# Méthodes et modèles pour un processus sûr d'automatisation

Jean-François Pétin

Nancy-Université
Université Henri Poincaré

Centre de Recherche
en Automatique de Nancy

**E**cole **S**upérieure d'**I**nformatique et **A**pplications de **L**orraine
**Ecole Doctorale** IAEM Lorraine
**DFD** Automatique

**UMR 7039**
**NANCY-UNIVERSITE**
**CNRS**

# HABILITATION A DIRIGER DES RECHERCHES

Université Henri Poincaré – Nancy I

Présentée par

**Jean-François PETIN**
Maître de Conférences
Docteur de l'Université Henri Poincaré – Nancy I

---

# Méthodes et modèles pour un processus sûr d'automatisation

---

Soutenue publiquement le 19/12/2007 devant le jury composé de :

| | | |
|---|---|---|
| *RAPPORTEURS :* | PR. J.-J. LESAGE | ENS CACHAN |
| | PR. E. NIEL | INSA LYON |
| | PR. J. ZAYTOON | UNIVERSITE DE REIMS CHAMPAGNE ARDENNE |
| *EXAMINATEURS :* | PR. D. MERY | UNIVERSITE HENRI POINCARE |
| | PR. G. MOREL | UNIVERSITE HENRI POINCARE |
| | PR. A. TOGUYENI | ECOLE CENTRALE DE LILLE |
| *INVITE* | PR. D. MAQUIN | INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE |

# Remerciements

Je tiens d'abord à remercier tout particulièrement Monsieur Gérard Morel, Professeur à l'Université Henri Poincaré Nancy I, Directeur adjoint du CRAN et responsable du projet SCP, qui, lors de ma thèse puis durant toutes ces années passées dans son équipe, a toujours su me communiquer sa passion de la recherche et m'a accompagné, lors de nos innombrables discussions, dans l'évolution de ma thématique de recherche. Qu'il trouve ici ma profonde reconnaissance.

Je remercie Monsieur Jean-Jacques Lesage, Professeur à l'ENS de Cachan, Directeur du LURPA, Monsieur Eric Niel, Professeur à l'INSA de Lyon et Monsieur Janan Zaytoon, Professeur à l'Université de Reims-Champagne-Ardenne, Directeur du CReSTIC, pour l'honneur qu'il m'ont fait d'être rapporteurs de ce mémoire d'habilitation.

Je tiens à remercier Monsieur Armand Toguyeni, Professeur à l'Ecole centrale de Lille, pour avoir accepté d'examiner ce mémoire et pour les échanges fructueux que nous avons pu avoir durant les projets où nous avons été amenés à collaborer.

Ma reconnaissance va bien sûr à Monsieur Dominique Méry, Professeur à l'Université Henri Poincaré Nancy I, pour avoir accepté d'examiner ce travail, pour l'accueil qu'il m'a réservé dans son équipe de recherche à l'issue de ma thèse ainsi que pour tous les conseils scientifiques avisés qu'il a su me communiquer durant toutes nos collaborations passées qui, je l'espère, se poursuivront encore longtemps.

Mes remerciements vont également à Monsieur Didier Maquin, Professeur à l'Institut National Polytechnique de Lorraine qui m'a fait l'honneur de participer au jury.

Enfin, je souhaite remercier tous mes collègues du CRAN, et particulièrement ceux qui, par leurs nombreuses discussions et leur amitié, nous permettent de travailler dans une ambiance conviviale.

# Préambule

Le présent document décrit mes activités universitaires d'enseignant/chercheur depuis ma soutenance de thèse en 1995 effectuée au CRAN (Centre de Recherche en Automatique de Nancy, UMR 7039) et mon affectation sur le poste n°1323 de l'ESIAL (Ecole Supérieure d'Informatique et Applications de Lorraine) de l'Université Henri Poincaré – Nancy I.

Un curriculum vitae concis présente, en début de document, une synthèse de mes activités de recherche, d'enseignement et d'administration.

Les thématiques développées en recherche concernent la formalisation de cadres de modélisation en vue de maîtriser la complexité croissante des processus d'automatisation due à une part de plus en plus importante de technologies de l'information et de la communication intégrées au cœur même des processus de production et des produits. Plus précisément, notre travail porte sur l'intégration d'approches méthodologiques, issues du **Génie Automatique**, et de modèles formels, issus du **Génie Informatique** et de l'Automatique des **Systèmes à Evénements Discrets** afin de garantir **a priori** le respect des exigences exprimées par les utilisateurs. Après avoir développé dans notre thèse une recherche technologique en réponse aux besoins de R&D industriels de la Direction des Etudes & Recherche d'EDF, nous avons cherché à rationaliser et à formaliser les résultats obtenus, notamment dans le cadre d'un stage post-doctoral au LORIA. Nous avons ensuite poursuivi cet effort de formalisation avec une cible différente puisque relative aux systèmes manufacturiers, et en particulier aux systèmes de production contrôlés par le produit. Notre projet de recherche réalise, en quelque sorte, une synthèse de ce parcours en proposant une action centrale visant à définir un cadre formel pour un **processus sûr d'automatisation** dans un contexte d'**ingénierie système** appliqué, dans deux actions complémentaires, aux domaines des systèmes manufacturiers et de production d'énergie.

Ce travail de recherche est cohérent avec mon parcours en enseignement initialement centré sur l'**Automatique des Systèmes à Evénements Discrets** puis prenant progressivement en compte la dimension « système » des automatismes, notamment au travers de leurs liens avec les progiciels de **pilotage de la production** (M.E.S.) ou des progiciels de **gestion intégrée d'entreprises** (E.R.P.).

# Sommaire

# Curriculum Vitae

## 1. ETAT CIVIL

**PETIN Jean-François**
Né le 1er juillet 1964 à Nice, 2 enfants.
Nationalité française.

Adresse professionnelle
CRAN UMR 7039, Nancy-Université, CNRS
Faculté des sciences, BP 239, 54506 VANDOEUVRE
Tel : +33 (0) 3 83 68 44 43 Fax : +33 (0) 3 83 68 44 59
E-mail :jean-francois.petin@cran.uhp-nancy.fr

## 2. FORMATION UNIVERSITAIRE

**Doctorat** de l'Université Henri Poincaré, en production automatisée, mention très honorable, en décembre 1995,

Titre : « Contribution méthodologique à l'actionnement et la mesure intelligents: application au projet esprit III – P.R.I.A.M. N°6188 ».
Directeur de Recherche : Pr. G. MOREL
Responsable de la Recherche : B. IUNG
Jury : Jury : J.F. Aubry (*Président*), M. Staroswiecki, R. Valette (*Rapporteurs*), D. Galara, B. Iung, M. Robert (*Examinateurs*), G. Morel (*Directeur de thèse*).

**DEA** Production Automatisée à l'Université Henri Poincaré NANCY I en juin 1991, mention Bien

Titre : « Expérimentation industrielle du concept de Contrôle, Maintenance et Gestion technique intégrés: modèles de référence »
Responsable de la Recherche : G. MOREL

**MST** Automatique et Commande Numérique, Faculté des Sciences, Université de Nancy I, juin 1990.

## 3. SITUATION ACTUELLE

**Maître de Conférences** au 4ème échelon de la classe normale**,** à l'Université Henri Poincaré NANCY I (emploi n°1323),

- en poste à l'**E**cole **S**upérieure d'**I**nformatique et **A**pplications de Lorraine (ESIAL) dirigée par Monsieur le Professeur André SCHAFF.
- chercheur au **C**entre de **R**echerche en **A**utomatique de **N**ANCY, Unité Mixte de Recherche UMR 7039, Nancy-Université, CNRS dirigée par Monsieur le Professeur Alain RICHARD, Thème SYMPA (Systèmes de Production Ambiants), Projet SCP (Systèmes Contrôlés par le Produit)

Rattaché à la **61ème section** du CNU.

# 4. CARRIERE

1998 **Maître de Conférences** - UHP Nancy I
(Emploi 61ème section du CNU n°1323)
ESIAL (Ecole Supérieure d'Informatique et Applications de Lorraine)

1997/98 **Ingénieur de Recherche** sous contrat avec le C.N.R.S. (projet ESPRIT IV – IAM Pilot) au Laboratoire Lorrain de Recherche en Informatique et Applications, U.M.R. 7503 **(LORIA)** dirigé par le professeur M. Cosnard, Equipe Model dirigée par le professeur D. Méry.

1996/97 **Ingénieur de Recherche** sous contrat avec le C.N.R.S. (projet ESPRIT III – REMAFEX) au Centre de Recherche en Automatique de Nancy **(CRAN**) dirigé par le professeur M. Véron, Equipe Génie des Systèmes Intégrés de Production dirigée par le professeur G. Morel.

1995/96 **Attaché Temporaire d'Enseignement et de Recherch**e - UHP Nancy I
1994/95 UFR STMIA - Faculté des Sciences.

1991/94 **Ingénieur d'Etudes** pour le financement de la thèse :
- à la Fondation de l'Industrie à l'E.N.S.E.M. dans le cadre d'une contrat avec EDF/DER de Chatou (1991/92)
- sous contrat avec le C.N.R.S. (1992/94) dans le cadre du projet européen ESPRIT III - PRIAM

# 5. SYNTHESE GENERALE DES ACTIVITES DE RECHERCHE ET D'ADMINISTRATION DE LA RECHERCHE

Production scientifique

| | |
|---|---|
| REVUES INTERNATIONALES AVEC COMITE DE LECTURE | **7** |
| REVUES NATIONALES AVEC COMITE DE LECTURE | **4** |
| PARTICIPATION A DES OUVRAGES | **3** |
| COLLOQUES INTERNATIONAUX AVEC COMITE DE LECTURE ET ACTES | **22** |
| COLLOQUES NATIONAUX AVEC COMITE DE LECTURE ET ACTES | **3** |
| COLLOQUES SANS COMITE DE LECTURE AVEC OU SANS ACTES | **12** |
| RAPPORTS DE CONTRATS DE PROJETS EVALUES | **4** |
| RAPPORTS DE CONTRATS INDUSTRIELS | **7** |

Encadrement doctoral et de recherche

| | |
|---|---|
| DOCTORATS D'UNIVERSITE | **2 + 3** |
| DIPLOME D'ETUDES APPROFONDIES ET MASTERS | **6** |

**PEDR** depuis le 1er octobre 2006

Participations à l'administration de la recherche

- Membre élu titulaire de la commission de spécialistes 61ème section de l'Université H. Poincaré depuis 2001
- Membre nommé suppléant de la commission de spécialistes 61/63ème section de l'Université de Metz depuis 2004
- Membre nommé au Conseil scientifique du CRAN depuis 2005

Animation scientifique

- Niveau Local :
  - **Responsable** du projet intitulé « Modèles et Méthodes Formelles pour l'automatisation des processus de production » du groupe thématique « Productique et Automatisation des Procédés Discrets » du CRAN (Contrat quadriennal 2000-2003 UMR 7039 CNRS-UHP-INPL).
  - **Représentant du CRAN** (avec D. Sauter et D. Maquin) au conseil des opérations de l'axe Sécurité et Sûreté des Systèmes (SSS) du Contrat de Plan Etat-Région 2007-2012 « Modélisation, Informations et Systèmes Numériques » (MISN).
  - **Coordinateur** pour le CRAN (avec N. Brinzei) du Centre d'innovation et de démonstration des technologies sûres de fonctionnement (SafeTech) inscrit à la fois dans le cadre du Contrat de Plan Etat-Région 2007-2012 et du Groupement d'Intérêt Scientifique « Surveillance, Sûreté et Sécurité des grands systèmes ».

- Niveau National :
  - **Co-Animateur** avec Bruno Denis (LURPA – ENS Cachan) de l'Action Spécifique CNRS/STIC n°198, RTP PCM 47 « Impact des NTIC en automatisation » (2003-2004).
  - Participation au projet « Reconfiguration des Systèmes à Evénements Discrets » soutenu et financé en 2007 par le GDR MACS. Coordinateur du projet : P. Berruet, Participants : CRAN, LAG, LAGIS, LESTER, LIESP.
  - **Coordinateur** du projet LABIME (LAngage d'expression des Besoins en Informations des Métiers d'Exploitation) labellisé en 2007 par le Groupement d'Intérêt Scientifique en Surveillance, Sûreté et Sécurité des Grands Systèmes (CRAN, ICD, CReSTIC, HeuDiaSyC, LAGIS, LAMIH, LORIA), 2007, Participants au projet : EDF, CRAN, LORIA.

- Niveau International :
  - Membre nommé par la SEE au comité technique « **TC 5.1. Manufacturing Plant Control** » de la société scientifique IFAC pour la période 2005-2008 (Chair: Carlos Eduardo Pereira (BR).

- Participation à des groupes de travail :
  - De 1992 à 1994, participant au groupe du CIAME Actionneurs intelligents De 1995 à 1998, participant au groupe du CIAME Interopérabilité

- De 1992 à 1996, participant au groupe Collaboration CAO Automatique (C2A), GdR Automatique du CNRS
- De 1998 à 2003 : participant au Groupement de Recherche en Productique (GRP) et au groupe de travail **COMPIL** (Commande et Pilotage)
- De 2000 à 2005, participant au groupe de travail **COSED** (Commande Opérationnelle des SED) sous l'égide club EEA (2000-2002) puis du GDR Automatique (2002-2005)
- Depuis 2005, participant au groupe de travail **INCOS** (Ingénierie de la Commande et de la Supervision) du GdR MACS.
- Membre de **l'AFIS** (Association Française d'Ingénierie Système) et de **INCOSE** (International Council on Systems Engineering) depuis 2005. Participation au groupe de travail Sûreté de Fonctionnement de l'AFIS (Association Française d'Ingénierie Système) depuis 2007.

Valorisation scientifique

| | |
|---|---|
| Participation au comité d'organisation de conférences | **3** |
| Organisations de tracks et sessions invités | **2** |
| Organisation de tutoriaux | **1** |
| Evaluation de publications en revues | **8** |
| Evaluation de communications en conférences internationales | **10** |
| Expertise auprès d'organismes de recherche ou dans des projets | **1** |
| Responsable ou participation à des projets évalués | **10** |
| Responsable ou participation à des contrats industriels | **7** |
| Invitation à des jurys de thèse | **1** |

## 6. SYNTHESE GENERALE DES ACTIVITES D'ADMINISTRATION DE L'ENSEIGNEMENT

- Membre élu au Conseil de l'ESIAL depuis 2004
- Membre nommé au Conseil de Perfectionnement de l'ESIAL depuis 2004

- Responsable pédagogique de la spécialisation **SIE** « Systèmes d'Informations d'Entreprises », 30 élèves par promotion de 1999/2000 à 2001/2002
- Responsable pédagogique de la spécialisation **ALSI** « Applications Logicielles pour les Systèmes Industriels » (20 élèves par promotion) depuis 2002/2003

- Responsable des stages de deuxième année (100 élèves) de 1999/2000 à 2002/2003
- Responsable des stages de troisième année depuis la rentrée 2007.

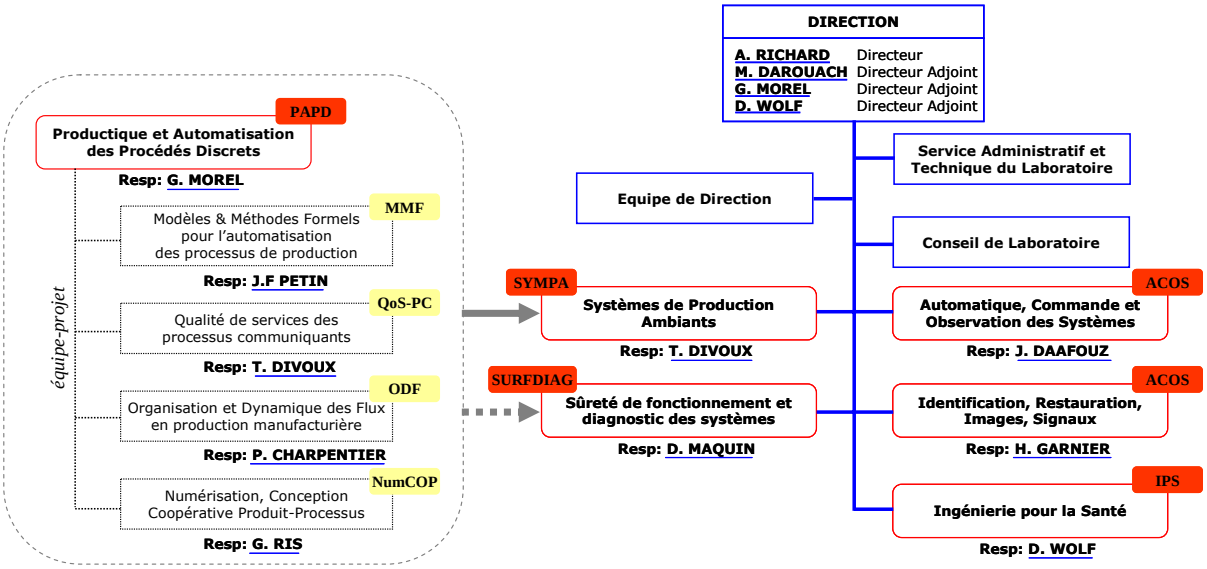
- Responsable de modules et d'unités d'enseignements : à ESIAL (Gestion de production, Ingénierie d'automatisation, Gestion intégrée des entreprises, Progiciels de gestion intégrée, Systèmes réactifs temps réel), en Master Ingénierie Système (Automatique des SED, Systèmes de Planification de la production)

# Introduction

Les travaux de recherche présentés dans ce mémoire ont été effectués au Centre de Recherche en Automatique de Nancy (CRAN), dirigé par le Professeur Alain Richard. Le CRAN, Unité Mixte de Recherche 7039 du Centre National de la Recherche Scientifique (CNRS), de l'Université Henri Poincaré Nancy I (UHP) et de l'Institut Polytechnique de Lorraine (INPL) développe, dans le cadre du quadriennal 2004-2007, cinq groupes thématiques de recherche (Figure 1), eux-mêmes structurés en équipe-projet:

- Sûreté de fonctionnement et diagnostic des systèmes (SURFDIAG), 5 équipes-projets,
- Automatique, Commande et Observation des Systèmes (ACOS), 3 équipes-projets,
- Identification, Restauration, Images et Signaux (IRIS), 2 équipes-projets
- Ingénierie Pour la Santé (IPS), 3 équipes-projets
- Systèmes de production Ambiants (SYMPA), 3 équipes-projets.

Le CRAN reconnaît aussi une équipe de recherche technologique (ERT) dont la thématique porte sur la **TRAC**abilité et l'impact des nouvelles technologies d'**I**dentification et de contrôle des produits sur les modes de gestion des chaînes **LOG**istiques des filières fibres (TRACILOG).




*depuis 2000* *depuis 2003*

Figure 1. Organisation thématique du CRAN (http://www.cran.uhp-nancy.fr)

Nos travaux s'inscrivent, depuis 1998, dans le groupe thématique « Productique et Automatisation des Procédés Discrets », et plus particulièrement dans l'équipe-projet « Méthodes et Modèles Formels pour l'automatisation des processus de production » dont j'ai assuré la responsabilité entre 2000 et 2003 (Figure 1), puis depuis 2003 dans l'équipe-projet « Systèmes Contrôlés par le Produit » (SCP) du groupe thématique SYMPA.

Dans le prolongement de notre formation initiale, nos travaux de recherche présentés dans cette habilitation portent sur la formalisation du processus d'automatisation afin de maîtriser la complexité inhérente à l'intégration d'une part de plus en plus importante de technologies de l'information et de la communication au cœur même des processus de production et des produits.

Automatiser un système de production consiste à définir un ensemble de règles de contrôle et de commande qui satisfasse au mieux les services attendus par les utilisateurs en observant et en agissant de manière optimale sur un ensemble de processus de transformations de matière et d'énergie. D'un point de vue plus formel, le problème d'automatisation peut être généralisé par l'équation de (Fusaoka *et al*. 1983), où *Behavioural Goal* représente le comportement attendu du système automatisé, *Dynamics* caractérise la dynamique du système physique à contrôler impliquant l'ensemble des équipements et machines de production et *Unknow Control Rules* constitue le résultat à obtenir en termes de règles de contrôle et de commande :

$$\textit{Unknown Control Rules} \wedge \textit{Dynamics} \supset \textit{Behavioural Goal (P1)}$$

Notre recherche en automatisation dépasse ce cadre conventionnel de l'Automatique des Systèmes à Evénements Discrets puisqu'il s'agit, par rapport à la mission globale de production d'un produit conforme aux exigences d'un client, de garantir le fonctionnement du système automatisé en ne se focalisant pas que sur la dynamique et le comportement des systèmes de commande mais en considérant également leurs aspects fonctionnels, organisationnels et informationnels en lien avec l'environnement dans lequel ils évoluent (Morel *et al.* 2003). En particulier, la complexité des interactions fonctionnelles, structurelles et dynamiques entre les composants d'un système automatisé joue un rôle fondamental dans la maîtrise de la qualité de services des applications à architecture distribuée que nous avons traitées depuis notre doctorat, que ce soit dans le cadre des projets européens en Actionnement et Mesure Intelligents [P1][P2][P3][P4][P5] visant à l'intégration des fonctions de commande, de maintenance et de gestion de l'information technique d'une installation industrielle, ou dans le cadre du projet « Système Contrôlé par le Produit » visant à conférer au produit un rôle actif dans le pilotage (ordonnancement et commande) de la production en tirant parti des progrès technologiques (RFID, communications sans fils, composants logiciels embarqués, etc).

Ce constat conduit à une généralisation du prédicat P1 où les *Exigences Système,* les *Systèmes de commande* et *Systèmes commandés* sont étendus à des considérations plus générales liées à leur organisation, leur architecture, leur structure informationnelle ou à certaines propriétés telles que l'interopérabilité, la sécurité ou encore la reconfigurabilité :

$$\textit{Système de commande} \wedge \textit{Système commandé} \supset \textit{Exigences Système (P2)}$$

Selon le degré de formalisation des différents termes des prédicats P1 et P2, plusieurs approches, reposant sur la synthèse et/ou l'analyse de modèles de commande, sont envisageables pour garantir le respect et la traçabilité des exigences (opérateur $\supset$) dans un processus sûr d'automatisation.

Dans le cadre théorique des Systèmes à Evénements Discrets (Cassandras & Lafortune 1999, Ramadge & Wonham 1987), nous avons cherché à systématiser la démarche de conception au travers de techniques de synthèse permettant la génération de superviseurs sûrs, réactifs et sans blocages (*Unknown Control Rules* du prédicat P1) à partir de la formalisation des comportements attendus (*Behavioural Goal* du prédicat P1) et de la dynamique du système physique à commander (*Dynamics* du prédicat P1). Nos principales contributions, en particulier dans le cadre de la thèse de D. Gouyon [TH2], ont porté d'une part, sur la proposition d'une démarche itérative et modulaire de synthèse [R2], et d'autre part sur la reconfiguration dynamique, à l'aide des techniques de synthèse, d'un système de contrôle par le produit [R5][R6]. Si les résultats obtenus se sont avérés pertinents, ils restent néanmoins limités aux seuls aspects comportementaux des systèmes à événements discrets et ne couvrent pas l'ensemble du processus d'automatisation (prédicat P2), notamment les activités de définition et d'analyse des exigences.

En effet, la pratique de l'automatisation industrielle, dans le cadre des projets de R&D européens en Actionnement et Mesure Intelligents et des thèses CIFRE de H. El Haouzi (Trane) [TH4] et de D. Evrot (INRS) [TH3], nous a montré les difficultés inhérentes à la formalisation des modèles relatifs à chacun des termes des prédicats P1 et P2 puisque c'est le raffinement concourant de l'ensemble de ces modèles qui permet de faire progressivement émerger une solution. Ce constat justifie l'utilisation conjointe de modèles et méthodes non formelles pour exprimer et raffiner les objectifs assignés au système à automatiser et de modèles plus formels pour concevoir ou implanter une solution appropriée [R1][R7][R8][R9][R10]. Dans ce contexte, l'utilisation de techniques de validation[1] basées essentiellement sur la simulation nous permet d'évaluer la confiance que nous pouvons accorder aux modèles vis-à-vis des besoins exprimés.

Ces travaux, suffisants pour converger vers une solution efficiente dans le cadre de R&D industrielles, ne permettent pas en revanche de garantir que les propriétés attendues par les utilisateurs soient conservées au cours du processus d'automatisation. Pour répondre à ces préoccupations, nous avons donc cherché à définir un cadre collaboratif de modélisation articulant des approches méthodologiques, issues du Génie Automatique, et des techniques formelles, issues du Génie Informatique et de l'Automatique des Systèmes à Evénements Discrets afin de formaliser un processus de construction incrémentale des modèles des exigences et du système automatisé. Cette évolution s'est concrétisée en 1997 par un stage post-doctoral au LORIA dans l'équipe Model du Pr. D. Méry. La méthode B nous a alors fourni le mécanisme formel de raffinement recherché pour permettre la vérification[2] a priori de la correction de chacun des termes du prédicat P2 et leur cohérence globale dans la phase de spécification des exigences d'un système à automatiser [R3][R4]. Des applications de cette approche pour la spécification formelle de systèmes de production ont été développées dans la thèse de P. Lamboley [TH1] et celle en cours de D. Evrot [TH3].

[1] La validation est la confirmation par examen et apport de preuves tangibles que les exigences particulières pour un usage spécifiques sont satisfaites (ISO 8402). Elle répond à la question « *construisons-nous le bon modèle* ?»

[2] La vérification est la confirmation par examen et apport de preuves tangibles que les exigences spécifiées ont été satisfaites (ISO 8402). Elle répond à la question « *construisons-nous correctement le modèle* ?».

Cette évolution de nos travaux relativement à la formalisation du processus d'automatisation a été alimentée par des problématiques spécifiques issues des différents domaines d'applications auxquels nous avons été confrontés et des diverses propriétés que l'on a cherché à garantir (Tableau 1).

| Domaines d'application | Propriété recherchée | Raffinement semi-formel | Raffinement formel | Synthèse |
|---|---|---|---|---|
| Actionnement et Mesure Intelligents | Interopérabilité | Projets PRIAM, EIAMUG, IAM-Pilot | | |
| | Cohérence des spécifications | | Thèse de Patrick LAMBOLEY | |
| Système Contrôlé par le produit | Reconfigurabilité | | | Thèse de David GOUYON |
| | Synchronisation de modèles | Thèse de Hind EL HAOUZI | | |
| Machines industrielles | Sécurité | Thèse de Dominique EVROT | | |

Tableau 1. Positionnement scientifique des projets et thèses codirigées ou en cours

Notre document se présente en onze chapitres.

Le chapitre I introduit, sur la base d'un état de l'art, le contexte de notre recherche relatif au processus d'automatisation. Nous détaillons dans les chapitres II, III et IV nos contributions respectivement à la synthèse de la commande, à la traçabilité des exigences et au raffinement formel de modèles.

Le chapitre V présente notre programme de recherche futur en précisant la problématique scientifique, nos objectifs de recherche et les résultats escomptés.
Nous abordons dans le chapitre VI nos activités d'enseignement et nos responsabilités pédagogiques.

Le chapitre VII conclut ce document en présentant un bilan de nos travaux avant de lister dans les chapitres suivants l'ensemble de notre production scientifique.

Afin de faciliter la lecture, sont recensés à la fin du document les références bibliographiques classées en deux grandes familles (scientifiques et techniques), la liste des acronymes utilisés, la liste des figures et la liste des tableaux.

Enfin, nos cinq publications majeures sont jointes en annexe.

# I Thématique de recherche : présentation & motivations

## 1. INTRODUCTION

Le processus d'automatisation regroupe l'ensemble des activités qui permettent de passer d'un besoin et d'exigences exprimées par les utilisateurs finaux au choix et au développement d'une solution cible (le système à faire). Pour exécuter et organiser toutes ces activités, il est nécessaire de mettre en place un système doté de ressources humaines, techniques et d'informations (le système pour faire) s'appuyant sur un ensemble de méthodes, de modèles, langages et outils proposés en Ingénierie d'automatisation (Figure 2).

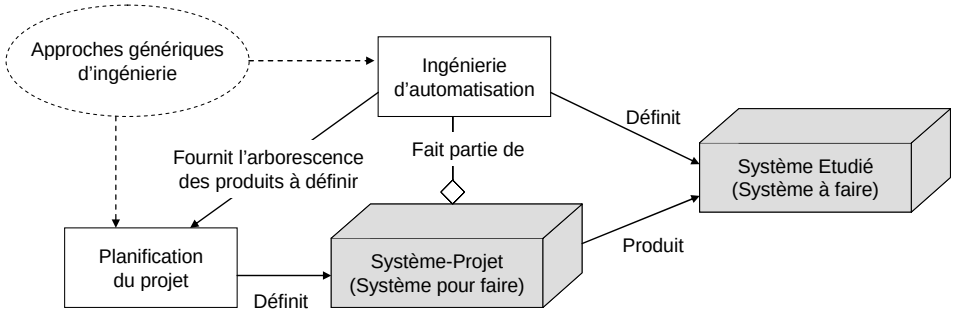



Figure 2. Relations entre Ingénierie d'automatisation, Système à faire et Système pour faire adaptées de (Fiorèse & Ménadier 2007)

Dans ce chapitre, nous analysons tout d'abord l'évolution actuelle des systèmes automatisés vers des **architectures** distribuées basées sur la coopération d'objets logiciels (*système à faire*) afin de répondre aux besoins croissants de flexibilité, d'adaptabilité des entreprises (Hollocks *et al.* 1997), voire de personnalisation des produits (Ollero *et al.* 2002, Nof *et al.* 2006). Cette augmentation de la complexité des architectures des systèmes automatisés se traduit par une difficulté croissante à apprécier et évaluer leurs **propriétés** structurelles et comportementales. Nous abordons ensuite les différentes activités d'un processus d'automatisation (*système pour faire*), depuis la formalisation des exigences jusqu'aux activités d'intégration, de validation, de vérification et de qualification (IVVQ). Enfin, nous présentons un ensemble de modèles et méthodes (*ingénierie d'automatisation*) contribuant à un processus d'automatisation sûr intégrant au mieux les exigences des utilisateurs.

### *1.1 Architectures des systèmes automatisés*

Pour faire face aux besoins croissants de flexibilité, de réactivité des entreprises, voire de personnalisation des produits (Ollero *et al.* 2002, Nof *et al.* 2006), les architectures de production actuelles évoluent vers une réduction du nombre de niveaux décisionnels intégrés de manière rigide autour d'un système d'informations dans les architectures hiérarchiques de type C.I.M. (Kosanke 1995). En effet, ce type d'architecture, même si elle s'est révélée efficace en termes de partage

d'informations ne confère pas aux entreprises l'agilité[3] prônée notamment par la communauté internationale en I.M.S. (Intelligent Manufacturing System) et constitue un frein à la réactivité de la chaîne de décision si l'on tient compte des temps de traitement de l'information opérés à chacun des niveaux. En ce sens, l'évolution des systèmes d'entreprise peut être caractérisée par le modèle E.I.C.M. « Enterprise Integration Capability Model » de (Hollocks *et al.* 1997)(Tableau 2) selon les architectures qu'ils mettent en œuvre : hiérarchisées, intégrées, distribuées (interopérables) et intelligentes au sens donné par (Valckenaers 2001) qui propose de doter chacune des entités du système de production d'une réelle intelligence afin de les rendre autonomes, capables de s'adapter à leur environnement et de coopérer pour satisfaire à la finalité globale.

| | |
|---|---|
| Niveau 5 : Adaptable | Les processus sont capables de s'auto-organiser et de s'adapter à des variations inconnues de leur environnement |
| Niveau 4 : Interopérable | Les processus peuvent coopérer en utilisant les services d'autres îlots pour atteindre leurs objectifs |
| Niveau 3 : Visible | Les processus sont capables de s'informer mutuellement en produisant des données compréhensibles pour leur environnement |
| Niveau 2 : Rigide | Les processus sont coordonnés par des liens rigides |
| Niveau 1 : Fragmenté | Les processus sont cloisonnés et opèrent de façon indépendante |

Tableau 2. Modèle « Enterprise Integration Capability Model » (Hollocks *et al.* 1997)

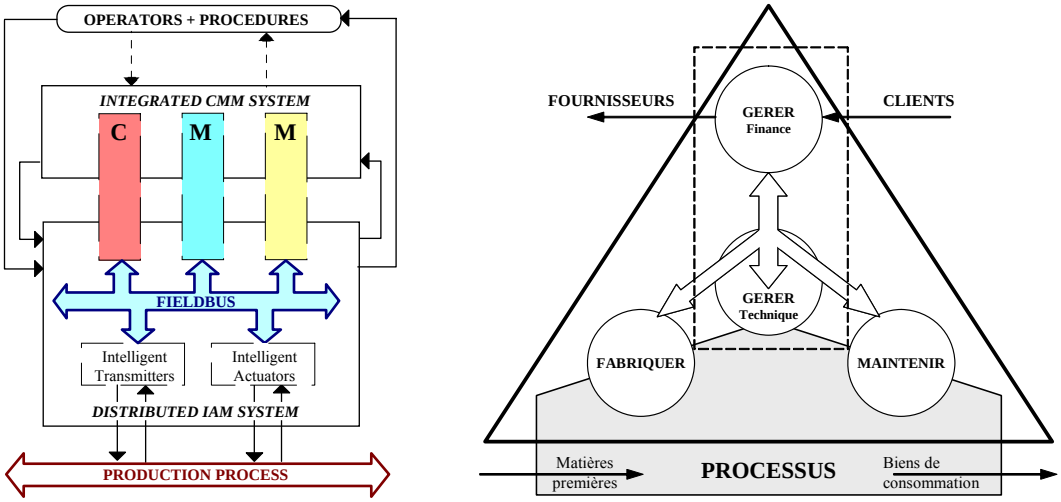
Tirant parti des progrès technologiques dans le domaine de la communication, des automatismes industriels (interfaces ou services Web embarqués dans les automates programmables industriels) ou encore dans les domaines de l'électronique et de l'informatique (RFID, réseaux de capteurs, composants logiciels embarqués, …), l'automatisation a suivi une évolution semblable à celle des architectures d'entreprise qui conduit les systèmes de commande à intégrer une part de plus en plus importante de technologies de l'information et de la communication distribuées au cœur même des processus de production et des produits.

Le défi que doit alors relever une automatisation à l'ère d'Internet consiste à assurer une certaine cohérence entre le monde logique traitant les flux d'informations et le monde réel composé d'objets et de flux physiques – Internet des objets (ITU, 2005) – afin, d'une part, de favoriser la réactivité et la reconfigurabilité des systèmes de contrôle et de commande de la production en réponse aux sollicitations d'un réseau d'acteurs, et, d'autre part, de fiabiliser la prise de décision grâce à des informations parfaitement représentatives de l'état courant des processus industriels.

En ce sens, le déploiement de capacités de traitement, de mémorisation et de communication de l'information embarquées dans les capteurs et actionneurs des équipements industriels a conduit, au début des années 90, à des architectures intégrées de commande, de maintenance et de gestion technique (Iung 1992, Iung *et al.* 2001)[R1]. En prolongement des travaux visant à intégrer la commande et la surveillance des installations industrielles (Vogrig *et al.* 1987, Lhoste 1994, El Khattabi 1993, Zamai *et al.* 1998), cette distribution de l'intelligence au plus près du

[3] Agility relates to the interface between the company and the market. Essentially, it is a set of abilities for meeting widely varied customer requirements in terms of price, specification, quality, quantity and delivery. (Katayama & Bennet 1997).

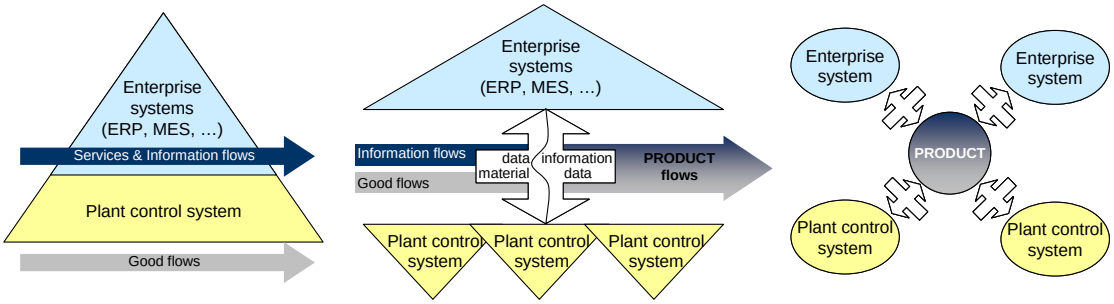
terrain (Wright & Bourne 1988, Iung 1992) avait pour objectif d'augmenter la pertinence et la crédibilité des informations fournies aux automatismes et/ou aux opérateurs humains par les chaînes d'actions et d'observations.
L'architecture résultante (Figure 3a) met en œuvre un ensemble de traitements d'actionnement et de mesure – incluant non seulement la commande mais également la surveillance, la reconfiguration de l'équipement et la production d'informations d'état fiables – embarqués dans les actionneurs et capteurs communiquant via un réseau de terrain avec des équipements périphériques de contrôle (C), de maintenance (M) et de gestion technique (M *pour technical Management*) offrant des services d'évaluation et d'optimisation de la production (disponibilité, coûts, délais, …). D'un point de vue fonctionnel, la gestion technique assure le lien entre les fonctions opérationnelles de production (commande et maintenance) et la gestion financière (Figure 3b) et préfigure ainsi l'émergence des M.E.S. (Manufacturing Execution System) qui prennent en charge l'interface entre les système de gestion d'entreprise de type E.R.P. et les applications de commande et de contrôle des installations de production.




a. Vue technologique b. intégration des fonctions CMM

Figure 3. Architecture de Commande, Maintenance et Gestion Technique intégrée

Le concept de « système contrôlé par le produit » constitue une étape supplémentaire dans la distribution de l'intelligence puisqu'il préconise d'embarquer des capacités informationnelles, décisionnelles voire opérationnelles dans le produit afin de lui attribuer un comportement actif dans le pilotage (ordonnancement et commande) de la production.
L'objectif est double. Il s'agit d'une part, de favoriser la reconfiguration dynamique des processus de commande en embarquant dans le produit une partie des traitements nécessaires à la commande des ressources de transformation et à son routage dans le système de production afin de répondre à des besoins de personnalisation (Da Silveira *et al.* 2001). D'autre part, l'objectif est d'assurer la synchronisation, par le produit, entre les flux d'informations manipulées par les systèmes d'entreprise (ERP, SCM, APS, MES, …) et les flux physiques transformés par le système de production (Figure 4b) en considérant le produit à la fois comme un bien matériel et comme un fournisseur de services. Ces flux ne sont, en effet, généralement pas synchrones, la mise à jour des informations relatives aux produits (lots, quantités produites, quantité consommées, temps passés, …) étant souvent réalisée après coup (Figure 4a).




a. intégration b. interopérabilité c. agilité

Figure 4. Du contrôle intégré au contrôle agile par le produit [R6]

Au-delà de l'impact technologique et normatif (IEC 62264, B2MML[4], technologies XML, EAI[5], Web-services embarqués dans les automates programmables industriels, etc) que souligne la Figure 5, l'intégration des automatismes aux systèmes de pilotage de la production (MES), voire aux systèmes de gestion d'entreprises (ERP, APS, SCM) conduit à étendre leurs fonctions traditionnelles de contrôle et de commande des installations industrielles à des fonctions de gestion technique aptes à fournir les informations requises (surveillance produit/procédé, enregistrement des états machines et opérateurs, traçabilité produit, etc).

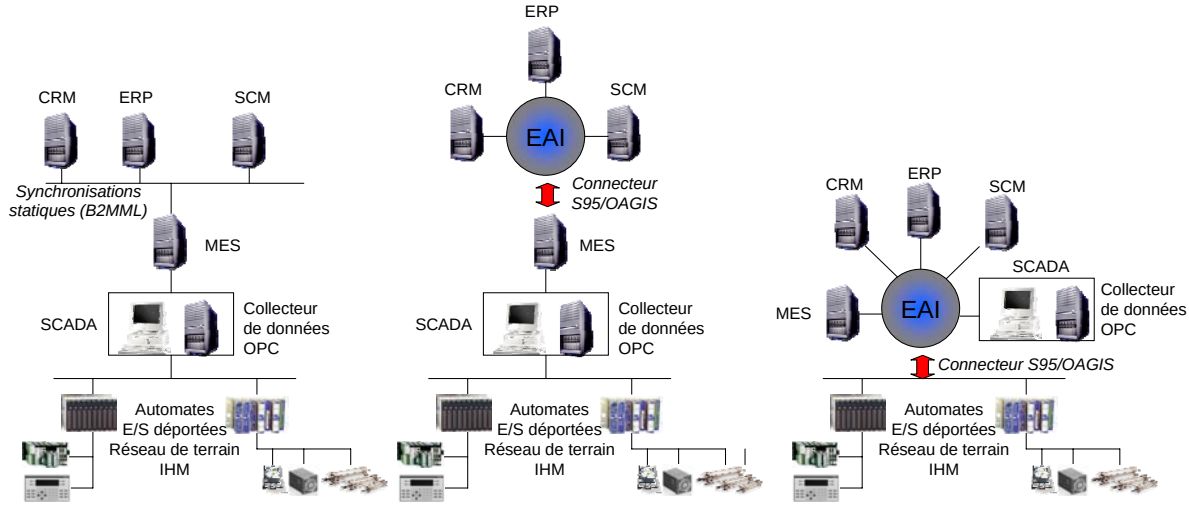



Figure 5. Evolution des architectures de contrôle et de commande [O2]

Enfin, cette évolution conduit au concept de Holonic Manufacturing System (Deen 2003) dans lequel les holons, possédant une composante physique et une composante informationnelle, agissent comme des entités individuelles autonomes qui coopèrent pour fabriquer des produits dans un environnement dynamiquement reconfigurable (McFarlane & Bussmann 2000) (Figure 4c). En ce sens, le concept de « Système contrôlé par le produit » est à rapprocher de la notion d'holon-produit, définie par l'architecture PROSA[6] (Valckenaers 2001), assurant la gestion des informations et décisions relatives au produit. Il est à noter que les applications actuelles de ces nouvelles organisations de production se situent essentiellement au niveau pilotage et peu au niveau temps réel (Lazansky *et al.* 2001).

[4] B2MML : Business to Manufacturing Markup Language
[5] EAI : Enterprise Application Integration
[6] PROSA : Product-Resource-Order-Staff Architecture

## *1.2 Propriétés des systèmes automatisés*

Définition
*La propriété est une qualité propre, une caractéristique intrinsèque (fonctionnelle, comportementale, structurelle ou organique, dépendante du temps ou non) que doit posséder une entité. Toute propriété traduit une attente, une exigence, une finalité à laquelle l'entité doit répondre.* (Chapurlat 2007)

Parmi l'ensemble des propriétés pouvant caractériser un système automatisé de production, nous pouvons citer les propriétés fonctionnelles (aptitude à répondre à ses missions), comportementales et dynamiques (réactions du système à des sollicitations de son environnement), ou encore relatives à ses performances, sa robustesse, son architecture, sa qualité (respect des normes en vigueur).

La section précédente a montré que la pénétration des technologies de l'information et de la communication au cœur des systèmes automatisés de production conduit à des architectures complexes constituées d'un ensemble de composants de nature hétérogène, relevant de domaines de connaissances différents et transdisciplinaires. Dans ce contexte, les propriétés d'un système ne sont en général pas réductibles aux propriétés de ses constituants pris isolément (objets logiciels de commande, de supervision, équipements de commande, équipements de transformation de matière et d'énergie, etc). Elles émergent, pour la plupart, d'un réseau d'interactions entre les constituants du système qui peut être à l'origine de comportements tant intentionnels que non intentionnels (Figure 6).

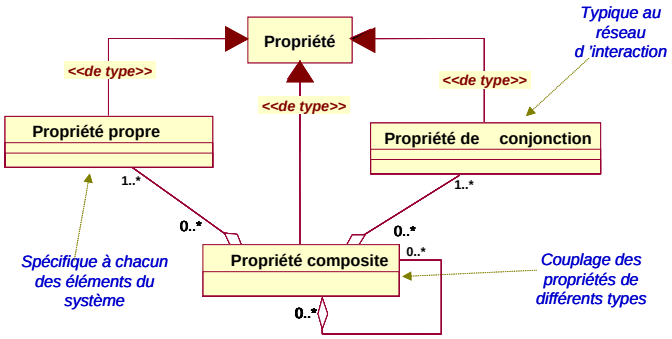



Figure 6. Typologie des propriétés d'un système (Chapurlat, 2007)

Ces interactions entre constituants, souvent néfastes, difficiles à prévoir et à maîtriser, peuvent entraîner une diminution de la disponibilité des systèmes de production, et par là même de leur efficacité (Johnson 2004).

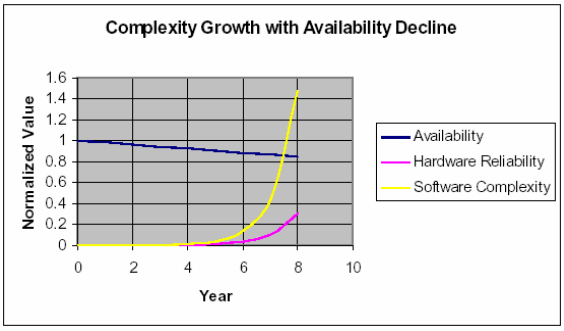



Figure 7. Scénario d'évolution de la complexité et de la disponibilité (Johnson 2004)

La complexité ainsi engendrée peut donc constituer un frein aux progrès en automatisation (Figure 7) qui se traduit par une difficulté croissante à apprécier et évaluer les propriétés des systèmes, notamment **comportementales** (relatives à la combinatoire des parcours possibles dans les scénarios de fonctionnement) et **structurelles** (relatives aux architectures, aux composants et leurs interfaces).

Les propriétés **comportementales** caractérisent la dynamique des systèmes et leur aptitude à délivrer, dans les pires conditions, des services conformes aux exigences. Parmi ces propriétés, nous pouvons citer :

- les propriétés de **sûreté** qui caractérisent le fait qu'une situation donnée (situation indésirable) ne peut se produire à un instant donné ou inversement qu'une condition (situation désirée) est toujours réalisée dans un état donné ;
- de **vivacité** qui caractérise le fait qu'une situation désirée pourra toujours être atteinte dans un état futur,
- ou encore de **sécurité** qui caractérise l'innocuité du système par rapport à son environnement et en particulier vis-à-vis de ses utilisateurs.

Ces propriétés peuvent être qualifiées **d'invariantes** si leur véracité est indépendante du temps ou de l'état du système, de **temporelles** si leur véracité varie en fonction du temps, ou **événementielles** si elles caractérisent la réponse du système à un stimuli. Comme indiqué précédemment, les propriétés comportementales caractérisant un système automatisé résultent des propriétés intrinsèques aux composants et à leurs interactions (composants logiciels de commande, circuits électromécaniques ou électropneumatiques, actionneurs et capteurs, etc).

Les propriétés **structurelles** caractérisent les architectures des systèmes. En particulier, dans le cadre des systèmes de contrôle et de commande à architecture distribuée, la coopération entre les traitements répartis dans les différents équipements (entrées/sorties déportées, actionneurs et capteurs intelligents, produits intelligents, systèmes de commande en réseau, etc) autour de multiples médias de communication suppose entre autres, selon le canevas proposé par (Sematech 1995) :

- une **interopérabilité** de communication, dite de classe A, où l'échange d'informations entre équipements repose sur le respect d'un protocole de communication et se matérialise par la définition de profils et normes d'accompagnement (EN 50 170) pour les actionneurs et capteurs,
- une **interopérabilité** de services, dite de classe B ou interfonctionnement selon (Thomesse 1999)(IEC 61804), qui impose la complémentarité des services (fonctions, informations, comportement) rendus par les équipements,
- une **interopérabilité** de classe C ou interchangeabilité caractérisant la substitution d'un équipement par un autre équipement supportant des services et des performances similaires ainsi qu'une compatibilité physique.

Cette dernière propriété sous-entend une notion d'adaptabilité des équipements de commande qui préfigure la propriété plus générale de **reconfigurabilité** qui va au-delà de la notion classique de flexibilité (Tsubone & Horikawa 1999). En effet, la flexibilité, définie comme l'aptitude à s'adapter à différentes règles de production, repose généralement sur la capacité des composants à exécuter différentes tâches prédéfinies ou sur les redondances fonctionnelles entre composants. La reconfigurabilité caractérise, quant à elle, la capacité des systèmes à modifier leur architecture et/ou les comportements de ses constituants (modification des règles de

commande par commutation de règles préalablement définies, par appel de composants stockés en bibliothèque ou par génération de nouveaux composants spécifiques) afin de s'adapter à un nouveau type de production ou à une production en présence de défaillances (Mehrabi *et al.* 2000).

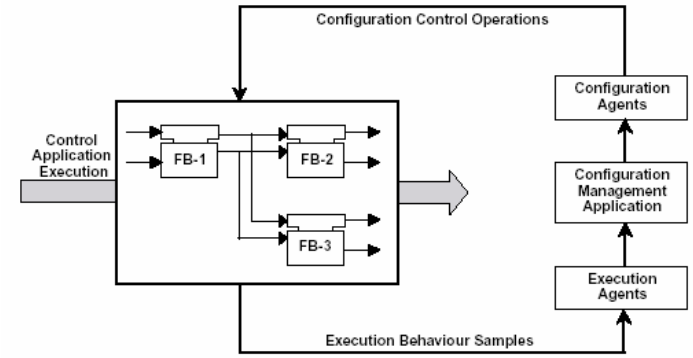



Figure 8. Boucle de reconfiguration (Xu *et al.* 2002)

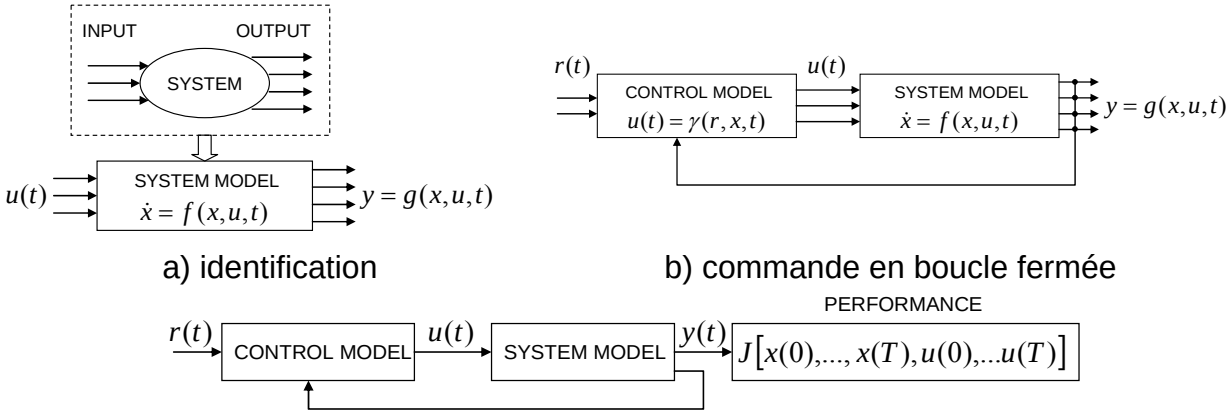
Selon que la reconfiguration soit réalisée « à froid » (hors ligne) ou « à chaud » (en ligne), qu'elle soit initiée par le système lui-même ou par un acteur extérieur, elle sera qualifiée par (Xu *et al.* 2002) de reconfiguration simple (modifications « à froid » initiées par un acteur extérieur), de reconfiguration dynamique (modifications « à chaud » initiées par un acteur extérieur) ou de reconfiguration intelligente (modifications « à chaud » initiées par le système lui-même). Cette dernière revient à introduire une boucle de contrôle ou plutôt de reconfiguration (Figure 8) impliquant des activités de surveillance et de diagnostic (*Execution Agents*) afin de définir quand et où une reconfiguration est exigée, des activités d'élaboration de la stratégie de commande la plus appropriée (*Configuration Management Application*), et enfin, des activités d'implantation de la nouvelle configuration (*Configuration Agents*).
Le développement d'un système automatisé respectant au mieux l'ensemble de ces propriétés repose sur un certain nombre de préconisations méthodologiques relatives au **processus d'automatisation**, afin d'encadrer les phases de spécification, de conception, de codage ou encore de vérification et de validation jusqu'à l'obtention d'une solution admissible voire optimisée, ainsi qu'aux **modèles et méthodes** formelles ou semi-formelles permettant de mesurer quantitativement et qualitativement les résultats de ce processus.

## 2. SYSTEME POUR FAIRE : LE PROCESSUS D'AUTOMATISATION

Le processus d'automatisation est un ensemble d'activités corrélées qui conduisent à la réalisation d'un système répondant aux finalités qui lui sont assignées. C'est donc un processus qui transforme des données d'entrées (besoins et contraintes des parties prenantes) en résultats de sortie (système à faire). Il peut se décomposer en projets intermédiaires successifs : spécification, conception, réalisation, etc.

Dans le cadre relatif à la modélisation des systèmes proposé par (Cassandras & Lafortune 1999), le processus conduisant à l'obtention des modèles de commande est décrit selon cinq étapes : une phase d'identification du système réel qui permet de définir un modèle représentant le comportement et la nature des transformations physiques de matière et d'énergie (Figure 9a), une phase de conception de l'architecture de commande suivie par une phase de définition des règles et

paramètres de commande (Figure 9b), une phase d'évaluation qui permet de s'assurer que le comportement résultant de l'interaction entre le modèle du système et la commande est conforme au comportement attendu sous certaines conditions de fonctionnement données (Figure 9c), et enfin, une phase d'optimisation qui permet de faire évoluer la commande du système vers une performance optimale.




a) identification

b) commande en boucle fermée

c) évaluation des performances

Figure 9. System theory selon (Cassandras & Lafortune 1999)

Il est à noter que cette approche se focalise sur l'élaboration d'un modèle comportemental de la commande et réduit, en cela, la notion de « système » au seul système à commander. Les cycles de développement classiques, tels que le cycle en V (Figure 10)(Frachet 1987, Calvez 1990), constituent une généralisation de ce processus en intégrant les phases relatives à l'expression du besoin (spécification), à la validation (test, recette) et en étendant la notion de « système » à l'ensemble des constituants du « système commandé » (partie commande, ensembles matériels et physiques, composants électromécaniques ou électropneumatiques, etc.).

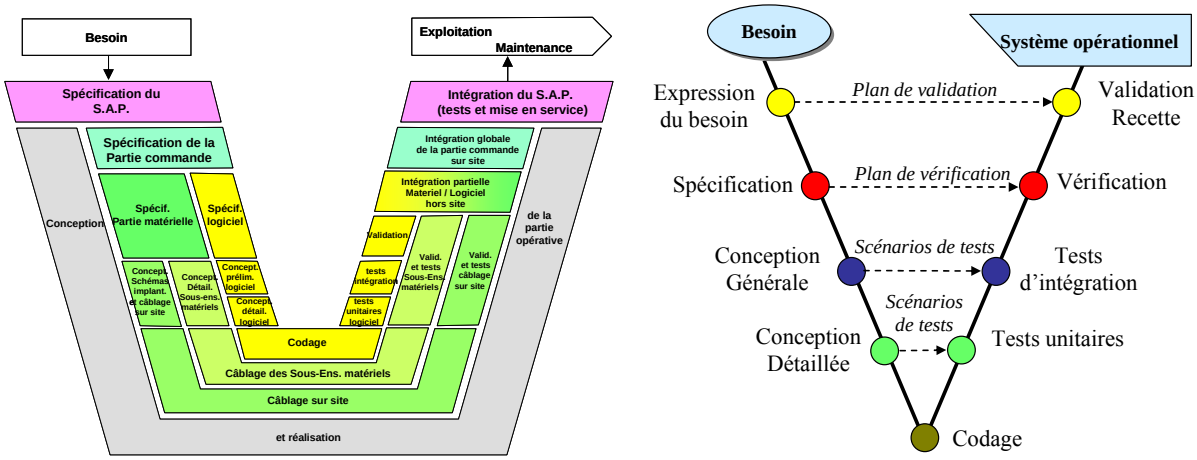



Figure 10. Cycles de vie en automatisation

Ces approches présentent l'avantage de proposer un cadre structurant le processus d'automatisation mais la succession des différentes activités qu'elles proposent peut apparaître comme trop contrainte dans la mesure où :

- leur application au développement de systèmes complexes à multiples niveaux de décomposition nécessite la juxtaposition de plusieurs cycles de

vie, les activités de conception d'un composant appelant les activités de définition des exigences de ces sous constituants (Forsberg *et al.* 2005),
- les activités de validation et de vérification commencent dès l'expression du besoin et ne sont pas limitées à la branche gauche du cycle en V, considérée comme postérieure aux activités de codage et de réalisation,
- ces approches reposent sur une démarche descendante partant du besoin jusqu'à la réalisation qui souvent, dans la pratique industrielle, doit être associée à une démarche ascendante basée sur la construction progressive d'une solution à partir de l'assemblage de composants existants (souvent appelé *COTS Components On The Shelves*).

D'autres approches ont été proposées, telles que le cycle en spirale (Boehm 1988), le « dual vee cycle » de (Forsberg *et al.* 2005) voire les méthodes agiles (XP eX*trem Programming*) mais nous retiendrons les approches en ingénierie système (Sheard 2006), supportée par les normes en Ingénierie Système (ISO 15288), qui rompt avec la notion de cycle (et donc de séquences d'activités) en la remplaçant par la notion de **processus techniques** qui coopèrent à la réalisation d'un produit (ici le système automatisé) et qui recouvrent les phases des cycles de vie présentées précédemment (Figure 11).

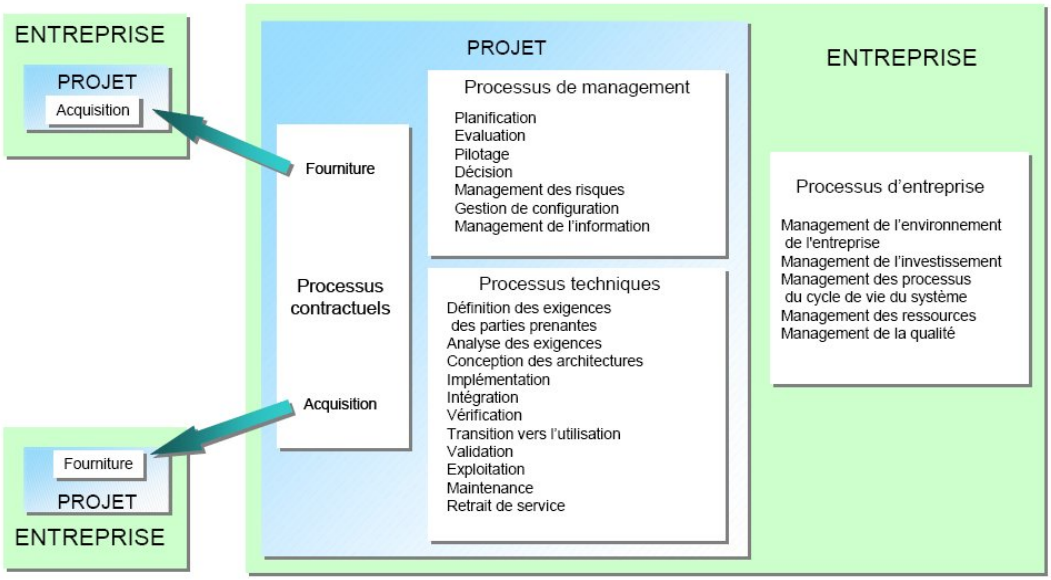


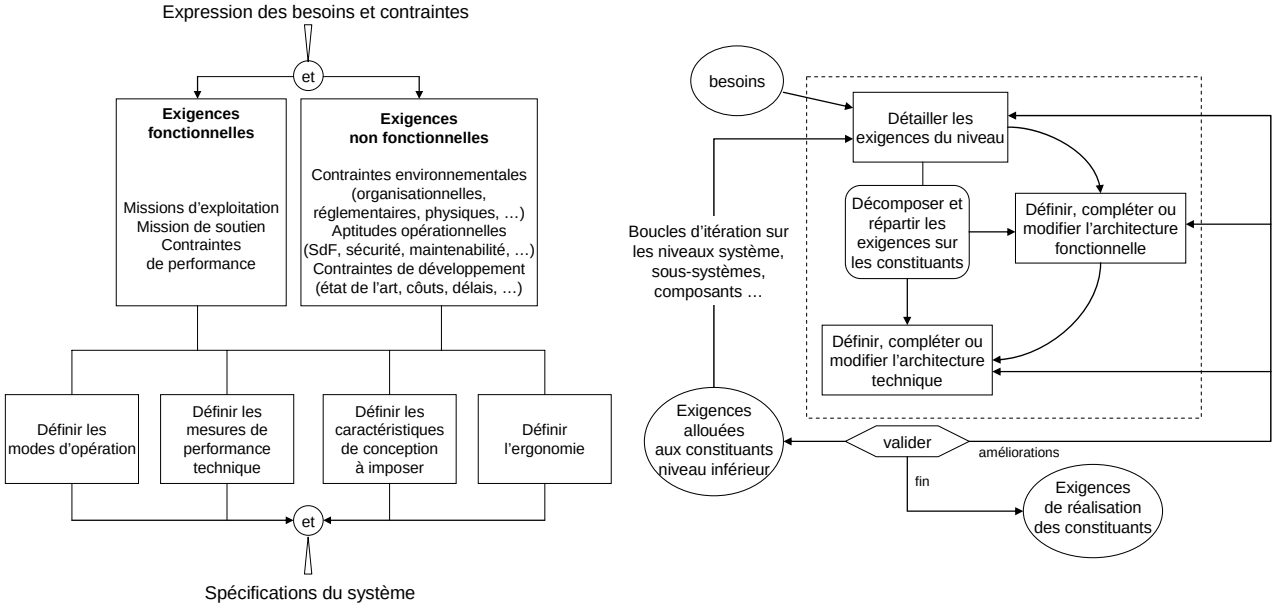

Figure 11. Cartographie des processus du cycle de vie du système (ISO 15288)

Les paragraphes suivants présentent une vue détaillée des principaux processus techniques dits de développement dans la norme ISO 15288 (Figure 11) impliqués en automatisation de système, à savoir les processus de définition et d'analyse des exigences, les processus de conception et les processus de vérification & validation.

### *2.1 Processus de spécification et d'analyse des exigences*

Le processus de définition et d'analyse des exigences repose sur l'expression, sous forme d'un cahier des charges, des besoins et contraintes des parties intéressées (utilisatrices et exploitantes). Ils s'expriment en termes d'objectifs, de missions ou encore de performances, assignés au système à développer au travers de scénarios d'utilisation, de fonctions ou services attendus ou encore de cas d'utilisation.

Afin de garantir l'adéquation de la solution développée à l'ensemble des besoins et contraintes, le processus de définition et d'analyse des exigences a pour objectif de produire des **modèles prescriptifs** raffinant l'ensemble des besoins définis dans le cahier des charges en vérifiant leur faisabilité, en les hiérarchisant et en ajoutant les exigences des parties prenantes concernées par le développement de la solution. Ces modèles peuvent être formalisés sous la forme d'exigences fonctionnelles traduisant le besoin (ce que doit faire le système) et d'exigences non fonctionnelles traduisant des contraintes imposées au système (Figure 12a).




a) typologie des exigences (IEEE 1220) b) boucle de spécification des exigences (Ménadier 2002)

Figure 12. Spécification des exigences

Définition

*Une **exigence** est un énoncé qui prescrit une fonction, une aptitude ou une caractéristique auxquelles doit satisfaire un produit ou un système dans un contexte donné. L'ensemble cohérent et complet des exigences est appelé **spécification** notamment lorsqu'il est validé et agréé par les parties prenantes.*

Il est à noter que la spécification des exigences est considérée comme un processus continu et itératif dans la mesure où les choix de conception peuvent générer de nouvelles exigences globales à prendre en compte au niveau « système » ou détaillées, résultant de l'allocation (ou de la répartition) des exigences « système » sur les sous fonctions et constituants (Figure 12b). Spécifier un système automatisé de production impose donc de définir et d'analyser l'ensemble des exigences tant au niveau « système » qu'au niveau de l'ensemble de ses constituants. Le prédicat P2 peut ainsi s'interpréter en termes de spécification comme :

$$\textit{Spécification système de commande} \wedge \textit{Spécification Système commandé} \supset \textit{Spécification des Exigences Système}$$

et formalise ainsi le problème relatif à la cohérence de ces spécifications.

### *2.2 Processus de conception*

Le processus de conception d'un système automatisé a pour objectif d'élaborer un ensemble de **modèles constructifs** (par opposition aux spécifications de nature prescriptive) décrivant, selon de multiples points de vue, une solution répondant aux exigences auxquelles le système doit répondre.

Les méthodologies nécessaires à l'obtention de ces modèles ont fait l'objet de nombreux travaux et de nombreuses classifications (Frachet 1987, Calvez 1990, Morel 1992, Denis 1994, Ménadier 2002). Parmi ces dernières, nous retiendrons :

- l'élaboration des modèles d'architectures **fonctionnelles** et **logiques**, décrivant respectivement le système automatisé comme une coopération de fonctions et de constituants (ou modules fonctionnels) en interaction. Parmi ces modèles, nous pouvons mentionner :
    - **structurels** décrivant l'agencement des fonctions et des modules logiques ainsi que la sémantique des flux informationnels et physiques qu'ils transforment ou s'échangent,
    - **comportementaux** décrivant l'enchaînement des éléments logiques identifiés dans les modèles structurels ainsi que les réactions de ces éléments logiques (élaboration des variables de sortie) en réponse aux stimuli auxquels ils sont soumis,
    - **informationnels** décrivant les structures de données sur lesquelles s'appuient les fonctions et constituants du système,
- l'élaboration de modèles d'architectures **physiques** décrivant les modèles d'architecture du système automatisé comme l'assemblage d'un ensemble de constituants technologiques (équipements de traitements, de stockage ou de communication de l'information, circuits électropneumatiques et électromécaniques, équipements de transformation de matière et d'énergie, actionneurs, capteurs, etc.)

Les modèles d'architectures fonctionnelles et logiques sont, en principe, invariants par rapport aux choix technologiques. En effet, les fonctions et modules fonctionnels qu'ils décrivent devront être alloués à des équipements de l'architecture physique. Ces équipements peuvent alors être considérés comme des supports à la réalisation des modules fonctionnels (automate programmable par exemple) ou comme des composants sur étagère (*COTS*) offrant parfois plus de fonctions que celles utilisées dans les modèles d'architecture logiques, ce qui peut induire des risques de comportements non voulus ou non maîtrisés. Ce problème est particulièrement crucial dans le cas de systèmes automatisés à architecture distribuée résultant de l'assemblage de composants hétérogènes provenant de constructeurs différents (actionneurs et capteurs intelligents par exemple) ou développés en interne par les acteurs impliqués dans le processus d'automatisation.

### *2.3 Processus de vérification & validation*

Le processus de validation et de vérification a pour objectif de s'assurer de l'adéquation des modèles et du système développé vis-à-vis des besoins exprimés par les utilisateurs et de la spécification des exigences.

Définition
*La **validation** est la confirmation que par examen et apport de preuves tangibles que les exigences particulières pour un usage spécifiques sont satisfaites. Elle répond à la question « construisons-nous le bon modèle ? ». (ISO 8402)*

En d'autres termes, la validation cherche à s'assurer que les productions de chacun des processus de spécification, de conception ou d'implémentation (modèles et système réalisé) sont conformes aux besoins exprimés par les utilisateurs (Figure 13). Les besoins étant le plus souvent exprimés par les utilisateurs de manière non formelle (cahier des charges textuel), les techniques de validation reposent souvent sur l'exécution symbolique des modèles (simulation et/ou émulation) et sur l'exécution de scénarios de tests sur le système réel. Les propriétés que l'on cherche à démontrer ont trait :

- à la **complétude** des modèles, en particulier de la spécification des exigences, afin de s'assurer qu'ils contiennent toute l'information nécessaire pour couvrir l'ensemble des besoins exprimés,
- à la **pertinence** des modèles et du système réalisé, afin de s'assurer qu'ils couvrent bien les attentes des utilisateurs.

Notons que la validation de la complétude reste dans la plupart des cas un vœu pieux dans la mesure où, outre le fait que les oublis objectifs et subjectifs ne peuvent être totalement évités dans la pratique, la notion de modèle est implicitement liée à celle de point de vue filtrant toute représentation de la réalité. La représentation des besoins ne peut alors plus être envisagée qu'au travers de plusieurs modèles dont les formalismes ne sont pas forcément compatibles et à partir desquels il n'est pas toujours aisé de construire une représentation commune de l'information.

Définition
*La **vérification** est la confirmation par examen et apport de preuves tangibles que les exigences spécifiées ont été satisfaites. Elle répond à la question « construisons-nous correctement le modèle ?». (ISO 8402)*

En d'autres termes, la vérification cherche à s'assurer que les modèles élaborés au cours des processus de spécification et de conception sont conformes à l'ensemble des exigences spécifiées (Figure 13). Les propriétés que l'on cherche à démontrer ont trait à :

- la **correction** des modèles afin de s'assurer qu'ils respectent les exigences de construction syntaxique et sémantique,
- la **cohérence** des modèles afin de s'assurer qu'il n'existe pas de contradiction dans l'information contenue à l'intérieur d'un ou plusieurs modèles ; on cherchera par exemple à détecter la présence d'informations contradictoires dans la spécification des exigences mais également entre les exigences spécifiées et les modèles de conception.

Les techniques de vérification reposent sur l'utilisation de techniques d'analyse formelle reposant sur des formalismes dotés d'une sémantique mathématique permettant de prouver leur correction et leur cohérence vis-à-vis de propriétés exprimées de manière formelle.

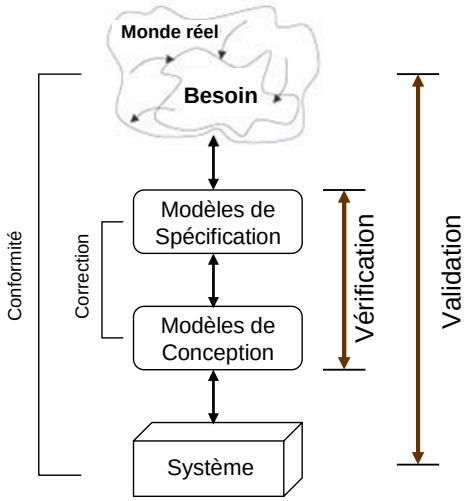



Figure 13. Processus de validation & vérification adapté de (Kamsu Foguem 2004)

Il est à noter que les activités de validation et de vérification peuvent couvrir l'ensemble des phases représentées dans les cycles de développement classiques et ne sont pas limitées, en particulier, à la branche gauche du cycle en V dans la mesure où elles s'appliquent aussi bien à l'analyse des modèles produits par les processus de spécification et de conception qu'à l'analyse du système réalisé. Il est à noter que cette vision diffère quelque peu du processus IVVQ[7] (Intégration, Vérification, Validation, Qualification) proposé notamment par l'AFIS (Association Française d'Ingénierie Système) qui s'applique quant à lui au système réalisé (branche gauche du cycle en V).

## 3. MODELES & METHODES EN INGENIERIE D'AUTOMATISATION

Les processus de spécification, de conception, d'implémentation, de validation & vérification présentés dans la section précédente s'appuient sur des méthodes, modèles, langages et outils pour supporter l'ensemble de leurs activités. Si l'on considère les prédicats d'automatisation (Fusaoka *et al*. 1983) présentés dans l'introduction de ce mémoire et que tout processus d'automatisation doit chercher à satisfaire, les méthodes, modèles et outils peuvent se répartir selon trois familles :

- les formalismes de modélisation de chacun des trois termes du prédicat sont formels[8], homogènes (mêmes principes de modélisation, langages formels dont la relation sémantique est connue) et permettent d'envisager la génération automatique (ou **synthèse**) d'un terme du prédicat (modèles de commande) à partir des modèles formels des deux autres termes (modèles des exigences et du système opérant à automatiser) ;

[7] Intégration : Opération d'assemblage et de tests (tests unitaires et tests d'intégration) permettant de s'assurer que la construction d'un produit ou d'un système s'effectue conformément aux données issues de la conception
Qualification : Décision prise par l'acquéreur, après une période probatoire d'exploitation en conditions réelles ou représentatives, démontrant la satisfaction du besoin réel.

[8] Langage formel : langage doté d'une sémantique mathématique basée sur des règles d'interprétation qui garantissent l'absence d'ambiguïté dans les descriptions produites et des règles de déduction permettant de raisonner sur les modèles afin de découvrir de potentielles incomplétudes, inconsistances ou pour prouver des propriétés.

- les formalismes de modélisation de chacun des trois termes du prédicat sont formels, homogènes (mêmes principes de modélisation, langages formels dont la relation sémantique est connue) et permettent de mettre en œuvre un processus de **vérification** de leur correction et de leur cohérence (modèles des exigences, de commande et du systèmes opérant à automatiser),
- enfin, les formalismes de modélisation de chacun des trois termes du prédicat sont propres aux différents domaines de modélisation, certains d'entre eux sont non formels ou semi-formels (souvent les modèles des exigences), ce qui ne permet pas de garantir le respect du prédicat autrement que par la définition d'un cadre méthodologique fixant la logique d'obtention des modèles, la définition de règles sémantiques ou normatives permettant l'interopérabilité des modèles, et in fine, leur **validation** par simulation.

Les trois sections suivantes présentent les méthodes et modèles permettant d'aborder le prédicat d'automatisation sous l'angle de la synthèse et de l'analyse (validation et vérification).

### *3.1 Méthodes et Modèles de synthèse de la commande*

La théorie de la supervision (Supervisory Control Theory, SCT) (Ramadge & Wonham 1987) fournit un cadre formel de modélisation des S.E.D. Il repose sur la notion de *procédé*, considéré comme un ensemble de processus de transformations physiques, et supposé émettre des évènements de manière spontanée. Le procédé est modélisé par un automate non temporisé G = (X, E, f, x0, Xm) qui représente tous les états possibles que peut atteindre le procédé sans contrôle et où :

- X est l'ensemble des états
- E est un alphabet d'entrée
- f est la fonction de transition d'états définie de $X \times E \rightarrow X$ qui associe un état de départ et un symbole d'entrée à un état d'arrivée
- $x_0$ est l'état initial
- $X_m$ est l'ensemble des états finaux ou états marqués ($X_m \subseteq X$)

En partant du principe que G ne respecte pas toujours naturellement les spécifications comportementales décrivant le fonctionnement général du procédé automatisé, il faut restreindre le comportement du procédé à un sous langage L(G) en interdisant ou en autorisant certains événements afin de le maintenir dans un ensemble d'états compatibles avec la spécification. Afin de pouvoir agir sur le comportement du procédé et le maintenir dans un espace d'états admissibles par une spécification donnée, la théorie introduit un *superviseur* S qui a pour mission d'autoriser ou d'interdire certains de ces évènements. Les interactions entre procédé et superviseur induisent les notions de contrôlabilité et d'observabilité. Ainsi, l'alphabet E du procédé G sera en fait partitionné en deux sous-ensembles disjoints $E = E_c \cup E_{uc}$ où :

- $E_c$ est l'ensemble des événements contrôlables, c'est-à-dire des événements que S est susceptible d'autoriser ou d'interdire
- $E_{uc}$ est l'ensemble des événements incontrôlables, c'est-à-dire des événements sur lesquels S ne peut pas agir.

