le passage à la configuration `l5c3` est présentée à la figure 6.18.

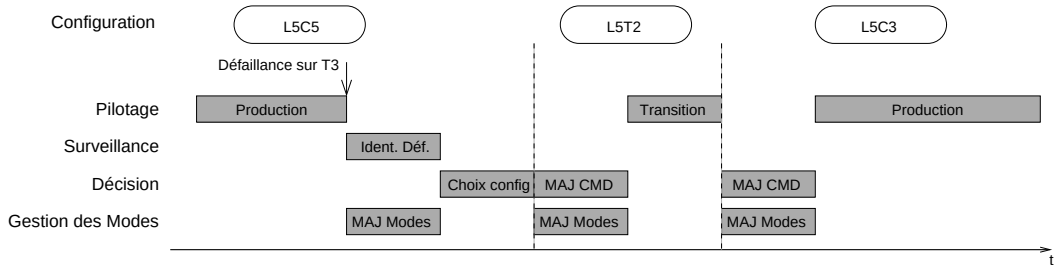


FIG. 6.18 – Scénario de reconfiguration du circuit avec configuration de transition

Une fois la reconfiguration terminée, le circuit ferroviaire peut continuer à fonctionner dans cette configuration.

6.4.5 Bilan concernant le fonctionnement

Cette section a présenté le fonctionnement du démonstrateur ferroviaire. Les concepts associés à la reconfiguration ont été, dans un premier temps, projetés sur l'application. Le bloc décisionnel permettant de choisir une configuration et utilisant différentes analyses a ensuite été présenté. Une description de la mise en place d'une nouvelle configuration a terminé cette section.

Les mécanismes de reconfiguration ont été étudiés sur le démonstrateur ferroviaire, l'utilisation de l'outil de décision permet de choisir, suite à une défaillance sur le circuit, une nouvelle configuration

6.5 Conclusion

Le langage de haut-niveau défini dans cette thèse a été utilisé sur un exemple complet : le démonstrateur ferroviaire. A partir de la modélisation du circuit du train, il a ainsi été possible

d'évaluer les performances du circuit avant de générer automatiquement le code de contrôle/-commande associé et d'appliquer les principes de reconfiguration.

Une partie des capacités de la plate-forme mise en place dans ce chapitre ont été utilisées pour l'instant. En effet, en plus du code de contrôle/commande, il faudrait générer ou tout du moins spécifier les parties de l'architecture de commande que sont la surveillance et la supervision pour laquelle seuls les mécanismes permettant de gérer manuellement les reconfigurations ont été mis en place.

De plus, une sous-partie du système électronique permettant la gestion du train a été utilisée. Ce système peut lui aussi être considéré en tant que système reconfigurable. L'interaction entre celui-ci et le circuit devra être considérée afin de tirer encore plus parti des avantages de ce système reconfigurable.

Conclusion et perspectives

Les travaux présentés dans cette thèse ont permis de préciser les concepts de reconfiguration dans les domaines des systèmes électroniques et des systèmes automatisés de production.

Cette généralisation a été obtenue au travers d'un langage de description commun ainsi que d'analyses de différents critères sur les modèles des systèmes considérés. Ces analyses sont définies dans un cadre générique permettant de développer et d'ajouter des analyses pertinentes pour l'application considérée, compte tenu des objectifs assignés au système.

Un fonctionnement théorique des systèmes reconfigurables, reprenant les modèles développés pour les systèmes de production a ensuite été défini. Un flot a été proposé pour l'implantation des systèmes considérés.

Les diverses propositions ont été validées sur une plate-forme expérimentale.

Résultats obtenus

Un langage pour la description des systèmes reconfigurables

La pierre angulaire des travaux présentés dans ce mémoire est constituée de la proposition d'un langage de description des systèmes reconfigurables, le langage DeSyRe.

Ce langage est principalement destiné à la description de systèmes reconfigurables manipulant des flux (de produits ou de données). Il s'appuie sur une séparation claire entre l'architecture du système et ses différentes configurations. Cette distinction place la configuration au centre du fonctionnement du système reconfigurable et facilite son exploitation.

Les configurations ont ensuite été situées par rapport aux différentes fonctions des architectures de contrôle/commande. Elles interviennent entre la planification qui fixe les objectifs au système pendant son fonctionnement et le pilotage qui constitue un deuxième niveau d'ordonnancement, chargé de remplir ces mêmes objectifs au sein de la configuration choisie.

Un cadre pour l'analyse et l'implantation des systèmes reconfigurables

Le cadre défini a permis d'utiliser l'ingénierie dirigée par les modèles dans les activités d'analyses des systèmes reconfigurables. Diverses analyses ont ainsi été proposées et ont été appliquées sur des exemples dans les domaines des systèmes de production et des systèmes électroniques. L'application des analyses proposées sur les configurations permet en outre de les classer et de choisir une configuration adéquate au cours du processus de reconfiguration.

Ce cadre a ensuite été étendu aux activités d'implantation. Le code de contrôle/commande associé à un système reconfigurable a été directement obtenu par application de transformations de modèles.

L'apport de l'ingénierie dirigée par les modèles pour l'étude des systèmes

Nous considérons que l'ingénierie dirigée par les modèles, bien que dans sa jeunesse, constitue un outil majeur pour l'étude et l'implémentation des systèmes.

Au cours de cette étude, l'ingénierie dirigée par les modèles a permis de mettre rapidement à l'épreuve les idées qui sont nées et de les développer. L'utilisation de ces outils a commencé par la modélisation des systèmes considérés. Elle s'est poursuivie lorsque nous avons souhaité analyser ces systèmes pour en ressortir des propriétés. Ces outils se sont ensuite révélés très efficaces pour l'implantation de la commande d'un système reconfigurable. Cette technologie s'est aussi révélée adéquate pour une utilisation en ligne, pour la gestion des différentes configurations du train et sa reconfiguration. Ainsi, toutes les activités développées au cours de la thèse ont pu bénéficier des apports de cet outil.

Le démonstrateur ferroviaire

L'application développée au cours de cette thèse a permis de valider toute l'approche haut niveau et notamment le fonctionnement d'un système reconfigurable. Le cadre d'implantation défini dans le chapitre 5 a été utilisé avec succès pour générer le code de commande de deux manières différentes. Un outil d'aide à la décision a ensuite été mis en place pour gérer le fonctionnement du train et aider à la reconfiguration. Des scénarios de reconfiguration ont alors été proposés.

Perspectives

Les travaux ouvrent de nombreuses perspectives intéressantes. Celles-ci concernent l'extension de la représentation, la construction des configuration, la place de la configuration dans l'architecture de contrôle/commande, l'implantation du système et l'évolution du démonstrateur. Ces perspectives sont discutées dans les paragraphes suivants.

Vers une représentation mixte pour les systèmes mécatroniques

Le langage DeSyRe, présenté dans ce mémoire a été à la fois utilisé pour décrire les procédés de systèmes automatisés de production et les systèmes électroniques. Cette capacité le rend apte à la modélisation des systèmes complexes, constitués à la fois d'un procédé reconfigurable et d'une électronique de commande associée reconfigurable. Le même langage pourrait alors être utilisé pour décrire les deux sous-systèmes ainsi identifiés.

Cette possibilité ouvre des perspectives intéressantes concernant l'évolution et la reconfiguration conjointe du procédé du système et de son électronique de commande (réseaux de communications, capteurs, actionneurs). Les notions de partitionnement utilisé dans le domaine du co-design logiciel/matériel pourraient alors être appliquées à ces systèmes mixtes et serviraient de base à la migration de code. Ainsi, dans ces systèmes, la reconfiguration pourrait être exploitée conjointement sur la partie opérative et la partie commande.

Construction de configurations

La construction des configurations à partir de l'architecture du système, pour une QdS fixée, n'a pas été abordée au cours de la thèse et constitue une perspective indispensable pour compléter la démarche proposée.

Cette construction doit permettre, à partir des objectifs fixés sur l'architecture logique et de contraintes sur l'architecture physique (état du système), d'obtenir une configuration qui pourra ensuite être déployée à l'aide du cadre d'implantation. Elle pourrait par exemple se baser sur des techniques d'ordonnancement, d'optimisation multi-critères ou d'aide à la décision. Cette construction pourra reposer sur les critères définis. Ces derniers sont d'ailleurs à perfectionner et d'autres critères pourraient aussi être définis.

Configuration et état du système

Le lien entre la configuration et l'état du système a été présenté de manière simplifiée. Deux composantes ont été considérées pour représenter l'état du système : le mode des ressources et l'état des produits dans le système.

Au niveau du lien entre le mode des ressources et les configurations, seule la cohérence entre le mode des ressources et la configuration ainsi que le coût de la configuration calculé à partir du mode d'arrêt ont été considérés. La prise en compte des autres modes, pour évaluer par exemple le coût de la réparation d'une ressource pourrait avantageusement compléter les travaux présentés dans ce mémoire. Une autre évolution à considérer serait la prise en compte de modes spécifiés au niveau des opérations des ressources.

Concernant l'état des produits, seule l'admissibilité d'un état des produits dans une configuration et la nécessité de passer par une configuration transitoire ont été développées. Une étude plus poussée de la transition de l'état des produits pour se conformer à la nouvelle configuration doit être menée. De plus, la connaissance de l'état des produits est nécessaire et nécessite l'étude de la fonction surveillance.

Configuration et ordonnancement

Nous avons proposé deux niveaux d'ordonnancement dans le cadre du fonctionnement d'un système reconfigurable.

Le premier, appelé *planification* vise à satisfaire les objectifs du système à long terme et conditionne le choix de nouvelles configurations.

Le second niveau d'ordonnancement est constitué par le *pilotage* qui interagit avec la commande en fonction de la configuration courante et des objectifs fixés. Le pilotage permet donc de lever les indéterminismes sur les choix qui subsistent après la mise en place d'une configuration donnée.

L'étude de la planification et du pilotage devra être approfondie pour aboutir à un système reconfigurable parfaitement autonome. Un lien très fort existe en effet entre ces deux fonctions du contrôle/commande et le choix ou la construction d'une configuration d'une part, et la stratégie d'ordonnancement d'autre part.

Extension du flot d'implantation

Les travaux pré-existants au sein du LESTER concernant les systèmes de production [Mouchard, 2002] ont développé la notion de composant sur étagère et introduit un modèle de composant générique, constitué de plusieurs vues reprenant les fonctions de l'architecture de contrôle/commande théorique. Les composants ont alors été utilisés dans une approche ascendante.

