Formalisation de la conception d'architecture de conduite des systèmes de production

Bruno Denis, Jean-Jacques Lesage, Guy Timon

FORMALISATION DE LA CONCEPTION D'ARCHITECTURE DE CONDUITE DES SYSTÈMES DE PRODUCTION

FORMALIZATION OF THE ARCHITECTURE DESIGN OF PRODUCTION SYSTEMS

Bruno DENIS, Jean-Jacques LESAGE & Guy TIMON

LURPA, 61 av. du Président Wilson, ENS de Cachan, 94235 CACHAN Cedex, France, Tél : +33 / 1 47 40 22 15, Fax : +33 / 1 47 40 22 20, e-mail : {denis, lesage, timon}@lurpa.ens-cachan.fr

Résumé : La conception de la conduite des systèmes complexes de production passe par une activité de conception de l'architecture de conduite. Dans cet article nous proposons un cadre méthodologique intégré pour formaliser l'activité de conception de l'architecture de conduite. Les modèles mis en place dans notre démarche sont présentés, ainsi qu'une méthode d'évaluation des d'architectures de conduite élaborées.

Abstract : In the control design of complex manufacturing systems, the design of control architecture is required. This paper deal with a integrated framework to make the design of control architecture as formal as possible. Models used in this framework are presented, and a valuable method of control architecture is constructed.

1. Introduction

La performance des systèmes automatisés de production est pour une part conditionnée par la performance de leur système de conduite. Le degré d'automatisation, de réactivité et de flexibilité exigé des systèmes de production allant croissant, le système de conduite pèse de plus en plus lourd dans leur conception - à la fois sur le plan technique et sur le plan économique. Les travaux dont les résultats sont exposés dans cet article portent sur la *conception de l'architecture de conduite des systèmes automatisés de production*. Les systèmes de production visés intègrent à la fois des équipements de l'informatique générale, de l'informatique industrielle, ainsi que des équipements de régulation et d'automatisme. Cette architecture de conduite est souvent représentée à la manière de la figure 1, qui ne présente qu'un point de vue matériel. Lors de la conception d'architecture de conduite, l'ingénieriste est confronté à des problèmes de nature très différente. Il doit assurer à la fois :

- la gestion des relations avec le client,
- la gestion de la complexité du système,
- la gestion de la diversité des technologies et de l'évolution du marché.

Le client exprime ses besoins, généralement de manière très informelle, sous la forme d'un appel d'offre, à des sociétés d'ingénierie. Celles-ci fournissent en retour une offre chiffrée répondant aux besoins exprimés. Pour élaborer cet avant-projet, l'ingénieriste doit entre autre concevoir une architecture de conduite qu'il lui sera difficile de remettre en cause dans la suite du projet. Pourtant cette architecture de conduite orientera de manière décisive la suite de la conception et de la réalisation du système. «L'architecte automaticien» doit donc affronter le paradoxe suivant :

- étudier et formaliser *suffisamment en détail* les besoins du client afin de lui proposer une solution *pertinente* (c'est-à-dire pressentir avant la conception une grande partie du travail de conception), avec des coûts calculés au plus juste dans l'espoir d'emporter le marché.
- étudier *suffisamment sommairement* l'appel d'offre pour ne pas engager trop de frais d'étude sur une affaire dont la société d'ingénierie n'a pas encore la charge contractuelle.

Seule l'expérience de l'ingénieriste permet de contourner ce paradoxe. Par la pratique de cas similaires déjà traités (l'expertise), il est capable de proposer une solution suffisamment fine, à l'issue d'une pré-étude rapide. Il n'en reste pas moins que des outils d'assistance à la conception et à l'évaluation de solutions d'architecture permettraient :

- de *rationaliser* et d'*homogénéiser* l'expertise à l'intérieur de la société d'ingénierie ;
- de *pérenniser* et de *partager* l'expertise capitalisée lors des études précédemment effectuées ;
- d'augmenter la *qualité* des architectures conçues, tout en faisant face à l'augmentation régulière de la complexité des systèmes de production ;
- d'augmenter la *réactivité* de l'ingénieriste vis-à-vis de l'évolution rapide du marché de l'offre en équipements de régulation et d'automatisme face à laquelle il ne peut pas compter sur ses expériences antérieures ;

- d'augmenter la *productivité* de la phase d'avant-projet à laquelle se situe la conception d'architectures.