Formellement, le superviseur est une fonction $S : L(G) \rightarrow 2^E$ qui pour chaque séquence $s \in L(G)$, retourne l'ensemble des séquences autorisées (Figure 14). Etant

donné un procédé G et un superviseur S, le comportement résultant du système en boucle fermée, noté S/G, est aussi un système à événements discrets qui peut être décrit par un automate dont le langage est défini de matière récursive par :

- $\varepsilon \in L(S/G)$
- $[(s \in L(S/G)) \text{ and } (s\sigma \in L(G)) \text{ and } (\sigma \in S(s))] \Leftrightarrow [s\sigma \in L(S/G)]$

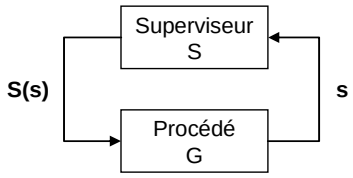



Figure 14. Boucle de contrôle par supervision

Les algorithmes de synthèse (Wonham & Ramadge 1987, Kumar *et al.* 1991) ont pour objectif de définir le superviseur S à partir de l'automate de procédé G et d'un automate $H_{spec}$ représentant un ensemble de contraintes sur certains événements de G (les alphabets de G et Hspec présentent une intersection non nulle). Ces algorithmes reposent principalement sur :

- la composition ou le produit synchrone des automates décrivant les modèles de procédés et de spécifications,
- la détermination des états défendus : un état défendu est tel qu'il existe un événement non contrôlable admis dans le modèle du procédé, mais interdit dans la spécification,
- la détermination des états faiblement défendus définis comme des états tels qu'il existe une séquence d'événements incontrôlables dans le produit synchrone procédé $\times$ spécification qui conduit à un état défendu,
- et enfin élimination des états défendus et faiblement défendus pour aboutir au superviseur.

A noter que, compte tenu de la difficulté à représenter la fonction S (il est difficile en pratique de lister S(s) pour tout $s \in L(S/G)$), le superviseur peut être « *réalisé* » par un automate à état fini lorsque L(S/G) est régulier.

Ces principes du contrôle par supervision et de synthèse de superviseurs, initialement proposés dans le formalisme des automates à états finis, ont été étendus à d'autres formalismes afin de pouvoir appréhender des notions telles que le temps ou le parallélisme. C'est le cas par exemple du langage synchrone Signal (Marchand *et al.* 2000), de la logique temporelle Lin (1993), d'automates temporisés (Gouin & Ferrier 1999), des réseaux de Petri autonomes, Godon & Ferrier 1997, Ghaffari *et al.* 2002) ou T-temporisés (Chen & Hanisch 1999). Ce cadre du contrôle par supervision a été également étendu au contrôle des Systèmes Hybrides (Moor *et al.* 2002), avec des algorithmes spécifiques aux formalismes de modélisation des systèmes hybrides (Heymann *et al*. 1998).

Si ces techniques ont prouvé leur efficacité pour la synthèse de la commande sur des cas d'études académiques (Niel *et al.* 2001), leur application industrielle rencontre aujourd'hui encore certaines limites. Le principal problème rencontré concerne le phénomène d'explosion combinatoire dû aux algorithmes utilisés pour effectuer la synthèse et aux formalismes employés par les modèles. Pour cette raison, différents algorithmes, utilisant des modèles plus compacts ou plus « expressifs » ont été développés. Une autre approche naturelle pour limiter le

phénomène d'explosion combinatoire consiste à décomposer le modèle du procédé et/ou du superviseur de sorte à réduire le nombre d'états à traiter : synthèse hiérarchique, modulaire ou décentralisée (Wong & Wonham 1998, Chafik 2000, Yoo & Lafortune 2002).

Une autre difficulté, en vue d'un passage à l'échelle industrielle, réside dans l'interprétation du rôle d'un superviseur dans le cadre SCT et de celui qu'un contrôleur doit jouer dans des systèmes de commande réactifs (Zaytoon & Carre-Menetrier 2001). En effet, dans le cadre SCT, le procédé est un générateur spontané d'événements et le superviseur agit sur celui-ci en autorisant ou interdisant certains événements contrôlables. Il existe donc une indétermination résiduelle entre deux événements contrôlables et autorisés pouvant conduire à deux évolutions distinctes du procédé (Shayman & Kumar 1995). Par opposition, le contrôleur d'un système de commande réactif force, de manière déterministe, les événements à se produire selon un ensemble de règles prédéterminées. Pour lever cette ambiguïté, des règles de priorité ont été proposées pour lever l'indétermination résiduelle du superviseur (Fabian & Hellgren 1998) et proposer une interprétation des événements contrôlables et incontrôlables en termes d'entrées et de sorties (Balemi *et al.* 1993, Nourelfath & Niel 2004).

Enfin, la dernière difficulté réside dans l'élaboration des modèles de procédé et de spécification à partir desquels les techniques de synthèse permettent la génération d'un modèle de superviseur, en particulier lorsque les systèmes à modéliser ont une taille conséquente (Hiraishi 2001). En effet, ces modèles constituent souvent des représentations détaillées, faisant intervenir des choix de modélisation subjectifs, dans des formalismes peu structurants et peu hiérarchisés, et n'autorisant pas de processus de construction incrémentale (Brandin *et al.* 2004).

En ce sens, des voies très prometteuses ont été proposées par (Roussel & Faure 2006) au travers d'une méthode de synthèse de la commande basée sur le raffinement progressif de spécifications algébriques, mieux adaptée à la description des propriétés comportementales attendues d'un système automatisé et insensible au phénomène d'explosion combinatoire, ou encore par (Zaytoon & Carre-Ménétrier 2001) à partir d'une spécification Grafcet que les algorithmes de synthèse permettent d'amender en tenant compte de contraintes de sûreté et de vivacité.

Clairement positionnées au niveau du processus de conception d'un système automatisé et limitées à l'étude de ses aspects dymamiques (prédicat d'automatisation P1 : *Unknown Control Rules* $\wedge$ *Dynamics* $\supset$ *Behavioural Goal*), la synthèse de la commande reste un outil efficace pour la conception de la commande d'un système mais doit être intégrée dans une approche plus globale de modélisation dans le cadre d'un processus d'automatisation (Sanchez & Macchieto 1995).

### *3.2 Méthodes et Modèles pour la validation*

Le processus de validation répond à la question « *construisons-nous le bon modèle ?* » et repose essentiellement sur l'analyse de deux propriétés des modèles élaborés vis-à-vis des besoins exprimés par les utilisateurs: la **complétude** et la **pertinence**.

### 3.2.1 Complétude des modèles

Dans le cadre d'un processus d'automatisation partant de l'expression des exigences jusqu'à l'implantation d'une solution, les modèles de S.E.D. ne peuvent à eux seuls assurer la modélisation de l'ensemble des points de vue à prendre en compte. Si l'on cherche à assurer la **complétude** des modèles élaborés, la représentation de systèmes automatisés de complexité industrielle requiert en effet l'utilisation de formalismes plus ou moins abstraits, plus ou moins formels, sous une forme fonctionnelle, ensembliste, etc, permettant d'appréhender, de manière pragmatique, intuitive, voire qualitative, le fonctionnement global d'un système à concevoir. Les modèles issus des domaines du Génie Informatique, tels que UML [9] ou ses extensions RT-UML (Selic 1998) ou UMsdL, et de l'Ingénierie Système, tels que SysML[10] ou les modèles proposés par Sagace (Penalva 1997), répondent en partie à ces préoccupations en proposant un ensemble de formalismes (diagrammes statiques, dynamiques, fonctionnels, structurels, informationnels, etc), à différentes phases du cycle de développement et intégrés au sein d'un cadre de modélisation.

En particulier, dans le cadre de la modélisation des exigences pour laquelle la propriété de complétude s'avère cruciale, ces approches proposent des formalismes permettant la définition et la structuration des exigences sous la forme de scénarios de fonctionnement (Cas d'utilisation de UML) ou de diagrammes à objets (Diagramme des exigences de SysML). Ces derniers, à l'instar des modèles proposés par la méthode KAOS (Heaven & Finkelstein 2004) et son outil support Objectiver, définissent une exigence comme un objet possédant des attributs propres (source de l'exigence, type, niveau de priorité, méthode de vérification etc.) et étant en relation avec d'autres exigences (relation de raffinement, d'implication) ou d'autres objets supports (acteurs, fonction, composants, etc.) (Figure 15).

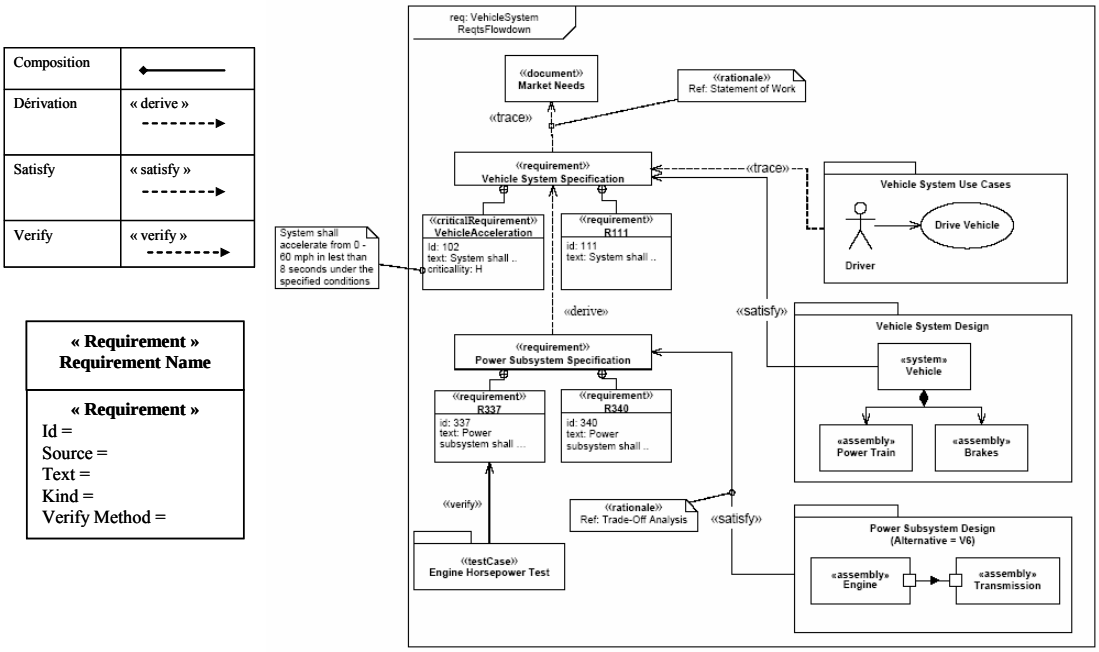



Figure 15. Diagramme des exigences (SYSML, 2006)

[9] UML : Unified Modelling Language, http://www.uml.org/

[10] SysML est un profil de UML2 avec des extensions adaptant UML aux besoins en ingénierie système. (source : http://www.omgsysml.org/)

Si ces modèles offrent des formalismes efficaces pour définir et structurer les exigences, ils n'intègrent que très peu de mécanismes ou d'outils permettant de s'assurer de la complétude des modèles ainsi obtenus. Ce constat n'est pas surprenant si l'on considère les aspects cognitifs et subjectifs de l'activité de modélisation qui mettent en jeu des processus intentionnels permettant à chaque modélisateur de transformer une intention de résultat en une attente de réalisation.
En ce sens, l'expression et la formalisation des connaissances et propriétés spécifiques à un domaine de modélisation (méta-modélisation UML, graphes conceptuels, etc.) devraient permettre de guider l'activité du modélisateur en lui proposant des règles sémantiques de construction des modèles qui l'amène à s'interroger sur leur cohérence. (Easterbrook, 2002) formule cette démarche comme la recherche d'un modèle qui respecte implicitement les propriétés du domaine et l'ensemble des besoins exprimés, selon le prédicat :

*Propriétés du Domaine* $\wedge$ *Spécifications* $\supset$ *Besoins exprimés* (Easterbrook, 2002)

A titre d'exemple, nous pouvons citer l'approche sémiotique de (Sfalcin 1992) qui propose d'aborder la modélisation fonctionnelle en associant à chaque flux des modalités Devoir-Faire, Vouloir-Faire, Savoir-Faire et Pouvoir-Faire qui caractérisent l'ensemble des flux nécessaires à la réalisation de toute activité, l'approche de (Paynter 1961) formalisée par (Féliot 1997) qui aborde la modélisation des processus de transformations physiques au travers d'une typologie de variables physiques qui caractérisent les échanges de matière et/ou d'énergie, ou enfin, les approches de (Vogel 1988) et (Wright & Bourne 1988) qui proposent de décrire les modèles de commande sous la forme de séquences d'actions logiques réutilisables (appelées actinomies). A noter que le mécanisme d'extensibilité, appelé « UML profile », défini comme une *« spécification qui spécialise un ou plusieurs méta-modèles standard d'UML »* facilite la modélisation et la réutilisation de telles connaissances « métiers » à l'intérieur d'un projet UML [R5].

### 3.2.2 Pertinence

Les techniques usuelles permettant de s'assurer de la pertinence et de la qualité d'un modèle ou d'un système réel vis-à-vis des attentes exprimées par les utilisateurs sont la simulation, l'émulation et le test. Selon la nature des interactions entre objets simulés et objets réels, (Isermann *et al.* 1999) distingue trois approches complémentaires (Figure 16): la simulation d'un modèle de commande couplé à des éléments réels non modélisés, la simulation en boucle fermée des modèles de commande et des processus physiques, et enfin l'émulation des systèmes physiques afin de valider, en plate-forme, le système de commande réel (Corbier 1989).

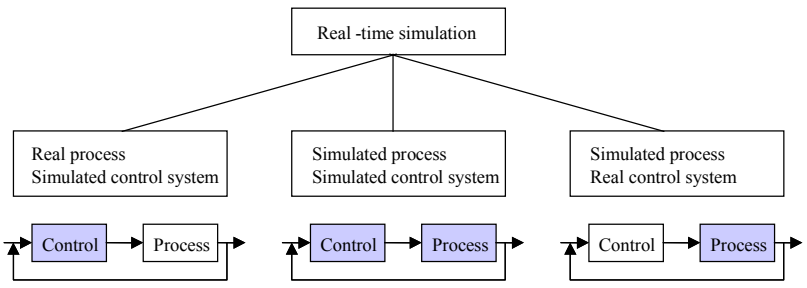



Figure 16. Techniques de simulation (Isermann *et al.*1999)

La simulation consiste à exécuter symboliquement un modèle selon un ou plusieurs scénarios ; elle repose donc sur une sémantique opérationnelle permettant d'évaluer, de manière déterministe, le comportement décrit par un modèle en réaction à des stimuli d'entrée. Sans être exhaustif, nous pouvons citer les outils basés sur :

- des modèles dynamiques des S.E.D., dont la (ou les) sémantique(s) opérationnelle(s) a(ont) été formalisée(s) : nous pouvons notamment citer les outils Statemate, ControlBuild, CPNtool, Scade, Esterel Studio modèles basés respectivement sur les modèles Statecharts (Harel 1987), Grafcet, Réseaux de Petri et les langages synchrones comme Signal, Lustre ou Esterel (Benveniste & Berry 1991),
- des modèles dynamiques de systèmes continus fondés sur des représentations algébro-différentielles (outil Matlab/simulink) ou sur des formalismes de type Bond Graph ou à objets (tel que le langage Modelica et son outil Dymola (Broenink 1999),
- des modèles statiques dotés d'une sémantique opérationnelle permettant de les exécuter (Zaytoon 1993, Tahir *et al.* 2003),
- des langages de programmation, tels que ceux proposés par la norme IEC611316-3 et la norme IEC 61499 dans le cadre des travaux de la communauté *holobloc* [11] et de projets comme OOONEIDA ou TORERO (Auinger *et al.* 2005, Ferrarini et al. 2005) avec des outils tels que FBDK (Univ. Calgary) ou CORFU (Tranoris & Thramboulidis 2006).

La simulation a pour inconvénient de ne pouvoir traiter, sous peine d'explosion combinatoire, l'ensemble des scénarios envisageables de manière exhaustive (fonctionnels vis-à-vis des vecteurs d'entrées admissibles ou structurels vis-à-vis des chemins d'exécution du modèle). En d'autres termes, elle permet de détecter certaines erreurs du modèle entraînant un comportement non conforme aux attentes des utilisateurs mais ne permet pas de garantir leur absence.

Dans ce contexte, la confiance que l'on peut accorder aux résultats de simulation repose sur l'expérience et l'expertise des utilisateurs leur permettant d'apprécier quantitativement et qualitativement les différents scénarios de simulation vis-à-vis, d'une part, des besoins exprimés (choix des scénarios, taux de couverture, interprétation des résultats, définition d'intervalles de confiance, etc.), et d'autre part, des sémantiques opérationnelles utilisées pour exécuter symboliquement les modèles. En effet, pour un modèle donné, il existe souvent plusieurs sémantiques opérationnelles admissibles (Harel et al. 1990, Lhoste et al. 1997) et/ou implémentées dans les outils de simulation (algorithmes d'interprétation, mécanismes d'évolution synchrones ou asynchrones, etc.) dont les différences peuvent engendrer des écarts non négligeables sur les résultats obtenus.

Si les techniques de simulation ont fait preuve de leur efficacité, notamment pour la validation d'applications complexes ou de grande taille en milieu industriel, elles ne permettent d'apporter qu'une présomption de conformité qui peut se révéler insuffisante pour le développement de systèmes soumis à de fortes contraintes de sécurité ou de sûreté de fonctionnement et pour atteindre le niveau SIL4 (*Safety Integrity Level*) de la norme IEC 61508 et de ses applications sectorielles (IEC 62061, IEC 61511, etc) relatives à la sûreté de fonctionnement des équipements E/E/EP (électrique/électronique/électronique programmable) ou encore les niveaux 4 et 5 de l'échelle Capability Maturity Model (Paulk 1995) caractérisant les processus de développement dans le domaine du logiciel (Tableau 3).

[11] www.holobloc.com

| | |
|---|---|
| Niveau 5 : Optimisé | Amélioration continue du processus par retour d'expérience quantitatif |
| Niveau 4 : Observable | Le processus et la qualité du logiciel produit sont observables |
| Niveau 3 : Défini | Le processus est documenté et peut être partiellement automatisé |
| Niveau 2 : Reproductible | Le processus est explicite et peut être reconduit d'une application à l'autre |
| Niveau 1 : Initial | Le processus n'est pas explicite |

Tableau 3. Capability Maturity Model (Paulk 1995)

## *3.3 Méthodes et Modèles pour la vérification*

Le processus de vérification répond à la question « *construisons-nous correctement le modèle ?* » et repose essentiellement sur l'analyse de deux propriétés relatives aux modèles élaborés: la **correction** syntaxique et sémantique et la **cohérence** intrinsèque et extrinsèque aux modèles (en particulier de la cohérence entre les modèles du système automatisé et le modèle des exigences).

### 3.3.1 Correction des modèles

En ce qui concerne la **correction** des modèles, l'objectif est de s'assurer du respect des règles syntaxiques et sémantiques de construction de ces derniers.

Dans le cas de modèles formels, en particulier ceux relevant de la modélisation des Systèmes à Evénements Discrets (Cassandras & Lafortune 1999), la vérification de leur correction est fondée sur :

- la définition mathématique des éléments de modélisation et de leur assemblage
- l'analyse de propriétés intrinsèques aux modèles telles que la vivacité, l'absence de blocage ou encore le déterminisme.

Dans le cas de modèles non formels ou semi-formels, en particulier dans le cas des modèles fonctionnels tels que SADT ou des modèles objets tels que UML, la vérification du respect des règles de construction peut prendre diverses formes :

- vérification empirique par un processus de relecture des modèles,
- vérification formelle par comparaison à un méta-modèle défini comme une représentation formelle et structurée des éléments syntaxiques et sémantiques d'un formalisme ou langage de modélisation (Pietrac 1999, Panetto 2006)(Figure 17).

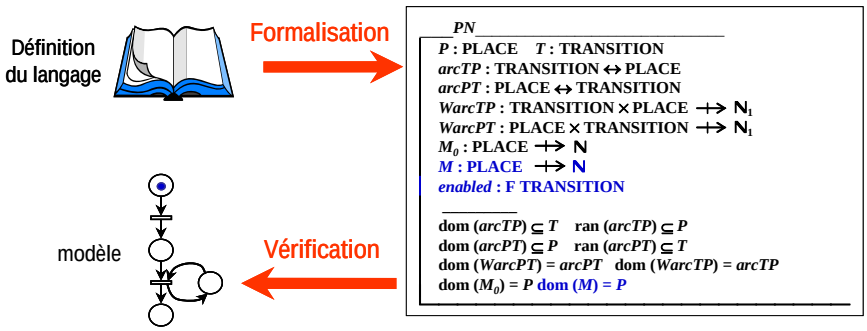



Figure 17. Méta-modélisation formelle des RDP avec le langage Z (Piétrac 1999)

### 3.3.2 Cohérence des modèles

En ce qui concerne la vérification de la **cohérence** des modèles, il s'agit de s'assurer que les informations contenues dans les différents modèles élaborés au cours du processus d'automatisation ne sont pas contradictoires, et, en particulier, que les modèles du système à développer respectent les propriétés spécifiées dans le modèle des exigences. Selon le degré de formalisation des différents modèles, deux approches peuvent être envisagées :

- dans le cas de modèles formels, la vérification de la cohérence des modèles du **système à faire** (modèles des exigences, modèles de sous-système, modèle d'architecture, etc) repose sur l'utilisation de techniques de preuves dont les deux principales sont le *model checking* et le *theorem proving*,
- dans le cas de modèles non formels ou semi-formels, la cohérence ne pouvant être démontrée mathématiquement sur les modèles du système à faire, l'attention se portera sur les processus d'élaboration de ces modèles (**système pour faire**) pour garantir la traçabilité des exigences en s'efforçant de montrer comment celles-ci sont modélisées (formalisées dans certains cas), propagées sur les sous-systèmes et composants, validées et/ou vérifiées.

#### 3.3.2.1 Approches formelles pour la vérification de la cohérence

Le ***model checking*** (Clarke *et al*. 2000) est destiné à la vérification de modèles comportementaux décrits sous la forme de modèles à états pouvant être, dans certains cas, temporisés. Le principe consiste à explorer, de manière exhaustive, l'ensemble des états atteignables par le modèle afin de vérifier qu'une propriété, considérée ici comme l'énoncé formalisé d'une exigence, est bien satisfaite. Les propriétés peuvent être relatives aux modèles de commande, à du code implantable, sous réserve de l'avoir retranscrit dans un formalisme utilisable par le model checker, ou encore à un modèle incluant la commande et les processus commandés.

Suivant les techniques utilisées, une propriété peut être exprimée sous une forme logique, éventuellement temporelle linéaire (Alur & Henzinger 1991) comme LTL (Linear Temporal Logic) ou CTL (Computational Tree Logic), ou spécifique au formalisme de modélisation comme (sans être exhaustif) dans les langages synchrones (Benveniste & Berry, 1991) tels que Signal, Lustre ou Esterel ou encore dans les langages SDL, LOTOS.

Plusieurs outils de model checking sont disponibles, parmi lesquels nous pouvons citer, de manière non exhaustive, PRISM, développé à l'Université de Birmingham, SMV développé à l'Université de Carnegie Mellon puis à Berkeley ou encore UPPAAL développé en collaboration entre le département d'Information Technology de l'université d'Uppsala en Suède et le département de Computer Science de l'université D'Aalborg au Danemark (Pettersson & Larsen 2000) (Figure 18). Ces outils présentent l'avantage de proposer un processus de preuves de propriétés entièrement automatique et de générer, en cas de violation de la propriété, un contre-exemple sous la forme d'une séquence de transitions d'états menant à la violation de la propriété.

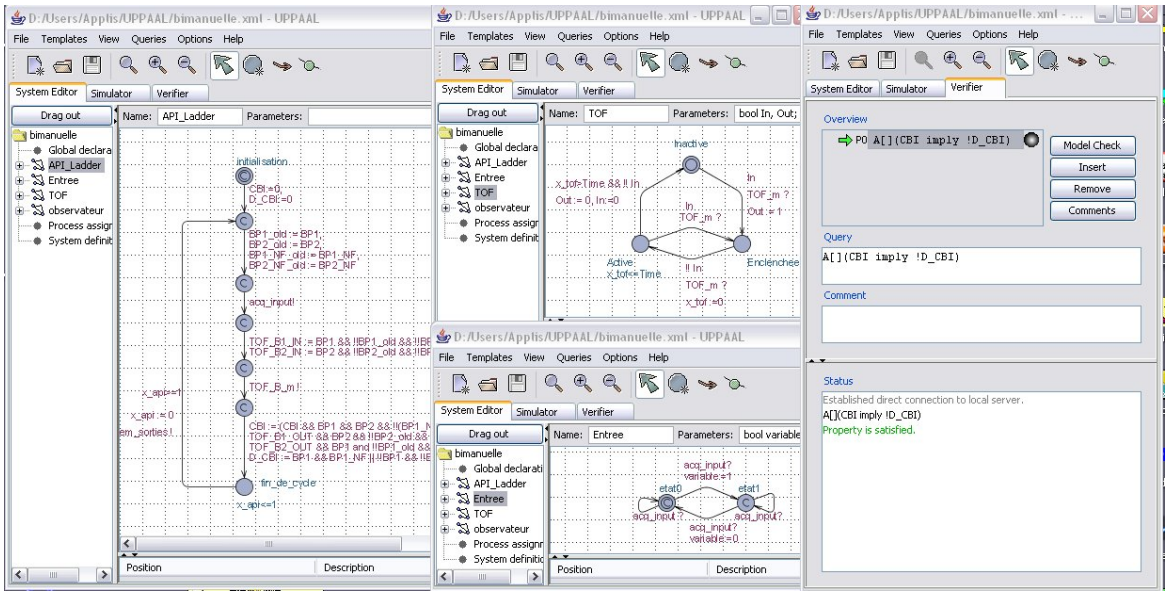



Figure 18. Exemple de vérification de propriété à l'aide d'UPPAAL [Rp11]

Les principales limites de ces approches sont dues, comme pour la synthèse de la commande :

- au phénomène d'explosion combinatoire engendré par l'exploration exhaustive de l'espace d'états ; des techniques basées sur les BDD, des règles d'abstraction ou un raisonnement compositionnel ont été proposées en ce sens,
- à l'extraction et la formalisation des propriétés à partir du modèle des exigences ; en cas d'échec de preuve, et malgré le contre-exemple généré, il est parfois difficile d'en identifier la cause : erreur dans le modèle de système ou dans la formalisation des propriétés ?

A cela, il convient d'ajouter le fait que ces approches sont limitées à l'analyse des propriétés comportementales des modèles (Prédicat P1 : *Unknown Control Rules* $\wedge$ *Dynamics* $\supset$ *Behavioural Goal*) et ne peuvent pas contribuer à la vérification des modèles fonctionnels ou structurels élaborés au cours du processus d'automatisation.

Le ***theorem proving*** est une méthode de vérification interactive pour laquelle la preuve de propriété repose sur des méthodes de déduction logique et sur la formulation progressive d'axiomes ou de théorèmes à partir d'une théorie de base. En d'autres termes, il s'agit de démontrer qu'une propriété, décrite sous la forme d'un théorème logique, peut être directement déduite du modèle à vérifier, des axiomes et de la théorie de base, en utilisant les règles de déduction de la logique utilisée. Les méthodes de déduction automatique sont implémentées au sein *d'assistants de preuve (theorem prover)* comme PVS, Coq, HOL ou Isabelle qui guide l'utilisateur dans la construction d'une preuve. Par ailleurs, ces techniques de preuves ont donné le jour à plusieurs méthodes de développement, comme les méthodes Z (Spivey 1989) et B (Abrial 1996) pour lesquelles des assistants de preuves ont été développés.

Ces approches reposent, pour la plupart, sur la logique du premier ordre et des formalismes ensemblistes plus abstraits que ceux utilisés par les techniques de model checking – souvent limités à des modèles comportementaux à états – qui

offrent une couverture de modélisation plus large incluant les aspects informationnels, fonctionnels ou structurels. En ce sens, elles offrent un cadre de modélisation plus naturel pour supporter un processus d'automatisation formulé selon le prédicat P2 (*Système de commande* $\wedge$ *Système opérant* $\supset$ *Exigences Système)*. De plus, une méthode telle que la méthode B, offre un mécanisme formel de raffinement des modèles qui permet de prouver la cohérence entre deux modèles correspondant à deux niveaux d'abstraction distincts, notamment en assurant la conservation des propriétés préalablement prouvées.

Si ces approches font état d'un certain nombre de succès industriels (Lano *et al.* 2000), notamment dans le domaine du ferroviaire[12] (Taouil-Traverson 1997), leurs principales limites dans le cadre d'un processus d'automatisation, concernent :

- la construction de la preuve : contrairement au *model checking*, l'obtention de la preuve résulte d'un processus interactif dans lequel l'utilisateur infère sur l'assistant de preuves par le choix de stratégies et la proposition d'axiomes,
- l'interprétation d'un échec de preuve : de manière identique au *model checking*, un échec de preuve peut provenir d'une erreur dans le modèle à vérifier et/ou dans la description de la propriété mais également, dans le cas du *theorem proving*, d'un problème d'indécidabilité (dans ce cas, un échec de preuve ne signifie pas que la propriété n'est pas vérifiée par le modèle),
- la complexité des théories sous-jacentes qui rend les assistants de preuves relativement difficiles à appréhender,
- enfin, la prise en compte du temps reste relativement délicate (Abrial & Mussat 1998) dans les formalismes abstraits de ces approches, ce qui limite souvent leur utilisation en phase de spécification.

#### 3.3.2.2 Approches non formelles pour la vérification de la cohérence

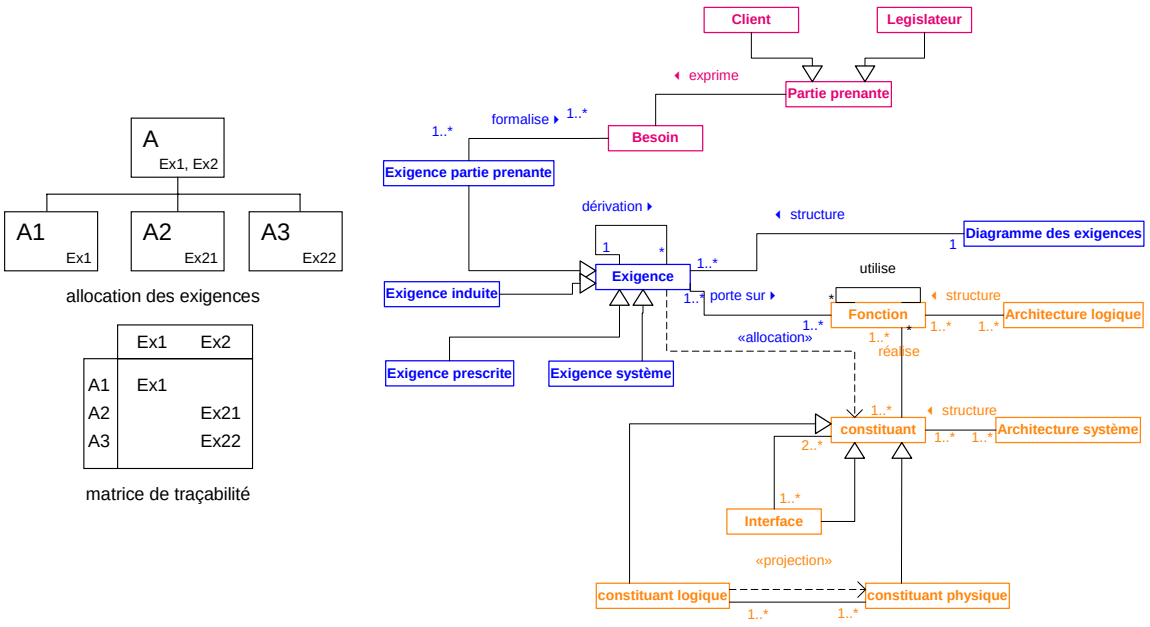
Si des modèles non formels ou semi-formels sont produits au cours du processus d'automatisation, l'absence de fondements mathématiques des formalismes utilisés rend délicate la vérification de la cohérence des informations contenues dans ces modèles. En particulier, comment garantir la cohérence entre une spécification non formelle des exigences et les différents modèles élaborés (multi points de vue) pour développer un système répondant à ces attentes ?

Ce point apparaît pourtant comme essentiel si l'on considère que la pratique d'une automatisation à échelle industrielle requiert l'utilisation de multiples formalismes permettant d'appréhender le fonctionnement global d'un système à concevoir. Ce problème rejoint celui abordé par l'**Ingénierie Dirigée par les Modèles** (Favre *et al.* 2006) ayant pour objectif d'offrir un cadre unificateur de modélisation pour les systèmes à logiciel prépondérant. Si cette approche est basée sur l'utilisation de modèles exprimés dans des **formalismes différents** pour couvrir les différentes étapes de développement, les différents niveaux d'abstraction ou encore les différents points de vue considérés, son originalité provient surtout de l'utilisation systématique de **méta-modèles** décrivant les formalismes utilisés et des **transformations** automatiques ou interactives entre modèles permettant le passage d'un domaine technique à un autre.

[12] http://www.clearsy.com/html/nos_projets.htm

Dans le domaine de l'automatisation, plusieurs approches ont été proposées en ce sens. Elles consistent principalement à :

- intégrer dans un formalisme de modélisation des **relations explicites vers d'autres modèles** ; nous pouvons citer, à ce titre, le diagramme des exigences proposé par SysML qui permet de faire référence à des objets (composants, fonctions, acteurs,etc) décrits dans d'autres diagrammes SysML (Figure 15) ou encore les approches mixant des diagrammes d'objets UML et une description comportementale sous forme de modèles statecharts ou de blocs fonctionnels de la norme IEC 61499 (Tranoris & Thramboulidis 2006, Auinger *et al.* 2005, Ferrarini *et al.* 2005, Zhang et al. 2005),
- définir des règles de **transformations de modèles** fondées sur leur méta-modèle, afin d'autoriser l'utilisation, dans un formalisme donné, d'un ensemble d'informations contenues dans d'autres modèles de l'étude, (Bon-Bierrel 1998, Pietrac 1999, Berruet 2006) [Rp3][Rp4],
- enfin, formaliser le **processus de modélisation** lui-même sous la forme d'un **référentiel commun**, tel que le modèle de données de l'AFIS ou les matrices de traçabilité des exigences (Figure 19), permettant d'assurer la cohérence entre les différents concepts et objets manipulés au cours du processus d'automatisation.




a) Allocation des exigences b) Modèle de données

Figure 19. Matrice de traçabilité et modèle de données (Ménadier 2002)

Ces approches contribuent à la cohérence des modèles élaborés au cours du processus d'automatisation en définissant des cadres de modélisation (Sowa & Zachman 1992) offrant une vision structurée et plus ou moins unifiée des différents concepts et objets de modélisation. En revanche, elles ne permettent pas d'obtenir de certitudes (sous forme de preuves ou de démonstrations) quant à la cohérence des informations contenues dans les différents modèles. En d'autres termes, elles permettent d'augmenter le niveau de qualité du processus de modélisation et présument donc de la qualité des résultats produits sans toutefois pouvoir la garantir.

# 4. CONCLUSION

Les recherches présentées dans ce mémoire ont pour objectif essentiel d'assurer la meilleure adéquation possible entre les besoins exprimés par les utilisateurs, leur spécification et les solutions développées :

- dans des contextes **industriels** (projets européens en actionnement et mesure intelligents, thèses en convention Cifre de D. Evrot avec l'INRS [TH3] et de H. El Haouzi avec la société Trane [TH4]) ou plus **académiques** (thèses de P. Lamboley [TH1] et de D. Gouyon [TH2]),
- pour garantir le respect de propriétés relatives à la **sécurité** de machines dangereuses (thèse de D. Evrot), à **l'interopérabilité** d'applications de commande distribuée (projets européens en actionnement et mesure intelligents) ou encore à la **reconfigurabilité** des systèmes pilotés par le produit (thèse de D. Gouyon).
- en développant des approches centrées sur la **synthèse de la commande de S.E.D.** (thèse D. Gouyon), des approches plus orientées « **système** » basées sur une modélisation **semi-formelle** des exigences (thèses de D. Evrot, H. El Haouzi) ou sur un **raffinement formel** des modèles avec la **méthode B** (thèse de P. Lamboley).

Enfin, le projet de recherche porte sur le développement d'un processus sûr d'automatisation, tirant profit des approches semi-formelles pour l'analyse qualitative du fonctionnement global d'un système automatisé, et formelles pour une évaluation quantitative de leurs propriétés.

# II Synthèse de la commande : application à la reconfiguration dynamique de la commande

## 1. INTRODUCTION

Relativement au prédicat d'automatisation de Fusaoka (1983)(*Unknown Control Rules* $\wedge$ *Dynamics* $\supset$ *Behavioural Goal (P1)* et dans le cadre théorique des Systèmes à Evénements Discrets (Cassandras & Lafortune 1999)(Ramadge & Wonham, 1987), la synthèse de la commande permet la génération automatique de superviseurs étant, par construction, sûrs, réactifs et sans blocage à partir de la formalisation des comportements attendus et de la dynamique du système physique à commander. Ces techniques ont prouvé leur efficacité pour la synthèse de la commande sur des cas d'études académiques (Niel *et al.* 2001) mais leur application industrielle rencontre aujourd'hui encore certaines limites telles que: (a) le **phénomène d'explosion combinatoire** dû aux algorithmes utilisés pour effectuer la synthèse et aux formalismes employés par les modèles, (b) **l'élaboration des modèles** de procédé et des spécifications dans des formalismes peu structurants en particulier pour des systèmes complexes de taille conséquente (Hiraishi 2001), et (c) l'**interprétation** de la nature permissive de superviseurs (autorisant ou interdisant certains événements) par comparaison aux contrôleurs des systèmes réactifs forçant, de manière déterministe, les événements à se produire selon un ensemble de règles prédéterminées (Zaytoon & Carre-Menetrier 2001).

La première partie de ce chapitre présente la proposition d'une **démarche itérative et modulaire de synthèse [R2][R11]** qui contribue à réduire le problème d'explosion combinatoire (a) et de modélisation (b) en s'appuyant sur une hiérarchisation des spécifications et sur la définition d'un mécanisme itératif de composition des modèles de procédé à partir de représentations élémentaires. L'implantation de l'architecture de commande résultante (c) repose alors sur un ensemble de règles d'allocation de priorités permettant d'interpréter la structure modulaire de superviseurs sous la forme de blocs fonctionnels de la norme IEC 61131-3.
La deuxième partie de ce chapitre utilise les résultats que nous avons obtenus en synthèse de la commande pour proposer une démarche de **reconfiguration** des **systèmes de pilotage par le produit [R5]**. L'objectif est de réduire considérablement les temps de conception d'une nouvelle configuration de commande exploitant au mieux les degrés de flexibilité du système de production pour réaliser un produit personnalisé présentant des caractéristiques spécifiques (Henry *et al.* 2004). Sur la base du concept de *virtual production line* introduit par (Qiu *et al.*, 2003), les algorithmes de synthèse nous permettent alors d'établir de manière automatique des règles de **routage du produit** à travers les différentes ressources de l'atelier ainsi que des règles **de commande** des différentes opérations de transformations ou de transport réalisées par ces **ressources [R6]**.
Ces recherches ont été initialisées en 2000 en saisissant l'opportunité offerte par le recrutement d'Alexia Gouin en tant qu'ATER dans le cadre du projet Méthodes & Modèles Formels pour l'automatisation des processus de production (Contrat quadriennal CRAN 2000-2003) puis développées dans la thèse de D. Gouyon [TH2].

# 2. SYNTHESE MODULAIRE ET ITERATIVE DE LA COMMANDE

## *2.1 Problème*

Le premier chapitre de ce mémoire a montré l'évolution actuelle des automatismes industriels vers des architectures de plus en plus distribuées, modulaires et basées sur la coopération d'objets logiciels (Auinger 2005) dont les bénéfices en termes de réutilisabilité, de flexibilité ou de maintenabilité ne sont plus à démontrer. Dans la plupart des approches d'ingénierie mises en œuvre pour déployer de telles architectures, la structuration de la commande est déduite d'une décomposition des objectifs « système » en fonctions, services ou encore en composants d'automatismes. Ces décompositions fonctionnelles ou objet aboutissent in fine à une architecture sous forme d'*agents* d'automatisation fournissant un ensemble donné de services en agissant sur les ressources qu'ils contrôlent (Vogrig et al 1987, Belhimeur 1989, Lhoste 1994, Zamai *et al.* 1998).

Dans le domaine de la synthèse de superviseurs, la modularité a été abordée comme un moyen de réduire le phénomène d'explosion combinatoire (Vahidi *et al.* 2006, Endsley *et al.* 2006) et a conduit à diverses extensions du cadre théorique (Wong & Wonham 1998) proposant des décompositions soit du modèle du procédé, soit du modèle de spécification, soit des deux parties (procédé et spécification). Les critères de décomposition retenus visent à réduire la complexité des modèles et de leurs espaces d'états et sont donc essentiellement liés à la recherche d'une structure optimale qui minimise les intersections entre alphabets. Cette modularité favorise par voie de conséquence la réutilisation des modèles de synthèse : il est possible de procéder à des modifications n'affectant qu'un sous-ensemble des modèles et de ne synthétiser que les seuls superviseurs concernés, ou d'utiliser des composants en bibliothèque (Chen *et al.* 2000).

Le premier problème posé consiste donc à intégrer, dans les approches de synthèse modulaire, des **critères de décomposition**, non seulement justifiés par des contraintes de réduction du phénomène d'explosion combinatoire mais surtout induits par l'**organisation fonctionnelle et structurelle du système de production** et prenant en compte les contraintes relatives à une démarche d'**ingénierie ascendante** favorisant la réutilisation de modèles ou composants de commande.

D'autre part, comme nous l'avons montré dans le premier chapitre, l'implantation de superviseurs synthétisés dans le cadre SCT pose le problème de leur interprétation. En effet, le superviseur SCT agit sur le procédé en autorisant ou interdisant certains événements contrôlables alors que le contrôleur d'un système réactif force les sorties à destination du système commandé (événements contrôlables) en fonction de son état et des informations qu'il délivre (événements incontrôlables). De plus, le superviseur SCT peut, par définition, présenter une indétermination résiduelle : deux événements contrôlables peuvent être simultanément autorisés et conduire à deux évolutions distinctes du procédé (Shayman & Kumar, 1995). Pour lever cette ambiguïté, des interprétations des événements contrôlables et incontrôlables en termes d'entrées et de sorties ont été proposées (Balemi *et al,* 1993, Nourelfath & Niel 2004) mais ne sont applicables que s'il existe au moins un événement contrôlable entre deux événements incontrôlables. Cette hypothèse se justifie dans le cas d'une hypothèse synchrone pour laquelle la réaction du système (une ou

plusieurs sorties correspondant à des événements contrôlables) à un stimulus extérieur (événement incontrôlable) est instantanée (et dans tous les cas doit survenir avant l'arrivée d'un nouveau stimulus d'entrée). Si cette hypothèse n'est pas vérifiée, il sera alors nécessaire de lever l'indétermination résiduelle du superviseur, ce qui peut se traduire par des priorités sur les transitions, voire par la suppression de certaines séquences (Fabian & Hellgren 1998).
Le second problème posé concerne **l'implantation** des superviseurs générés, c'est-à-dire la définition de **règles de priorité** permettant le passage d'un superviseur à un contrôleur déterministe puis l'implantation des superviseurs générés dans le cadre d'une synthèse modulaire sous la forme de **blocs fonctionnels**, notamment dans le cadre des standards en vigueur dans le domaine des automatismes industriels comme la norme IEC 61131-3 ou la norme IEC 61499.

En d'autres termes, les problèmes posés conduisent à proposer un cadre méthodologique, tels que ceux proposés par (Zaytoon & Carré-Ménétrier, 2001) ou (Sanchez & Macchieto 1995) permettant d'intégrer les algorithmes de synthèse dans un processus plus complet d'automatisation couvrant les phases « amont » de modélisation et « aval » d'implantation sur une architecture cible.

### *2.2 Contribution*

La démarche de synthèse que nous proposons est fondée sur l'hypothèse d'une **architecture hiérarchique et coordonnée de superviseurs** [R2][R11][C13][C15]. Elle se base, pour cela, sur une mise en œuvre **itérative** des algorithmes de synthèse qui consiste à réutiliser des superviseurs d'un niveau inférieur pour construire le modèle de procédé d'un superviseur de niveau supérieur, et ce, depuis les niveaux les plus technologiques.

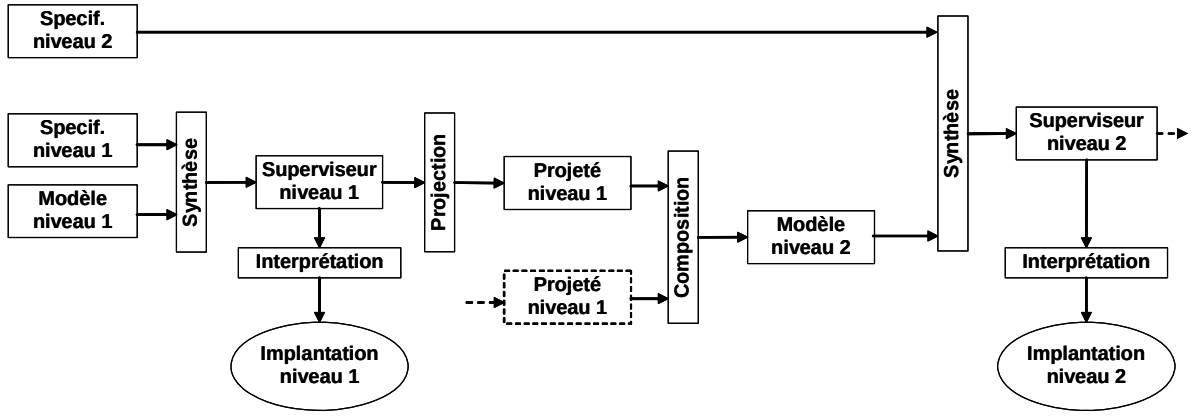



Figure 20. Proposition d'une démarche de synthèse modulaire et itérative

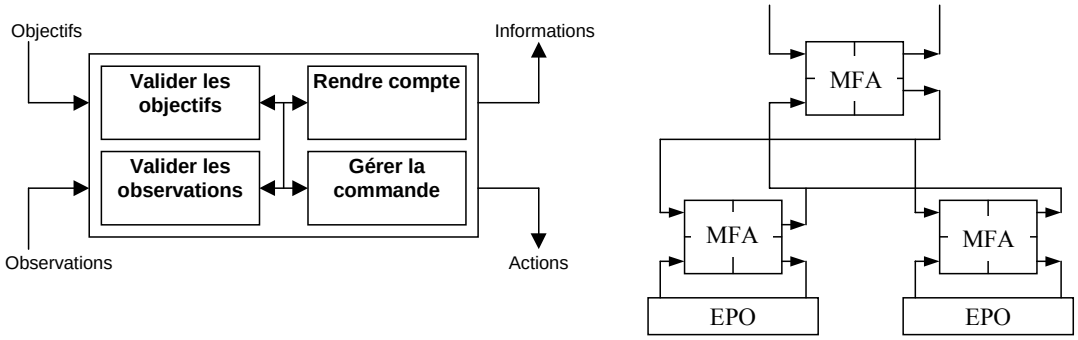
Notre approche combine ainsi les aspects structurants proposés par les méthodes d'automatisation actuelles (approches orientées objet ou par composants) et les techniques de synthèse modulaire (Figure 20):

- les critères utilisés pour structurer le système de commande sont essentiellement basés, d'une part, sur la structure physique du procédé lui-même en ce qui concerne les niveaux les plus technologiques relatifs aux actionneurs et équipements de terrain, et, d'autre part, aux regroupements successifs, dans une démarche « bottom-up », de ces équipements en unités fonctionnelles,

- les algorithmes de synthèse (Wonham & Ramadge 1987) seront utilisés pour générer automatiquement les superviseurs associés à chaque module (ou objet d'automatisation) de l'architecture de commande ; pour un module donné, le superviseur sera synthétisé à partir d'un modèle de spécification décrivant les missions allouées à ce module et à partir d'un modèle du procédé donné par les superviseurs de niveau inférieur.

### 2.2.1 Synthèse d'une architecture hiérarchique coordonnée de superviseurs

Les systèmes manufacturiers laissent apparaître en profondeur un caractère répétitif autour d'éléments technologiques, le plus souvent standard, auxquels il est possible d'associer un comportement logique indépendant du contexte de leur utilisation. Cette description comportementale, intrinsèque à un élément technologique mis au service d'un système de commande, modifie le schéma classique « partie opérative / partie commande » en y intégrant une interface ayant pour rôle de filtrer les commandes émises à destination des équipements technologiques en vérifiant leur compatibilité vis-à-vis de leur état courant et de filtrer les observations en les comparant au comportement normal modélisé (Vogrig *et al.* 1987, Lhoste, 1994, Zamai *et al.* 1998). Ces interfaces composent une véritable bibliothèque de modules réutilisables de contrôle et de commande des équipements technologiques, en particulier des actionneurs et capteurs.
La généralisation de cette approche à des niveaux moins technologiques en s'appuyant sur une démarche ascendante et itérative composant les comportements élémentaires pour construire des comportements de plus en plus complexes, permet d'aboutir à une structuration modulaire hiérarchisée des Systèmes Automatisés de Production sous la forme d'objets d'automatisation (Figure 21). Ces objets assurent la réalisation d'un service en réponse à une sollicitation externe (provenant d'objets d'automatisation de niveau hiérarchique supérieur ou équivalent) en agissant sur les ressources dont ils disposent (objets d'automatisation de niveaux inférieurs ou équivalents). Le système de commande résultant est alors constitué d'un ensemble de modules coordonnés et/ou coopérants.




a) Module Fonctionnel d'Automatisme (MFA) b) Organisation fonctionnelle hiérarchique

Figure 21. Commande hiérarchique coordonnée (Belhimeur, 1989)

S'inspirant de ce type d'architecture de commande, le premier niveau de l'architecture hiérarchique et coordonnée de superviseurs que nous proposons correspond à la commande des actionneurs. Le processus de synthèse utilisé est tout à fait conventionnel. Des modèles du procédé sont élaborés pour modéliser les comportements des équipements technologiques en l'absence de commande ; sont

donc considérés dans ces modèles, l'ensemble des comportements physiquement admissibles. Les états marqués du modèle représentent les états dans lesquels les caractéristiques physiques observables du procédé sont stabilisées alors que les états non marqués représentent les états transitoires du procédé. Les spécifications correspondant aux comportements attendus de ces actionneurs sont construites de manière parcellaire en considérant séparément les réactions souhaitées aux différents stimuli d'entrée puis composées par produit synchrone pour obtenir le modèle de spécification complet. Ces modèles permettent de synthétiser des superviseurs associés à chaque famille d'équipements technologiques. Nous utilisons pour cela, l'outil de synthèse de superviseurs TCT[13].

Il est à noter que la **contrôlabilité** des événements présents dans ces modèles est interprétée en termes d'entrées et de sorties : les événements incontrôlables sont associés aux stimuli reçus par la commande de l'actionneur (les demandes émanant de niveaux de commande hiérarchiquement supérieurs et/ou les mesures effectuées par les capteurs), les événements contrôlables sont associés aux sorties émises par la commande de l'actionneur (signaux électriques vers les pré-actionneurs et comptes-rendus élaborés à destination des niveaux de commande hiérarchiquement supérieurs).

La démarche appliquée pour obtenir les superviseurs de coordination des niveaux supérieurs est itérative. Les spécifications, propres à chaque superviseur, sont proposées par le modélisateur de manière classique. En revanche, le modèle de procédé d'un superviseur de coordination de niveau n+1 est obtenu automatiquement à partir des superviseurs de niveau n par (Figure 22) :

- projection des superviseurs de niveau n en ne conservant que les évènements en interactions avec le niveau supérieur n + 1,
- instanciation des alphabets des projetés, dans le cas où deux superviseurs identiques sont impliqués dans la commande de niveau n, afin de distinguer leurs alphabets respectifs,
- changement de la contrôlabilité des événements : en effet, les événements en entrée (resp. sortie) qui étaient vus par le niveau n comme incontrôlables (resp. contrôlables) sont vus par le niveau n+1 de façon complémentaire,
- modification par ajout de self-loop des alphabets de chaque projeté afin de disposer d'un alphabet commun pour l'ensemble des projetés,
- composition synchrone.

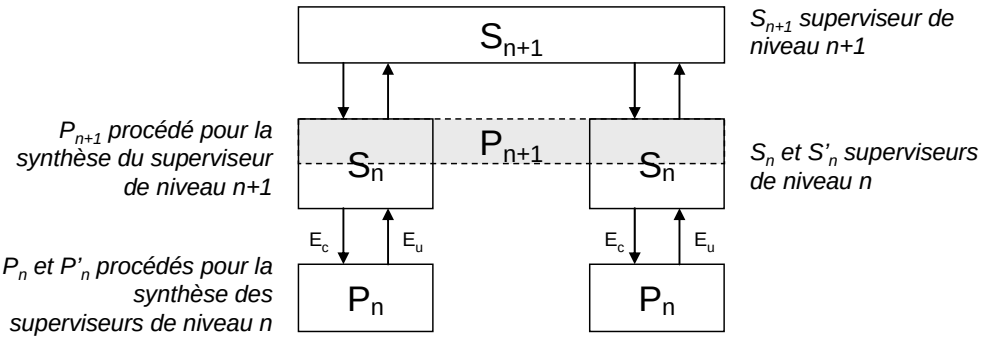
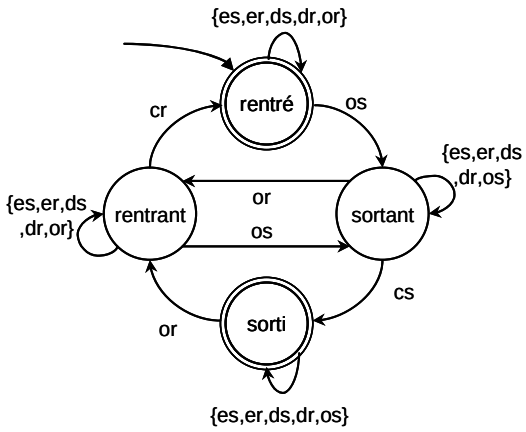



Figure 22. Construction du modèle de procédé pour un superviseur de niveau n+1

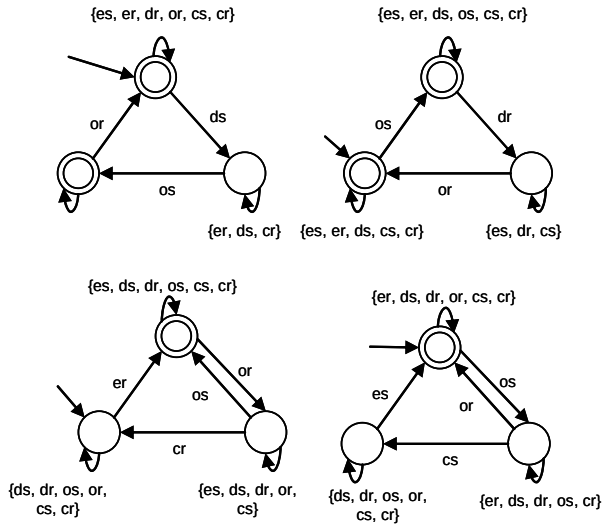
[13] TCT : développé à l'Université de Toronto et téléchargeable à l'adresse http://odin.control.toronto.edu/DES/

Il est à noter que les différents superviseurs d'un même niveau ont des alphabets disjoints (le seul élément pouvant être commun est l'élément neutre $\varepsilon$), évitant ainsi une partie des blocages possibles, à l'image de ce que Leduc (2002) propose en intercalant une interface d'échanges entre un superviseur de coordination et les éléments coordonnés. Cependant, cette propriété ne garantit pas dans l'absolu que certaines spécifications des superviseurs de coordination ne conduisent à des impasses. Ceci peut-être le cas si le superviseur de coordination présente un groupe d'états absorbants dans lequel les événements d'un des superviseurs de niveau inférieur ne seront jamais autorisés provoquant ainsi le blocage de ce dernier. Ce cas de figure n'a jamais été rencontré pour les spécifications que nous avons proposées relativement aux exemples étudiés mais un travail à venir pourrait consister à proposer des règles d'écriture des spécifications visant à respecter les conditions nécessaires relatives à la préfixe-clôture des langages mises en évidence pour la supervision distribuée par (Jiang & Kumar 2000) ou (Fabian & Kumar, 2000).

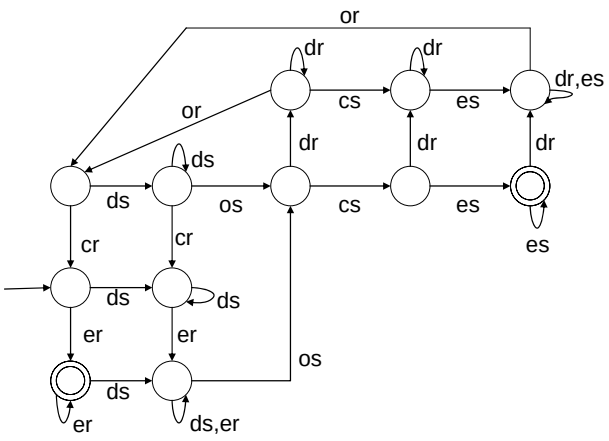
Pour illustrer la démarche, nous l'appliquons sur un manipulateur dont l'objectif est de déplacer un produit d'un poste de prise vers un poste de pose selon un cycle en U à l'aide de deux vérins double effet, munis d'un distributeur bistable 5/2, assurant les déplacements horizontaux et verticaux et d'une ventouse assurant la préhension.




a. Modèle de procédé




b. Modèles de spécification




c. Superviseur synthétisé

| Abréviation | Nom de l'événement |
|---|---|
| cr | capteur de fin de course, vérin rentré |
| cs | capteur de fin de course, vérin sorti |
| or | ordre de rentrée du vérin |
| os | ordre de sortie du vérin |
| er | état rentré du vérin |
| es | état sorti du vérin |
| dr | demande de rentrée du vérin |
| ds | demande de sortie du vérin |

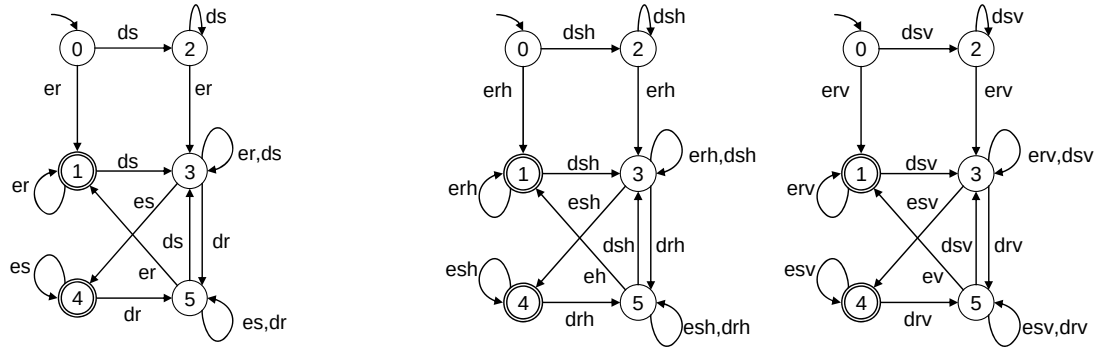
d. Tableau des événements

Figure 23. Synthèse d'un superviseur pour un vérin double effet 5/2 bistable

La Figure 23.a présente les modèles de procédé pour le vérin double effet muni d'un distributeur 5/2 bistable. La Figure 23.b présente les modèles de spécification associés au vérin sous la forme de deux automates décrivant les règles de génération des ordres de rentrée ou de sortie en fonction des demandes et de l'état courant du vérin, et de deux automates décrivant les règles de génération des comptes-rendus en fonction des observations réalisées par les capteurs et de l'état courant du vérin. Le modèle de spécification complet du vérin est obtenu par composition synchrone de ces quatre automates (21 états, 68 transitions). La Figure 23.d détaille la signification des événements impliqués dans ces différents modèles ainsi que leur contrôlabilité dans le cadre de la synthèse des superviseurs de niveau 1. Enfin, la Figure 23.c présente le superviseur associé au vérin et synthétisé, à l'aide de TCT, à partir du modèle de procédé (Figure 23.a) et de la composition des spécifications (Figure 23.b).
Le superviseur ainsi synthétisé peut être utilisé pour la commande des vérins assurant les déplacements horizontaux et verticaux (technologie identique). La même opération est réalisée pour synthétiser le superviseur associé à la ventouse.

Pour obtenir le superviseur de niveau 2 chargé de la coordination entre les deux vérins assurant les déplacements horizontaux et verticaux ainsi que la ventouse, il est tout d'abord nécessaire de procéder à la génération du modèle de procédé comme indiqué à la page précédente :

- projection des superviseurs de niveau 1, pour ne conserver que les événements relatifs aux demandes et comptes-rendus en provenance/à destination du superviseur de coordination (Figure 24.a),
- instanciation du projeté afin de différencier les alphabets des superviseurs associés aux deux vérins horizontal et vertical (indices v et h, Figure 24.b),
- modification de la contrôlabilité des événements ; les événements correspondant aux demandes de rentrée et de sortie (ds, dr) sont incontrôlables pour les superviseurs associés aux vérins mais deviennent contrôlables pour le superviseur de coordination et inversement pour les événements correspondant aux comptes-rendus (es, er),
- surcharge des alphabets des projetés (via des selfs) pour disposer d'un alphabet commun et **composition synchrone**.




a. projection du superviseur de la Figure 23.b

b. instanciation aux vérins horizontal (h) et vertical (v)

Figure 24. Projection des superviseurs associés aux vérins

Les spécifications du superviseur de coordination décrivent alors les règles régissant la synchronisation des mouvements horizontaux et verticaux ainsi que la mise en marche ou l'arrêt de la ventouse. Cette spécification peut s'écrire sous la forme de

l'automate présenté à la Figure 25. Il est à noter que deux événements en interaction avec un niveau de commande hiérarchiquement supérieur sont introduits : un événement incontrôlable pour ce niveau qui déclenche un cycle de prise et de pose (dcy) et un événement contrôlable qui signifie l'achèvement de ce cycle (fcy).

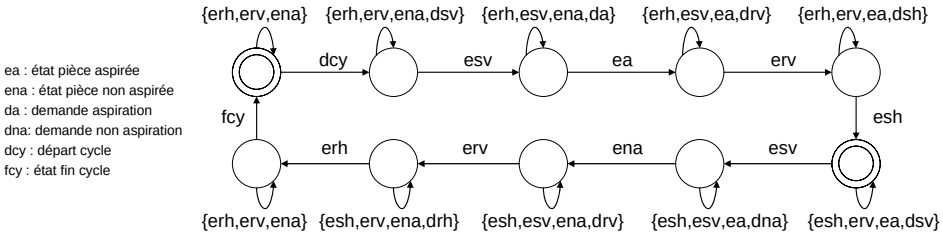



Figure 25. Modèle de spécification pour le superviseur de coordination de niveau 2

A partir de ce modèle de spécification et du modèle de procédé obtenu par composition des projetés des superviseurs de niveau 1, les algorithmes de synthèse génèrent un superviseur de 20 états et 63 transitions.
Cette démarche peut naturellement être poursuivie de manière itérative et ascendante pour synthétiser les superviseurs correspondant à des fonctions de coordination de plus en plus complexes.

### 2.2.2 Implantation des superviseurs

Les superviseurs de l'architecture hiérarchique coordonnée ne peuvent directement être implantés sur une architecture cible sous forme de blocs fonctionnels, préconisée par les normes IEC 61131-3 et 61499, même si l'organisation modulaire synthétisée est en parfaite adéquation avec la structuration de ce type de programmes. Comme indiqué au début de ce chapitre, les superviseurs doivent être préalablement interprétés afin d'aboutir à des contrôleurs déterministes pour lesquels la propriété suivante doit être vérifiée : *deux transitions t1 et t2 ne peuvent posséder le même état d'origine que si les deux événements associés aux transitions sont incontrôlables.*
Cette hypothèse forte est justifiée par les deux points suivants :

- deux transitions contrôlables sortant d'un même état signifieraient que deux actions sont possibles dans un état donné, alors qu'un contrôleur réactif ne doit en forcer qu'une seule,
- deux transitions, une contrôlable et une incontrôlable, peuvent générer une réaction non déterministe dépendant de la période d'échantillonnage durant laquelle les événements sont vus.

L'interprétation est basée sur un mécanisme d'allocation de priorités sur les transitions (Figure 26). Dans le premier cas, la priorité sera donnée à l'événement qui appartient à l'alphabet de plus haut niveau ; si les deux événements sont de même niveau, les priorités doivent se référer à des choix de conception de l'application. Quel que soit le choix, le contrôleur restera dans tous les cas dans un espace d'états autorisés. Dans le second cas, la priorité dépend de la contrôlabilité des événements, les événements incontrôlables étant prioritaires sur les événements contrôlables, préservant ainsi la propriété de réactivité à l'occurrence spontanée d'événements.

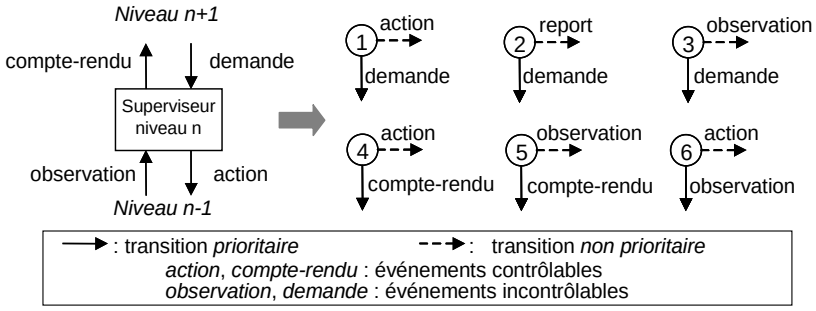



Figure 26. Allocation de priorités sur les transitions

Munis de priorités, les événements des superviseurs sont interprétés en termes de variables d'entrée et de sortie d'un bloc fonctionnel. Plus précisément, les événements incontrôlables sont interprétés comme des fronts montants des variables d'entrée alors que les événements incontrôlables sont interprétés comme des fronts montants des variables de sortie état provoquant le franchissement d'une transition vers un état dans lequel la sortie correspondante est activée et maintenue jusqu'à désactivation de cet état.
Ces règles peuvent être codées assez facilement (Figure 27):

- selon une formulation algébrique par activation désactivation synchrone : $Si_{t+1} = Ai_t \vee (Si_t \wedge \neg Di_t)$ où Ai représente les conditions d'activation d'un état ($Si_t$) et Di les conditions de désactivation
- en utilisant un algorithme d'interprétation par activation/désactivation synchrone,
- dans un des langages Structured Text ou Ladder de la norme IEC61131-3.

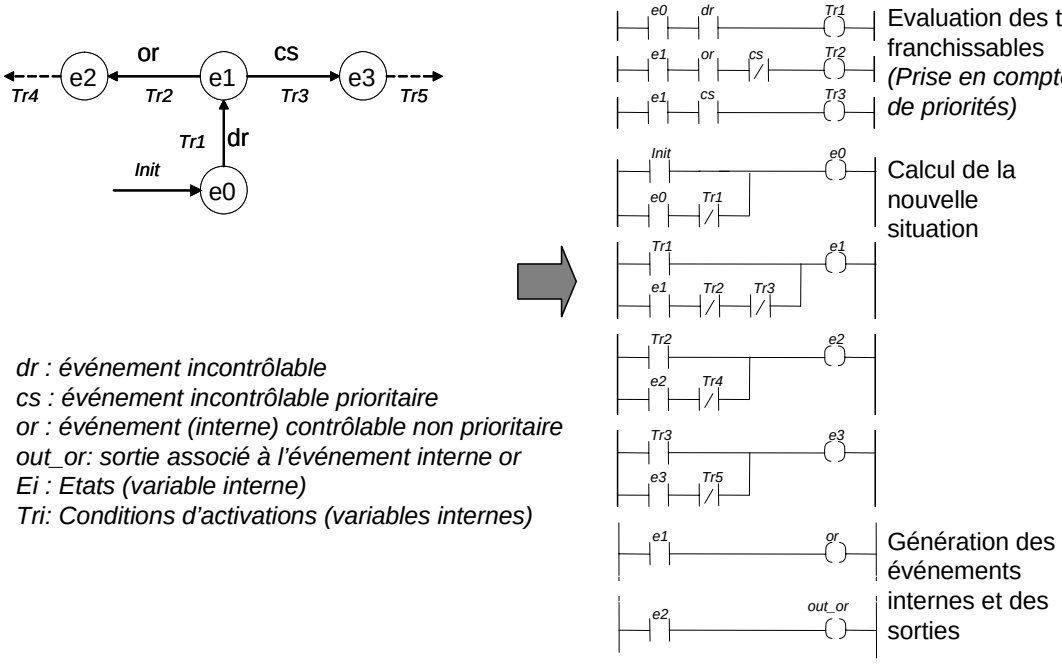



Figure 27. Génération de code Ladder

Cette méthode d'interprétation a été utilisée pour tester avec succès notre approche depuis la phase de modélisation jusqu'à l'implantation de code sur une architecture cible. L'exemple du manipulateur assurant un cycle de prise/pose développé au paragraphe précédent a ainsi été implanté avec succès sur une plate-forme de l'AIP-

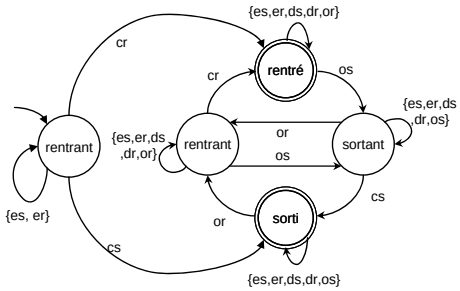
PRIMECA Lorrain équipée d'automates Siemens Série 7. Pour faciliter l'implantation des superviseurs que nous synthétisons, un outil a été développé TCT2LD pour extraire les superviseurs fournis par TCT, proposer un mécanisme automatique (lorsque ceci est possible) ou interactif d'allocation des priorités et enfin pour produire du code Ladder téléchargeable dans un automate programmable industriel.

### *2.3 Discussion sur les résultats obtenus*

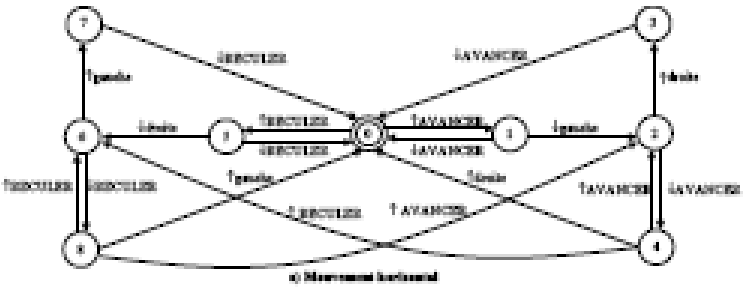
La synthèse modulaire et itérative proposée répond en partie à nos attentes dans la mesure où :

- l'élaboration des modèles de procédé est systématique à partir du niveau 2 puisque les modèles sont automatiquement obtenus à partir de la composition des projetés des superviseurs de niveau inférieur,
- l'écriture des spécifications est facilitée puisque la mission globale de la commande est décomposée en sous-missions, chacune d'entre elles étant associée à un superviseur donné,
- la démarche a été validée jusqu'à la génération de code sur plusieurs exemples mis en œuvre sur les plates-formes de l'AIP-PRIMECA Lorraine.

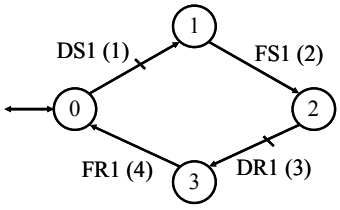
Néanmoins, il nous faut quand même admettre que le travail d'élaboration des modèles préalablement à l'utilisation des algorithmes de synthèse reste un problème délicat. En effet, dans le cadre de notre étude, il est apparu que plusieurs modèles, dont les différences pouvaient être plus ou moins importantes (marquage, structure et permissivité), étaient envisageables pour représenter les spécifications ou le procédé aux niveaux les plus technologiques.
Nous avons donc cherché à appliquer notre démarche avec différents modèles admissibles pour le problème considéré, afin d'essayer de mesurer l'influence des variations apportées sur les résultats obtenus par la synthèse.



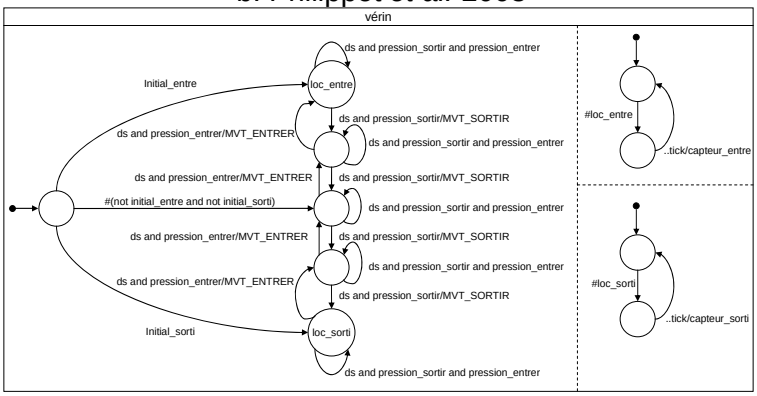

a. Gouyon *et al.* 2003



b. Philippot *et al.* 2003




c. Piétrac, 2002



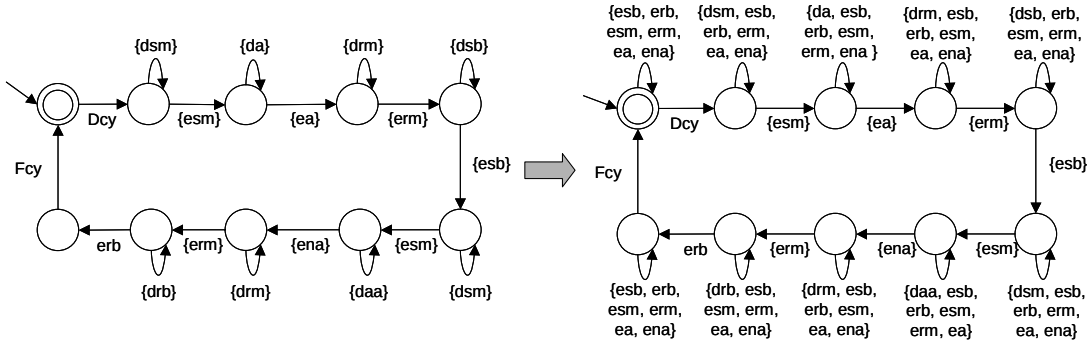

d. Gaffé, 2003

Figure 28. Différents modèles d'une même partie opérative

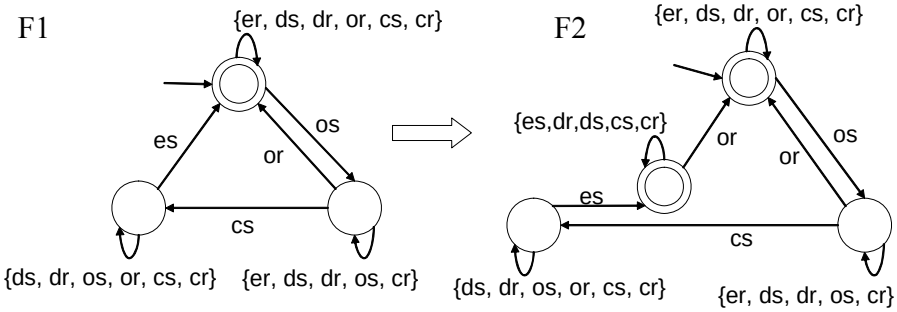
En ce qui concerne le procédé, il est apparu que plusieurs modèles sont admissibles en fonction du niveau de détail que l'on souhaite prendre en compte. Dans le cadre d'une session organisée par le groupe de travail COSED (à l'époque GdR Automatique), les membres du groupe ont été amenés à confronter leurs modèles de partie opérative sur un cas d'étude commun sans toutefois pouvoir tirer de conclusions pertinentes quant aux modèles les plus adaptés au processus de synthèse (Figure 28).