Cette notion de composant de contrôle/commande n'a pas été reprise dans cette thèse. Il semble néanmoins intéressant d'étudier la relation entre le modèle de description et les composants de contrôle/commande, ces derniers pouvant servir de base à une démarche d'implantation

solide, particulièrement dans le cas du découpage de l'application par rapport à l'architecture physique. L'obtention de la représentation de haut niveau à partir des composants de contrôle commande peut aussi être envisagée. Cette articulation entre les deux approches permettrait de bénéficier d'une approche de conception pour la description de haut niveau. Elle permettrait aussi une analyse des composants de contrôle/commande du point de vue de la reconfiguration.

L'autre découpage considéré, selon l'axe logique du système n'a été que partiellement étudié. Nous pensons que la poursuite de l'étude de ce découpage aboutirait à la mise en place d'une implémentation par agents. Le découpage mixte (physique et logique) pourrait conduire à l'adoption d'une architecture constituée de composants contrôlant les éléments du procédé et d'agents suivant la réalisation des différents produits sur le système.

Le flot de génération de code n'a actuellement été utilisé que sur un exemple. L'étude d'autres exemples, notamment en électronique, permettra de généraliser l'approche.

Evolution du démonstrateur

L'application démonstrateur ferroviaire développée au cours des travaux de thèse offre de nombreuses perspectives d'évolution.

Actuellement, seuls les programmes associés à la commande sont générés, mais la génération de programmes de surveillance peut aussi être envisagée, en utilisant le cadre défini pour l'implantation. L'utilisation de l'extension de celui-ci à l'utilisation des composants de contrôle/commande permettra alors d'apporter une solution plus ouverte pour la génération de code.

Une partie des fonctions du contrôle/commande n'a pas été automatisée comme la surveillance, le pilotage, la planification ou la gestion des modes. Un fonctionnement totalement automatisé du train nécessite le développement de ces fonctions qui font intervenir à la fois le programme de décision sur le PC et l'automate programmable.

L'architecture électronique proposée permet d'offrir la possibilité d'étendre la reconfiguration du système à sa partie électronique. L'étude de l'évolution conjointe des deux sous-systèmes peut alors être considérée.

Bibliographie

[Achour and Rezg, 2004] Achour, Z. and Rezg, N. (2004). Commande par supervision des systèmes à évènements discrets basée sur l'utilisation du grafcet et la théorie des régions. *APII-JESA Journal Européen des Systèmes Automatisés*.

[Altera, 2006] Altera (2006). Page d'informations sur les FPGA Stratix II. `http://www.altera.com/products/devices/stratix2/st2-index.jsp`.

[André, 1996] André, C. (1996). Representation and Analysis of Reactive Behaviors : A Synchronous Approach. In *Proceedings of the IMACS-IEEE International Conference on Computational Engineering in Systems Applications, CESA'96*, pages 19–29.

[Auguin et al., 2003] Auguin, M., Chehida, K. B., Diguet, J.-P., Fornari, X., Fouilliart, A.-M., Gaamrat, C., Gogniat, G., Kajfasz, P., and Moullec, Y. L. (2003). Partitioning and CoDesign tools and methodology for Reconfigurable Computing : The EPICURE philosophy. In *Proceedings of the Third International Workshop on Systems, Architecture, Modeling and Simulation SAMOS'03*, pages 46–51, Samos, Grèce.

[Beierlein et al., 2004] Beierlein, T., Fröhlich, D., and Steinbach, B. (2004). Model-Driven Compilation of UML-Models for Reconfigurable Architectures. In *Second RTAS Workshop on Model-Driven Embedded Systems MoDES'04*, Toronto, Ontario, Canada.

[Belabbas and Berruet, 2004] Belabbas, A. and Berruet, P. (2004). FMS reconfiguration based on Petri nets models. In *Proceedings of the IEEE International Conference on System Man and Cybernetics SMC'04*, The Hague, Netherlands.

[Berruet, 1998] Berruet, P. (1998). *Contribution au recouvrement des Systèmes Flexibles de Production Manufacturière : Analyse de la tolérance et reconfiguration*. PhD thesis, Ecole Centrale de Lille & Université des Sciences et Technologies de Lille, Lille.

[Berruet et al., 2003] Berruet, P., Coudert, T., and Philippe, J.-L. (2003). Integration of Dependability Aspects in Transitic Systems. In *Proceedings of the IMACS-IEEE International Conference on Computational Engineering in Systems Applications, CESA'03*.

[Berruet et al., 2002] Berruet, P., Craye, E., and Toguyeni, A.-K.-A. (2002). *Maîtrise des Risques et Sûreté de Fonctionnement des Systèmes de Production, (Coordonné par E. Niel et E. Craye)*, chapter 6, Modèles et Algorithmes pour la surveillance supervision, pages 145–196. Section Productique du Traité IC2 (B. DUBUISSON). Hermes.

[Berruet and Kindler, 2006] Berruet, P. and Kindler, E. (2006). Nested Anticipation in Design of Reconfigurable Manufacturing Systems. *International Journal of Computing Anticipory Systems*.

[Berruet et al., 1997] Berruet, P., Toguyeni, A.-K.-A., Elkhattabi, S., and Craye, E. (1997). Characterisation of tolerance in FMS. In *Proceedings of the 15^{th} IMACS World Congress*, volume 5, pages 409–414.

[Berthomieu et al., 2003] Berthomieu, B., Ribet, P.-., and Vernadat, F. (2003). L'outil TINA – Construction d'espaces d'états abstraits pour les réseaux de Petri et réseaux Temporels. In Hermes, editor, *Modélisation des Systèmes Réactifs, MSR'2003*.

[Bois, 1991] Bois, S. (1991). *Intégration de la gestion des modes de marche dans le pilotage d'un système automatisé de production*. PhD thesis, Université des Sciences et Technologies de Lille, Lille.

[Bézivin, 2005] Bézivin, J. (2005). On the Unification Power of Models. *Software and System Modeling (SoSym)*, 4(2) :171–188.

[Bézivin and Jouault, 2005] Bézivin, J. and Jouault, F. (2005). Using atl for checking models. In *Proceedings of the International Workshop on Graph and Model Transformation (GraMoT)*, Tallinn, Estonia.

[Bézivin et al., 2005] Bézivin, J., Jouault, F., Rosenthal, P., and Valduriez, P. (2005). Modeling in the large and modeling in the small. In Uwe Aßmann, Mehmet Aksit, A. R., editor, *Proceedings of the European MDA Workshops : Foundations and Applications, MDAFA 2003 and MDAFA 2004, LNCS 3599*, pages 33–46. Springer-Verlag GmbH.

[Calvez, 1990] Calvez, J.-P. (1990). *Specification et conception des systemes : une methodologie*. Masson.

[Cotting and Burken, 2001] Cotting, C. and Burken, J. J. (2001). Reconfigurable Control Design for the Full X-33 Flight Envelope. In *AIAA Guidance, Navigation and Control Conference*, Montreal, Quebec, Canada.

[Coussy, 2003] Coussy, P. (2003). *Synthèse d'Interface de Communication pour les Composants Virtuels*. PhD thesis, Université de Bretagne Sud, Lorient.

[Craye, 1994] Craye, E. (1994). *Contribution au Contrôle/Commande de Systèmes Flexibles de Production Manufacturière*. Habilitation à diriger les recherches, Université des Sciences et Technologies de Lille, Lille.

[Crow and Rushby, 1991] Crow, J. and Rushby, J. (1991). Model-Based Reconfiguration : Towards an Integration with Diagnosis. In *Proceedings, AAAI-91*, volume 2, Anaheim, Canada.

[Dangoumau, 2000] Dangoumau, N. (2000). *Contribution à la Gestion des Modes des Systèmes Automatisés deProduction*. PhD thesis, Ecole centrale de Lille & Université des Sciences et Technologies de Lille, Lille.

[Deschamps et al., 2006] Deschamps, E., Henry, S., and Zamaï, E. (2006). Synchronization of Operating Part Model in Failure Context. In *Proceedings of the International Conference on Computational Intellicgence for Modelling Control and Automation CIMA'06*, Sydney, Australia.

[Drolet et al., 1996] Drolet, J., Abdulnour, G., and Rheault, M. (1996). The Cellular Manufacturing Evolution. *Computers and Industrial Engineering*, 31(1/2) :139–142.

[Durand, 1998] Durand, E. (1998). *Description et vérification d'architectures d'applications temps réel : CLARA et les réseaux de Petri temporels*. PhD thesis, Ecole Centrale de Nantes & Université de Nantes, Nantes.

[Esterel Technologies, 2006] Esterel Technologies (2006). Page d'accueil de Esterel Technologies.
`http://www.esterel-technologies.com/`.

[Faucou et al., 2001] Faucou, S., Déplanche, A.-M., and Trinquet, Y. (2001). Operative Architecture Design and Modelling for the Validation of Real-Time Applications. In *Emerging Technologies and Factory Automation ETFA'01*, volume 2, pages 649–652, Antibes, France.

[Feldmann and Rexford, 2000] Feldmann, A. and Rexford, J. (2000). IP network configuration for traffic engineering. Technical Report 000526-02, AT&T Labs – Research.

[Gouyon, 2004] Gouyon, D. (2004). *Contrôle par le Produit des Systèmes d'Exécution de la Production : Apport des Techniques de synthèse*. PhD thesis, Université Henri Poincaré, Nancy.

[Gueganno and Duhaut, 2005] Gueganno, C. and Duhaut, D. (2005). A Versatile and Open Software/Hardware Architecture for Mechatronic Systems. In *ISIE IEEE International Symposium on Industrial Electronics*, Dubrovnik, Croatia.

[Hamani, 2005] Hamani, N. (2005). *Contribution à la modélisation et à la vérification pour la gestion des modes des systèmes automatisés de production*. PhD thesis, Ecole Centrale de Lille & Université des Sciences et Technologies de Lille, Lille.

[Hannan, 1996] Hannan, R. (1996). *Computer Integrated Manufacturing from concepts tot realisations*. Addison-wesley.

[Henry, 2005] Henry, S. (2005). *Synthèse de lois de commande pour la configuration et la reconfiguration des systèmes industriels complexes*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble.

[Hoffmann and Meyr, 2002] Hoffmann, A. and Meyr, H. (2002). *Architecture Exploration for Embedded Processors with LISA*. Kluwer.

[IEC, 2003] IEC (2003). *Automates programmables. Partie 3 : Langages de programmation*. IEC 61131-3.

[IEEE/DASC/VASG, 2002] IEEE/DASC/VASG (2002). *IEEE Standard VHDL Language Reference Manual*. 10762000.

[ITRS, 2005] ITRS (2005). International technology roadmap for semicoductor 2005 edition.