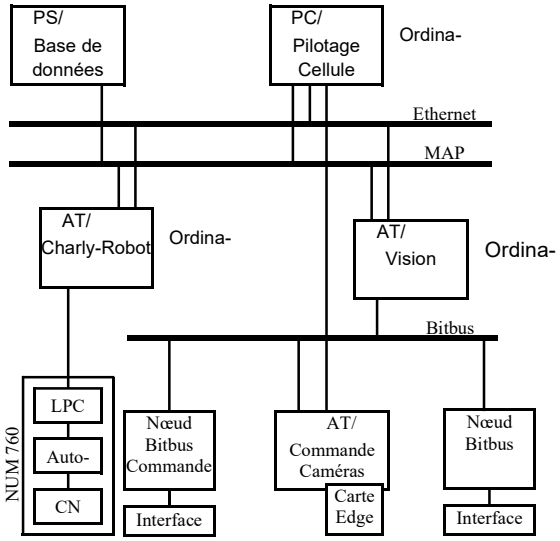

Fig.1 : exemple de la vue matérielle d'une architecture de conduite [Richard & al. 93]

L'ingénieriste est donc confronté à un environnement complexe et très évolutif. Des *outils de modélisation* et des *méthodes de conception* doivent lui fournir le moyen de gérer cette complexité tout en tendant vers une qualité toujours accrue des systèmes développés.
C'est cette optique que nous proposons dans la section suivante, dans un cadre méthodologique pour «l'architecte automaticien».

2. Démarche de conception d'architecture

La démarche que nous proposons prend en charge la conception d'architecture à partir des spécifications générales du système, et permet l'élaboration d'une architecture matérielle (fig. 2).
La première étape de la démarche consiste à extraire des modèles de spécifications du système de production tous les aspects fonctionnels qui seront implantés dans l'architecture de conduite. Cela assure une indépendance de notre démarche vis-à-vis des méthodes et modèles de spécification usuellement employé par l'ingénieriste. Un modèle dit *modèle d'implantation* est alors élaboré. Dans une deuxième phase l'architecte propose une architecture lui paraissant répondre au problème adéquat sous la forme d'un *modèle organique*. C'est dans la troisième phase de la démarche que l'architecte affecte les aspects fonctionnels, décrits dans le modèle d'implantation, dans les composants de l'architecture décrits dans le modèle organique. Il élabore ainsi un modèle dit *modèle d'affectation*. Une solution d'architecture est ainsi élaborée. Sa description rigoureuse au travers des modèles précités nous permet d'envisager des évaluations de cette solution. L'architecte pourra alors choisir une solution finale d'architecture, parmi les différentes solutions qu'il aura évaluées. La section qui suit présente les modèles utilisés dans notre démarche, et la section 4 de cet article présentera une des techniques d'évaluations d'architecture misent en place.

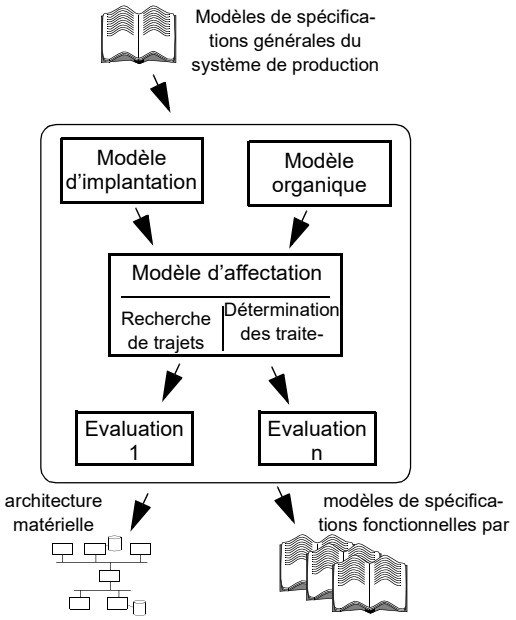

Fig.2 : démarche de conception d'architecture

3. Modélisation de l'architecture

3.a Modèle d'implantation

Le modèle d'implantation a pour rôle :

- de regrouper et d'intégrer, en un seul et unique formalisme, les seules informations pertinentes pour la conception d'architecture. Par exemple, toute fonction exprimée dans la spécification du système de production devant être supportée par le système de conduite doit être représentée dans le modèle d'implantation. De même, toute donnée devant être stockée ou devant circuler y sera représentée ;
- de préciser les événements déclencheurs des fonctions de conduite. Par exemple, si la spécification décrit une fonction qui consomme des données afin d'en produire de nouvelles, le modèle d'implantation devra préciser quelles conjonctions d'événements sont nécessaires à l'accomplissement de la fonction (c'est-à-dire *«quand»* la fonction doit être réalisée) ;
- de préciser si les données exprimées dans la spécification sont fugaces ou pérennes. En d'autres termes, leur durée de vie est-elle limitée au temps de transfert dans le canal de communication qui relie la fonction productrice à la fonction consommatrice, ou est-il nécessaire de prévoir un espace de stockage assurant la pérennité de l'information ?

Le modèle d'implantation est alors construit *comme si l'architecture était réduite à un seul et unique*

constituant (une machine virtuelle). Il constitue à ce titre un modèle transitoire entre les modèles de spécification du système, élaborés sans aucune référence à l'architecture, et le modèle d'affectation qui prend en compte à la fois les contraintes dues à l'exécution dans des constituants microprogrammés, et les contraintes dues à la répartition de la conduite dans plusieurs constituants.