En ce qui concerne les modèles de spécification, plusieurs paramètres peuvent influencer leur élaboration :

- le **marquage des états**, défini dans (Ramadge & Wonham 1987) comme *« $L_m(G) \subset L(G)$ is a distinguished subset of these sequences that may be « marked », or recorded, perhaps representing completed « tasks » (or sequences of tasks) carried out by the physical process that G is intended to model »*. Cette définition peut être interprétée différemment par chaque utilisateur, en fonction de ses attentes en terme de comportement souhaité du procédé.
- la **permissivité des modèles**, notamment en tolérant plus ou moins d'événements en boucle sur les états (Figure 29.a).
- la **structure des modèles** : à partir d'un texte souvent langage naturel décrivant les contraintes de fonctionnement du procédé commandé, la formalisation sous forme algébrique est atteignable (Roussel & Faure 2006) alors qu'elle est beaucoup plus délicate sous la forme d'un automate à états finis, notamment lorsqu'il s'agit d'identifier les états significatifs. Par exemple, nous avons proposé deux spécifications admissibles pour les vérins selon que l'on introduise, ou pas, un état permettant de maintenir les comptes-rendus d'états « rentré » ou « sorti » du vérin (Figure 29.b).




a. modification de la permissivité des modèles




b. modification de la structure des modèles

Figure 29. Variabilité des modèles de spécifications

Au regard des différents tests que nous avons pu réaliser [R4], il apparaît que l'influence des choix de marquage, de permissivité ou de structure s'avère déterminante dans la mesure où toute modification, fut-elle minime, des modèles de spécifications et/ou des modèles du système à contrôler (tout en restant cohérent vis-à-vis des systèmes ou contraintes représentées) conduit invariablement à des résultats de synthèse très différents, pouvant aller jusqu'à l'absence de solution.
D'autre part, la taille des automates synthétisés restant relativement importante : nous obtenons des automates de [20 états, 63 transitions], [138 états, 558 transitions] et [38 états, 43 transitions] pour modéliser respectivement une commande de prise/pose à l'aide d'un vérin et d'une ventouse, une commande de déplacement à l'aide de 3 vérins (8 positions atteintes) et leur synchronisation pour réaliser un cycle de transport d'un produit d'une position à une autre !

Pour toutes ces raisons, il nous semble que l'utilisation industrielle de ces techniques ne peut se justifier que pour des systèmes soumis à de fortes contraintes de sécurité et de sûreté de fonctionnement pour lesquels la commande générée doit être réputée sûre, sans blocage et conforme aux spécifications ou lorsque les temps de développement d'un système de commande doivent être très fortement réduits comme cela peut-être le cas pour faire face à la variabilité des produits et à la réduction des temps de mise sur le marché.

# 3. APPLICATION DE LA SYNTHESE A LA RECONFIGURATION DES SYSTEMES DE COMMANDE

## *3.1 Problème*

Comme nous l'avons montré dans le premier chapitre de ce mémoire, l'introduction des technologies de l'information et de la communication dans les systèmes de production favorise les modèles d'organisation à la commande pour de très petites séries, voire pour la réalisation de **produits personnalisés** (Da Silveira *et al.* 2001).
Les changements de production induits par ces nouvelles organisations peuvent être relatifs à la **nature** de la production – ensemble des caractéristiques techniques et fonctionnelles de produits fabriqués en petites séries ou même de manière unique – la **qualité** – l'exigence d'une qualité supérieure à celle initialement prévue peut nécessiter l'engagement de ressources de transformations capables de l'obtenir – ou la **quantité** des produits. Dans tous les cas, ces changements peuvent conduire à l'ajout ou à la suppression de certaines ressources matérielles (par exemple des machines ou des robots dans un système manufacturier) par rapport à l'ensemble de celles engagées dans la production en cours et/ou la modification des programmes de commande initialement mis en œuvre relativement à ces ressources.

En réponse à ces préoccupations, l'Engineering Research Center for Reconfigurable Manufacturing System (ERC/RMS) de l'université du Michigan introduit le concept de système manufacturier reconfigurable défini de la manière suivante : *un système reconfigurable est conçu afin de pouvoir **rapidement adapter** sa capacité de production et ses fonctionnalités en réponse à de nouvelles circonstances en réarrangeant ou en changeant ses composants* (Mehrabi *et al.* 2000).

La solution classique à ce problème repose sur la flexibilité des systèmes de production et de leur commande. Elle consiste à pré-intégrer à la commande l'ensemble des potentialités de contrôle du système exploitant les redondances fonctionnelles entre les machines, ainsi que les règles de commutation entre les diverses stratégies de commande (Toguyeni *et al.* 2006). Même si la norme IEC61499 fournit un cadre de modélisation parfaitement adapté grâce à la séparation entre algorithmes de traitement et graphes de contrôle d'exécution (ECC) (Figure 30) et aux bibliothèques de composants réutilisables (Auinger *et al.* 2005), cette approche conduit à une augmentation très sensible de la complexité des systèmes de commande, voire à des phénomènes d'explosion combinatoire à mesure que le nombre de machines et la variabilité des produits augmentent.

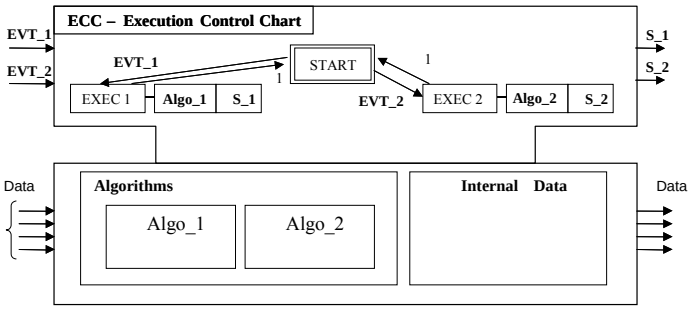



Figure 30. Bloc fonctionnel de la norme IEC 61499

La deuxième solution consiste à définir, avec un **cycle de développement très court** (voire en ligne dans certains cas), une nouvelle commande utilisant au mieux les capacités de ces systèmes de production pour répondre aux missions qui lui sont confiées (réactions à des aléas de fonctionnement ou à une variation de la demande) sous contraintes de qualité de service [O2][C19]. Dans ce contexte, la reconfiguration dynamique des systèmes de commande est définie par (Xu *et al.* 2002) comme un processus bouclé (Figure 8) incluant :

- la définition du **contexte de reconfiguration** permettant notamment d'identifier quand, où et pourquoi une reconfiguration est exigée ; cette phase peut reposer sur des processus de surveillance et de diagnostic dans le cas où la reconfiguration répond à une ou plusieurs défaillances du système de production,
- **l'élaboration de la stratégie de commande** la plus appropriée, c'est-à-dire la définition des règles de routage du produit – séquences de transformations fonctionnelles à appliquer au produit pour le faire passer de l'état brut à l'état fini – et les règles de commande des ressources – gestion des opérations exécutables par les ressources et l'ensemble des transferts permettant de les interconnecter.
- **l'implantation de la nouvelle configuration** incluant notamment la génération de composants de commande exécutables et la gestion de la commutation entre l'ancienne et la nouvelle configuration de commande.

Les défis posés par la reconfiguration dynamique concernent essentiellement la contrainte temporelle très forte relative aux phases d'élaboration et d'implantation d'une nouvelle configuration de commande. En d'autres termes, l'objectif est de proposer des méthodes, modèles et outils susceptibles de générer et d'implanter le plus rapidement possible une commande admissible au regard des missions qui lui sont confiées.

## *3.2 Contribution*

### 3.2.1 Elaboration d'une configuration : apport des techniques de synthèse

En vue de réduire les temps d'élaboration d'une nouvelle configuration, les techniques de synthèse, qui permettent de générer automatiquement un système de commande à partir des modèles de procédé et de spécification, apparaissent comme une voie très intéressante.

En ce sens, (Henry *et al.* 2004) propose une approche de synthèse, basée sur des algorithmes d'ordonnancement, permettant de particulariser la commande pour la réalisation d'un produit spécifique en intégrant dans les modèles de procédé non seulement les caractéristiques fonctionnelles et comportementales des machines mais aussi les effets qu'elles induisent sur le produit. (Qiu *et al.*, 2003) introduit la notion de *virtual production line* qui distingue clairement la synthèse des **règles de routage du produit** à travers les différentes ressources de l'atelier, de la synthèse des **règles de commande des différentes opérations** de transformations ou de transport réalisées par ces ressources.
Ce découpage entre commande relative au routage du produit et commande relative aux ressources peut être considéré comme une architecture d'implantation possible du concept de systèmes contrôlés par le produit. En effet, ce concept préconise d'embarquer des capacités informationnelles, décisionnelles voire opérationnelles dans le produit afin notamment de lui attribuer un comportement actif dans l'organisation et la conduite de la production. Ce comportement actif peut être matérialisé par un contrôleur, associé à chaque instance de produit, devant assurer son routage dans un système de production et déclencher l'exécution d'opérations par les ressources. La formalisation de ce cadre de raisonnement conduit à raffiner le prédicat proposé par (Fusaoka *et al.* 1983) :

$$\textit{Unknown Control Rules} \wedge \textit{Dynamics} \supset \textit{Behavioural Goal (P1)}$$

en deux prédicats :

$$\textit{Manufacturing system capabilities} \wedge \textit{Unknown product control rules} \supset \textit{Product manufacturing plan (P3)}$$

$$\textit{Resource dynamics} \wedge \textit{Unknown resource control rules} \supset \textit{Resource capabilities (P4)}$$

respectivement relatifs aux propriétés à satisfaire pour générer les **règles de routage** d'un produit à partir de ses spécifications (gamme logique) et des capacités des ressources de production (**P3**), et aux propriétés à satisfaire pour générer la **commande des ressources** (**P4**). Il est à noter que ces deux prédicats ne sont pas indépendants dans la mesure où la spécification de chacune des ressources du système (*resource capabilities* de P4) sert de base à l'élaboration du modèle des capacités du système de production (*Manufacturing System Capabilities* de P3) au travers des opérateurs de projection et de fusion détaillés par la suite.
Ainsi la phase d'élaboration d'une nouvelle reconfiguration peut être supportée par les techniques de synthèse déjà utilisées dans la première partie de ce chapitre pour générer automatiquement le **contrôle du routage** spécifique à chaque instance de produit, afin de permettre leur personnalisation, mais également la **commande des ressources**, en exploitant leurs degrés de flexibilité pour configurer les opérations qu'elles réalisent [R5][C17]. Si les algorithmes de synthèse permettent la génération

automatique des superviseurs, l'élaboration des modèles de procédé et de spécification reste une activité non automatisable qui peut pénaliser à la fois les temps d'obtention d'une nouvelle commande mais également la qualité des résultats de synthèse. Nous avons donc cherché à systématiser l'obtention de ces modèles.

Spécifications produit (prédicat P3)

Les modèles de spécification utilisés pour synthétiser les règles de routage sont basés sur une représentation ordonnée des différents états de transformations du produit sous la forme d'un automate à états finis, dans lequel n'apparaissent que les comptes-rendus (*noté RP pour report*) relatifs aux transformations réalisées. Cet automate correspond, d'un point de vue du contrôle par le produit, aux « séquences logiques » d'états morphologiques et/ou spatiaux du produit (Figure 31). Il est à noter qu'une extraction automatique de ces informations à partir d'un modèle de données MES, tel que celui proposé par la norme IEC62264, est envisageable.

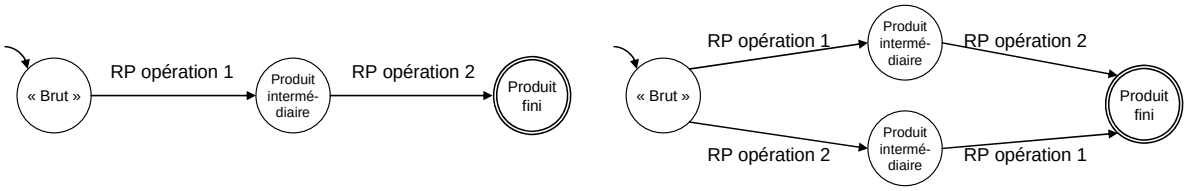



Figure 31. Exemples de modèles de spécifications du produit

Modèles des ressources (prédicats P3 et P4)

La modélisation des ressources est limitée dans le cadre des **modèles de procédé du prédicat P3**, à l'énumération des multiples opérations qu'une ressource peut exécuter. L'alphabet de ces modèles se compose donc de demandes d'opération (*RQ OPk*) et de comptes-rendus (*RP OPk*) où *k* est l'identifiant d'une opération.
Les ressources sont décrites par une famille d'états *travail* (chaque état de cette famille correspond aux différentes opérations que la ressource peut exécuter) et un état *attente.* Les ressources de transformation sont donc représentées par un automate à n+1 états où n est le nombre d'opérations supportées par la ressource. Il est à noter que l'état d'attente est un état marqué puisqu'il est considéré comme la fin légale d'une séquence de travail et correspond à un état d'équilibre. Les ressources de transport sont modélisées de manière légèrement différente dans la mesure où les états stables correspondent aux différentes localisations atteignables par la ressource de transport ce qui leur confère un statut d'état marqué.

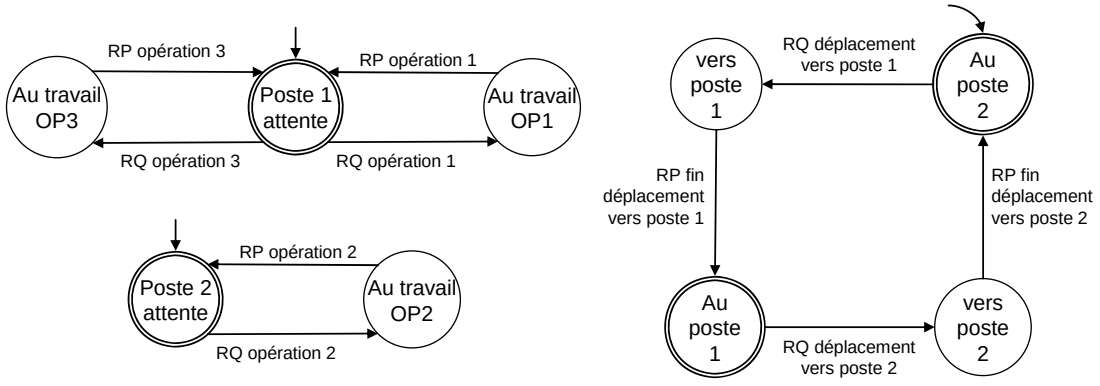



Figure 32. Modélisation des ressources de transformation et de transport

Ces modèles servent de base à l'élaboration des :

- **modèles de spécification du prédicat P4** : dans le cadre de la démarche de synthèse itérative et modulaire présentée dans la première partie de ce chapitre, les spécifications associées au niveau hiérarchique le plus élevé correspondent à un raffinement des modèles des ressources de la Figure 32. Nous avons démontré dans [R6] que les modèles de la Figure 32 sont équivalents à la **projection** des modèles de spécifications utilisés dans la démarche de synthèse itérative, en ne conservant que les événements en interactions avec les contrôleurs de produit (de type RQ et RP),
- **modèles de procédé pour le prédicat P3** à l'aide d'un **opérateur de fusion** permettant d'agréger les modèles des différentes ressources de transformation et de transport selon la topologie de l'atelier. Cet opérateur de fusion est justifié par le fait que la composition des modèles de ressources de transformation et de transport ne peut se limiter à une composition synchrone trop permissive pour prendre en compte les contraintes de localisation imposées par la topologie de l'atelier.

L'opérateur que nous avons proposé, dans l'esprit de la composition conditionnelle présentée par Košecká (1995), permet donc d'agréger les automates des modèles de ressources pour obtenir un modèle du système de production:

- les états des automates sont fusionnés selon des classes d'équivalence d'états qui associent les capacités des ressources en termes d'opérations à leurs localisations spatiales,
- les fonctions de transition sont appliquées sur leurs classes d'équivalence d'états respectives.

Plus formellement, l'opérateur de fusion est défini pour deux automates G1 et G2 :

$$G1 = \{E1, \Sigma 1, \alpha 1, e_0, E1_m\} \quad \text{et} \quad G2 = \{E2, \Sigma 2, \alpha 2, e2_0, E2_m\}$$

Soit la fonction *sit* définie sur $E1 \cup E2 \rightarrow \Re^3 \times F$ qui associe à un produit dans un état $e \in E1 \cup E2$ des coordonnées spatiales (sur $\Re^3$) et des opérations en cours sur le produit (sur $F= \{Op_1, Op_2, \ldots Op_n, 0\}$. Cette fonction *sit* sert de base à la définition d'une relation d'équivalence telle que deux états $e1_i$ et $e2_j$ seront considérés équivalents si $\exists (e1_i , e2_j) \in E1 \times E2$ / *sit* $(e1_i) =$ *sit* $(e2_j)$. La classe d'équivalence pour e est notée ê.

La fusion consiste alors à rechercher tous les $e2_j \in E2$ (respectivement $e1_i \in E1$) qui satisfassent la relation d'équivalence pour chaque $e1_i \in E1$ (respectivement $e2_j \in E1$). La classe d'équivalence notée $\hat{e}1_i$ (respectivement $\hat{e}2_j$) fusionne les états $e1_i$ et $e2_j$ avec :

$$\hat{e}1_i = \begin{cases} (e1_i , e2_j) \text{ si } \exists\, e2_j \in E2 \text{ / } sit\ (e1_i) = sit\ (e2_j) \\ e1_i \text{ sinon} \end{cases}$$

$$\hat{e}2_i = \begin{cases} (e1_i , e2_j) \text{ si } \exists\, e1_i \in E1 \text{ / } sit\ (e1_i) = sit\ (e2_j) \\ e2_j \text{ sinon} \end{cases}$$

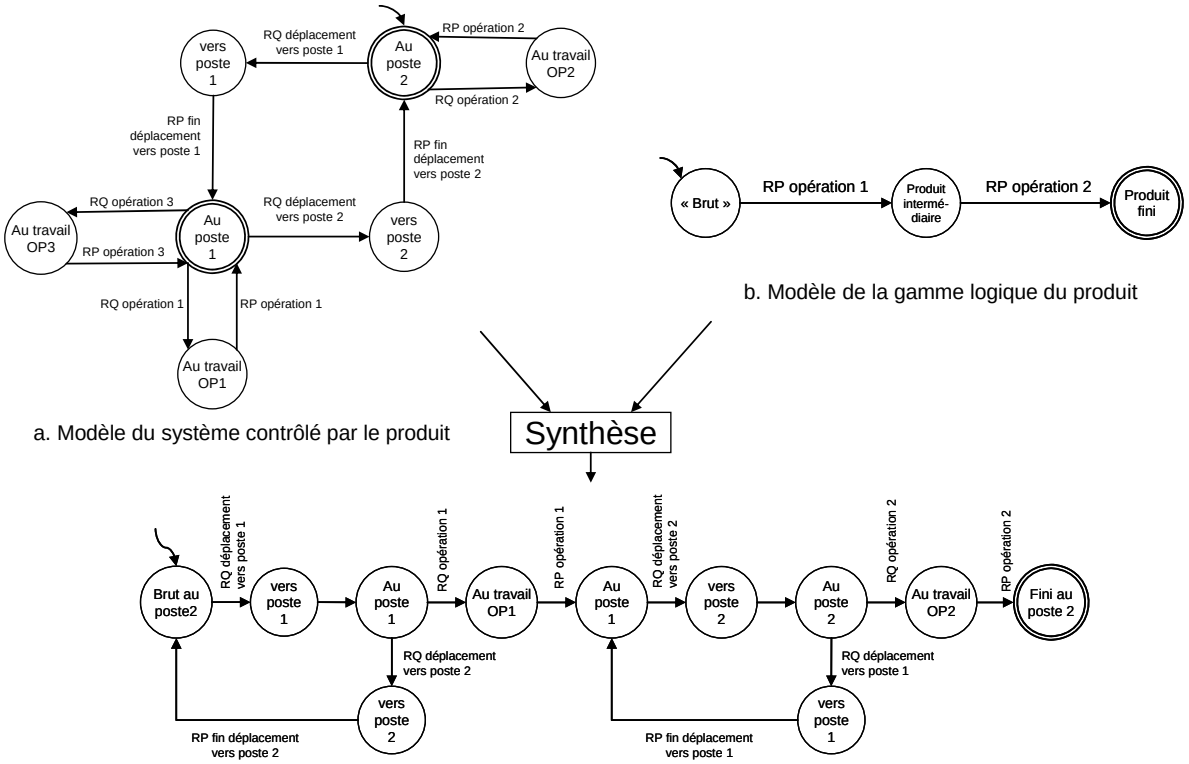
L'automate résultant $G = \{E, \Sigma, \alpha, \hat{e}_0, E_m\}$ est défini comme suit :

- $E = \{\hat{e}1_i\} \cup \{\hat{e}2_j\}$
- $\Sigma = \Sigma 1 \cup \Sigma 2$
- pour $\hat{e} \in E$, $\sigma \in \Sigma$, $\alpha : X \times \Sigma \rightarrow X$

- $\alpha(\hat{e}, \sigma) = \begin{cases} \alpha(\hat{e}1_i, \sigma_k) = \hat{e}1_p \text{ pour } \sigma_k \in \Sigma 1 \text{ tel que } \alpha 1(e1_i, \sigma_k) = e1_p \\ \alpha(\hat{e}2_j, \sigma_h) = \hat{e}2_q \text{ pour } \sigma_h \in \Sigma 2 \text{ tel que } \alpha 2(e2_j, \sigma_h) = e2_q \end{cases}$
- $\hat{e}_0 = \hat{e}1_0$
- $E_m = \{\hat{e}1_{m,i} / e1_{m,i} \in E1_m\} \cup \{\hat{e}2_{m,j} / e2_{m,j} \in E2_m\}$

Synthèse du superviseur associé au produit (prédicat P3)

Appliqué aux modèles de ressources de la Figure 32, cet opérateur aboutit au modèle de la Figure 33.a correspondant à une représentation topologique des capacités du système de production et de ses ressources. Ce modèle est utilisable comme modèle de procédé pour réaliser la synthèse du superviseur (prédicat P3) assurant le routage d'une instance donnée de produit dans l'atelier en fonction des caractéristiques définies par sa gamme logique Figure 33.b. Le superviseur ainsi obtenu représente donc l'ensemble des trajectoires admissibles compte tenu des ressources de production engagées (Figure 33.c).




a. Modèle du système contrôlé par le produit

b. Modèle de la gamme logique du produit

c. Ensemble des trajectoires de fabrication contrôlées par le produit

Figure 33. Synthèse du superviseur associé au produit

### 3.2.2 Implantation d'une nouvelle configuration

Les **superviseurs associés aux produits** représentent l'ensemble des trajectoires admissibles de fabrication. Afin que le produit puisse piloter sa fabrication en temps réel, il est nécessaire de préciser les règles de choix des trajectoires que le produit aura à réaliser. Pour cela, nous proposons de transformer les superviseurs générés en graphes pondérés selon des critères de coût, de disponibilité des ressources ou encore de temps de fabrication. Le problème revient alors à faire une recherche de chemin dans un graphe et des solutions ont été proposées en utilisant des coûts statiques ou dynamiques liés à chaque transition, ou des algorithmes d'optimisation,

tels que l'algorithme de Dijkstra (1959). Selon le choix du type de pondération sur les transitions (statique ou dynamique), le choix d'une trajectoire pourra être réalisé avant le lancement en fabrication du produit ou bien en temps réel pendant la fabrication. Deux solutions d'implantation du superviseur pondéré sont envisageables :

- la première consiste à définir un pilote centralisé, comparable aux machines d'échanges proposées par (Munerato 1988) pour synchroniser les échanges dans un atelier robotisé. Ce pilote enregistre les demandes émises par les contrôleurs des différentes ressources, transmet ces demandes à une ressource sollicitée (concept de postes génériques) en fonction des critères de pondération, réalise l'allocation de la ressource retenue en fonction de sa disponibilité et la libère lorsque l'opération requise est terminée. Cette approche s'apparente à l'implantation de l'architecture de reconfiguration proposée par (Toguyeni *et al.* 2006).
- la seconde solution consiste à utiliser l'architecture d'implantation du contrôle par le produit. Dans le cas du système flexible de l'AIP-PRIMECA qui a servi de support de test à notre proposition, le produit se déplace dans l'atelier sur une palette équipée d'une étiquette électronique. Le superviseur peut alors être codé sous la forme d'une structure de données et d'un pointeur désignant l'état actif et le (ou les) événement(s) autorisé(s). Lorsque le produit arrive sur un poste de travail, il se positionne devant un coupleur de lecture/écriture de l'étiquette, associé à ce poste de travail. La commande du poste peut alors accéder aux événements autorisés (demandes) et y répondre.

Cette dernière solution, plus pragmatique vis-à-vis de notre plate-forme d'expérimentation a été simulée avec succès à l'aide de modèles réalisés sur un outil de simulation en temps réel des S.E.D. (outil ControlBuild[14]).

En ce qui concerne la **reconfiguration de la commande des ressources**, nous avons opté [R6][C18] pour une approche modulaire basée sur l'architecture de (Xu *et al.* 2002) et sur le formalisme de la norme IEC 61499 (Figure 34).

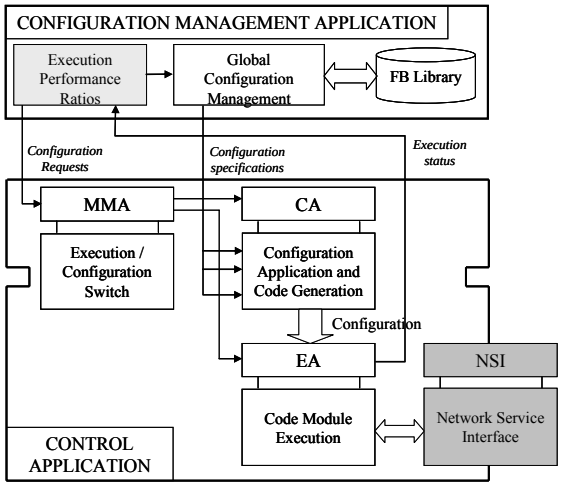


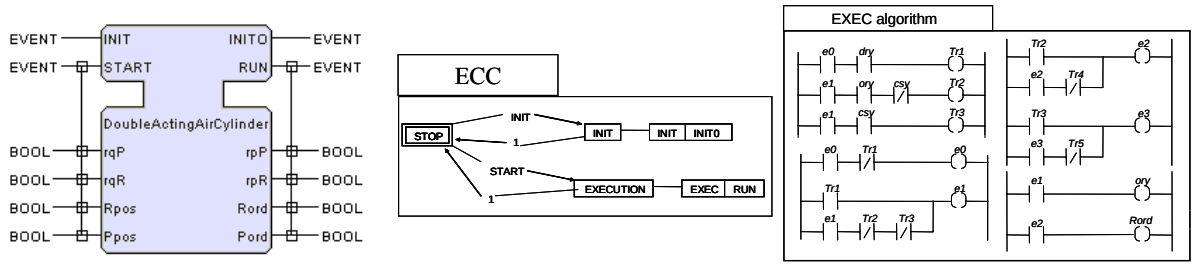

Figure 34. Architecture pour la reconfiguration dynamique (Xu *et al.* 2002)

Dans cette architecture, la partie intitulée « **Configuration Management Application** » concerne essentiellement la phase d'élaboration d'une nouvelle

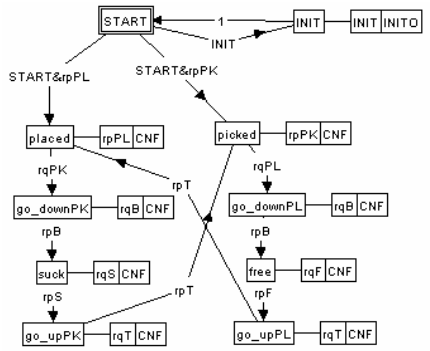
[14] ControlBuild, TNI-Software, http://www.tni-software.com/fr/produits/controlbuild/

configuration et est supportée dans notre cas par le processus de synthèse de la commande. La partie « Control Application » concerne la phase d'implantation :

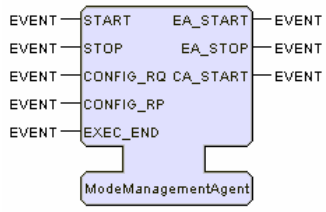
- l'agent CA (**Configuration Agent**) génère la structure de commande sous forme de blocs fonctionnels à partir des résultats de la synthèse : les superviseurs de plus bas niveau correspondant aux équipements technologiques sont implantés et sauvegardés en bibliothèque sous la forme des agents NSI (**Network Service Interface**)(Figure 35.a), les superviseurs de coordination sont implantés sous la forme d'automates dont certains états sollicitent l'exécution d'autres blocs fonctionnels (Figure 35.b),
- l'agent EA (**Execution Agent**) exécute la structure générée et informe l'agent de gestion des modes lorsqu'un point de configuration est atteint (Exec_end),
- l'agent MMA (**Mode Management Agent**) gère la commutation entre deux configurations (Figure 35.c) en autorisant alternativement l'exécution de l'agent CA (Configuration Agent) ou de l'agent EA (Execution Agent).

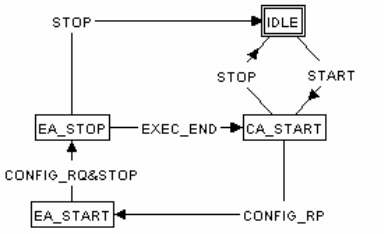



a. Network Service Interface (superviseurs « actionneurs »)




b. superviseur de coordination




c. mode management Agent

Figure 35. Agents de gestion des modes, d'exécution et de configurations [C18]

Ce cadre de modélisation basée sur la mise en œuvre de blocs fonctionnels de la norme IEC 61499 a été appliqué à la formalisation d'une architecture de commande reconfigurable d'une plate-forme de l'AIP-PRIMECA (Système Flexible de Production) puis simulée sur l'outil FBDK[15].

## 4. CONCLUSION

Dans ce chapitre, nous avons montré l'intérêt des techniques de synthèse de la commande dans le cadre théorique de la commande par supervision. Nos principales contributions concernent la proposition d'une démarche itérative et modulaire de synthèse en adéquation avec les architectures actuelles des

[15] FBDK : Function Block Development Kit (FBDK), Calgary University, www.holobloc.com

automatismes industriels et son application à la reconfiguration des systèmes de commande en réponse à la variabilité croissante des produits. Dans ces deux cas, notre attention s'est plutôt focalisée sur les phases situées en amont de l'utilisation des algorithmes de synthèse afin de systématiser la construction des modèles de procédé et de spécifications et, sur les phases situées en aval, afin de proposer des mécanismes d'implantation efficaces.

Malgré des résultats tangibles, nous sommes néanmoins forcés de reconnaître que la synthèse de la commande reste une technique complexe extrêmement sensible au phénomène d'explosion combinatoire ainsi qu'à la qualité des modèles de procédés et de spécifications lui servant de points d'entrée. D'autre part, elle est essentiellement utilisable dans le cadre d'un processus de conception comportementale d'un système automatisé (prédicat P1) mais n'intègre pas les niveaux d'abstraction et la multiplicité des points de vue nécessaires pour couvrir la phase de spécification (prédicat P2). C'est essentiellement à ce dernier point que sont consacrés les deux chapitres suivants.

# III Approches semi-formelles en R&D industriels pour la validation/vérification des exigences

## 1. INTRODUCTION

Ce chapitre présente les travaux réalisés dans le cadre de **collaborations industrielles** pour assurer le respect d'exigences au cours du cycle de développement d'un système automatisé de production selon le prédicat P2 (*Système de commande* $\wedge$ *Système opérant* $\supset$ *Exigences Système*). Dans ce contexte de R&D industrielles, les modèles utilisés pour identifier puis représenter les diverses exigences des utilisateurs seront semi-formels. La traçabilité des exigences sera alors garantie par la proposition de **cadres méthodologiques** permettant d'allouer des exigences sur les fonctions, composants et équipements d'une architecture industrielle, la validation finale étant, dans la plupart des cas, obtenue par simulation. Les travaux présentés dans ce chapitre concernent :

- les projets européens en Actionnement et Mesure Intelligents (AMI), avec pour principale exigence de garantir **l'interopérabilité** des applications de commande distribuée. Notre principale contribution porte sur la définition d'une bibliothèque de services dans le cadre d'un processus de standardisation de profils d'équipements (services et communication)[R1][R10]. L'approche est basée sur un cadre structurant d'expression des besoins en commande, maintenance et gestion technique (CMM) puis sur des mécanismes de projection et d'allocation de ces exigences sur les architectures distribuées AMI [R1] afin d'en déduire les interfaces de communication.
- la collaboration avec l'INRS[16], avec pour objectif de garantir la **sécurité** de machines potentiellement dangereuses et intégrant une part prépondérante de logiciel. Notre principale contribution porte sur le couplage de modèles orientés « ingénierie système » tel que SysML et des modèles plus formels du domaine des S.E.D. [C22][TH3]. La démarche permet la structuration et le raffinement des exigences de « sécurité machine » préconisée par les normes en vigueur (IEC 61508, IEC 62061, IEC 12100, …) puis leur projection et leur allocation sur une architecture de composants matériels et logiciels dont le comportement peut être représenté à l'aide de modèles de S.E.D. et dont les propriétés identifiées par les diagrammes des exigences SysML peuvent être vérifiées par simulation ou avec des outils de model checking.
- la collaboration avec la société TRANE, avec pour objectif de proposer un environnement permettant d'évaluer la **pertinence** d'une architecture de pilotage de la fabrication par le produit à l'aide de la technologie RFID. Notre principale contribution porte sur l'identification des apports de cette technologie pour la synchronisation de lignes de fabrication en flux tirés à l'aide de la méthode design for six-sigma (DFSS) couplée à un environnement de simulation événementielle distribuée permettant de dissocier la modélisation des unités de production géographiquement réparties, de la modélisation de leurs systèmes de pilotage [R7][TH4].

[16] INRS: Institut National de Recherche et de Sécurité

# 2. INTEROPERABILITE DES SYSTEMES D'ACTIONNEMENT ET DE MESURE INTELLIGENTS

## *2.1 Problème*

Les projets européens ESPRIT en **Actionnement et Mesure Intelligents**[17], dont les principaux acteurs émanent du secteur de l'énergie, ont porté sur le développement du concept de **système intégré de contrôle-commande, de maintenance et de gestion technique**[18] en vue d'augmenter le partage d'informations entre ces trois îlots d'automatisation. Parmi les bénéfices attendus de l'intégration des fonctions de contrôle-commande, de maintenance et de gestion technique, nous pouvons citer l'optimisation des procédures de conduite par la mise à disposition des opérateurs d'informations fiables, validées et tenant compte des observations de maintenance, l'amélioration des procédures de maintenance en favorisant notamment les actions préventives, voire prédictives basées sur des informations de suivi des équipements et des fonctions, et enfin, la mise en œuvre d'une réelle gestion technique basée sur l'historisation des informations de conduite et de maintenance.
La solution proposée par les partenaires de ces projets repose sur la distribution de capacités de traitement, de mémorisation et de communication de l'information dans les actionneurs et capteurs [R8][R9][O1][C1][C2][C3]C4]. En effet, longtemps considérés comme des éléments triviaux, les actionneurs et capteurs manipulent traditionnellement une information analogique relativement pauvre et non fiabilisée pouvant entraîner des incohérences informationnelles aux conséquences parfois catastrophiques. La distribution d'une certaine intelligence dans ces équipements complète leurs missions classiques par des fonctionnalités relatives à la validation et la synthèse d'informations qui augmentent le degré de crédibilité des chaînes d'action et d'observation. Les équipements intelligents, connectés via un réseau de communication, fournissent ainsi une représentation informationnelle fiable du processus physique sur laquelle il est alors possible de bâtir une organisation intégrée des fonctions de contrôle-commande, de maintenance et de gestion technique (Figure 36).

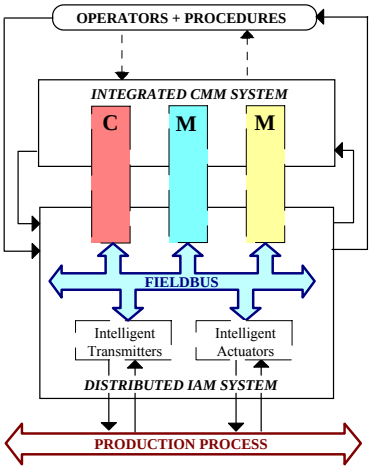



Figure 36. Le système d'Actionnement et de Mesure Intelligents dans son contexte

[17] ESPRIT II-2172 Distributed Intelligent Actuators and Sensors
ESPRIT III-6188 Prenormative Requirements for Intelligent Actuation and Measurement (IAM)
ESPRIT III-8244 European Intelligent Actuation and Measurement User Group
[18] CMM system : integrated Control, Maintenance & technical Management system

Dans ce contexte, les fonctions d'actionnement et de mesure sont distribuées dans les systèmes d'automatisation conventionnels de contrôle-commande, de maintenance ou de gestion technique mais également dans les actionneurs et capteurs intelligents. La principale difficulté posée par la mise en œuvre de ce type d'architecture distribuée intégrant des solutions propriétaires hétérogènes, indépendantes, voire concurrentes, concerne l'interopérabilité de ces dernières. De nombreuses définitions de l'interopérabilité existent dans la littérature :

- l'interopérabilité est la capacité des systèmes informatiques et des processus qu'ils supportent à échanger des données et de permettre le partage d'informations et de connaissances (EIF, 2004) ;
- l'interopérabilité est la capacité que possèdent deux ou plusieurs systèmes ou composants à échanger des informations puis à exploiter les informations venant d'être échangées (IEEE, 1990) ;
- l'interopérabilité est la capacité, approuvée mais non garantie par la conformité à un ensemble de standards, d'équipements hétérogènes, généralement développés par différents fabricants, à travailler ensembles dans un environnement en réseau (IEEE, 2000) ;
- l'interopérabilité est la capacité de systèmes, unités ou ressources à fournir des services ou à utiliser des services d'autres systèmes, unités ou ressources et d'utiliser ces services échangés pour leur permettre de coopérer de manière effective (DoD, 1994) ;
- l'interopérabilité est la capacité à communiquer, exécuter des programmes, transférer des données entre différentes unités fonctionnelles de manière à ce qu'un utilisateur n'ait besoin que de peu ou pas de connaissances sur les caractéristiques de ces unités (ISO/IEC 23821, 1993).

Si l'on se réfère à la classification proposée par le SEMATECH[19] et relative aux capteurs et actionneurs, nous pouvons distinguer trois types d'interopérabilité (Staroswiecki & Bayart 1996):

- une interopérabilité de **classe A** qui caractérise la capacité de plusieurs équipements à échanger de l'information ; ce type d'interopérabilité, souvent dénommé **interopérabilité de communication**, repose sur la définition de protocoles et de profils de communication,
- une interopérabilité de **classe B** qui caractérise la capacité de plusieurs équipements à coopérer à la réalisation d'un objectif commun ; cette interopérabilité, aussi qualifiée d'**interopérabilité de services** par (Thomesse 1999) impose la complémentarité des services délivrés par les équipements (fonctions, informations, comportement) et repose sur des langages de description de services (tels que DDL Device Description Language reconnu par la norme ISO 61804),
- une interopérabilité de **classe C** qui caractérise la capacité d'équipements à être interchangeables pour la réalisation d'une mission donnée.

Dans le cadre des projets européens en Actionnement et Mesure Intelligents, le challenge consistait à prendre en compte conjointement les besoins fonctionnels en contrôle-commande, en maintenance et en gestion technique exprimés par les utilisateurs et les contraintes opérationnelles liées à l'offre technologique afin d'établir une proposition pré-normative :

[19] SEMATECH*: Device interoperability guideline for sensors, actuators and controllers*, (1995), Technology Transfer Standard 94102567A-STD, http://www.sematech.org

- d'interfaces fonctionnelles des équipements impliqués dans le système distribué d'actionnement et de mesure en vue d'assurer leur interopérabilité de services par la description d'un ensemble de services élémentaires de contrôle-commande, de maintenance et de gestion technique, indépendamment du support de communication,
- de compagnons de communication qui assurent l'adaptation des interfaces fonctionnelles à un réseau particulier (optimisation de la charge du réseau, construction des trames, ...).

Pour atteindre cet objectif, le processus de modélisation ne peut pas reposer uniquement sur l'intégration de solutions propriétaires indépendantes mais nécessite la coopération des utilisateurs et des offreurs autour d'un modèle de référence commun permettant l'identification des besoins des utilisateurs, la définition de services (fonctions, informations) satisfaisant ces besoins et enfin l'allocation de ces services sur les équipements de l'architecture A.M.I. et C.M.M. en tenant compte des contraintes technologiques imposées par les fournisseurs. Notre principale contribution au projet PRIAM est relative à la définition de ce modèle de référence et au développement de l'outil informatique support [R1][R10][C5][C7].

## *2.2 Contribution*

### 2.2.1 Expression des besoins

Le modèle des besoins fonctionnels représente l'ensemble des besoins relatifs au fonctionnement d'un système A.M.I. Le problème méthodologique posé consiste à guider les utilisateurs d'un système A.M.I., tant en phase d'exploitation (utilisateurs finaux de conduite, maintenance et gestion technique) qu'en phase d'ingénierie ou de conception, dans l'expression de leurs besoins fonctionnels, informationnels et comportementaux.

Dans cette optique, plusieurs niveaux de classification des exigences ont été proposés afin de permettre à un agent modélisateur d'exprimer de manière systématique les services qu'il attend d'un système d'actionnement et de mesure intelligent :

- selon les points de vue contrôle-commande, maintenance et gestion technique,
- selon un regroupement des fonctions d'actionnement et de mesure en sept classes plus génériques (documentation, configuration, paramétrage, test, état opérationnel, historiques, modes d'utilisation)

Les exigences, exprimées sous la forme de listes de requêtes et de comptes-rendus représentant les flux informationnels transformés par les fonctions intelligentes d'Actionnement et de Mesure, peuvent être spécifiées selon deux niveaux d'abstraction :

- à un niveau fonctionnel indépendant de toute répartition dans des équipements et de toutes contraintes organisationnelles de l'entreprise (salle de commande, local électrique, conduite locale, …)
- à un niveau organisationnel qui tient compte de la répartition géographique des fonctions propre à chaque site de production.

Notons, qu'en retour, ces mécanismes de structuration (classification des fonctions A.M.I., points de vue, niveaux d'abstraction) vont aider l'agent à identifier des informations auxquelles il n'aurait peut-être pas pensé à priori.
Afin de fédérer les diverses contributions des utilisateurs et des offreurs et de permettre la convergence vers une proposition normative, un outil informatique supportant notamment la phase d'expression des exigences et les classifications proposées a été développé dans le cadre du projet ESPRIT III-PRIAM 6188. A titre d'exemple, la Figure 37 présente une liste de requêtes et de comptes-rendus relative aux exigences d'un agent de maintenance.

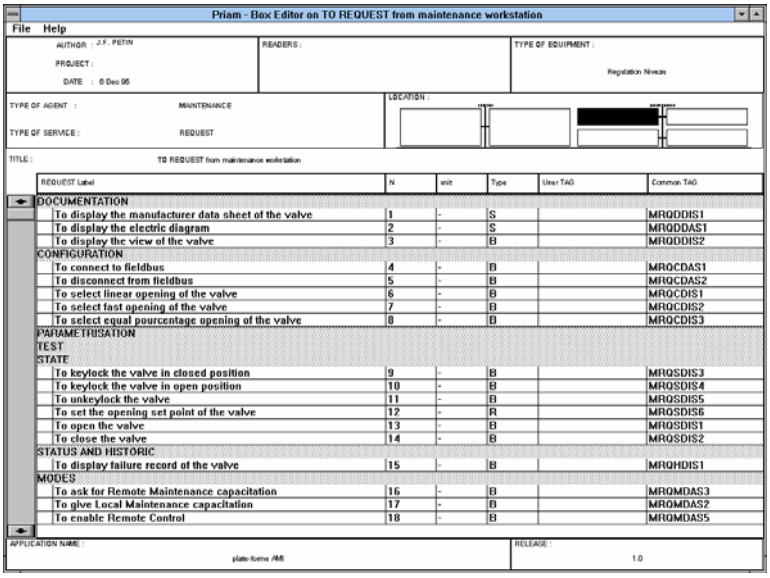


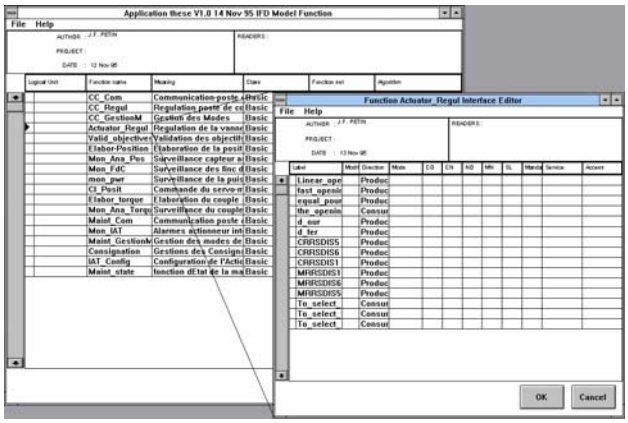

Figure 37. Exemple d'expression des exigences pour un agent de maintenance

### 2.2.2 Définition d'une bibliothèque de services

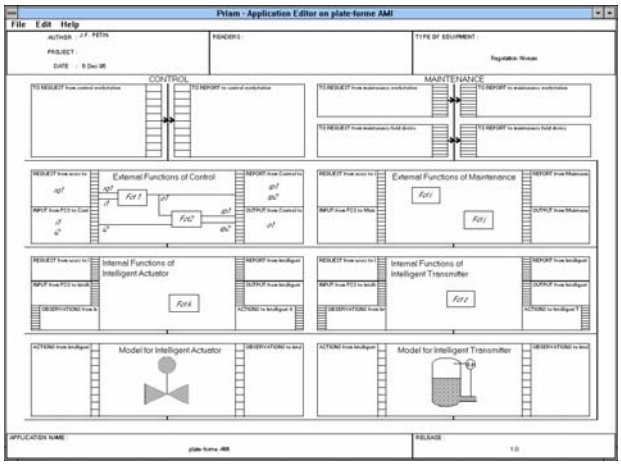
Le processus de modélisation d'un système A.M.I. se poursuit par la définition d'une bibliothèque de services permettant de répondre aux exigences identifiées lors de la phase précédente. Ces services sont décrits sous la forme d'une liste de fonctions élémentaires incluant une description de leurs interfaces informationnelles et de leur comportement. Les interfaces informationnelles d'une fonction sont constituées des requêtes et comptes-rendus associés aux services que rend la fonction ainsi que d'informations additionnelles nécessaires à son exécution. Le comportement est décrit soit sous une forme algorithmique textuelle, soit de manière plus détaillée dans un des langages de la norme IEC1131-3. Ces modèles comportementaux peuvent être indépendants de toute solution technologique lorsqu'ils sont spécifiés par des utilisateurs. La figure Figure 38a présente un exemple de listes de fonctions et de leurs interfaces informationnelles sur l'outil PRIAM.

A partir de la liste des fonctions élémentaires d'actionnement et de mesure, il est alors possible de procéder à leur allocation sur les différents équipements impliqués dans le système A.M.I. : actionneurs, capteurs, équipements centralisés de contrôle-commande, de maintenance et de gestion technique. La Figure 38b présente un exemple d'allocation des fonctions élémentaires sur deux équipements centralisés de contrôle-commande et de maintenance et sur deux équipements de terrain. Il est à noter que la connaissance des interfaces informationnelles de chacune des

fonctions élémentaires permet la génération automatique, lors de la phase d'allocation, des interfaces informationnelles de chacun des équipements impliqués.




a. identification des fonctions et de leurs interfaces informationnelles




b. Distribution des fonctions dans les équipements A.M.I

Figure 38. Identification et allocation des fonctions sur l'outil PRIAM

### 2.2.3 Interopérabilité : les interfaces fonctionnelles standard

Sur la base d'une liste exhaustive de fonctions A.M.I. résultant d'un consensus entre les différents utilisateurs et offreurs impliqués dans les projets européens en Actionnement et Mesure Intelligents, il a été possible de procéder à la définition d'une proposition pré-normative d'Interfaces Fonctionnelles.
Un ensemble d'attributs spécifiques vient compléter les attributs relatifs aux fonctions, aux informations ou aux flots de données (Figure 39) afin de définir une sémantique des Interfaces Fonctionnelles Standard permettant de construire sans ambiguïté les vecteurs de communication induits par la distribution des fonctions dans les équipements : classes normatives (obligatoire, optionnel, propriétaire), services de communication (continu, périodique, événementiel, …), informations qualitatives (index de validité, domaine de valeurs, …).

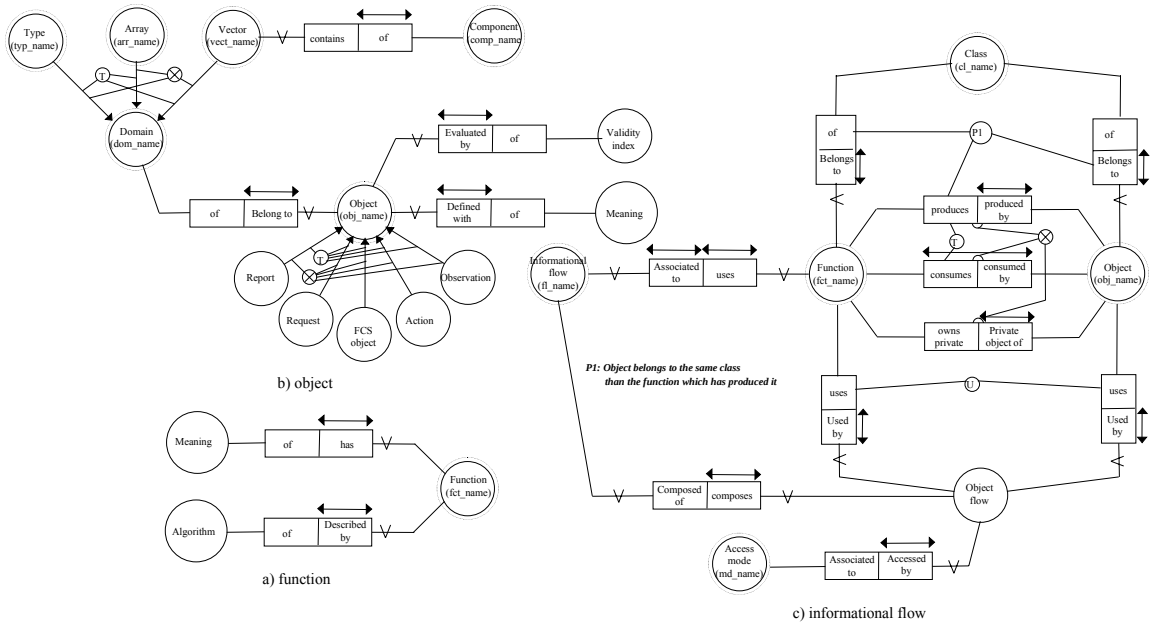



Figure 39. Sémantique des Interfaces Fonctionnelles Standard (formalisme NIAM)

### 2.2.4 Traçabilité des exigences

D'un point de vue de la traçabilité des exigences des utilisateurs mais aussi des offreurs, la démarche proposée et l'outil PRIAM support doit concilier :

- une démarche descendante selon le point de vue des utilisateurs qui repose sur l'expression des besoins, l'identification de fonctions élémentaires permettant d'y répondre et enfin leur allocation sur les équipements présents dans le système A.M.I. à développer,
- une démarche ascendante selon le point de vue des offreurs qui agrège des fonctions particulières spécifiées, développées et mises en œuvre par les fournisseurs.

Afin d'assurer la cohérence entre tous ces points de vue, plusieurs mécanismes ont été développés sur l'outil PRIAM (Figure 40):

- chaque fonction identifiée, spécifiée puis allouée sur un équipement fait systématiquement référence à une exigence spécifiée dans le modèle des besoins sous la forme de listes de requêtes/comptes-rendus,
- lors de l'allocation sur les équipements, un mécanisme de vérification permet de s'assurer que toutes les fonctions identifiées par les utilisateurs ont bien été allouées sur un équipement support permettant leur réalisation.

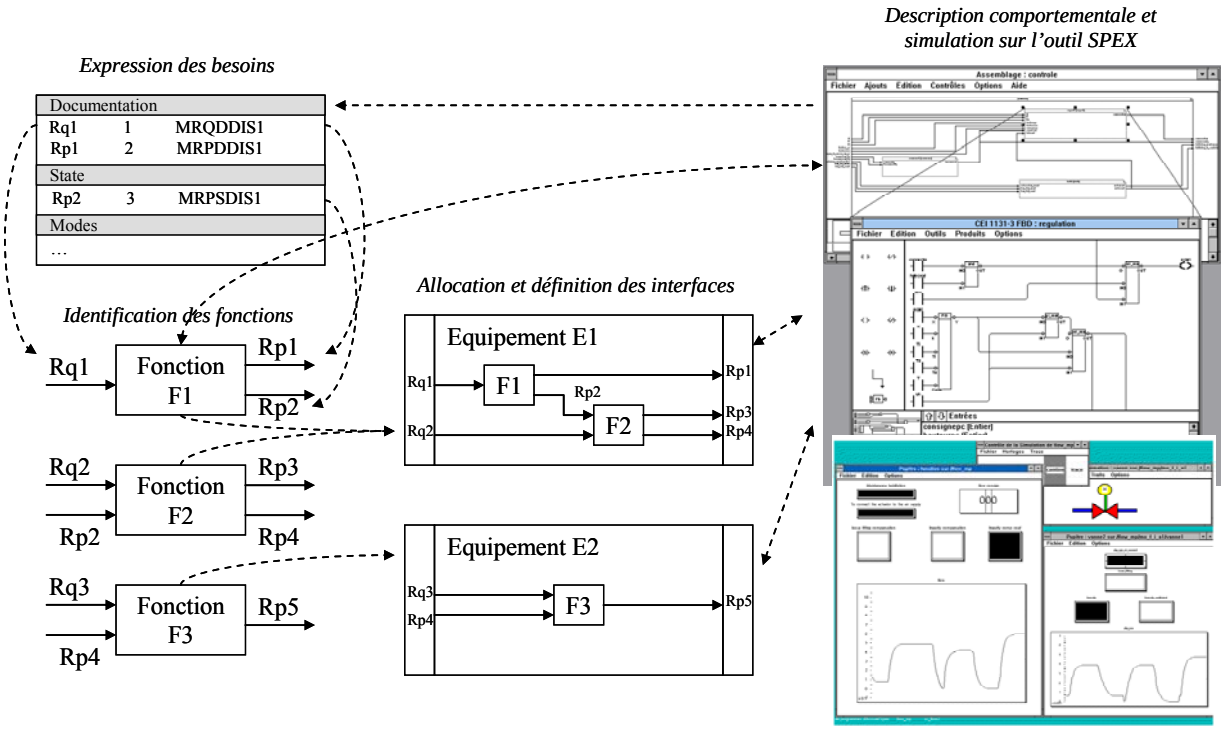



Figure 40. Traçabilité des exigences sur les outils PRIAM et SPEX

Enfin, d'un point de vue comportemental, l'intégration de l'outil PRIAM avec un outil de simulation comportementale en temps réel, l'outil SPEX, développé et commercialisé par un partenaire du projet PRIAM a été proposée. Cet outil permet de construire des comportements complexes à partir de comportements élémentaires dans des formalismes tels que le Grafcet, le langage à relais, le langage C ou plus récemment le langage à blocs fonctionnels de la norme IEC 1131-3 et de les simuler de manière interactive par l'intermédiaire d'un tableau de bord, ou de manière automatique par scénario de test et analyse de traces. Il est donc

parfaitement adapté pour les descriptions et simulations comportementales des fonctions A.M.I. qui permettent de s'assurer du respect des exigences exprimées par les utilisateurs aux différents niveaux de modélisation supportés par l'outil PRIAM (fonctions élémentaires, équipements intégrant un sous-ensemble de ces fonctions et enfin, système d'actionnement et de mesure dans sa globalité). Afin de réaliser une simulation dans un contexte émulé d'exploitation, l'outil PRIAM propose également, par l'intermédiaire de l'outil SPEX, de coupler les modèles comportementaux développés à des modèles d'émulation des constituants physiques (vannes, pompes, transmetteurs, …) du système d'Actionnement et de Mesure Intelligents.

# 3. SECURITE DES MACHINES INDUSTRIELLES

## *3.1 Problème*

Comme nous l'avons souligné dans le chapitre II, la pénétration des technologies de l'information et de la communication au cœur des systèmes automatisés de production conduit à des architectures complexes constituées d'un ensemble de composants de nature hétérogène. Les travaux développés dans le cadre des projets européens en Actionnement et Mesure Intelligents ont mis l'accent sur les problèmes d'interopérabilité engendrés par ce type d'architecture. Cet accroissement de la complexité des systèmes a également un impact sur leur sûreté de fonctionnement et leur sécurité. En effet, leurs propriétés ne sont plus réductibles aux propriétés de leurs constituants pris isolément (composants logiciels de commande, circuits électromécaniques ou électropneumatiques, actionneurs et capteurs, etc) mais émergent d'un réseau d'interactions entre ces constituants qui peut être à l'origine de comportements néfastes et difficiles à prévoir.

Dans le domaine des machines industrielles, la sécurité des opérateurs effectuant des opérations de chargement ou de déchargement manuel, de réglage ou de changement d'outil dans une zone « dangereuse » est pris en charge par un ensemble de dispositifs de protection (barrière immatérielle, dispositifs de commande sécurisée, arrêts d'urgence, …) dont la commande est traditionnellement réalisée en logique câblée, la logique programmée étant prohibée pour des raisons de fiabilité des parties hardware. L'avènement des automates programmables de sécurité (APIdS) a quelque peu changé la donne puisqu'ils offrent des garanties de sûreté de fonctionnement, notamment grâce à des architectures redondantes, permettant d'envisager le traitement des fonctions de sécurité par une logique programmée. Cette évolution se traduit par une augmentation de la complexité de la commande des machines industrielles due :

- à une imbrication des fonctions de commande relatives à leur exploitation et de celles relatives à leur sécurité,
- à une augmentation de la complexité des traitements de sécurité induite par le potentiel de la logique programmée.

Or si les APIdS apportent des réponses pour la partie matérielle de la commande, le traitement des fonctions de sécurité par une logique programmée ne peut être effectif (et envisageable) que si le niveau de sécurité atteint par l'ensemble « automate de sécurité + logiciel associé » est au moins équivalent à celui de la chaîne électromécanique.

D'autre part, le développement de machines industrielles potentiellement dangereuses est encadré par un ensemble de normes telles que :

- la directive 98/37/CE [DE 98/37/CE], couramment appelée directive « machines », qui définit un ensemble d'exigences essentielles de sécurité à mettre en place lors de la conception pour prévenir les risques potentiels,
- les normes dites « harmonisées » qui donnent présomption de conformité à la directive, comme la norme ISO 12100 qui définit une méthodologie et des principes techniques d'intégration de la sécurité pour les machines ou la norme CEI 62061, déclinaison au secteur de la machine de la norme CEI 61508 qui définit quatre niveaux de SILs (Safety Integrity Levels) dépendant de la gravité et de la probabilité de défaillance.

De manière générale, les normes traitant du développement de la partie logicielle présentent des prescriptions relatives au processus de développement. Les préconisations importantes mettent l'accent sur :

- la nécessité de tenir compte de l'**environnement du logiciel**, à savoir les interactions avec les parties matérielles de la machine ainsi que les interactions avec l'opérateur,
- l'utilisation de **composants logiciels** permettant d'augmenter la lisibilité de l'application et de faciliter leur validation, en particulier pour les composants supportant une fonction de sécurité,
- l'utilisation de **méthodes formelles** de développement notamment pour les applications de niveau SIL 4.

Si l'on se réfère à ces prescriptions normatives, il est donc nécessaire de mettre en œuvre un processus de développement intégrant :

- des modèles orientés « **système** » pour appréhender le fonctionnement global de la machine et de ses dispositifs de sécurité afin de considérer le logiciel de commande dans son environnement en incluant les comportements induits par les parties mécaniques ou électriques de la machine,
- des **modèles comportementaux** permettant la simulation ou la preuve de propriétés de chacun des composants logiciels supportant une fonction de sécurité afin de garantir que leurs propriétés intrinsèques contribuent efficacement à satisfaire les propriétés de sécurité identifiées au niveau « système ».

La principale difficulté réside dans la **projection** (et la traçabilité) de propriétés de sécurité exprimées, à un niveau « système », dans des formalismes semi-formels tels que UML ou SysML, sur des propriétés élémentaires relatives à chacun des composants logiciels pouvant être vérifiées par l'utilisation de modèles formels autorisant la simulation ou la preuve.

### *3.2 Contribution*

Dans le cadre de la thèse CIFRE de Dominique Evrot en collaboration avec l'INRS, nous avons donc cherché à combiner des approches orientées « système » avec des modèles comportementaux formels afin de pouvoir couvrir le plus complètement possible les diverses phases classiques d'automatisation industrielle (spécification, conception, implantation) en s'efforçant de montrer comment les diverses exigences de sécurité doivent être identifiées, formalisées, validées et/ou vérifiées, propagées et suivies (traçabilité) [C20][C22][C24].

Pour couvrir les aspects « système », l'outil de modélisation retenu est le langage SysML. SysML est un profil d'UML2 qui étend son champ initial d'application, le génie informatique, à l'ingénierie des systèmes. Il supporte la spécification, l'analyse, la conception et la validation d'une très large variété de systèmes et de systèmes de systèmes pouvant inclure des éléments logiciels et matériels, de l'information, des processus, des personnels, … Ce choix est notamment justifié par :

- la couverture en termes de modélisation offerte par SysML au travers des diagrammes statiques et dynamiques d'UML2 (Figure 41) auxquels il convient notamment d'ajouter le diagramme des exigences supportant la définition et la structuration des exigences sous la forme de diagrammes à objets ;
- la modélisation de la sémantique des diagrammes sous la forme de méta-modèles, ce qui permet d'envisager leur intégration avec les modèles de conception et de vérification formelle plus sereinement qu'avec des approches de modélisation dédiées « système » telles que SAGACE ou dédiée « ingénierie des exigences » telles que KAOS,
- l'outillage disponible sous la forme de plug-in intégrable aux outils de modélisation UML tels que Rational[20], Rapsody[21], Artisan Studio[22] ou encore MagicDraw[23], qui constitue un critère important dans le contexte à vocation industrielle de ces travaux.

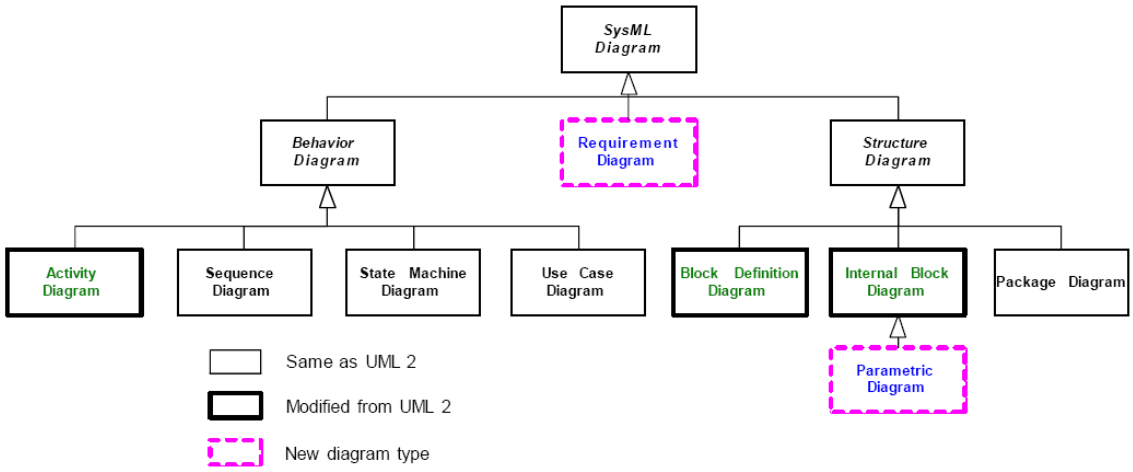



Figure 41. Correspondances entre les diagrammes SysML et UML2

### 3.2.1 Expression des exigences de sécurité

La modélisation des exigences est le résultat d'un processus complexe et itératif qui raffine un ensemble de besoins abstraits exprimés par les utilisateurs pour prendre progressivement en compte un ensemble de contraintes fonctionnelles et/ou techniques émanant des développeurs. Pour le cas d'une machine industrielle potentiellement dangereuse, ce processus doit être complété par un processus d'appréciation (estimation, évaluation) et de réduction (prévention, protection, …) du risque tel que celui préconisé par la norme ISO 12100.
Afin de structurer la modélisation de ces différents types d'exigences (dans notre cas, exigences fonctionnelles, exigences de sécurité, exigences techniques) à

[20] IBM
[21] I-Logix : www.I-Logix.com
[22] Artisan Software Tools, Inc : www.artisansw.com
[23] No Magic Inc., http://www.magicdraw.com/

différents niveaux d'abstraction, le diagramme des exigences de SysML formalise les liens entre exigences sous la forme de relations entre objets, notamment des relations :

- de **composition** qui traduisent le raffinement d'une exigence « mère » en un ensemble d'exigences « filles » et permettent ainsi d'aborder le problème de la modélisation des exigences à différents niveaux d'abstraction (Figure 42),
- de **dérivation** qui représentent un lien de dépendance (souvent causal) entre plusieurs exigences

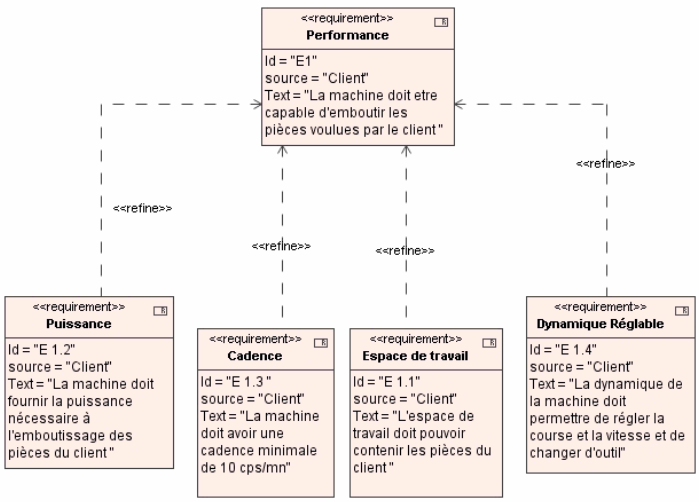



Figure 42. Raffinement d'exigences fonctionnelles pour une presse industrielle

Par rapport à la modélisation des besoins proposée dans le cadre du projet PRIAM, les relations entre objets du diagramme des exigences SysML présentent l'avantage de pouvoir structurer l'expression des exigences en :

- associant à chaque **exigence fonctionnelle** un ensemble de **propriétés de sécurité** que nous avons modélisées sous la forme d'un stéréotype de classe (Figure 43),
- **structurant** progressivement le processus d'analyse et de réduction du risque de manière duale au processus de raffinement des exigences fonctionnelles ; en d'autres termes, si une exigence abstraite E1 (associée à une propriété de sécurité P1) est raffinée en deux exigences E2 et E3 (respectivement associées aux propriétés de sécurité P2 et P3), cela signifie que les propriétés P2 et P3 impliquent la propriété P1. Dans le cadre de l'exemple d'application développé en collaboration avec l'INRS sur une presse industrielle, cette preuve de raffinement a été réalisée à l'aide du prouveur **COQ**.

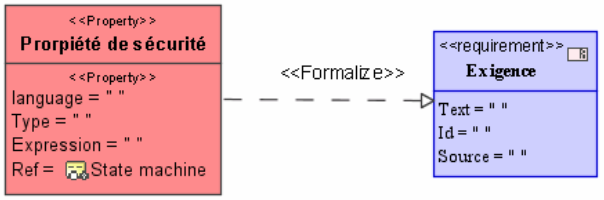



Figure 43. Stéréotype de classe pour les propriétés de sécurité

Enfin, de manière similaire à l'approche mise en œuvre dans le cadre des projets européens en Actionnement et Mesure Intelligents, il est alors nécessaire **d'allouer** les exigences identifiées sur un ensemble de **fonctions** ou **composants**. A la différence de PRIAM, ce processus d'allocation peut se réaliser de **manière**

**progressive** et conjointement au processus de raffinement des exigences, ce qui facilite la modélisation conjointe des points de vue « utilisateurs » et « offreurs ». En effet, les relations *satisfy* ou *allocate* permettent d'établir un lien entre un objet du diagramme d'exigences SysML (exigence ou propriété) et des objets décrits dans d'autres diagrammes SysML (activités, composants, cas d'utilisation, etc) quelque soit le niveau d'abstraction considéré (Figure 44).

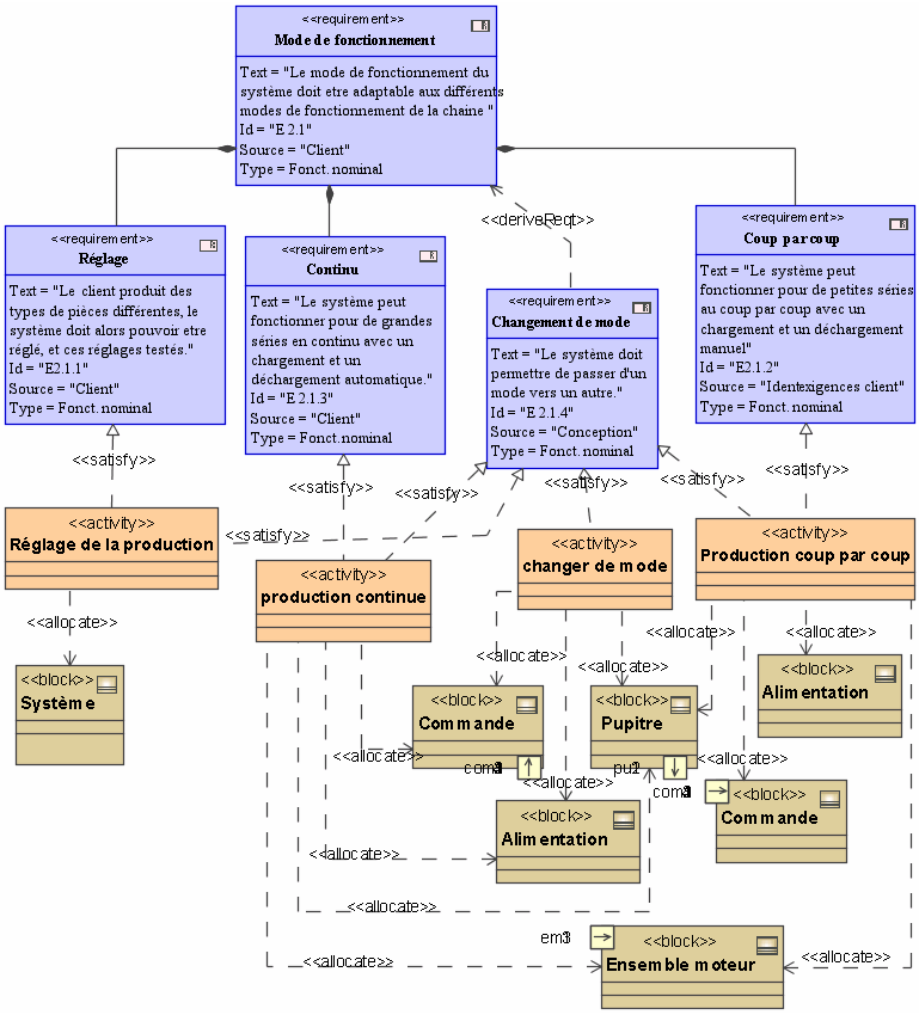



Figure 44. Exemple d'allocation des exigences pour une presse industrielle

### 3.2.2 Conception et Vérification des composants

Sur la base de cette analyse « système », il est alors possible de procéder au développement des composants identifiés, en particulier des composants de commande supportant les fonctions de sécurité. Pour les applications « machine industrielle » et dans le cadre de notre collaboration avec l'INRS, les modèles utilisés à ces fins sont les modèles classiques de l'automatique des S.E.D.
Comme nous l'avons montré, le principal intérêt de cette approche réside dans l'identification des propriétés de sécurité au niveau « système » conformément aux prescriptions normatives et à leur projection sur une architecture de composants d'automatismes, réduisant ainsi le problème inhérent à l'identification et à la modélisation des propriétés des composants à vérifier (Figure 45).

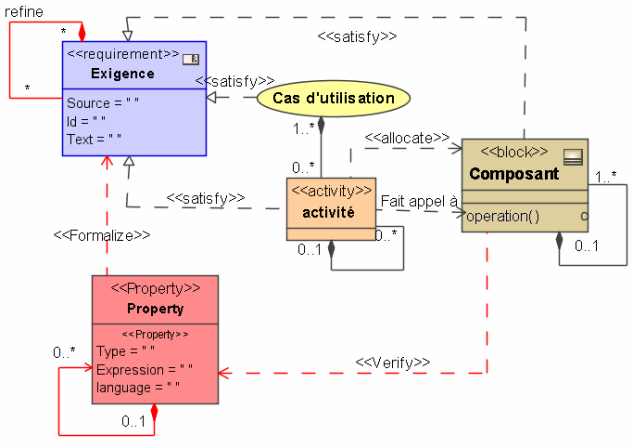



Figure 45. Traçabilité des exigences

Pour chaque composant d'automatisme, il est alors possible de procéder à sa vérification par :

- **simulation**, les propriétés héritées de la modélisation SysML nous fournissant alors les éléments indispensables à l'analyse de traces après exécution d'un scénario de tests ; dans le cadre de l'application développée avec l'INRS, l'outil de simulation utilisé est l'outil ControlBuild (évolution de l'outil SPEX utilisé dans le cadre des projets européens en Actionnement et Mesure Intelligents), les propriétés de sécurité étant injectées sous la forme de **post-conditions** dont le simulateur vérifie systématiquement le respect lors de l'exécution d'un scénario de test.
- **preuves de propriétés** à l'aide d'un outil de model checking ; dans le cadre de l'application développée avec l'INRS, l'outil de model checking utilisé est l'outil UPPAAL, les propriétés de sécurité identifiées par la modélisation SysML servant de base à la formalisation des axiomes en logique temporelle.

Afin de faciliter la prise en compte des propriétés de sécurité identifiées lors de l'analyse SysML par les outils de simulation et de model-checking, nous avons développé un schéma XML permettant l'échange de données entre les outils SysML (MagicDraw), de simulation comportementale (ControlBuild) et de model-checking (UPPAAL) grâce à des transformations XSLT (Figure 46).

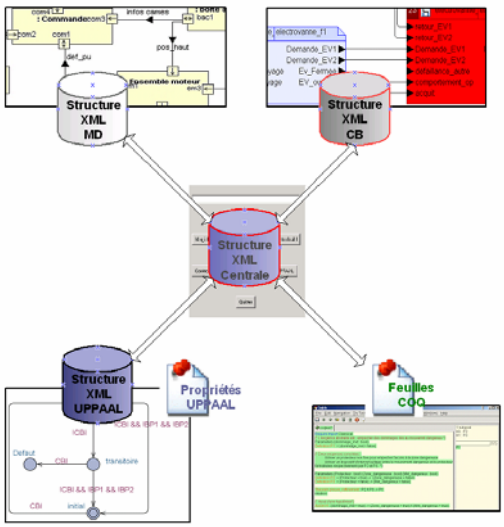



Figure 46. Architecture du démonstrateur

# 4. SYSTEMES CONTROLES PAR LE PRODUIT

## *4.1 Problème*

Le projet « Système Contrôlé par le Produit » du CRAN s'appuie sur une interprétation du concept de « système holonique de production » (Valckenaers 2001) pour rendre actif le produit dans l'organisation et la conduite de la production. Cette interprétation tire parti des progrès et de la miniaturisation croissante des technologies de l'information et de la communication (identification automatique, communication sans fil, micro-technologies, composants logiciels embarqués, etc) qui devraient permettre d'embarquer des capacités informationnelles, décisionnelles voire opérationnelles dans le produit.
Ce produit actif devient donc un objet composite vu comme un **bien matériel** par le système physique de fabrication et comme de **l'information** ou un service par les systèmes de contrôle et de pilotage de la production. De par sa nature, il assure une **synchronisation** entre les **objets physiques** du procédé et les **objets logiques** de contrôle et de gestion qui leur sont associés (Figure 4). Il conduit ainsi à repenser l'intégration hiérarchique ERP/MES/Automatisation des procédés de production pour rechercher le meilleur équilibre possible entre décisions centralisées et décisions distribuées en apparence antagonistes et faciliter ainsi le ré-ordonnancement local.