[Jin and Nahrstedt, 2004] Jin, J. and Nahrstedt, K. (2004). QoS Specification Languages for Distributed Multimedia Applications : A Survey and Taxonomy. *IEEE Multimedia*, 11(3) :74–87.

[Jouault and Bézivin, 2006] Jouault, F. and Bézivin, J. (2006). KM3 : a DSL for Metamodel Specification. In *Proceedings of 8th IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems, LNCS 4037*, pages 171–185, Bologna, Italy.

[Jouault et al., 2006a] Jouault, F., Bézivin, J., Consel, C., Kurtev, I., and Latry, F. (2006a). Building DSLs with AMMA/ATL, a Case Study on SPL and CPL Telephony Languages. In *Proceedings of the 1st ECOOP Workshop on Domain-Specific Program Development (DSPD), July 3rd, Nantes, France*.

[Jouault et al., 2006b] Jouault, F., Bézivin, J., and Kurtev, I. (2006b). TCS : a DSL for the Specification of Textual Concrete Syntaxes in Model Engineering. In *GPCE'06 : Proceedings of the fifth international conference on Generative programming and Component Engineering*.

[Jouault and Kurtev, 2005] Jouault, F. and Kurtev, I. (2005). Transforming models with atl. In *Proceedings of the Model Transformations in Practice Workshop at MoDELS 2005*, Montego Bay, Jamaica.

[Kamimura et al., 2001] Kamimura, A., Murata, S., Yoshida, E., Kurokawa, H., Tomita, K., and Kokaji, S. (2001). Self-Reconfigurable Modular Robot. In *Proceedings of 2001 IEEE/RSJ International Conference on Intelligent Robots an Systems (IROS2001)*, pages 606–612.

[Kernighan and Ritchie, 1990] Kernighan, B. W. and Ritchie, D. M. (1990). *Le Langage C, Norme ANSI*. Masson, Prentice Hall.

[Khatab, 2000] Khatab, A. (2000). *Contrôle et contrôle Stabilisant des Systèmes à Evénements Discrets Temporels : Application au Recouvrement des Défaillances des Systèmes de Production*. PhD thesis, Institut National des Sciences Appliquées de Lyon, Lyon.

[Kotay and Rus, 1999] Kotay, K. and Rus, D. (1999). Locomotion versatility through self-reconfiguration. *Robotics and Autonomous Systems*, 26(2-3) :217–232.

[Kramer, 1990] Kramer, J. (1990). Configuration Programming – a framework for the development of distributable systems. In *Proc. of the IEEE Int. Conference on Computer Systems and Software Engineering CompEuro90*.

[Lamotte et al., 2005a] Lamotte, F., Berruet, P., and Philippe, J.-L. (2005a). A model for the reconfiguration of manufacturing systems. In *Proceedings of the 16th IFAC World Congress*, Prague.

[Lamotte et al., 2005b] Lamotte, F., Berruet, P., and Philippe, J.-L. (2005b). Using model engineering for the criticality analysis of reconfigurable manufacturing systems architectures. *International Journal on Manufacturing Technology and Management, Special Issue on Manufacturing Under Changing Environment*. A paraître.

[Lamotte et al., 2005c] Lamotte, F., Berruet, P., and Philippe, J.-L. (2005c). Using model transformation for the analysis of the architecture of a reconfigurable system. In *17th IMACS World Congress*, Paris.

[Lamotte et al., 2006] Lamotte, F., Berruet, P., and Philippe, J.-L. (2006). Evaluation of Reconfigurable Manufacturing Systems configurations using tolerance criteria. In *Proceedings of the IEEE IECON Conference*, Paris.

[le Moullec, 2003] le Moullec, Y. (2003). *Aide à la conception de systèmes sur puces hétérogènes par l'exploration paramétrable des solutions au niveau système*. PhD thesis, Université de Bretagne Sud, Lorient.

[Martin et al., 1993] Martin, E., Sentineys, O., Dubois, H., and Philippe, J. (1993). Gaut : An architectural synthesis tool for dedicated signal processors. In *EURO-DAC*.

[Mathworks, 2006] Mathworks (2006). Page d'accueil de mathworks. `http://www.mathworks.com`.

[Medvidovic and Taylor, 2000] Medvidovic, N. and Taylor, R. N. (2000). A Classification and Comparison Framework for Software Architecture Description Languages. *IEEE Transactions on Software Engineering*, 26(1) :70–93.

[Mehrabi et al., 2002] Mehrabi, M.-G., Ulsoy, A.-G., Koren, Y., and heytler, P. (2002). Trends and perspectives in flexible and reconfigurable manufacturing systems. *Journal of Intelligent Manufacturing*, 13 :135–146.

[Mendez, 2002] Mendez, H. (2002). *Synthèse des lois de surveillance pour les procédés industriels complexes*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble.

[Mitchell et al., 1998] Mitchell, S., Naguib, H., Coulouris, G., and Kindberg, T. (1998). Dynamically Reconfiguring Multimedia Components : A Model-based Approach. In *8th ACM SIGOPS European Workshop*, Lisbon.

[Mouchard, 2002] Mouchard, J.-S. (2002). *Proposition d'une approche méthodique pour la conception des systèmes automatisés de production, application aux systèmes transitiques.* PhD thesis, Université de Bretagne Sud, Lorient.

[Moyne et al., 2004] Moyne, J., Korsakas, J., and Tilbury, D. (2004). Reconfigurable Factory Testbed (RFT) : A distributed testbed for Reconfigurable Manufacturing Systems. In *Proceedings of the 2004 Japan – USA Symposium on Flexible Automation*, Denver, Colorado.

[Murata, 1989] Murata, T. (1989). Petri nets : Properties, analysis and applications. *Proc. IEEE*, 77(4) :541–580.

[OMG, 2002] OMG (2002). OMG Meta Object Facility (MOF) Specification.

[OMG, 2003a] OMG (2003a). OMG Unified Modeling Language (UML) Specification.

[OMG, 2003b] OMG (2003b). Response to the UML 2.0 OCL RfP.

[Ramadge and Wonham, 1987] Ramadge, P. and Wonham, W. (1987). Supervisory control of a class of direct event process. *Journal of Control and Optimization*, 25(1).

[Rezg, 1996] Rezg, N. (1996). *Contribution à la sécurité opérationnelle des systèmes : Mises en œuvre d'une structure de surveillance basées sur les RdP Contrôlés*. PhD thesis, Institut National des Sciences Appliquées de Lyon, Lyon.

[Rouxel et al., 2005] Rouxel, S., Diguet, J.-P., and Gogniat, G. (2005). UML Framexork for PIM and PSM Verification of SDR Systems. In *Software Defined Radio Technical Conference*.

[Ryu et al., 2003] Ryu, K., Son, Y., and Jung, M. (2003). Modeling and specifications of dynamic agents in fractal manufacturing systems. *Computers in Industry*, 52 :161–182.

[Saad, 2003] Saad, S. (2003). The reconfiguration issues in manufacturing systems. *Journal of Materials Processing Technology*, 138(1,3) :277–283.

[SAE, 2004] SAE (2004). *Architecture Analysis & Design Language (Aadl)*. AS5506.

[Telemecanique, 2003a] Telemecanique (2003a). *Automates Modicon Premium, Processeurs, Manuel de Mise en œuvre*. TLX DM xxF fre.

[Telemecanique, 2003b] Telemecanique (2003b). *Description du logiciel PL7*. TLX DR PL7 xx fre.

[Toguyeni, 2001] Toguyeni, A. (2001). *Contribution à la tolérance aux fautes des systèmes flexibles de production manufacturière*. Habilitation à diriger les recherches, Université des Sciences et Technologies de Lille, Lille.

[Tsai and Sato, 2004] Tsai, T. and Sato, T. (2004). A UML model of agile production planning and control system. *Computers in Industry*, 53 :133–152.

[Ueda, 1992] Ueda, K. (1992). A concept for bionic manufacturing systems based on DNA-type information. In *Proceedings of the IFIP Eighth International Prolamat Conference*, pages 853–863, Tokyo.

[Valette, 2002] Valette, R. (2002). Les Réseaux de Petri.

[Welch and Purtilo, 1997] Welch, D. J. and Purtilo, J. M. (1997). Domain-Driven Reconfiguration in Collaborative Virtual Environments. Technical Report CS-TR-3772, University of Maryland.

[Wills and et al., 2001] Wills, L. and et al. (2001). An Open Platform For Reconfigurable Control. *IEEE Control Systems Magazine*, 21 :49–64.

[Xilinx, 2006] Xilinx (2006). Page d'accueil de la société Xilinx. `http://www.xilinx.com`.

[Xu et al., 2002] Xu, Y., Brennan, R.-W., Zhang, X., and Norrie, H. (2002). A Reconfigurable Concurrent Function Block Model and its Implementation in Real-Time Java. *Journal of Integrated Computer-Aided Engineering*, 9 :263–279.

[Zamaï, 1997] Zamaï, E. (1997). *Architecture de Surveillance-Commande pour les Systèmes à Evénements Discrets Complexes*. PhD thesis, Université Paul Sabatier, Toulouse.