Le choix du formalisme retenu a été conduit en considérant la nature des entités que l'on souhaite décrire. Le modèle d'implantation doit en effet représenter l'ensemble des traitements nécessaires à l'exécution des fonctions de conduite, les stockages et les flots de donnée, ainsi que les règles de synchronisation entre traitements. Un formalisme à base de flots de données a été choisi pour deux raisons : une grande diffusion dans le domaine du génie automatique, et de nombreux travaux ayant déjà porté sur des extensions à ce formalisme [Bruyn & al. 88] [France 92] [Ward 86].

Le modèle d'implantation est, comme le montre la figure 3, un réseau d'interconnexion de traitements, reliés par les données qui circulent. Le modèle se construit à partir de quatre primitives graphiques : des *traitements*, des *stockages*, des *flots*, et des entités *externes*.
Les traitements sont séparés en trois classes distinctes. Elles correspondent à des mises en œuvre différentes au niveau de leur exécution par un constituant microprogrammé. On distingue ainsi :
- les *traitements procéduraux* (P), qui sont caractérisés par un début et une fin, entre lesquels des séquences en nombre fini s'enchaînent de manière continue,
- les *traitements de contrôle/commande* (CC), qui nécessitent une exécution permanente, et qui sont représentés par un automate fini traduisant les relations séquentielles entre entrées et sorties,
- les *traitements de coordination de traitements* (en traits pointillés), qui contrôlent les autres traitements du modèle d'implantation par le biais d'activations et de désactivations.

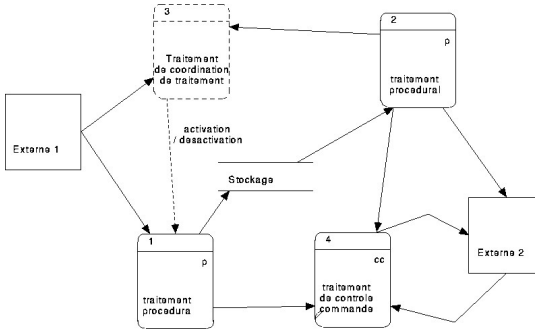

Fig.3 : exemple de modèle d'implantation

Les stockages de données permettent la mémorisation de tout type de donnée, indépendamment de sa taille et de son contenu. Ils doivent être vus par les traitements comme des services qui leurs sont offerts pour stocker (écrire) et déstocker (lire) des données.
Les entités externes indiquent la provenance ou la destination des flots traités par le système.

Les flots permettent de relier les traitements entre eux et de relier les traitements avec les stockages de données. Ils sont séparés en trois classes distinctes correspondant à la nature des informations qu'ils véhiculent. On distingue ainsi :
- les *flots de donnée*, qui relient les traitements aux stockages et inversement,
- les *flots de message*, qui sont porteurs à la fois d'une donnée et d'un événement ; ils relient les traitements entre eux et permettent à un traitement émetteur de déclencher l'exécution d'un traitement récepteur par l'événement contenu dans le flot, tout en lui transmettant une donnée,
- les *flots de coordination*, qui sont issus des traitements de coordination de traitement, permettent l'activation et la désactivation des traitements procéduraux et des traitements de contrôle-commande.

A ce formalisme de base (modèle de flot de données), des enrichissements ont été apportés pour préciser à la vue de quels événements un traitement doit être exécuté. Cet enrichissement est destiné à évaluer ultérieurement à quels moments des ressources CPU seront nécessaires, et quels flots de données vont être induits.

Les traitements procéduraux se comportent en transformateurs de données (fig. 4). Le début de leur exécution est conditionné par l'apparition d'une conjonction d'événements, véhiculés dans les messages, qui marque le début du traitement. En fin de traitement, les données produites sont regroupées au sein d'une disjonction. Selon les alternatives rencontrées durant le traitement, une et une seule des disjonctions possibles sera émise. Les stockages de données ne contribuent pas à la synchronisation des traitements. Ce sont les traitements qui lisent ou écrivent des données dans les stockages selon leur propre initiative.
La description d'un traitement procédural doit donc être complétée par le renseignement du triplet $(C(t), D(t), S_c(t))$ avec :
- $C(t)$ l'ensemble des conjonctions associées au traitement t,
- $D(t)$ l'ensemble des disjonctions associées au traitement t,
- $S_c(t)$ l'ensemble des stockages de données à lire à la suite d'une conjonction associée au traitement t,

et où $\forall d \in D(t)$, d est le couple $(S_e(d), M(d))$ dans lequel :

- $S_e(d)$ est l'ensemble des stockages qui vont être effectués lors de la disjonction d ,
- $M(d)$ est l'ensemble de messages qui sont émis lors de la disjonction d .