Dans le cadre de la thèse de Hind El Haouzi en contrat Cifre avec la société Trane, nous avons cherché à évaluer la pertinence du concept de « système contrôlé par le produit » sur une application industrielle réelle. Organisée en flux tirés selon le standard DFT (Demand Flow Technology) [24], la Société Trane, fabricant de climatiseurs et appareils réfrigérants désirait mettre en œuvre des solutions d'identification automatique telles que le RFID, en vue :

- d'améliorer la **synchronisation des flux de produits** aux points de couplage entre les lignes principales et les lignes secondaires d'alimentation afin de pouvoir anticiper les répercussions d'un aléa de production sur l'ensemble des lignes,
- de garantir la **traçabilité** des opérations de fabrication dans un contexte normatif contraignant (en particulier en ce qui concerne les compresseurs) et dans le cadre d'une production à la demande (configurateur de gamme pour les petites unités et personnalisation pour les grosses unités),
- gérer les interactions entre les prises de **décisions centralisées** par l'ERP et les prises de **décisions locales** par les opérateurs, qui, dans le cadre de la DFT, possèdent les compétences pour plusieurs postes de travail adjacents de sorte à pouvoir optimiser leur activité en fonction de la situation courante de production.

Le besoin en R&D exprimé par la Trane portait sur le développement d'un outil de **benchmarking** basé sur un émulateur du système de production de la Trane pour évaluer, sur des durées de production crédibles, différentes architectures centralisées/distribuées reposant sur le concept de produit actif et la technologie RFID.

[24] Chaque atelier Trane est composé de lignes principales et de lignes secondaires pouvant être localisées sur des sites différents qui alimentent les lignes principales par des produits semi-finis.

### *4.2 Contribution*

Notre principale contribution [R7][C21] porte donc sur la mise en œuvre d'une démarche couplant :

- la méthode « **design for six-sigma** » (DFSS) afin d'identifier les besoins et attentes de l'entreprise vis-à-vis du pilotage centralisé/distribué par le produit et de proposer un certain nombre d'indicateurs permettant de quantifier l'apport de ces nouvelles architectures ; le choix de cette approche a été imposé par l'entreprise qui l'utilise comme méthodologie standard pour l'amélioration continue de ses processus et le développement de nouveaux projets,
- un environnement de **simulation événementielle distribuée** afin de mesurer en quoi et quand les informations portées par le produit peuvent être pertinentes pour décentraliser le pilotage et répondre aux besoins identifiés ; afin de permettre la validation de différentes stratégies de pilotage sur la base d'un même processus physique, **l'émulation** des entités physiques, représentant les flux et unités de production géographiquement réparties, est dissociée de la **simulation** des entités logiques représentant les prises de décisions par les systèmes de pilotages.

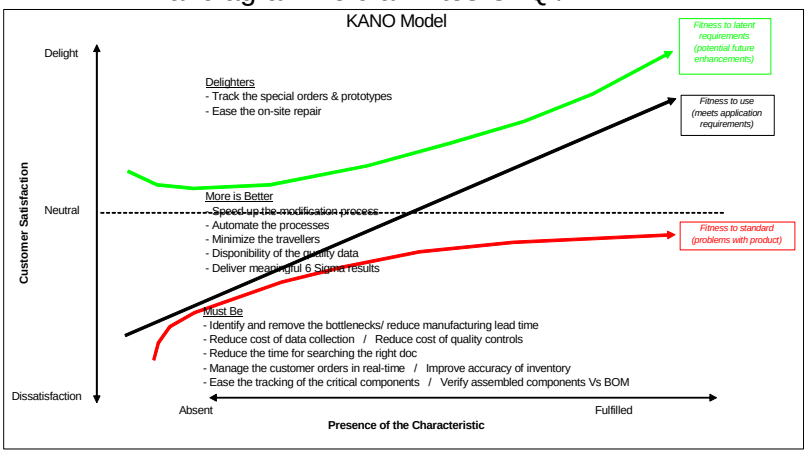
A l'origine limitée aux techniques de MSP (Maîtrise Statistique des Procédés), l'approche six sigma est généralement considérée comme une méthodologie permettant d'améliorer a posteriori un produit ou une prestation, en mesurant et en analysant les dysfonctionnements. **Design For Six Sigma** adapte les concepts généraux de la méthode à la conception ou à la reconception de produits ou processus existants. Son apport en conception réside dans la traduction des attentes et besoins des clients en caractéristiques mesurables représentant le leur degré de satisfaction ("**critical to quality**" ou **CTQ**) qui permettent de choisir et de fiabiliser les produits, prestations et processus répondant au mieux aux attentes. DFSS préconise 5 phases principales relatives à la définition des exigences des utilisateurs (*define*), la définition de caractéristiques quantifiables et mesurables traduisant le respect de ces exigences (*measure*), l'analyse des solutions envisageables (*analyse*), la conception du produit ou processus (*design*) et enfin la vérification des résultats de conception vis-à-vis des indicateurs spécifiés (*verify*).
Si cette approche s'avère très utile pour structurer notre étude et pour définir une stratégie globale de déploiement de la technologie RFID dans l'entreprise, sa principale difficulté réside dans la capacité à obtenir et disposer d'informations quantitatives fiables permettant de mesurer l'adéquation des produits et processus aux attentes des utilisateurs et, en particulier, aux CTQ (Kwak & Antbari 2006). Dans le cadre de produits ou processus n'existant pas, cette difficulté est renforcée par le fait que l'on ne dispose pas dans ce cas de données statistiques. Ce constat nous a conduit à proposer la simulation événementielle comme outil de mesure des indicateurs CTQ.

La première phase de l'étude (*Define*) a donc consisté à identifier l'ensemble des besoins de l'entreprise vis-à-vis de la technologie RFID et de l'approche de pilotage par le produit. Cette approche, basé sur une modélisation des processus de l'entreprise, non restreints à la production, a permis de dégager les processus clefs pouvant espérer tirer parti de l'introduction de la technologie RFID sur les produits. Sur cette base, le travail a alors consisté à prioriser les besoins et définir des

indicateurs pertinents et quantifiables permettant de mesurer le degré de satisfaction des utilisateurs vis-à-vis des différents besoins identifiés (*Measure*). Les résultats de cette phase sont synthétisés par la Figure 47: le principal indicateur retenu comme élément d'évaluation des architectures contrôlés par le produit est le *Lead Time* (temps passé par un produit sur les lignes de fabrication et d'assemblage). Ce critère dépend en effet fortement des temps d'attente sur les zones de stockage aux points de couplage entre les lignes principales et les lignes d'approvisionnement. Les architectures de pilotage par le produit proposées (*analyse*) autorisent un suivi en temps réel des produits et composants sur les lignes qui devrait permettre d'anticiper les retards éventuels de ces lignes par la modification locale de l'ordonnancement de la production. Elles devraient donc conduire à une réduction du *lead time* sans augmentation notable des stocks aux points de découplage.

| PROCESS | IMPROVEMENTS / PROFITABILITY | DIGITIZATION CONTRIBUTION | RFID or BARCODE CONTRIBUTIONS | CTQ | Grade |
|---|---|---|---|---|---|
| ENGINEERING | Identify and **remove the bottlenecks** | Analysis of the real lead times | Detect and/or store automatically the inputs and the outputs at each manufacturing step | **Lead times** | 3 |
| ENGINEERING | Speed up the modification process | Disponibility of data in real-time | (MES contribution) | ex: non-updated permanent drawings rate | 2 |
| ENGINEERING | Automate the processes | Send informations to the equipments | (MES contribution) | Labor time & non-conformities costs due to wrong setting | 2 |
| ENGINEERING | Track the special orders & prototypes | Have a quick access to the location data | Detect and/or store automatically the inputs and the outputs at each manufacturing step | TBD | 1 |
| MANUFACTURING | Reduce cost of data collection | Use hand-held devices & readers | Speed up information recording | non-added value time | 3 |
| MANUFACTURING | Reduce cost of quality controls | Digitize the controls | Reduce the data capture error | quality control time | 3 |
| MANUFACTURING | Reduce the time for searching the right doc | Display automatically the doc | Identify automatically the products | Doc display time | 3 |
| QUALITY | Ease the tracking of the critical components | Have a digitized Log Sheet | Provide an identification at each Quality check | Log Sheet filling time | 3 |
| QUALITY | Verify assembled components Vs BOM | Get as-built BOM for each product | Automatically identification of the components at each manufacturing step | Assembly errors cost | 3 |
| QUALITY | Disponibility of the quality data | Have a centralized database | (MES contribution) | Decentralized metrics quantity | 2 |
| QUALITY | Deliver meaningful 6 Sigma results | Make the data analysis easier / Reliability of the recorded data | Reduce the data capture error | TBD | 2 |
| QUALITY | Ease the on-site repair | Have distributed information | Store information into the products and the components | TBD | 1 |
| SUPPLY CHAIN | Manage the customer orders in real-time | Have a quick access to the location data | Automatically identification of the products at each manufacturing step | Unit tracking time | 3 |
| SUPPLY CHAIN | Improve accuracy of inventory | Have the right consumption in real-time | Automatically identification of the components at each manufacturing step | Inventory accuracy rate | 3 |
| SUPPLY CHAIN | Minimize the travellers | Display the documentation on line | (MES contribution) | Travellers preparation time / Sheets quantity | 2 |

a. diagramme d'affinités CTQ / RFID




b. diagramme KANO

Figure 47. Principales exigences et indicateurs issus de l'étude six sigma

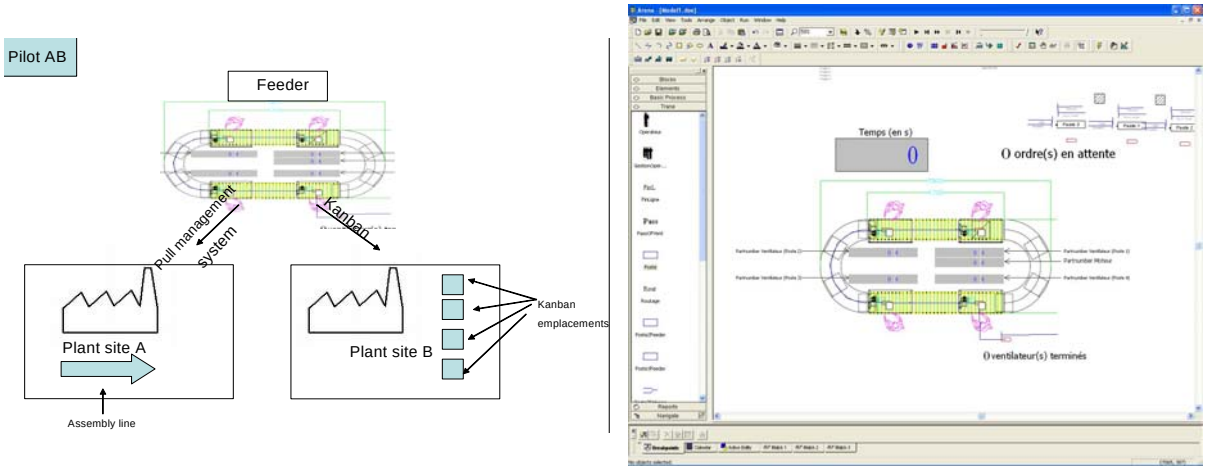
Les phases suivantes de conception et de vérification (*design* et *verify*) reposent sur un environnement de simulation événementielle distribuée nous permettant de valider quantitativement cette hypothèse. Notre postulat de départ est basé sur le fait que la simulation de décisions logistiques dans un contexte distribué est fondée sur des approches multi-modèles permettant de dissocier la modélisation de systèmes

logistiques intégrant des unités de production géographiquement réparties, de la modélisation de leurs systèmes de pilotage. L'intérêt cette séparation entre modèles du système physique de production et modèles de pilotage est de faciliter le test de différentes stratégies de pilotage. Dans le cadre de nos applications basées sur le concept de système contrôlé par le produit, nous avons proposé de systématiser le processus de construction des modèles de simulation par instanciation d'un ensemble d'entités de base :

- des **entités physiques** modélisées sous la forme d'un réseau de files d'attente, caractérisant les processus de transformations dans l'espace (systèmes de transport), le temps (unités de stockage) et la forme (machines de fabrication ou d'assemblage) des flux de produits ; les attributs de ces entités concernent le délai de fonctionnement, la disponibilité des ressources, leurs capacités, etc,
- des **entités logiques** modélisées sous la forme d'heuristiques ou d'algorithmes représentant les décisions centralisées de pilotage fournies par les ERP ou les ordonnanceurs ; ces entités logiques permettent la création/lancement des produits ainsi que la réservation des entités physiques dans le réseau de files d'attente,
- des **entités logiques/physiques** proches du concept d'holon (Valckenaers 2001) représentant des objets en capacité de modifier le flux physique et de prendre des décisions locales; ces objets représentent, dans notre application, les produits dotés de capacités décisionnelles ainsi que les opérateurs.

Le résultat de ce processus de modélisation aboutit à un ensemble de modèles hétérogènes (réseaux de files d'attente, modèles logiques basés sur des heuristiques). Leur **synchronisation** repose sur une exécution alternative des modèles, ce qui revient à stopper l'écoulement du temps dans le réseau de files d'attente pendant la durée d'exécution des modèles logiques. Ces mécanismes sont supportés par les entités physiques/logiques et donnent naissance à des échanges de messages entre **décisions locales** et **décisions centralisées** qui représentent le contrôle par le produit sur le pilotage de la production.

Dans le cadre du simulateur développé à la Trane, le processus pilote retenu est composé d'une chaîne de montage de ventilateurs avec 4 postes de travail qui alimente deux chaînes principales de montage situées dans deux usines distinctes du groupe. La simulation des processus physiques est réalisée sur l'outil ARENA, les modèles logiques sont codés en VBA et appelés par les entités ARENA.




a. pilote Trane b. simulation Arena

Figure 48. Environnement de simulation du pilote Trane

Cet environnement de simulation nous a permis de tester différentes configurations et stratégies de contrôle par le produit. A titre d'exemple, la figure présente les résultats de simulation relatifs à deux emplacements différents (hypothèse H1 et H2) où sont autorisés les échanges d'information entre produit et système de pilotage et leur impact sur le temps de séjour des produits sur la chaîne d'assemblage (lead time).

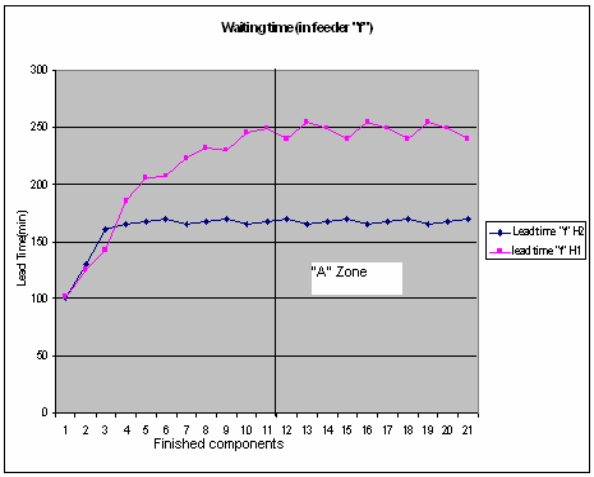



Figure 49. Extrait de l'analyse des résultats de simulation sur le pilote Trane

Si ces premiers résultats de simulation s'avèrent encourageants tant sur le plan des outils d'ingénierie permettant de construire un système contrôlé par le produit répondant au mieux aux attentes des utilisateurs, que sur la pertinence des architectures contrôlés par le produit, une formalisation de notre approche de R&D industrielle demeure indispensable. Cette formalisation pourrait généraliser les travaux de Hind mais aussi de R. Pannequin (Pannequin 2007), basés sur des plates-formes de simulation ad-hoc, afin d'identifier les architectures de simulation ainsi que les mécanismes de synchronisation génériques entre modèles hétérogènes de simulation (modèles événementiels, modèles multi-agents, modèles logiques basés sur des heuristiques, …) à l'aide de techniques couplant HLA et les modèles DEVS (Pujo *et al.* 2006).

## 5. DISCUSSION

Dans le cadre des différents travaux de R&D menés en collaboration avec l'industrie, la vérification et/ou la validation des exigences a bien souvent été obtenue par l'intermédiaire de la simulation. Si cette technique s'avère efficace pour détecter certaines erreurs de conception et/ou d'interprétation dans le traitement des exigences des utilisateurs, elle présente néanmoins certains risques liés au **degré de formalisation de la sémantique** des modèles servant de support à la simulation.

En effet, l'utilisation de modèles des Systèmes à Événements Discrets (SED) pour décrire le comportement de Systèmes Réactifs (Harel & Pnueli 1985), caractérisés par de fortes contraintes temporelles et de sûreté de fonctionnement, amène à s'interroger sur la capacité de ces modèles à supporter simultanément les propriétés de **réactivité** et de **déterminisme** permettant de satisfaire au moins partiellement ces contraintes.

Nous qualifions de **réactivité** la capacité d'un système, en interaction permanente avec son environnement, à traiter tous les événements lorsqu'ils se produisent, ou autrement dit à être « **synchrone** » par rapport aux événements produits par son environnement. Le terme synchronisme utilisé ici dans le sens de la réactivité ne sera pas à confondre avec une autre définition associée au synchronisme que nous utiliserons pour qualifier les évolutions internes au système et désignant la *simultanéité d'occurrence d'événements se produisant à un même instant relatif à une même échelle de temps*. Ce synchronisme d'évolution se traduit souvent par une horloge commune utilisée pour synchroniser les évolutions de processus parallèles.
Deux processus ainsi synchronisés, lorsqu'ils doivent s'échanger des variables, introduisent alors implicitement un retard non nul dans leur communication. Ce type de **synchronisme interne** peut être qualifié de *synchronisme mou* au regard d'hypothèses plus fortes telles que celles qu'introduit le *synchronisme fort* à durée de traitement nulle (Benveniste & Berry 1991). De la même manière, on peut parler d'*asynchronisme externe* pour qualifier les relations entre un système et son environnement, et d'*asynchronisme interne* lorsqu'il s'agit de caractériser des exécutions non synchronisées de processus parallèles. Cette distinction entre *externe* et *interne* est particulièrement nécessaire si l'on considère l'ingénierie des systèmes réactifs pour lesquelles la contrainte de **synchronisme externe** (réactivité) doit être respectée (la réaction du système doit se faire à temps nul en réponse aux événements produits par l'environnement, ou, autrement dit, en imposant au système de se synchroniser avec son environnement), leurs caractéristiques internes pouvant être indifféremment, au moins en théorie, totalement synchrones, partiellement synchrones ou partiellement asynchrones (Benveniste *et al.* 1999).

Les **sémantiques opérationnelles** associées aux modèles de S.E.D., tels que les Statecharts, le Grafcet ou encore le Langage Synchrone Signal servent de base à leur exécution symbolique en simulation. Cependant, en fonction du degré de formalisation de ces sémantiques et des mécanismes d'interprétation des modèles, certaines formes d'ingénierie peuvent ne pas être toujours possibles sans conditions. Examinons en particulier le cas du **raffinement** et de la composition par **assemblage de modules**. Nous définissons le **raffinement** comme une technique permettant de compléter progressivement un modèle de système « de l'intérieur », c'est à dire en restant dans les mêmes frontières d'isolement du système (et de ses définitions théoriques). Cette technique impose alors que les relations que peut entretenir un modèle partiel d'un système, à un instant donné de l'ingénierie, soient spécifiées comme restant internes au modèle ou comme faisant partie de l'interface avec l'environnement du système, surtout lorsque ses caractéristiques internes et externes sont différentes. Nous étudierons aussi la capacité des modèles à admettre une ingénierie par **composition de modules** pour laquelle la composition des propriétés internes et externes de chaque composant doit permettre de garantir, en toute cohérence avec les définitions théoriques du modèle utilisé, tant les propriétés externes qu'internes (Figure 50).

La seconde propriété importante des systèmes réactifs est le **déterminisme** de leur comportement. Là aussi, le terme *déterminisme* est une expression qui admet plusieurs définitions selon qu'il s'agit de décrire les caractéristiques internes ou externes d'un modèle de système réactif. Par exemple, on parlera de **déterminisme d'évolution** (déterminisme interne, ou encore, opérationnel) si, pour un modèle donné, *à un instant quelconque du temps, il n'y a, pour chacun des instants*

*antérieurs ou ultérieurs, qu'un état et un seul qui soit compatible avec le premier*. C'est en particulier cette propriété interne qui fait distinguer le déterminisme ou non d'un automate à états finis [HOP 79]. Le **déterminisme externe** (ou encore, déterminisme causal) d'un modèle sera quant à lui défini par sa capacité à produire, *pour une même variation de ses événements d'entrée, une même variation de ses événements de sortie*.

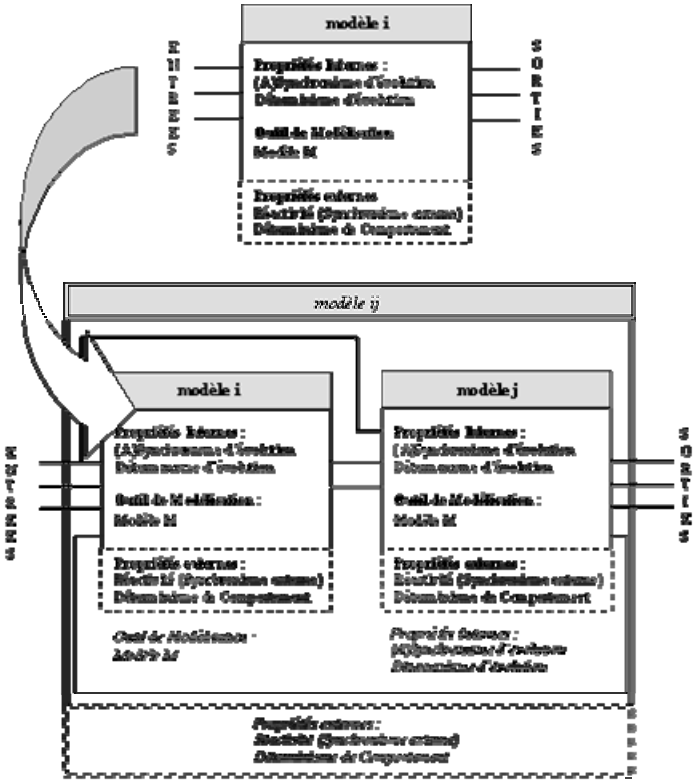



Figure 50. Composition par assemblage de modèles

Selon les modèles de S.E.D. utilisés et les sémantiques opérationnelles associées, la propriété de déterminisme présente un certain de nombre de dépendances avec les propriétés internes et/ou externes de synchronisme, d'asynchronisme ou de leurs combinaisons.

### 5.1.1 Propriétés externes de déterminisme / réactivité

Le Tableau 4 recense tout d'abord les propriétés externes de Réactivité et de Déterminisme des modèles Statecharts, Grafect et du langage synchrone Signal en mettant en évidence leurs définitions sémantiques qui permettent de respecter séparément les contraintes de **réactivité** ou de **déterminisme**, ou les deux simultanément.
Relativement à ces propriétés, les solutions sémantiques des modèles **Statechart** et **Grafcet** sont proches, même si leurs définitions ne font jamais référence les unes aux autres. En fait, ce qui distingue ces modèles sont plus des nuances syntaxiques que réellement sémantiques. Ainsi, les mécanismes de gestion des sous-états dans des super-états en Statechart sont très proches de ceux mis en œuvre en Grafcet par le forçage dans une hiérarchie de grafcets. D'autre part, les nouvelles propositions normatives du Grafcet définissant l'**Hyper-Grafcet** comme une extension du modèle Grafcet s'inspirant fortement des mécanismes liés à la

**hiérachie d'état** des Statecharts, montre un exemple de rapprochement syntaxique possible entre les deux formalismes tout en notant la proximité de leurs définitions sémantiques. En effet, l'algorithme **Sans Recherche de Stabilité** du Grafcet, à rapprocher de l'**interprétation par STEP** des Statecharts, privilégie la réactivité sans garantie de détermisme du comportement. L'algorithme **Avec Recherche de Stabilité** du Grafcet, proche de l'**interprétation par Micro-Steps** des Statecharts, privilégie quant à lui le **déterminisme de comportement**, sans garantir la réactivité.

| | **Sémantique « Externe »** | | |
|---|---|---|---|
| | ***Réactivité*** ***Risque de non-déterminisme*** | ***Déterminisme*** ***Risque de non-réactivité*** | ***Réactivité et*** ***Déterminisme*** |
| **STATECHARTS** | **Interprétation par STEP** [HAR 90] | **Interprétation par MICRO-STEP** [HAR 87] | **Consistance Globale d'un STEP : Sémantique déclarative** [PNU 91] **Sémantique opérationnelle** [PNU 91][GUE 97] |
| **GRAFCET** | **Algorithme SRS** (Sans Recherche de Stabilité) [AFC 83] [GRE 85] | **Algorithme ARS** (Avec Recherche de Stabilité) [AFC 83] [GRE 85] | **Double échelle de temps** [UTE 93][LHO 97] **Joueur Grafcet** [LHO 97][LES 98] **Automate équivalent** [ROU 94] |
| **SIGNAL** | Afin de vérifier que la spécification est conforme aux hypothèses fondant le langage Signal (réactivité et déterminisme), une recherche de **solution pour le système d'équations** correspondant aux primitives du langage utilisées pour décrire la spécification est réalisée. Si une solution est trouvée, la spécification est conforme, sinon, elle est rejetée [LEG 86]. | | |

Tableau 4. Propriétés externes des modèles

Les sémantiques des modèles **Grafcet** ou **Statecharts** peuvent être indifféremment traduites en langage **Signal** sous la forme de trois ensembles d'équations qui permettent l'évaluation du franchissement des transitions, l'activation ou la désactivation des états, ainsi que l'exécution des actions. Dans le cas d'un algorithme **sans recherche de stabilité**, le franchissement des transitions sera évalué par rapport aux stimuli (internes ou externes) retardés d'un temps d'horloge, ce qui revient à définir une horloge globale permettant la synchronisation de tous les signaux. Dans le cas de l'algorithme **avec recherche de stabilité**, il convient d'évaluer le franchissement des transitions par rapport à des stimuli internes retardés d'un temps d'horloge et des stimuli externes qui ne doivent être présents que lorsqu'une **situation stable** est atteinte. Cela revient à définir l'horloge des signaux externes comme un **sous-échantillonnage** (selon le critère de stabilité) de l'horloge associée en interne aux signaux caractérisant les franchissements de transitions et les changements d'états. Dans tous les cas, la recherche d'une solution de la spécification Signal permet soit de garantir le respect de la propriété de **réactivité** en cas de succès, soit de rejeter ces spécifications dans le cas contraire.

### 5.1.2 Propriétés internes de synchronisme / asynchronisme

Le Tableau 5 présente enfin les interprétations internes synchrones et/ou asynchrones que mettent en œuvre chacun des modèles pour satisfaire plus ou moins partiellement les propriétés externes de Réactivité et de Déterminisme ainsi que la sémantique opérationnelle permettant leur mise en œuvre.

<table>
<tr><th></th><th colspan="3">Sémantique « Interne »</th></tr>
<tr><th></th><th>Synchronisme Interne</th><th>Asynchronisme Interne</th><th>Mise en œuvre</th></tr>
<tr><td rowspan="4">STATECHARTS</td><td colspan="2">EN THEORIE : SYNCHRONISME PARFAIT<br>(évolutions et diffusion à durée nulle)</td><td rowspan="2"></td></tr>
<tr><td colspan="2">EN PRATIQUE</td></tr>
<tr><td>PARTIEL :<br>L'interprétation par Step et/ou Micro-Step introduit un retard dans la propagation des événements internes, ou encore, les états parallèles évoluent à une même date mais avec une durée petite mais non nulle.</td><td>PARTIEL :<br>Utilisation des dépendances entre transitions et entre états dans la hiérarchie d'états [PNU 91]</td><td>Algorithmique<br>(en ligne)[GUE 96]</td></tr>
<tr><td colspan="3">LIMITE : Causalité Circulaire</td></tr>
<tr><td rowspan="2">GRAFCET</td><td>TOTAL :<br>Joueur Grafcet à 3 échelles de temps (recherche de stabilité dans les évolutions structurelles et par forçage) [BIE 96][BIE 97] [LES 98][DUM 99]</td><td>PARTIEL :<br>Utilisation de l'ordre de Hiérarchie de Forçage dans le Joueur Grafcet [GIA 91][LES 92] [LHO 97]</td><td>Algorithmique<br>Joueur Grafcet (en ligne)<br>OU<br>Recherche (hors ligne) des situations non conformes dans l'Automate Équivalent [ROU 94] [ROU 96]</td></tr>
<tr><td colspan="3">LIMITE : Situations stationnaires<br>SAUF dans l'approche par extraction de l'automate équivalent [ROU 94][ROU 96]</td></tr>
<tr><td rowspan="4">SIGNAL</td><td colspan="2">EN THEORIE : SYNCHRONISME PARFAIT<br>(évolutions et diffusion à durée nulle)</td><td rowspan="2"></td></tr>
<tr><td colspan="2">EN PRATIQUE</td></tr>
<tr><td></td><td>TOTAL :<br>Traitements exécutés en utilisant les dépendances de variables déterminées par résolution préalable d'équations dans Z/3Z et analyse des graphes de dépendance</td><td>Recherche (hors ligne) de conformité de la spécification</td></tr>
<tr><td colspan="3">LIMITE : Causalité circulaire</td></tr>
</table>

Tableau 5. Propriétés internes des modèles

La sémantique des Statecharts est basée sur une **hypothèse théorique** de **synchronisme parfait**, c'est à dire d'une simultanéité entre stimuli et réactions et de communications à temps nul. Cette définition, faisant apparaître explicitement le terme « synchrone », est cependant confrontée à des difficultés de mise en œuvre

pratique. Ainsi, les interprétations pratiques par **Step** et/ou **Micro-step** permettent de respecter partiellement cette hypothèse, uniquement dans le cas d'une construction de la spécification par **raffinement**. Dans ce cas, et à l'image de ce qui était préconisé pour le **Grafcet** en termes d'**algorithmes SRS et ARS**, cette propriété de synchronisme parfait est alors « repoussée » vers l'extérieur du modèle (revenant alors à une propriété de réactivité), c'est à dire que le seul synchronisme subsistant en interne est celui d'une **exécution synchrone** (sur la base d'une même horloge) des comportements parallèles induisant un **retard** dans les communications entre ces comportements. Afin de rendre cohérentes les sémantiques externes et internes et donc pour autoriser une construction de la spécification par **assemblage modulaire**, la sémantique proposée par (Pnueli 1991) cherche en fait à déterminer la **relation d'ordre** existant entre les diverses variables caractérisant la spécification afin de l'utiliser pour **ordonnancer les évolutions**. A ce titre, cette sémantique conduit à un **asynchronisme interne** du traitement des évolutions au bénéfice d'un synchronisme parfait apparent, tant interne qu'externe. Cette démarche, dont la mise en pratique « en ligne » a été présentée par (Gueguen 1996), est à relier à celle « hors ligne » retenue par le **langage Signal** qui préconise le calcul des dépendances de variables permettant d'ordonner, quand une relation d'ordre stricte existe, le traitement des évolutions.
Les promoteurs du modèle **Grafcet** ont eu une tout autre démarche. Partant d'une séparation artificielle entre les définitions sémantiques théoriques du modèle et leur mise en pratique par des algorithmes d'interprétation SRS et/ou ARS, la volonté de garantir au modèle Grafcet, quelles que soient les modalités de sa mise en œuvre, des propriétés externes de réactivité et déterminisme en adéquation avec les hypothèses de synchronisme (interne) d'évolution, a conduit à la définition de **2 échelles de temps** « sans commune mesure ». Le premier « Joueur Grafcet » (Lhoste *et al.* 1997), précisant les modalités d'application de ces nouveaux postulats temporels, préconise alors une évolution structurelle synchrone nécessitant une boucle de recherche de stabilité pour s'affranchir d'une quelconque recherche de dépendance entre les évolutions et une évolution par forçage asynchrone, utilisant l'ordre connu de la hiérarchie de forçage pour s'affranchir d'une boucle de recherche de stabilité pour ces évolutions.

Notons enfin que les **limites sémantiques** des modèles étudiés sont communes. En effet, l'existence possible d'une **causalité circulaire** entre les variables (Signal), les transitions (Statecharts) ou les états (situations stationnaires en Grafcet), met en défaut la propriété recherchée de réactivité.

Pour conclure, l'analyse de la sémantique de trois modèles de S.E.D. a permis de souligner l'importance de préciser le contexte d'exécution de la simulation dans la mesure où des sémantiques et/ou interprétations différentes conduiront à des résultats quelques fois radicalement différents. Ces difficultés justifient notamment le travail présenté au chapitre suivant et portant sur la modélisation et le raffinement de spécifications prouvées.

## 6. CONCLUSION

Dans ce chapitre, nous avons présenté des résultats de R&D industrielles visant à identifier et formaliser un ensemble d'exigences, puis à les allouer sur une architecture cible et enfin, à modéliser le comportement d'un système devant, en

théorie, couvrir ces exigences. Dans le contexte industriel où se situent ces travaux, les principales techniques d'identification et de modélisation des exigences sont fondées sur des modèles semi-formels et des approches méthodologiques permettant leur structuration. Les relations entre ces modèles semi-formels et les modèles de description architecturale et comportementale du système en cours de développement restent le plus souvent non formalisées et validées par simulation. Au mieux, les techniques de méta-modélisation permettent d'aboutir à une représentation des concepts manipulés par chacun des modèles et d'établir ainsi une correspondance entre objets relatifs aux exigences et objets relatifs aux modèles du système.

Il nous est donc apparu important de compléter ces approches de R&D industrielles par un **processus formel d'ingénierie dirigée par les modèles** utilisant le raffinement et la preuve de spécification comme fondements d'un processus sûr d'automatisation.

# IV Cadre formel de spécification à l'aide du langage B

## 1. INTRODUCTION

Dans le contexte fixé par le prédicat P2 (*Système de commande* $\curlywedge$ *Système opérant* $\supset$ *Exigences Système*), ce chapitre présente les travaux portant sur la définition d'un cadre formel de spécification des systèmes automatisés de production. Il constitue ainsi une rationalisation des résultats obtenus dans le chapitre précédent en s'appuyant sur le **langage B** et le mécanisme formel de raffinement des modèles qu'elle propose.

Cette activité s'appuie sur un stage post-doctoral réalisé au LORIA dans l'équipe MODEL du Pr Méry dans le cadre d'une collaboration industrielle (contrat EDF/DER/3C Chatou – LORIA – CRAN n°ARD P3136R, 1998) visant à évaluer l'utilisation d'approches formelles en milieu industriel.

Pour répondre aux besoins de validation et de vérification des systèmes automatisés, les normes en vigueur actuellement telle que l'IEC 61508 recommandent l'utilisation de méthodes formelles lorsque ces systèmes sont soumis à de fortes contraintes de sécurité et de sûreté de fonctionnement leur conférant un niveau SIL 4 (*Safety Integrity Level*). Cependant, malgré le consensus sur le fait que les premières phases de spécification sont les plus importantes pour garantir le développement d'une solution conforme aux exigences, de nombreux travaux se concentrent sur la vérification formelle de modèles de conception (Clarke *et al.* 2000, Cassandras & Lafortune 1999).

En ce sens, notre principale contribution vise à faciliter, au plus tôt dans le cadre d'une ingénierie système, une **représentation commune et consensuelle** des **services** attendus d'un système automatisé par les différents acteurs du procédé d'automatisation **[R4][TH1]**. Le mécanisme formel de raffinement supporté par la méthode B nous permet de garantir la **cohérence** globale d'une spécification construite à partir d'une représentation abstraite des exigences fonctionnelles que l'on contraint progressivement pour tenir compte de l'architecture du système à concevoir, de la répartition des exigences sur les composants et des contraintes techniques de développement tout en préservant les relations logiques établies par le prédicat P2. Les mécanismes formels de preuves permettent d'établir la **correction** intrinsèque des spécifications vis à vis de contraintes et connaissances expertes du domaine concerné formalisées sous la forme de patrons de conception ou de profils UML **[R3]**.

## 2. RAFFINEMENT FORMEL DE SPECIFICATION

### *2.1 Problème*

L'intégration d'une part de plus en plus importante de technologies de l'information et de la communication au cœur même des processus de production et des produits

conduit naturellement à une augmentation de la complexité des systèmes automatisés avec pour conséquence sur le procédé de modélisation, la nécessité de maîtriser les **interactions** entre les différentes représentations nécessaires pour couvrir la dimension comportementale d'une automatisation (processus physiques et logiques) mais également ses composantes architecturales et informationnelles. Ces représentations relèvent de disciplines bien identifiées (automatique, génie informatique, physique, modélisation des exigences, …) et sont donc souvent dédiées à la conception dans un domaine d'application donné.

Dans ce contexte et même si les modèles supportant ces différentes représentations « métiers » offrent, selon leur degré de formalisation, différents moyens de vérification des **propriétés intrinsèques** au domaine de modélisation couvert (simulation, preuves formelles), la **cohérence entre les modèles** – et donc la vérification des propriétés ou des exigences « système », reste délicate dans le cadre de ces formalismes. En effet, la vérification de cohérence entre modèles est souvent limitée à des recommandations d'ordre méthodologique basées sur la définition de cadres descriptifs, tels que GERAM ou encore Zachmann, qui permettent d'appréhender la modélisation d'un système comme un ensemble cohérent de modèles hétérogènes dont les relations sont préconisées par le cadre d'ingénierie considéré.

Ce constat souligne l'importance des premières phases de **spécification** pour garantir le développement d'une solution complexe – interdisciplinaire et résultant de la coopération entre de multiples composants – conforme aux exigences des utilisateurs. Cette problématique relève du champ disciplinaire de **l'ingénierie « système »** visant à maîtriser les processus techniques d'ingénierie des systèmes, les processus d'intégration d'un système à partir de ses composants existants ou développés spécifiquement et à maîtriser les relations contractuelles entre les différents acteurs d'un projet.

En ce sens, le développement du langage **SysML**[25], défini comme un profil d'UML 2.0., enrichit les formalismes de spécification présents dans UML pour y intégrer des éléments conceptuels correspondant aux besoins de l'ingénierie des systèmes : formalisation des exigences, décomposition, vision fonctionnelle, cycle d'intégration/vérification/validation/qualification. A l'instar d'UML, le langage SysML peut donc être considéré comme une boîte à outil proposant un ensemble de modèles couvrant la spécification de système mais il n'intègre aucun aspect méthodologique. Les choix de types de modèle à utiliser aux différentes étapes du processus d'ingénierie système restent notamment ouverts. D'autre part, le manque de formalisation de SysML ou d'UML, malgré les efforts faits en ce sens notamment par la méta-modélisation ou encore en couplant la modélisation UML et les réseaux de Petri (Bouabana-Tebibel & Belmesk 2007), reste un obstacle pour le passage de diagrammes SysML vers des outils de modélisation formelle, quantitative et probabiliste permettant la simulation ou la preuve de propriétés.

[25] SysML customizes the UML™, the industry standard for modeling software-intensive systems, for systems engineering applications. It supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems. These systems may include hardware, software, information, processes, personnel, and facilities. *Source OMG, http://www.sysml.org/*

C'est pourquoi, un certain nombre de praticiens en ingénierie système ou en automatisation (Shell 2001, Johnson 2004) considère que le passage vers une véritable **spécification formelle** est nécessaire :

- pour établir un **modèle complet, pertinent, cohérent et correct** de ce que le système doit faire et partager une compréhension commune et cohérente des services attendus d'un système automatisé,
- **maîtriser la complexité** des architectures des systèmes automatisés (propriétés comportementales et structurelles relatives aux architectures, aux composants et leurs interfaces)
- **retarder** l'utilisation des langages ou modèles orientés métier nécessaires pour supporter toute activité de conception.

Une telle spécification formelle s'inscrit naturellement dans les premières phases du cycle classique de Validation/Vérification (Figure 51) et doit reposer sur **un langage formel unifié** dont le niveau d'abstraction permet de satisfaire un large éventail de besoins de modélisation : comportement, structure, information, communication, etc.

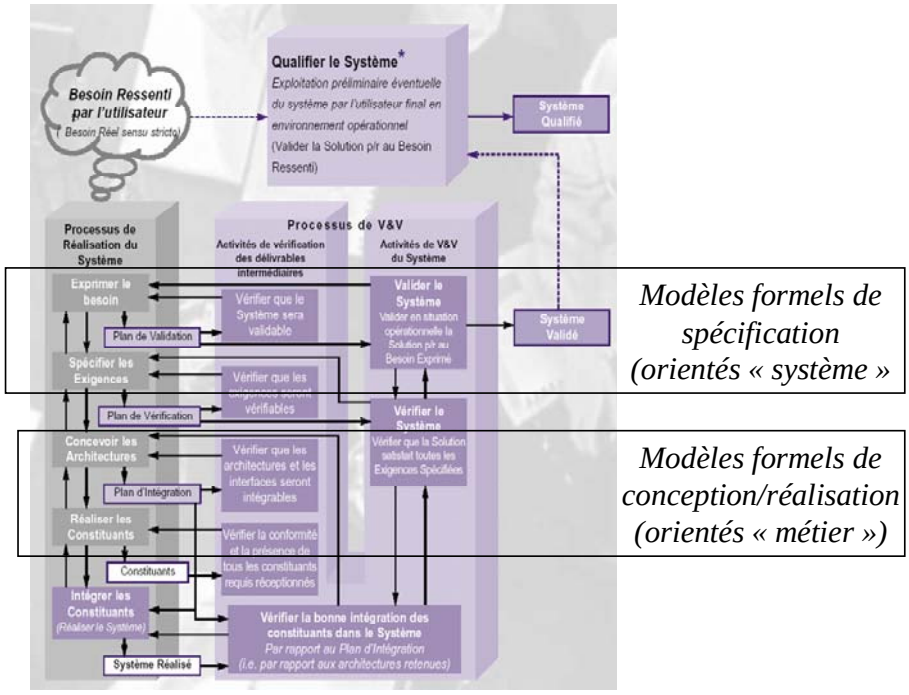



Figure 51. Spécification formelle dans un cycle V&V

D'autre part, si l'on tient compte des aspects cognitifs de l'activité de spécification permettant à chaque acteur d'un projet d'automatisation de transformer "*une intention de résultat, associée à la perception d'un besoin ou d'un désir à satisfaire, … en une attente de réalisation transmise à un tiers"* (Lhote *et al.* 1999), il semble naturel d'aborder la spécification de manière progressive suivant une approche collaborative et interdisciplinaire. Le langage formel unifié, support à la spécification formelle, doit donc autoriser une **description incrémentale des modèles** tout en garantissant, dans le cadre d'une ingénierie dirigée par les modèles, la cohérence entre des spécifications établies à des niveaux différents d'abstraction.

Dans le paragraphe suivant, nous montrons, après une brève présentation du langage B, en quoi cet outil de modélisation offre les mécanismes de **vérification** et de **raffinement** recherchés.

## *2.2 Le langage B*

Conçu par J.-R. Abrial (1996), le langage B est dédié à la spécification, la conception et l'implantation de logiciel et utilise la preuve de propriétés à la fois comme mécanisme de vérification des propriétés mais aussi comme mécanisme de construction des modèles. On parle alors de *proof-oriented modelling*.

Le langage B est principalement basé sur :
- la théorie des ensembles et les fonctions, relations et opérateurs associés,
- la logique du premier ordre avec les opérateurs classiques ($\neg$ P, P$\vee$ Q, P $\wedge$Q, P $\Rightarrow$ Q, P $\Leftrightarrow$ Q) ($\forall$X • p, $\exists$ X • p),
- la notation en machine abstraite.

Une machine abstraite B est définie par la structure de la Figure 52. La première partie de la machine décrit les objets et informations du modèle :
- une machine a un nom,
- la clause SETS contient la définition des ensembles relatifs au problème,
- la clause CONSTANTS contient la définition effective des constantes (un élément d'un ensemble, une fonction, …)
- la clause PROPERTIES permet la définition de propriétés relatives aux constantes (telles que la définition d'une fonction par exemple),
- la clause VARIABLES contient la définition des variables du modèle

La deuxième partie de la machine définit les aspects dynamiques des variables. Les opérations $O_1 \ldots O_n$ décrivent comment les variables d'états sont modifiées selon le formalisme des substitutions généralisées ([S], [x:= f(y)]), où *Si (x)* représente la nouvelle valeur de la variable *x* après exécution de l'opération *Oi*. La substitution d'une variable est conditionnée par sa **pré-condition** de manière plus ou moins déterministe (la clause PRE n'empêche pas la substitution mais n'offre aucune garantie quant au respect de l'invariant, la clause SELECT doit être vérifiée avant exécution mais n'est pas exclusive, plusieurs branches de l'opération pouvant être exécutées, la clause IF doit être vérifiée mais est totalement déterministe).

**MACHINE** *m*
**SETS** *s*
**CONSTANTS** *c*
**PROPERTIES** *p*
**VARIABLES** *x*
**INVARIANT** *I(x)*
**INITIALISATION** *init(x)*
**OPERATIONS**
*O1= select P1(x) then S1(x)*
*…*
*On = select Pn(x) then Sn(x)*
**END**

Figure 52. Machine B

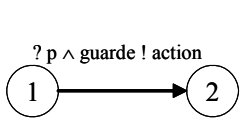
**L'invariant** *I(x)* définit, sous la forme d'un prédicat, les propriétés relatives aux variables qui doivent être préservées lors de l'initialisation et après exécution d'une opération. Les conditions à vérifier pour établir l'invariant, appelées « obligation de preuves », sont générées à partir du texte de la machine et expriment les hypothèses requises pour la préservation des propriétés de l'invariant :

$$\text{(INV1) } Init(x) \Rightarrow I(x)$$
$$\text{(INV2) } I(x) \wedge P(x) \Rightarrow I(S(x))$$

(INV1) exprime le fait que les conditions initiales doivent établir l'invariant et (INV2) le fait que partant d'une situation où l'invariant est respecté et où la pré-condition P de l'opération est établie, la transformation de la variable doit déboucher sur une situation où l'invariant est préservé. (INV2) doit être préservé pour chaque opération Oi de la machine. A noter que l'atelier B[26] propose un prouveur interactif permettant d'aider le modélisateur dans l'obtention de la preuve d'un invariant.

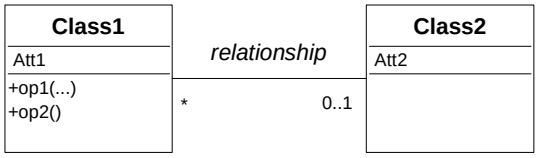
D'autre part, le langage B offre des mécanismes de composition des machines permettant d'encapsuler ou d'importer des variables et/ou des opérations (clauses SEES et USES pour le partage de données dans l'invariant et/ou les pré-conditions, clauses EXTENDS et INCLUDES pour l'invocation d'opérations respectant ou pas l'invariant de la machine possédant l'opération). Ces mécanismes s'avèrent particulièrement efficaces pour structurer la modélisation de systèmes complexes.

Pour conclure cette brève présentation, le langage B a prouvé son efficacité dans le domaine du développement de logiciel en raison de son pouvoir d'expression couvrant la description comportementale mais aussi informationnelle des applications (Figure 53) mais également dans le domaine des systèmes technologiques à logiciel prépondérant notamment dans les applications ferroviaires (Behm *et al.* 1999).




**MACHINE** G1
**SETS** STATE /* set of states*/
**VARIABLES** st1, st2, guarde, p, s
**INVARIANT** st1 ⊂ STATE ∧ st2 ⊂ STATE
**OPERATION**
t(s,p) = select guard ∧
s ∈ st1 ∧ p ∈ T
THEN st1 := st1 – {s}
|| st2 := st2 ∪{s}
|| action
**END**

a. Etat-transition en B (Lano 2000)




**MACHINE** Class 1
**SETS** UML_OBJECTS;
ATT_DOMAIN
**VARIABLES** class1
**INVARIANT**
class1 ⊆ UML_OBJECTS
∧ class_id ∈ class1 ⤖ NAT
∧ att1 ∈ class1 → ATT_DOMAIN
**…**
**OPERATIONS**
Op1= begin … end
Op2 = pre P1(u) THEN u: P2(u)
**END**

**MACHINE** Relationship
**USES Class1, Class2**
**SETS** UML_OBJECTS;
**VARIABLES** relation
**INVARIANT**
relation ⊆ UML_OBJECTS ∧
relation ∈ class1_id → class2_id

b. B & Systèmes d'informations basé sur (Meyer 2001)

Figure 53. B & la modélisation comportementale et informationnelle

Au-delà de ces considérations, le langage B a retenu notre attention pour deux raisons essentielles :

- sa capacité à **spécifier** et à **vérifier** le **fonctionnement global** d'un système (ce que le système doit faire) sans préjuger des solutions de conception qui seront mises en œuvre (le comment).
- les mécanismes de **raffinement** supportés par le langage B.

[26] Atelier B : développé et commercialisé par la société CLEARSY

La spécification formelle à l'aide du langage B permet de décrire les services attendus d'un système sans nécessairement préciser la manière (algorithmes, architecture, …) avec laquelle ses services pourront être rendus. Ce pouvoir d'expression est notamment du à l'utilisation de primitives abstraites non déterministes et à la notion de **post-conditions**. Ces dernières établissent une propriété que doit vérifier le résultat d'une opération sans tenir compte de la manière dont a été obtenu le résultat. En particulier, la primitive

ANY x' WHERE p(x') THEN x :=x'

spécifie les propriétés attendues sur le résultat d'une opération au travers d'un prédicat p(x') et définit le comportement de l'opération comme étant n'importe quelle substitution conduisant à ce résultat.

Le mécanisme de **raffinement** est utilisé pour décrire le modèle à plusieurs niveaux d'abstraction en enrichissant au fur et à mesure la description des variables et des événements tout en préservant les invariants déjà prouvés. On peut résumer cette approche par :

*(M1,G1) raffiné par (M2,G2) raffiné par … (Mn,Gn)*

*Mi* est la i-ème itération du modèle et satisfait l'objectif *Gi* ainsi que les objectifs des itérations précédentes. La relation *raffiné par* assure la préservation des objectifs à la condition qu'ils soient prouvés. Cela signifie que si un nouveau modèle est dérivé de *(Mn, Gn)* il faudra prouver que ce nouveau modèle raffine bien l'ancien modèle tout en garantissant les nouvelles propriétés. Considérons le raffinement de la machine de la Figure 52 selon : *variable y, invariant J(x,y), operation Oi(y)* $\widehat{=}$ *Pre Qi(y) then Ti(y)*, où J représente la relation formelle entre la variable abstraite x et la variable raffinée y et éventuellement des propriétés locales à y (appelé invariant de collage), alors le résultat obtenu par la substitution Ti(y) devra être conforme aux propriétés établies par la substitution Si(x). Ceci peut s'établir en vérifiant

$$I(x) \wedge J(x,y) \wedge Qi(y) \Rightarrow Pi(x) \wedge J(Si(x), Ti(y))$$

ce qui revient à montrer que la substitution Ti qui s'exécute sous sa pré-condition conduit à un état raffiné relié, par l'invariant de collage, à un état abstrait vérifiant I.

## *2.3 Contribution*

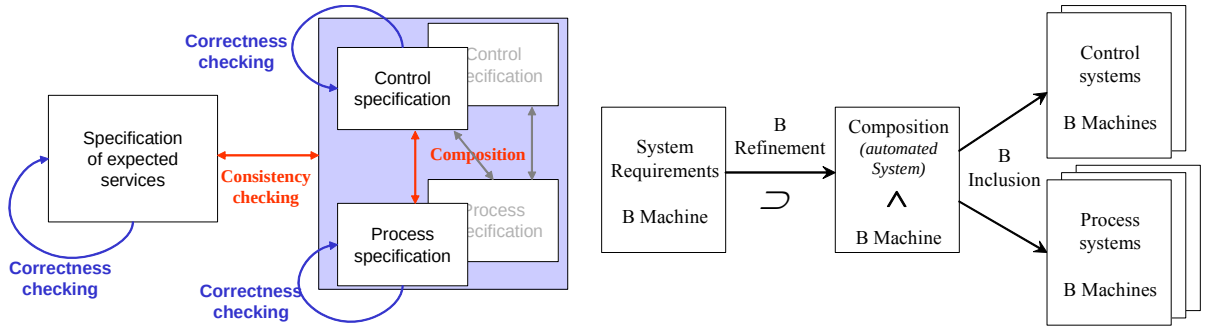
Notre contribution à la spécification formelle de systèmes automatisés à l'aide du langage B porte sur :

- la définition d'un **cadre méthodologique** pour structurer le processus de modélisation en B selon le prédicat P2 (*Système de commande* $\wedge$ *Système opérant* $\supset$ *Exigences Système*) [R4][C6][C8][C9][C10][C12][C23],
- la proposition d'un **joueur B** permettant de prendre en compte, dans le processus de vérification, les interactions de type action/réaction caractérisant les relations entre un système automatisé et son environnement [C20].

### 2.3.1 Cadre méthodologique

Conformément aux relations logiques établies par le prédicat d'automatisation P2, le cadre méthodologique proposé repose sur la modélisation en B des exigences sous

la forme d'une ou plusieurs machines ainsi que sur la modélisation du système automatisé devant répondre à ces exigences, sous la forme d'une composition de machines représentant le système de contrôle et de commande et de machines représentant le procédé à automatiser (Figure 54a).




a. Modèles du prédicat d'automatisation b. Equivalence en B

Figure 54. Le prédicat d'automatisation et son équivalence en B [R4]

Les opérateurs $\wedge$ et $\supset$ présents dans le prédicat d'automatisation sont respectivement représentés en B par Figure 54b:

- les **mécanismes de composition** B permettant de construire une machine « système automatisé » invoquant les opérations logiques de contrôle-commande et physiques de transformation de matière et d'énergie décrites dans des machines spécifiques (*control machines*, *process machines*),
- le **mécanisme de raffinement** permettant d'établir que les opérations et invariants définis dans une machine abstraite décrivant toutes les exigences des utilisateurs (exigences fonctionnelles, comportementales, informationnelles, structurelles, …) sont bien préservées, dans une relation de raffinement, par la machine décrivant le système automatisé.

Les mécanismes de preuves sont donc utilisés pour vérifier, d'une part, les **propriétés locales** de chaque spécification afin d'établir leur correction vis-à-vis de règles syntaxiques ou sémantiques relatives au domaine modélisé, et, d'autre part, pour prouver la relation de raffinement existant entre le modèle des exigences et la spécification d'une solution répondant à ces exigences. Cette dernière preuve permet notamment de vérifier les **propriétés « système »** non réductibles aux propriétés des constituants pris isolément mais émergeant d'un réseau d'interactions entre ces constituants.

L'ensemble des machines B obtenu est alors considéré comme une **ré-écriture cohérente du cahier des charges** située en amont des représentations orientées métiers. Ces spécifications doivent servir de base aux processus de conception des différents sous-systèmes identifiés, qui nécessitent souvent l'utilisation de modèles dédiés (Lecomte 2002) – le langage B peut néanmoins être utilisé en conception lorsqu'il s'agit de composants logiciels. A titre d'exemple, la prise en compte du temps dans le langage B n'étant pas explicite, l'utilisation des modèles de l'automatique continue ou discrète reste indispensable pour la conception des systèmes de commande et la vérification de leur comportement.

### 2.3.2 Joueur B

Dans une machine B, les assertions et les propriétés invariantes doivent être préservées par toutes les opérations du modèle. Ceci signifie que le processus de vérification consiste à établir qu'un prédicat réputé vrai **avant l'exécution de toute opération** le demeure **après exécution** de ladite opération sous réserve que ses pré-conditions soient vérifiées.
Dans le cadre de la modélisation des systèmes réactifs, les propriétés « système » ne s'établissent en général qu'après la prise en compte d'un cycle action/réaction qui caractérise la réaction du système à un stimulus extérieur. Or dans le cas d'un modèle B faisant intervenir des opérations décrivant le comportement de l'environnement (opération caractérisant les stimuli admissibles), des opérations décrivant les actions du système de commande (modèles de commande) et des opérations représentant la réaction du système physique aux actions de commande (modèles de procédé), les propriétés ne pourront être invariantes qu'entre l'instant d'apparition du stimuli déclenchant et l'instant où a lieu la réaction du procédé. En d'autres termes, les propriétés invariantes ne pourront être préservées que par un **cycle d'opérations** et non par une opération unique.

Afin de prendre en compte cette notion de cycle d'exécution des opérations dans le processus de vérification des invariants B, nous avons dû incorporer un moteur permettant de représenter explicitement l'**ordre d'exécution des opérations** dans les différentes machines B [C20]. Il repose sur un système de drapeaux pour lequel la variable *Drapeau* peut prendre les valeurs $D_{ij}$, où *i* correspond à l'ordre d'exécution de la machine, et *j* à l'ordre d'exécution de l'opération dans la machine. La condition *Drapeau* = $D_{ij}$ est insérée dans la garde (SELECT) de l'opération *ij*, la variable *Drapeau* étant actualisée par les substitutions des opérations.

```
OPERATION ij =
    SELECT Drapeau = Dij
    THEN IF Gij THEN Sij || Drapeau := D(i+1)1
            ELSE Drapeau := Fi(j+1)
End;
```

*Où Gij et Fij représentent respectivement la garde et la substitution de l'opération ij*

La prise en compte de ce drapeau dans les invariants (qui deviennent $I(x) \wedge D_{11}$ où $D_{11}$ représente un début de cycle) permet de limiter leur portée aux instants où le cycle complet d'opérations a bien été exécuté. En d'autres termes, cela revient à définir un joueur B donnant alternativement la « parole » aux machines modélisant le système de commande et à celles représentant le comportement du procédé afin de n'établir les invariants que lorsque un cycle complet action/réaction (commande/procédé) a bien été exécuté.

### 2.3.3 Exemples d'application

Le procédé de modélisation proposé a été développé dans le cadre de la thèse de Patrick Lamboley et mis en œuvre, dans le cadre de nos relations contractuelles avec EDF, pour spécifier l'automatisation d'un sous-ensemble restreint d'un procédé continu de production d'énergie. L'intérêt de l'approche réside essentiellement dans

la modélisation des exigences des utilisateurs indépendamment de toutes contraintes de réalisation, notamment au travers de la primitive ANY … WHERE … THEN (Figure 55).

```
MACHINE     System_requirements
SETS        PRESSURE = NAT; FLOW = NAT; VOLUME = NAT,
            MODE={Automatic, Manual}, MODECHG={M2A, A2M}, HISTORIC;
CONSTANTS   Lmax = 100, Input_flow_max = 100
VARIABLES   request, input_flow, output_flow, tank_level, mode,
            historic, chg_mode_historic, monitoring, level
INVARIANT   (tank_level = Lmax ⇒ Output_flow > 0) ∧ …                    (I1)
            ∧historic ⊆ HISTORIC ∧ monitoring ∈ historic → NAT           (I2)
OPERATIONS
Level_control (request) =
    PRE     request ∈ NAT
    ANY     new_output_flow, new_tank_level
    WHERE   new_output_flow ∈ [0, 100] ∧ new_tank_level = request
    THEN    tank_level := new_tank_level || output_flow := new_output_flow END
Change_mode =
    ANY     newmode
    WHERE   newmode ∈ NAT
            ∧ (newmode = automatic ⇒ tank_level ∈ [request - 10, request + 10]))
    THEN    mode := newmode
Create_historic (h) =
    SELECT  newmode = change_mode(mode) ∧ newmode /# newmode$0
    THEN    historic := historic ∪ {h}
            || IF newmode = Automatic and newmode$0 = Manual
            THEN monitoring(h) := A2M ELSIF monitoring(h) := M2A
            || level := request END
END
```

Figure 55. Exemple de spécification des exigences extrait de [R4]

Le procédé de modélisation a également été éprouvé, dans le cadre de notre collaboration avec l'INRS, pour l'automatisation d'une presse industrielle [C20]. L'exemple relatif à la spécification du système de type « procédé continu » étant présenté en annexe de ce document [R4], nous avons choisi d'illustrer notre approche sur l'exemple de la presse industrielle, et plus particulièrement sur un de ces modes de marche : le mode en coup par coup, suffisamment simple pour être exposé dans le corps de ce document mais néanmoins représentatif de la démarche.

#### 2.3.3.1 Spécification des exigences

L'exemple étudié est relatif à une presse mécanique permettant d'emboutir des produits métalliques à l'aide d'un outil dont la translation verticale est assurée par un vilebrequin équipé de deux capteurs « point mort » haut et bas. Un embrayage, actionné par une valve pneumatique, assure la transmission entre le moteur et le vilebrequin. La protection de l'opérateur est assurée par une commande bimanuelle (aucun mouvement descendant n'est possible si l'opérateur ne pose pas ses deux mains sur la commande). Dans le mode de marche coup par coup, l'opérateur commande chaque opération d'emboutissage : la presse est à l'état initial du cycle en position haute, une commande de l'opérateur entraîne la descente de l'outil jusqu'à ce que le point mort bas soit atteint et remonte ensuite pour s'arrêter en position haute. Si l'opérateur libère la commande bimanuelle lors de la descente ou en cas d'arrêt d'urgence, l'outil remonte immédiatement.

Le premier modèle décrit en B est un modèle volontairement très abstrait qui se contente de décrire les principaux états de fonctionnement du mode coup par coup. En d'autres termes, ce modèle caractérise les transitions d'états de la machine (caractérisation des variables avant/après) sans différencier les conséquences (mouvement) des causes (stimuli de l'opérateur). En conséquence, les variables du premier niveau de spécifications sont les suivantes : *state1* représente l'état abstrait de la presse pouvant prendre les valeurs à l'arrêt (As *stopped1*), descente (*falling1*) ou montée (*rising1*), deux variables (*start*, *emergency*) représentent les actions de l'opérateur (arrêt d'urgence et commande bimanuelle).

**MODEL** Press_System 1
**SETS** STATES1= {stopped1, falling1, rising1};
PIN= {on, off}
**VARIABLES** state1, start, emergency
**INITIALISATION**
state1= stopped1 || start = off || emergency = off

L'invariant caractérise, dans l'espace d'états considéré, les propriétés S1 et S2 : le mouvement de descente n'est permis que si et seulement si les mains de l'opérateur sont sur les deux mains commande et l'arrêt d'urgence n'est pas engagé (S1), le mouvement de remontée est permis si et seulement si l'arrêt d'urgence n'est pas engagé.

**INVARIANT**
start ∈ PIN ∧ emergency ∈ PIN ∧ state1 ∈ STATES1 ∧
(state1= rising1 ⇒ emergency= off) ∧ *(S1)*
(state1= falling1 ⇒ start= on ∧ emergency= off) *(S2)*

Le comportement attendu de la presse est décrit par trois opérations : lorsque l'opérateur déclenche la commande bimanuelle, la presse descend, lorsque la presse atteint le point mort bas, elle remonte et enfin lorsqu'elle atteint le point mort haut, elle s'arrête. Deux opérations décrivent les protections de l'opérateur: la presse s'arrête si la commande bimanuelle est relâchée pendant la phase de descente ou si l'arrêt d'urgence est enclenché quelque soit l'état de la presse. Enfin, deux opérations précisent les conditions de redémarrage après un arrêt incidentel (arrêt en phase de descente suite à un arrêt d'urgence et arrêt en phase de montée).

**OPERATIONS**

Starting =
**Select** state1 = stopped1 ∧ start = off
**Then** state1 := falling1 || start := on || emergency := off
**End**;

Direction_shift =
**Select** state1 = falling1
**Then** state1 := rising1 **End**;

Stop_up =
**Select** state1 =rising1
**Then** state1 := stopped1 || start ∈ PIN **End**;

Start_relax_stop =
**Select** state1 = falling1 ∧ start = on
**Then** state1 := stopped1 || start := off **End**;

Emergency_stop =
**Select** state1 = falling1 ∧ emergency = off
**Then** state1 := stopped1 || emergency := on
**When** state1 = rising1 ∧ emergency = off
**Then** state1 := stopped1 || emergency := on **End**;

Rising_restart =
**Select** state1 = stopped1 ∧ emergency = on
**Then** state1 := rising1 || start ∈ PIN || emergency := off
**End**;

Falling_restart =
**Select** state1 = stopped1 ∧ start = off ∧ emergency = on
**Then** state1 := falling1 || start := on || emergency := off
**End**;

Cette première machine constitue une **première formalisation des exigences** des utilisateurs quant au fonctionnement de la presse et à sa sécurité dans la mesure où

l'on se place dans la position d'un observateur constatant les modifications d'état du système sans préciser les règles de commande permettant d'atteindre ce comportement. Il est à noter que les deux dernières opérations (*falling_restart*, *rising_restart*) peuvent avoir simultanément leur garde respective à vrai. Ceci peut s'expliquer facilement par le fait que l'état d'arrêt n'étant pas mémorisé, les conditions de redémarrage restent non déterministes.

Ce point justifie de procéder à un **raffinement** de cette première machine pour y détailler l'état d'arrêt et le scinder en deux : arrêt en position haute (*top_stopped*), arrêt en cours de descente (*falling_stopped*) et arrêt en cours de remontée (*rising_stopped*). Ceci se traduit par le raffinement de la variable State1 en une variable State2 ayant un nouveau domaine de définition et par la modification des gardes et des substitutions des opérations impliquant la variable state1. Le lien entre les variables state1 et state2 est défini dans un **invariant de collage** que l'on doit vérifier afin de s'assurer de la cohérence de la spécification raffinée vis-à-vis de la spécification initiale.

**REFINEMENT** Press_System_2
**REFINES** Press_System_1
**SETS** STATES2 = {falling_stopped2, rising_stopped2, top_stopped2 , falling2, Rising2}
**VARIABLES** start, emergency, state2
**INVARIANT**
…….. ∧
***/* invariant de collage */***
state2 ∈ {falling_stopped2, rising_stopped2, top_stopped2} ⇔ state1= stopped1 ∧
state2= falling2 ⇔ state1= falling1 ∧
state2= rising2 ⇔ state1= rising1 ∧
…..
**OPERATIONS**
…
Rising_restart =
Select state2= rising_stop2 ∧ emergency= on
**Then** state2:= rising2 || start :∈ PIN || emergency:=off **End**;
…
**END**

Cette machine est, pour la suite, considérée comme la spécification formelle des exigences que devra respecter la solution proposée en termes d'automatisation.

#### 2.3.3.2 Automatisation de la presse

La spécification de la presse est selon le prédicat P2 décomposée en spécifications relatives au **procédé**, à la **commande** mais également à son **environnement** matérialisé ici par les actions des opérateurs. Pour des soucis de concision et compte tenu de la simplicité de l'exemple proposé, nous avons choisi de représenter ces opérations au sein de la même machine B (Press_System3) sans utiliser les primitives de composition de B dans la mesure où celles-ci n'engendrent pas de différence notoire quant au processus de preuve.
Selon l'interprétation que nous proposons du prédicat P2, cette nouvelle machine devra être un **raffinement** des spécifications, ce qui signifie que la solution proposée sera en tout point conforme aux **exigences formalisées**. De nouvelles variables devront être introduites permettant de décrire les fonctions de commande (commande de l'électrovanne préactionnant l'embrayage), les variables représentatives de l'état du procédé (position vilebrequin, capteurs « point mort », …) et de l'environnement (commande bimanuelle, arrêt d'urgence).

La modélisation du comportement de l'environnement est très simple dans la mesure où les stimuli arrêt d'urgence et commande bimanuelle sont supposés incontrôlables.

```
Operator_actions =
Begin   start3 :∈ PIN || emergency3 :∈ PIN  End;
```

La modélisation du procédé décrit la rotation du vilebrequin ($\alpha$ représente la position du vilebrequin et *crank* représente l'enclenchement de l'embrayage). Cette opération produit deux variables (*bdc*, *tdc*) représentant respectivement les capteurs « point mort » bas et haut.

```
Move =
If pressure = on
        Then
                If α = 359 then α:= 0 || crank:=on
                Else α:= α+1 || crank:=on
Else crank := off  End;

Dead_centres =
          if α = 0 then bdc := off || tdc = on
          else if α = 180 then bdc=on || tdc= off
          else bdc=off|| tdc=off  End;
```

La description de la commande de la presse nécessite l'introduction de nouvelles variables (*old_emergency3*, *old_start3*) mémorisant les actions de l'opérateur afin de pouvoir exploiter la notion de fronts montants et descendants de ces actions. Ces nouvelles variables conduisent à une modification des gardes des opérations spécifiées dans les deux premières machines. A titre d'exemple, l'opération *starting* dont le résultat se traduit par la commande de l'électrovanne préactionnant l'embrayage (*pressure*) est ainsi raffinée en :

```
Starting =
Select   state3= top_stopped3 ∧
         start3= on ∧ oldstart3= off  /* Start rising edge */
Then  state3:= falling3 || start3:= on || emergency3:= off || oldemergency3:=emergency3 || oldstart3:=start3 ||
         pressure:= on  End;
```

A noter que la variable *state3* constitue une variable interne de la commande dont l'évaluation fait l'objet d'une nouvelle opération[27] *state*. Le résultat produit par cette opération sera supposé cohérent (la vérification du raffinement le démontrera) avec la variable abstraite de la spécification des exigences *state2*.

```
State =
Select tdc = on state3 = rising3
Then state3:= top_stopped3
When …
End;
```

Enfin, l'**invariant de collage** permet d'établir les liens entre toutes les variables concrètes décrivant le comportement du procédé, de la commande et de l'environnement et les variables présentes dans la spécification des exigences.

**INVARIANT**

$state2 \in \{falling\_stopped2, rising\_stopped2, top\_stopped2\}$
$\Leftrightarrow$ crank = off $\wedge$
$state3 \in \{falling\_stopped3, rising\_stopped3, top\_stopped3\} \wedge$
$state2=falling2 \Leftrightarrow crank= on \wedge 0 \leq \alpha < 180 \wedge state3 = falling3 \wedge$
$state2=rising2 \Leftrightarrow crank=on \wedge 180 \leq \alpha < 359 \wedge state3 = rising3 \wedge \ldots$

[27] L'ajout d'une opération dans un raffinement est possible en considérant que la nouvelle opération raffine l'opération « nulle » (*skip*) de la spécification abstraite.

Enfin, les propriétés « système » de sécurité S1 et S2 identifiées dans la spécification des exigences sont **projetées** sur les différentes opérations (qui correspondent en réalité aux différents composants du système) en tenant compte des relations de raffinement des variables. A titre d'exemple, le terme *state2 = rising2* de la propriété système S1 devient *state3 = rising3* (projection sur la commande pour laquelle la variable *state3* ne représente plus un état observé mais un état calculé par la commande) et *crank = on* $\wedge$ *180* $\leq\alpha<$ *359* (projection sur le modèle de procédé).

*

$$(state3 = rising3) \wedge (crank = on \wedge 180 \leq \alpha < 359) \Rightarrow emergency3 = off) \wedge \quad (S1)$$
$$(state3 = falling3) \wedge (crank = on \wedge 0 \leq \alpha < 180) \Rightarrow start = on \wedge emergency = off)$$
$$\wedge \ldots \quad (S2)$$

Etablir la **preuve de cet invariant** (collage et projection des propriétés de sécurité) revient à établir la **conformité de la spécification** du système automatisé « presse » vis-à-vis de la spécification formelle des **exigences**. Cette preuve repose sur l'hypothèse que le système de commande est capable d'évaluer correctement l'état de la presse en fonction des variables « capteurs » produites par les opérations relatives au procédé et que le système physique réagit correctement aux stimuli de la partie commande. Ces obligations de preuves constituent en fait les **propriétés locales** relatives :

- à la commande

$$state = rising\ 3 \Rightarrow pressure = on \wedge 180 \leq \alpha < 359 \Leftrightarrow state3 \in \{rising3, rising\_stopped3\}$$

- au procédé

$$pressure = on \Leftrightarrow crank = on$$

Afin de procéder à la vérification de ces hypothèse, il convient d'introduire dans la spécification le **joueur B** présenté au paragraphe précédent permettant de ne considérer les invariants qu'à l'issue d'un cycle environnement/commande/procédé. Le prouveur de l'Atelier B réalise alors les démonstrations nécessaires, de manière plus ou moins interactive selon la complexité des preuves, sous réserve que les **spécifications** de la **commande** et du **procédé** soient conformes au comportement attendu.

Pour conclure, la présentation de cet exemple très simple est destinée à illustrer la démarche que nous poursuivons pour établir **a priori** le prédicat d'automatisation au travers du mécanisme de **raffinement** tout en fournissant un **cadre méthodologique** pour la spécification en B des systèmes automatisés.

# 3. FORMALISATION DES CONNAISSANCES EN SPECIFICATION

## *3.1 Problème*

La méthode formelle de spécification en B présentée au paragraphe précédent doit permettre d'augmenter la qualité – pertinence, complétude, correction et cohérence – des **modèles** élaborés. Cependant, si l'on considère que l'activité cognitive de spécification repose sur des **connaissances** et **propriétés** relatives à un domaine d'application, il en résulte que les modèles de spécification doivent non seulement

**vérifier les besoins des utilisateurs** mais également **être conforme aux connaissances** dont on dispose pour décrire ce domaine. En effet, ces dernières sont indépendantes du modèle de système particulier spécifié et doivent être préservées durant la phase d'analyse. (Easterbrook, 2002) formule ce problème comme la recherche d'un modèle qui respecte implicitement les propriétés du domaine et l'ensemble des besoins exprimés, selon le prédicat :

*Propriétés du Domaine* $\wedge$ *Spécifications* $\supset$ *Besoins exprimés* (Easterbrook, 2002)

L'expression et la **formalisation des connaissances** et propriétés spécifiques à un domaine de modélisation au moyen d'un langage formel unique et identique aux formalismes utilisés en spécification – ou par défaut compatibles entre eux – s'avère donc nécessaire en vue de :

- guider l'activité du modélisateur en lui proposant des **règles sémantiques** de construction des modèles lui permettant d'intégrer aussi bien les points de vue des différents acteurs que les propriétés invariantes de leur domaine d'application,
- s'assurer de la **correction des modèles** développés vis-à-vis des **connaissances** « métier » relatives au domaine d'application,
- favoriser la **réutilisation** de la connaissance acquise pour faciliter l'élaboration des modèles de spécification.

Cette formalisation suppose de considérer un niveau d'abstraction supplémentaire pour lequel l'objet de la modélisation n'est plus le système lui-même mais les concepts syntaxiques et/ou sémantiques nécessaires à la réalisation d'un modèle particulier. Ces concepts, considérés comme des éléments de modélisations instanciables à un domaine d'utilisation particulier, peuvent être formalisés par une collection d'éléments génériques ou ***constructs***, définis par la norme ISO 19440 comme "*a textual or graphical artefact devised to represent in an orderly way the diverse information on common properties and elements of a collection of phenomena*". Cette approche s'inscrit dans le cadre des travaux autour de la méta-modélisation et des ontologies (Panetto 2006)(Gruber 1993).

## *3.2 Contribution*

### 3.2.1 Formalisation de *constructs* à l'aide du langage B

Nous avons donc cherché à voir dans quelle mesure le langage B pouvait être utilisé pour la formalisation de connaissances en spécification de systèmes automatisés [R3][C11][C14][C16]. Le principal avantage réside dans l'utilisation d'un langage formel unique pour la représentation des connaissances (① de la Figure 56) et la spécification des systèmes automatisés (② de la Figure 56) qui facilite la définition d'un mécanisme d'instanciation entre ces deux niveaux d'abstraction.
La formalisation de *constructs* en B est basée sur l'identification, par des experts, des **constantes** et des **propriétés invariantes** du domaine d'application – modélisées au travers des clauses *Set*, *Constant*, *Properties*, *Variables* et *Invariant* – mais également, lorsque cela est possible, de **comportements génériques** – modélisés au travers des clauses *Initialisation* et *Operations*. Il est à noter que les propriétés invariantes peuvent être **locales** à un *construct* donné – établies sous la forme d'un prédicat reliant les variables de la machine décrivant le construct – mais également caractériser les **règles d'assemblages** des *constructs* – l'invariant établit

dans ce cas une relation logique ou de typage entre des variables appartenant à des *constructs* différents et partageables au travers des primitives de composition supportées par le langage B. Ces *constructs*, formalisés sous la forme de machines B traduisent ainsi un ensemble de règles plus ou moins génériques pour la spécification d'un système automatisé de production.