Quatrième partie

Annexes

Annexe A

Exemples décrits avec le langage DeSyRe

A.1 Exemple des deux machines

A.1.1 Architecture

Listing A.1 – Description textuelle de l'architecture de l'exemple des Deux Machines

```
1  Architecture DeuxMachines ;
2
3  logical architecture {
4    function TF; -- Fonction de transfert generique
5    function F2;
6    function F3;
7    function F1;
8
9    product P2;
10   product P1;
11   product P3;
12
13   sequence P1_S2 [P1 -> P2] : F2, F1, F3;
14   sequence P1_S1 [P1 -> P2, P3] : F1, F2, F3;
15 }
16
17 physical architecture {
18   resource IN {
19     port IN_P : inout;
20   }
21
22   resource M2 {
23     port M2_in_out : inout;
24   }
25
26   resource OUT {
27     port OUT_P : inout;
28   }
29
30   resource M1 {
31     port M1_in_out : inout;
32   }
33
34   transporter Cv {
35     realizes M1M2;
36   }
37
38   transporter R {
39     realizes M2OUT, M1M2, INM1, M1OUT, INM2, M2M1;
40   }
41
42   connection M1M2 (M1.M1_in_out,M2.M2_in_out);
43   connection INM2 (IN.IN_P,M2.M2_in_out);
```

```
44    connection M2OUT (M2.M2_in_out,OUT.OUT_P);
45    connection INM1 (IN.IN_P,M1.M1_in_out);
46    connection M1OUT (M1.M1_in_out,OUT.OUT_P);
47    connection M2M1 (M2.M2_in_out,M1.M1_in_out);
48  }
49
50  operation O1(F1,M1);
51  operation O2(F1,M2);
52  operation O3(F2,M1);
53  operation O4(F3,M2);
54  stocking ST1(P1,IN);
55  stocking ST2(P2,OUT);
56  stocking ST3(P3, OUT);
```

A.1.2 Une configuration

Listing A.2 – Représentation textuelle d'une configuration pour l'exemple des deux machines

```
1   configuration normal {
2     logical configuration {
3       instance IF3 of F3;
4       instance IF1 of F1;
5       instance ITF of TF;
6       instance IF2 of F2;
7     }
8
9     physical configuration {
10      transfer sequence TS_INM1 {
11        transfer INM1 using R;
12      }
13      transfer sequence TS_M2OUT {
14        transfer M2OUT using R;
15      }
16      transfer sequence TS_M1M2 {
17        transfer M1M2 using Cv;
18      }
19    }
20
21    transfer operation TO1 : ITF using TS_INM1;
22    transfer operation TO2 : ITF using TS_M1M2;
23    transfer operation TO3 : ITF using TS_M2OUT;
24    stationary operation SO1 : IF1 using O1;
25    stationary operation SO2 : IF2 using O3;
26    stationary operation SO3 : IF3 using O4;
27
28    operation sequence OS1 realize P1_S1 : TO1, SO1, SO2, TO2, SO3, TO3;
29  }
```

A.2 Convoyeur

A.2.1 Architecture

Listing A.3 – Représentation textuelle de l'architecture du convoyeur

```
1   Architecture Convoyeur;
2
3   logical architecture {
4     function F2;
5     function F1;
6     function TF;
7     function F3;
8
9     product P0;
```

```
10    product P1;
11    product P2;
12    product P3;
13
14    sequence S1[P0->P1] : F1, F2, F3;
15    sequence S2[P0->P2] : F1, F2;
16    sequence S3[P0->P3] : F1, F3;
17  }
18
19  physical architecture {
20    resource M3[510, 770] {
21      port M3_p[510, 990] : inout;
22    }
23
24    resource Z2[2190, 990] {
25      port Z2_in[2230, 850] : inout;
26      port Z2_out[2250, 1250] : inout;
27      port Z2_p[2450, 1010] : inout;
28    }
29
30    resource Z5[1890, 990] {
31      port Z5_out[1950, 1130] : inout;
32      port Z5_in[1950, 890] : inout;
33    }
34
35    resource OUT[1930, 390] {
36      port OUT_p[1810, 450] : inout;
37    }
38
39    resource Z4[790, 990] {
40      port Z4_out[850, 850] : inout;
41      port Z4_in[850, 1230] : inout;
42      port Z4_p[690, 990] : inout;
43    }
44
45    resource IN[1030, 410] {
46      port IN_p[1290, 450] : inout;
47    }
48
49    resource M2[1250, 1510] {
50      port M2_p[1510, 1530] : inout;
51    }
52
53    resource Z1[1490, 770] {
54      port Z1_out[1750, 770] : inout;
55      port Z1_p[1570, 690] : inout;
56      port Z1_in[1370, 770] : inout;
57    }
58
59    resource M1[2510, 770] {
60      port M1_p[2630, 990] : inout;
61    }
62
63    resource Z3[1490, 1290] {
64      port Z3_out[1390, 1290] : inout;
65      port Z3_in[1750, 1290] : inout;
66      port Z3_p[1630, 1430] : inout;
67    }
68
69    resource Z6[1110, 990] {
70      port Z6_out[1190, 890] : inout;
71      port Z6_in[1190, 1110] : inout;
72    }
73
74    transporter Cv[1490, 1110] {
75        realizes Z3Z4, Z6Z1, Z1Z2, Z4Z1, Z3Z6, Z2Z3, Z1Z5, Z5Z3;
76    }
77
78    transporter R13[510, 1170] {
79        realizes Z4M3, M3Z4;
```

```
80    }
81
82    transporter R12[1750, 1510] {
83        realizes M2Z3, Z3M2;
84    }
85
86    transporter R11[2490, 1210] {
87        realizes M1Z2, Z2M1;
88    }
89
90    transporter R14[1490, 410] {
91        realizes INZ2, Z4OUT, INZ4, Z2OUT, Z1Z2, Z4Z1, INZ1, Z2Z1, Z1Z4,
              Z1OUT;
92    }
93
94    connection M2Z3[1630, 1530] (M2.M2_p, Z3.Z3_p);
95    connection Z1Z5[1850, 830] (Z1.Z1_out, Z5.Z5_in);
96    connection INZ2[2150, 630] (IN.IN_p, Z2.Z2_p);
97    connection Z2OUT[2250, 590] (Z2.Z2_p, OUT.OUT_p);
98    connection Z6Z1[1290, 830] (Z6.Z6_out, Z1.Z1_in);
99    connection Z4Z1[1050, 770] (Z4.Z4_out, Z1.Z1_in);
100   connection Z1OUT[1750, 630] (Z1.Z1_p, OUT.OUT_p);
101   connection Z2Z1[1730, 1090] (Z2.Z2_out, Z1.Z1_in);
102   connection Z2Z3[2050, 1290] (Z2.Z2_out, Z3.Z3_in);
103   connection Z5Z3[1870, 1210] (Z5.Z5_out, Z3.Z3_in);
104   connection Z2M1[2590, 1070] (Z2.Z2_p, M1.M1_p);
105   connection Z4M3[630, 890] (Z4.Z4_p, M3.M3_p);
106   connection INZ1[1390, 630] (IN.IN_p, Z1.Z1_p);
107   connection Z1Z2[2030, 770] (Z1.Z1_out, Z2.Z2_in);
108   connection Z4OUT[910, 650] (Z4.Z4_p, OUT.OUT_p);
109   connection Z3Z6[1290, 1190] (Z3.Z3_out, Z6.Z6_in);
110   connection Z1Z4[1390, 1110] (Z1.Z1_out, Z4.Z4_in);
111   connection M3Z4[590, 1030] (M3.M3_p, Z4.Z4_p);
112   connection Z3M2[1510, 1430] (Z3.Z3_p, M2.M2_p);
113   connection M1Z2[2530, 930] (M1.M1_p, Z2.Z2_p);
114   connection INZ4[810, 610] (IN.IN_p, Z4.Z4_p);
115   connection Z3Z4[1090, 1290] (Z3.Z3_out, Z4.Z4_in);
116 }
117
118 operation OpM1_F3(F3, M1);
119 operation OpM3_F2(F2, M3);
120 operation OpM2_F3(F3, M2);
121 operation OpM3_F3(F3, M3);
122 operation OpM2_F1(F1, M2);
123 operation OpM1_F1(F1, M1);
124
125 stocking OpIN_P0 (P0, IN);
126 stocking OpOUT_P1 (P1, OUT);
127 stocking OpOUT_P2 (P2, OUT);
128 stocking OpOUT_P3 (P3, OUT);
```

A.2.2 Configuration CV1

Listing A.4 – Représentation textuelle de la configuration CV1 du convoyeur

```
1  configuration CV1 {
2
3  logical configuration {
4    instance ITF of TF;
5    instance IF1 of F1;
6    instance IF2 of F2;
7    instance IF3 of F3;
8  }
9
10 physical configuration {
11   connector TS_INM2 {
12     transfer INZ1 using R14;
```

```
13      control from Z1.Z1_p to Z1.Z1_out;
14      transfer Z1Z5 using Cv;
15      control from Z5.Z5_in to Z5.Z5_out;
16      transfer Z5Z3 using Cv;
17      control from Z3.Z3_in to Z3.Z3_p;
18      transfer Z3M2 using R12;
19    }
20    connector TS_M2M3 {
21      transfer M2Z3 using R12;
22      control from Z3.Z3_p to Z3.Z3_out;
23      transfer Z3Z4 using Cv;
24      control from Z4.Z4_in to Z4.Z4_p;
25      transfer Z4M3 using R13;
26    }
27    connector TS_M3M1 {
28      transfer M3Z4 using R13;
29      control from Z4.Z4_p to Z4.Z4_out;
30      transfer Z4Z1 using Cv;
31      control from Z1.Z1_in to Z1.Z1_out;
32      transfer Z1Z2 using Cv;
33      control from Z2.Z2_in to Z2.Z2_p;
34      transfer Z2M1 using R11;
35    }
36    connector TS_M1OUT {
37      transfer M1Z2 using R11;
38      transfer Z2OUT using R14;
39    }
40  }
41
42  stationary operation S_F1 : IF1 using OpM2_F1;
43  stationary operation S_F2 : IF2 using OpM3_F2;
44  stationary operation S_F3 : IF3 using OpM1_F3;
45
46  transfer operation T_INM2 : ITF using TS_INM2;
47  transfer operation T_M2M3 : ITF using TS_M2M3;
48  transfer operation T_M3M1 : ITF using TS_M3M1;
49  transfer operation T_M1OUT : ITF using TS_M1OUT;
50
51  operation sequence OS_S1 realize S1 : T_INM2, S_F1, T_M2M3, S_F2,
        T_M3M1, S_F3, T_M1OUT;
52
53  }
```

A.2.3 Configuration CV2

Listing A.5 – Représentation textuelle de la configuration CV2 du convoyeur

```
1   configuration CV2 {
2   logical configuration {
3     instance IF1 of F1;
4     instance IF2 of F2;
5     instance IF3 of F3;
6     instance ITF of TF;
7   }
8
9   physical configuration {
10    connector TS_INM1 {
11      transfer INZ2 using R14;
12      transfer Z2M1 using R11;
13    }
14    connector TS_M1M3 {
15      transfer M1Z2 using R11;
16      transfer Z2Z1 using R14;
17      transfer Z1Z4 using R14;
18      transfer Z4M3 using R13;
19    }
20    connector TS_M3OUT {
```

```
21      transfer M3Z4 using R13;
22      transfer Z4OUT using R14;
23    }
24  }
25
26  stationary operation S_F1 : IF1 using OpM1_F1;
27  stationary operation S_F2 : IF2 using OpM3_F2;
28  stationary operation S_F3 : IF3 using OpM3_F3;
29
30  transfer operation T_INM1 : ITF using TS_INM1;
31  transfer operation T_M1M3 : ITF using TS_M1M3;
32  transfer operation T_M3OUT : ITF using TS_M3OUT;
33
34  operation sequence OS_S1 realize S1 : T_INM1, S_F1, T_M1M3, S_F2, S_F3
        , T_M3OUT;
35
36  }
```