Un traitement procédural vérifie alors les propriétés suivantes :

- le traitement t est déclenché lorsque l'équation booléenne (1) est vérifiée, c'est-à-dire lorsqu'il existe une conjonction pour laquelle tous les messages associés sont émis,

$$\sum_{c_i \in C(t)} \left(\prod_{m \in c_i} m \right) = 1 \qquad (1)$$

- $\forall d \in D(t)$, lorsque la disjonction se produit, une donnée est émise dans tous les stockages de $S_e(d)$, puis tous les messages de $M(d)$ sont émis.

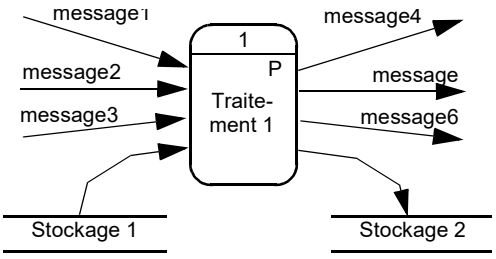

Fig.4 : exemple de représentation graphique d'une traitement procédural

La figure 4 propose un exemple de traitement procédural. Il est déclenché lorsque l'on a :

$$\begin{aligned}&(message_1 \cdot message_2) + \\ &(message_2 \cdot message_3) + \\ &(message_1 \cdot message_3) = 1\end{aligned} \qquad (2)$$

En fin d'exécution on pourra :

- soit stocker un résultat dans le stockage 2 et émettre les messages $message_4$ et $message_5$,
- soit émettre les messages $message_5$ et $message_6$.

En d'autres termes, la description graphique de la figure 4 doit être complétée par le triplet $(\{c_1, c_2, c_3\}, \{d_1, d_2\}, \{stockage_1\})$ avec

- $c_1 = \{message_1, mesage_2\}$
- $c_2 = \{message_2, mesage_3\}$
- $c_3 = \{message_1, mesage_3\}$
- d_1 est le couple $(\{stockage_2\}, \{message_4, message_5\})$
- d_2 est le couple $(\varnothing, \{message_5, message_6\})$

En résumé l'activité d'élaboration d'un modèle d'implantation est donc une activité de traduction et d'enrichissement d'un ensemble de spécifications du système de production en un modèle de spécification du seul système de conduite. Cette tache peut être formalisée, voire même automatisé, à l'aide par exemple de la métamodélisation telle qu'elle est décrite dans [Denis & al. 93].

3.b Modèle organique

Elaborer un modèle organique consiste à produire une représentation physique de l'architecture de conduite imaginée par l'architecte pour répondre au problème. C'est souvent le seul point de vue utilisé pour décrire une architecture de conduite comme par exemple dans [Richard & al. 93].

L'absence de formalisme «universel» permettant de représenter la structure organique d'une architecture de conduite, nous à conduit à adopter une représentation proche des usages de la représentation du modèle des équipements de commande. Toutefois, la norme française NF E 04-203-3 intitulée «Régulation, mesure et automatisme des processus industriels : représentation symbolique», nous propose pour quelque organes de traitements une représentation graphique que nous avons utilisée.

Le modèle organique est, comme le montre la figure 5, une interconnexion d'*équipements de traitement*, d'*équipements de stockage* et d'*équipements de dialogue*, reliés entre eux par des *media de communication*.

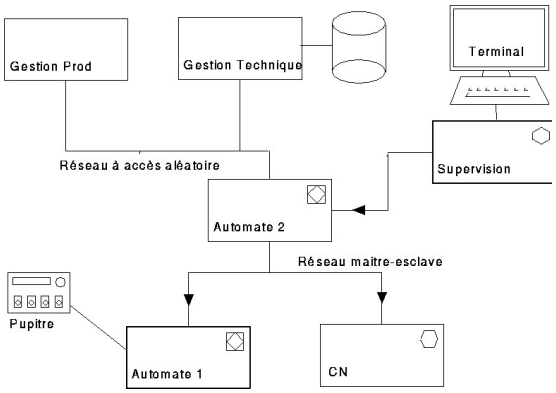

Fig.5 : exemple de modèle organique

Notre démarche doit être apte à supporter les évolutions importantes des technologies de commande et à s'adapter à des systèmes cibles très variés. C'est pourquoi les primitives du modèle organique imposent à l'architecte de poser des solutions potentielles d'architecture en terme de *classes d'équipements* ayant des capacités techniques identifiées et non en terme de *composants technologiques* du marché de l'offre.

La liste des équipements types que nous proposons dans la suite de cet article n'est ni exhaustive, ni générale. Elle se veut cependant représentative des systèmes de production industriels à dominante manufacturiers. Elle peut être facilement complétée ou spécialisée pour d'autre domaines cibles.

Parmi les équipements de traitement, nous distinguons ainsi :

- les Automates Programmables Industriels (API),
- les Systèmes Numériques de Contrôle/Commande (SNCC),

- les Calculateurs que l'on subdivise en calculateurs de processus (informatique embarquée), en calculateurs de supervision (édition de journaux, de bilans, calculs scientifiques), et en calculateurs banalisés.