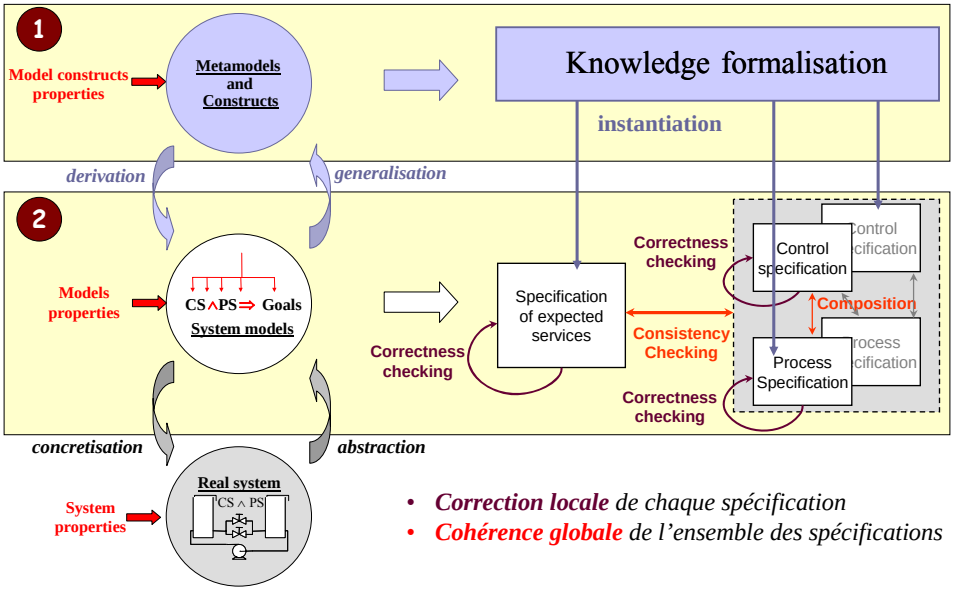



Figure 56. Niveaux d'abstraction en spécification formelle

Afin de permettre la construction d'une spécification en B, il est alors nécessaire de définir un mécanisme d'instanciation adapté. Deux règles de ré-écriture ont été proposées en ce sens :

- une machine B représentant un *construct* est dupliquée, le nom de son instance étant composé du nom du *construct* préfixé par le nom de l'instance (*instance1.construct*$_i$)
- la spécification contenant l'ensemble des instances est décrite sous la forme d'une machine B qui appelle, par la primitive EXTENDS, les différentes instances de *constructs* (*EXTENDS instance1.construct*$_i$, *instance2.construct*$_i$, *instance2.construct*$_j$, etc). Par définition, cette machine hérite donc des clauses *constants*, *sets*, *variables*, *invariants*, *initialisation* et *operations* définies dans les *constructs*. En particulier, cela signifie que les propriétés des *constructs* (locales ou règles d'assemblages entre *constructs*) feront partie, **par construction**, des propriétés invariantes du modèle de spécification.

### 3.2.2 Exemple d'application

L'exemple d'application proposé concerne la formalisation de connaissances pour la spécification de procédés de transformation de matière et d'énergie. Notre approche repose sur les travaux de thèse de F. Mayer (Mayer 1995) qui avait montré qu'un procédé systémique de modélisation pouvait être semi-formalisé sur la base d'un paradigme ensembliste et de C. Féliot (Féliot 1997) qui ont confirmé qu'un processus physique de transformation de matière peut être modélisé sous la forme d'un réseau d'opérateurs.

Les opérateurs génériques de temps, d'espace et de forme proposés par Feliot sont définis comme des transformateurs de couples de variables physiques de flux, d'effort, de déplacement et d'impulsion sur la base des travaux de Paynter (Paynter 1961). Ces opérateurs sont formalisés en B sous la forme:

- de couples de variables définies par Feliot et instanciées à un domaine d'application donné (les systèmes hydrauliques dans notre cas),
- de propriétés invariantes caractérisant les relations entre les variables de Paynter (Figure 57),
- d'une description abstraite du comportement du transformateur qui a comme seul objectif, dans un premier temps, de maintenir les relations invariantes entre les variables de Paynter (clause *Any … Where … Then*).

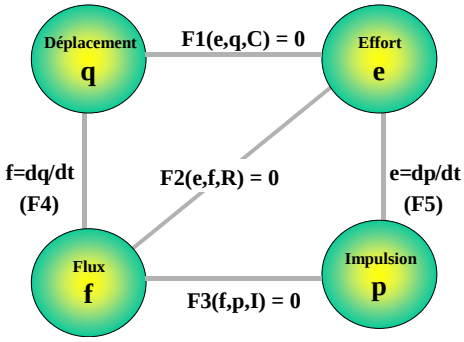
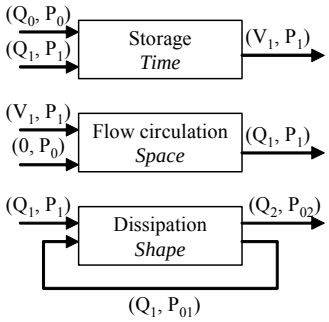



Figure 57. Tetrahédre d'états (Paynter 1961)

La Figure 58 présente la formalisation d'un opérateur de Temps appliqué au domaine de l'hydraulique (stockage d'un fluide dans une cuve) où $(Q_0, P_0)$ et $(Q_1, P_1)$ représente respectivement les couples (débit, pression) en entrée et sortie et où $(V_1, P_1)$ représente le volume et la pression dans la cuve. Les opérateurs d'espace et de forme ont été modélisés en suivant les mêmes principes. L'invariant représente les dépendances paramétrées entre les variables de la machine B.




**MACHINE** Hydraulic_Time_processor
**SETS** EFFORT, FLOW, DISP, IMPULSE
**CONSTANTS** A, $\rho$, g
**VARIABLES** $V_1$,$Q_0$,$P_0$,$Q_1$,$P_1$
**INVARIANT**
$V_1 \in DISP \wedge P_1 \in EFFORT \wedge$ ... */* type checking */*
$\wedge (V_1 = (A / (\rho * g)) * P_1)$ */* invariant relationships */*
**OPERATIONS** $(V_1,P_1) \leftarrow$ Time $((Q_0,P_0),(Q_1,P_1)) =$
PRE $(Q_0, P_0) \in$ FLOW $\rightarrow$ EFFORT $\wedge$ …
THEN ANY new$V_1$, new$P_1$
WHERE (new$V_1$, new$P_1$) $\in$ DISP $\rightarrow$ EFFORT $\wedge$ ($V_1$ = (A / ($\rho$*g)) * $P_1$)
THEN $V_1$ := new$V_1$ || $P_1$ := new$P_1$
**END ;**

*a) Feliot's processors* *b) B formalisation of a Time processor*

Figure 58. Première formalisation B des opérateurs de (Féliot 1997)

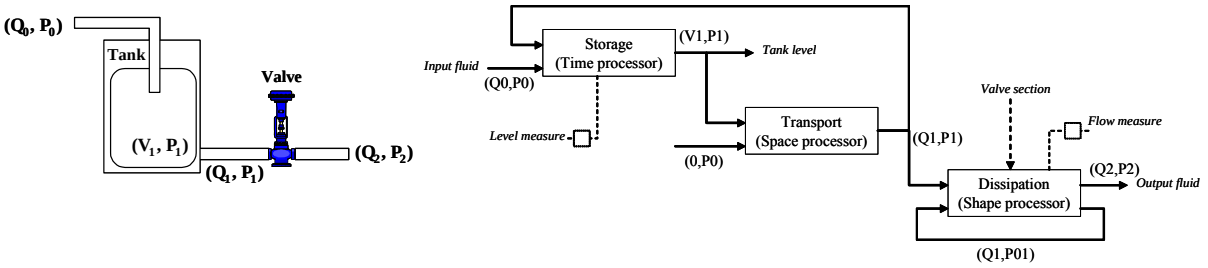
Un raffinement de ces premières machines permet d'introduire une description qualitative, sous la forme de tendances (augmentation, diminution, stabilité) des transformations réalisées par les opérateurs de Feliot, considérée comme suffisante en phase de spécification (Figure 59).

| Produced Variables | Output Variation | $P_0$ | $\Delta Q = Q_0 - Q_1$ |
|---|---|---|---|
| $P_1$ | → | → | |
| | ↓ | ↓ | |
| | ↑ | ↑ | |
| $V_1$ | → | | $\Delta Q = 0$ |
| | ↓ | | $\Delta Q < 0$ |
| | ↑ | | $\Delta Q > 0$ |

**REFINEMENT** Hydraulic_Time_processor
**VARIABLES** $Vr_1$, $Qr_0$, $Pr_0$, $Qr_1$, $Pr_1$
**INVARIANT**
$(Vr_1 \in DISP) \wedge (Vr_1 = V_1) \wedge$
$(Qr_1 \in FLOW) \wedge (Qr_1 = Q_1) \wedge ...$
**OPERATIONS**
$(Vr_1, Pr_1) \leftarrow$ Time $((Qr_0, Pr_0),(Qr_1, Pr_1)) =$
PRE $(Q_0, P_0) \in$ FLOW → EFFORT $\wedge$ …
THEN SELECT $(Q_{r0}-Q_{r1})>0$
THEN ANY $newV_1$, $newP_1$
WHERE $(newV_1,newP_1) \in$ DISP → EFFORT
$\wedge\ newV_1 > V_{r1} \wedge (newV_1 = (A / (\rho * g)) * newP_1)$
THEN $V_{r1} := newV_1$ || $P_{r1} := newP_1$
**END**

Figure 59. Modélisation qualitative en B des transformateurs de Féliot

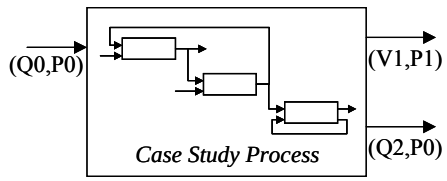
Par instanciation de ces *constructs* formels, il est alors possible de procéder à la spécification d'un procédé physique. La Figure 60a présente un exemple simple de procédé hydraulique composé d'une cuve et d'une vanne de régulation. Les opérateurs de Feliot requis pour la spécification d'un tel procédé (temps, espace et forme) sont associés aux différents organes impliqués dans le procédé. Il est à noter qu'un même composant physique peut être associé à plusieurs opérateurs : c'est notamment le cas de la vanne impliquée dans la circulation des fluides qui est représentée à la fois comme un opérateur d'espace (transport du fluide) mais également comme un opérateur de forme afin de tenir compte des pertes de charges. Le procédé sera alors représenté comme un **réseau d'opérateurs interconnectés** (Figure 60b).




a) exemple de procédé b) modélisation sous la forme d'un réseau d'opérateurs

Figure 60. Modélisation d'un procédé hydraulique à l'aide des opérateurs de Féliot

Ce réseau peut ensuite être formalisé en B (Figure 61). Les machines décrivant les *constructs* sont instanciées en préfixant leur nom (*Tank.Time*, *Valve.Space*, et *Valve.Shape*) et invoquées par une machine *Case_study_process* (clause EXTENDS), les paramètres d'appel étant définis par les arcs du réseau.




**MACHINE** Case_study_process
**VARIABLES** $Q_0$, $P_0$, $V_1$, $P_1$, $Q_1$, $Q_2$, $P_{01}$
**EXTENDS** Tank.Time, Valve.Space, Valve.Shape
**INVARIANT** $(V_1=0 \Rightarrow Q_2=0) \wedge (Q_2=Q_1) \wedge$ …
**OPERATIONS**
$((Q_2, P_0),(V_1, P_1)) \leftarrow$ Case study process $(Q_0, P_0) =$
$(V_1, P_1) \leftarrow$ Tank.Time $((Q_0, P_0),(Q_1, P_1))$
|| $(Q_1, P_1) \leftarrow$ Valve.Space $((V_1, P_1),(0, P_0))$
|| $((Q_2,P_2),(Q_1,P_{01})) \leftarrow$ Valve.Shape $((Q_1,P_1),(Q_1,P_{01}))$
**END ;**

Figure 61. Formalisation en B d'un réseau d'opérateurs

Trois niveaux de vérification peuvent alors être effectués sur la spécification formelle de procédé ainsi obtenue :

- la clause EXTENDS impose aux opérations invoquées de respecter les invariants des machines dans lesquelles elles sont définies (i.e. dans les *constructs*). Ceci implique en particulier que les **invariants de typage** contenus dans le *construct* doivent être vérifiés par les paramètres d'appel de l'opération invoquée ; en d'autres termes, la construction du réseau d'opérateurs (et en particulier de leur connexion) correspond à une réalité physique en accord avec les connaissances formalisées,
- des **propriétés locales** relatives au domaine d'application et/ou au procédé peuvent être intégrées et vérifiées dans la machine représentant le réseau d'opérateurs ; à titre d'exemple, l'invariant de la Figure 61 indique que le débit est identique avant et après la vanne et que le débit de sortie de la cuve ne peut être que nul si celle-ci est vide,
- des **propriétés relatives à la structure du réseau** d'opérateurs peuvent être introduites sous la forme d'un invariant ; en particulier, Féliot a démontré que seules certaines **séquences** d'opérateurs étaient **admissibles** du point de vue des lois de la physique (par exemple la séquence Temps/Espace/Forme/Espace/Temps. Les opérateurs étant caractérisés par la nature de leur flux d'entrée et de sortie, la représentation de la séquence dans l'invariant se fera sous la forme de relation de typage.

Pour conclure la présentation de cet exemple, les principaux intérêts de la formalisation en B de connaissances expertes en modélisation des procédés physiques peuvent se décliner selon :

- un axe **modélisation** puisque la réutilisation de connaissances expertes permet de systématiser la démarche de construction des spécifications, notamment grâce à l'utilisation d'un formalisme unique pour la modélisation des connaissances et la spécification,
- un axe **vérification** puisque l'instanciation de *constructs* formels permet de garantir la correction du modèle vis-à-vis des connaissances expertes formalisées, tant sur le plan structurel que comportemental.

## 4. CONCLUSION

Les travaux présentés dans ce chapitre s'inscrivent dans le cadre d'une **ingénierie système** visant à faciliter, au plus tôt, une **représentation commune et consensuelle** des services attendus d'un système automatisé par les différents acteurs du procédé d'automatisation. Ils ont pour objet de proposer, notamment dans la phase initiale de spécification, une méthode formelle vérifiant le prédicat d'automatisation P2 (*Système de commande* $\wedge$ *Système opérant* $\supset$ *Exigences Système*). De manière complémentaires aux travaux développés en modélisation des S.E.D., notamment dans le cadre de la théorie de la Supervision (Ramadge & Wonham), pour lesquels les objectifs du système à automatiser et les comportements des processus opérants sont parfaitement connus et modélisés, notre approche se caractérise par un **processus incrémental de spécification** qui permet d'aboutir progressivement à une vision commune et cohérente des différents processus d'ingénierie (ingénierie d'automatisation, ingénierie de mécanisation, …).

La méthode proposée est basée sur l'utilisation d'un langage formel unique, le langage B, pour construire une spécification abstraite, qui constitue une ré-écriture cohérente du cahier des charges, située en amont des représentations orientées métiers. Les mécanismes formels de preuves permettent d'établir la correction intrinsèque des spécifications vis de vis de contraintes et connaissances du domaine concerné et d'assurer leur cohérence globale en préservant les relations logiques établies par le prédicat.
Si le langage B s'avère performant dans le cadre de la spécification des systèmes automatisés, il ne se prête guère en revanche à la phase de conception de ces systèmes qui impose un outil de description explicite du temps. La relation formelle entre les spécifications B et les modèles de SED reste encore à établir et fait partie des axes de recherche à développer. D'autre part, la spécification d'un système complexe impose bien souvent l'utilisation conjointe de modèles semi-formels permettant d'éliciter et d'appréhender globalement le besoin, et de modèles formels permettant de prouver certaines de leurs propriétés. En ce sens, les travaux autour de B et UML (Meyer 2001) constituent un atout incontestable de ce langage, d'autant plus que les approches développées autour de OOONEIDA (Auinger *et al.* 2005) ou CORFU (Tranoris & Thramboulidis 2006) établissent un lien entre UML et la modélisation des SED selon la norme IEC61499.

# V Programme de recherche : méthodes et modèles pour un processus sûr d'automatisation

Mon projet de recherche intitulé « méthodes et modèles pour un processus sûr d'automatisation », a pour objectif de contribuer à la maîtrise de la qualité des modèles produits au cours du processus d'automatisation depuis l'identification et la modélisation des exigences jusqu'au déploiement d'une solution.

Afin de couvrir la dimension « système » d'une automatisation, justifiée notamment par une intégration de plus en plus forte des automatismes au sein des systèmes d'entreprises, notre démarche de travail se fonde :

- sur des méthodes issues du Génie Automatique et des modèles formels issus de l'automatique des Systèmes à Evénements Discrets et du Génie Informatique,
- sur des expériences industrielles fortes qui se sont avérées fructueuses sur le plan des résultats et particulièrement motivantes pour les chercheurs.

Le programme de recherche, développé ci-après, est construit sous une forme qui nous a permis de le proposer comme une action de l'équipe/projet Systèmes InterOpérants du CRAN partagée entre le groupe thématique SYMPA et SURFDIAG lors de la définition du quadriennal 2009-2012.

## 1. CONTEXTE ET ENJEUX

A l'ère du déploiement des nouvelles technologies de l'information et de la communication, l'un des enjeux majeurs de l'entreprise est **d'adapter dynamiquement** son offre de produits (biens ou service), en fonction du nouveau comportement des consommateurs qui révèle une demande de plus en plus personnalisée et génératrice de concurrence (Brown *et al.* 1995, Ollero *et al*. 2002, Nof *et al.* 2006). Dans le domaine du contrôle de la production, cette évolution conduit à repenser l'organisation hiérarchique des circuits de prise de décision au profit de structures distribuées et coopérantes, beaucoup plus flexibles, caractérisées par une capacité de réorganisation des entités qui les composent (McFarlane & Bussmann 2000). Ainsi, tirant parti des progrès technologiques dans le domaine de la communication sans fil, des automatismes industriels (interfaces ou services Web embarqués dans les automates programmables industriels) ou encore dans les domaines de l'électronique et de l'informatique (RFID, réseaux de capteurs, composants logiciels embarqués, …), l'automatisation des systèmes de production intègre une part de plus en plus importante d'**objets communicants**, **mobiles** pour certains, offrant un ensemble de services pour le contrôle, la commande mais aussi la surveillance, le diagnostic ou la configuration des systèmes de production. En particulier, le concept de **produit actif** est un type particulier de systèmes mobiles (McFarlane *et al.* 2003), porteur, au-delà de ses caractéristiques physiques, d'un ensemble de connaissances et de capacités de communication lui conférant un degré d'autonomie vis-à-vis des systèmes environnants et des possibilités d'interaction et de synchronisation avec cet environnement.

Dans ce contexte, garantir une certaine cohérence entre un monde virtuel contrôlant des flux d'informations et de décisions et le monde réel composé d'objets et de flux physiques, assurer l'interopérabilité entre les systèmes d'entreprises (E.R.P., M.E.S., …) et les systèmes de contrôle et de commande de la production, fiabiliser la prise de décision grâce à des informations parfaitement représentatives de l'état courant des processus industriels, favoriser la réactivité et l'adaptabilité des systèmes de contrôle et de commande de la production, constituent autant de défis que doit relever une automatisation à l'ère d'Internet.

Du point de vue de leur ingénierie, les systèmes de contrôle et de commande de la production ont à faire face à un paradoxe dans la mesure où la **complexité** de leurs architectures basées sur la coopération d'objets logiciels ne cesse d'augmenter (Jonhson 2004) et que, dans le même temps, **la durée de mise sur le marché** de nouveaux produits et, par voie de conséquence, des systèmes capables de les réaliser, ne cesse de se réduire. Il s'agit ainsi pour les chercheurs confrontés à la maîtrise des propriétés de ces nouveaux systèmes de biens et de services, non plus de se focaliser sur le comportement des « composants élémentaires » dont l'analyse et/ou la synthèse repose sur des méthodes et des outils déjà maîtrisés par la communauté scientifique et/ou l'industrie, mais d'appréhender la complexité d'un système constitué d'objets communicants, mobiles, dont les propriétés résultent de phénomènes émergents que l'on souhaite confiner, au travers de modèles et de méthodes susceptibles d'engendrer, dans un temps raisonnable, une certaine confiance dans la **qualité de service** des systèmes développés.

En réponse à cette problématique, l'**Ingénierie Système** (Ménadier, 2002, Sheard 2006) propose un ensemble de normes décrivant les processus du métier d'ingénieur système (IEEE 1220, EIA 632, ISO 15288) ainsi que des méthodes et outils pour maîtriser le développement des systèmes et produits complexes (Penalva, 1997). En ce sens, l'INCOSE[28] est notamment à l'origine, en coopération avec l'OMG, du développement de **SysML** extension d'UML, centré initialement sur le développement de logiciels, pour le développement de systèmes en général. Si, les travaux réalisés autour de SysML offrent un ensemble cohérent de formalismes supportant les multiples vues nécessaires pour représenter l'architecture et le fonctionnement global d'un système complexe à concevoir, leur applicabilité au développement de systèmes automatisés soumis à de fortes contraintes temporelles et de sûreté de fonctionnement reste à démontrer à ce jour.

Ainsi, la définition de cadres de modélisation et de modèles de référence des processus en ingénierie système en vue de faciliter le déploiement cohérent d'un ensemble de modèles abstraits, intuitifs, qualitatifs – nécessaires pour représenter le fonctionnement global d'un système automatisé – conjointement à des modèles plus formels – adaptés à la modélisation, la vérification et la validation de leurs propriétés en présence de phénomènes d'émergence – constitue un des verrous scientifiques majeurs à lever en vue de garantir le respect et la traçabilité des exigences au cours d'un processus sûr d'automatisation.

[28] INCOSE : International Council on System Engineering dont l'AFIS (Association Française d'Ingénierie Système) est membre

## 2. OBJECTIF DE RECHERCHE

Dans un contexte d'ingénierie système, notre projet de recherche a pour objectif de formaliser et d'outiller un processus sûr d'automatisation qui devra couvrir le plus complètement possible les diverses phases classiques d'une automatisation industrielle (spécification, conception, implantation) en s'efforçant de montrer comment les diverses exigences (en termes de sécurité, de sûreté de fonctionnement, d'interopérabilité, de reconfigurabilité, etc.) peuvent être :

- identifiées puis **spécifiées** à partir des besoins exprimés par les utilisateurs, le plus souvent en langage naturel,
- propagées et **raffinées** au cours des différentes phases du processus de développement, notamment en les allouant à des sous-systèmes, des fonctions ou encore des composants,
- **vérifiées** et **validées** dans le cadre d'une approche multi-modèles combinant des formalismes plus ou moins abstraits, plus ou moins formels, sous une forme fonctionnelle, ensembliste, …, permettant d'appréhender, de manière pragmatique, intuitive, voire qualitative, le fonctionnement global d'un système à concevoir,
- **suivies** afin d'offrir les garanties requises, en termes de traçabilité des exigences, par les processus de qualification ou de certification (IEC 61508).

Ce projet procède d'une recherche **méthodologique**, qui cherche à combiner des approches formelles et non formelles basées sur le raffinement de modèles/propriétés pour garantir le respect et la traçabilité des exigences au cours du processus d'automatisation. Il sous-tend également une recherche **technologique** qui permet la définition d'un contexte de recherche tiré par les applications (notamment en termes d'exigences et de propriétés à satisfaire) et qui doit servir de cadre de validation aux résultats méthodologiques.
Notre challenge scientifique est ainsi d'apporter des solutions pour la maîtrise de la qualité d'une automatisation à l'ère des nouvelles technologies de l'information et de la communication en cherchant :

- à améliorer les **processus de développement** des applications de contrôle et de commande de la production en spécifiant, vérifiant et validant, au plus tôt, une vision commune et consensuelle des services attendus,
- à prendre en compte des exigences nouvelles d'interopérabilité, d'adaptabilité ou de mobilité des architectures de commande en particulier dans le cadre des **systèmes interopérants de production** mettant en œuvre des technologies telles que l'intelligence ambiante.

Par rapport à cet objectif de recherche, notre projet est complémentaire des travaux relatifs à l'analyse et la synthèse des Systèmes à Evénements Discrets contribuant à l'évaluation prévisionnelle de la sûreté de fonctionnement des S.E.D., thème dont la reconnaissance par l'IFAC s'est traduite par l'organisation du premier workshop DCDS[29] en 2007. L'originalité de notre projet réside :

- dans la prise en compte d'une dimension « système » qui se traduit par la nécessité pour les modèles de couvrir non seulement les caractéristiques comportementales des systèmes mais aussi leurs composantes

[29] 1st workshop on Dependable Control of Discrete Systems, 13-15 juin 2007, Cachan, http://www.lurpa.ens-cachan.fr/dcds07/

informationnelles, fonctionnelles, architecturales, etc. (Zhang *et al.* 2005, Auinger 2005, Tranoris &Thramboulidis 2006*,* Ferrarini *et al.* 2005),
- dans la volonté de mettre l'accent sur les phases situées en amont de la phase de conception couverte classiquement par les modèles de S.E.D., en particulier pour mettre en œuvre une démarche incrémentale de spécification (Shell 2001, Gout & Lambolais 2004) reposant sur le raffinement progressif des différentes exigences (Garcia-Duque *et al.* 2006) formalisées à un niveau « système ».

Il est à noter que, vis-à-vis du large domaine couvert par la sûreté de fonctionnement des S.E.D., de l'évaluation prévisionnelle de la sûreté de fonctionnement (validation/vérification, analyse fiabiliste à base de modèles probabilistes) jusqu'à l'exploitation des systèmes (surveillance, diagnostic, maintenance, etc.), notre projet se limite aux aspects relatifs à la vérification, la validation et la traçabilité des exigences dans un processus d'automatisation.

## 3. ACTIONS DE RECHERCHE

En cohérence avec l'objectif de recherche préalablement exposé, nous proposons d'initier trois actions de recherche. La première, à vocation méthodologique, a pour objet de développer un **cadre formel d'automatisation** dans un contexte d'**ingénierie système**. Les résultats de cette action doivent alimenter deux actions de recherche complémentaires, à vocation technologique, qui prolongent notre implication passée dans le domaine des systèmes manufacturiers, et plus particulièrement des **systèmes contrôlés par le produit**, mais aussi dans le domaine des **systèmes de production d'énergie** dans le cadre de nos collaborations avec EDF.

### *3.1 Action « Méthodes et Modèles formels pour l'automatisation »*

Cette action porte sur la définition d'un processus d'automatisation sûr mixant des approches de modélisation des Systèmes à Evénements Discrets (Cassandras & Lafortune 1999), avec des approches orientées Objet (UML, SysML, …) ou Système (Penalva 1997, Ménadier 2002, Sheard 2006), plus ou moins formelles, issues du génie automatique, du génie informatique et de l'intelligence artificielle distribuée. L'intérêt de combiner ces types d'approches est double puisqu'il s'agit :
- d'une part, d'étendre les fondements de la théorie des Systèmes en général et de la modélisation des Systèmes à Evénements ou Objets Discrets en particulier afin de rendre compte de la **complexité informationnelle** relative à l'intégration des automatismes dans l'entreprise (IEC 62264) et de la **complexité comportementale** qui émerge des interactions multiples et dynamiques entre les produits et les divers agents (processus, opérateurs, fournisseurs, clients…) concourrant à la production de biens et de services,
- d'autre part, d'augmenter le **niveau d'abstraction** des modèles de S.E.D en proposant des modèles plus généralistes et/ou plus abstraits pour identifier les exigences auxquelles doit répondre le système à développer et définir les **spécifications** fonctionnelles, dynamiques et structurelles ainsi que leurs interactions respectives en termes de comportements, d'architectures et de cycle de vie.

Des approches en ce sens ont été proposées autour de l'utilisation d'UML et de modèles Statecharts ou de blocs fonctionnels de la norme IEC 61499 (Tranoris & Thramboulidis 2006, Chiron & Kouiss 2005). Les méthodes formelles trouvent ici un vaste champ d'application à explorer, puisqu'elles permettent d'élaborer des modèles abstraits précis et, par là même, de vérifier certaines propriétés dès la spécification. En outre, elles permettent une traçabilité fine du processus de développement pour des besoins de certification ou de maintenance.
Dans ce contexte, notre action a pour objet de formaliser une démarche de raffinement des exigences – identification d'exigences « système » puis raffinement en sous-exigences et allocation sur des sous-systèmes ou composants – en s'appuyant d'une part, sur le mécanisme formel de raffinement supporté par la **méthode B**, et, d'autre part, sur le formalisme proposé dans le diagramme des exigences (Requirements diagram) de **SysML**. Une première contribution à la formalisation d'un processus sûr de développement, construite sur la base des diagrammes d'exigences et de composants de SysML ainsi que sur l'utilisation d'outils de simulation comportementale et de model checking, a fait l'objet d'une communication internationale [C22]. L'objectif, à terme, est de pouvoir proposer un patron (*pattern*) formel définissant les objets d'études, présents dans tout processus d'automatisation, et leurs relations.

D'autre part, dans le cadre d'une thèse financée par une bourse MENESR à partir de la rentrée 2007 en collaboration avec le professeur T. DIVOUX (équipe/projet Systèmes Contrôlés en Réseau du CRAN), nous cherchons à prendre en compte, dans la modélisation des applications de commande, le rôle prépondérant joué par la communication et son impact sur la sûreté de fonctionnement. En particulier, nous cherchons à vérifier l'adéquation entre la qualité de service (QoS) requise pour l'application de commande, notamment en terme de retards, cohérence, sûreté, ou encore de sécurité, et les services offerts par le médium de communication, notamment en termes de bande passante, de topologie (redondance des liens), disponibilité, cryptographie. Nous envisageons, pour cela, de nous appuyer sur une modélisation conjointe des applications de commande et de communication afin de définir une stratégie de reconfiguration dynamique, basée sur la typologie et les variables de commande du réseau (réservation mémoire, réservation des chemins, mécanismes de priorité, redondance des liens) pouvant influer sur les propriétés de réactivité, de déterminisme et de sûreté des applications de commande.

### *3.2 Action « Analyse et synthèse des systèmes contrôlés par le produit »*

Cette action se situe dans le prolongement des travaux de l'équipe-projet « Systèmes Contrôlés par le Produit » (Contrat quadriennal du CRAN 2003-2007) dont l'objectif était de définir, développer et déployer des méthodes d'automatisation ayant pour objet le Contrôle et la Gestion par le Produit des Procédés Industriels en Entreprise. Les contributions de cette équipe/projet ont porté sur la définition de nouvelles architectures de systèmes intelligents de production favorisant l'adaptabilité d'organisations de production en réseaux devant fournir à temps et à qualité constante une quantité variable d'un produit personnalisé fabriqué selon des procédés différents sur des sites distants. Cette action a maintenant pour objectif de conforter les fondements des résultats obtenus par la mise en œuvre de moyens de validation, de vérification et de synthèse permettant de garantir les propriétés comportementales des architectures proposées.

**Simulation de systèmes de pilotage par le produit**
Les capacités décisionnelles conférées au produit dans le pilotage des chaînes de production ou dans les applications logistiques conduisent à dissocier la modélisation des systèmes physiques (intégrant des ressources ou des unités de production géographiquement réparties) de la modélisation de leurs systèmes de pilotage respectifs. Le résultat de ce processus de modélisation aboutit à un ensemble de **modèles hétérogènes** (modèles événementiels, modèles multi-agents, modèles logiques basés sur des heuristiques, …). L'objectif est donc de définir les structures adaptées à ces modèles, et d'autre part, des **mécanismes de synchronisation** génériques permettant d'ordonnancer l'exécution et les messages échangés entre ces modèles (HLA, DEVS). Deux cas d'applications sont envisagés : la simulation d'une architecture de pilotage par le produit, basée sur la technologie RFID dans le cadre de la thèse en cours de H. El Haouzi en collaboration avec la société TRANE, la simulation conjointe temps-réel et événementiel du Système de Production Flexible de l'AIP-PRIMECA Lorraine. Dans ce dernier cas, il s'agit de coupler les résultats de **simulation temps-réel** réalisés pour valider la commande de chacun des postes de travail du système flexible avec les résultats d'une **simulation événementielle** permettant d'évaluer la pertinence du contrôle par le produit de ce système flexible. A terme, cette action devra conduire au développement d'une plate-forme d'évaluation d'un système de pilotage partiellement ou totalement distribué basée sur une représentation modulaire des chaînes de production et/ou logistiques et pouvant intégrer des modèles variés de pilotage.

**Reconfiguration des S.E.D. basée sur la synthèse de contrôleurs**
Sur la base des résultats obtenus dans le cadre de la thèse de David Gouyon [TH2], l'objectif est de définir, et de mettre en œuvre sur le Système Flexible de l'AIP-PRIMECA Lorraine, une architecture d'implantation d'un système de commande **dynamiquement reconfigurable** qui pourra être basée :

- sur la proposition de (Xu *et al.* 2002) et les formalismes de la norme IEC61499 pour l'implantation d'une structure d'accueil favorisant l'intégration ou la suppression en ligne de composants de commande,
- sur la synthèse de contrôleurs pour l'élaboration de nouvelles règles de commande ou sur une librairie de composants réutilisables lorsque la reconfiguration se limite à une réorganisation des applications de commande.

Les difficultés majeures sont relatives à la **gestion des configurations** des applications de commande, notamment de leur paramétrage, ainsi qu'à la définition de conditions admissibles de **commutation** entre stratégies de commande.

### *3.3 Projet LABIME*

La conduite de ces procédés met en jeu un ensemble de processus complexes couvrants des modes opératoires variés (en production, en arrêt, en démarrage, etc) adaptés à la criticité des modes d'exploitation rencontrés (conduite normale, conduite incidentelle et accidentelle) selon des constantes de temps différentes (conduite en temps réel, maintenance hors ligne, …). Aujourd'hui, ces processus reposent sur des interactions entre les différents métiers des opérateurs et des systèmes propriétaires, hétérogènes et généralement limités à la phase de production normale. Dans ce contexte, l'apparition d'alarmes ou de défauts provoque souvent le basculement d'une conduite dite « normale » vers une conduite dite « accidentelle » visant essentiellement à préserver la sécurité des personnels, à conserver le système de

production et à limiter les rejets sur l'environnement. La mise en œuvre d'une conduite « incidentelle », limitant ce basculement systématique en conciliant les objectifs de production et de sécurité, constitue un gisement de productivité important. Cette conduite incidentelle mise sur la capacité du système d'information à fournir aux opérateurs les moyens leur permettant d'analyser la situation d'état dans lequel se trouve le procédé, de choisir des procédures appropriées à ces situations et permettant de produire dans des conditions de sécurité acceptable, de décliner ces procédures sous la forme de plans d'actions et enfin d'assurer le suivi et la surveillance du système de production, ce qui suppose notamment :

- la mise à disposition des opérateurs de conduite, en **temps réel**, d'informations **synthétiques**, **fiables** et directement **interprétables** émanant des algorithmes de surveillance, de détection, voire de pronostic des équipements et matériels,
- l'élaboration, à partir de ces informations, d'indicateurs relatifs à la disponibilité des systèmes et sous-systèmes dans lesquels interviennent les équipements et matériels surveillés en tenant compte de la **structuration hiérarchique** utilisée par les opérateurs (objets techniques de base, fonctions, unité de production),
- la **coordination** des différents corps de métiers en exploitation – conduite, maintenance – intervenant sur les équipements, quelque soit leur localisation – en salle de commande centralisée, sur site.

En réponse à la problématique industrielle posée par EDF/DER, le projet **LABIME** (LAngage d'expression des Besoins en Informations des Métiers d'Exploitation) dont j'assure la coordination (partenaires : **EDF**, **CRAN**, **LORIA**) a été labellisé en septembre 2007 par le Groupement d'Intérêt Scientifique « Surveillance, Sûreté et Sécurité des Grands Systèmes » (GIS 3SGS impliquant les laboratoires ICD, CReSTIC, HeuDiaSyC, LAGIS, LAMIH, CRAN, LORIA, CEA, EDF/DER). Son premier objectif concerne la spécification des besoins en informations pour une exploitation plus performante et plus sûre des futurs systèmes où les nouvelles technologies (communication sans fil, informatique nomade, …) pourraient apporter un progrès. Cette analyse des besoins, qui ne doit pas préjuger d'une répartition des tâches de conduite, de commande ou de surveillance entre les hommes et les automatismes, se situe très clairement en **amont** de la prise en compte des **facteurs humains** dans la définition de l'imagerie de conduite ainsi que de la répartition homme/système et en amont des **réalisations informatiques** supportant la conduite centralisée ou déportée sur site.
D'un point de vue scientifique, le principal verrou concerne la **modélisation des informations d'exploitation** du procédé**.** Dans le cadre d'une **ingénierie système dirigée par les modèles**, un langage de description, indépendant de la répartition homme/système et des réalisations informatiques doit permettre d'extraire et de formaliser les connaissances relatives au fonctionnement d'une tranche et d'identifier les informations requises pour une exploitation sûre et optimale des procédés satisfaisant les exigences de sécurité et de sûreté de fonctionnement. Il doit donc se situer en amont des langages « métiers » de représentation du procédé basés sur les modèles de la mécanique et de représentation des automatismes basés sur les modèles de l'informatique. En d'autres termes, ce langage de description doit être un **langage métier** sur lequel pourront se fonder les développements ultérieurs des systèmes de conduite et des automatismes en tenant compte de l'offre et des technologies. Enfin, ce langage doit être directement compréhensible par les exploitants en proposant des représentations graphiques pouvant être associées à des représentations plus formelles.

### *3.4 Plate-forme « SafeTech »*

Enfin, pour mener à bien ce projet de recherche, nous avons décidé de nous impliquer très fortement dans le développement d'un **centre d'innovation et de démonstration des technologies sûres de fonctionnement (SafeTech)** en acceptant la coordination de ce programme (avec N. Brinzei). Ce centre, inscrit à la fois dans le cadre du Contrat de Plan Etat-Région 2007-2012 et du Groupement d'Intérêt Scientifique « Surveillance, Sûreté et Sécurité des grands systèmes » (GIS 3SGS voir paragraphe 4.3. pour plus de détails), est destiné, d'une part, à accompagner les travaux de recherche qui seront menés dans ces projets et, d'autre part, à offrir aux entreprises un site de démonstration. Les défis techniques auxquels fera face SafeTech comprennent la conception de systèmes sûrs de fonctionnement, la tolérance active aux défauts, la connectivité sans faille, la fiabilité, la sécurité, la qualité des services, etc. Pour le CRAN, SafeTech doit s'organiser autour de plusieurs éléments comprenant notamment : des plates-formes industrielles pilotes pour validation des concepts de sûreté et sécurité des systèmes complexes, un réseau de capteurs et son système intelligent de surveillance, une plate-forme collaborative et de démonstration supportant les différents outils/méthodes employés pour l'évaluation prévisionnelle de la sûreté de fonctionnement. Ce centre doit constituer un élément important, structurant, et novateur qui devrait amplifier l'impact de notre projet de recherche.

## 4. JUSTIFICATION

### *4.1 Logique scientifique*

Les questions de sûreté et sécurité des systèmes intégrant du logiciel jouent un rôle primordial pour l'adoption des systèmes intégrant des éléments logiciels dans un contexte économique, juridique et sociétal. Les domaines scientifiques concernés par ces enjeux ne sont pas nouveaux et couvrent la modélisation et l'évaluation prévisionnelle de la sûreté de fonctionnement ou encore la supervision, le diagnostic et l'adaptation des systèmes déployés afin de les rendre tolérants aux fautes.

Plus précisément, notre projet, qui porte sur la formalisation d'un processus sûr de développement de systèmes automatisés, s'inscrit :

- **localement**, dans le prolongement des travaux réalisés par l'équipe/projet « Méthodes et Modèles Formels pour l'automatisation des processus industriels » dont j'ai assuré la coordination entre 2000 et 2003 dans le cadre du contrat quadriennal UMR du CRAN.
- **nationalement**, dans la communauté traitant de l'analyse et de la synthèse des modèles de S.E.D. en vue de garantir, par vérification formelle ou par construction, leurs propriétés comportementales ; cette communauté est présente depuis 2000 dans les groupes de travail COSED (Commande Opérationnelle des S.E.D.) du GdR Automatique et ASSF (Automatisation et Systèmes Sûrs de Fonctionnement) du Groupement de Recherche en Productique et plus récemment dans la fusion de ces deux groupes au sein du groupe INCOS (Ingénierie de la Commande et de la Supervision) du GdR MACS.

- **internationalement**, dans le cadre du TC 5.1 de l'IFAC « Manufacturing Plant Control » et de la communauté en ingénierie système fédérée autour du groupe de réflexion INCOSE dont l'AFIS est la composante française.

### *4.2 Partenariats*

Les membres associés à ce projet sont impliqués dans des groupes de recherche ou de travail :

- groupe INCOS (Ingénierie de la Commande et de la Supervision) du GdR MACS co-animé depuis 2005 par N. Rezg et A. Toguyeni,
- groupe de travail « Résilience des Systèmes » de l'AFIS coordonné par J.F. Gajewski (Astrium EADS) et dont la création est récente (mars 2007)

Notre collaboration sur ce thème avec des chercheurs du LORIA (UMR 7503), et plus spécifiquement avec l'équipe MOSEL dirigée par le Pr. D. Méry, apporte des compétences dans le domaine des outils et langages formels mis en œuvre.
De plus, de par la volonté de coopérer scientifiquement et industriellement, les membres du projet entretiennent des partenariats privilégiés :

- avec le LESTER, le LAGIS, le LAG et PRISMa dans le cadre du projet RECSED (Reconfiguration des S.E.D.) soutenu par le GdR MACS et coordonné par P. Berruet (LESTER) ou, de manière plus informelle, avec le LURPA,
- dans le cadre de collaborations industrielles, notamment avec l'INRS, la société TRANE, EDF, TNI, …

### *4.3 Cohérence avec la politique institutionnelle*

Notre projet se positionne de façon très cohérente par rapport à la logique développée au sein du département « Sciences et technologies de l'information et de l'ingénierie » du **CNRS** et à un des objectifs affichés visant à *garantir la sécurité et la sûreté des systèmes matériels et logiciels*. Au niveau de **l'ANR**, les programmes ARA Sécurité, systèmes embarqués et intelligence ambiante, puis SeTin (Sécurité et Informatique) soutiennent fortement ce domaine de recherche.

Ce projet s'intègre également dans le cadre du **Groupement d'Intérêt Scientifique** « Surveillance, Sûreté et Sécurité des Grands Systèmes » (GIS 3SGS) qui a pour objectif de fédérer des équipes scientifiques (ICD, CReSTIC, HeuDiaSyC, LAGIS, LAMIH, CRAN, LORIA, CEA, EDF/DER) en vue de proposer une démarche scientifique globale indispensable pour aborder la sécurité et la sûreté de fonctionnement des systèmes en prenant en compte l'ensemble des interactions entre surveillance, sûreté et sécurité dans un modèle global complexe et intégré. En ce sens, le GIS 3SGS doit apporter des réponses au défi scientifique actuel sur la cohérence entre des modèles globaux représentant le système, et des modèles développés à l'échelle d'un composant ou d'un sous-système.

Au niveau régional, ce projet s'intègre dans le **Contrat de Plan Etat-Région** 2007-2012 « Modélisation, Informations & Systèmes Numériques » (MISN) et plus particulièrement dans l'axe « **Sécurité et Sûreté des Systèmes** » (3S). Cet axe a pour objectif de fédérer les communautés lorraines (CRAN, GREEN, IECN, LORIA, LIEN, LICM, LITA) issues de différentes disciplines (informatique, automatique,

électrotechnique, électronique) qui abordent les questions de la sûreté et de la sécurité des systèmes intégrant du logiciel, et de promouvoir leurs interactions au travers des opérations interdisciplinaires soutenues financièrement sur deux ans. Notre participation au conseil des opérations en tant que représentant du CRAN (avec le Pr. D. Sauter et le Pr. D. Maquin) traduit notre implication dans ce projet.

Enfin, notre projet s'inscrit dans le cadre de l'axe Surveillance, Sécurité et Sûreté des Grands Systèmes de la proposition de Fédération de Recherche Charles Hermitte, Informatique, Automatique, Mathématiques de Lorraine regroupant l'IECN, le LORIA et le CRAN.

## 5. RESSOURCES

Liste des participants : J.F. Pétin, D. Gouyon, G. Morel, D. Evrot, H. El Haouzi, D. Dobre,

A cette liste de participants, il convient d'ajouter l'arrivée de Gilbert Habib pour un travail de thèse débuté en septembre 2007 sur un financement du ministère (bourse MENESR).

Outre le financement externe des doctorants principalement sous forme de conventions CIFRE, ce projet tirera ses ressources financières de son implication dans des programmes de R&D en collaborations industrielles (société TRANE, INRS, EDF/DER, …), dans les programmes régionaux (CPER MISN, axe Sécurité et Sûreté des Systèmes) et au niveau du GIS 3SGS.

Son ancrage, en formation, sur le Master Ingénierie Système en Electronique Electrotechnique Automatique Productique Réseau (IS-EEAPR) et sur l'ESIAL, notamment dans le cadre de projets d'initiation à la recherche ou de micro-thèse, permet de sensibiliser les étudiants aux métiers de la recherche et de constituer ainsi un socle d'étudiants susceptibles de poursuivre leur cursus en 3$^{ème}$ cycle.

# VI Activités d'enseignement et de son administration

## 1. FORMATION INITIALE ET CONTINUE

Mon activité d'enseignement a débuté en tant que vacataire en deuxième année de thèse, financée sous contrat avec le CNRS dans le cadre de programmes européens, puis en tant qu'A.T.E.R. sur un demi-poste en 1994/95 et 1995/96 et à nouveau en tant que vacataire jusqu'en 1998. Durant cette période, mes activités d'enseignement se sont effectuées dans les formations de l'UFR STMIA (Sciences et Techniques Mathématiques, Informatique, Automatique) de l'Université H. Poincaré (Licence Sciences de la Production Industrielle, IUP Génie Electrique et Informatique Industrielle, DESS Productique et Automatisation Intégrée) dans le domaine de l'automatisation des systèmes intégrés de production et de leur ingénierie (automatique des S.E.D., informatique industrielle).

Lors de ma nomination en septembre 1998 en tant que Maître de Conférences à ESIAL, je me suis engagé à développer un projet pédagogique autour des Systèmes d'informations industriels. Ceci m'a notamment conduit à accepter en janvier 1999 la **responsabilité technique du projet AIP_ERP**, sous la direction de G. Morel (ISIAL) et G. Roemer (Oracle-Division Est) ; ce projet, mené en collaboration avec l'AIPL et Oracle, avait pour objectif de mettre une plate-forme expérimentale sur les progiciels de gestion intégrée (ERP) à disposition des filières et écoles nancéiennes.

Les résultats de ce projet, qui ont fait l'objet d'une communication orale [GT2], ont mis en évidence la nécessité d'une collaboration, pour la mise en œuvre d'un ERP en milieu industriel, entre des spécialistes de la production et des spécialistes de l'informatique. L'ESIAL proposant historiquement des enseignements dans les domaines de l'informatique et de la productique, j'ai donc contribué, avec l'aide d'un collègue informaticien, à la mise en place d'une nouvelle spécialisation fédérant ces deux disciplines autour des **Systèmes d'Informations d'Entreprises** (SIE). Cette spécialisation, venant compléter l'offre initiale d'ESIAL (3 spécialisations en Ingénierie du Logiciel, Réseau-Télécommunications-Services et Applications Logicielles pour les Systèmes Industriels), a accueilli ses premiers élèves lors de l'année universitaire 2000/2001 et concerne aujourd'hui environ un tiers des élèves de 2ème et 3ème année de l'ESIAL. Dans le cadre de cette spécialisation et sur les bases de l'expérience acquise dans le cadre du projet AIP_ERP, j'ai également proposé un nouveau module (cours, TD et TP) autour des ERP (Enterprise Resource Planning) et des SGDT (Systèmes de Gestion de Données Techniques).

Après avoir assuré la responsabilité de la nouvelle spécialisation SIE pendant deux années, j'ai eu l'opportunité, en 2002/2003, de recentrer mes activités d'enseignement autour de mes thématiques de recherche en acceptant notamment la responsabilité de la spécialisation ALSI « **Applications Logicielles pour les Systèmes Industriels** » (Voir Tableau 6 de synthèse de mes responsabilités d'enseignement). Cette spécialisation forme des ingénieurs spécialisés dans le développement et l'exploitation des Nouvelles Technologies de l'Information et de la Communication dans les systèmes industriels de production.

En formation initiale, mon service, quantitativement développé dans le Tableau 7, comporte des cours magistraux, des travaux dirigés et pratiques, pour la plupart en lien avec ma thématique de recherche, centrés sur le concept de « système intégré de production. Notre objectif à ESIAL est de contribuer à la formation d'ingénieurs ayant des connaissances et compétences dans le domaine de l'ingénierie des systèmes industriels de production afin de pouvoir développer, intégrer et exploiter les applications logicielles présentes dans ces systèmes (Progiciels de gestion intégrée, Manufacturing Execution System, GMAO, SGDT, Supervision industrielle, …). En ce sens, j'ai donc une implication dans les domaines de :

- **l'automatique des Systèmes à Evénements Discrets** : modélisation (automates à états finis, réseaux de Petri, Grafcet, Statecharts, Langage synchrone Signal), ingénierie (analyse fonctionnelle, comportementale, informationnelle, simulation, émulation), technologie (automates programmables industriels, supervision industrielle, M.E.S.)
- **de la gestion intégrée des entreprises** (Gestion des stocks, gestion de production, MRP, introduction aux ERP, Paramétrage de l'ERP ADONIX, Analyse décisionnelle avec la méthode GRAI).

Depuis 2005/2006, dans le cadre de la mise en place du LMD, j'ai contribué à la mise en place de deux Unités d'Enseignement dont j'assure la responsabilité pour le Master Ingénierie Systèmes : en 1ère année, en « Automatique des S.E.D. » (Automates à états, Réseau de Petri, Statecharts, Grafcet, modèles stochastiques) et en 2ème année, en « Systèmes de Planification de la Production » (Analyse des organisations de production à l'aide de la méthode GRAI, paramétrage des ERP).

| Type | Période | Formation | Activités |
|---|---|---|---|
| Unités d'enseignement | Depuis 1999 | ESIAL | **Responsable de modules**<br>- Systèmes réactifs (3ème année)<br>- Progiciels de Gestion Intégrée (3ème année) transformé en Gestion Intégrée des entreprises (2ème année) |
| | Depuis 2002-2003 | | **Responsable de modules**<br>- Gestion de Production (tronc commun 2ème année)<br>- Ingénierie d'automatisation |
| | Depuis 2005-2006 | Master Ingénierie Système | **Responsable de modules**<br>- Automatique des S.E.D. (M1)<br>- Systèmes de Planification de la Production (M2) |
| Stages | De 1999/2000 à 2002/2003 | ESIAL | **Responsable des stages** de 2ème année |
| | Depuis 2007 | | **Responsable des stages** de 3ème année |
| Spécialisations | De 1999/2000 à 2001/2002 | ESIAL | **Responsable pédagogique** de la spécialisation « Systèmes d'Informations d'Entreprises » |
| | Depuis 2002/2003 | ESIAL | **Responsable pédagogique** de la spécialisation « Applications Logicielles pour les Systèmes Industriels » |
| Administration de l'enseignement | Depuis 2004 | ESIAL | **Membre élu au Conseil de l'ESIAL** |
| | | | **Membre nommé au conseil de perfectionnement** de l'ESIAL |
| | Depuis 2006 | ESIAL | **Membre nommé** au groupe de pilotage du projet « Qualité » de l'ESIAL |

Tableau 6. Synthèse des responsabilités d'enseignement

| Public | Discipline | Type | 98/99 | 99/00 | 00/01 | 01/02 | 02/03 | 03/04 | 04/05 | 05/06 | 06/07 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ESIAL 3 | Systèmes réactifs temps réel | CM | 14 | 10 | 10 | 10 | 10 | 10 | 10 | | | 74 |
| | | TD | 28 | | | | 2 | 2 | 4 | | | 36 |
| | | TP | 21 | 40 | 20 | 10 | 10 | 10 | 8 | | | 119 |
| | Progiciels ERP & SGDT | CM | 12 | 12 | 10 | 10 | 10 | 10 | | | | 64 |
| | | TD | 8 | 8 | 4 | 4 | 4 | 6 | | | | 34 |
| | | TP | 48 | 48 | 32 | 32 | 48 | 28 | | | | 236 |
| | Modélisation d'entreprise | CM | | | | | | 4 | | | | 4 |
| | | TD | | | | | | 8 | | | | 8 |
| | Automatisation & MES | CM | | | | | | | 2 | | | 2 |
| | | TD | | | | | | | 2 | | | 2 |
| | | TP | | | | | | | 8 | | | 8 |
| | Jeu d'entreprise | TD | | | | | | | | | 20 | 20 |
| | Projets | TP | 40 | 40 | 32 | 35 | 25 | 25 | 25 | 25 | | 247 |
| ESIAL 2 | Ingénierie d'automatisation | CM | 8 | 6 | 6 | 8 | 8 | 14 | 14 | 14 | 4 | 82 |
| | | TD | | | | 10 | 12 | 12 | 12 | 14 | | 60 |
| | | TP | 60 | 48 | 32 | 48 | 40 | 20 | 40 | 12 | 12 | 312 |
| | Gestion de production | CM | | | | 4 | 4 | 10 | 8 | 2 | 4 | 32 |
| | | TD | | | | 8 | 48 | 20 | 18 | 18 | 18 | 130 |
| | | TP | | | | | | 12 | 16 | 12 | 18 | 58 |
| | Gestion intégrée des entreprises (ERP) | CM | | | | | | 8 | 8 | 8 | 8 | 32 |
| | | TD | | | | | | 20 | 18 | 18 | 12 | 68 |
| | | TP | | | | | | 36 | 20 | 6 | 16 | 78 |
| | Modélisation d'entreprise | CM | | | | 6 | 6 | | | | | 12 |
| | | TD | | | | 10 | | | | | | 10 |
| | | TP | | | | 32 | 20 | | | | | 52 |
| | Systèmes d'informations | CM | | | 8 | | | | | | | 8 |
| | | TD | | | 16 | | | | | | | 16 |
| | | TP | | | 24 | | | | | | | 24 |
| ESIAL 1 | Modèles S.E.D. | TD | | | 24 | 4 | 4 | | | | | 32 |
| | | TP | | | 16 | | 12 | | | | | 28 |
| DESS PAI | Systèmes d'informations | CM | | 3 | | | | 4 | | | | 7 |
| | | TD | | | | | 4 | | | | | 4 |
| | | TP | 12 | 8 | | | | | | | | 20 |
| | Organisation et Gestion des flux en entreprise | CM | | | | | 6 | 4 | 8 | | | 18 |
| | | TD | | | | 6 | | | 8 | | | 14 |
| | | TP | | | | | | | 12 | | | 12 |
| | Ingénierie d'automatisation | CM | 8 | 6 | 8 | | | | | | | 22 |
| | | TD | 16 | 16 | 16 | | | | | | | 48 |
| | Projets | TP | 40 | 31 | 6 | 20 | | | | | | 97 |
| DESS ATTI | Automatique séquentielle | CM | | 4 | 4 | | | | | | | 8 |
| | | TP | 16 | 32 | 32 | | | | | | | 80 |
| Master Ingénierie Système | M2 Systèmes de planification | CM | | | | | | | | 8 | 8 | 16 |
| | | TD | | | | | | | | 16 | 12 | 28 |
| | | TP | | | | | | | | 32 | 32 | 64 |
| | M1 Automatique des SED | CM | | | | | | | | 10 | 14 | 24 |
| | | TD | | | | | | | | 16 | 18 | 34 |
| | | TP | | | | | | | | 32 | 32 | 64 |
| | M2R Ingenierie système formelle | CM | | | | | | | | | 3 | 3 |
| ENSTIB 3 | ERP et MES | CM | | | | | | | 2 | 4 | 4 | 10 |
| | | TD | | | | | | | 4 | | | 4 |
| | | TP | | | | | | | | 8 | 8 | 16 |
| IUP GMP | Automatique des S.E.D. | CM | 4 | 4 | 4 | 4 | | | | | | 16 |
| | | TD | 4 | 4 | | | | | | | | 8 |
| | Micro-thèse | TP | | | | 8 | | | | | | 8 |
| IUP GEII | Automatique séquentielle | TD | 12 | 16 | 12 | | | | | | | 40 |
| | | TP | | | | 12 | | | | | | 12 |
| | Micro-thèse | TP | | | 20 | | | | | | | 20 |
| DEA PA | Modèles S.E.D. | CM | | 4 | 3 | 2 | 3 | 3 | 3 | | | 18 |
| | Ingénierie du CIM | CM | | 4 | 4 | 3 | 6 | 4 | 4 | | | 25 |
| TOTAL eq TD | | | 295 | 288 | 300 | 244 | 257 | 262 | 241 | 236 | 226 | |

Tableau 7. Récapitulatif quantitatif de mes enseignements entre 1998 et 2006

A ces heures conventionnelles d'enseignements, il convient d'ajouter :

- des encadrements de projets proposés en soutien de nos actions de recherche (projets 2A ESIAL, projet DESS PAI, DESS ATTI) ou en collaboration industrielle (projets industriels de 3ème année de l'ESIAL) pour répondre aux objectifs de professionnalisation de ces formations
- des encadrements de projets d'initiation à la recherche (ESIAL 2ème année) ou de micro-thèses (IUP GMP, IUP GEII) à destination d'étudiants représentant un vivier potentiel pour alimenter nos filières de DEA, de Master Recherche et de doctorat. Il est à noter que plusieurs étudiants actuellement en thèse dans notre équipe-projet du CRAN sont issus de la spécialisation ALSI de l'ESIAL.

D'autre part, j'interviens également dans les formations par la recherche de l'UFR STMIA, notamment, de 2001 à 2005, en DEA Production Automatisée autour des modèles formels pour l'automatisation (méthode B, langage synchrone Signal) dans les modules « outils et ingénierie du CIM » et « Modèles comportementaux pour les SED » et depuis 2005/2006 en Master Ingénierie Système, en M2 Recherche, dans l'unité d'Enseignement « Ingénierie Système Formelle ».

Enfin, pour supporter ces enseignements, j'ai contribué au développement de plusieurs projets et travaux pratiques à l'AIP-PRIMECA Lorraine[30]. La section suivante détaille ma contribution depuis 1998 à un ces projets ; le projet e-Production.

## 2. PROJET E-PRODUCTION A L'AIP-PRIMECA LORRAINE

Pour répondre aux évolutions actuelles vers l'entreprise « numérique », nous avons souhaité organiser les enseignements de la spécialisation « **Applications Logicielles pour les Systèmes Industriels** » autour de deux axes :

- un axe système construit sur trois niveaux d'organisation: un niveau planification de la production pour lequel les E.R.P. offrent un ensemble de modules paramétrables couvrant les grandes fonctionnalités d'une entreprise, un niveau commande et supervision des machines et processus de production, et un niveau exécution de la production (M.E.S.) assurant l'interface entre les deux précédents niveaux autour d'un système d'information standardisé (IEC 62264).
- un axe « ingénierie » proposant un ensemble de modèles, de méthodes et d'outils permettant le développement et l'intégration des applications logicielles présentes dans chacun des trois niveaux de l'axe système.

Le projet **e-Production** de l'AIP-PRIMECA Lorraine a pour vocation de mettre à disposition des filières utilisatrices une infrastructure de type entreprise agile (Figure 62). Cohérent avec la structuration souhaitée pour la spécialisation ALSI, je participe donc activement à sa mise en œuvre. Cette participation s'est traduite par différentes actions relatives au déploiement d'un progiciel de gestion intégrée (ERP Adonix) et des outils de modélisation associés (outil d'analyse des organisations de production GraiTools basé sur la méthode GRAI) et au développement d'un système flexible de production contrôlé par le produit.

[30] AIP : Atelier Inter-Etablissement de Productique

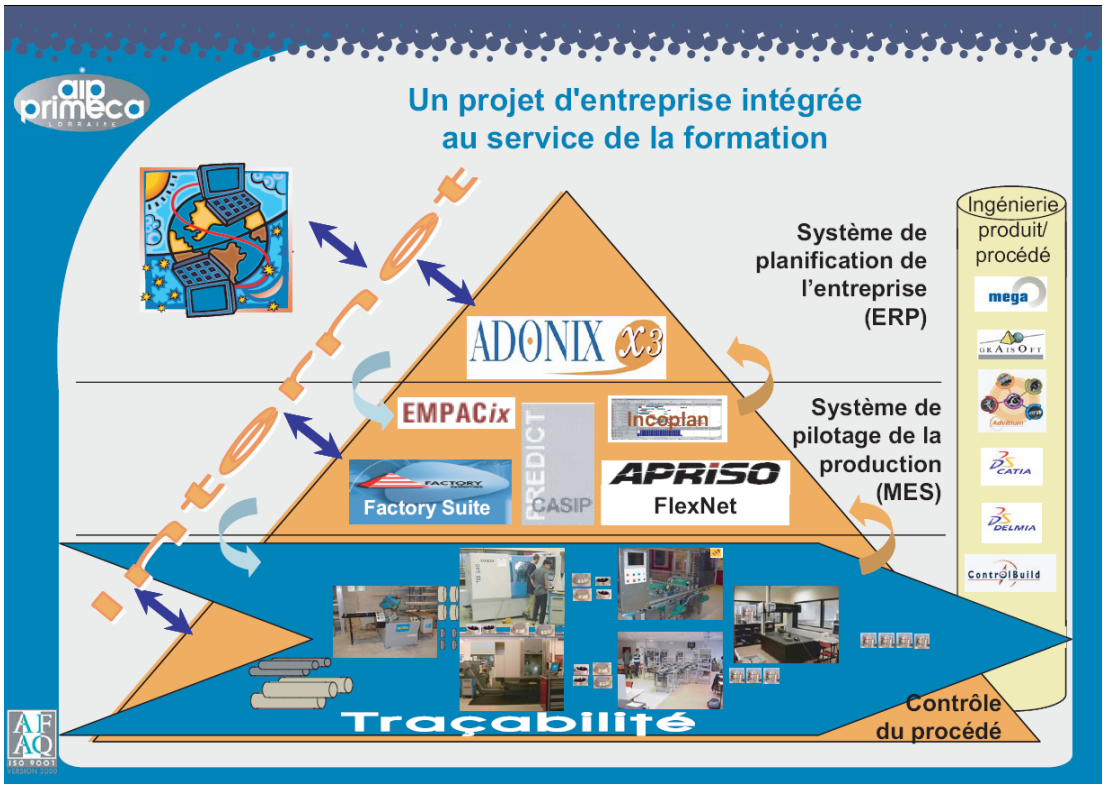



Figure 62. Projet e-Production de l'AIP-Priméca Lorraine

### *2.1 Système de planification de la production - ERP*

Notre première contribution au déploiement d'un progiciel de gestion intégrée à l'AIP-PRIMECA Lorraine remonte en 1998, date de ma nomination sur le poste de maître de conférences à l'ESIAL (Tableau 8). Le **projet AIP_ERP** avait pour mission de mettre à disposition des filières et écoles nancéiennes une plate-forme expérimentale sur les progiciels de gestion intégrée (ERP). Le projet mené en collaboration avec la société ORACLE (ERP Oracle Applications) entre 1998 et 2000 a débouché sur le développement d'une maquette expérimentale présentant un sous-ensemble restreint des fonctionnalités couvertes par les modules d'un progiciel ERP sur un cas d'étude réel suffisamment représentatif d'une problématique industrielle et a permis d'acquérir une expérience conséquente sur la démarche d'ingénierie et de mise en œuvre. Néanmoins, si le projet a atteint ses objectifs en termes de formation des enseignants, divers problèmes techniques et économiques n'ont pas permis de mettre à disposition le produit auprès des étudiants.

| | |
|---|---|
| **Partenaire** | AIPL (JP Drapier, O. Nartz), Oracle (E.Roemer, P. Colmenero), ESIAL (**JF Pétin**), ESSTIN (G. Bloch, L. Lossent), ENSGSI (F. Mayer, P. Truchot), DESS PAI (G. Morel, E. Levrat), DESS ACSI (O. Foucault), IUP GMP (JY Bron) |
| **Activité** | **Responsable** technique du projet<br>**Rédaction** de 3 livrables (Identification des processus cibles, Analyse des écarts, Dossier de paramétrage), |

Tableau 8. Projet AIP_ERP (1998-2000)

Après une phase de réflexion où plusieurs supports ERP ont été analysés, nous avons orienté notre choix en 2002 vers le progiciel ADONIX de taille plus réduite que Oracle Applications mais suffisamment représentatif tant par son positionnement sur le marché national que par les fonctionnalités offertes.

De manière complémentaire, afin de renforcer notre expérience dans les phases d'analyse et de paramétrage des ERP, j'ai participé entre 2004 et 2005 à l'**action collective GAC Lorraine** (Grai Accompagnement Consultants) soutenue par la DRIRE de la région Lorraine. Ce projet avait pour objectif de transférer la méthodologie GRAI vers les consultants et le milieu industriel au travers de cas d'expérimentation impliquant des entreprises lorraines. Le rôle de l'AIP consistait à accompagner les consultants dans leur démarche d'audit basée sur la méthode GRAI. A titre personnel, cet accompagnement (audit dans les entreprises, modélisation avec GRAI) représente un effort d'une vingtaine d'hommes/jour sur les années 2004 et 2005.

| | |
|---|---|
| **Partenaire** | AIPL/UHP (JY Bron, **JF Pétin**, H. Panetto).<br>GRAISOFT ( G. Doumeingts, H. Kromm)<br>JMM consultants (J.M. Macaud), Cosmo Consult (M. Kowalski, G. Coffre), GraiLor (S. Déon, E. Lebohec), Links conseil (Y. Rehby), Fauvel (B. Fauvel) |
| **Activité** | Suivi et accompagnement des consultants<br>Participation à la rédaction des livrables du projet |

Tableau 9. Action collective GAC Lorraine (2004-2005)

L'expérience acquise dans le cadre des projets AIP_ERP et GAC Lorraine m'a permis de proposer un module d'enseignement orienté autour du déploiement de l'ERP Adonix. Cet enseignement, destiné principalement aux élèves en dernière année d'ESIAL ainsi qu'aux étudiants en deuxième année du Master Ingénierie Système de l'UHP, a pour objectif de mettre en pratique, au travers d'un projet d'une trentaine d'heure/étudiant, une démarche partant de la modélisation d'un besoin exprimé par une entreprise jusqu'au paramétrage d'une solution progicielle ERP. Le principe consiste à placer les étudiants dans un rôle de consultant fonctionnel, spécialisé dans le domaine de la gestion industrielle, auquel une entreprise fait appel pour formaliser ses besoins (fournis aux étudiants sous la forme d'interviews), poser un diagnostic sur son organisation, proposer une architecture cible, analyser les écarts entre la cible et la couverture fonctionnelle de l'ERP Adonix, en déduire son paramétrage et le valider en déroulant un scénario adapté. Les trois premières phases sont supportées par la méthode GRAI et son outil support GraiTools. Les trois dernières phases font l'objet d'un livrable.
Le retour d'expérience concernant cet enseignement a été décrit dans une communication au colloque AIP-PRIMECA 2007 **[C25]**.

### *2.2 Système flexible de production - SFP*

Depuis 2003, je suis coordinateur, côté enseignant, des actions et projets relatifs au Système Flexible de Production (SFP) de l'AIP-PRIMECA. Ce système est composé de six postes de travail, permettant la réalisation de produits composés d'un assemblage de 2 à 4 pièces de forme cylindrique, et d'un convoyeur assurant le transport des produits. Ces actions, dont les développements ont été principalement

supportés par les ingénieurs de l'AIP-PRIMECA mais aussi par quelques projets d'étudiants, ont porté sur la modélisation de la commande du SFP à l'aide de l'outil ControlBuild, la modélisation du système physique à des fins d'émulation, le développement de la commande sur l'AGL Siemens STEP7.
Il est à noter que les étiquettes électroniques, dont sont munies les palettes transportant les produits, permettent aux palettes de jouer un rôle actif dans le choix de leur trajectoire sur le SFP en fonction des informations de gammes stockées sur les étiquettes et des capacités des postes. En ce sens, le SFP constitue un support de validation aux travaux de recherche de l'équipe-projet « Systèmes Contrôlés par le Produit » du CRAN, utilisé notamment dans le cadre de la thèse de David Gouyon.

### *2.3 M.E.S.*

Pour compléter l'offre de l'AIP-PRIMECA autour des domaines de l'automatisation et de la planification (ERP) et proposer ainsi un exemple aussi représentatif que possible d'une entreprise intégrée, un projet M.E.S. a été initié en 2006. Piloté par David Gouyon, ce projet porte sur la mise en œuvre de l'outil FlexNet (MES) et d'un ensemble de périphériques (terminaux d'atelier, imprimantes et lecteurs codes barres, système RFID, …) venant compléter les solutions déjà présentes et mises en œuvre à l'AIP-Priméca Lorraine dans les domaines de la maintenance (Empacix, Casip), de l'ordonnancement (Incoplan), de la supervision (Factory Suite). L'objectif est d'assurer la traçabilité des produits depuis la réception en stock des barres d'aluminium jusqu'à l'assemblage des rondelles sur le SFP. Une base de données au format de la norme IEC 62264 a été mise à disposition par l'AIP-PRIMECA pour permettre aux étudiants de se familiariser avec les concepts et objets d'un M.E.S. et pour servir de support à l'échange de données entre applications (ERP, serveur OPC, MES, …) via la technologie XML et B2MML.

## 3. PROJET INGENIERIE FORMELLE DES SYSTEMES

Dans le cadre du master Ingénierie Système co-habilité par l'Université H. Poincaré et l'Institut National Polytechnique de Lorraine depuis 2005/2006, et dans le cadre d'une nouvelle spécialisation de l'ESIAL intitulée « Logiciels Embarqués », nous développons actuellement en collaboration avec le Pr. D/ Méry (LORIA) et le Pr. G. Morel (CRAN), un enseignement centré autour de l'Ingénierie Formelle des Systèmes.

Cet enseignement doit permettre aux étudiants d'acquérir les bases théoriques nécessaires pour appréhender scientifiquement l'analyse, la modélisation et le développement de systèmes, en particulier lorsque ceux-ci comportent une part prépondérante de logiciels soumise à de fortes contraintes de sécurité et de sûreté de fonctionnement. Il sera basé sur des notions essentielles pour ce type d'applications telles que le raffinement de modèles et la vérification de propriétés, sur des langages et formalismes reconnus tels que UML, SysML, xtUML, la méthode B ou les langages synchrones et sur les outils associés (MagicDraw, Atelier B, SCADE, Matlab/simulink, …) ainsi que sur les normes en vigueur (IEC61508, DO 178B, …) et les procédures de certification associées.

# VII Conclusion Générale

Nos activités de recherche sur la formalisation d'un processus sûr d'automatisation, menées au sein du CRAN et présentées dans ce mémoire, ont été à l'origine initiées par les applications développées dans le cadre de nos collaborations industrielles, en particulier dans le cadre des programmes européens en Actionnement et Mesure Intelligents, avec pour objectif de rationaliser les résultats de ces applications. En effet, la recherche technologique développée dans ces projets, suffisante pour converger vers la proposition de solutions d'automatismes innovantes, ne répond que partiellement aux exigences scientifiques de vérification des modèles requises dans un processus sûr de développement. Pour répondre à ces préoccupations, nous nous sommes orientés vers l'utilisation de modèles et méthodes formelles relevant des Systèmes à Evénements Discrets, notamment dans le cadre de la théorie de la supervision de Ramadge et Wonham, et plus largement du Génie Informatique, pour couvrir non seulement la dimension comportementale d'une automatisation mais également ses composantes informationnelles et structurelles, en particulier dans le cadre de son intégration aux systèmes d'informations d'atelier.
Notre positionnement scientifique est donc très clairement situé au carrefour entre les domaines de la modélisation des Systèmes à Evénements Discrets et de l'Ingénierie Système avec une volonté d'ancrer nos recherches dans un contexte industriel fort. Ce positionnement se traduit par une implication dans les groupes de travail INCOS du GdR MACS et Sûreté de Fonctionnement en Ingénierie Système de l'AFIS mais également par le développement de collaborations industrielles, notamment dans le cadre de conventions CIFRE. Les résultats obtenus depuis 1996 nous ont permis d'acquérir une reconnaissance locale (coordination de l'équipe/projet MMF du CRAN entre 2000 et 2003, coordination du centre SafeTech depuis avril 2007, membre du conseil des opérations de l'axe 3S du CPER MISN, …) nationale (coordinateur de l'Action Spécifique n°198, RTP PCM 47 du CNRS/DSTIC) et internationale (membre nommé par la SEE du TC 5.1. de lFAC Manufacturing Plant Control).
C'est dans ce contexte scientifique très prenant et motivant, en y associant une composante relationnelle forte aussi bien vis à vis du monde académique qu'industriel et estudiantin, national et international, que nous souhaitons poursuivre nos activités à la fois de recherche mais aussi d'enseignement et d'administration par un transfert permanent de nos connaissances et compétences.

D'un point de vue purement quantitatif, notre travail a fait l'objet d'une quarantaine de publications dont 11 dans des revues (internationales ou nationales) et 2 dans des ouvrages. Certaines de ces publications ont été coécrites avec des industriels et des universitaires extérieurs au CRAN, attestant notre ouverture et notre reconnaissance de la communauté industrielle et scientifique. Nous avons participé à une quinzaine de projets ou contrats dont 7 dans un cadre Européen. Nous avons co-encadré 5 étudiants en thèse (dont 2 en cours devant soutenir en 2007/2008 et 1 ayant débuté en septembre 2007), 6 DEA, comme l'attestent les publications communes et notre participation aux jurys. Nous avons aussi participé à l'organisation de deux manifestations internationales et co-organisé 2 sessions ou tracks dans des conférences internationales de haut niveau. Nous avons enfin exercé plusieurs activités de critique scientifique pour des revues ou des colloques.

# VIII Production scientifique

## 1. REVUES AVEC COMITE DE LECTURE

- Internationales (7)

**R1** **J.F. Pétin**, B. Iung, G. Morel (1998). Distributed Intelligent Actuation and Measurement system within an integrated shop-floor organisation, *Computers In Industry Journal*, *Special issue on Intelligent Manufacturing System, VOL 37*, pp 197-211, Elsevier Sciences Publisher, ISSN 0166-3615, 1998

**R2** D. Gouyon, **J.F. Pétin**, A. Gouin (2004). A pragmatic approach for modular control synthesis and implementation, *International Journal of Production Research*, vol. 42, n° 14, pp. 2839-2858, Taylor & Francis Publisher, ISSN 0020-7543, Jul 2004.

**R3** H. Panetto, **J.F. Pétin** (2005). Metamodelling of production systems process models using UML stereotypes, *International Journal of Internet and Enterprise Management*, Vol. 3/2, 155-169, Inderscience Publisher, ISSN: 1476-1300

**R4** **J.F. Pétin**, G. Morel, H. Panetto (2006). Formal specification method for production systems automation. *European Journal of Control*, Volume 12/2, 115-130, Hermès Science Publishers, March/April 2006, ISBN 2-7462-1480-6

**R5** **J.F. Pétin**, D. Gouyon, G. Morel (2007) Supervisory synthesis for product-driven automation and its application to a flexible assembly cell, *IFAC Control Engineering Practice journal*, Volume 15, Issue 5, May 2007, Elsevier Science Publisher, ISSN 0967-0661

**R6** D. Gouyon, **J.F. Pétin**, G. Morel (2007). A product-driven reconfigurable control for shop floor systems. *Studies in Informatics and Control*, Volume 16, n°1, March 2007, National Institute for R&D in Informatics Publisher, ISSN: 1220-1766.