A.3 Exemple électronique

A.3.1 Architecture

Listing A.6 – Représentation textuelle de l'architecture de l'exemple électronique

```
1   architecture tsi;
2
3   logical architecture {
4     function transfert;
5     function acquisition;
6     function traitement_grossier;
7     function traitement_fin;
8
9     product image_brute;
10    product image_traitee;
11    product image_annotee;
12    product alarme;
13
14    sequence choix_image [image_brute -> image_traitee] : acquisition,
          traitement_grossier;
15    sequence annotation [image_traitee -> image_annotee, alarme] :
          traitement_fin;
16  }
17
18  physical architecture {
19    resource IN {
20      port IN_p : inout;
21    }
22
23    resource OUT {
24      port OUT_p : inout;
25    }
26
27    resource B1[290, 830] {
28      port B1_out[570, 870] : inout;
29      port toC1[370, 710] : inout;
30    }
31
32    resource B4[2290, 830] {
33      port B4_in[2170, 870] : inout;
34      port B4_out : inout;
35      port toC4[2350, 710] : inout;
36    }
37
38    resource Carte2[970, 270] {
39      port C2_p[1050, 430] : inout;
40      port C2_d[830, 310] : inout;
```

```
41      port C2_c : inout;
42    }
43
44    resource Carte1[290, 270] {
45      port C1_p[370, 430] : inout;
46      port C1_c : inout;
47      port C1_d[570, 290] : inout;
48    }
49
50    resource B3[1590, 830] {
51      port toC3[1690, 710] : inout;
52      port B3_in[1490, 870] : inout;
53      port B3_out[1850, 870] : inout;
54    }
55
56    resource Carte4[2290, 270] {
57      port C4_p[2370, 430] : inout;
58    }
59
60    resource B2[950, 830] {
61      port toC2[1050, 710] : inout;
62      port B2_out[1210, 870] : inout;
63      port B2_in[830, 870] : inout;
64    }
65
66    resource Carte3[1630, 270] {
67      port C3_p[1690, 430] : inout;
68    }
69
70    transporter DLink[370, 50] {
71        realizes C1C2, C2C1;
72    }
73
74    transporter Bus[1270, 70] {
75        realizes B3B2, B1B2, B4B3, B3C3, B4C4, B2C2, C4B4, C1B1, C2B2,
              B2B3, B1C1, B2B1, B3B4, C3B3, B4OUT;
76    }
77
78    transporter CLink {
79       realizes INC1;
80    }
81
82    transporter OutLink {
83       realizes C2OUT;
84    }
85
86    connection B3B2[1350, 790] (B3.B3_in, B2.B2_out);
87    connection C3B3[1810, 570] (Carte3.C3_p, B3.toC3);
88    connection B3C3[1610, 570] (B3.toC3, Carte3.C3_p);
89    connection B4B3[2010, 790] (B4.B4_in, B3.B3_out);
90    connection C2B2[1170, 570] (Carte2.C2_p, B2.toC2);
91    connection C2C1[710, 210] (Carte2.C2_d, Carte1.C1_d);
92    connection B2B1[690, 950] (B2.B2_in, B1.B1_out);
93    connection C4B4[2470, 570] (Carte4.C4_p, B4.toC4);
94    connection B2B3[1350, 950] (B2.B2_out, B3.B3_in);
95    connection B3B4[2010, 970] (B3.B3_out, B4.B4_in);
96    connection C1C2[710, 370] (Carte1.C1_d, Carte2.C2_d);
97    connection B1B2[690, 790] (B1.B1_out, B2.B2_in);
98    connection B1C1[470, 570] (B1.toC1, Carte1.C1_p);
99    connection C1B1[270, 570] (Carte1.C1_p, B1.toC1);
100   connection B4C4[2250, 570] (B4.toC4, Carte4.C4_p);
101   connection B2C2[950, 570] (B2.toC2, Carte2.C2_p);
102   connection B4OUT (B4.B4_out, OUT.OUT_p);
103   connection INC1 (IN.IN_p, Carte1.C1_c);
104   connection C2OUT (Carte2.C2_c, OUT.OUT_p);
105 }
106
107 operation C1Acq (acquisition, Carte1);
108 operation C2trf (traitement_fin, Carte2);
109 operation C2trg (traitement_grossier, Carte2);
```

```
110 operation C4trf (traitement_fin, Carte4);
111 operation C4trg (traitement_grossier, Carte4);
```

A.3.2 Configuration Eon1

Listing A.7 – Représentation textuelle de la configuration Eon1

```
1  configuration Eon1 {
2  logical configuration{
3    instance I_transfert of transfert;
4    instance I_acquisition of acquisition;
5    instance I_traitement_grossier of traitement_grossier;
6    instance I_traitement_fin of traitement_fin;
7  }
8
9  physical configuration {
10   connector TS_INC1 {
11     transfer INC1 using CLink;
12   }
13   connector TS_C1C2 {
14     transfer C1C2 using DLink;
15   }
16   connector TS_C2C4 {
17     transfer C2B2 using Bus;
18     control from B2.toC2 to B2.B2_out;
19     transfer B2B3 using Bus;
20     control from B3.B3_in to B3.B3_out;
21     transfer B3B4 using Bus;
22     control from B4.B4_in to B4.toC4;
23     transfer B4C4 using Bus;
24   }
25   connector TS_C4C3 {
26     transfer C4B4 using Bus;
27     control from B4.toC4 to B4.B4_in;
28     transfer B4B3 using Bus;
29     control from B3.B3_out to B3.toC3;
30     transfer B3C3 using Bus;
31   }
32   connector TS_C3OUT {
33     transfer C3B3 using Bus;
34     control from B3.toC3 to B3.B3_out;
35     transfer B3B4 using Bus;
36     control from B4.B4_in to B4.B4_out;
37     transfer B4OUT using Bus;
38   }
39 }
40
41 stationary operation S_acquisition : I_acquisition using C1Acq;
42 stationary operation S_traitement_grossier : I_traitement_grossier
       using C2trg;
43 stationary operation S_traitement_fin : I_traitement_fin using C4trf;
44
45 transfer operation T_INC1 : I_transfert using TS_INC1;
46 transfer operation T_C1C2 : I_transfert using TS_C1C2;
47 transfer operation T_C2C4 : I_transfert using TS_C2C4;
48 transfer operation T_C4C3 : I_transfert using TS_C4C3;
49 transfer operation T_C3OUT : I_transfert using TS_C3OUT;
50
51 operation sequence OS_choix_image realize choix_image : S_acquisition,
       T_C1C2, S_traitement_grossier;
52 operation sequence OS_annotation realize annotation : T_C2C4,
       S_traitement_fin, T_C4C3;
53
54 }
```

A.3.3 Configuration Eon2

Listing A.8 – Représentation textuelle de la configuration Eon2

```
configuration Eon2 {
logical configuration{
  instance I_transfert of transfert;
  instance I_acquisition of acquisition;
  instance I_traitement_grossier of traitement_grossier;
}

physical configuration {
  connector TS_INC1 {
    transfer INC1 using CLink;
  }
  connector TS_C1C2 {
    transfer C1C2 using DLink;
  }
  connector TS_C2OUT {
    transfer C2OUT using OutLink;
  }
}

stationary operation S_acquisition : I_acquisition using C1Acq;
stationary operation S_traitement_grossier : I_traitement_grossier
   using C2trg;

transfer operation T_C1C2 : I_transfert using TS_C1C2;

operation sequence OS_choix_image realize choix_image :
        S_acquisition, T_C1C2, S_traitement_grossier;
}
```

Annexe B

Les configurations définies pour le circuit du train

B.1 Description des configurations

La table B.1 décrit succintement les différentes configurations utilisées sur le démonstrateur. Cet ensemble de configurations est constitué de configurations minimales (l5c3, l5c4, l5c5, l5c6, l5c7), de configurations composées (l5c1, l5c2), de configurations dérivées (postfixées d'une lettre) et de configurations de transition (l5t1, l5t2). Leurs noms ont été donnés de manière historique, l5c1 étant la première configuration à avoir été décrite. De nouvelles configurations sont apparues au fur et à mesure des besoins.

Nom	Description
l5c1	Traitement2 sur la grande boucle, Traitement1 sur la petite boucle
l5c1b	Dérivée de l5c1 n'utilisant ni `C1`, ni `T3`
l5c1c	Dérivée de l5c1b n'utilisant pas `A2`
l5c2	Traitement2 sur la boucle extérieure et sur la boucle intérieure en parallèle
l5c3	Traitement2 sur la boucle intérieure uniquement
l5c3b	Dérivée de l5c3 n'utilisant pas `A5`
l5c4	Traitement1 sur la petite boucle
l5c4b	Dérivée de l5c4 sans `A2`
l5c5	Traitement2 sur la grande boucle
l5c5b	Dérivée de l5c5 n'utilisant ni `C1`, ni `T3`
l5c5c	Dérivée de l5c5b n'utilisant pas `A2`
l5c5d	Dérivée de l5c5 n'utilisant pas `T11` (pour la sortie)
l5c6	Traitement2 autour de `A5`
l5c6b	Dérivée de l5c6 sans `T4`
l5c7	Traitement2 en utilisant les trois boucles
l5t1	Configuration de transition : passage de la boucle intérieure à la petite boucle
l5t2	Configuration de transition : passage de la petite boucle à la boucle intérieure

TAB. B.1 – Configurations définies pour le démonstrateur

B.2 Résultats d'analyse

Les résultats d'analyse des différentes configurations pour les critères ne dépendant pas de l'état du système sont donnés au tableau B.2.

config	resources	rcri	2-dlsafe	P1_Traitement2
l5c1	19	11	oui	17.0
l5c1b	17	11	non	15.0
l5c2	19	11	oui	7.0
l5c3	11	10	non	7.0
l5c3b	10	10	non	7.0
l5c4	16	11	oui	-1.0
l5c4b	15	13	non	-1.0
l5c5	19	9	oui	17.0
l5c5b	17	9	non	15.0
l5c5c	16	9	non	15.0
l5c6	13	9	oui	8.0
l5c6b	13	9	non	10.0
l5c7	19	9	oui	23.0

TAB. B.2 – Résultats des analyses sur les configurations

Annexe C

Fonctionnement des grafcet associés aux éléments du circuit

C.1 Organisation de la mémoire

La mémoire de l'automate a été organisée afin de prendre en charge les configurations et de permettre la reconfiguration à travers la connexion modbus TCP. L'organisation de la mémoire est décrite à la figure C.1.