Chaque équipement de stockage sera géré par un équipement de traitement qui assurera les accès aux données. L'utilisation d'un équipement de stockage traduit la volonté de l'architecte d'utiliser un système (logiciel et mémoire de masse) dédié à la gestion d'un volume important de données.

Parmi les équipements de dialogue avec l'opérateur on distingue :
- les terminaux alphanumériques,
- les terminaux graphiques,
- les pupitres.

Parmi les media de communication on distingue :
- les media bi-connexions que l'on subdivise en media informatique à accès maître-esclave ou à accès aléatoire, et en media «fils à fils» comme par exemple entre un pupitre de boutons et de voyants et un API, ou entre deux API que l'on relie avec quelques entrées-sorties ;
- les media multi-connexions que l'on subdivise en réseaux de capteurs, en réseaux d'automates à accès maître-esclave, et les réseaux locaux à accès aléatoire.

3.c Modèle d'affectation

Elaborer un modèle d'affectation consiste, lorsque le modèle d'implantation est élaboré et qu'un premier jet du modèle organique est mis en place, à *projeter* le modèle d'implantation sur le modèle organique.

Pour ce faire, les traitements vont être affectés à des équipements de traitement pour assurer leur exécution, de même les stockages vont trouver des mémoires pour accueillir leurs données, et les flots de données vont se voir affecter un trajet à travers les équipements de communication.

Nous proposons une démarche qui consiste dans un premier temps à affecter chaque traitement et chaque stockage à un équipement de traitement et/ou de stockage. Une assistance est ensuite proposée à l'architecte pour déterminer les trajets des flots de données induits par ces choix. A ce niveau le modèle organique est transformé en un graphe. La recherche d'un trajet dans le modèle organique pour un flot donnée se traduit alors par la recherche d'un chemin dans le graphe associé.

La stratégie de construction du graphe est la suivante :
- à chaque équipement de traitement et chaque équipement de dialogue est associé un sommet du graphe,
- à chaque media multi-connexions à accès aléatoire est associé un sommet du graphe,
- pour tout media bi-connexions entre deux équipements de traitement (ou entre un équipement de traitement et un équipement de dialogue), on place une arête entre les deux sommets correspondants,
- pour tout media multi-connexions à accès maître-esclave une arête est placée entre le sommet associé à l'équipement maître, et chaque sommet correspondant aux équipements esclaves,
- pour toute multi-connexion à accès aléatoire, une arête est placée entre le sommet associé au media et chaque sommet correspondant aux équipements connectés.

La figure 6 illustre cette stratégie de construction du graphe associé. Assister l'architecte pour trouver le trajet d'un flot dont on connaît l'équipement de traitement source et l'équipement de traitement destination, consiste donc à identifier les deux sommets associés aux deux équipements de traitement mis en jeu, puis à chercher dans le graphe un chemin propre. Si plusieurs chemins sont identifiés, une pondération des arêtes (représentant par exemple le débit de transmission) peut aider l'architecte à choisir la meilleure solution comme celle correspondant au chemin le plus «court» au sens du graphe.

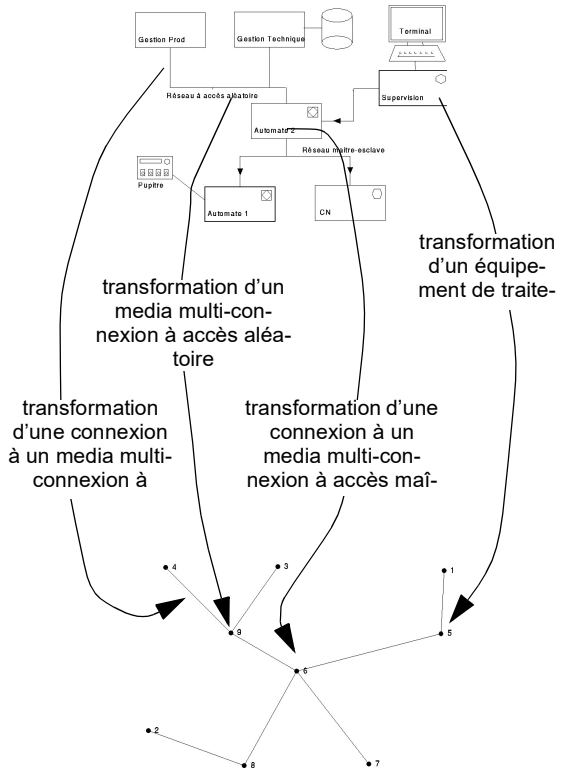

Fig.6 : exemple de transformation d'un modèle organique en graphe

Pour faire circuler les données, et coordonner les traitements dans une architecture de conduite, des traitements et des flots techniques sont nécessaires. Ils ne sont pas issues des modèles de spécifications, mais ils sont induits de la répartition de la conduite décrite dans le modèle organique. L'apparition de ces traitements et flots techniques a une grande incidence sur

les performances globales d'une architecture, mais ils sont rarement pris en compte dans la conception d'architecture car d'un niveau de détail très fin (paradoxe de l'ingénieriste qui doit étudier suffisamment en détail les besoins du client afin de lui proposer une solution pertinente, tout en les étudiant suffisamment sommairement pour ne pas engager trop de frais d'étude sur une affaire dont la société d'ingénierie n'a pas encore la charge contractuelle). Nous proposons une détermination systématique des traitements et flots techniques, basée sur le chemin des flots dans le graphe et la constitution de modèles de référence pour chaque type de flot.
L'étude des traitements techniques est un point de passage important pour prendre en compte toutes les incidences d'un choix de répartition dans une architecture de conduite. Le caractère systématique de notre étude la rend automatisable, donc transparente pour un architecte utilisateur.