**R7** H. El Haouzi, A. Thomas, **J.F. Pétin** (2007). Contribution to reusability and modularity of Manufacturing Systems Simulation Models: application to distributed control simulation within DFT context. *International Journal of Production Economics*, Elsevier Science Publisher, à paraître en 2007, doi:10.1016/j.ijpe.2006.12.067, ISSN: 0925-5273

- Nationales (4)

**R8** **J.F. Pétin**, B. Iung, G. Morel, P. Lhoste (1992). Expérimentation Industrielle du concept ESPRIT-DIAS, *Revue d'Automatique et de Productique Appliquées*, Volume 5, n°4, pp 9-25, ISSN 0990-7009, Editions Hermes, 1992

**R9** **J.F. Pétin**, G. Morel, B. Iung, P. Lhoste (1993). Contribution to an industrial strategy of factory of future, *Automatics, Scientific Bulletin 64,* n° 1546, pp 205-220, University of Mining and Metallurgy of Cracow (Poland), ISSN 0454-4773,1993.

**R10** **J.F. Pétin**, B. Iung, E. Neunreuther, G. Morel (1996). Contribution méthodologique à l'Actionnement et la Mesure Intelligents, *Journal Européen des Systèmes Automatisés*, Volume 30, n°6, pp 897-918, Hermès Science Publishers, ISSN 0296-1598, 1996

**R11** D. Gouyon, **J.F. Pétin**, A. Gouin (2003). Application de techniques de synthèse en ingénierie d'automatisation, *Revue électronique SEE Sciences et technologies de l'Automatique*, Volume 0, 2003.

## 2. PARTICIPATION À DES OUVRAGES

**O1** B. Iung, P. Lhoste **J.F. Pétin**, G. MOREL. (1991). *Spécification, conception, réalisation et expérimentation d'un actionneur intelligent*, Capteurs intelligents et micro-actionneurs intégrés, pp 101-102, Ed. CEPADUES, 1992.

**O2** E. Zamai, F. Rigaud, **J.F. Pétin**, P. Berruet, A. Toguyeni (2007)., *Principes des architectures de pilotage de procédés industriels*, Chapitre de « Conduite des systèmes industriels », Techniques de l'Ingénieur, Juillet 2007

**O3** T. Johnson, G. Morel, **J.F. Pétin** (2007). Reliability, maintenance, and safety, *Chapter 5.3 of Springer Handbook of Automation*, Springer Verlag, à paraître en 2007.

## 3. CONFERENCES AVEC COMITE DE LECTURE ET ACTES

- Internationaux (22)

**C1** B. Iung, **J.F. Pétin**, G. Morel (1993). Development and integration of intelligent actuator in a manufacturing system, in *Int. Conf. on Advanced Mechatronics ICAM'93*, pp 191-196, Tokio, Japan, 2-4/08/1993.

**C2** H. Panetto, J.M. Rivière, **J.F. Pétin** (1993). Towards a unified approach for Intelligent Actuators and Sensors, in *IEEE/SMC International Conference on Systems, Man and Cybernetics, Systems Engineering in the service of humans*, pp 177-182, Le Touquet, France, Oct. 17-20, 1993.

**C3** G. Morel, P. Lhoste, B. Iung, **J.F. Pétin**, F. Corbier, O. Douchin (1993). Discrete Event Automation Engineering: outline of the PRIAM project, conférence invitée, *International Conference Automation 1993*, *25th BIAS, WS 2*, pp 1105-1116, Milan, Italy, 23-25/11/93.

**C4** H. Panetto, P. Lhoste, **J.F. Pétin**, E. Bon (1994). Synchrony in Discrete Events Systems Modelling, in *20th Annual Conference of the IEEE Industrial Electronics Society on Industrial Electronics Control and Instrumentation, IECON'94*, Vol 3, pp 1527-1532, Bologne, Italie, September 5-9, 1994.

**C5** G. Morel, F. Mayer, **J.F. Pétin**, P. Lhoste (1995). System-oriented Education Engineering, Proceedings of the *11th ISPE/IEE/IFAC International Conference CARS & FOFS on CAD/CAM Robotics and Factory of Future*, pp 1126-1133, Pereira, Colombie, 28-30/08/1995, ISBN 95782-2-5.

**C6** **J.F. Pétin**, D. Méry, H. Panetto, B. Iung (1996). Validation of software components for Intelligent Actuation and Measurement, Proceedings of the *6th International Symposium on Robotics And Manufacturing (ISRAM'96), World Automation Congress*, Volume 3, pp 631-637, ISBN 1-889335-00-2, Montpellier, 27-30/05/1996.

**C7** **J.F. Pétin**, B. Iung, G. Morel (1996). From Intelligent Actuators to Intelligent Actuation: outline of DIAS, PRIAM and EIAMUG Projects, Proceedings of the *5th International Conf. on new actuators*, *ACTUATORS '96*, Axon Technologie Consult GMbH, pp 462-465, Bremen, Germany, 26-28/06/1996.

**C8** D. Méry, **J.F. Pétin** (1998). Formal engineering methods for modelling and verification of control systems, in *9th IFAC symposium on Information Control in Manufacturing*, Volume 1, pp 141-146, Nancy, June 24-26, 1998, ISBN 0-08-042928-9.

**C9** **J.F. Pétin**, G. Morel, D. Méry, P. Lamboley (1998). Process control engineering : contribution to a formal structuring framework with the B method, *Lecture Notes in Computer Science, Vol 1393*, Proceeedings of the B98 Conference "Recent advances in the development and use of the B method", pp 198-209, Didier Bert Editor, Springer-Verlag Publisher, ISSN 0302-9743, 1998

**C10** P. Lamboley, **J.F. Pétin**, D. Méry (1999). Towards a formal engineering framework for process automation, in *7th IEEE International Conference on Emerging Technologies and Factory Automation*, Vol. 2, pp 1167-1175, 18-22/10/1999, Elsevier, ISBN 0-7803-5670-5.

**C11** **J.F. Pétin**, P. Lamboley (2000). Formal design patterns for production systems engineering, in *Advanced Summer Institute, ASI 2000*, pp 156-163, Bordeaux, France, 18-20/09/2000.

**C12** G. Morel, **J.F. Pétin**, P. Lamboley (2001). Formal specification for manufacturing systems automation, in *10th IFAC symposium on Information Control problems in Manufacturing, INCOM'01,* Vienne, 20-22/09/2001.

**C13** D. Gouyon, A. Gouin, **J.F. Pétin** (2002). Application de techniques de synthèse en ingénierie d'automatisation, in *Conférence Internationale Francophone d'Automatique, CIFA 2002*, pp 62-67, Nantes, 8-10/07/2002.

**C14** H. Panetto, **J.F. Pétin**, D. Méry (2002). Formalisation of enterprise modelling standards using UML and the B method, in proceedings of the *8th International Conference on Concurrent Enterprising, ICE2002*, 17-19/06/2002, Rome, Italy, pp. 93-101, ISBN : 0-85358-113-4

**C15** D. Gouyon, **J.F. Pétin**, A. Gouin, "Modèles du procédé et de ses spécifications pour la synthèse de la commande", in *4ème colloque francophone sur la Modélisation des Systèmes Réactifs (MSR)*, pp 45-60, Metz,6-8/10/2003, ISBN 2-7462-0778-8

**C16** H. Panetto, **J.F. Pétin** (2003). Setting up UML stereotypes for Production systems modelling, in *10th ISPE International Conference on Concurrent Engineering Research and Application, Enhanced Interoperable Systems,* pp 747-754, Madeira - Portugal, 26-30/07/2003.

**C17** D. Gouyon, **J.F. Pétin**, G. Morel (2004). Control Synthesis For Product-Driven Automation, Proceedings of *7th IFAC Workshop on Discrete Event Systems*, WODES'04, pp 19-24, Reims, 22-24 septembre 2004.

**C18** **J.F. Pétin**, T. Klein, G. Morel (2005). Function block model for dynamic reconfiguration of Discrete Event Systems, proceedings of the *17th IMACS World Congress - Scientific Computation, Applied Mathematics and Simulation*, Paris, 11-15 juillet 2005.

**C19** **J.F. Pétin**, P. Berruet, A. Toguyeni, E. Zamai (2005). Impact of Information and Communication emerging technologies in Automation engineering: outline of the INTICA project, proceedings of the *1st NeCST Workshop on Networked Control Systems and Fault Tolerant Control,* EC/IST*,* pp 51-57, Ajaccio, France, 6-7 October 2005

**C20** D. Evrot, **J.F. Pétin**, D. Méry (2006). Formal specification of safe manufacturing machines using the B method: application to a mechanical press, to be published in the proceedings of *12th IFAC symposium on Information Control problems in Manufacturing,* St-Etienne, France, 17-19 May 2006.

**C21** H. El Haouzi, A. Thomas, **J.F. Pétin** (2007). Auto-ID implementation based on six sigma methodology and discrete event simulation, *International Conference on Industrial Engineering and Systems Management (IESM'07)*, May 30 - June 2 2007, Beijing, CHINA.

**C22** D. Evrot, **J.F. Pétin**, G. Morel, P. Lamy (2007). Using SysML for identification and refinement of machinery safety properties, *1st IFAC Workshop on Dependable Control of Discrete-event Systems*, Cachan, France, 13-15 juin 2007.

- Nationaux (3)

**C23** **J.F. Pétin**, G. Morel (1999). Ingénierie formelle des systèmes automatisés de production, Conférence invitée aux Journées Doctorales d'Automatique, JDA'99, Nancy, France 21-23 septembre 1999.

**C24** D. Evrot, P. Lamy, **J.F. Pétin** (2006), Conception et validation de systèmes en sécurité machine : un comparatif de méthodes et d'outils informels, 15ème colloque sur la maîtrise des risques et la sûreté de fonctionnement (Lamda-mu), 10-12 octobre 2006, Lille.

**C25** **J.F. Pétin**, J.Y. Bron, B. Zoz, H. Panetto, F. Mayer (2007). Gestion industrielle et pédagogie : retours d'expériences avec l'ERP Adonix, 10ème Colloque national AIP-PRIMECA, 18-20 avril 2007, La Plagne.

# 4. MANIFESTATIONS AVEC OU SANS COMITE DE LECTURE ET A DIFFUSION RESTREINTE

- Internationales invitées (1)

**CDR1** **J.F. Pétin** (1997). Conformance analysis of the CMM methodological approach to the IAM-Pilot", *1st Workshop on Intelligent Actuation and Measurement*, (IAM-Pilot E.P. 23525), Milan, Octobre 1997

- Nationales (2)

**CDR2** **J.F. Pétin**, B. Iung, G. Morel (1997). Spécification d'un outil de modélisation des systèmes d'Actionnement et de Mesure Intelligents: l'outil PRIAM, Actes des *Journées d'Etude sur les Logiciels pour le traitement de l'Image, du Signal et pour l'Automatique (ELISA'97)*, Nancy, 25-26 Mars 1997

**CDR3** J.P. Muller, **J.F. Pétin**, G. Morel, B. Vachon, C. Pegard, E. Brassart, N. Hutin, S. Dembele, A. Janex, B. Morello, J.C. Ravassard, A. Bourjault (1998). Conception de systèmes de transport collectif d'objets*,* Actes des *Premières Journées du Pôle Microrobotique et Microsystèmes*, *Axe prioritaire Machines Intelligentes du Département SPI du CNRS*, pp 61-66, Besançon, 13-14 janvier 1998.

- Communications à des groupes de travail nationaux (9)

**GT1** **J.F. Pétin** (1994). *Actionnement et Mesure Intelligents: P.R.I.A.M. (Prenormative Requirement for Intelligent Actuation and Measurement) et E.I.A.M.U.G. (European Intelligent Actuation and Measurement User Group)***,** Communication au groupe "Interopérabilité" du C.I.A.M.E., Paris, 18/10/1994

**GT2** **J.F. Pétin** (1999). *Progiciels de Gestion Intégrée (ERP), Enjeux pour l'enseignement et la recherche en Productique*, Conférence plénière des journées du GRP (Groupement de Recherche en Productique) Nancy, 26/11/1999

**GT3** **J.F. Pétin** (1999). *Applications de la méthode B en ingénierie formelle des systèmes automatisés de production*, Groupe de travail COMPIL du Groupement de Recherche en Productique, Nancy 26 novembre 1999

**GT4** **J.F. Pétin**, L. Piétrac (2000). *Applications des méthodes Z et B en ingénierie formelle des systèmes automatisés de production*, Conférence plénière des journées du GRP (Groupement de Recherche en Productique), Annecy, 13 mars 2000

**GT5** **J.F. Pétin**, D. Gouyon (2001). *Modélisation de partie opérative pour l'analyse et la synthèse de la commande*, Groupe de travail COSED, GDR Automatique, 14 décembre 2001

**GT6** D. Gouyon, A. Gouin, **J.F. Pétin**. *Application de techniques de synthèse en ingénierie d'automatisation*. Groupe de Travail COSED, GDR Automatique, 15 Mars 2002.

**GT7** **J.F. Pétin**, D. Gouyon, A. Gouin (2002). *Sensibilité de la méthode de synthèse vis à vis des spécifications : une étude de cas*, Groupe de travail COSED, GDR Automatique, 12 décembre 2002

**GT8** N. Boudjlida, H. Panetto, K. Benali, **J.F. Pétin** (2002). *Vers un modèle unifié pour la modélisation en entreprise*, Assises Nationales du GDR I3 (Information, Intéraction, Intelligence), Nancy, 6 décembre 2002.

**GT9** **J.F. Pétin**, B. Denis (2003). *AS 198: Impact des NTIC en Automatisation, Groupe de travail INCOS,* Journées du GDR MACS/STP, Bordeaux 16-17/10/2003

# 5. MEMOIRES

**M1** **J.F. Pétin** (1991). Expérimentation industrielle du concept de Contrôle, Maintenance et Gestion technique intégrés: modèles de référence**,** DEA Production Automatisée à l'Université Henri Poincaré NANCY I, juin 1991.

**M2** **J.F. Pétin** (1995). Contribution méthodologique à l'actionnement et la mesure intelligents: application au projet esprit III – P.R.I.A.M. N°6188. Doctorat de l'Université Henri Poincaré, en production automatisée, décembre 1995.

# IX Formation par la recherche

## 1. CODIRECTION DE THESES

**TH1** **Patrick Lamboley**
*Proposition d'une méthode formelle d'automatisation de systèmes de production à l'aide de la méthode B,* Doctorat en Production Automatisée de l'Université H. Poincaré- Nancy I,
Soutenue le : 18 septembre 2001
Directeur de thèse : Pr. G. Morel (50%)
Responsable de Recherche : **J.F. Pétin** (50%)
Jury : D. Méry (Président), A. Haurat, J Zaytoon (Rapporteurs), M. Zaremba, J.F. Pétin (examinateurs), G. Morel (Directeur de thèse).
Sujet : Les travaux de thèse de Patrick Lamboley ont contribué à la proposition d'une méthode formelle d'automatisation reposant sur une utilisation particulière des mécanismes de raffinement et de preuves de propriétés supportés par la méthode B pour garantir la correction, la complétude et la cohérence d'une spécification des services attendus d'un système automatisé par les différents acteurs impliqués dans son développement.
Financement : Bourse MENESR
Co-publications : 4 communications [C9][C10][C11][C12] et de rapports de contrat industriel [Rp9][Rp10]
Situation actuelle : Responsable marketing « Automates » (Schneider)

**TH2** **David Gouyon**
*Contrôle par le produit des systèmes d'exécution de la production : apport des techniques de synthèse*, Doctorat en Production Automatisée de l'Université H. Poincaré- Nancy I,
Soutenue le : 6 décembre 2004
Directeur de thèse : Pr. G. Morel (50%)
Co-directeur de thèse : **J.F. Pétin** (50%, autorisation locale à co-diriger une thèse)
Jury : J.J. Lesage (Président), V. Carre-Ménetrier, J.P. Bourey (Rapporteurs), A. Richard (examinateur), G. Morel (Directeur de thèse), J.F. Pétin (Co-directeur de thèse, autorisation du Conseil scientifique de l'UHP).
Sujet : Les travaux de thèse de David Gouyon ont contribué à formalisation d'un cadre de modélisation du contrôle par le produit des systèmes d'exécution de la production basé sur les techniques de synthèse dans le cadre de la théorie de la commande par supervision de Ramadge & Wonham. L'originalité de ces travaux consiste à placer le produit au centre de la démarche d'automatisation en assurant l'interopérabilité entre le contrôle par le produit de son routage à travers le système et le contrôle des ressources du système de fabrication.
Financement : Bourse MENESR
Co-publications : 3 revues [R2][R5][R6], 3 communications [C13][C15][C17] et 3 communications à des groupes de travail [GT5][GT6][GT7]
Situation actuelle : Maître de conférences à l'Université H. Poincaré

**TH3** **Dominique Evrot**
*Définition d'un processus de développement de Systèmes Automatisés Sûrs pour l'Industrie Manufacturière : Application aux Automates Programmable de Sécurité* (encadrement : 50%), Doctorat de l'Université Henri Poincaré Nancy I
Soutenue le : thèse en cours débutée en janvier 2005, soutenance prévue en janvier 2008.
Directeur de thèse : Pr. G. Morel (50%)
Co-directeur de thèse : **J.F. Pétin** (50%, autorisation locale à co-diriger une thèse)
Sujet : Les travaux de thèse de Dominique EVROT, porte sur la définition d'un processus de développement des applications logicielles de commande des machines industrielles qui incluent la gestion de la sécurité des opérateurs. L'approche mise en oeuvre combine des modèles non formels tel que SysML à des modèles formels (méthode B, model checking) pour couvrir les diverses phases classiques d'une automatisation (spécification, conception, implantation) en s'efforçant de montrer comment les diverses exigences de sécurité doivent être formalisées, validées et/ou vérifiées, propagées et suivies (traçabilité).
Financement : convention CIFRE avec l'INRS (Institut National de Recherche et de Sécurité)
Co-publications : 2 communications [C20][C24].

**TH4 Hind El Haouzi**

*Processus d'intégration de nouvelles technologies de type RFID pour la décision distribuée et la synchronisation des flux physiques et informationnel*, Doctorat de l'Université Henri Poincaré Nancy I

Soutenue le : thèse en cours débutée en septembre 2005 (soutenance prévue au 1er trimestre 2008)

Directeur de thèse : Pr. A. Thomas (50%)

Co-directeur de thèse : **J.F. Pétin** (50%, autorisation locale à co-diriger une thèse)

Sujet : Les travaux de thèse de Hind EL HAOUZI ont pour objectif d'évaluer l'intérêt d'une architecture de pilotage de la production par le produit en plaçant celui-ci au centre de la démarche d'automatisation. Le travail devra notamment déboucher sur la proposition de modèles et d'outils permettant la simulation de ce type d'architecture en intégrant des représentations temporelles différentes (modèles événementiels, à temps continu, à temps discrets) pour la modélisation des processus de décision, des systèmes de commande et des systèmes physiques.

Financement : convention CIFRE avec la société TRANE

Co-publications : 1revue [R7] et 1 communication [C21]

**TH5 Gilbert Habib**

*Conception d'architectures de commande distribuées sûres de fonctionnement : intégration des systèmes de communication sans fil,* Thèse de l'Université H. Poincaré,

Soutenue le : thèse en cours débutée le 1er septembre 2007.

Directeur de thèse : Pr. T. Divoux (50%)

Co-directeur de thèse : **J.F. Pétin** (50%, autorisation locale à co-diriger une thèse en cours)

Sujet : Les nouvelles technologies de l'information et de la communication conduisent aujourd'hui à des architectures de commande distribuées basées sur la coopération d'objets logiciels dans lesquelles la communication joue un rôle prépondérant. Ce travail a pour objectif de prendre à compte cette dimension dans la modélisation des applications de commande et son impact sur la sûreté de fonctionnement.

Financement : Bourse MENESR

## 2. CODIRECTION DE STAGIAIRES DE DEA

**DEA1 Jean-Christophe Blaise**

*Méta-Modélisation de textes de normes sécuritaires*, DEA de Production Automatisée, Henri Poincaré-Nancy I, 1996,

Directeur de Recherche : P. Lhoste (50%),

Responsable de Recherche : **J.F. Pétin** (50%)

**DEA2 Patrick Lamboley**

Spécifications Formelles d'un Système d'Actionnement et de Mesure Intelligents à l'aide de la méthode B, DEA de Production Automatisée, Université H. Poincaré, mention Bien, 1997

Directeur de Recherche: G. Morel (50%),

Responsable de Recherche: **J.F. Pétin** (50%).

**DEA3 Julien Beaudinet**

Implantation d'une spécification formelle B: application au système de conduite d'une plate-forme expérimentale en Actionnement et Mesure Intelligents, D.E.A. Production Automatisée, Université Henri Poincaré, Nancy 1, 1999

Directeur de Recherche: G. Morel (50%)

Responsable de Recherche: **J.F. Pétin** (50%)

**DEA4 David Gouyon**

Application de techniques de synthèse de la commande en ingénierie d'automatisation, D.E.A. Production Automatisée, Université Henri Poincaré, Nancy 1, 2001,

Directeur de Recherche: G. Morel (50%),

Responsable de Recherche: **J.F. Pétin** (50%)

**DEA5** **Cyrille Limousin**
Vérification formelle des spécifications d'une commande machine incluant des contraintes sécuritaires, D.E.A. Production Automatisée, Université Henri Poincaré, Nancy 1, 2005,
Directeur de Recherche: P. Lhoste (50%),
Responsable de Recherche: **J.F. Pétin** (50%)

**DEA6** **Thomas Klein**
Reconfiguration en ligne des architectures de commande : applications en e-control, D.E.A. Production Automatisée, Université Henri Poincaré, Nancy 1, 2005,
Directeur de Recherche: G. Morel (0%),
Responsable de Recherche: **J.F. Pétin** (100%)

# X Valorisation

## 1. PROJETS EUROPEENS ET CONTRATS DE RECHERCHE

- Projets Européens et Internationaux avec comité d'évaluation et rapports ou communications (6)

**P1** **ESPRIT II-D.I.A.S. n°2172 Distributed Intelligent Actuators and Sensors**
Durée : 36 mois (1989 – 1992)
Partenaires *:* EDF (F), EDP (P), ENEL (I), MONTEFIBRE (I); SEMA-GROUP (F, B), HARTMANN & BRAUN (D), BAILEY ESACONTROL (I), MENTEC (IRL), EFACEC (P), I.S.T. (P), Sous-traitant EDF : CRAN (F)
Responsable CRAN : G. Morel
Participants du CRAN : B. Iung, **J.F. Pétin**
Contribution : Développement et expérimentation de trois prototypes d'actionneurs intelligents sur le site industriel de la centrale thermique EDF du Havre.

**P2** **ESPRIT III-PRIAM n°6188 Prenormative Requirements for Intelligent Actuation and Measurement**
Durée : 36 mois (1992 – 1995)
Partenaires : EDF (F), ELF (F),EDP (P),ENEL (I),LABORELEC (B), MONTEFIBRE (I), SEMA-GROUP (F,B), T.N.I. (F), HARTMANN & BRAUN (D), ELSAG BAILEY (I), BIFFI (I), BAILEY SEREG (I) ET ETS BERNARD (F), CRAN (F)
Responsable CRAN : G. Morel, B. Iung
Participants du CRAN : **J.F. Pétin**
Contribution : Proposition d'une approche méthodologique pour la modélisation des systèmes d'Actionnement et de Mesure Intelligents et d'un outil informatique dédié (WorkPackage 6).

**P3** **ESPRIT IV- REMAFEX n°20874** REmote Maintenance for Facility EXploitation
Durée : 18 mois (1997 – 1998)
Partenaires : EDP(P), IBERDROLA(SP), EFI(N), SEMA GROUP(F), CCIL-AEROPORT DE SATOLAS(F), CRAN (F)
Responsable CRAN : G. Morel, B. Iung
Participants du CRAN : J.B. Leger, F. Mayer, **J.F. Pétin**
Contribution : formalisation d'une approche méthodologique permettant d'intégrer la Maintenance au Contrôle et à la Gestion Technique. Ma contribution porte plus particulièrement sur la validation et la vérification formelle du fonctionnement d'un tel système intégré.

**P4** **ESPRIT IV- IAM Pilot n°23525** Intelligent Actuation and Measurement Pilot
Durée : 18 mois (1997 – 1998)
Partenaires : ENEL (I), SEMA-GROUP (F,B), CISE (I), IFAK (D), APAX (GB), CRAN (F)
Responsable CRAN : **J.F. Pétin**
Participants du CRAN : G. Morel, E. Neunreuther
Contribution : assurer la cohérence méthodologique du développement du démonstrateur technologique en Actionnement et Mesure Intelligents.

**Rp1** F. Russo, P. Culot, C. Diedrich, **J.F. Pétin**, M. Macachek, *Periodic Progress Reports*, Octobre 97 et Avril 98

**Rp2** **J.F. Pétin**, E. Neunreuther, G. Morel (1998). Analysis of conformance to IAM models, *Deliverable D.2,* October 16, 1998, 37 pages.

**P5** **ESPRIT IV - I.M.S. n°21955 Working group in Intelligent Manufacturing System**
Durée : 36 mois (1997 – 2000)
Coordinateurs : Pr H. Van Brussel, Dr P. Valckenaers, Katholieke Universiteit Leuven (Belgique)
Responsable CRAN : B. Iung
Participants du CRAN : B. Iung, G. Morel, **J.F. Pétin**
Partenaires*:* UNIV LEUVEN KATHOLIEKE (B), CRAN (F), E.P.F.L. (S), ISEP/IPP (P), FHG/IPK (D), UNIV STOCKHOLM (SW), SINTEF (N), UNIV EINDHOVEN (NL), UNIV KARLSRUHE (D), UNIV EDINBURGH (UK), UNIV PATRAS (GR), UNIV WESTMINSTER (UK), ABB (S), ALCATEL FINCO (B), ALFAMICRO (P), AMT (IRL), BICC (UK), DAIMLER-BENTZ (D), UNILEVER (NL), VTT AUTOMATION (SF).

**P6** **UEML IST–2001–34229 Thematic Network *(Unified Enterprise Modelling Language)***
Durée: 15 mois (2002-2003)
Coordinateur **:** COMPUTAS (NO)
Partenaires : INRIA (CRAN et LORIA) (FR), UB1 (FR), GRAISOFT (FR), UoT (IT), UoN (BE), CIMOSA (DE), IPK (DE), UPV (ES)
Responsable CRAN : H. Panetto
Participants du CRAN : H. Panetto, **J.F. Pétin**

**Rp3** M. Petit, H. Panetto, **J.F. Pétin**, K. Benali, N. Boudjlida, G. Berio et al., *Deliverable 1.1: State of the Art in Enterprise Modelling*, Septembre 2002

**Rp4** T. Knothe, **J.F. Pétin**, K. Benali et al., *Deliverable 2.1: Initial sets of requirements*, Novembre 2002

- Projet national avec comité d'évaluation et rapports ou communications (3)

**P7** **Projet Micro-robotique collective, Pôle Micro-robotique et Micro-systèmes, Département SPI du CNRS,**
Coordinateur : A. Bourjault (LAB)
Partenaire : IIUN (Neuchâtel), CRAN, CRI, GRACSY, LAB-IMFC, LMS, LAI, LMARC-IMFC, LAAS, LPMO-IMFC
Participants du CRAN: G. Morel, **J.F. Pétin**
Contribution : conception de systèmes de transport collectif d'objets basée sur la spécification du comportement collectif à partir des interactions des robots entre eux et avec l'environnement sous la forme de structures spatio-temporelles.

Communication [CDR3] aux journées du pôle à Besançon, 13-14 janvier 1998

**P8** **Action Spécifique INTICA (Impact des NTIC en automatisation), CNRS/STIC n°198, RTP PCM 47**
Durée : 12 mois (2003/2004)
Coordinateur : **J.F Pétin** (CRAN), B. Denis (LURPA)
Partenaires : CRAN, LURPA, LAG, LAGIS, LGEF/INSA Lyon, LESTER, CReSTIC, LIMI
Participants du CRAN: **J.F. Pétin**, G. Morel
Contribution : sûreté de fonctionnement de systèmes de commande distribués autour de réseaux non déterministes et intégration de la commande au sein des architectures de pilotage et de conduite des systèmes de production (M.E.S., produits intelligents, …)

Organisation d'une journée d'étude le 27 janvier 2005 à Paris

Communication [**C19]** aux Workshop du projet européen NeCST (Networked Control System & Fault Tolerant Control)

**P9** **Action RECSED soutenue par le GDR MACS (en cours)**
Durée : 1 an (en cours)
Coordinateur : P. Berruet
Partenaires : LESTER, CRAN, LAGIS, LAG
Partenaires du CRAN : **J.F. Pétin**, D. Gouyon
Contribution : cette action fait suite à l'AS INTICA et porte plus spécifiquement sur la conception de systèmes de commande reconfigurable pour les S.E.D.

**P10** **Groupement d'Intérêt Scientifique « Surveillance, Sûreté et Sécurité des Grands Systèmes » (GIS 3SGS)**
*Projet LABIME (LAngage d'expression des Besoins en Informations des Métiers d'Exploitation*
Durée : 2 ans (2007/2008, 2008/2009)
Partenaires : CRAN, EDF, LORIA
Participants du CRAN : D. Dobre, D. Gouyon, G. Morel, **J.F. Pétin** (coordinateur du projet)

- Projet régional avec comité d'évaluation et rapports ou communications

**P11** **REGION LORRAINE, Projet fédérateur de recherche Etat – Région**
*Sûreté industrielle des systèmes – Opération 01 : Conception coordonnée de systèmes industriels sûrs et de qualité*
Durée : 4 ans (1994 – 1998)
Partenaires : CRAN, CRIN, ENSGSI, GREEN, LAEI (Metz)
Participants du CRAN : G. Morel, **J.F. Pétin**, B. Iung, J. Ragot, J.F. Aubry, R. Husson

3 Rapports d'activités

- Contrats industriels sans comité d'évaluation et avec rapports (8)

**P12** **Contrat de recherche EDF/DER Chatou (1992-1999)**

**Rp5** **Contrat n° P37/2K4714**
**J.F. Pétin**, B. Iung, G. Morel. *Actionneurs Intelligents: évaluation technique et économique pour l'insertion dans les centrales*, 4 notes de synthèse, Septembre 1992.

**Rp6** **Chatou, Contrat n° P37/2L1834**
**J.F. Pétin**, E. Neunreuther, B. Iung, G. Morel. *Modélisation des activités pour la description et l'élaboration des diagrammes fonctionnels*, 2 Volumes, Janvier 1994.

**Rp7** **Contrat n° P37/2L 5948**
E. Neunreuther, J.F. Pétin, B. Iung. *Spécification des blocs fonctionnels élémentaires EDF dans le formalisme ST de la norme IEC 1131-3*, Juillet 1994.

**Rp8** **Contrat n° P37/2M 1347**
E. Neunreuther, J.F. Pétin, H. Panetto, B.Iung. *Méthodologie de Validation d'une bibliothèque de blocs fonctionnels élémentaires*, 2 Volumes, Juin 1995.

**Action de R&D P3136R (EDF/DER Chatou, CRAN, LORIA)**

**Rp9** P. Lamboley, B. Mermet, **J.F. Pétin**, D. Méry. *Spécification de la documentation d'exploitation d'une installation de production d'électricité: étude bibliographique*, 46 pages Avril 1999.

**Rp10** P. Lamboley, **J.F. Pétin**, B. Mermet, D. Méry. *Spécification de la documentation d'exploitation d'une installation de production d'électricité: proposition d'une approche méthodologique*, 71 pages, Septembre 1999.

**P13** **INRS, Contrat établi dans le cadre de la convention CIFRE de D. Evrot**

**Rp11** **J.F. Pétin**, D. Evrot, G. Morel. *Définition d'un processus de développement de systèmes automatisés sûrs pour l'industrie manufacturière*. Deux rapports d'avancement des travaux (45 et 57 pages), décembre 2005 et décembre 2006

**P14** **Trane**, **Contrat établi dans le cadre de la convention CIFRE de H. El Haouzi**

**Rp12** H. El Haouzi, A . Thomas, **J.F. Pétin**. *Intégration de nouvelles technologies d'identification pour la traçabilité et la synchronisation de flux dans une chaîne logistique*, Rapport d'avancement des travaux, décembre 2006

# 2. PROTOCOLES DE COLLABORATION ET DE RECHERCHE

- G. Morel, **J.F. Pétin**, Convention de collaboration entre l'Université H. Poincaré, le CNRS, la Société INRS et le CRAN, Convention n° 5052332 – Janvier 2005

- A. Thomas, **J.F. Pétin**, Convention de collaboration entre l'Université H. Poincaré, le CNRS, la Société TRANE et le CRAN, Septembre 2005

# XI Rayonnement scientifique

## 1. PARTICIPATION A L'ADMINISTRATION DE LA RECHERCHE

- Membre élu titulaire de la commission de spécialiste 61ème section de l'Université H. Poincaré depuis 2001
- Membre nommé suppléant de la commission de spécialiste 61/63ème section de l'Université de Metz depuis 2004
- Membre nommé au Conseil scientifique du CRAN depuis 2005

## 2. ANIMATION SCIENTIFIQUE

### *2.1 Niveau Local*

- **Responsable** du projet intitulé « **Modèles et Méthodes Formelles pour l'automatisation des processus de production** » du groupe thématique « Productique et Automatisation des Procédés Discrets » du CRAN avec pour mission principale de mener une action d'animation et de coordination permettant de réaliser les objectifs définis dans le contrat quadriennal 2000-2003 du CRAN (UMR 7039 CNRS-UHP-INPL). Ce projet de 5 enseignants-chercheurs, 2 post-doctorants et 7 doctorants, construit autour de deux actions « Cadres formels pour l'automatisation de processus de production » et « Interopérabilité des modèles d'entreprise » a porté sur la définition de cadres formels pour le développement de systèmes automatisés et sur l'intégration de ces systèmes aux processus d'entreprise. Son impact se mesure au niveau de sa production (6 revues, 1 ouvrage, 19 communications, 3 thèses soutenues), par des participations à des programmes européens mais également par la proposition d'actions au niveau national (Action spécifique CNRS n°198 Impact des NTIC en Automatisation) et européen (INTEROP Network of Excellence, Interoperability Research for Networked Enterprises Applications and Software).

- **Coordinateur** pour le CRAN (avec N. Brinzei) du **Centre d'innovation et de démonstration des technologies sûres de fonctionnement (SafeTech)** inscrit à la fois dans le cadre du Contrat de Plan Etat-Région 2007-2012 et du Groupement d'Intérêt Scientifique « Surveillance, Sûreté et Sécurité des grands systèmes ». Ce centre doit offrir un plateau technologique unique en France. Ce plateau est destiné, d'une part, à accompagner les travaux de recherche qui seront menés dans ces projets et, d'autre par, à offrir aux entreprises un site de démonstration. SafeTech constituera un outil de coordination des recherches et permettra la mise en réseau des compétences dans le domaine de la sûreté de fonctionnement. Au delà du contexte régional (CPER, pôle MIPI, pôle Fibres), SafeTech affiche une ambition nationale et internationale (promotion et mise en place de projets européens pour le 7e programme cadre européen notamment). Les défis techniques auxquels fera face SafeTech comprennent la conception de systèmes sûrs de fonctionnement, la tolérance active aux défauts, la connectivité sans faille, la fiabilité, la sécurité, la qualité des services, etc. Pour le CRAN,

SafeTech doit s'organiser autour de plusieurs éléments comprenant notamment : des plates-formes industrielles pilotes pour la validation des concepts de sûreté et sécurité des systèmes complexes, un réseau de capteurs et son système intelligent de surveillance, une plate-forme collaborative et de démonstration supportant les différents outils/méthodes employés pour l'évaluation prévisionnelle de la sûreté de fonctionnement.

- **Représentant du CRAN** (avec D. Sauter et D. Maquin) au conseil des opérations de l'axe Sécurité et Sûreté des Systèmes (SSS) du Contrat de Plan Etat-Région 2007-2012 MISN.

### *2.2 Niveau National*

- **Co-Animateur** avec Bruno Denis (LURPA – ENS Cachan) de l'**Action Spécifique CNRS/STIC n°198, RTP PCM 47 « Impact des NTIC en automatisation »** financée sur la période septembre 2003 - décembre 2004. Cette A.S., comptant 8 laboratoires participants (CRAN, LURPA, CReSTIC, LAG, LAGIS, LESTER, LGEF/INSA Lyon, LIMI,) a eu pour objet l'étude de l'impact des nouvelles technologies sur la conception et l'exploitation des automatismes industriels. Deux aspects complémentaires ont été abordés : la sûreté de fonctionnement de systèmes de commande distribués autour de réseaux non déterministes (en particulier autour d'Ethernet) et leur intégration au sein des architectures de pilotage et de conduite des systèmes de production. Les résultats de cette action ont été présentés à la communauté scientifique et industrielle lors d'une journée bilan le 27 janvier 2005 et ont fait l'objet d'une communication [C19].
- Participation au projet « Reconfiguration des Systèmes à Evénements Discrets » soutenu et financé en 2007 par le GDR MACS. Coordinateur du projet : P. Berruet, Participants : CRAN, LAG, LAGIS, LESTER, LIESP.
- **Coordinateur** du projet **LABIME** (LAngage d'expression des Besoins en Informations des Métiers d'Exploitation) labellisé en septembre 2007 dans le cadre du GIS 3SGS (cf chapitre V, paragraphe 3.3.).

### *2.3 Niveau Européen et International*

- Membre nommé par la SEE au comité technique « TC 5.1. Manufacturing Plant Control » de la société scientifique IFAC pour la période 2005-2008 (Chair: Carlos Eduardo Pereira (BR).

### *2.4 Participation à des groupes de travail*

- De 1992 à 1994, participant au groupe du CIAME Actionneurs intelligents, 1 communication.

- De 1995 à 1998, participant au groupe du CIAME Interopérabilité.

- De 1992 à 1996, participant au groupe Collaboration CAO Automatique (C2A), GdR Automatique du CNRS, relatif aux langages synchrones (Lustre, Signal, Esterel, …).

- De 1998 à 2003 : participant au Groupement de Recherche en Productique (GRP) et plus particulièrement au groupe de travail COMPIL (Commande et Pilotage), 2 communications en conférence plénière et 1 communication dans le cadre du groupe COMPIL.
- De 2000 à 2005, participant au groupe de travail COSED (Commande Opérationnelle des SED) sous l'égide club EEA (2000-2002) puis du GDR Automatique (2002-2005), 3 communications.

- Depuis 2005, participant au groupe de travail INCOS (Ingénierie de la Commande et de la Supervision) du GdR MACS.

- Membre de **l'AFIS** (Association Française d'Ingénierie Système) et membre de **l'INCOSE** (International Council on Systems Engineering) depuis 2005. Participation au groupe de travail Sûreté de Fonctionnement de l'AFIS (Association Française d'Ingénierie Système).

## 3. ORGANISATION DE MANIFESTATIONS SCIENTIFIQUES

### *3.1 Participation au comité d'organisation de conférences*

- 9th IFAC Symposium on Information Control Problems in Manufacturing, Nancy & Metz, 24-26 juin 1998
- Vice-président du comité d'organisation du 4ème colloque francophone sur la Modélisation des Systèmes Réactifs (MSR'03), Metz,6-8/10/2003.
- Forum AFIS (Association Française d'Ingénierie Système), Nancy, 28 et 29 novembre 2007.

### *3.2 Organisation de tracks et sessions invités*

**SI1** Symposium **IFAC INCOM'1998** (June 24-26, 1998, Nancy)

- Participation au Comité d'Organisation (Président : P. Lhoste)
- **Organisateur** d'une session invitée « *Advanced automation engineering* », 5 papiers

**SI2** **Co-Organisateur** avec H. Panetto d'une session invitée au symposium **IFAC INCOM'2001** (September 20-22, 2001, Vienna, Austria) : « *Impact of Formal Models and Methods on Automation* », 5 papiers.

### *3.3 Organisation de tutoriaux*

Co-organisation (avec J.M. Roussel, LURPA-ENS Cachan) du tutorial « **Conception de la commande de SED sûrs de fonctionnement »** pour l'Ecole des JDMACS (Reims, 12-13 juillet 2007) sur sollicitation du Pr. J. Zaytoon, 4 conférenciers.

# 4. CRITIQUES SCIENTIFIQUES

## *4.1 Invitation à des jurys de thèse*

Jean-Louis LALLICAN, Proposition d'une approche composant pour la conception de la commande des systèmes transitiques. Doctorat de l'Université de Bretagne Sud, soutenu le 12 décembre 2007.
*Jury* : Michel Combacau, Jean Pierre Elloy (rapporteurs), Jean Paul Guyomar, **Jean-François Pétin**, André Rossi (examinateurs), Jean Luc Philippe (Directeur de Thèse), Pascal Berruet (Co-directeur de thèse).

## *4.2 Conférences*

- INCOM 01, INCOM 04, INCOM 06, IFAC Symposium on Information Control Problems in Manufacturing.
- IFAC World Congress 2005, 16th IFAC World Congress, July 4-7, 2005, Prague, Czech Republic.
- MSR 01, MSR 03, Colloque sur la Modélisation des Systèmes Réactifs.
- MOSIM 01, 3ème conférence francophone de Modélisation et de Simulation, 25-27 avril 2001, Troyes.
- INFORSID 2005, 23ème colloque national Inforsid, 4 au 27 Mai 2005 à Grenoble.
- ICRA 07, IEEE International Conference on Robotics and Automation, 10 to 14 April 2007, Rome.

## *4.3 Revues*

- JESA, Hermès (Journal Européen des Systèmes Automatisés) depuis 2002
- CEP, Elsevier (IFAC Control Engineering Practice) depuis 2004
- JIM, Springer (Journal of Intelligent Manufacturing) depuis 2006
- CII, Elsevier (Computers in Industry) depuis 2001
- EAAI, Elsevier, (IFAC Engineering Application of Artificial Intelligence) depuis 2007

# 5. CONSULTANCE – EXPERTISE

Expertise de la demande de convention CIFRE **N° 200/2004** (Sté ALSTOM POWER CENTRALES – LURPA – S. Limal) en 2004, sollicitation de l'ANRT (Pierre Vidal).

# Références Bibliographiques

## - Scientifiques


Abrial J.R., Mussat L. (1998). Introducing dynamic constraints in B. *Lecture Notes in Computer Science*, B'98 : The 2nd Int. B Conference (D. Bert Ed), Vol. 1393, pp 83–128, Springer Verlag.

Abrial J.R. (1996). The B Book: Assigning Programs to Meanings*. Cambridge Univ. Press*, ISBN 0-521-49619-5.

Alur R., Henzinger T.A. (1991). Logics and models of real time : a survey, *Lecture Notes in Computer Science*, Real time: theory and practice, REX Workshop, Vol. 600, pp 74-106

Auinger F., Brennan R., Christensen J., Lastra L.M., Vyatkin V. (2005). Requirements And Solutions To Software Encapsulation And Engineering In Next Generation Manufacturing Systems: OOONEIDA Approach. *International Journal of Computer Integrated Manufacturing*, Vol. 18(7), p572-585.

Balemi S., Hoffmann G.J., Gyugyi P., Wong-Toi H., Franklin G.F. (1993). Supervisory control of a rapid thermal multiprocessor. *IEEE Trans. on Automatic Control*, Vol. 38(7).

Behm P., Benoit P., Faivre A., Meynadier J.-M. (1999). Météor : a succesful application of B in large project. In *Formal Methods symposium*, Toulouse, France.

Belhimeur A. (1989). Contribution à l'étude d'une méthode de conception des automatismes des systèmes de conduite des processus industriels. Thèse de Doctorat en Automatique, Université des Sciences et Techniques de Lille.

Benveniste A., Berry G. (Eds) (1991).The synchronous approach to reactive and real time system. *Special Issue of the Proceedings IEEE*, Vol. 79(9), 09/1991.

Benveniste A., Caillaud B., Le Guernic P. (1999). From synchrony to asynchrony, in J.C.M. Baeten and S. Mauw, Editors, CONCUR'99, « Concurrency Theory », 10th International Conference, *Lecture Notes in Computer Science*, Vol. 1664, p. 162-177, Springer Verlag.

Boehm B. (1988). A Spiral Model of Software Development and Enhancement, *Computer*, Vol. 21(5), IEEE, May 1988, pp 61 – 72.

Bon-Bierel E. (1998). Contribution à l'intégration des modèles de systèmes de production manufacturière par méta-modélisation, Thèse de l'Université de Nancy I.

Brandin B.A., Malik R., Malik P. (2004). Incremental verification and synthesis of discrete-event systems guided by counter-example, *IEEE Transaction on Control Systems Technology*. Vo. 12(3), May 2004, pp 387-401.

Broenink J.F. (1999). Object-oriented modelling with bond graphs and Modelica, *International Conference on Bond Graph Modelling and Simulation* (ICBGM'99), San Francisco, January.

Brown J., Sackett P.J., Wortmann J.C. (1995). Future Manufacturing Systems – toward the extended enterprise, Computers in Industry, Vol. 25(3), pp 123-137.

Bouabana-Tebibel T., Belmesk M. (2007). An object-oriented approach to formally analyze the UML 2.0 activity partitions, *Information and Software technology*, Vol. 49, pp 999–1016.

Calvez J.P. (1990). Spécification et conception des systèmes. *Masson*.

Cassandras C.G., Lafortune S. (1999). Introduction to discrete event systems. *Kluwer Academic*. ISBN 0-7923-8609-4.

Chafik S. (2000). Proposition d'une structure de contrôle par supervision hiérarchique et distribuée : application à la coordination, Thèse de Doctorat de l'INSA de Lyon.

Chapurlat V. (2007). Vérification et validation de modèles de systèmes complexes : application à la modélisation d'entreprise. Habilitation à Diriger des Recherches, Université de Montpellier II.

Chen H., Hanisch H.M. (1999). Control synthesis of timed discrete event systems: part one and two, 14th Triennial IFAC World Congress, Beijing, P. R. China.

Chen Y.L., Lafortune S., Lin F. (2000). Incremental model evolution and reusability of supervisors for discrete event systems, *Automatica*, Vol. 36, pp. 243-259.

Chiron F., Kouiss K. (2005). Distributed control systems : from design to real implementation with UML 2.0, *IESM05 International Conference on Industrial Engineering and Systems Management*, Marrakech (Morocco), May 16-19th.

Clarke E.M., Grunberg O., Peled D.A. (2000). Model Checking, *The MIT Press*.

Corbier F. (1989). Modélisation et émulation de la partie opérative pour la recette en plateforme d'équipements automatisés, Thèse de l'Université de Nancy I.

Da Silveira G., Borenstein D., Fogliatto F.S. (2001). Mass customization: literature review and research directions. *Int. Journal of Production Economics,* Vol. 72**,** pp 1-13.

Deen S.M. (Ed.). (2003). Agent-Based Manufacturing - Advances in the Holonic Approach, *Springer*.

De Lamotte F., Berruet an P., Philippe J.L. (2006). 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, Inderscience Enterprises Ltd.

Denis B. (1994). *Assistance à la coneption et à l'évaluation de l'architecture de conduite des systèmes de production complexes*. Thèse de Doctorat de l'Université de Nancy I.

Dijkstra, E.W. (1959). A note on two problems in connexion with graph*, Numerische Matematik*, Vol. 1, pp. 269-271.

Easterbrook S. (2002). An introduction to formal modelling in requirements engineering, *10th joint International Requirements Engineering Conference*, Essen, Germany, 09/2002.

El Khattabi S. (1993). Intégration de la surveillance de bas niveau dans la conception des systèmes à événements discrets : application aux systèmes de production flexibles. Thèse de l'Université de Sciences et Technologie de Lille, 29/09/1993.

Endsley E. W., Almeida E. E., Tilbury D. M. (2006). Modular finite state machines: development and application to reconfigurable manufacturing cell controller generation. *Control Engineering Practice*, Vol. 14, pp 1127-1142.

Fabian M., Hellgren A. (1998). PLC-based Implementation of Supervisory Control for Discrete Event Systems. *Proceedings of the 37th IEEE Conference on Decision and Control,* Tampa, Florida.

Fabian M., Kumar R. (2000). Mutually nonblocking supervisory control of discrete event systems, *Automatica*, Vol. 36, pp. 1863-1869.

Favre J.M., Estublier J., Blay-Fornarino M. (2006). L'ingénierie dirigée par les modèles : au-delà du MDA. Informatique et Systèmes d'Informations. Hermes, Lavoisier 2006, ISBN 2-7462-1213-7.

Féliot C. (1997). Modélisation des systèmes complexes: intégration et formalisation de modèles. Thèse de Doctorat de l'Université des Sciences et Technologies de Lille.

Ferrarini L, Veber C., Schwab C., Tangermann M., Prayati A. (2005). Control functions development for distributed automation systems using the Torero approach. *16th IFAC World Congress*, Prague (Czech Republic), July 4-8, 2005.

Fiorèse S, Ménadier J.P. (2007). Découvrir et comprendre l'Ingénierie Système. Rapport AFIS

Forsberg K., Mooz H., Cotterman H. (2005). Visualizing Project Management: Models and Frameworks for Mastering Complex Systems, 3rd Edition, *Wiley*, ISBN: 978-0-471-64848-2.

Frachet J.P. (1987). Une introduction au génie automatique: faisabilité d'une chaîne intégrée d'outils CAO pour la conception et l'exploitation des machines automatiques industrielles, Thèse d'Etat, Université Nancy I.

Fusaoka A., Seki H., Takahashi K. (1983). A description and reasoning of plant controllers in temporal logic. *Proceedings of the 8th Int. Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, pp. 405-408.

García-Duque J., Lopez-Nores M., Pazos-Arias J.J., Fernández-Vilas A., Díaz-Redondo R.P., Gil-Solla A., Ramos-Cabrer M., Blanco-Fernández Y. (2006). Guidelines for the incremental identification of aspects in requirements specifications, *Requirements Engineering*, Vol. 11(4). pp 239–263, Springer.

Ghaffari A., Rezg N., Xie X. (2003). Algebraic and geometric characterization of Petri net controllers using the theory of regions, *Proceedings of 6th International Workshop on Discrete Event Systems*, WODES'02, Saragoza, Espagne, 2-4/10/2002.

Godon A., Ferrier J.L. (1997). DES control design using petri nets, *European Control Conference*, ECC'97, Bruxelles, 1-4/07/1997

Gouin A., Ferrier J.L.. (1999). Modeling and supervisory control of timed automata, *APII – Journal Européen de Systèmes Auomatisés*, Vol. 33(8-9).

Gout O., Lambolais T. (2004). Construction incrémentale de modèles comportementaux UML. *Congrès Approches Formelles dans l'Assistance au Développement de Logiciels*, AFADL'04, Besançon, France, pages 29-42, Juin 2004.

Gruber T.R. (1993). Toward principles for the design of ontologies used for knowledge sharing. In *Formal Ontology in Conceptual Analysis and Knowledge Representation* (N. Guarino and R. Poli, eds.), Kluwer Academic Publishers.

Gueguen H. (1996). Synchronisme dans l'évolution des Statecharts. *Colloque francophone sur la Modélisation des Systèmes réactifs*, MSR'96, p. 367.

Harel D. (1987). Statecharts: a visual approach to complex systems. *Science of computer programming*, Vol. 8(3), 231-275.

Harel D., Pnueli A. (1985). On the development of reactive systems in logic and models of concurrent systems, *NATO ASI Series*, Vol. 13, p. 477-498, K.R. Apt Editions, Springer Verlag.

Harel D., Lacover H., Naamad A., Pnueli A., Politi M., Sherman R. Shtull-Trauting A., Trakhtenbrot M. (1990). *Statemate : a working environment for the development of complex reactive systems*, IEEE Trans. on Software Engineering, Vol. 16(4), p. 403-416, April 1990.

Heaven W. and Finkelstein A. (2004). A UML Profile to Support Requirements Engineering with KAOS. *IEE Proceedings - Software*, Vol. 151, pp. 10-27.

Henry S., Zamai E., Jacomino M. (2004). Real time reconfiguration of manufacturing systems, *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, La Hague, Hollande.

Heymann M., Lin F., Meyer G. (1998). Synthesis and viability of minimally interventive legal controllers for hybrid systems, *Discrete Event Dynamic Systems: Theory and Application*, Vol. 8, pp 105-135.

Hiraishi K (2001). Synthesis of supervisors using learning algorithm of regular languages, *Discrete Event Dynamic Systems: Theory and Applications*, Vol. 11, pp 211-234.

Hollocks B.W., Goranson H.T., Shorter D.N., Vernandat F.B. (1997). Assessing Enterprise Integration for Competitive Advantage, Workshop 2, WG1 in *Enterprise Engineering and Integration : Building International Consensus* (eds. Kosanke K., Nells J.G.), International conf. on Enterprise Modelling and Modelling Technology, pp. 96-107, Springer-Verlag, Berlin.

Isermann R., Schaffnit, Sinsel (1999). Hardware-in-the-loop simulation for the design and testing of engine-control systems. Control Engineering Practice Vol. 7 (1999), pp 643-653.

Iung B. (1992). *Contribution à la distribution de l'intelligence dans les équipements de niveau zéro des processus industriels complexes*. Thèse de doctorat, Spécialité Production Automatisée, Université de Nancy I.

Iung B., Neunrether E., Morel G. (2001). Engineering process of Integrated – Distributed shop floor architecture based on interoperable field components, *International Journal of Computer Integrated Manufacturing*, Vol. 14(3), pp 246-262.

Jiang S., Kumar R. (2000). Decentralized control of discrete event systems with specializations to local control and concurrent systems. *IEEE Transactions on Systems, Man and Cybernetics, part B,* Vol. 30(5), pp. 653-660.

Johnson T. L. (2004). Improving automation software dependability: a role for formal methods ? 11th IFAC *International Conference on Information Control Problems*, Salvador-Bahia, Brazil, 4-7/04/2004.

Kamsu Foguem B. (2004), Modélisation et vérification des propriétés de systèmes complexes: application aux processus d'entreprise, Thèse de doctorat de l'Université de Montpellier II.

Katayama H., Bennet D. (1997). Agility, adaptability and leanness : a comparison of concepts and a study of practice, International Journal of Production Economics, Vol 60(61), pp 43-51.

Kosanke K. (1995). CIMOSA – Overview and status. Computers in Industry, Vol. 27, pp 101-109,

Košecká J. (1995). Supervisory control theory for autonomous mobile agents, Dissertation proposal presented to the Faculties of the University of Pennsylvania in Partial Fulfilment of the Requirements for the Degree of Doctor of Phylosophy.

Kumar R., Garg V., Marcus S.L. (1991). On controllability and normality of discrete event dynamical systems. S*ystems & Control Letters,* Vol. 17, pp. 157-168.

Kwak Y.H., Anbari F.T. (2006). Benefits, obstacles, and future of six sigma approach *Technovation*, Vol. 26, Issues 5-6, May-June 2006, pp 708-715

Lano K., Bicarregui J., Kan P. (2000). Experiences of using formal methods for chemical process control specification, *Control Engineering Practice*, Vol. 8(1).

Lazansky J., Stepankova O., Marik V., Pechoucek M. (2001). Application of the multi-agent approach in production planning and modelling. *Engineering Applications of Artificial Intelligence*, Vol. 14(3), June 2001, Pages 369-376.

Lecomte T. (2002). Event Driven B: language, tool support and experiments, *International Workshop on Refinement of Critical Systems* : *Methods, Tools and Experience*, in conjunction with the 2nd Conference of B and Z Users (ZB 2002), January 23 - 25, 2002, Grenoble, France.

Leduc R. J. (2002). Hierarchical Interface-based Supervisory Control. *Doctoral Thesis*, Dept. of Elec. & Comp. Engineering., University of Toronto.

Lhoste P. (1994). Contribution au génie automatique : concepts, modèles, méthodes et outils, Habilitation à diriger des recherches, Université de Nancy I.

Lhoste P., Faure J.M., Lesage J.J., Zaytoon J. (1997). Comportement Temporel du Grafcet, *RAIRO-APII-JESA (Journal Européen des Systèmes Automatisés)*, Vol. 31(4), p. 695-711, Hermès.

Lhote F., Chazelet P., Dulmet M. (1999). The extension of principles of cybernetics towards engineering and manufacturing, *Annual Reviews in Control*, Vol. 23/1, pp. 139-148.

Lin F. (1993). Analysis and synthesis of discrete event systems using temporal logic, *Control-theory and Advanced Technology*, Vol. 9(1), pp. 341-350.

Marchand H., Bournai P., Le Borgne M., Le Guernic P. (2000), Synthesis of discrete-event controllers based on the Signal environment. *Discrete Event Dynamic Systems: Theory and Applications*, Vol. 10, pp. 325-346.

Mayer F. (1995). Contribution au Génie Productique: Application à l'Ingénierie pédagogique en Atelier Inter-Etablissements de Productique Lorrain. Thèse de l'Université H. Poincaré-Nancy I.

McFarlane D., Bussmann S. (2000). Developments in holonic production planning and control, *International Journal of Production Plannig and Control*, Vol. 11(6), pp. 522-536

McFarlane D., J. Sarma, G. Chirn, J. Wong, A. Ashton (2003). Auto-ID systrems and intelligent manufacturing control, *Journal of Engineering Applications of Artificial Intelligence*, Vol.16, pp 365 – 376

Mehrabi M.G., Ulsoy A.G. and Koren Y. (2000). Reconfigurable manufacturing systems: Key to Future Manufacturing. *Journal of intelligent Manufacturing*, Vol. 11(4):403-419.

Meyer E. (2001). Développements formels par objets: utilisation conjointe de B et d'UML. Doctorat de l'Université Nancy 2, mars 2001.

Mellor S.J., Kendall S., Uhl A., Weise D. (2004). *Model Driven Architecture,* Addison-Wesley Pub Co, March, ISBN: 0201788918.

Ménadier J.P. (2002). Le metier d'intégration de systèmes. *Hermès Science Publications*, Lavoisier, ISBN 2-7462-0596-3

Moor T., Daroven J.M., Raisch J. (2002). Strategic refinements in abstraction based supervisory control of hybrid systems, *6th International Workshop on Discrete Event Systems*, WODES'02, Saragoza, Spain, 2-4/10/2002.

Morel G. (1992). *Contribution à l'Automatisation et à l'ingénierie des Systèmes Intégrés de Production*. Habilitation à diriger des recherches, Université de Nancy I.

Morel G., Panetto H., Zaremba M., Mayer F. (2003). Manufacturing enterprise control and management system engineering: rationales and open issues. *IFAC Annual reviews in Control,* Vol. 27(2), pp 199-209, ISSN: 1367-5788

Munérato F. (1988). Robotisation intégrée d'un îlot de production manufacturière: aspects contrôle/commande et communication. Thèse de Doctorat de l'Université de Nancy I.

Niel E., Pietrac L., Regimbal L. (2001). Advantages and drawbacks of the logic programm synthesis using supervisory control theory.10th IFAC/INCOM'01 Symposium, Vienna, Austria.

Nof S.Y., Morel G., Monostori L., Molina A. and Filip F. (2006). From plant and logistics control to multi-enterprise collaboration. *IFAC Annual Reviews in Control*. 30/1, 55–68.

Nourelfath M., Niel E. (2004). Modular supervisory control of an experimental automated manufacturing system. *IFAC Control Engineering Practice.* Vol.12, pp 205-216.

Ollero A., Morel G., Bernus P., Nof S.Y., Sasiadek J., Boverie S., Erbe H., Goodall R. (2002). From MEMS to Enterprise systems. *IFAC Annual Reviews in Control*, vol. 26(2), pp. 151-162

Pannequin R. (2007). Proposition d'un environnement de modélisation et de test d'architectures de pilotage par le produit de systèmes de production. Doctorat de l'Université H. Poincaré-Nancy I.

Panetto H. (2006). Meta-modèles et modèles pour l'intégration et l'interopérabilité des applications d'entreprises de production, Habilitation à Diriger des Recherches, Univ. H. Poincaré, 4/12/2006.

Paulk M.C. (1995). How ISO 9001 compares with the CMM, *IEEE Software*, pp 74-83, January 1995

Paynter M., Analysis and design of engineering systems, *M.I.T. Press*, ISBN 0-262-16004-8.

Pénalva J-M. (1997). La représentation par les systèmes en situation complexe. Doctorat de l'Université d'Orsay.

Paul Pettersson and Kim G. Larsen (2000). *Bulletin of the European Association for Theoretical Computer Science*, Vol. 70, pages 40-44.

Piétrac L. (1999). Apport de la méta-modélisation formelle pour la conception des systèmes automatisés de production, Thèse de l'Ecole Normale Supérieure de Cachan.

Pujo P., Pedetti M., Giambiasi N. (2006). Formal DEVS modelling and simulation of a flow-shop relocation method without interrupting the production, *Simulation Modelling Practice and Theory*, Vol. 14, pages 817–842.

Qiu R., Wysk R., Xu Q. (2003). Extended structured adaptive supervisory control of shop-floor controls for an e-manufacturing system. *International Journal of Production Research*, Vol. 41(8), pp. 1605-1620.

Ramadge P.J., Wonham W.M. (1987). Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, Vol. 25(1).

Roussel J.M., Faure J.M. (2006). Designing dependable logic controllers using algebraic specifications, *Control Engineering Practice*, Vol. 14(10), pp 1143-1155.

Sanchez A., Macchieto S. (1995). Design of procedural controllers for chemical processes, *Computers Chemical. Engineering*. Vol. 19, pp. S381-S386.

Selic B. (1998). Using UML for modelling complex real-time systems, *Lectures Notes in Computer Science*, 1474, pp. 250-262, ISSN: 0302-9743.

Sfalcin A. (1992). Contribution d'une approche sémiotique à la réutilisation des composants de commande des Machines et Systèmes Automatisés de Production. Doctorat de l'Univ. Nancy I.

Shayman M. A., Kumar R. (1995). Supervisory control of nondeterministic systems with driven events via prioritized synchronization and trajectory models, *SIAM Journal on Control and Optimization*, pp. 469-497, Mars 1995.

Sheard S. (2006). Complex Systems Science and its Effects on Systems Engineering. *European Systems Engineering Conference*, 18-20 September 2006, Edinburgh, UK

Shell T. (2001). Systems functions implementation and behavioural modelling: system theoretic approach, *International Journal of Systems Engineering*, Vol. 4(1).

Sowa J.F., ZAchman J.A. (1992). Extending and formalizing the framework for Information system architecture, IBM Systems Journal, Vol. 31(3), pp 590-616.

Spivey J.M. (1989). The Z notation a reference manual, *Prentice Hall International*.

Staroswiecki M., Bayart M. (1996). Models and languages for the interoperability of smart instruments, *Automatica*, Vol 32(6), June 1996, Pages 859-873.

Tahir O., Cardoso J., Sibertin-Blanc C. (2003). Génération automatique de diagrammes états-transitions à partir de diagrammes de séquences UML: une approche basée sur la sémantique des réseaux de Petri, *Modélisation des Systèmes Réactifs* (actes de MSR'03, 6-8/10/2003, Metz, France), Hermes Lavoisier, pp 505-520.

Taouil-Traverson S.,(1997). Stratégie d'intégration de la méthode B dans la construction de logiciel critique, Thèse de l'Ecole National Supérieur des Télécommunications, référence ENST 97E017.

Thomesse J.P. (1999). Fieldbuses and interoperability,*Control Engineering Practice*, Vol. 7,pp 81-84.

Toguyeni A. K. A., Craye E., Sekhri L. (2006), Study of the diagnosability of automated production systems based on functional graphs, *Mathematics and Computers in Simulation,* Vol. 70(5-6), Pages 377-393

Tranoris C., Thramboulidis K. (2006). A tool supported engineering process for developing control applications, *Computers in Industry*, Vol. 57(5), June 2006, Pages 462-472.

Tsubone H., Horikawa M. (1999). A comparison between machine flexibility and routing flexibility, *The International Journal of Flexible Manufacturing Systems,* Vol. 11, Pages 83-101

Vahidi A., Fabian M., Lennartson B. (2006). Efficient supervisory synthesis of large systems. *Control Engineering Practice*, Vol. 14(10), pp 1157-1167.

Valckenaers P. (Editor) (2001), Special issue: Holonic Manufacturing Systems, *Computer In Industry*, Vol. 46(3), pp. 233-331

Vogel C. (1988). Le Génie Cognitif. *Collection Sciences Cognitives*, Edition Masson, 1988, ISBN 2-225-81332-5

Vogrig R., Baracos P., Lhoste P., Morel G., Salzemann B. (1987). Flexible manufacturing shop. *Manufacturing Systems*, Vol. 16(3).

Wong K.C., Wonham W.M. (1998). Modular control and coordination of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications,* Vol. 8(3).

Wonham W.M., Ramadge P.J. (1987). On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, Vol.25(3).

Wright P.K., Bourne D.A. (1988). *Manufacturing intelligence*. Addison-Wesley, ISBN 0-201-13576-0.

Xu Y., Brennan R., Zhang X., Norrie H.. (2002). A Reconfigurable Concurrent Function Block Model and ist implementation in Real-Time Java, *Journal of Integrated Computer-Aided Engineering,* Vol. 9, pp 263-279.

Yoo T.-S., Lafortune S. (2002). A general architecture for decentralized supervisory control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, Vol. 12.

Zamaï E., Chaillet-Subias A., Combacau M. (1998). An architecture for control and monitoring of discrete events systems, *Computers in Industry,* Vol. 36(1-2), pp 95-100.

Zaytoon J., Carre-menetrier V. (2001). Synthesis of a correct control implementation for manufacturing systems. *Int. Journal of Production Research*, Vol. 39, pp. 329-345.

Zaytoon J., Extension de l'analyse fonctionnelle à l'étude de la sécurité opérationnelle des systèmes automatisés de production, Thèse de l'Université Lyon.

Zhang W., Diedrich Ch., Halang W.A. (2005). Specification of Function Block Applications with UML. *Proc. IEEE Intl. Conf. on Robotics and Automation*, Barcelona, Spain.

## - Techniques & Standards

DE 98/37/CE (1998). Directive machine 98/37/CE du 22 juin 1998 concernant le rapprochement des législations des états membres relatives aux machines. *Journal Offciel des Communautés Européennes,* n°L.207 du 23 juillet 1998, 46p.

DO 178B. *Software considerations in airborne systems and equipment certification*

EIA 632 (1998). Processes for engineering a system, avril 1998

IEC 61131-3 (1993). *Programmable Logic Controllers (PLC),* Part 3: programming Languages.

IEC 61499 (2005). *Function blocks for industrial-process measurement and control systems*

IEC 61508 (1998). *Functional safety of electrical/electronic/programmable electronic safety-related systems*, 7 parties

IEC 61511 (2003). *Functional safety - Safety instrumented systems for the process industry sector*. Part 1: Framework, definitions, system, hardware and software requirements, Part 2: Guidelines for the application of IEC 61511-1, Part 3: Guidance for the determination of the required safety integrity levels

IEC 62264 (2002). *Enterprise-control system integration*. Part 1. Models and terminology. Part 2: Model object attributes. ISO/IEC FDIS Standard, IEC and ISO, Geneva, Switzerland.

IEC 61804 (2003). Function block for process control, Part1 : overview of system aspects, Part 2: Function blocks for industrial-process measurement and control systems.

IEC 62061 (2005). *Safety of machinery* - Functional safety of electrical, electronic and programmable electronic control systems

IEEE 1220 (1999). *Standard for application and Management of the Systems Engineering Process*

International Telecommunication Union (2005), The Internet of Things, ITU Internet Reports 2005, http://www.itu.int/internetofthings

ISO 12100 (2003). Safety of machinery – Basics concepts, general principles for design

ISO 19440 (2006). Entreprise Intégrée – Constructions pour la modélisation d'entreprise, ISO, ISO TC 184/SC5/WG1 - CEN TC 310/WG1 , Genève

ISO/IEC 15288 (2002). *System Life Cycle Processes and its Guide ISO*, ISO TC 184/SC7/JTC1

ISO 8402 (1995). Management de la qualité et assurance de la qualité - Vocabulaire

Sematech (1995), *Device Interoperability Guideline for Sensors, Actuators, and Controllers*, Technology Transfert 94102567A-STD, February. www.sematech.org

SYSML (2007). *OMG SysML Specification v1.0.*, Extends OMG's Unified Modeling Language (UML) for Systems Engineering, http://www.omgsysml.org/

UML (2003). *UML 1.5 specification*, OMG. http://www.orm.org

# Acronymes

| | |
|---|---|
| APS | Advanced Planning and Scheduling |
| CIM | Computer Integrated Manufacturing |
| CMM | Control, Maintenance and technical Management system |
| COTS | Components On The Shelves |
| EICM | Enterprise Integration Capability Model |
| ERP | Enterprise Resource Planning |
| HMS | Holonic Manufacturing Systems |
| IAM | Intelligent Actuation and Measurement |
| IEC | International Electrotechnical Commission |
| IDM | Ingénierie Dirigée par les Modèles |
| IMS | Intelligent Manufacturing System |
| ISA | Instrument Society of America |
| ISO | International Organisation for Standardisation |
| MES | Manufacturing Execution System |
| MESA | Manufacturing Execusion Systems Association |
| OPC | OLE for Process Control |
| RFID | Radio Frequency IDentification |
| SCM | Supply Chain Management |
| SED | Systèmes à Evénements Discrets |
| SCT | Supervisory Control Theory |
| UML | Unified Modeling Language |
| XML | eXtensible Markup Language |
| SYSML | System Modelling Language |

# Liste des Figures

# Liste des Tableaux

# Annexes : Publications Internationales

## CONFERENCE AVEC COMITE DE LECTURE ET ACTES

D. Evrot, **J.F. Pétin**, G. Morel, P. Lamy (2007). Using SysML for identification and refinement of machinery safety properties, 1st IFAC Workshop on Dependable Control of Discrete-event Systems, Cachan, France, 13-15 juin 2007.

## REVUES AVEC COMITE DE LECTURE

**J.F. Pétin**, B. Iung, G. Morel (1998). Distributed Intelligent Actuation and Measurement system within an integrated shop-floor organisation, *Computers In Industry Journal*, *Special issue on Intelligent Manufacturing System, VOL 37*, pp 197-211, Elsevier Sciences Publisher, ISSN 0166-3615, 1998

D. Gouyon, **J.F. Pétin**, A. Gouin (2004). A pragmatic approach for modular control synthesis and implementation, *International Journal of Production Research*, vol. 42, n° 14, pp 2839-2858, Taylor & Francis Publisher, ISSN 0020-7543, Jul 2004.

**J.F. Pétin**, G. Morel, H. Panetto (2006). Formal specification method for production systems automation, *European Journal of Control*, Volume 12/2, 115-130, Hermès Science Publishers, March/April 2006, ISBN 2-7462-1480-6

**J.F. Pétin**, D. Gouyon, G. Morel (2007). Supervisory synthesis for product-driven automation and its application to a flexible assembly cell, *IFAC Control Engineering Practice journal*, Volume 15, Issue 5, May 2007, Elsevier Science Publisher, ISSN 0967-0661

ELSEVIER

Computers in Industry 37 (1998) 197–211

COMPUTERS IN INDUSTRY

# Distributed intelligent actuation and measurement (IAM) system within an integrated shop-floor organisation

Jean-François Pétin *, Benoit Iung, Gérard Morel

*Nancy Research Centre for Automatic Control UPRES-A 7039, CNRS and University Henri Poincaré, Faculté des Sciences, BP 239, 54506 Vandoeuvre lès Nancy Cedex, France*

**Abstract**

The world of instrumentation is undergoing a major evolution shift as embedded intelligence and field-bus communication allow a migration of functionality into all parts of distributed automation systems. This innovative architecture allows an integrated shop-floor organisation by providing Control, Maintenance and technical Management (CMM) activities with a common and efficient representation of the process. On the basis of the industrial research and development initiated by European projects, evolution based on intelligent field devices and leading towards more adaptive automation systems is discussed. An engineering framework, as needed to structure the system models in line with the distribution and integration requirements, is suggested through a case study based on the level control system of our pilot laboratory. 

*Keywords:* Integrated shop-floor system; Intelligent systems; Distributed architecture; System engineering; Formal engineering

## 1. Introduction

Computer Integrated Manufacturing (C.I.M.) [1] promotes the integration of the enterprise activities. The main aim is to share knowledge and information on production plant needed for increasing the enterprise competitiveness.

Applying C.I.M. to shop-floor level leads to the integration of the operational agents responsible for the plant. These agents cover three main categories of activities: control activities designed to achieve the plant application mission, maintenance activities ensuring the availability of the production system resources, and technical management activities optimising the production costs by modifying control or maintenance procedures, tools and materials. An integrated organisation requires that information from the process is made available for common use by the Control, Maintenance and technical Management (CMM) activities [2].

The embedded intelligence extended to field devices and field-bus communication [3] provides a big opportunity for better solution of the CMM users' requirements. Migration of functionality into all parts of a distributed Intelligent Actuation and Measurement (IAM) system helps in providing an informational representation of the production process as efficiently as is possible. This representation is distributed through field-bus architecture to all integrated activities of CMM.

First industrial experiments with these innovative architectures have been initiated by European Esprit

* Corresponding author. CRAN/GSIP, Faculté des Sciences, BP 239, 54506 Vandoeuvre les Nancy Cedex, France. Tel.: +33-3-83-91-24-08; e-mail: jean-francois.petin@cran.u-nancy.fr

0166-3615/98/$ - see front matter 
PII: S0166-3615(98)00099-2

projects in IAM, to which we contributed as academic experts. [1] These projects aimed to demonstrate the industrial feasibility of CMM & IAM and illustrate the resulting benefits through experimental pilots. They proposed a normative framework for interoperable solutions based on intelligent devices from multiple suppliers. Main features of the researches and developments are highlighted in this section.

Major results and concepts of these projects have been scientifically validated and improved using the level control system implemented on our pilot laboratory.

The first aspect that has been thoroughly studied relates to the distributed architecture. Evolution from system with autonomous field-devices integrated in a hierarchical organisation towards more interoperable and adaptable system is discussed in Section 2.

The second aspect relates to the engineering process. The normative framework of the IAM Esprit projects is augmented by a structured engineering framework that systematically leads to architectures supporting requirements of distributed organisations. Section 3 presents our approach through the level system case study.

The formalisation of this structured engineering framework leads to proposals for future work whose goal would be to master a more intelligent design process of adaptable systems. The proposals are outlined in Section 5.

## 2. European researches and developments

The first work towards an integrated CMM organisation supported by a distributed IAM architecture started in the 1980s under the aegis of the Commission of the European Communities.

### *2.1. Integrated CMM system*

It was at that time that a group of users in the European power supply industries formed a joint group that addressed a key common problem faced by their industries: the lack of information exchanged between the current islands of automation at the shop-floor level.

The solution advocated by this core group was to integrate the control and maintenance activities through the technical management system, in order to share a common representation of the plant functions. That led to experiments with the concept of an integrated CMM system. Initiated in the project ESPRIT II-2172 DIAS, this concept is further developed in the current project ESPRIT IV-20874 REMAFEX [4].

Benefits of such an integrated organisation of the shop-floor are delivered by:

- optimisation of control based on synthetic and validated information,
- upgrading of maintenance from curative to predictive procedures based on efficient reports on the devices failures and degradations,
- real-time implementation of technical management by recording historical data allowing to manage cost-optimised control and maintenance activities.

### *2.2. Distributed IAM system*

A distributed architecture based on intelligent actuators and transmitters communicating via field-bus has been promoted to support the integrated CMM organisation. In the context of industrial projects, intelligent actuators and transmitters represent distribution of information processing and sharing directly into the field devices.

The need for embedded processing is justified by the lack of useful data provided by traditional field instrumentation based on analog (e.g., 4–20 mA) signal standards. More pertinent information is normally necessary in order to operate the plant with increased efficiency.

Distribution of information processing down to the field-devices adds to their classical missions new functionality related to data validation and synthesis, monitoring of device functions, etc. Additional processing is expected to provide a consistent and significant representation of the physical process through

[1] ESPRIT II-2172 DIAS ‘Distributed Intelligent Actuators and Sensors’; ESPRIT III-6188 PRIAM ‘Prenormative Requirements for Intelligent Actuation and Measurement’; ESPRIT III-6244 EIAMUG ‘European Intelligent Actuation and Measurement User Group’, ESPRIT IV-23525 IAM-PILOT ‘Intelligent Actuation and Measurement Pilot’.

an 'informational bus'. This is the foundation for the integration of the CMM activities. In other words, the closer is the data representation to the physical flow in the process, the better is the semantics of its informational representation for business flow.

### 2.3. IAM technology needed for CMM applications

Taking into account the technological offer from the suppliers of automation devices, distributed IAM architecture results from an integration of intelligent field devices with additional functions resident in external devices (e.g., computers) designed to provide actuation and measurement services required by the CMM applications (Fig. 1).