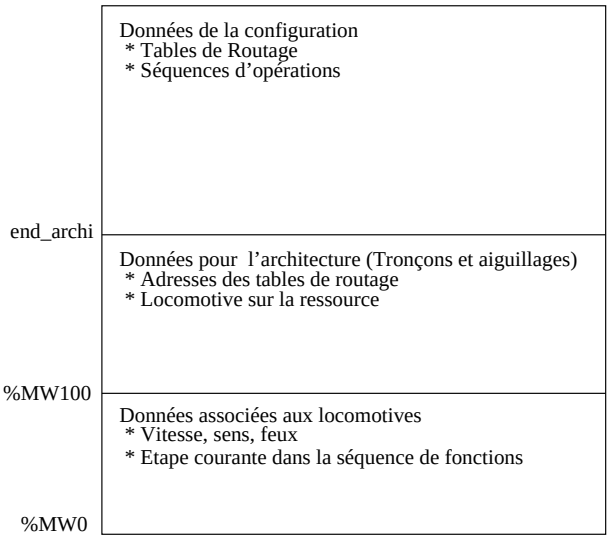


FIG. C.1 – Organisation de la mémoire de l'automate

La mémoire de l'automate est décomposée en trois blocs. Le premier bloc contient des informations relatives aux locomotives telles que leur vitesse, leur direction ou l'état de leurs feux (allumés ou éteints). L'adresse de l'étape courante dans la séquence d'opérations actuellement réalisée par chaque locomotive est aussi figurée, cette adresse est calculée par rapport à la fin de l'architecture, ce qui permet d'accéder au contenu de l'étape par une indirection. Les informations sont organisées par type de donnée, une entrée du fichier IOA indiquant l'adresse de l'élément pour le premier train. Cela permet d'accéder aisément aux informations pour chaque train, de manière dynamique en utilisant l'adressage indexé du langage ST (%MWx[y]).

Les données relatives aux différents composants sont stockées à partir de l'adresse 100. Les données associées à chaque composant sont : le numéro de la locomotive qui se trouve sur le tronçon, l'adresse de l'étape courante dans la séquence opératoire réalisée par cette locomotive,

ainsi que les adresses des tables de routage associées au composant. Le numéro de la locomotive présente sur le composant permet d'accéder à ses différents paramètres (vitesse, sens, feux). Par contre, Une copie de l'adresse de l'étape courante dans la séquence d'opérations parmi les données associées à la ressource sur laquelle est le train est nécessaire pour pouvoir accéder à la valeur qui y est stockée. En effet le langage ST n'accepte aucun calcul sur les adresses (une construction du type `%MW200[%MW10[1]]` n'est pas possible). Les adresses des tables de routage ne sont pas fixées pour l'architecture et sont mises à jour lors d'une reconfiguration car leur taille est dynamique.

La dernière partie de la mémoire est composée de la description des séquences d'opérations et des tables de routage. Cette partie est donc relative à la configuration. Une première liste donne l'adresse de chaque séquence d'opérations, l'adresse d'une séquence est donc accessible à l'adresse `%MWf[n]` où `f` est l'adresse de la fin de l'architecture et `n` le numéro de la séquence dont on souhaite avoir l'adresse. Les séquences sont ensuite décrites les unes à la suite des autres. Chaque étape est décrite dans un mot, une valeur positive correspond à une séquence de transfert (le numéro correspond à celui utilisé dans les tables de routage), une valeur négative correspond à la réalisation d'une opération de traitement, un 0 signifie que la fin de la séquence est atteinte. Les tables de routage sont enregistrées sous la forme de trois tables. La première contient les indices des ports d'entrée pour le composant, une valeur de -1 signifie "port indéterminé", la seconde table contient les ports de sortie et la troisième le numéro de la séquence de transfert réalisée, une valeur nulle de cette dernière indique la fin de la table.

C.2 Les blocs fonctionnels

Les blocs fonctionnels permettent d'interpréter les tables de routages associées aux composants, ils fournissent en sortie, les informations nécessaire à l'exécution des SFC. Les différents blocs sont présentés à la figure C.2. Les tables de routages apparaissent en entrée des blocs sous la forme de trois tableaux, le premier contient les différentes séquences (`sequences`), le second les ports d'entrée (`in_ports`) et le troisième les ports de sortie (`out_ports`). Les autres entrées du bloc diffèrent suivant le type d'élément associé. S'il s'agit d'un tronçon, deux entrées booléennes indiquent si le train est entré par le port A ou le port B, la valeur de ces entrées est déterminée à partir du numéro d'étape (1 ou 2 pour le port A et 7 ou 8 pour le port B), une dernière entrée du bloc l'informe de la séquence de transfert actuellement réalisée par le train présent sur le tronçon. Pour un aiguillage, une première entrée (`waiting`) indique si le composant est en attente ou non. Les autres entrées informent l'aiguillage de la séquence de transfert actuellement réalisé par les trains se trouvant sur les différents tronçons adjacents à l'aiguillage (`seq_x`).

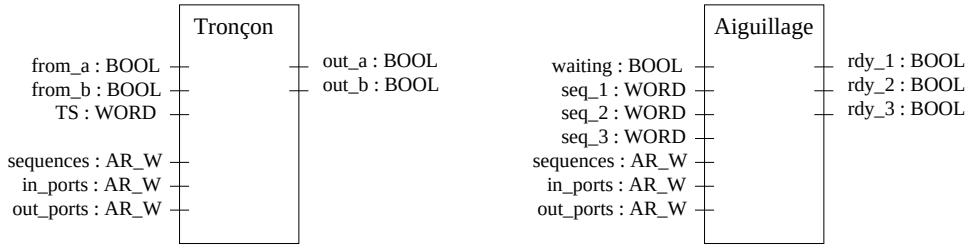


FIG. C.2 – Blocs fonctionnels associés aux composants

A partir des différentes entrées, les blocs fonctionnels déterminent des informations nécessaires à l'évolution des SFC. Dans le cas des tronçons, il s'agit de déterminer par quelle

extrémité le train devra sortir du tronçon. Dans le cas d'un aiguillage, les sorties indiquent à un seul tronçon qu'il doit se préparer à recevoir un train en provenance de l'aiguillage.

L'activation des blocs fonctionnels est réalisée pendant l'exécution du traitement préliminaire. Les différentes informations alimentant les blocs sont récupérés au niveau des étapes des SFC ou dans la mémoire de l'automate. Chaque sortie des blocs fonctionnels peut ensuite être lue par un programme écrit en ST. Ces programmes sont différents pour les tronçons et les aiguillages et sont proposés aux listings C.1 et C.2.

Listing C.1 – Code ST d'un DFB associé à un tronçon

```
1  IF ((From_a OR From_b) AND (Ts > 0)) THEN
2    Index := 0;
3    WHILE NOT (Seqs[Index] = 0
4      OR (Seqs[Index] = Ts AND From_a AND (Iports[Index] = 0 OR Iports[
           Index] = -1))
5      OR (Seqs[Index] = Ts AND From_b AND (Iports[Index] = 1 OR Iports[
           Index] = -1))) DO
6          INC Index;
7    END_WHILE;
8    IF Seqs[Index] = 0 THEN (*On n'a pas trouve la ligne correspondante
       *)
9      RESET Out_a;
10     RESET Out_b;
11     SET Ts_end;  (* Fin de la sequence *)
12   ELSE
13     IF Oports[Index] = 0 THEN
14       SET Out_a;
15       RESET Out_b;
16       RESET Ts_end;
17     ELSE
18       SET Out_b;
19       RESET Out_a;
20       RESET Ts_end;
21     END_IF;
22   END_IF;
23 ELSE
24   RESET Out_a;
25   RESET Out_b;
26   RESET Ts_end;
27 END_IF;
```

Listing C.2 – Code ST d'un DFB associé à un aiguillage à trois branches

```
1  RESET Rdy_0;
2  RESET Rdy_1;
3  RESET Rdy_2;
4
5  IF (NOT Waiting) THEN
6      Index := 0;
7      WHILE NOT (Seqs[Index] = 0 OR (Seqs[Index] = Seq_0 AND Iports[
           Index] = 0)) DO
8          INC Index;
9      END_WHILE;
10     IF Seqs[Index] <> 0 THEN
11       IF Oports[Index] = 0 THEN
12         SET Rdy_0;
13       ELSIF Oports[Index] = 1 THEN
14         SET Rdy_1;
15       ELSIF Oports[Index] = 2 THEN
16         SET Rdy_2;
17       END_IF;
18     END_IF;
19
20     Index := 0;
21     WHILE NOT (Seqs[Index] = 0 OR (Seqs[Index] = Seq_1 AND Iports[
           Index] = 1)) DO
22         INC Index;
```