Une fois élaboré, le modèle d'affectation fournit à l'architecte une vue complète des traitements qui seront exécutés dans une structure d'équipements donnée. En ce sens, il personnifie plus sûrement une architecture de conduite que le modèle organique. Un modèle d'affectation *est une solution d'architecture*. Il est intrinsèquement intéressant d'avoir formalisé à travers trois modèles une activité peu explorée du génie automatique : la conception d'architecture. Mais nous allons monter qu'il est maintenant possible de compléter cette approche par des méthodes d'évaluation des architectures conçues.

4. L'évaluation d'une solution d'architecture

Un des indicateurs de performance d'une solution d'architecture couramment évoqué par les architectes, est le *temps de réponse*. Si la connaissance des performances de chaque équipement de l'architecture permet facilement de donner une valeur minimale d'un temps de réponse, un modèle de comportement dynamique de l'architecture permet quant à lui d'estimer plus sûrement par simulation une valeur réaliste de ce temps de réponse.

Afin de fournir des indications sur les temps de réponse, le modèle d'évaluation doit être un modèle du comportement *dynamique* de l'architecture. Notre choix c'est porté sur les *réseaux de Petri temporisés et colorés*, car la temporisation rend le modèle simulable, donc évaluable. La coloration permet quant à elle d'accroître la concision et la généricité des modèles élaborés.

Le *modèle évaluable* se construit à partir du modèle d'affectation qui traduit à la fois l'aspect fonctionnel de l'architecture issu des spécifications au travers du modèle d'implantation, et à la fois l'aspect organisation au travers des traitements techniques (traitements de scrutation, …). L'originalité de ce modèle est de tenir compte de ces deux aspects. Dans de nombreux travaux, des descriptions de la dynamique d'une spécification de système ont été proposées [Bourey & al. 88] [Moitessier & al. 92] [Zaytoon 93], et dans d'autres travaux des modèles du comportement dynamique de l'architecture organique sont étudiés [Ajmone & al. 85] [Di Stephano & al. 93] [Natarajan 85] [Riat 92]. Le modèle d'évaluation que nous proposons permet quant à lui de décrire le *comportement dynamique du modèle d'implantation projeté sur le modèle organique*.
La démarche proposée est la suivante :

- chaque traitement et chaque stockage du modèle d'affectation vont voir leur comportement dynamique décrit par un réseau de Petri générique (fig. 7) dont l'évolution interne est conditionnée par l'apparition de messages et par des demandes de ressources (telles que la lecture d'une donnée dans un stockage, ou telles que la demande de temps CPU à un équipement de traitement support). L'évolution de chacun de ces RdP conduit à ce qu'ils génèrent à leur tour des envois de messages ;
- chaque équipement, qu'il soit de traitement ou de communication, va être décrit dynamiquement par un réseau de Petri générique le décrivant comme une ressource prête à répondre aux sollicitations des traitements et des stockages ;
- un modèle évaluable sera enfin construit par assemblage des différents réseaux de Petri génériques.

Les réseaux de Petri illustrant dans la suite de cette section ne se veulent pas être les modèles les plus complets, mais les plus pertinents pour l'étude de l'architecture. Il ont été élaborés pour refléter le comportement de toute une architecture, et non pour étudier précisément un type d'équipement de l'architecture organique, ou une partie de la spécification fonctionnelle du système. Le lecteur trouvera des modèles plus fins concernant par exemple les réseaux dans [Gressier 87] ou concernant par exemple les calculateurs dans [Lepold 92].
Les modèles que nous proposons ont été conçus de manière à être les plus génériques possibles. Ils constituent une bibliothèque que l'architecte peut compléter ou modifier, et dans laquelle il va venir chercher les éléments nécessaires à la constitution de la représentation dynamique de son modèle d'implantation projeté sur son modèle organique.

Le modèle des stockages proposé à titre d'exemple décrit tout $s_i \in S$, où S est l'ensemble des stockages du modèle d'implantation (figure 7 et tableau 1). Il est organisé autour des deux places P8 et P7 qui représentent respectivement les données en cours d'accès par un traitement et celles qui ne le sont pas. En amont, les places P1, P2, P3 et les transitions T1, T2, T3 représentent une file FIFO (first in, first out)

de taille $n(s_i)$. Dès qu'une demande d'accès sort de la file (transition T3), une demande de ressource CPU est émise. Une couleur telle que $\langle s_i, t_j, l, e_k \rangle$ doit être interprétée comme une demande de lecture (l) traitée par le stockage s_i pour le traitement t_j, demande qui occupe l'emplacement e_k dans la file d'attente.