Consequently, the engineering studies have to map two complementary processes. On one hand, the users have to define the complete support needed by all human organisations that are interacting with the CMM automation system throughout its life cycle. On the other hand, the suppliers of automation devices are looking for guidance on real user needs when evolving their products into intelligent field devices. The project Esprit III-6188 PRIAM has investigated a structured approach and generic reference models of an anticipated vision of functional and technological processes shared between users and suppliers.

The key point to be addressed is related to interoperability among all processing systems of the IAM architecture, as depicted in Fig. 1. It implies:

- communication interoperability, i.e., the compliance to a common communication protocol when exchanging information [5],
- functional interoperability, i.e., the coordination of complementary services in terms of functionality, performance, reliability, . . . , needed to achieve CMM missions.

Interfaces among the IAM devices have been defined by the project Esprit III-6188 PRIAM at the appropriate level required to support the standardisation process for open systems.

The key idea of PRIAM was to define, in a functional standard, a consistent set of IAM services needed to satisfy CMM requirements. Field devices would provide functions selected from this set. Interoperability is ensured through communication standards that define the input/output interfaces of the devices according that implement the selected functions.

The project ESPRIT III-8244 EIAMUG shared these generic concepts with a representative cross

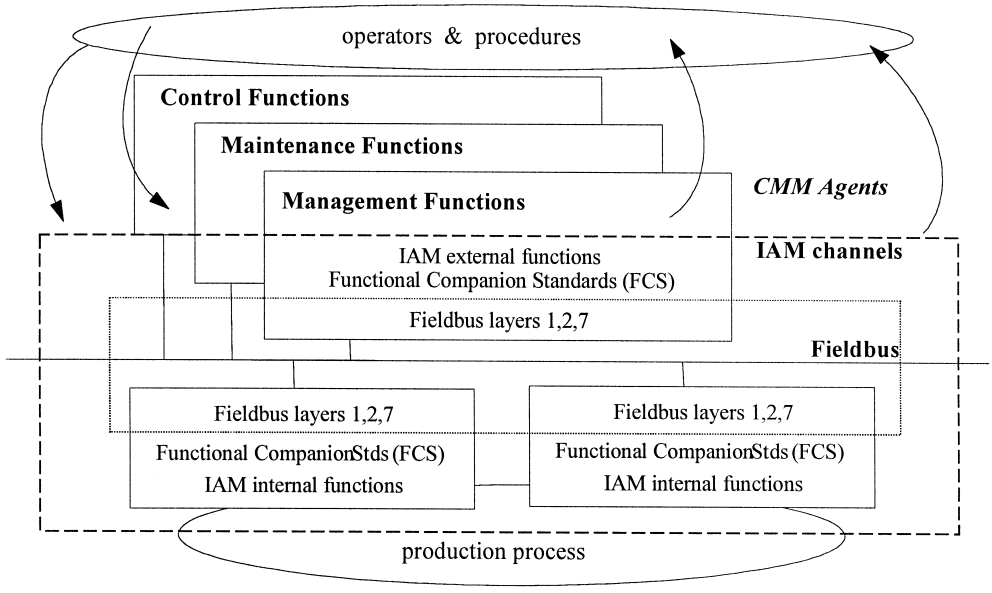



Fig. 1. CMM & IAM systems.

section of user industries (chemical, oil, gas, food, water treatment, etc.) and provided inputs to the current international standardisation effort in IEC TC65/WG6 [6] related to field-bus communications and function blocks for process control.

### 2.4. Functional standards

Functional Companion Standard defines IAM services as a triplet {F, O, D} where F is the function supplying a CMM service (Fig. 2a), O the set of objects produced or consumed by the function (Fig. 2b) and D the informational flow which connects functions to objects (Fig. 2c). Each triplet {F, O, D} must be considered as distribution-independent, i.e., non splittable according to distribution criteria.

The modelling approach is then similar to the specification of data flow diagrams, such as SA or SA-RT [7]. Dedicated attributes have been added to cater for the specific features of the CMM & IAM such as:

- normative classes (basic, optional, vendor),
- types of communication services (persistent, periodic, ...),
- access right management (access mode control, enabling protocols, ...),
- validation of process information (validity index, domain values, ...).

Benefits of such an approach depend on the independence of the functional standard from any distribution criteria and technological constraints. Future evolution of the IAM technology could be achieved by encouraging the suppliers to provide devices that progressively implement actuation or measurement services from the functional standard.

### 2.5. Communication standard

The Communication Companion Standard is a description of the services provided by a particular field device. It includes the services required for communicating produced/consumed information.

Such a description is in line with existing approaches such as D.D.L. (Device Description Language) initiated by suppliers of field devices.

Functional Communication Standard defines for each IAM device, a set of functions F, selected from the functional standard {F, O, D}. The informational

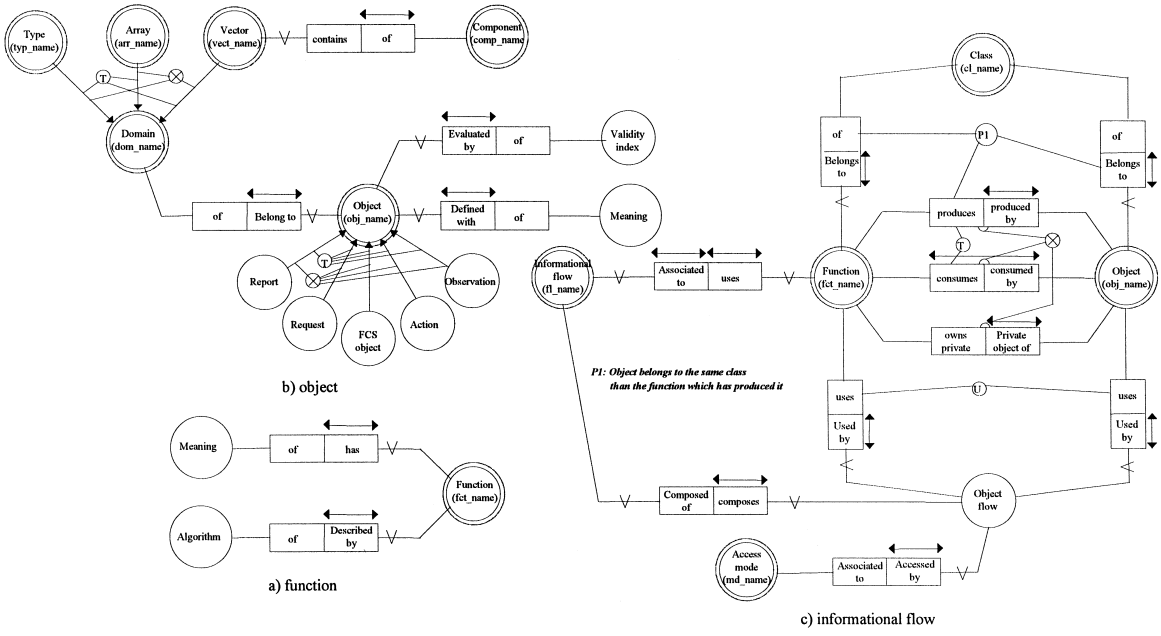



Fig. 2. Extract of the Functional Companion Standard meta-model in NIAM [8] formalism.

interfaces of a device are computed as a sub-set of the objects O associated to the selected functions F through the data flows D. Data flows which connect functions contained within the device are composed of private objects, internal to the device. Otherwise, they would appear as inputs/outputs of the device and will be available for exchanges through field-bus communications.

The communication standard may include the technological features of a communication network. The adjustment to a particular field-bus is done by assembling information for individual communication services and by formatting the resulting data packages. Field-bus studied in the context of the PRIAM project was to FIP standard. Extensions based on CENELEC EN50170 are used in the European pilot for IAM currently developed by the project ESPRIT IV-23525 IAM-PILOT.

### *2.6. CMM & IAM tools*

In order to assist the partners of PRIAM and EIAMUG projects in establishing pre-normative proposals for IAM, a software tool has been developed. [2] Concepts handled by the tool are based on the meta-model of the functional companion standard complete with particularisation and generalisation mechanisms allowing a progressive convergence towards a consensus in the pre-normative proposal.

The tool handles materials for the static description of the IAM services needed for CMM applications. It introduces systematic guidelines by focusing the requirements on seven classes of required IAM support: access right management, documentation, configuration, parametrisation, test, state of measurement/actuation and diagnostics.

Dynamic validation of these functional requirements requires a description of the behaviour associated with the functions that are simulated. This implies connection of the PRIAM tool to a dedicated software environment.

PRIAM tool has been tightly coupled to the tool SPEX. [3] The latter is an automation engineering tool allowing description and simulation of complex behaviour by gathering elementary behaviours described in the IEC 1131-3 standard. Note that the PRIAM tool is also open to other environments (such as Matlab/Simulink for process control simulation) thanks to its IMPORT/EXPORT mechanisms. [4]

Mapping of functional requirements in a distributed architecture is partially supported by the PRIAM tool. As a static description, the tool provides mechanisms for the selection of functions from those already defined, and for automatic generation of informational interfaces of the IAM devices covered by the communication companion standard.

In order to analyse conformance of IAM device to the functional requirements, a behavioural description and simulation of the resulting distributed architecture is required. It is handled by the tool SPEX on the basis of the skeleton structures provided by the PRIAM tool.

Simulation of a distributed application has to take into account the behaviour of the communication system. This problem can be solved by adding a model of field-bus protocols [9] to the behavioural description.

However, the field-bus timing issues need to be considered very carefully when comparing the temporal properties of the modelled system. Field-bus performance is usually sufficient to support the real-time constraints of process control systems. Analysis shall be aimed at synchronisation of the exchanged variables by modelling their temporal modalities (synchronous hypothesis, field-bus failures up to disruption, validity of information transmitted, ...).

### *2.7. Outlook from European research and development into CMM & IAM*

The European research and development projects presented in the previous sections have demonstrated some benefits of an integrated CMM shop-floor organisation supported by a distributed IAM architecture. Most obvious benefits are in costs (reduction of

[2] The tool PRIAM has been specified by CRAN and prototyped by T.N.I.

[3] Spex is a T.N.I. product.

[4] Matlab/Simulink is a Math Works product.

wiring costs, wire terminations and associated documentation), increasing availability of the field-devices, provision of additional information (e.g., allowing the move from scheduled towards predictive maintenance) and dependability (efficiency of the information processed by CMM systems).

However, intelligent field devices have today only a limited autonomy within hierarchical organisations. Considering the need to react to changes in the environment, IAM architectures have to be more flexible and delegate more responsibilities to the field devices. This challenge requires answers to the following questions:

- what are the functions required in order to increase the autonomy of the field devices (to proceed from a basic reflex behaviour towards allocation of decision tasks as advocated by multi-agents [10] or self organised systems [11]?
- what engineering process would develop IAM devices with the required level of intelligence?

These questions are discussed in the following sections using the engineering approach and solutions obtained on the IAM platform in our laboratory (Fig. 3).

This case study focuses on the control of the water level Lv1(t) in a vertical storage tank by modifying a Qo(t) flow rate output, through a modulating valve *(LCV2)*, according to a level measurement provided by a differential pressure transmitter *(DP1)*, and in presence of an external perturbation represented by the input flow into the tank Qi(t).

## 3. Towards a real embedded intelligence?

Formal analysis of quantitative results of the IAM European projects implies references to a scientific classification for qualifying the level of intelligence embedded in field-devices. The aim is to qualitatively measure the real improvements provided by the processing, storing and communication capabilities allocated to the field-devices that have developed from a basic reflex behaviour based on validated process data towards more autonomous behaviour based on process information.

Enterprise Integration Capability Model [12] proposes five levels of system integration (fragmented, rigid, visible, interoperable, adaptable).

According to this model, our experimental laboratory platform has been implemented with:

- a rigid distributed architecture supported by filtering the behaviour of the field devices,
- a visible architecture, by extending the behaviour filters to the whole architecture through automation modules,
- an interoperable architecture by implementing IAM devices.

Our ongoing work is to reach the adaptable level thanks to autonomous IAM devices.

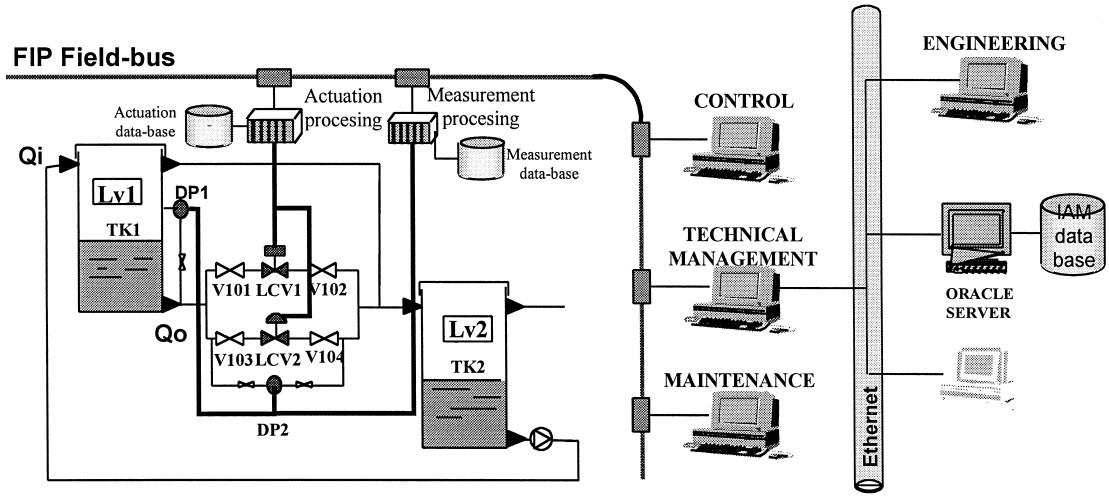



Fig. 3. CRAN experimental platform on CMM & IAM.

### 3.1. From fragmented to interoperable distributed architecture

In a classical architecture, the actuators and transmitters deal with raw signals that characterise physical transformations or observations. These raw signals are independently processed by the control, maintenance and management systems, leading to a fragmented architecture (Fig. 4).

Partitioning between the CMM systems, the actuators and the transmitters is characterised by problems with data consistency (due to possible failures with signal transfer and interpretation) and uniqueness. This has a negative effect on control efficiency and flexibility (systems reconfiguration) as well as on the costs for maintenance operations.

The first improvement consists in providing the CMM systems with validated data by assigning data validation procedures to the field-devices themselves. These validations are processed according to the knowledge the field devices have about their own operation. In other words, the field devices are given additional monitoring tasks that use theoretical models of their functions.

This model, so called behaviour filter [13,14], is able to decide whether to execute (or not) actions following control requests, to monitor the execution of the control functions in order to produce synthetic reports about the device state and status, and to manage its operating modes. For example, the behaviour filter of the modulating valve involved in the level control system is able to accept or reject the positioning request according to its internal state (current position) and status (current availability). It is also able to validate its data by checking their technological validity (threshold) and their functional consistency (comparison of actual with the expected state).

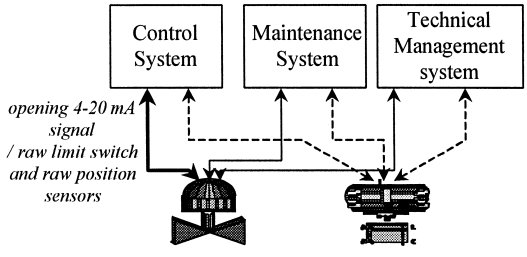



Fig. 4. Fragmented architecture for the level CMM system.

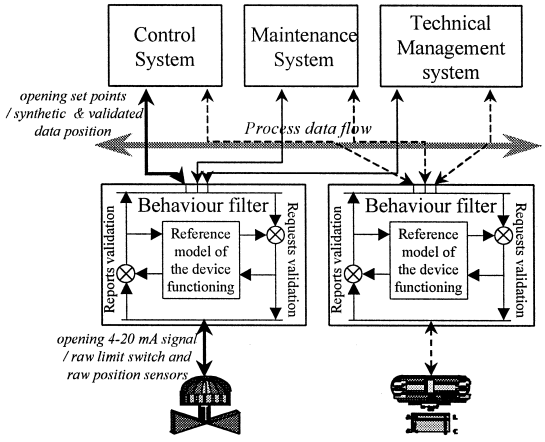



Fig. 5. Rigid architecture for the level CMM system.

This validated data flow can be provided to higher CMM tasks via point to point connection. It would be a major improvement if the data could be shared, through a communication network, for common use by all CMM activities: this would lead to a rigid architecture (Fig. 5).

The challenge in embedding some intelligence in the field devices is in the evolution from processing data towards processing information. Relevant information enriches the process data by additional reports about the functioning of the devices with an external view as seen from their environment. Field devices are expected to process information that relates not only to their own internal operations but also to their functionality within the CMM application.

In order to reach this goal, the process knowledge has to be made available, as far as possible, directly from the field devices. The devices must be able to evaluate their own actions with regard to the complete process application. This knowledge is concerned with the functioning of the process either as described by its physical behaviour or as interpreted by human operators from their experience.

For example, a report produced by the modulating valve about its position should be computed from its own instrumentation but correlated, according to the process physical laws, with upstream and downstream measurements. Set points given to the valve could be rejected if they do not comply with the state of the entire process.

This capability allows each field-device to understand the actions of other devices so that and optimise its behaviour in the system context. This leads to a visible architecture (Fig. 6). This step can be considered as an extension to maintenance and technical management domains of automation functional modules [3] involving four main functions: *to validate the objectives, to process the reports, to validate the observations* and *to execute the actions*. The first two are closely linked to the process information world in relation to the module functionality while the other two are connected to the physical world in relation to the device operations.

However, organisation of such a modular architecture is hierarchical. The services provided by each device are only triggered by the CMM systems, even if the execution of these services is monitored by the devices themselves with regard to their own operations and their functionality.

This architecture may be completed by giving the devices ability to trigger the control points of complementary services provided by other field devices in order to optimise their actions in the system context. Thus, the coordination of the field devices by the CMM systems is augmented by cooperation between the field devices, leading to an interoperable architecture (Fig. 6).

In this case, a service provided by a field device (e.g., flow control by modulating valve) can trigger one or more complementary services provided by some other field devices (e.g., flow measurement provided by a transmitter), in order to achieve its functionality.

At this step, a major point has to be noted. The network of possible interactions between the field devices is strictly defined. The structure is modelled through data flows (external requests and reports to/from the devices) connecting the processing nodes (services enabled to be triggered by each field device). This static structure ensures the interoperability of the field devices in the way described in the functional standards proposed by the PRIAM project.

However, giving more responsibility to the field devices in order that they would react to changes in their environment means letting them have the ability to modify in real time the structures of possible interactions with each other devices.

### *3.2. From interoperable to adaptable distributed architecture*

The main challenge in developing agile architecture would be to confer to the IAM system the same capabilities as those proposed for the Holonic Manufacturing Systems [11]. Each IAM field-device should be both self-controlling and self-executing (autonomy) while cooperating with other IAM devices via communication and negotiation abilities [15].

The interoperable architecture of our level control system almost possesses the autonomy properties of

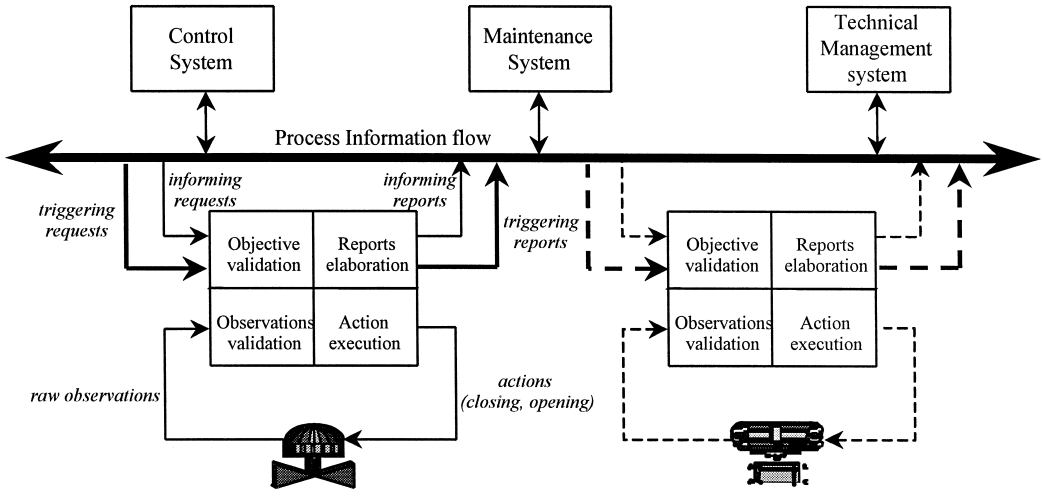



Fig. 6. Visible and interoperable architecture of the level control platform.

holonic systems. However, self-controlling and self-maintaining functions of our IAM devices follow mainly static models (internal or environment) and are not able to react to environmental changes. It still remains to integrate the device self-capabilities with dynamic models of the devices environment, relevant to artificial intelligence (self-learning or reasoning).

To achieve cooperation objective, our interoperable architecture is based on predefined communication (i.e., a predefined network of cooperations). It does not implement negotiation capabilities needed for optimising actions in a changing environment. We need to provide the IAM devices with dynamic reconfiguration mechanisms that would allow cooperative development of mutually acceptable plans and their consequent execution that would achieve desired system functionality.

For example, our modulating valve should modify its internal control rules to adapt to a change of the mechanical valve (from linear valve to a fast opening one).

The architectures presented above cover various scenarios of distribution of intelligence to the field devices. Engineering approach for the development of such architectures is proposed in Section 4.

## 4. CMM & IAM intelligent engineering

### *4.1. Objectives*

Defining the intelligence embedded in a CMM & IAM architecture usually depends on the designer's intuition. In a normative context, this way of working stands in the way of a common specification of requirements for modular system structures. Modular structures are needed for defining complementary services required from IAM, that are to be distributed and implemented in an interoperable and adaptive architecture.

Our objective is to propose an 'intelligent' engineering framework that would progressively refine models based on elementary interactions between process system and CMM system which represent the functionality of the plant. This incremental structure of requirements would systematically leads to modular specifications of interoperable IAM solutions within the CMM systems.

### *4.2. Formalisms for modelling the CMM system & IAM system structuring framework*

According to the Capability Maturity Model [16] proposed as ISO 9000 application guideline for software systems, the CMM & IAM engineering framework has to be either qualitatively defined (up to level 3) in order to be re-usable, or formalised in order to enable quantitative validations and feedback (levels 4 and 5). Our proposal for CMM & IAM engineering framework is presented:

- in the binary entity-relationship model NIAM [8] that allows to capture of semantics using its graphical capabilities,
- using the B [17] method in order to formally verify its properties [18].

Introduced by Nijssen, the NIAM model support the semi-formalisation of binary natural language by defining the objects of a talk and the facts that implicate them. Elementary sentence expresses a fact that relies two objects. Based on set theory, the NIAM entity-relationship formalism is often used for the analysis of information systems.

Introduced by J-R Abrial, the B Method is a formal method for specification, design and implementation of software applications. The B method provides the Abstract Machine Notation (A.M.N.) which is a formal notation used to describe abstract functions in terms of *INVARIANT* defining the properties to be satisfied, *SETS* and *VARIABLES* to be processed, and *OPERATIONS* to modify the variables while maintaining the *INVARIANT*.

The *REFINEMENT* mechanism allows an incremental engineering approach by progressively enriching the initial machine with additional details while maintaining the properties previously proved. The B method also provides structured clauses (*SEE, IMPORT, EXPORT, EXTENDS,...*) which allow a modular specification.

The B proof mechanisms are based on rule-bases inference engine (known as theories) with rule-rewriting and pattern matching facilities. The tool Atelier B, that supports the B method, generates proof obligations [19] that are the underlying hypotheses needed for proving the invariant. These proofs obligations can be automatically proved (if included in the known theories) or interactively with user intervention.

### *4.3. Structured framework for CMM & IAM engineering*

The problem to be solved by the structured engineering framework is the mapping of the functional requirements of the end-users onto the technological description of the process. This problem can be modelled as an interaction between functional agents (human operators and/or automation systems), operating in the world of information whose aim is to achieve the system functionality, and technological objects that have to be physically modified.

Let us consider our example of level control system: a C(M)(M) agent has to modify (to maintain/to optimise) a water stock.

It is modelled in NIAM formalism (Fig. 7a) by two objects (the CMM agent and the process) and a relation between them (CMM agent wants to modify process quantity, i.e., the water stock).

The entities are respectively modelled in the B method by:

- a process machine for describing how the tank level *(level)* evolves according to the output flow *(out_flow)* and to the section of the tank *(sec)* which is assumed to be constant.

**MACHINE** Level_Process
...
**OPERATIONS**
level ← lv_process (out_flow) =
PRE out_flow ∈ NAT
THEN level := (1/Sec) * (((last_out_flow − last_in_flow)*dt) + ((out_flow − in_flow)*dt))
|| last_out_flow := out_flow
|| last_in_flow := in_flow
END

- a control system for managing the physical transformations, i.e., to process the level set point *(level_rq)* against the measured level of the tank *(level_obs)* in order to produce a flow variation set point *(flow_rq)*.

**MACHINE** Level_Controller
...
**OPERATIONS**
flow_rq ← lv_control (level_rq, level_obs) =
PRE (level_obs ∈ NAT) ∧ (level_rq ∈ NAT)
THEN IF level_obs ≠ level_rq
THEN fl. (level_obs ≠ last_level_obs):
flow_rq := fl
ELSE flow_rq := 0
END
last_level_obs := level_obs
END

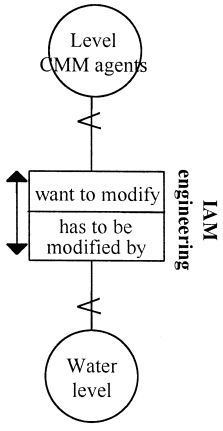



Fig. 7. NIAM model of the level control system.

These two machines are included in a machine whose operation describes the relation between the information and physical worlds, and whose invariant describes the functionality of the process control system.

**MACHINE** Level_Control_System
...
**INCLUDES** Level_Process, Level_Control
**OPERATIONS** Lv_system =
PRE Level ≠ Level_Rq
THEN
Level ← lv_process (Out_Flow)||
Flow_Rq ← lv_control (Level_Rq, Level_Obs)
END
**INVARIANT**
** / variables types * /*
(Level ∈ DomLv) ∧ (Level_obs ∈ DomFlObs) ∧ (Flow ∈ DomFl) ∧ (gap ∈ NAT) ∧ (level_rq ∈ DomLvRq) ∧ (Flow_Rq ∈ DomFlRq)
** / functionality * /*
((Level ≤ Level_Rq + gap) ∧ (Level ≥ Level_Rq − gap))

Solving this process/control interaction is based on the emergence mechanism [20] which stipulates that interactions between two entities define additional properties not included in the properties of each entity.

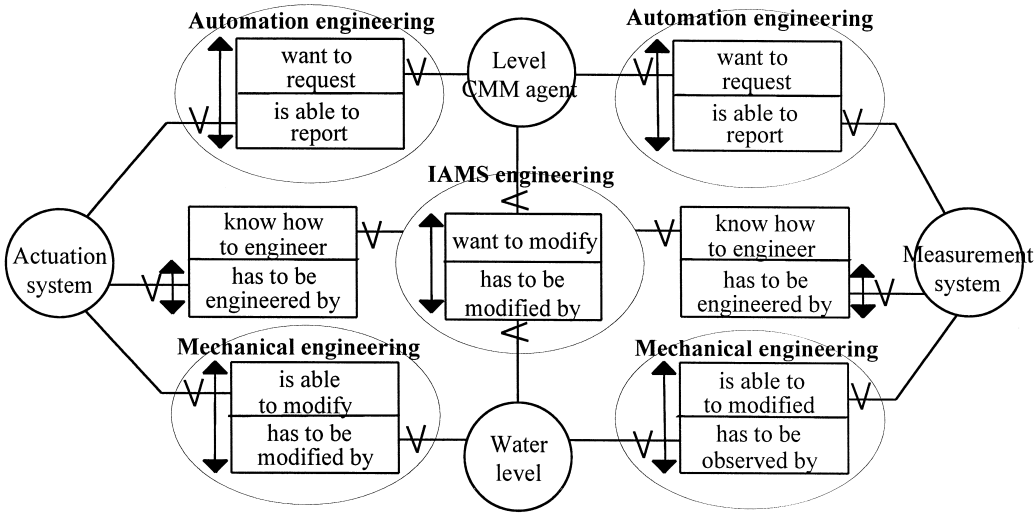



Fig. 8. Emergence of actuation and measurement systems.

Often used to describe emerging behaviour of complex systems, specially in the field of Distributed Artificial Intelligence or Multi-Agents Systems [10], this mechanism has been proposed by Ref. [21] to describe emerging engineering processes for production systems. Application of the emergence mechanism to our level control problem leads to identification of two operators that operate at the interface between the symbolic and physical worlds: actuation operator that transforms an informational CMM request into physical effects and measurement operator that transforms a physical effect into significant CMM information.

According to NIAM formalism, these actuation and measurement operators are defined as entities resulting from the interactions of the previous entities of the model (Fig. 8).

The emergence mechanism is represented in the B method using the refinement mechanism. Indeed, verifying the invariant of the *Level_Control_System* machine generates some proof obligations that characterise type checking between physical and informational flows. Refinement allows to solve this interaction invariant by introduction of additional machines (and operations) related to actuation and measurement operators.

The proof failures encountered by the B prover are used not only to enrich the specifications (invariant and/or operations may not be well specified) but also to systematically and progressively define the control system structure by implementing the emergence mechanism.

This in turn leads to specification of two additional machines respectively related to actuation and measurement and having the functionality to modify the water flow and to measure the level. In the same way as the level control system, these machines also result from an interaction between a CMM agent (who wants to modify the flow or to modify the value of a level observation) and the physical process.

For instance, the actuation machine has the a structure including:

- a control machine *(Flow_Control)* that describes the CMM agent flow requirements, i.e., modification of the water flow by opening a valve *(opening_rq)*,
- a process machine *(Flow_process)* that describes the physical law between the opening of the valve *(opening)*, the flow *(outflow)* and the level of the tank *(level)*.

**MACHINE** Flow_Actuation
INCLUDES, Flow_Process, Flow_Control
**INVARIANT**
... */variables types */ ∧
(dflow ≤ flow_rq + δ) ∧ (dflow ≥ flow_rq − δ)
**OPERATIONS** Flow_act (flow_rq) =

PRE flow_rq ∈ NAT
THEN dflow ← flow_process (opening, level)
|| opening_rq ← flow_ctrl (flow_rq, flow_obs)
|| flow := dflow
END

Definitions of these additional actuation and measurement operators are used to refine the initial specification of the level control system. The first specification is considered as the higher level of abstraction (level control functionality) while its refinement introduces the way this high level functionality will be achieved by actuation and measurement operators.

**REFINEMENT** Ref_Level_control_system
**REFINES** Level_Control_System
**IMPORTS** Flow_Actuation, Lvl_Measurement
**OPERATIONS** Level_system =
PRE Level Level_Rq
THEN Level ← lv_process (Out_Flo)||
Flow flow_act (flow_rq)||
Level_obs level_meas (level)||
Flow_Rq ← lv_control (Level_Rq, Level_Obs)
END

Applying the same structuring concept leads to identification of several refinements (Fig. 9) of product/process interactions. They characterise the automation engineering quantities to be modified (water volume, output flow rate, opening of the valve, position of the valve stem and servo-motor position) and their associated mechanical objects that perform the required modifications.

The benefit of the structured engineering is that it allows modelling of the CMM & IAM systems in terms of elementary objects characterised by mechanical and automation properties. The objects have their own functionality and the physical resources that support it.

Structured engineering provides the end-users, system designers and vendors of a CMM & IAM equipment with: (1) an implementation-independent guideline for specifying the users' requirements which must be systematically collected for each identified agents. This helps the users in ensuring the completeness of their CMM requirements; (2) a data-flow structured model of the CMM services, considered as non divisible according to distribution criteria. These services may be implemented in different, heterogeneous processing devices.

The B method allows the structured formalisation of the services involved in the CMM system and the description of their functionality at a high level of abstraction.

The description of the operational behaviour, such as a PID or other, more complex algorithms, is not relevant to the modelling interoperable architectures. Once the services interoperability has been defined,

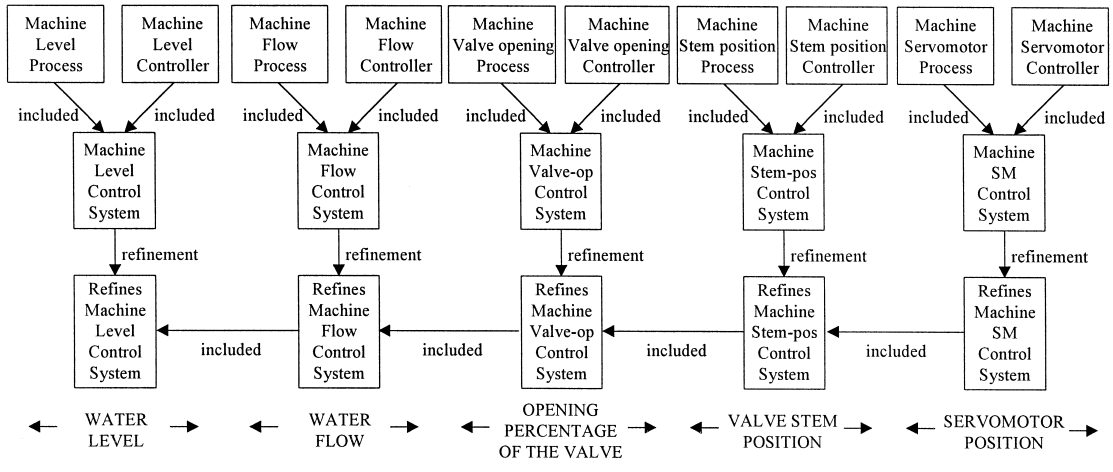



Fig. 9. Architecture of the level CMM system.

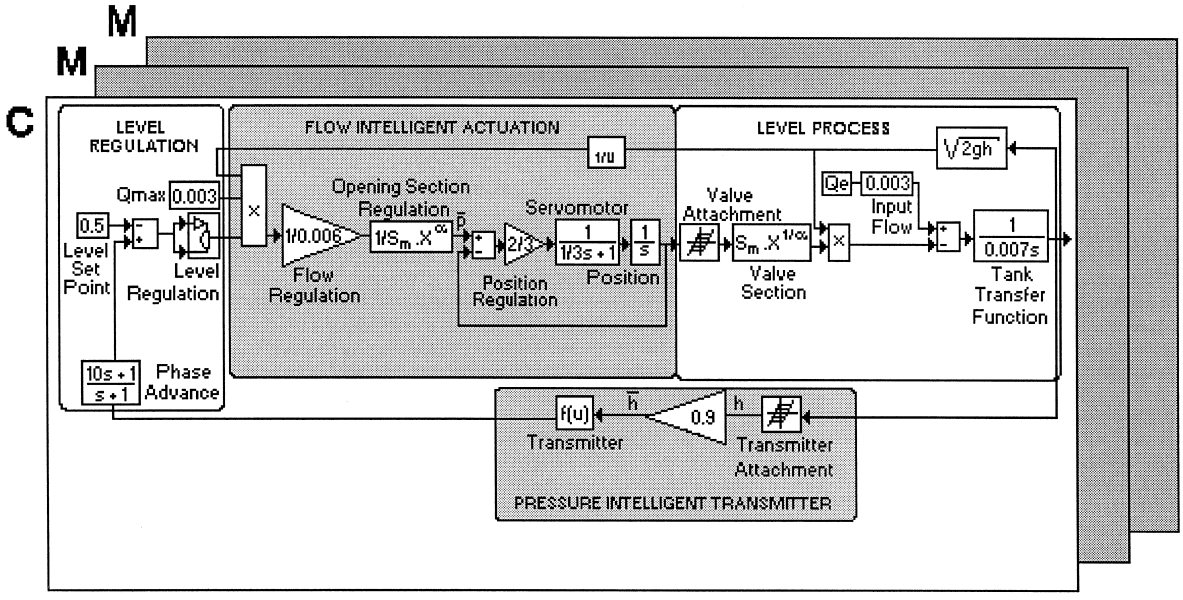



Fig. 10. Matlab/Simulink specification of the functional level control.

the suppliers of the automation devices will define the behaviour of the services that support the interfaces defined previously, using their own technical knowledge.

For example, Fig. 10 shows the structured control description of the level control system which identify functional control loops that are technologically distributed and placed in individual IAM devices. This example shows a possible implementation simulated on the Matlab/Simulink tool.

Specification of such complementary structured services helps in developing functionally interoperable devices. It is also the first step towards interchangeability, allowed by a functional autonomy allocated to the IAM services. Indeed, reaction to a change in the mechanical environment of a CMM agent, as for example the replacement of a valve (linear to non linear), would assume to commute in real-time the associated control algorithm without changing the entire control system.

## 5. Conclusion

From an industrial point of view, using a CMM & IAM system allows:

- easier plant operations that take into account an integrated shop-floor organisation for control, maintenance and technical management,
- mastering of the system engineering by reducing its global complexity, even if each of its individual components becomes more complicated.

The concepts and models mentioned above are being developed, verified and promoted in the ongoing accompanying measures: ESPRIT IV-25525 IAM-PILOT and INCO-DC-96/1744 EIAM-IPE (European Intelligent Actuation and Measurement International Promotion and Exploitation).

Academic analysis of the results of the Esprit projects on IAM has shown that: (1) the intelligent field-devices we developed are provided with a part of the autonomy required by the holonic concept but suffer from a lack of adaptability to react to some environment changes; (2) the engineering approach leading to standardised solutions has to be completed by a structuring framework that can ensure long-lasting IAM solutions conforming to the CMM users' needs.

From the scientific point of view, using a real intelligent CMM & IAM system requires: (1) to move from an Enterprise Integration paradigm to the

Intelligent Systems one, in order to develop more adaptable CMM & IAM [22] architecture; (2) to upgrade the CMM-IAM modelling process by formalising it as much as possible [23], in order to provide a certified and re-usable engineering framework.

This work is expected to be progressed through our participation in ESPRIT IMS-WG no. 21955 Long Term Research working group.

## Acknowledgements

Section 1 of this paper presents results from European Projects. We thank the industrial partners of these projects for their significant contribution.

## References

[1] Proceedings of the workshop on Process Modelling for Enterprise Integration with CIMOSA. Edited by CIM-OSA association, University of Technology in Loughborough, Leicester, UK, March, 1996.

[2] G. Morel, B. Iung, D. Galara, F. Russo, Prototyping a sub-concept of Computer Integrated Manufacturing Engineering (CIME): the integrated Control, Maintenance and Technical Management System (CMMS), International Journal on Intelligent Actuation and Soft Computing—Trends in R&D and Applications, TSI Press Series, Vol. 2, ISBN 0-96274551-5-4, pp. 25–30.

[3] D. Galara, F. Russo, G. Morel, B. Iung, Update on the European state of the art of Intelligent field devices, Proceedings of the International Conference on Intelligent Systems in Process Engineering, SnowMass, USA, AIChE Symposium Series, Vol. 92, 1996, pp. 339–342.

[4] J.B. Leger, B. Iung, Ferro De Beca, J. Pinoteau, A new approach of Distributed Maintenance System: the REMAFEX way of Working, Proceedings of ASI97 (ICIMS-NOE EP III-9251), Budapest, Hungary, July 14–17, 1997.

[5] P. Lorenz, F. Beaudoin, M. Castello, Intelligent transmitters: need for interoperability and interchangeability, Proceedings of the International exhibition for sensors and systems, Nurnberg, Germany, October 1993.

[6] IEC TC65/WG6 (PT1CD2), Function Blocks for Industrial Process Measurement and Control Systems, Part 1, May 27, 1997.

[7] Hatley, D., Pirbhai, I., SA-RT, Strategies for real-time sytems specification, Dorset House Publishing, 1988.

[8] Nijssen, G.M., Halpin, T.A., Conceptual Schema and Relational Database design, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[9] O. Seghrouchni, J.D. Decotignie, Formal model for a real-time distributed computer control system on field-bus, 2nd international conference on Industrial Automation, Vol. 1, Nancy, June 7–9, 1995, pp. 281–286.

[10] J. Ferber, P. Carle, Actors and agents as reflective objects: a Mering IV perspective, IEEE Transcations on Systems 21 (6) (1991) .

[11] H. Van Brussel, P. Valckenaers, L. Bongaerts, J. Wyns, Architectural and system design issues in Holonic Manufacturing Systems, Proceedings of the 3th IFAC Workshop on Intelligent Systems (IMS '95), Bucharest, Romania, October 24–26, 1995.

[12] T. Goranson, Enterprise Integration Capability Model, Brief report on Workshops 1 and 2, ICEIMT '97.

[13] P. Lhoste, G. Morel, From discrete event behavioural modelling to intelligent actuation and measurement modelling, Proceedings of ICIMS-NOE conference (ASI '96) of Life Cycle Approaches to Production Systems, Toulouse, France, Juin 1996.

[14] M. Combacau, M. Courvoisier, A hierarchical and modular structure for F.M.S. control and monitoring, Proceedings of the 1st IEEE International conference on A.I., Simulation and Planning in High Autonomy systems, Tucson (USA), March 26–27, 1990, pp. 80–88.

[15] J.H. Christensen, Holonic architecture and standards directions, Proceedings of the 1st European conference on Holonic Manufacturing Systems, Hannover, Germany, December 1, 1994.

[16] M.C. Paulk, How ISO 9001 compares with the CMM, IEEE Software, January 1995, pp. 74–83.

[17] J.R. Abrial, The B Book: Assigning Programs to Meanings, Cambridge Univ. Press, 1996.

[18] N. Brown, D. Méry, Environment for refining and developing concurrents programs, FME '93: Industrial Strenght Formal Methods, April 1993.

[19] J.S. Ostroff, Formal methods for the specification and design of real-time safety critical systems, Journal of Systems and Software 18 (1) (1992) 33–60.

[20] E. Bonnabeau, J.L. Desalge, A. Grumbach, Characterising emergence mechanism phenomena: a critical review, International Review of Systemic, Vol. 9, No. 3, ISSN 0980-1472, Dunod publishing, 1995.

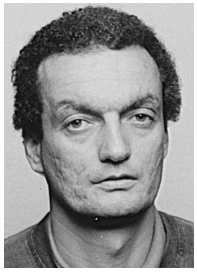
[21] F. Mayer, G. Morel, P. Lhoste, Towards manufacturing engineering based on semi-formal systemic engineering, Proceedings of the 14th international congress on Cybernetic, Namur, Belgium, August 21–25, 1995.

[22] E. Neunrether, B. Iung, G. Morel, J.B. Leger, Engineering process modelling of an Intelligent Actuation and Measurement System: from users' needs definition to the implementation, Proceedings of IMS'97, Seoul, July 21–23, 1997, pp. 69–74.

[23] J.F. Pétin, G. Morel, D. Méry, P. Lamboley, Process control engineering: contribution to a formal structuring framework with the B method, in: D. Bert (Ed.), Lecture Notes in Computer Science, Vol. 1393, B '98: Recent Advances in the Development and Use of the B method, Springer-Verlag, ISSN 0302-9743, pp. 198–209.

**Jean François Pétin**, obtained his PhD degree in Automation Engineering from the University Henri Poincaré-Nancy I in 1995. He has belonged to the team 'Production Integrated System Engineering' of the Automatic Research Centre of Nancy (CRAN) until September 97. He currently works at a postdoctoral position as CNRS research engineer. His research focuses on the formalisation of a structured engineering framework for the integrated production systems based on embedded intelligence. He participated to the project ESPRIT III-6188 PRIAM and manages a work-package in the project ESPRIT IV-23525 IAM-Pilot which promotes the industrialisation of intelligent field devices.

**Benoît Iung**, Dr. Benoît IUNG obtained his PhD degree in Manufacturing Engineering from the University of Nancy I in 1992. He is currently an associate professor at C.I.M. engineer school (E.S.I.A.L.) where he ensures the responsibility of the last year. Since 1987, he has belonged to the Automatic Research Centre of Nancy (CRAN) and more precisely to the team 'Production Integrated System Engineering'. His research concerns mainly the dependability of the production system by promoting the methodological and technological compromise between integration and distribution of 'intelligent' components. He participated and managed tasks in several European projects (ESPRIT II-DIAS, ESPRIT III-EIAMUG, ESPRIT III-PRIAM, ESPRIT IV-REMAFEX, INCO-DC EIAM-IPE 961744 with China). Dr. B. Iung is also involved in the European IMS program through its responsibility in the ESPRIT IV IMS-WG no. 21955.



**Gérard Morel**, is Professor at the University Henri Poincaré-Nancy I and Director of I.S.I.A.L., an institute that coordinates eight scientific Master Degrees on Engineering Sciences. He is responsible for the Production Engineering Group of the Automatic Research Centre of Nancy (CRAN) that coordinates four research teams in the field of Manufacturing engineering and Discrete Event Systems automation. He also manages one of these four research teams working on the Engineering of Production Integrated Systems. He is currently involved in the IFAC Technical Committee on Advanced Manufacturing Technology (MIT-TC) and on Social Impact of Automation (IEN-TC).

INT. J. PROD. RES., 15 JULY 2004, VOL. 42, NO. 14, 2839–2858

Taylor & Francis
Taylor & Francis Group

# Pragmatic approach for modular control synthesis and implementation

D. GOUYON†*, J. F. PETIN† and A. GOUIN‡

Within the framework of Supervisory Control Theory, synthesis algorithms enable automatic generation of control rules from behavioural models of the process to be controlled and of goals to be achieved. The paper highlights a pragmatic use of these algorithms within an automation engineering context and focuses on two key issues: process and goals modelling, and supervisory controller implementation. In this way, the approach presented combines the synthesis techniques and algorithms with an object-oriented automation method that supplies guidelines for the analysis, design and implementation of a modular control system. This pragmatic approach is applied to the synthesis and implementation of the control of an assembly station. The paper highlights the difficulties of the modelling step and its great influence on the synthesis result itself, and also demonstrates the feasibility of the synthesis approach through a successful implementation in conformance with the International Electrotechnical Commission 61131-3 standard.

## 1. Introduction

Automatic synthesis of control systems, as proposed by Fusaoka *et al.* (1983), consists of defining the (unknown) control rules of the (known) dynamics of a physical system, and starting from the behavioural goals (known) to be met while satisfying the following condition: *Control rules* $\wedge$ *Dynamics* $\supset$ *Goal*. In this way, Supervisory Control Theory (SCT) (Ramadge and Wonham 1987) defines a formal framework aiming to analyse Discrete Event Systems (DES) and provides algorithms that enable an automatic synthesis of supervisory controllers in such a way that the controlled plant behaves according to some given goals. Even if these various algorithms (Wonham and Ramadge 1987, Kumar *et al.* 1991) have demonstrated their efficiency, industrial applications of synthesis techniques in the area of automated manufacturing systems remain very limited.

To explain this situation, several reasons can be pointed out (Zaytoon and Carré-Ménétrier 2001): the synthesis algorithms demand correct models of the system and of its goals that are not easy to capture in an industrial context (Morel *et al.* 2001), synthesis may lead to supervisory controllers having an unrealistic size due to a possible explosion of the state space, and the model interpretation given by SCT is more permissive than the one required to implement deterministic control systems.

To cope with these difficulties, the approach proposed in the present paper takes advantage of combining the SCT formal framework and synthesis algorithms with

Revision received March 2004

†Centre de Recherche en Automatique de Nancy, UMR 7039, CNRS, Université H. Poincaré, INPL, Campus scientifique, BP239, F-54506 Vandoeuvre-lès-Nancy Cedex, France.

‡Laboratoire d'Automatique de Grenoble, UMR 5528, CNRS, INPG, Université J. Fourier, ENSIEG, BP46, F-38402 St Martin d'Hères, France.

*To whom correspondence should be addressed. e-mail: gouyon@cran.uhp-nancy.fr

*International Journal of Production Research* ISSN 0020–7543 print/ISSN 1366–588X online 
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540410001705194

automation engineering methods based on the reuse of control software objects (Combacau and Courvoisier 1990, Elkhattabi *et al.* 1992, Lhoste and Morel 1996, Feliot and Staroswiecki 1998, Tiller 2001). It leads to a pragmatic approach where modular supervisory controllers are obtained by applying an iterative algorithm of synthesis and are implemented inside Programmable Logic Controllers (PLC) by translating them into functional control blocks in conformance with the International Electrotechnical Commission (IEC) 61131-3 standard (1993) recommendations.

The paper is organized as follows. Issues about the process and goals modelling as well as implementation in the SCT framework are given in section 2. In section 3, an approach combining automation engineering methods with synthesis techniques is presented. In section 4, this approach is applied to a station of the AIP-Primeca Lorraine experimental assembly system. Results of this application, especially those dealing with the modelling step and its great influence on the synthesis result itself, are discussed in section 5. Some conclusions and future works are drawn in section 6.

## 2. Modelling and implementation issues within the SCT framework

### 2.1. *SCT framework*

SCT (Ramadge and Wonham 1987) provides a formal framework for DES analysis based on the models of the process to be automated (called a *generator*) and the supervisory controller (called a *supervisor*). The process model is described by an automaton of the following form:

$$G = (Q, \Sigma, \delta, Q_m, q_0)$$

where $Q$ is a set of states $q$, $\Sigma$ is a non-empty set of event labels called an *alphabet*, $\delta$ is a transition function described by states transitions, $Q_m \subseteq Q$ is the set of *marked* (terminal) *states* and $q_0 \in Q$ is the initial state.

It is assumed to 'generate' spontaneously controllable and uncontrollable events. Controllable events are those whose occurrence can be disabled while uncontrollable events cannot be prevented and are permanently enabled.

The supervisor can affect the behaviour of the process model by enabling or disabling controllable events to maintain the process in a space of acceptable states for a given specification. It is described by an automaton of the following form:

$$S = (X, \Sigma, \xi, X_m, x_0)$$

where $X$ is a set of states $x$, $\Sigma$ is the alphabet used by G, $\xi$ is transition function, $X_m$ is the set of *marked states* and $x_0$ is the initial state.

In this framework, synthesis algorithms (Wonham and Ramadge 1987, Kumar *et al.* 1991) generate automatically the optimal supervisor that enables the maximum set of events that do not contradict the goal specifications. These specifications define the enabled sequences of events that belong to the G alphabet. Considering the generator model as the physically possible sequences and the goal specifications as the legal sequences, the synthesized supervisor is expected to inhibit the process behaviour so that only desirable sequences are generated. Benefit is that the generated supervisor is correct, by construction, with regards to the specifications and is deadlock-free.

2.2. *Modelling issues*

Executing synthesis algorithms requires concrete and detailed models of the process behaviour and of the goal to achieve them, which are more or less considered as given inputs of the control design activities. However, practical automation of complex systems often relies on a progressive refinement of the models (Brandin *et al.* 2000, Hiraishi 2001) that are not considered within the engineering process as fixed points. Indeed, the modelling of systems of industrial complexity requires a preliminary analysis based on the use of abstract and more or less formal representation formalisms under a shape of function, set, etc. allowing one to grasp, in a pragmatic, intuitive, even qualitative way, the global functioning of a system to be designed (Marikar *et al.* 1998).

If one considers the SCT formalisms and methods for process and goal modelling, it clearly appears that:

- more structured representations such as Petri nets (Xie 1994), synchronous languages (Marchand *et al.* 2000), or predicate (Sanchez and Macchieto 1995) and temporal (Fusaoka *et al.* 1983) logics would be preferable to capture the behaviour of complex systems; and
- the definition of a methodological framework that could help the designer to transform progressively the end-user's requirements and the plant operations into formal specifications of the expected behaviour and process dynamics should be very helpful to facilitate the use of synthesis techniques within an automation life cycle.

A well-known answer to these needs consists of decreasing the complexity of the modelling phases by promoting a decomposition strategy that leads one to introduce modularity and hierarchy within the SCT framework. In the field of supervisory control, the interest of modularity has been widely demonstrated through extensions of the theoretical frame that propose the following:

- Distributed (figure 1) (Fen and Wonham 1990) or hierarchical (Zhong and Wonham 1990, Gohari and Wonham 1998) decomposition of the supervisory controllers to be synthesized that implicitly requires a decomposition of the associated goals specifications.

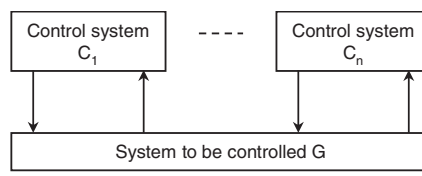



Figure 1. Distributed supervisory control (Fen and Wonham 1990).

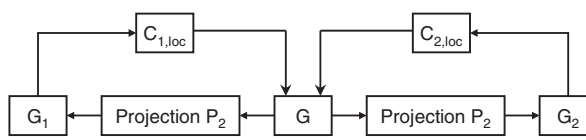



Figure 2. Decentralized supervisory control.

- Modular modelling of the process (*generator*) to be controlled (Yoo and Lafortune 2002) (figure 2).
- Mixed approaches where both process models and supervisory controllers are modelled in a modular way (Chafik and Niel 2000).

Whatever the type of decomposition, the objective is mainly to decrease the explosion of state space (De Queiroz and Cury 2002) generated by the algorithms of synthesis, notably by processing these algorithms on smaller models that involve a reduced number of states and transitions. In this way, the chosen criteria for the decomposition of the models are those that lead to a set of automata that minimizes the intersections between their alphabets, in order to synthesize, in an independent way, several supervisors of smaller size.

The benefit of these approaches is real to master the complexity of the modelling phase. However, focusing the decomposition rules on the only problem of state space size hides more or less the methodological aspects involved by modularity and hierarchy within control systems.

### 2.3. *Implementation issues*

Implementation issues must be addressed to use synthesis techniques fully within an automation life cycle. The main problem results from the different interpretations of the supervisor model given by SCT and by traditional automation methods for the design and implementation of real-time control systems.

Indeed, within the framework of the SCT, the process (generator) is supposed to generate events in a spontaneous way; the only way for the supervisor to affect the behaviour of the process is then to enable or disable the controllable events. Moreover, the synthesis algorithms (Wonham and Ramadge 1987, Kumar *et al.* 1991) provide the maximal permissive supervisor that maintains the process behaviour in legal states and sequences for a given specification. Therefore, the result can be described by automata where several sequences of controllable events are enabled from a given state.

Nevertheless, a reactive control system is expected to force some events to occur and not only to enable and disable some of them. It is often based on a set of predetermined evolution rules that calculate the appropriate actions (controllable events) to be applied on the process according to observations (uncontrollable events) of its current state. This computation is said to be deterministic in the sense that a given sequence of observations always generates the same sequence of actions.

These various interpretations of what is called *supervisor* by SCT and *controller* by the approaches of the forcing events (Marikar *et al.* 1998) require some additional features to make these two notions compliant with implementation issues.

A first solution considers that real systems require the addition of an external control agent that forces some events to occur. It consists in modifying the SCT theoretical framework by adding an intermediate model dedicated to the reactive and deterministic process control (figure 3) (Charbonnier *et al.* 1999). In this case, process and control models are supposed to be known. A generated supervisor aims to ensure that a process associated with its control behaves according to the given specifications. This approach is too restrictive for the initial objectives of synthesis and implementation.

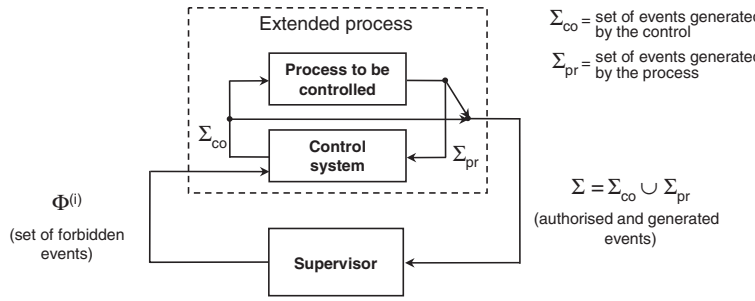



Figure 3. Supervisory control principle according to Charbonnier *et al.* (1999).

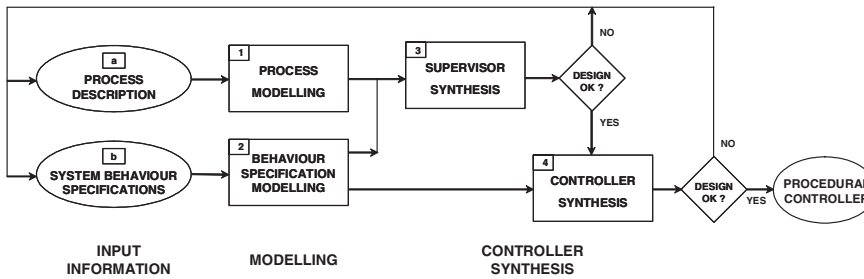



Figure 4. Controller synthesis procedure (Sanchez and Macchieto 1995).

Another approach consists of preserving the SCT theoretical framework while introducing an input–output interpretation of the SCT controllable and uncontrollable events (Balemi *et al.* 1993). This reactive interpretation requires that every supervisor sequence must have at least one controllable event between two uncontrollable events. Indeed, it means that the controller must react to a given input by emitting an output before a new input occurrence. Assuming this hypothesis is verified, this interpretation can be applied for implementation of SCT supervisors (Brandin 1996, Niel *et al.* 2001).

At least more direct approaches that can transform an SCT supervisor into a controller have been proposed (Sanchez and Macchieto 1995, Fabian and Hellgren 1998, Marikar *et al.* 1998) (figure 4). This transformation requires removing the non-deterministic choices contained in the supervisor and translating the evolutions rules of a finite state automaton in terms of target languages such as the ladder diagram of the IEC 61131-3 standard (1993). The benefit is that the SCT approach is kept without any additional interpretation constraints. However, to guarantee that the controller maintains the properties of the supervisor, such as controllability and deadlock-freeness, a formal translation mechanism is required. Most of the above-mentioned approaches are based on the representation of automaton behaviour in terms of algebraic equations and a partial order relation between equations for their implementation.

The approach presented herein conserves the basic foundations of SCT. It aims at defining a methodological framework for using this theory within a real automation life cycle (figure 4). More particularly, modelling and implementation issues will be addressed by combining existing automation methods that provide guidelines for a structured design and implementation of control systems and the SCT framework.

## 3. Approach for modular control synthesis and implementation

### 3.1. *Automation engineering*

During the last decade, industrial practices in automation engineering have promoted the reuse of 'on-the-shelf' control components in the same way as generic hardware components in electronics. From a technical point of view, this is justified by the development of distributed control system (remote I/O, fieldbus, intelligent actuators and sensors) and by the introduction of the functional block concept in the programming languages such as those supplied by the IEC 61131-3 (1993) or IEC 61499 (2000) standards.

Automation methods have followed the same evolution. Object-oriented reasoning that included modularity and hierarchy in the design of control systems has been widely explored (Combacau and Courvoisier 1990, Elkhattabi *et al.* 1992, Lhoste and Morel 1996, Feliot and Staroswiecki 1998, Pétin *et al.* 1998). These methods are more or less based on bottom-up decomposition of a control system in terms of functional elementary modules that ensure the control and the monitoring of their associated resources. In these bottom-up approaches, structured design starts by the process modelling.

Manufacturing systems appear to be built with similar technological elements—cylinders, pumps, motors—involved in the various actuation and measurement processes. Most of the time these field devices are standard and can be associated with a logical behaviour independent from their context of use. It results that the behaviour of such a manufacturing system can be represented as an orderly network of interconnected elementary behaviours (Tiller 2001) that could be plugged from a library of standardized components.

An original application of this work considers that a field-device behavioural model can be used as a model for low-level control and monitoring. It modifies the classical distribution 'control system/process system' by adding a local control of the field devices acting as an interface between technology and functional features (figure 5) (Vogrig *et al.* 1987) in order:

- to filter the functional requests submitted by the control system and to compute appropriate actions to be applied on the technological device in such a way that these actions are compliant with the current state and status of the device; and

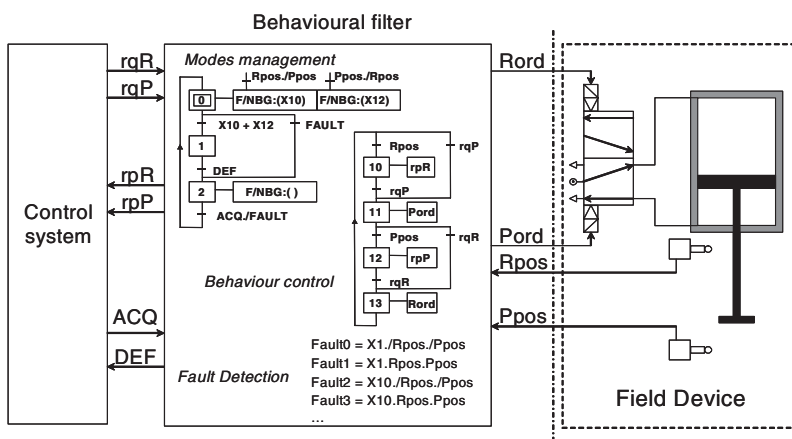



Figure 5. Device control placed as an interface.

- to filter the observations supplied by the device sensors and to compute reports about the state and status of the device to be sent to the control system in such a way that these reports are consistent with the expected behaviour.

In other words, the 'behavioural filter' constitutes a step towards autonomous agents that can manage their mission according to the current state of their resources, to monitor themselves and to elaborate a synthetic report about their functioning state. It could be implemented on supports such as PLC, distributed remote I/O or embedded electronics in the case of intelligent actuators and sensors. This clear distinction between what refers to functional control objectives of the application and what depends on the technology of the field devices is efficient to increase control flexibility and adaptation to changing needs and technology.

Generalization of this approach at less technological levels leads to iteratively aggregate basic behaviours to build more complex ones in a bottom-up manner. The resulting hierarchical architecture involves a set of coordinated and/or cooperative modules having to control and monitor the execution of their own mission (Combacau and Courvoisier 1990, Elkhattabi *et al.* 1992, Lhoste and Morel 1996). These modules, called an 'automation object', can accept, or not, the requests they receive, to choose the appropriate actions they will transmit to their resources (lower level modules), to monitor the execution of these actions, and to report to the higher level modules.

### 3.2. *Modular control synthesis*

The present approach combines the iterative way of thinking proposed by the automation object-oriented methods and the modular synthesis techniques:

- Criteria used for structuring the control system are not only driven by state-space explosion issues (De Queiroz and Cury 2002), but also must be given by the structure of the physical process itself; it leads to an iterative bottom-up design starting from the field device models.
- Synthesis algorithms (Wonham and Ramadge 1987) will be used to generate automatically a supervisor associated with a module (or *automation object*) of the control architecture; for a given module, the supervisor will be computed from a specification describing the mission allocated to this module and from the process model (*generator*) given by the lower level supervisors associated to its resource modules.

Like a behavioural filter, these supervisors ensure the orderly occurrence of events into the supervised modules by filtering functional requests and observations to separate functional aspects from technological ones and to guarantee consistency with an expected behaviour.

Consider the synthesis of the supervisors associated with the control of the field devices. These supervisors (denoted S1 and S1′ in figure 6) are allocated to level 1 in the control hierarchy scale. Two supervisors of the same level do not share events because each part of the system to be controlled has only one supervisor. Supervisors can be synthesized, in a classic way, from the elementary behavioural models of the devices (denoted P1 and P1′ in figure 6) and from specifications describing the rules of filtering (figure 5) presented in section 3.1.

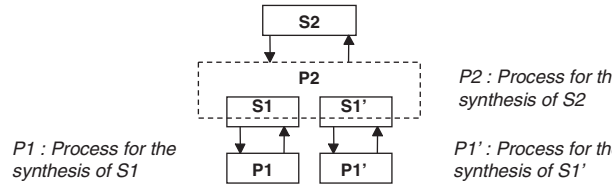



Figure 6. Iterative synthesis process.

A supervisor of higher levels (level $n > 1$) in the control hierarchy scale can then be synthesized from the specification of its goals which are the coordination of lower level resources, and from a process model given by the following:

- Projection of the lower level supervisors (level $n-1$) considered as required resources to keep the only observable events from the given level ($n$).
- Controllability status modification: kept events that were controllable in the lower level supervisors are seen as uncontrollable by the higher level and vice versa.
- Synchronous product of the above projections (they do not share events).

Deadlock freeness is preserved when synthesizing supervisors at level $n$ because:

- level $n-1$ supervisors are deadlock free by construction (property ensured by synthesis techniques); and
- it is assumed that level $n-1$ supervisors have disjoint alphabets; consequently, no deadlock problem may be introduced when coordinating these supervisors.

This last assumption is justified in the case of a hierarchical coordinated architecture where the same level modules do not directly exchange information. However, it is not verified in the case of heterarchical architecture where both coordination and cooperation between same level modules are enabled. In this second case, preserving properties assume conditions, as studied in works on decentralized control (Jiang and Kumar 2000), such as prefix-closure of the alphabets.

Expected behaviour is defined through specification dedicated to each supervisor to be synthesized. Assuming the same hypothesis about the control architecture, these specifications are disjoint. Consequently, conformance of each supervisor at each hierarchical level with its own expected behaviour is preserved by synthesis techniques.

All the operations upon automata—synchronous product, projection, supervisor synthesis—are performed using the software tool TCT developed at the University of Toronto (available at: http://www.odin.control.toronto.edu/DES/).

### 3.3. *Modular implementation of supervisors*

A previous synthesis step results in a hierarchical set of supervisors. The current step aims at implementing these supervisors in a target language supported by most of the PLC (IEC 1993). As discussed in section 2.3, several ways can be used to achieve this implementation objective. Chosen approaches are the ones that provide translation mechanisms from supervisor to controller (Fabian and Hellgren 1998, Marikar *et al.* 1998) without modifying the SCT framework (Charbonnier *et al.* 1999) and its interpretation (Balemi *et al.* 1993).

In these approaches, two main steps have to be performed: translation from the most permissive supervisor into a deterministic controller and coding the controllers into a programmable language. Our proposal adapts the translation and coding rules proposed by Fabian and Hellgren (1998) to take into account the modular structure of the supervisors.

Translation rules have to simplify the supervisors in such a way that the following property is satisfied: two transitions t1 and t2 may exit the same state if and only if the two events associated to both transitions are uncontrollable. This strong hypothesis is justified by the fact that:

- two controllable transitions from a same state mean that two actions are possible from a given situation while a reactive controller has to force one of them; and
- two transitions, one controllable and the other uncontrollable, may generate non-deterministic reaction depending on the sampling period where events are seen.

To avoid these kinds of situations, a mechanism of priority allocation must be applied. In the first case, priority will be given to the event that belongs to an alphabet of the highest level supervisor (in figure 7, *request* has higher priority than *action*); if the two events have the same hierarchical level, allocating priority refers to a designing choice. Whatever is the choice, the controller is nevertheless proved to maintain the process in enabled states. In the second case, priority depends on the controllability of the events: uncontrollable events have higher priority than the controllable ones (in figure 7, *observation* has higher priority than *action*), preserving the reactivity property and the orderly occurrence of events.

Coding rules of such deterministic controllers into the chosen target programmable language, which is ladder diagram of IEC 61131-3, is based on algebraic equations that represent the synchronous activation ($Ai_t$) and deactivation ($Di_t$) of a state ($Si_t$) according to $Si_{t+1} = Ai_t \vee (Si_t \wedge \neg Di_t)$ (figure 8).

Implementation of these algebraic equations on a sequential machine requires an executing algorithm that evaluates 'activation' variables before 'state' variables to ensure non-blocking and reactivity properties of the implemented controller. The chosen algorithm is said to be 'without stability research' and consists, for each

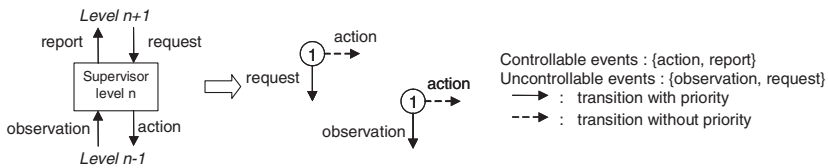



Figure 7. Example presenting the priorities used.

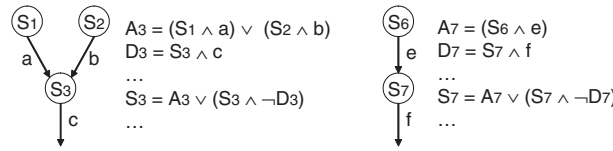



Figure 8. Algebraic translation by activation/deactivation.

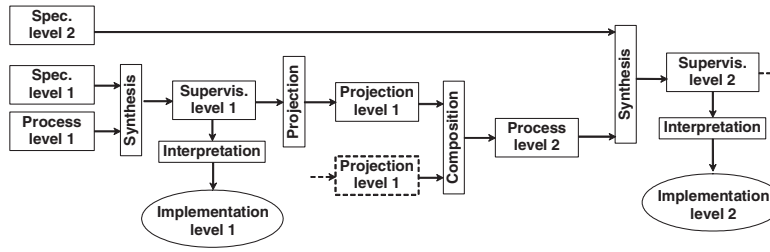



Figure 9. Overview of modular control synthesis.

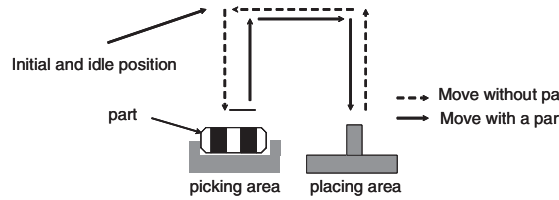



Figure 10. Overview of the pneumatic manipulator.

sampling period of the PLC, in: reading external events (uncontrollable), evaluation of the new situation (Ai and Di calculation), deactivation and activation of states (updating Si), and calculation of internal events and output (controllable) writing.

At least the hierarchical structure of our synthesized supervisors and controllers is coded using the Function Blocks (FBs) notation provided by the IEC 61131-3 standard.

### 3.4. *Overview of the proposed approach*

The above approach is based upon an iterative way of reasoning that leads to a modular control synthesis. It can be summarized by figure 9, which shows the various stages of synthesis, projection, composition and coding of supervisory controllers. The next section demonstrates the feasibility of the proposal through a case study from the initial steps of modelling to the implantation stages.

## 4. Application

Application focuses on the control of a pneumatic manipulator involved in an assembly station of the manufacturing cell of the Atelier Inter-Etablissement de Productique Lorrain (AIP-PRIMECA Lorraine). It allows a product (part) to move from a picking post to a placing post following a 'U' cycle (figure 10). The horizontal and vertical moves are performed by double-acting air cylinders provided with magnetic position detectors and respectively piloted by a 5/2 bi- and monostable solenoid valve. The holding system is carried out by a system of vacuum generators with a Venturi effect. The control system is made up of a Siemens S7 PLC.

### 4.1. *Synthesis of supervisors associated to control of field devices*

#### 4.1.1. *Process models*

The double-acting air cylinders with a control valve are shown (figure 11a) by two marked states, drawn by a double circle, in which the observable devices'

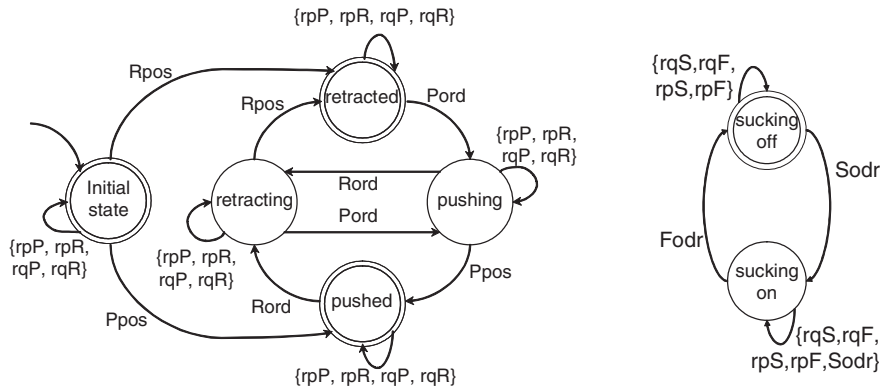



Figure 11. Models of the field devices: (a) air cylinder with a 5/2 bi-stable valves; and (b) holding system.

| Label | Significance | Controllability |
|---|---|---|
| rqR | cylinder retracting request | uncontrollable |
| rqP | cylinder pushing request | uncontrollable |
| rpR | cylinder retracted position report | controllable |
| rpP | cylinder pushed position report | controllable |
| Rord | cylinder retracting order | controllable |
| Pord | cylinder pushing order | controllable |
| Rpos | cylinder in a retracted position (given by detectors) | uncontrollable |
| Rpos | cylinder in a retracted position (given by detectors) | uncontrollable |
| rqS | sucking request to vacuum cups | uncontrollable |
| rqF | freeing request for vacuum cups | uncontrollable |
| rpS | sucked part report | controllable |
| rpF | free part report | controllable |
| Sord | sucking order for vacuum cups | controllable |
| Ford | freeing order for vacuum cups | controllable |

Table 1. Events for figures 11–15.

behaviour is steady (a cylinder in a pushed or retracted position) and by two moving states that represent the evolution between two steady states. The model of the holding system (vacuum generator) is described by two steady states: 'sucking on' or 'sucking off' (figure 11b). As the vacuum generator is not equipped with a held part detector, a part will be considered as being held as soon as the vacuum generator is in 'sucking on' state. A complete list of events is shown in table 1.

### 4.1.2. *Specifications of field devices control*

The specification of the devices' behaviour is done according to the basic filtering functions assigned to the 'behaviour filters' presented in section 3.1 without considering monitoring and fault detection aspects.

For the air cylinders, two automata describe rules that filter or validate some pushing and retracting requests according to the device current state. For example, a pushing request generates a pushing order if the cylinder is not already in a 'pushed state' and if no 'retracting request' is active (figure 12).

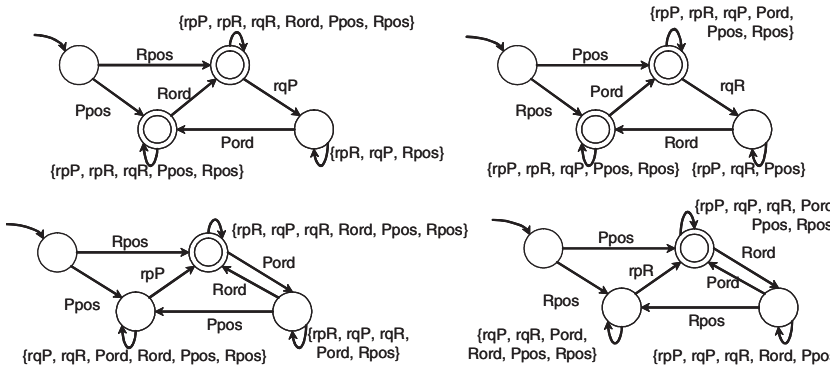



Figure 12. Specifications used for the filter of a cylinder and its bi-stable valve.

Two other automata describe the filtering of observations: an event occurring from a position detector ('retracted', for example) may generate a report (retracted report) only if it follows an order (pushed order) that has been computed and sent to the solenoid valve as a reaction to a validated request received by the device controller (figure 12).

The complete specification of the cylinder controller results from the synchronous product of these four automata (21 states, 68 transitions). The same approach is used for the specification of the vacuum system (eight states, 25 transitions).

### 4.1.3. *Synthesis of field devices control*

From the behavioural models of these devices and their specification, a TCT tool is used to generate the largest (or 'supremal') controllable language that defines the most permissive supervisor (figure 13) as usually done in an SCT synthesis framework.

### 4.1.4. *Implementation of the synthesized supervisors*

Translation and coding rules presented in section 3.3 are then applied to implement three controllers associated with the two air cylinders and the vacuum generator. Algebraic equations are translated into ladder networks (e.g. figure 14) that are implemented into three Functional Blocks of the S7 Siemens PLC.

Each controller has been independently tested on the field devices. The observed behaviour is in conformance with the expected behaviour.