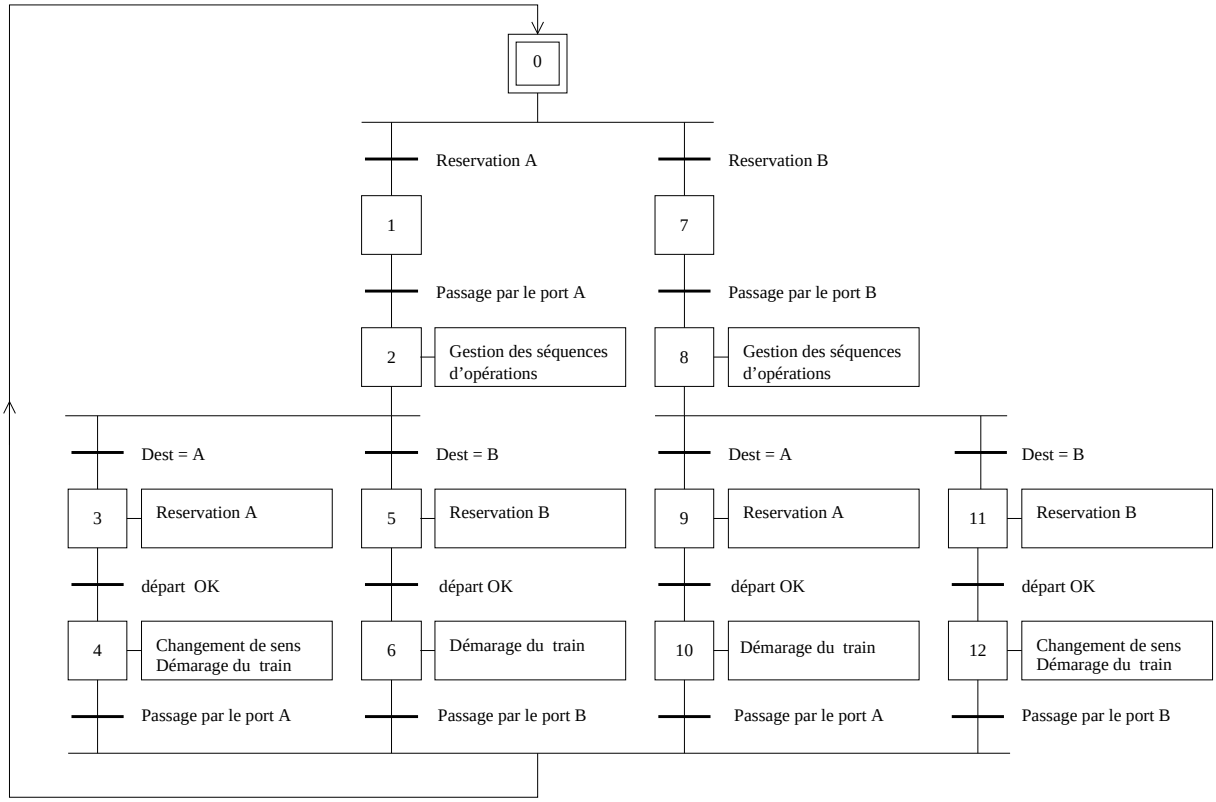
```
23      END_WHILE;
24      IF Seqs[Index] <> 0 THEN
25        IF Oports[Index] = 0 THEN
26          SET Rdy_0;
27        ELSIF Oports[Index] = 1 THEN
28          SET Rdy_1;
29        ELSIF Oports[Index] = 2 THEN
30          SET Rdy_2;
31        END_IF;
32      END_IF;
33
34      Index := 0;
35      WHILE NOT (Seqs[Index] = 0 OR (Seqs[Index] = Seq_2 AND Iports[
           Index] = 2)) DO
36          INC Index;
37      END_WHILE;
38      IF Seqs[Index] <> 0 THEN
39        IF Oports[Index] = 0 THEN
40          SET Rdy_0;
41        ELSIF Oports[Index] = 1 THEN
42          SET Rdy_1;
43        ELSIF Oports[Index] = 2 THEN
44          SET Rdy_2;
45        END_IF;
46      END_IF;
47  END_IF;
```

C.3 Les SFC associés aux composants

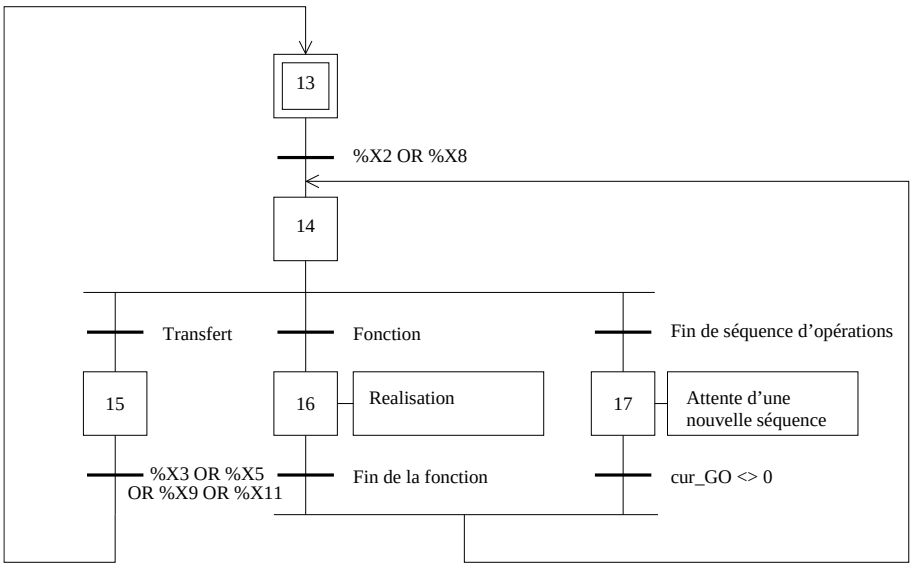
Deux SFC sont associés à chaque tronçon et sont représentés à la figure C.3. Le premier, représenté à la figure C.3(a), dirige les déplacements du train alors que le second interprète les séquences d'opérations associées. Le SFC gérant les déplacements du train est composé de quatre branches correspondant aux quatre scénarios de déplacement possible pour un train lorsqu'il est sur un tronçon. Le train peut entrer par le port A et sortir par le port B, entrer par le port A et sortir par le port A, entrer par le port B et sortir par le port A et enfin entrer par le port B et sortir par le port B. Chaque scénario débute par une réservation du tronçon, réalisée par l'un des deux aiguillages adjacents au tronçon. Le tronçon attend alors l'arrivée du train par le port correspondant. Une fois que la locomotive est sur le tronçon, elle est stoppée et la séquence d'opérations correspondante est gérée par le second SFC associé au tronçon. L'évolution de ce SFC permet de déterminer quelle sera la destination du train à l'aide de la table de routage, la réservation du tronçon d'arrivée est alors effectuée et une fois le démarrage du train confirmé, il reçoit l'ordre de démarrer. Le scénario se termine lorsque le port de sortie est franchi.

Le second SFC associé aux tronçons est dessiné à la figure C.3(b). Il est chargé de faire évoluer les séquences d'opérations et démarre son exécution lorsque le SFC principal atteint l'étape 2 ou l'étape 4. Trois cas peuvent se produire. Si l'action courante est un transfert alors le SFC principal reprend la main, le SFC de gestion des séquences de fonction attend l'arrivée dans l'étape 3, 5, 9 ou 11 du SFC principal pour se réinitialiser. Dans le cas où le transfert en cours est terminé et qu'un traitement doit être réalisé sur le tronçon, ce traitement est réalisé et la position courante de la séquence d'opérations est incrémentée. Si la fin de la séquence d'opérations est atteinte, le tronçon atten une nouvelle séquence d'opérations.

Le SFC associé à un aiguillage est présenté à la figure C.4. Il est constitué d'une branche par connexion réalisée par cet aiguillage. Chacune de ces branches est composée de deux étapes, la première transition est franchie lorsque les SFC des tronçons en amont et en aval sont prêts pour effectuer le transfert, ces SFC sont alors dans les étapes 1 ou 7 pour le tronçon en attente d'un train et dans les étapes 3, 5, 9 ou 11 pour le tronçon attendant le départ du train. L'aiguillage



(a) SFC principal



(b) Traitement des séquences d'opérations

FIG. C.3 – SFC associés à un tronçon

est positionné au cours de cette première étape et le SFC y reste tant que le train se trouve sur le tronçon en entrée, il attend ensuite l'arrivée du train sur le tronçon d'arrivée pour revenir à l'étape initiale.

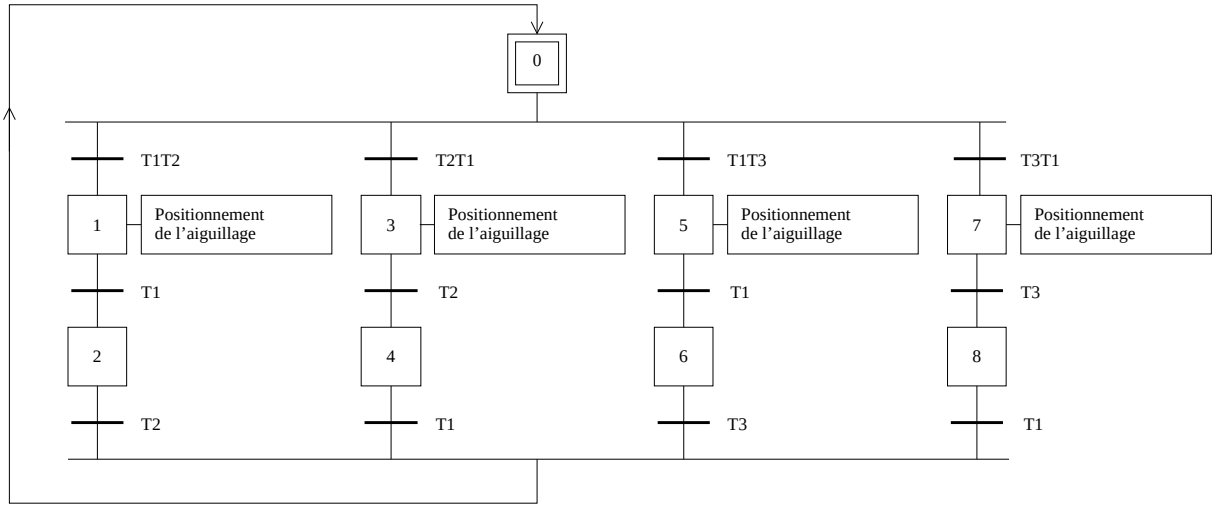


FIG. C.4 – Le SFC de commande d'un aiguillage

C.4 Traitements préliminaires et postérieurs

Un traitement préliminaire et un traitement postérieur sont associés à l'exécution d'un SFC. Le traitement préliminaire est exécuté avant l'interprétation du SFC alors que le traitement postérieur est évalué une fois le SFC interprété.

Deux actions sont réalisées lors du traitement préliminaire. Dans un premier temps, toutes les entrées/sorties nécessitant une acquisition sont évaluées (ce traitement est spécifié dans le fichier IOA). Ensuite, les différents blocs fonctionnels sont activés avec l'état courant des différents composants.

Le traitement postérieur gère le positionnement des aiguillages par rapport à l'état des SFC des différents aiguillages.

Annexe D

Logiciels manipulés pour le démonstrateur ferroviaire

Plusieurs logiciels ont été utilisés ou développés dans le cadre du démonstrateur ferroviaire. Cette annexe récapitule ces différents outils.

D.1 ATL

Un ensemble de transformations de modèles ATL est utilisé pour l'obtention du code de contrôle-commande des automates et pour les analyses de la modélisation du circuit (architecture + configurations). Ces transformations sont lancées à partir d'un ensemble de *Makefiles* définissant les étapes nécessaires à l'obtention d'un modèle donné.

D.2 PL7-Pro

PL7-Pro est un outil développé par Schneider-Electric pour la programmation des automates de la série TSX-Premium tels que celui qui est utilisé dans l'application.

Nous l'utilisons pour lire les SFC générés par transformation de modèles et les envoyer dans l'automate. Ces programmes sont ensuite envoyés à l'automate.