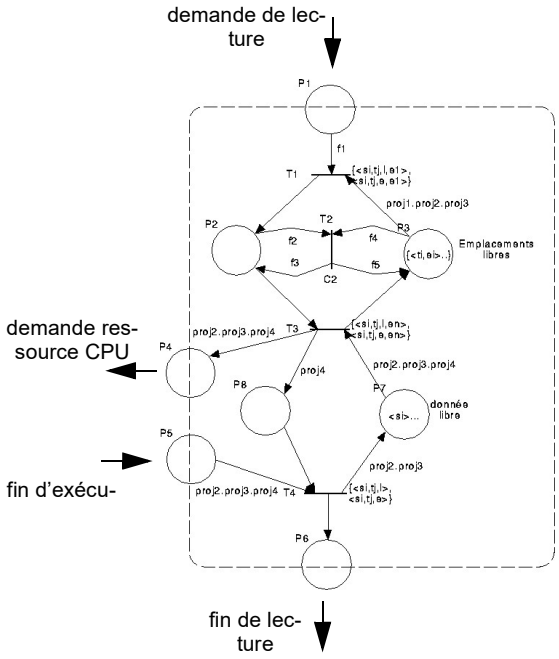

Fig.7 : modèle générique du comportement dynamique des stockage

Le modèle d'une architecture particulière est obtenu en assurant le «routage» des jetons d'un générique à l'autre. La figure 8 montre les arcs et les transitions support du routage, il s'agit là encore d'un modèle générique, c'est dans la description des fonctions associées aux arcs qu'est particularisé le modèle. Une fois les modèles génériques de comportement des éléments de l'architecture établis, et cela une fois pour toute dans un domaine d'application donné, la construction du modèle évaluable d'une architecture est complètement systématique, donc automatisable.

Il donne une nouvelle vision du modèle d'affectation qui, cette fois ci, est évaluable par des techniques de simulation. Evaluer ces temps de réponse, c'est réaliser un scénario de sollicitations, simuler le fonctionnement de l'architecture sous ces sollicitations, et confronter les résultats de la simulation avec ceux attendus dans le guide d'expression des besoins. Pour ce faire, il convient d'utiliser des outils de simulation de réseaux de Petri temporisés [Besombes 90] [Fitrzyk 93] [Tankoano & al. 91] [Valette & al. 85]. La construction systématique d'un modèle de comportement de l'architecture montre une fois de plus l'intérêt de la formalisation du cœur de l'architecture en phase d'évaluation.

Tab. 1 : description du modèle dynamique des stockages

	Description
f1	$f_1(\langle s_i, t_j, l, e_1 \rangle) = \langle t_j, s_i, l \rangle$ $f_1(\langle s_i, t_j, e, e_1 \rangle) = \langle t_j, s_i, e \rangle$
f2	id
f3	$f_3(\langle s_i, t_j, l, e_k \rangle) = \langle t_j, s_i, l, e_{k+1} \rangle$ $f_3(\langle s_i, t_j, e, e_k \rangle) = \langle t_j, s_i, e, e_{k+1} \rangle$
f4	$f_4(\langle s_i, t_j, l, e_k \rangle) = \langle s_i, e_{k+1} \rangle$ $f_4(\langle s_i, t_j, e, e_k \rangle) = \langle s_i, e_{k+1} \rangle$
f5	$f_5(\langle s_i, t_j, l, e_k \rangle) = \langle s_i, e_k \rangle$ $f_5(\langle s_i, t_j, e, e_k \rangle) = \langle s_i, e_k \rangle$
C2	$\{ \langle s_i, t_j, l, e_k \rangle, \langle s_i, t_j, e, e_k \rangle, k \in [1, n(s_i)[\}$
M3	$\sum_{s_i \in S} \left(\sum_{j=1}^{n(s_i)} \langle s_i, e_j \rangle \right)$
M7	$\sum_{s_i \in S} \langle s_i \rangle$

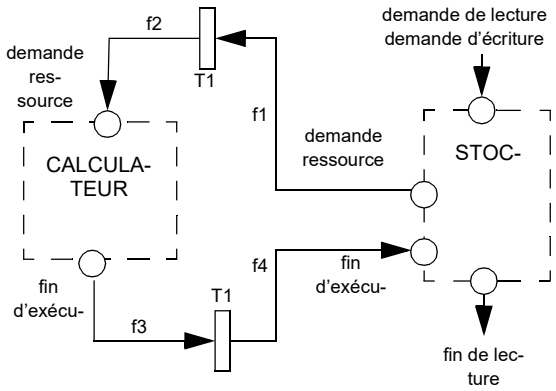

Fig.8 : extrait du modèle globale de routage des demandes de ressource CPU

5. Conclusions

Dans cet article nous nous sommes attachés à la formalisation de l'ensemble de la démarche de conception de l'architecture de conduite des systèmes de production. Nous avons dégagé les activités de l'architecte, en proposant une méthode de conception d'architecture.