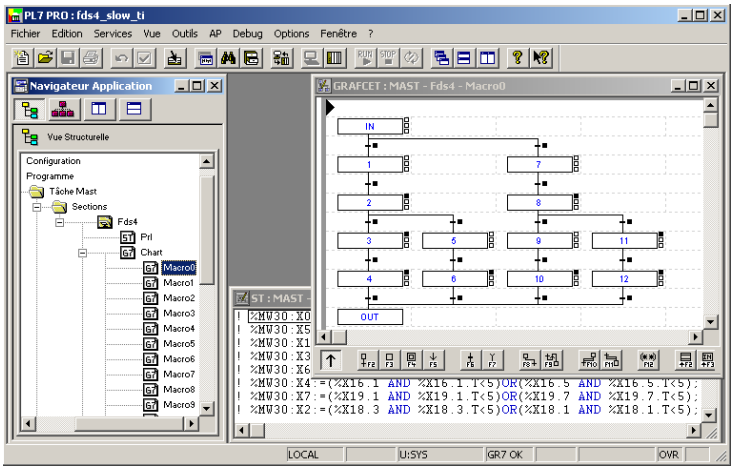


FIG. D.1 – Capture d'écran de PL7-Pro

D.3 Interface de commande des trains

L'interface de commande des train est une application écrite en Visual-Basic pour envoyer les ordres de déplacement a chaque train à partir du PC développée au cours de la thèse.

Cette interface communique avec l'automate afin d'appliquer la commande désirée par celui-ci. La lecture des ordres imposés par l'automate est réalisée périodiquement en utilisant le protocole modbus-TCP.

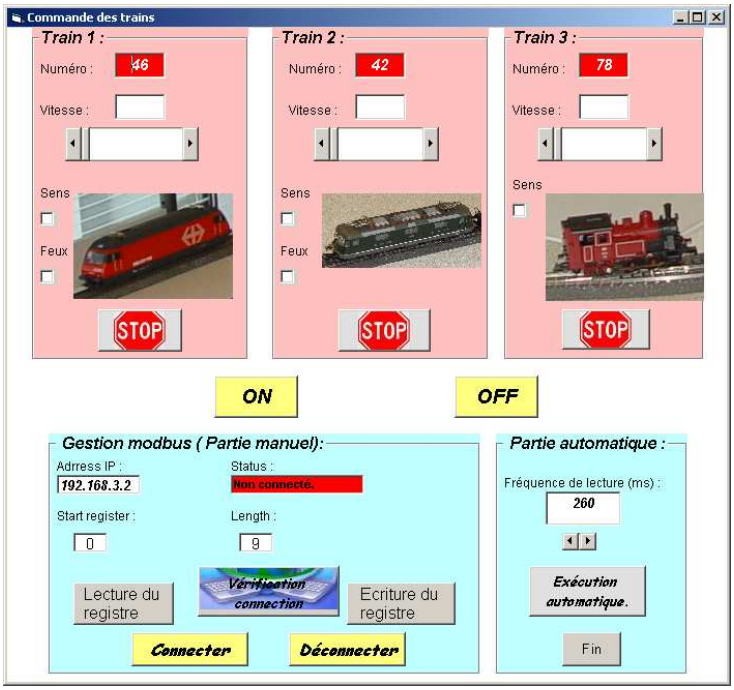


FIG. D.2 – Capture d'écran de l'interface de commande des trains

D.4 Interface de reconfiguration

L'interface de reconfiguration a été développée en Java. Elle manipule la modélisation DeSyRe du système ainsi que le modèle décrivant le mode des ressources en appelant directement des transformations de modèles écrites en ATL.

Cette interface permet de spécifier le mode courant des ressources dans un premier onglet présenté sur la figure D.3.

Un deuxième onglet représenté à la figure D.4 permet de sélectionner la configuration a appliquer. Le choix se fait parmi une liste de configurations, classées suivant leur intérêt par rapport à la situation courante.

Le dernier onglet de l'application, représenté sur la figure D.5 permet d'envoyer la configuration choisie (sous la forme de tables de routage) à l'automate et de piloter le circuit dans cette configuration en proposant de lancer les différentes opérations de la configuration sélectionnée. Cet onglet dialogue avec l'automate à l'aide du protocole Modbus-TCP en utilisant une version modifiée de la bibliothèque JaMod [1] qui permet l'envoi de requêtes UNI-TE à l'automate.

[1] http ://jamod.sourceforge.net

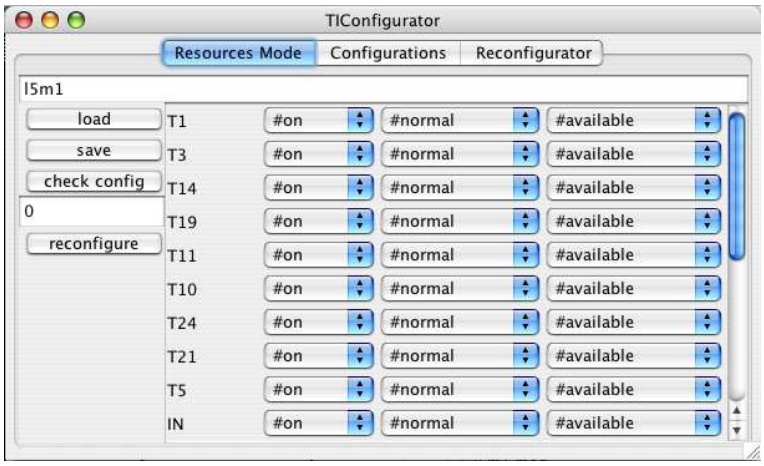


FIG. D.3 – Interface de gestion des modes

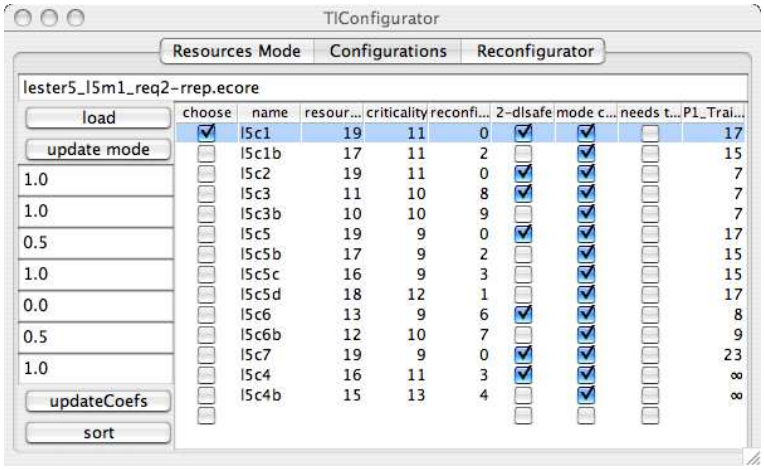


FIG. D.4 – Panneau de sélection de la configuration

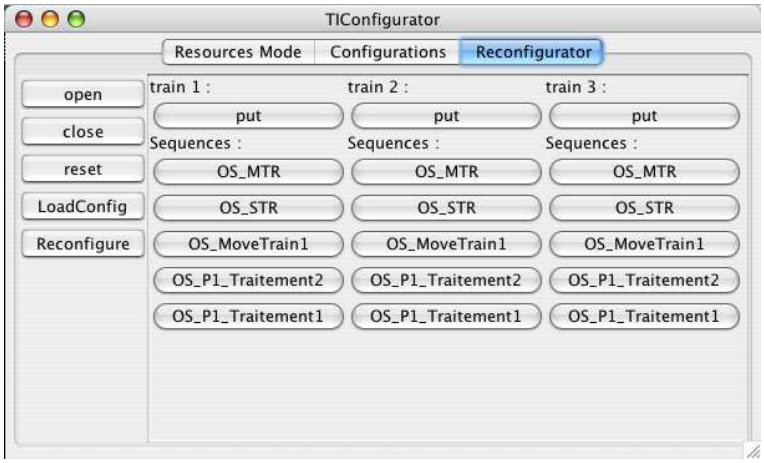


FIG. D.5 – Communication avec l'automate

Index

Table des figures

Listings

Proposition d'une approche haut niveau pour la conception, l'analyse et l'implantation des systèmes reconfigurables

La qualité de service attendue pour les systèmes de production et les systèmes électroniques, rend incontournable l'utilisation de la reconfiguration qui permet d'organiser le système au mieux pour répondre à l'objectif fixé. L'exploitation du caractère reconfigurable d'un tel système nécessite une prise en compte de cette capacité de la conception à l'implantation.

La phase de conception repose sur une description du système reconfigurable selon deux axes. Le premier sépare l'architecture du système de ses configurations. Le second distingue la partie logique, décrivant les traitements à réaliser et la partie physique décrivant les ressources constituant le système.

Une fois l'architecture et les configurations décrites, des analyses permettent au concepteur d'évaluer l'architecture ou les configurations en terme de performances, de tolérance aux pannes ou de coût. Ces analyses peuvent aussi intervenir pour guider la stratégie de reconfiguration au cours du fonctionnement du système. Elles font appel à des outils d'ingénierie dirigée par les modèles permettant d'obtenir automatiquement les modèles d'analyse à partir de la description du système.

Un cadre d'implantation, utilisant également l'ingénierie dirigée par les modèles, est proposé pour la génération d'un code de contrôle.commande à partir de la description de haut niveau. Toute l'approche a été déployée sur une application réelle développée au cours des travaux de thèse et présentée dans ce document.

Mots clés : Systèmes reconfigurables, Approche haut-niveau, Ingénierie Dirigée par les Modèles, Reconfiguration, Contrôle/Commande

High level approach for the design, analysis and implementation of reconfigurable systems

The expected quality of service of manufacturing or electronic systems makes reconfiguration capabilities essential. Reconfiguration consists in a reorganization of the system to better comply with a given objective. To fully take advantage of all the reconfiguration capabilities of the system, designers need to take them into account throughout the whole development process.

The design phase is performed on a description of the reconfigurable system, which is split along two axes. The first axis separates the system's architecture from its configuration. The second one separates the logical part of the system describing the processes from the physical part of the system that describes its resources.

Once the system has been described, analyses help designers in the evaluation of its performances, breakdown tolerance and cost. Analyses can also be used for leading the reconfiguration strategy. They make use of model engineering techniques, which enable automatic transformations from the description model to analysis models.

An implementation framework, which also uses model engineering is described for the control-command code generation from the high-level description. The whole approach has been tested on a real platform built during the thesis.

Keywords : Reconfigurable Systems, High-level approach, Model Engineering, Reconfiguration, Control/-Command

Discipline : *Sciences et Technologies de l'Information et de la Communication*

Spécialité : *Automatique et Informatique Industrielle*

Thèse préparée au Laboratoire d'Electronique des Systèmes Temps Réel (LESTER)
CNRS FRE 2734 – Université de Bretagne Sud
Centre de Recherche
BP 92116
56321 Lorient Cedex
France