L'intérêt de la formalisation de l'architecture a été renforcé en montrant qu'elle rendait possible l'évaluation des architectures conçues. L'évaluation du comportement dynamique a été présenté dans cet

article. D'autres critères d'évaluation ont pu être mis en place comme par exemple des critères d'estimation statistique des trafics sur les média et des taux de charge des équipements de traitement. Actuellement des critères prometteurs sont à d'étude, comme les critères de disponibilité et de sûreté de fonctionnement.

Références

[Bruyn & al. 88] W. Bruyn, R. Jensen, D. Keskar et P. Ward. "ESML: an extended systems modeling language based on the data flow diagram". ACM software engineering notes, vol. 13, n°1, pp. 1-19, 1988.

[France 92] R. France. "Semantically extended data flow diagrams: a formal specification tool". IEEE transaction on software engineering, vol. 18, n°4, pp. 329-346, 1992.

[Ward 86] P. Ward. "The transformation schema: an extension of the data flow diagram to represent control and timing". IEEE transaction on software engineering, vol. 12, n°2, pp. 198-210, 1986.

[Denis & al. 93] B. Denis, J.J. Lesage et G. Timon. "Toward a theory of integrated modelling" Revue Sciences et techniques de la conception, vol. 2, n°1, pp. 87-96, 1993.

[Richard & al. 93] J. Richard, E Bajic et M. Véron. "Système d'information d'une cellule d'usinage auto-contrôlée". Ingénierie des systèmes d'information, vol 1, n°3, pp. 373-392, 1993.

[Bourey & al. 88] BOUREY J.P. Bourey et J.C. Gentina. "Structuration de la partie procédurale du système de commande de cellules flexibles de production" dans les Actes du Congrès Automatique 1988, pp. 233-242, 1988.

[Moitessier & al. 92] F. Moitessier, J.F. Aubry, J.C. Derniame et C. Zanne. "Une méthode de conception des systèmes de commande pour des processus hybrides rapides" dans les Actes de ADPM'92, pp. 3-9, 1992.

[Zaytoon 93] J Zaytoon. "Extension de l'analyse fonctionnelle à l'étude de la sécurité opérationnelle des systèmes automatisés de production". Thèse de doctorat de l'Institut National des Sciences Appliquées de Lyon, 1993.

[Ajmone & al. 85] M. Ajmone et G. Chiola. "Construction of generalized stochastic Petri net models of bus oriented multiprocessor systems by stepwise refinements: a case study", in Proceedings of the International conference modelling techniques and tools for performance analysis, pp. 265-278, 1985.

[Di Stephano & al. 93] A. Di Stephano, O. Mirabella et C. Zappalà. "Featuring FDDI in a process control environment". Computer in industry, vol. 21, n°1, pp. 35-49, 1993.

[Natarajan 85] K.S. Natarajan. "Performance analysis of prefetch strategies for database access". in Proceedings of the international conference modelling techniques and tools for performance analysis, pp. 213-229, 1985.

[Riat 92] J.C. Riat. "Validation par simulation d'architecture de commande d'atelier dans l'industrie de production manufacturière. Application aux systèmes automatisés de production d'un grand constructeur automobile", Thèse de doctorat de l'Université des Sciences et Technologies de Lille, 1992.

[Gressier 87] M.E. Gressier. "La modélisation au moyen des réseaux de Petri Stochastiques : application aux systèmes informatiques fortement synchrones", Thèse de doctorat du Conservatoire National des Arts et Métiers, 1987.

[Lepold 92] R. Lepold. "Performability evaluation of degradable computer systems based on stochastic Petri nets", Thèse de l'institut de recherche polytechnique de l'Université de Haute Alsace, 1992.

[Besombes 90] B. Besombes. "Un système d'aide à la conduite d'atelier flexible basé réseaux de Petri colorés", Thèse de doctorat de l'Université Lyon I, 1990.

[Fitrzyk 93] D. Fitrzyk. "Atelier logiciel pour automaticien : contribution à la réalisation d'un outil d'analyse et de simulation de réseaux de Petri" Mémoire d'ingénieur CNAM, 1993

[Tankoano & al. 91] J. Tankoano, D. Boudebous et J.C. Derniame. "PETRI-S : un simulateur de systèmes de production automatisés décrits à l'aide de réseaux de Petri interprétés colorés", APII, vol. 25, n° 1, pp. 1-30, 1991.

[Valette & al. 85] R. Valette, V. Thomas et S. Bachmann. "SEDRIC : un simulateur à événements discrets basé sur les réseaux de Petri", APII, vol. 19, pp. 423-436, 1985.