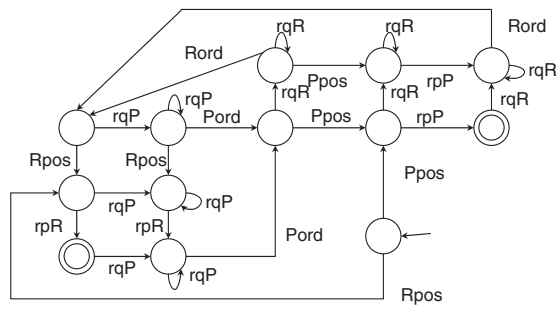



Figure 13. Supreme control of the air cylinder with a 5/2 control valve.

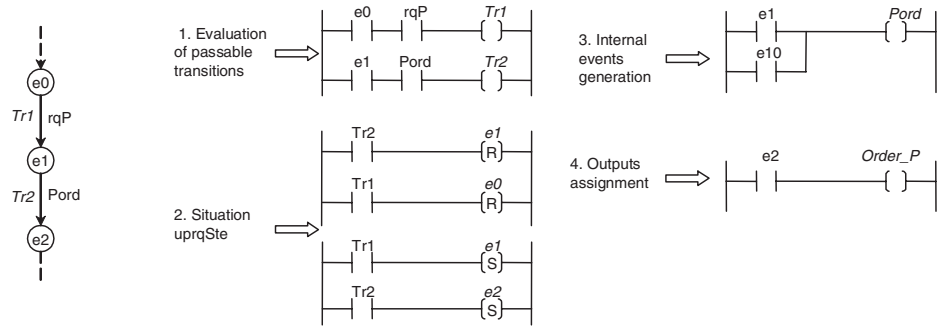



Figure 14. Example of translation into a ladder diagram.

## 4.2. *Coordination module synthesis*

The previous step aimed to synthesize generic controllers usable for a given class of field devices. The next step has to deal with the synthesis of a coordinating supervisor that schedules the requests sent to the controllers of the air cylinders and vacuum cups in such a way that a 'pick-and-place' move is realized. As stated in section 3.2, synthesis of this coordinating supervisor is based on a process model that is automatically built from the local supervisors of the field devices. Figure 15 shows the partial process model built from the supervisor associated with the air cylinder that ensures the horizontal move. This partial process (noted as PPM1) model results from the following:

- Projection of the local supervisors is done by preserving the only events that are interacting with the superior level and controllability status modification (*requests* are now seen as controllable while *reports* are now seen as uncontrollable).
- Instantiation of the generic supervisor to each field device is done by renaming the projected events (e.g. by adding an 'h' for the horizontal cylinder in figure 15) to create a specific alphabet for each instance.

The same procedure is applied to build the other partial process (noted as PPM2 for a vertical air cylinder and PPM3 for a vacuum system). The global process model (noted as GPM) results from the synchronous product of PPM1, PPM2 and PPM3 using the SYNC procedure provided by TCT.

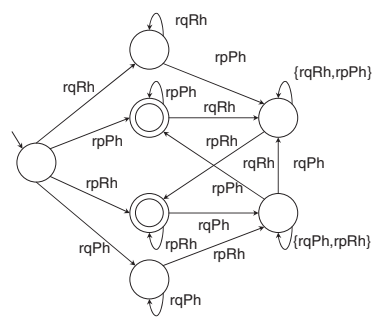



Figure 15. Supervisor projection for a cylinder and its 5/2 bi-stable valve.

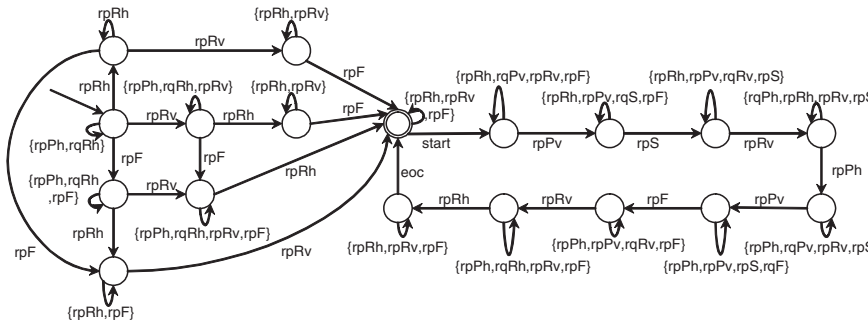



Figure 16. Coordination of the supervisory specification.

The specification model (figure 16) describes THE coordinating rules of the three field devices in order to realize a pick-and-place cycle. In the corresponding automaton, two main parts can be identified: initialization steps toward the initial position and the pick-and-place cycle.

Synthesis of the coordinating supervisor is generated from this specification model and the above global process model GPM. It involves 38 states and 106 transitions: 18 states are devoted to the control of the pick-and-place cycle and 20 states refer to initialization. As already done, the coordinating supervisor is translated and coded into ladder diagrams. They are implemented within an FB that calls the FB of section 4.1.4 (control of the field devices behaviour).

The whole program, including the different FB which were developed using our synthesis approach, has been successfully tested on the manipulator of our assembly cell. However, there are clear limitations, which are discussed below.

## 5. Discussion

The feasibility of our approach has been shown using a realistic case study. However, some modelling and implementation limits have still been encountered. To evaluate their impact on the synthesis result itself, several test cases are proposed on this example.

### 5.1. *Modelling impact on the synthesis result*

When establishing the models of process and specifications, it clearly appears that several solutions, which could more or less differ from each other, are enabled to capture the process behaviour and user requirements. Among the various possible models, test cases are proposed to reflect two problem classes: identification of the marked states and identification of the significant states.

Models involved in those test cases are as follows:

- Two different specifications (F1 and F2 in tables 2 and 3) are proposed for the synthesis of the cylinder controller; they differ by the signification of the involved states and the persistence of the *reports* (figure 17a).
- Four specifications (C1, C2, C3 and C4 in table 3) are proposed for the synthesis of the coordinating module; they mainly differ by their level of permissiveness and by their initialization (figure 17b), noted *init* in table 3.
- For each specification model, two different state markings have been evaluated.

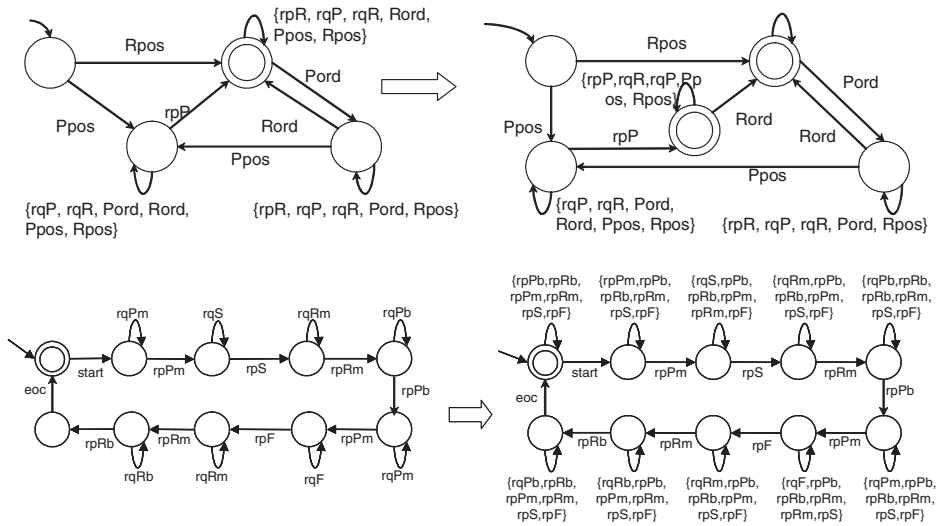



Figure 17. Various examples of test sets to the specifications models: (a) device control specification: addition of a state (report persistence); and (b) coordination module specification: degree of permissiveness in the cycle.

| | Specification F1 | | Specification F2 | |
|---|---|---|---|---|
| Devices control | Normal | All marked | Normal | All marked |
| Bi-stable device | [13,26,2] | [33,80,33] | [13,30,2] | [41,102,41] |
| Monostable device | [13,31,1] | [13,31,13] | [13,35,1] | [13,35,13] |
| Projection bi-stable | [9,24,2] | [11,32,11] | [7,20,2] | [9,28,9] |
| Projection monostable | [8,20,1] | [8,20,8] | [6,16,1] | [6,16,6] |

Table 2. Influence of specifications marking on the result of synthesis.

| | Device specification F1 | | Device specification F2 | |
|---|---|---|---|---|
| Coordination | Normal | All marked | Normal | All marked |
| Spec C1: Non-permissive cycle | [0,0,0] | [0,0,0] | [0,0,0] | [0,0,0] |
| Spec C2: Permissive cycle | [25,63,2] | [29,77,6] | [41,184,2] | [41,184,41] |
| Spec C3: Non-permissive cycle + init | [0,0,0] | [0,0,0] | [0,0,0] | [0,0,0] |
| Spec C4: Permissive cycle + init | [38,106,1] | [38,106,8] | [38,164,1] | [38,164,38] |

Table 3. Influence of the specifications structure on the synthesis result.

5.1.2. *Identification of the marked states*

Marked states are defined by Ramadge and Wonham (1987) as the end of sequences belonging to the marked language $L_m$, where $L_m(G) \subset L(G)$ is a distinguished subset of these sequences that may be 'marked' or recorded, perhaps representing completed 'tasks' (or sequences of tasks) carried out by the physical process that G is intended to model. This definition can be interpreted differently by every modeller with regards to its expectations about process behaviour.

For the model of the field devices, the problem is quite simple. Indeed, stable states of these elements can be clearly identified. For example, marked states of the air cylinder model correspond to a cylinder end of move, i.e. 'pushed' or 'retracted' states (figure 11).

On the other hand, identification of the states that have to be marked is more difficult when modelling goal specifications. For example, which states should be marked in the specification of the pick-and-place cycle: are they limited to the only state that characterizes the initial or idle position of the manipulator, or are they extended to the whole states that characterize the end of a manipulator move (horizontal and/or vertical)?

As far as the synthesis algorithms (and more particularly the TCT procedures) depends on which states are marked, the generated supervisors are far different from each other when a change, even if small, is introduced in the marks of the specification automata. The first two lines of table 2 present two different markings for the specifications of air cylinder supervisors:

- If all the states are marked (note *all marked*), the size of the generated supervisors is not realistic with regards to the given problem; however, even if most of the involved states have no meaning, the controller that can be implemented from this supervisor works successfully; the main benefit is that finding the marked states is a systematic procedure.
- If the marked states correspond to the end of a cylinder move (noted as *normal*), the generated supervisors are more realistic and better correspond to the expected controller; they also require more intuitive interpretation from the modeller.

In table 2, notation [8,20,4] means that the automaton has eight states, 20 transitions and four marked states. Note that the differences between the sizes of the device controllers, according to the different marks, become blurred when a projection is applied to give rise to the process model of a supervisory higher level (two last lines of table 2).

### 5.1.2. *Identification of significant states*

The other problem is related to the various ways in which to write specification models. Take the example of the air cylinder specification. According to the 'behaviour filter' concept, it has to filter the actions and observations to/from the field device according to its internal current state. These filtering constraints can be easily written in natural language as predicates (figure 18). Formalization of these properties as algebraic equations is reachable and Roussel *et al.* (2004) showed that a synthesis procedure is applicable from them. On the other hand, modelling these properties as finite state automata is a more delicate task. How does one capture

**P1.** A 'pushing request' assigned to the 'behaviour filter' generates a 'pushing order' to be applied on the cylinder control valve if and only if the cylinder is not already in a 'pushed state' and no 'pushing request' is active.

**P2.** A 'pushed state' is reported if and only if the evolution of the position detector from 0 to 1 follows an exit order emitted beforehand.

...

Figure 18. Examples of predicates for the specifications.

the significant and marked states? How does one define the required permissiveness given by self-loops?

Considering the various possible answers to these questions, several models can be written to capture the specification of the expected behaviour. Test cases are applied to evaluate the sensitiveness of the synthesis procedure with regards to the specification panel (figures 12 and 16).

Considering device control, the F1 specification differs from the F2 specification by the number of significant states. Indeed, generation of a pushed or retracted report is considered as a fugitive event in one case and as a persistent state in the other. The influence of these different versions on the synthesis result (table 1) seems weak. In the case of *normal* marked states, the supervisor involves the same number of states (13) and few additional transitions (26 for F1 bi-stable instead of 30 for F2 bi-stable). These differences are justified as far as the additional transitions are self-loops that enable the persistent report. On the other hand, these different versions of device control specifications lead to very different coordinating supervisors (table 3). For example, specifications F1 and F2 lead to very different coordinating supervisors ([25,63,2] and [41,184,2]) for a same specification C2.

Among the variations introduced by C1, C2, C3 and C4, the degree of permissiveness seems to be the element that generates the major differences on the synthesis results (table 3). Permissiveness is characterized by the number of events that are enabled in the automata self-loops.

Self-loops of the C1 and C3 specifications involve a single controllable event (see the first automaton in figure 17b) that corresponds to the action that should be performed in a given state. C1 and C3 are then said to be the most permissive ones in the sense that all unexpected events are disabled in the self-loop. These specifications capture the accurate behaviour of the manipulator but do not allow the synthesis of a supervisor (table 3). Indeed, only a single uncontrollable event—the expected one—is enabled for exiting a state; that means that all states are considered as forbidden by the synthesis procedure and are consequently eliminated.

The opposite way of modelling (cf. the second automaton in figure 17b) consists of enabling all the uncontrollable events in the different self-loops. It allows the synthesis of a supervisor, but the relevance of such a specification can be discussed. Indeed, for a given state of the specification, occurrence of any uncontrollable events is considered as normal behaviour of the process. This requires a strong hypothesis, which is not realistic, stating that any detector failures can be avoided.

Defining an alternative solution (C2 and C4 specifications in figure 16), whose permissiveness is relevant for synthesis but nevertheless realistic, remains an uncertain task.

At last, the results obtained using the C4 specification are homogeneous: only few differences concern the number of transitions. It is due to the self-loops involving persistent *reports* that distinguish the F1 and F2 specifications.

More generally, it must be admitted that the pragmatic way of modelling has been driven by the implicit knowledge about what the resulting supervisors have to look like. Indeed, specifications have been written again, through iterative attempts, until the synthesized supervisor, and consequently its controller, look like the ones already implemented. In the same way, the use of formal abstract methods such as the B method allows one to elicit progressively the good invariants of a given problem. Synthesis techniques may also provide help for defining the good models of specification.

5.2. *Impact of the size of supervisors on implementation*

If one looks at the behaviour control of air cylinders, the simplest synthesized supervisor involves 13 states and 25 transitions. Vogrig *et al.* (1987) proposed an equivalent controller for the same device that requires only four states and six transitions of a Grafcet model. The increase of the size between our supervisor and this Grafcet model is still preserved when implementing its associated controller. Indeed, this controller is coded within 55 ladder networks while only 12 ladder networks are needed to code the Grafcet model of the controller.

In the same way, the control of the whole manipulator—three local controllers for the field devices and one for the coordination module—is described through 100 states and transitions and coded by more than 400 ladder networks. When comparing the scale of our case study with complex industrial systems, the size of the synthesized program can still be described as being too big.

## 6. Conclusion and future work

This paper has presented a modular approach for control synthesis within the SCT framework. It relies on the use of automation object-oriented methods that introduce a bottom-up hierarchical reasoning that gradually aggregates the control and process behaviours starting from the most technological levels. It mainly aims at providing methodological help for the preliminary modelling works as well as some coding rules for the implementation on industrial control architecture compliant with the IEC 61131-3 standard.

In this way, major benefits are the systematization of the modeller work and the modularity of the process and goal models. Indeed, process (or SCT *generator*) modelling takes a systematic character as far as the models are built either by instantiation from a library of generic field devices or by projection and synchronous product of lower level supervisors. Specification modelling is simpler because properties to be satisfied are distributed among the various levels of control. The projection mechanism is very useful because it reduces the number of handled states by keeping at every step of the iteration only a subpart of the processed alphabet. At last, implementation is facilitated by the modular structure of our supervisors and controllers, which is close to the architecture promoted by most of the development tools of the current PLC.

Application of the proposed approach using the case study available at the AIP-PRIMECA Lorraine has demonstrated its feasibility. Nevertheless, the writing of the specifications remains a difficult task which has a real impact on the result of the synthesis procedures. Indeed, the last section of the paper has shown that several accurate formalizations of the same specification could lead to very different synthesis results.

In the same way as the *behaviour filter* concept promotes tightly coupled control and monitoring systems, further work should take advantage by taking into account, within the control synthesis, the description of the normal behaviour of a system as well as its incidental behaviour. Indeed, control synthesis including monitoring aspects should lead to a clear distinction between what is referred to as the normal behaviour and what is a failure occurrence. This helps in reducing the problem linked to the definition of the permissiveness degree of a given specification.

The approach is currently being extended to product-driven manufacturing systems. In this kind of modern architecture, the product is supposed to have an active role within the production scheduling. Synthesis techniques will then be

used to generate automatically the different product possible trajectories (*supervisor*) among the available machines (*generator*) in such a way that the performed operations are compliant with the product definition (*specification*). The controllers will be embedded in the product information-processing capabilities and coupled with the machine controllers that are developed according to the modular control synthesis presented.

## References


Balemi, S., Hoffmann, G. J., Gyugyi, P., Wong-Toi, H. and Franklin, G. F., 1993, Supervisory control of a rapid thermal multiprocessor. *IEEE Transactions on Automatic Control*, **38**, 1040–1059.

Brandin, B. A., 1996, The real-time supervisory control of an experimental manufacturing cell. *IEEE Transactions of Robotics and Automation*, **12**, 1–14.

Brandin, B. A., Malik, R. and Dietrich, P., 2000, Incremental system verification and synthesis of minimally restrictive behaviours. American Control Conference, ACC'00, Chicago, IL, USA, pp. 4056–4061.

Chafik, S. and Niel, E., 2000, Hierarchical-decentralized solutions of supervisory control. 3rd International Symposium on Mathematical Modelling, MATHMOD, Vienna, Austria.

Charbonnier, F., Alla, H. and David, R., 1999, The supervised control of discrete event systems. *IEEE Transactions on Control Systems Technology*, **7**, 175–187.

Combacau, M. and Courvoisier, M., 1990, A hierarchical and modular structure for F. M. S. control and monitoring. 1st IEEE International conference on A. I., Simulation and Planning in High Autonomy systems, Tucson, AZ, USA, pp. 80–88.

De Queiroz, M. H. and Cury, J. E. R., 2002, Synthesis and implementation of local modular supervisory control for a manufacturing cell. Proceedings of the 6th International Workshop on Discrete Event Systems, 2–4 October, Zaragoza, Spain, pp. 377–382.

Elkhattabi, S., Corbeel, D. and Gentina, J. C., 1992, Integration of dependability in the conception of FMS. 7th IFAC/INCOM Symposium, Toronto, Canada, pp. 169–174.

Fabian, M. and Hellgren, A., 1998, PLC-based implementation of supervisory control for discrete event systems. 37th IEEE Conference on Decision and Control, Tampa, FL, USA.

Feliot, C. and Staroswiecki, M., 1998, A syntactic approach for functional modelling of physical systems. 9th International Workshop on Principles of Diagnosis, Cape Cod, USA, pp. 174–181.

Fusaoka, A., Seki, H. and Takahashi, K., 1983, A description and reasoning of plant controllers in temporal logic. 8th International Conference on Artificial Intelligence, Karlsruhe, 8–12 August, pp. 405–408.

Gohari, P. and Wonham, W. M., 1998, A linguistic framework for controlled hierarchical DES. Proceedings of the 4th IEE Workshop on Discrete Event Systems, WODES '98, Calgary, Canada.

Hiraishi, K., 2001, Synthesis of supervisors using learning algorithm of regular languages. *Discrete Event Dynamic Systems: Theory and Applications*, **11**, 211–234.

International Electrotechnical Commission, 1993, *IEC 61131-3, Programmable Controllers—Part 3: Programming Languages* (IEC).

International Electrotechnical Commission, 2000, *IEC/PAS 61499-1, Function Blocs for Industrial Measurement and Control Systems—Part 1: Architecture* (IEC).

Jiang, S. and Kumar, R., 2000, Decentralized control of discrete event systems with specializations to local control and concurrent systems. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, **30**, 653–660.

Kumar, R., Garg, V. and Marcus, S. L., 1991, On controllability and normality of discrete event dynamical systems. *Systems and Control Letters*, **17**, 157–168.

Lhoste, P. and Morel, G., 1996, From discrete event behavioural modelling to intelligent actuation and measurement modelling. ASI Annual Conference of the ICIMS-NOE, Toulouse, France.

Lin, F. and Wonham, W. M., 1990, Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Transactions on Automatic Control*, **35**, 1330–1337.

Marchand, H., Bournai, P., Le Borgne, M. and Le Guernic, P., 2000, Synthesis of discrete-event controllers based on the signal environment. *Discrete Event Dynamic Systems: Theory and Applications*, **10**, 325–346.

Marikar, M. T., Rotsein, G. E. and Macchietto, S., 1998, An integrated environment for the design of procedural controllers. 9th IFAC/INCOM Symposium, Nancy, France, vol. II.

Morel, G., Pétin, J. F. and Lamboley, P., 2001, Formal specification for manufacturing systems automation. 10th IFAC/INCOM Symposium, Vienna, Austria.

Niel, E., Pietrac, L. and Regimbal, L., 2001, Advantages and drawbacks of the logic program synthesis using supervisory control theory. 10th IFAC/INCOM'01 Symposium, Vienna, Austria.

Pétin, J. F., Iung, B. and Morel, G., 1998, Distributed intelligent actuation and measurement (IAM) system within an integrated shop-floor organisation. *Computers in Industry*, **37**, 197–211.

Ramadge, P. J. and Wonham, W. M., 1987, Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, **25**, 206–230.

Roussel, J. M., Faure, J. M., Lesage, J. J. and Medina, A., 2004, An algebraic approach for dependable logic control systems design. *International Journal of Production Research*, **42**, 2859–2876.

Sanchez, A. and Macchieto, S., 1995, Design of procedural controllers for chemical processes. *Computers in Chemistry and Engineering*, **19**, S381–S386.

Tiller, M., 2001, *Introduction to Physical Modelling with Modelica* (Dordrecht: Kluwer).

Vogrig, R., Baracos, P., Lhoste, P., Morel, G. and Salzemann, B., 1987, Flexible manufacturing shop. *Manufacturing Systems*, **16**, 43–55.

Wonham, W. M. and Ramadge, P. J., 1987, On the supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, **25**, 637–659.

Xie, X., 1994, Some results on the integration of manufacturing systems using Petri nets. IEEE Conference on Systems, Man and Cybernetics, San Antonio, TX, USA.

Yoo, T.-S. and Lafortune, S., 2002, A general architecture for decentralized supervisory control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, **12**, 335–377.

Zaytoon, J. and Carré-Ménétrier, V., 2001, Synthesis of a correct control implementation for manufacturing systems. *International Journal of Production Research*, **39**, 329–345.

Zhong, H. and Wonham, W. M., 1990, On the consistency of hierarchical supervision in discrete event systems. *IEEE Transactions on Automatic Control*, **35**, 1125–1134.

European Journal of Control (2006)0:114–129


European Journal of Control

# Formal Specification Method for Systems Automation

Jean-François Pétin*, Gérard Morel** and Hervé Panetto***

Université Henri Poincaré Nancy I, Centre de Recherche en Automatique de Nancy, UMR 7039, CNRS – UHP – INPL, Campus scientifique, BP 239, 54506 Vandoeuvre-lès-Nancy Cedex, France

*Currently automatic control deals with the theoretical modelling techniques applied to formally define the behaviour of a control system when the system goals and the process behaviour to be controlled are well defined. Although these approaches are efficient in the design and implementation phases for controlling the dynamics of automatized systems, other tools are also required in the early stages of the process of engineering a system. This paper deals with a specification method aimed at proving that the system goals, as required by the users, are formally refined towards the real target automation system with completeness, consistency, unambiguousness and correctness guarantees. Our specification method is based on the B language to globally verify, from formal constructs, the predicate: Control Systems Requirements* $\wedge$ *Process Systems Requirements* $\Rightarrow$ *Production System Requirements. A case study illustrates our approach and opens issues on the way to industrial practice.*

**Keywords:** Automation Engineering, Formal specification, Formal verification, Requirements analysis, Systems Engineering

## 1. Introduction

Industrial automation systems are embedding Information Technology (IT) intensively to achieve increasingly complex applications. Industrial automation standards, such as the safety-related IEC 61508[1], strongly recommend the use of formal methods to control the complexity of software-intensive applications and their related ease-of-use design techniques [27, 34].

Conceptual and practical approaches have been widely explored by organizations related to computer sciences and automatic control: examples are software verification [9], symbolic [8], timed and probabilistic [30] model checking or automatic synthesis [7]. However, in spite of the consensus that early phases of a system definition are the most important in ensuring that the target system will satisfy the user's requirements, most of these models and tools address the design and implementation phases.

Over these later stages, many systems engineering and automation engineering practitioners [16, 38] consider that the time is ripe to formalize the earlier stages of specification. This means providing a set of guidelines, generic constructs and formal verification tools to establish a non-ambiguous, correct, consistent and complete model of understanding of what a system has to do according to the users' requirements before designing its behaviour according to the multiple engineers' practices and techniques [24].

Writing formal specifications for a given system may be based on an *a posteriori* approach using automatic verification techniques, such as model

*E-mail: jean-francois.petin@cran.uhp-nancy.fr
**E-mail: gerard.morel@cran.uhp-nancy.fr
***E-mail: herve.panetto@cran.uhp-nancy.fr

[1]IEC 61508, Functional safety of electrical/electronic/programmable electronic (E/E/PE) safety-related systems.

*Received 2 April 2004; Accepted 2 December 2005*
*Recommended by A. Giua and M. Steinbuch*

checking [8], as the mean of checking the link between the resulting specification and the required properties. Writing formal specifications may also be based on an *a priori* approach where the proof is used to progressively refine an abstract specification with simple properties to be checked into more and more tangible and precise models. In such a system proof-oriented specification, the models are related by a refinement relationship that is able to ensure specification correctness and consistency by preserving properties through the model transformations.

Considering that the specification activity reflects the heuristic capabilities of human intuition, the process of specifying a system should be able to reuse acquired knowledge to facilitate elicitation of requirements and model definition. To this end, a model-driven specification of automation systems is based on the definition of reusable patterns and constructs that are generic for well-identified problem classes and whose completeness and non-ambiguity have been established once and for all.

This paper combines these two forms of reasoning to propose a formal specification method for automation systems where

- correctness and consistency of the models are ensured using a proof-oriented specification that allows the system goals to be progressively refined and split into process/control subsystems models while preserving the link between the formal specification and required properties (goals);
- completeness and non-ambiguity of these specifications are supported by a model-driven specification that promotes the reuse of acquired knowledge in the definition of specific artefacts for the specific automation domain.

Our work is first founded on Fusaoka's automation predicate [14], which postulates that the design of automation systems consists in defining the (unknown) control rules of the (known) dynamics of a physical system, starting from the behavioural (known) goals to be met:

$$\text{Control Rules} \wedge \text{Dynamics} \supset \text{Goal} \quad (1).$$

This predicate provides formal guidelines that strengthen the proposed formal specification method for automation systems by defining logical relationships between system goals specifications, process specifications and control specifications [18] covering the various points of views involved in automation engineering (dynamics, behaviour, information, communication, etc.):

$$\text{Control Specifications} \wedge \text{Process Specifications} \supset \text{System Specifications} \quad (2).$$

After a brief overview of models and languages that approximately fulfil these requirements above, Section 2 introduces the B language as an efficient formalism to support model-driven specification and proof-oriented specification. A formal specification method based on the B language is then presented: B instantiation mechanisms are proposed to reuse formal automation constructs and correspondence between B objects and mechanisms, and mathematical operators of the predicate (2) are given to support the proof-oriented specification. This section ends with the description of a case study that is used to illustrate the method.

Section 3 proposes an illustration of a model-driven specification through the formalization of generic constructs for physical processes modelling. Generic rules that help in establishing a complete and non-ambiguous process specification are given and applied using the case study.

Section 4 applies, using the same case study, a proof-oriented specification to refine the system goals into process/control sub-systems models while preserving correctness and consistency with regards to the initial properties. A global overview of this specification is given in the Appendix using UML notation.

The final section provides some conclusions and open issues to be resolved in order to put this method into industrial practice.

## 2. Formal Specification for Systems Automation Based on the B Language

Our objective is to propose a formal automation method that combines a proof-oriented specification, based on an incremental reasoning, with a model-driven specification, based on the reuse of generic constructs. To support this approach, we require a formal abstract language that provides support for property verification, proved refinement, specification composition and instantiation.

### 2.1. Candidate Models and Languages for a Formal Specification Method

We briefly describe candidate models, methods or languages that could be more or less able to cover

these various requirements. Formal languages and methods are based upon mathematical models that allow a precise formulation of the properties a model has to satisfy and provide proof mechanisms that allow the verification of these properties.

In the area of automatic control, and more precisely as far as the Discrete Events Systems are concerned, several approaches have been explored for

- the definition of provable models in the design phase, such as analysis techniques for Petri Nets or Finite State Automata [7], theorem proving and model checking [1, 8], or control synthesis [35];
- the verification of PLC programs by applying formal techniques to check the properties satisfied by controllers implemented using IEC 61131-3 programming languages [9, 13, 36].

Even if these approaches have been proved to be efficient in the design and implementation phases, two common characteristics make them inadequate for the specification phase:

- Expression of the underlying mathematical representation is limited to the modelling of system dynamics and hardly covers the description of other system properties as required for users' requirements and automation constructs modelling.
- Most of these formalisms, except Petri nets, do not support incremental modelling, such as formal refinement mechanisms.

At higher level of abstraction, systems engineering approaches or unified languages, such as UML,[2] have proven, in practice, to represent useful views of systems as a result of static, dynamic and functional diagrams, at different phases of the development cycle. Moreover, the concept of knowledge reuse is included in the UML object oriented approach because it contains extensibility mechanisms, called "UML profiles", which can be used to tailor it to specific domains. A "profile" may be defined as a "specification that specialises one or several standard UML meta-models, called reference meta-models". On this basis, Model Driven Architecture[3] [26] provides a set of guidelines, generic knowledge and IT constructs to make the definition of IT systems functionality as independent as possible from any technology platform.

Even if UML provides a wide notation toolbox that covers the requirements in terms of expressiveness equally as well for the system specification as for the modelling and reuse of automation constructs [29], it suffers from a lack of methodological guidelines and formal semantics. Efforts have been made towards UML formalization [11, 22, 17] and formal extensions, such as RT-UML [37] or UMsdL.[4] However, methodological rationales, which should underlay the modelling process with UML, such as an incremental reasoning based on the refinement of the models, are still unresolved issues.

### 2.2. The B Language

Introduced by Abrial [2], the B Method is a formal method for the specification, design and implementation of software applications that support properties proofs. An abstract B model consists of a section defining the mathematical structures related to the problem to be solved and a section containing elements of state variables, operations and invariance properties of the model. Proof obligations are generated from the model to ensure that properties are effectively met. A model is assumed to be closed, and this means that every possible change of state variables is defined by operations.

The B language is founded on

- set theory with classical set operators ($S \cup T, S \cap T, S \subset T, x \in S, \#S$), function and relationship ($A \leftrightarrow B \triangleq \mathbb{P}\, A \times B$);
- first order logic with classical operators of a 2-valued proposal logic ($\neg P, P \vee Q, P \wedge Q, P \Rightarrow Q, P \Leftrightarrow Q$) and quantifiers ($\forall\ X\ .\ p, \exists\ X\ .\ p$).

A B model is defined by the structure shown in Fig. 1.

A model has a name *m*; the clause SETS contains definitions of sets of the problem; the clause CONSTANTS allows the designer to introduce information related to the mathematical structure of the problem to be solved; and the clause PROPERTIES contains the effective definitions of the constants. Another point is that sets and constants can be considered similar to parameters.

The second section of the model defines the dynamic aspects of the state variables and properties of variables using the invariant. Operations $O1 \cdots On$ are used to describe state variable modifications

[2]UML (2003), UML 2.0 superstructure specification, ptc/03608-02, OMG, www.uml.org.
[3]Model Driven Architecture is an OMG trademark, http://www.omg.org/mda/.
[4]UMSDL European ITEA project n° 99028, ITEA office, www.itea-office.org.

```
MACHINE m
SETS s
CONSTANTS c
PROPERTIES p
VARIABLES x
INVARIANT I(x)
INITIALISATION init(x)
OPERATIONS
    O1= Pre P1(x) Then S1(x)
    ...
    On = Pre Pn(x) Then Sn(x)
END
```

**Fig. 1.** B formal model.

owing to generalized substitutions; $Si(x)$ contains the new value of variable $x$ after the execution of operation $Oi$. The substitution of a variable is feasible with respect to its pre-condition or guard. The invariant $I(x)$ states that variable $x$ is always in a given set of possible values, which is assumed to be initialized with respect to the initial conditions and which is preserved by any operation of the list of operations. Conditions of verification, called proof obligations, are generated from the text of the model, and they express underlying hypotheses required for the preservation of invariant properties:

$$(\text{INV1})\text{Init}(x) \Rightarrow I(x)$$
$$(\text{INV2})I(x) \wedge P(x) \Rightarrow I(S(x))$$

(INV1) states the initial condition that should establish the invariant. (INV2) should be checked for every operation $Oi$ of the model; it states that starting from a situation where the precondition of Operation $Oi$ and the invariant are verified, the variable transformation leads to a new value of $x$ where the invariant is still preserved. The B proof mechanism is founded on rule-bases using an inference engine supported by the Atelier B tool.[5] Several proof techniques are available, but the proof tool is not able to automatically prove every proof obligation, and interaction with the proof tool is required. Each proven obligation enriches the set of known theories and can be used for other proofs.

The refinement calculus is used to relate models at varying levels of abstraction by enriching variables and operation descriptions while preserving the already proven invariants. We summarize the approach as follows:

$$(M1, G1) \text{ refined by } (M2, G2)$$
$$\text{refined by } \cdots \text{ refined by} (Mn, Gn)$$

$Mi$ is the $i$-th model iteration that satisfies the goal G$i$ and the goals of lower iteration number. The refined by relationship ensures the preservation of goals, but the proof of refinement must be given. This means that if a new model is derived from ($Mn$, $Gn$), we should prove that the new model refines the previous model and preserves the new properties. Consider a refinement of the machine shown in Fig. 1 given by variable $y$, invariant $J(x,y)$, operation $Oi(y) \triangleq$ Pre $Qi(y)$ then $Ti(y)$, where $J$ represents the formal link between the abstract variable $x$ and the refined variable $y$ and some local properties over $y$. Refinement proof obligations are generated from the following predicates for each operation Oi:

$$I(x) \wedge J(x, y) \wedge Qi(y) \Rightarrow Pi(x) \wedge J(Si(x), Ti(y)).$$

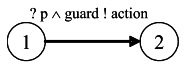
The B language has proved its efficiency for software system development because of its powerful refinement mechanism that supports a complete process of engineering from specification to code, and because of the expressiveness that makes possible the B description of various system views, such as dynamical [20] (see Fig. 2) or informational [17]. Applying the B language for automation engineering has been addressed by academic [20, 28, 33] as well as by industrial trials [4, 21].

### 2.3. System Model-Driven Specification Using the B Language

The objective of model-driven specification is to formalize generic users' and engineers' expertise to provide a collection of generic constructs for a specific application domain. These constructs include, within a meta-model, syntactic and semantic building blocks derived from theoretical representations, or results from experience, as well as their assembly rules. Specification models can then be obtained (see Fig. 3) by instantiating proved constructs, considered as modelling generic primitives, into a model with a well-defined form (syntax) and meaning (semantics). These generic constructs can aid establishing the goals, process or control specifications of the predicate (2).

Generic guidelines in the area of systems engineering [15] can be applied to ensure the completeness and non-ambiguity of the goal specification. Indeed,

[5]Atelier B is a product of ClearSy, http://www.clearsy.com.




```
MACHINE G1
SETS STATE, T /* sets of states and events*/
VARIABLES st1, st2, guard, p, s
INVARIANT st1 ⊂ STATE ∧ st2 ⊂ STATE
OPERATION t(s,p) =
      PRE guard ∧ s ∈ st1 ∧ p ∈ T
      THEN st1 := st1 − {s} || st2 := st2 ∪{s} || action
END
```

**Fig. 2.** B formalization of a state-transition diagram [20].

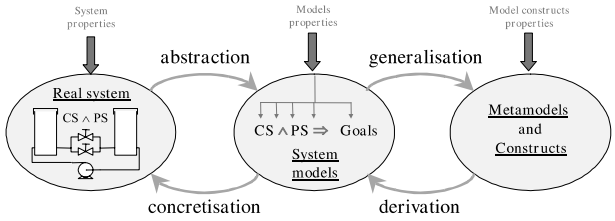



**Fig. 3.** Model-driven specification process.

according to these guidelines, an abstract system model can be said to be composed of elementary processors provided with four kinds of information or material flows, stating what the system has to do, when the system has to do it, what it is required to be able to do it, and what the system requires to know how to do it. In this context, generic constructs result from the formalization of elementary processors and their flow typology. Instantiating these constructs requires filling and identifying all connected flows and, consequently, helps in ensuring the completeness of the resulting specification.

In the same way, the specification of process systems should take advantage of using the Paynter's classification [31] as generic modelling guidelines for the physical variables and rules, to be sure of obtaining a model that is compliant with the physics of a process, expressed in term of causality or material and energy flows balance. Section 3 details the formalization of such expert knowledge.

Finally, as far as the control specification is concerned, methodological expertise and approaches, such as structuring functional models through automation objects covering control, monitoring and technical mode management issues [23] or action-based language for controlling cell manufacturing [39], can be used as inputs for the formalization of control generic constructs that ensures the completeness of the control specification.

Formalization, using the B language, of these constructs is based on the definition of the domain constants and properties through the static part of a B machine, on the modelling of generic behaviour through the dynamical part of a B machine and, finally, on the identification by experts of some invariant of the domain. Note that these invariants' properties can be local to a given construct, but can also define groupings of constructs and rules for valid groupings of constructs.

Reusing the formalized constructs to establish a specification model requires defining an instantiation support for the B language. We propose a two-step mechanism. First, the collection of generic constructs is prefixed with a specific name associated with each instance; this leads to a collection of machines associated with each object involved in the specification to be established. The resulting specification is formalized within a single B machine built as a network of interconnected constructs that call the instantiated constructs using the clause EXTENDS. For example, a generic construct machine is instantiated into instance1.construct machine, which is called by the specification M1 machine through the clause, EXTENDS instance1.construct. The consequence of using the EXTENDS clause is that all the instantiated machines inherit from the clauses constants, sets, variables, invariants, initializations and operations defined in the generic construct machine. More precisely, the instantiated constructs verify the same invariants as those defined in the generic constructs. Moreover, if the constructs' invariants contain the description of grouping rules, the specification model can be presumed to be correct with regards to these. In this sense, formalization of generic constructs efficiently contributes to the completeness and non-ambiguity of the specification.

## 2.4. System Proof-Oriented Specification Using the B Language

Proof-oriented specification for systems automation [28] consists of giving a formal meaning to the automation predicate operators to verify (see Fig. 4a) the following:

- the local properties of each given specification that allows confirmation of their correctness with regards to syntactic or semantics rules;
- the specifications' consistency, especially by proving that the refinement relationship between system goals on one side and target process/control systems on the other side is established.

Models involved in this predicate are formalized through B machines, whereas operators ($\wedge$ and $\supset$ operators) are established with equivalent B structuring mechanisms (see Fig. 4b).

The System requirements machine represents all the users' requirements in terms of abstract functional behaviour expressed using static information, dynamic

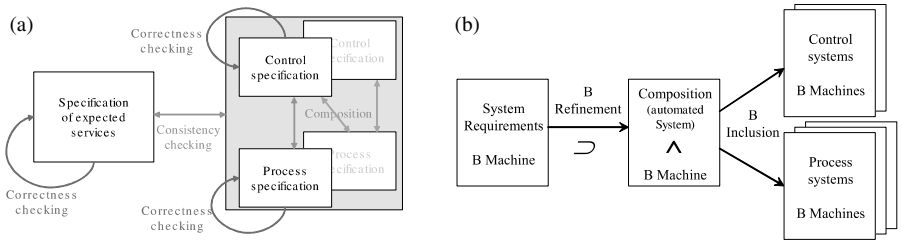



**Fig. 4.** (a) Automation predicate and (b) B correspondence.

information processing and invariant safety properties of a B machine. A very powerful primitive of the B language is used to support this modelling. Consider the following B operation: operation O(v) $\stackrel{\wedge}{=}$ any $w$ where P($w$) then $v := w$, where $v$ and $w$ are machine variables and $P$ a predicate over $w$. This means that the operation result $v$ is equal to any $w$ if $w$ satisfies the predicate $P$. In other words, we are able to describe the properties an expected result has to satisfy (what the system has to do), without describing how this result may be obtained (how the system is doing it).

The B composition machine formalizes the $\wedge$ operator using the B object oriented features, allowing the operations call and the use of public variables. The primitives are used to interconnect B control and process machines that describe the functional behaviour of the components involved in automation systems, such as software control, the physical process or electrical wiring.

The B refinement mechanism is used to prove that the B system requirements machine can be correctly and consistently refined into the composition machine by proving that the initial invariants are preserved.

The resulting specification is then the foundation of the design and implementation phases that can be supported by the B language (for the software systems) or by dedicated techniques, such as Matlab[6] [21] or LUSTRE[7] [4] which can be used for the design of the reactive control part (see Fig. 5).

### 2.5. Presentation of the Case Study

The proposed formal automation method based on the B language is illustrated using the case study (see Fig. 6). This case study is part of an industrial demonstrator which has been developed in European projects for evaluating the interoperability of distributed intelligent actuation and measurement field-devices [32]. The studied sub-system is limited to a tank and a valve. The tank is upstream fed by a water flow and the valve is used to control the level within that tank. Meaning of the variables is the following: $Q_0$ and $P_0$ are respectively the flow rate and the pressure of water flow entering the tank, $V_1$ is the water volume within the tank, $Q_1$ and $P_1$ are respectively the flow rate and the pressure of water flow exiting the tank, $Q_2$ and $P_2$ are respectively the water flow rate and pressure in the circuit located after the valve and which feed the downstream system.

The model-driven specification is illustrated by formalizing generic constructs with reference to process modelling and more precisely to hydraulic systems. These constructs are instantiated for elaborating the specification model of the case study process. According to the predicate (2) and its B correspondence, a proof-oriented specification is then used to ensure the consistency between the goals' specification on one hand and the process and control specifications on the other hand.

## 3. Model-Driven Specification of the Case Study

This section applies a model-driven specification process in the area of physical processes modelling. Recent work in this area [25] has highlighted the benefit of using object-oriented representations to mix theoretical modelling with identification techniques, when necessary. In this context, a physical system is modelled as a network of interconnected processors acting as white boxes in the case of theoretical modelling or black boxes in the case of identification. To facilitate their reusability, most of these processors are described independently of each other and without any causality constraints. However, when building a physical system representation by connecting these independent processors, the modeller has to take into account the following:

- The relationships between input and output flows of each elementary processor, which may be

[6]Matlab/Simulink is a product of The Mathworks company.
[7]Lustre is a modelling and programming language for synchronous reactive systems.

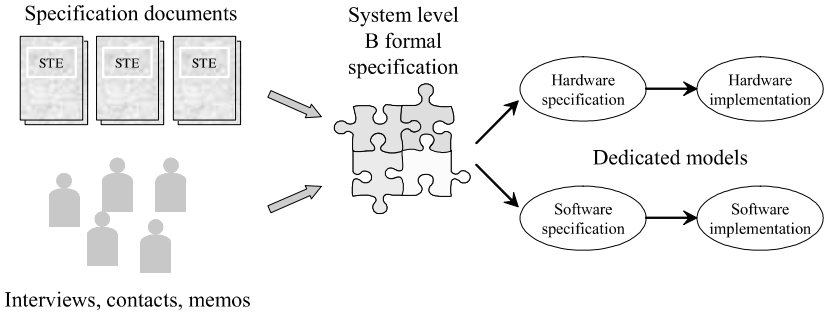



**Fig. 5.** From B specification towards design and implementation [21].

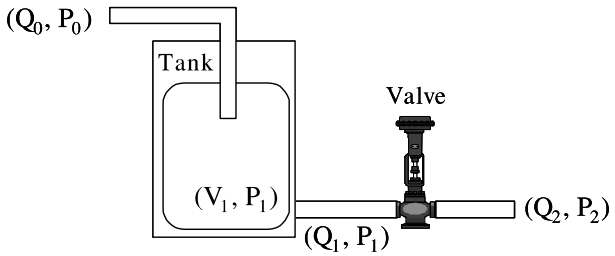



**Fig. 6.** Case study.

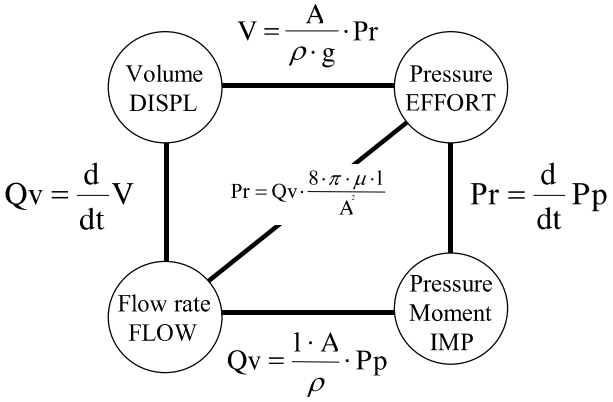



**Fig. 7.** B formal typology of state variables for physical systems.

constrained by some physical rules of conservation (energy or flow balance, specific features of the transformations, etc.).

- The connection between elementary processors, which are physically limited to some enabled configurations (causality relationships between physical variables). In the context of our study, the key issue we wish to address is correctness checking, with regards to physical laws, of the process structure provided by object-oriented representations. Our approach is based on the formalization, using the B method, of some expert knowledge in the area of physical systems modelling and on proven mechanisms that help in re-using the formalized knowledge. More precisely, our work refers to Paynter's classification of the physical variables [31] and its application within a systemic approach proposed by Feliot [13]. We summarize these results before detailing their B formalization.

### 3.1. Theoretical Foundations

Physical variables have been classified by Paynter into four basic sets: effort, flow, impulse and displacement. Applied to hydraulic systems related to our case study, Effort, Flow, Displacement and Impulse variables correspond respectively to variables of pressure, fluid flow, volume and moment of pressure. Paynter's tetrahedron of states (see Fig. 7) provides the relationships between these variables.

Considering that some of those physical variables are tightly coupled, Paynter has proposed considering three main couples: (effort, flow), known as power; (effort, displacement), known as potential energy and (impulse, flow), known as kinetic energy. From these definitions, three main physical processors ensuring the couples' transformation have been identified: from power to power (P/P), with a proportional relationship; from energy to power (E/P), with a derivative relationship, and from power to energy (P/E), with an integral relationship.

Feliot has formally demonstrated [12] that these three processors are strictly equivalent to the three basic processors proposed by system theory: (P/P), (P/E) and (E/P) processors are respectively equivalent to Shape, Time and Space systemic operators. From the process modelling point of view, Shape, Time and Space operators respectively support the physical transformations of a product, its storage and its transportation. Based on system theory, Feliot has proposed some structuring rules for interconnecting these operators.

Our contribution consists in formalizing these theoretical foundations, using the B method, with two objectives:

- to provide basic constructs for the modelling of process systems and more particularly, in the context of our case study, for the modelling of hydraulic systems;
- to provide a formal framework for the modelling of a particular process system using the instantiation of the basic constructs.

### 3.2. Formalisation of Basic Constructs for Process Modelling

As a first level of abstraction, the systemic Feliot's processors (Fig. 8a) are formalized within B machines without taking into account a precise description of the dynamics of the supported transformations. It leads to the definition of three constructs that include the following:

- A definition of the variable couples involved in the transformation.
- The invariant relationships that link the two variables inside each couple, these relationships are those defined by the Paynter's tetrahedron of states.
- a very abstract description of the transformation by only specifying its inputs and output, which must maintain the previous invariant.

Figure 8b presents the B formalization of a Time processor in the area of hydraulic systems where ($Q_0$, $P_0$) and ($Q_1$, $P_1$) represent respectively the input and output fluid of the storage system with two couples (flow, pressure), whereas ($V_1$, $P_1$) represents the volume and the pressure within the tank. In the same way, the Shape processor has been formalized with (Flow, Pressure) as input, and another (Flow, Pressure) as the output, by taking into account the problem of energy dissipation, and the Space processor is defined as transforming (Volume, Pressure) and (Flow, Pressure) into (Flow, Pressure). Note that all the relationships described within the invariant and the operation involve some parameters, such as '$A$' (tube or tank section), which should be further tuned when modelling a particular physical system and when technical features of the various actuators, sensors, and transmitters are known.

As a second level of abstraction, we introduce a qualitative description of the transformation dynamics by detailing the behavioural trends of the transformation (see Fig. 9). Our objective is to focus on the early phase of specification by providing a well-structured process model to be used for defining control goals; therefore, this qualitative description can be said to be sufficiently efficient.

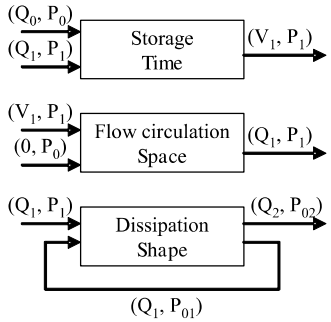
### 3.3. Completeness and Non-ambiguity Checking

Applying formal constructs to specify the physical process of the case study presented in the Appendix consists of first identifying the generic constructs that have to be used. This means that the components involved in the case study process (valve, tank, circuit, etc.) must be associated with a generic processor, as defined by the constructs (hydraulic_time, hydraulic_space and hydraulic_shape processors). Note that the same process component can be associated with several generic constructs. For example, the valve is involved in water transport as a space processor, but also provokes water flow dissipation as a hydraulic_shape processor. Conversely, a generic construct can be associated with several process components: transport, for example, is performed using valves and circuits.

The second step consists of defining the physical process model as a network of interconnected processors that are instantiated from the generic constructs (see Fig. 10, a graphical representation of the network).

This network is formalized in the Case_study_process B machine (see Fig. 11). Hydraulic generic constructs are customized into specific case study processors by renaming constructs' names, variables and operations according to the instantiation mechanism defined in Section 3.3. (Hydraulic_time_processor machine is customized into Tank.Time machine, Hydraulic_space_processor into Valve.Space machine, and Hydraulic_shape_processor into Valve.Shape machine). These machines are then invoked within the Case_study_process B machine by the "extends" B mechanism. Operations are called with parameters that depend on the edges of the network.

Two levels of model verification can then be performed. The first is directly linked to the instantiation process. Indeed, using the B extends clause requires that the invoked operations in the Case_study_process machine satisfy the invariant as defined in the generic constructs when the pre-condition (the assumption required to safely execute an operation) associated with the called operation is true. This pre-condition mainly refers to the typology of physical variables and processors represented in the constructs. We illustrate this mechanism using the example of the Case_study_process machine shown in Fig. 11: the tank.time invoked operation produces a couple




```
MACHINE Hydraulic_Time_processor
SETS EFFORT, FLOW, DISP, IMPULSE
CONSTANTS A, ρ, g
VARIABLES V1,Q0,P0,Q1,P1
INVARIANT
   V1 ∈ DISP ∧ P1 ∈ EFFORT ∧ ... /* type checking */
∧ (V1 = (A / (ρ*g)) * P1) /* invariant relationships */
OPERATIONS (V1,P1) ← Time ((Q0,P0),(Q1,P1)) =
      PRE (Q0, P0) ∈  FLOW → EFFORT   ∧   …
      THEN  ANY newV1, newP1
      WHERE   (newV1, newP1) ∈ DISP → EFFORT
                  ∧ (V1 = (A / (ρ*g)) * P1)
      THEN V1 := newV1 || P1 := newP1
END
```

**Fig. 8.** B formal typology of physical processors. (a) Feliot's processors. (b) B formalization of a Time processor.

| Produced Variables | Output Variation | $P_0$ | $\Delta Q = Q_0 - Q_1$ |
|---|---|---|---|
| $P_1$ | → | → | |
| | ↓ | ↓ | |
| | ↑ | ↑ | |
| $V_1$ | → | | $\Delta Q = 0$ |
| | ↓ | | $\Delta Q < 0$ |
| | ↑ | | $\Delta Q > 0$ |

```
REFINEMENT Hydraulic_Time_processor
VARIABLES Vr1, Qr0, Pr0, Qr1, Pr1
INVARIANT
     (Vr1 ∈ DISP) ∧ (Vr1 = V1)  ∧
     (Qr1 ∈ FLOW) ∧ (Qr1 = Q1)  ∧ ...
OPERATIONS
(Vr1, Pr1) ← Time ((Qr0, Pr0),(Qr1, Pr1 )) =
PRE (Q0, P0) ∈  FLOW → EFFORT ∧ …
THEN
  SELECT (Qr0-Qr1)>0
  THEN ANY newV1, newP1
  WHERE (newV1,newP1) ∈ DISP → EFFORT
   ∧ newV1 > Vr1 ∧ (newV1 = (A / (ρ*g)) * newP1)
  THEN Vr1 := newV1  || Pr1 := newP1
END
```

**Fig. 9.** B formal qualitative behaviours for physical systems processors.

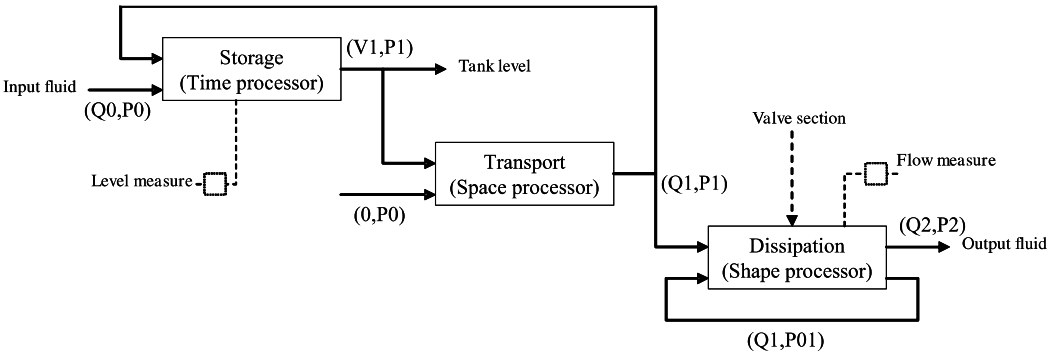
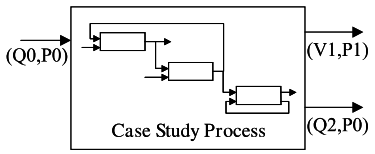



**Fig. 10.** Network of basic processors.

of variables ($V_1$, $P_1$) that is used as input parameters by the valve.space invoked operation. The first level of verification consists of verifying that the precondition within the hydraulic_space construct is maintained true by applying the above input parameters.

The second level of verification concerns new properties that can be locally added to the




MACHINE Case_study_process
VARIABLES $Q_0$, $P_0$, $V_1$, $P_1$, $Q_1$, $Q_2$, $P_{01}$
EXTENDS Tank.Time, Valve.Space, Valve.Shape
INVARIANT $(V_1=0 \Rightarrow Q_2=0) \wedge (Q_2=Q_1) \wedge \ldots$
OPERATIONS
$((Q_2, P_0),(V_1, P_1)) \leftarrow$ Case study process $(Q_0, P_0) =$
$(V_1, P_1) \leftarrow$ Tank.Time $((Q_0, P_0),(Q_1, P_1))$
$\| (Q_1, P_1) \leftarrow$ Valve.Space $((V_1, P_1),(0, P_0))$
$\| ((Q_2,P_2),(Q_1,P_{01})) \leftarrow$ Valve.Shape $((Q_1,P_1),(Q_1,P_{01}))$
END

**Fig. 11.** B formal network of the physical system.

Case_study_process machine. These properties may refer to the following:

- Specific characteristics of the physical system expressed independently of the control or the users' requirements; the invariant of Fig. 11 mentions (a) that no output fluid can be observed when the tank is empty and (b) that the flows out of the tank before (Q1) and after (Q2) the valve are the equal (only the power is changed owing to the energy dissipation).
- Structural properties of the interconnected processes that represent legal sequences with respect to physical laws.

As an example of the last point, Feliot has demonstrated that only some sequences of processes are enabled with respect to causality principles producing a set of legal sequences, such as Time/Space/Shape/Space/Time. Knowing that the Time, Space, and Shape processors are characterized by their input and output couples, the legal sequence is coded into an invariant that checks that the successive input/output processor flows have the correct type as follows:

**INVARIANT** ... /* see Figure 10 */ $\wedge$ $(Q_0,P_0) \in$ FLOW $\rightarrow$ EFFORT $\wedge$ $(Q_1,P_1) \in$ FLOW $\rightarrow$ EFFORT $\wedge$ $(V_1,P_1) \in$ DISPL $\rightarrow$ EFFORT $\wedge$ ... the same type checking for all the operations parameters

To conclude this section, proving a specification of the process system based on instantiated formal constructs means that

- from a structural point of view, the process system model is well structured with regards the physical rules; the network of interconnected processors is said to be compliant with the basic constructs and their assembly rules;
- from a dynamical point of view, the variables involved in the process system behave as expected by the modeller.

This process specification can then be further integrated, by the use of the B common language, into a more complex representation of the system, including the control model and the requirement model as recommended by the automation predicate.

## 4. Proof-Oriented Specification of the Case Study

Proof-oriented specification is then used to ensure the consistency between the goals specification on one hand and the process and control specifications on the other hand according to the predicate (2) and the B correspondences given in Section 2.4. To increase the comprehensibility of the B models, an informal representation of the relationship between these specifications is given in the Appendix using an UML notation. B specification of the physical process system is assumed to be given by the Case_study_process machine of the previous section.

### 4.1. Goals Specification

To illustrate the systemic approach, the requirement specification is proposed to merge classical control problems (such as level control loop and mode management) with information control problem (such as historic management). The System requirements machine (see Fig. 12) defines these services using three operations: level_request, change_mode, and create_historic.

Production functionality is described using the "any ... where ... then ..." primitive described in Section 3.4. The post-conditioned operation describes the initial values of variables to be processed (input_flow, request) and the result to be obtained (output_flow, tank_level). This simply postulates that we have to modify the output water flow (output_flow becomes new_output_flow) if we want the new level

```
MACHINE     System_requirements
SETS        PRESSURE = NAT; FLOW = NAT; VOLUME = NAT,
            MODE={Automatic, Manual}, MODECHG={M2A, A2M}, HISTORIC;
CONSTANTS   Lmax = 100, Input_flow_max = 100
VARIABLES   request, input_flow, output_flow, tank_level, mode,
            historic, chg_mode_historic, monitoring, level
INVARIANT   (tank_level = Lmax ⇒ Output_flow > 0) ∧ …                (I1)
            ∧ historic ⊆ HISTORIC ∧ monitoring ∈ historic → NAT      (I2)
OPERATIONS
Level_control (request) =
    PRE     request ∈ NAT
    ANY     new_output_flow, new_tank_level
    WHERE   new_output_flow ∈ [0, 100] ∧ new_tank_level = request
    THEN    tank_level := new_tank_level || output_flow := new_output_flow END
Change_mode =
    ANY     newmode
    WHERE   newmode ∈ NAT
            ∧ (newmode = automatic ⇒ tank_level ∈ [request - 10, request + 10]))
    THEN    mode := newmode
Create_historic (h) =
    SELECT  newmode = change_mode(mode) ∧ newmode$0 # newmode$0
    THEN    historic := historic ∪ {h}
            || IF newmode = Automatic and newmode$0 = Manual
            THEN monitoring(h) := A2M ELSIF monitoring(h) := M2A
            || level := request END
END
```

**Fig. 12.** B system requirement.

(new_tank_level) to equal the requested level (request). In other words, this abstract substitution simply states the input/output expected modification without any other behavioural details on its realization. Invariant properties of the production activity are given in the invariant (I1), saying that reaching the maximum level of the tank provokes output valve opening, i.e. an output flow other than zero.

Mode management is described using the same primitive, and details the conditions required for changing the production mode. Changing the mode of the system from automatic to manual can be done whatever the physical or control situations are, but changing from manual to automatic requires the system to be near a steady point where a control loop can be applied.

The last operation formalizes the need to monitor and to produce a history of mode commutations. It is done with the help of

- a historic class having two attributes, the type of commutation (from Manual to Automatic or vice-versa noted M2A and A2M) and the corresponding request when the commutation is operated;
- an invariant (I2) stating the relationships between class and attributes;
- an operation describing when a new occurrence of historic class must be created and the value of its attributes (the $ operator in B is used to make the distinction between a variable value before and after execution of an operation).

Note that, at this level of abstraction, the system variables are defined from an end-user point of view. Only three physical variables are used: the input and output flows, and the level of the tank. We have seen that the physical reality is far from that when having to consider three different variables of flow $(Q_0, Q_1, Q_2)$, two pressure variables $(P_0, P_1)$ and the volume of water $(V1)$ inside the tank.

### 4.2. Control Specification

The B Control machine results, in this example, from an intuitive modelling of the controller, which manages operation modes and the associated control actions (see Fig. 13). In automatic mode, the control is supposed to maintain the level of the tank near the value given by the user request. This control is done by increasing or decreasing the percentage of valve opening (opening). Note that this abstract specification of control just states what has to be done to

```
MACHINE      Control
SETS          MODE = {manual, automatic}
VARIABLES    control_mode, level
INVARIANT    control_mode ∈ MODE ∧ request ∈ NAT ∧ opening ∈ NAT
             ∧ level ∈ NAT ∧ …
OPERATIONS
opening ← Control (request, level) =
      SELECT control_mode = automatic ∧ request ∈ NAT ∧ level ∈ NAT
      THEN   IF request > level THEN opening . (opening<opening$0)
             ELSIF request < level THEN opening . (opening>opening$0)
             ELSIF request = level THEN opening . (opening=opening$0)
             END
      WHEN   state = manual
      THEN   IF request = 0  THEN opening := 0
             ELSIF request = 100 THEN opening := 100
             END
      END
mode_management =
      SELECT control_mode = automatic
      THEN   control_mode := manual
      WHEN   control_mode = manual ∧ level > request – 10 ∧ level < request + 10
      THEN   control_mode := automatic
      END
END
```

**Fig. 13.** B Control Machine.

maintain the requested level without detailing the algorithm to be used (e.g. PID). Mode management describes the exact procedure for commutation taking into account the general recommendations given in the system_requirements machine.

Note that creation of the mode commutation history remains the same as in the systems_requirements machine in the absence of any detail of the database management system that should be used. The B variables, invariant and operation related to this part are not presented again in the control machine.

### 4.3. Consistency and Correctness Checking

Next step consists in ensuring the refinement relationship between system requirements and a model of the automation system.

#### *4.3.1. Formalization of the Automation System*

Abstract representation of the automation system is provided by combining the control and Case_study_process machines using the B structuring extends clause that allows invocation of operations (see Fig. 14). Variables processed are defined as $Q_i$ for the input flow, $Q_o$, $P_o$ for the output flow, and $V_t$, $P_t$, for the volume and pressure in the tank.

The use of the extends clause means that all the invariants described in the process and control machines must be preserved by the automation_system machine. This description leads to a failure notification by the theorem proof tool.

Analysis of the proof obligations, i.e. the underlying hypotheses that have to be satisfied for a proof success, gives some indication as to what should be corrected in our specification:

$$Vt \in \text{DISP} \Rightarrow Vt \in \text{NAT} \wedge Q \in \text{EFFORT}$$
$$\Rightarrow Q \in \text{NAT (extract of proof obligation)}$$

The problem highlighted is the following: the process model works with physical variables that belong to Paynter sets (effort, flow, displacement and impulse), whereas the control machine deals with informational variables that belong to the set of natural or real numbers.

Beyond this type checking problem, it appears that the conformance between the variables involved in the process and control models has not been considered. Indeed, the control model deals with an intuitive

```
MACHINE     Automation_System
EXTENDS     Case_study_process, Control
VARIABLES   User_request, Chg_mode_request, Qo, Pi, Vt, Pt, Qi,
OPERATIONS
Level_control (request) =
    ((Qo, Pi),(Vt, Pt)) ← Case study process (Qi, Pi)  || Qo ← Control (user_request, Vt);
Change_mode =
    SELECT Chg_mode_request ∈ BOOL
    THEN mode_management  /* operation call from Control machine */ ; END
```

**Fig. 14.** Formalization of the Automation System machine.

concept of level, whereas the process model speaks of a volume. In the same way, the control model is supposed to calculate the required opening percentage of the valve, whereas the process model deals with the output flow rate and tube section.

An alternative to this conflict consists in defining new operations within the process model that will be in charge of transforming the following.

- An opening percentage calculated by the control system into a physical effect understandable by the process model; this can be done by modifying the status of *A* (tube section) within the Valve.Shape machine that is now defined as a variable, and by introducing a new operation Set_parameters (A) whose function is to initiate the calculation within the Valve.Shape machine with the section value that is required by the control.
- A physical volume *(V ∈ DISP)* into an informational variable ($L \in$ NAT*)* that represents the level inside the tank; this can be done by introducing in the Tank.Time machine a new operation $L \leftarrow$ Get_level *(V)* whose function is to supply a type transformation, and to calculate the level of the tank from the knowledge of its volume and section (owing to a proportional relationship between the two values).
- These two operations added respectively to the Valve.Shape and Tank.Time process machines are in fact systemic operators of Nature (that aim to modify the type of processed variables, from the physical world to the informational domain and vice-versa). They actually represent the abstract behaviour of the Actuation and Measurement system.

### 4.3.2. Refinement Checking

The last stage of our approach consists in verifying that the automation system we have defined satisfies the properties given by the requirements model. This consistency checking is performed using the B refinement mechanism. The properties to be proved are the following:

```
REFINEMENT      Automation_System
REFINES         System_requirements
CONSTANTS       atmospheric_pressure
INVARIANT
    /* existing variables in both abstract and refined machines */
    Qo = input_flow ∧ Qi = output_flow ∧ Vt = tank_level
    /* not existing in the abstract machine */
    ∧ Pi = atmospheric_pressure ∧ Pt = k * tank_level
OPERATIONS
    /* same operations as those defined in Figure 14 */
```

**Fig. 15.** Automation_system as a refinement of System_requirements.

- the services provided by the automation system are included in those required by the end-user;
- the invariant properties of the automation system are included in those defined by the requirements model.

These assumptions comply with the definition of the B refinement. What has to be proved is that the Automation_System machine actually constitutes a refinement of the System_requirements machine. We postulate that this refinement relationship exists, and we describe, using the B formalism, the logical links between the abstract variables defined in the requirements (input_flow, output_flow and tank_level) and the more tangible requirements ($Q_o$ $Q_i$, $V_t$, $P_i$, $P_t$) processed in the Automation_system machine. This can easily be done by enriching the invariant as follows (see Fig. 15). The B theorem prover is used to demonstrate that the assumed refinement relationship is preserved by the machines involved in our specification.

A proof success implies that the combined operations included in the B refined machine (i.e. operations of the Automation_system machine and consequently the called operations supplied by the Control and Process machines) satisfy the invariant and the post-conditions (i.e. produce the same results)

of the operation related to System_requirements machine.

In other words, proving the refinement ensures that the actual operations performed by the control and process systems are compliant with the functional operations expected within the system requirements.

## 5. Conclusion

There is a growing interest in methods and tools that facilitate the validation of software-intensive automation systems. This interest becomes a legal requirement when dealing with safety-critical systems; the IEC 61508 safety-related standard strongly recommends the use of verification methods to be applied in the certification process by the suppliers, integrators or independent external authorities, but without defining how they can be applied. Among many techniques enabling improvement in quality, formal approaches appear to be suitable for checking, from the early phase of a project life cycle, the completeness, the non-ambiguity, the consistency and the correctness of all the various specifications a system is intended to meet.

Most formal methods aim to check *a posteriori* the correctness of a designed system with regard to required goals. In this case, the formal statement is very sensitive to a host of implicit relationships inferred by the requirement process, itself based on natural or skill-oriented languages. This paper presents an alternative system strategy aimed at making a more robust automation by checking *a priori* proved properties from the earlier phases of specification. Our method combines model-driven and proof-oriented specifications based on the B language and its refinement mechanism as a unique and integrating framework.

Although these interdisciplinary exchanges between computer science and automation-related approaches demonstrate that this *a priori* formal method of modelling allows one to verify the highest levels of safety integrity of automation systems, common experiments on laboratory-scale and industrial-scale case-studies emphasize the effort that still must be employed to make the proposed engineering framework effective in practice.

An important limitation for the acceptance of the proposed approach into a well-established automation engineering process is the mathematically oriented notation of the B language. A means of bridging this gap at the specification level is to propose equivalent representations of a B specification using more accepted formalisms. To this end, translations of UML diagrams into B machines have been proposed [17,22] as well as paraphrasing of the B model into a natural language text to facilitate its understanding [10]. As far as the design and implementation levels are concerned, formal refinements of B models towards skill-oriented formalisms remain unresolved issues, although identified as crucial by industrial practitioners [4,27]. Some partial results are available for translating B into continuous [21], event-oriented [6], reliability-oriented [28], or timed [3] models, as well as into programming languages such as C code [5].

## References


1. Abrial JR, Borger E, Langmaack (eds). Formal methods for industrial applications: specifying and programming the steam boiler control. Lecture Notes in Computer Science, State of the art-Survey, ISBN 3-540-61929-1
2. Abrial JR. The B book: assigning programs to meanings. Cambridge University Press, ISBN 0-521-49619-5
3. Abrial JR, Mussat L. Introducing dynamic constraints in B. In: Proceedings of the second international B conference, Montpellier, France. April 22–24, 1998, Lecture Notes in Computer Science 1393, Springer, ISBN 3-540-64405-9
4. Ait-Ameur Y, D'Ausbourg B, Boniol F, Delmas R, Wiels V. A component based methodology for description of complex systems: an application to avionics systems. In: European Systems Engineering Conference, Toulouse, France, 21–24, 2002
5. Bert D, Cave F. Construction of finite labelled transition systems from B abstract Systems. In: Integrated Formal Methods, IFM2000, *LNCS* 1945, Springer-Verlag, 2000, pp 235–254
6. Bert D, Boulmé S, Potet ML, Requet A, Voisin L. Adaptable translator of B specifications to embedded C programs, In: Proceedings of FME 2003: formal methods, LNCS 2805, Pise, 2003, pp. 94–113
7. Cassandras CG, Lafortune S. Introduction to discrete event systems. Kluwer Academic Publisher, ISBN 0-7923-8609-4.
8. Clarke EM, Grunberg O, Peled DA. Model checking, The MIT Press, 2000
9. Craigen D, Gerhart S, Ralston T. Formal methods technology transfer : implements and innovation, Formal Methods by Hinchey MG, Bowen JP. (eds). Prentice Hall International, 1995, pp 399–419
10. Diallo D, Vailly A. Paraphrasing of specification written in B, Networking Inform Syst. J. 1998; 1: 2–3
11. Dupuy S, Bousquet L. Validation of UML models thanks to Z and Lustre. In Formal Methods Europe, Berlin, Germany, March 2000.
12. Féliot C, Cassar J. Ph, Staroswiecki M. Towards a language of physical systems, IEEE SMC Conference, Beijing, Oct. 4–10, 1996.
13. Feldmann K, Colombo AW, Schnur C, Stöckel T. Specification, design, and implementation of logic controllers based on coloured petri net models and the

standard IEC 1131, IEEE Trans Control Syst Technol, 1999; 7: 6
14. Fusaoka A, Seki H, Takahashi K. A description and reasoning of plant controllers in temporal logic. In: Proceedings of the international joint conference on artificial intelligence. Karlsruhe, Germany, 8–12 August, 1983, 405–408
15. Iung B, Morel G, Leger JB. Proactive maintenance strategy for harbour crane operation improvement. Robotica 2003; 21: 313–324
16. Johnson TL. Improving automation software dependability: a role for formal methods ? In: proceedings of 11th IFAC International Conference on Information Control Problems. April 4–7, 2004, Salvador-Bahia, Brazil.
17. Laleau R, Pollack F. Coming and going from UML to B : a proposal to support traceability in rigorous IS development. In: International conference on formal specification and development in Z and B (ZB2002), vol. 2272 of Lecture notes in computer science, Grenoble, France, January 2002. Springer Verlag, pp 517–534
18. Lamboley P. Production Systems Automation Formal Method Proposal, PhD Thesis, University Henri Poincaré, Nancy 1, 2001 (in French)
19. Lano K, Bicarregui J, Kan P. Experiences of using formal methods for chemical process control specification, Control Eng. Practice 2000; 8: 1
20. Lano K. The B language and method, a guide to practical formal development. Springer Verlag, 1996, ISBN 3-540-76033-4
21. Lecomte T. Event Driven B: language, tool support and experiments. In: Proceedings of RCS'02 international workshop on refinement of critical systems : methods, tools and experience, in conjunction with the 2nd Conference of B and Z Users (ZB 2002), January 23–25, 2002, Grenoble, France
22. Ledang H. Automatic translation from UML specifications to B. In: proceedings of the International Workshop on refinement of critical systems: methods, tools and experience, in conjunction with the 2nd Conference on B and Z, January 23–25/01/2002, Grenoble, France
23. Lhoste P, Morel G. From discrete event behavioural modelling to intelligent actuation and measurement modelling. In: proceedings of the ASI annual Conference of the ICIMS-NOE, 1996, Toulouse
24. Lhote F, Chazelet Ph, Dulmet M. The extension of principles of cybernetics towards engineering and manufacturing. IFAC Annu. Rev. Control 1999; 23(11): 1–11
25. Mattsson SE, Elmqvist H, Otter M. Physical system modelling with Modelica. Control Eng Practice 1998; 6: 501–510
26. Mellor SJ, Kendall S, Uhl A, Weise D. Model driven architecture, Addison-Wesley Pub Co. 2004, ISBN: 0201788918
27. Moik A. Engineering-related formal method for the development of safe industrial automation systems. Autom Technol Practice Int J 2003; 1: 45–53
28. Morel G, Mery D, Leger JB, Lecomte T. Proof-oriented fault-tolerant systems engineering: rationales, experiments and open issues. In: 7th IFAC Symposium on Cost Oriented Automation, COA'2004, Gatineau, Québec, Canada, June 2004.
29. Panetto H, Pétin JF. Metamodelling of production systems process models using UML stereotypes. Int J Internet Enter Manage 2005; 3(2): 155–169
30. Parker D, Kwiatkowska M, Norman G. Controller dependability analysis by probabilistic model checking. In: IFAC/INCOM2004 symposium, Salvador-Bahia, Brazil, April 4–7 2004
31. Paynter M. Analysis and design of engineering systems, M.I.T. Press, Cambridge, ISBN 0-262-16004-8.
32. Petin JF, Iung B, Morel G. Distributed intelligent actuation and measurement system within an integrated shop-floor organisation, Comput Industry J 37: 197–211
33. Pétin JF, Morel G, Méry D, Lamboley P. Process control engineering: contribution to a formal structuring framework with the B method, Lectures notes in Computer Science 1998; 1393: 198–209
34. Polzer K. Ease of use in engineering – availability and safety during runtime, Autom Technol Practice Int J 2004; 1: 49–60
35. Ramadge PJ, Wohnham WM. Supervisory control of a class of discrete event processes, SIAM J Control Optim 1987, pp 206–230.
36. Roussel J-M, Faure J-M. An algebraic approach for PLC programs verification, In: 6th IFAC / WODES, Zaragoza, Spain, 2–4 October, 2002, pp. 303–308.
37. Selic B. Using UML for modelling complex real-time systems, Lectures Notes in Computer Science 1998; 1474: 250–262
38. Shell T. Systems functions implementation and behavioural modelling: system theoretic approach. Int J Systems Eng 2001; 4
39. Wright PK, Bourne DA. Manufacturing intelligence, Addison-Wesley, 1988, ISBN 0-201-13576-0

## Appendix

All UML graphical notations are very helpful in improving the readability of the B machines presented in this paper. The best practices found in the computer science literature start a UML study by defining the main objects classes involved in the application considered. This can be done using "Use case" diagrams, or more precisely, using class diagrams.

Applying Fusaoka's predicate to our case study leads to the following class diagram (see Fig. A1), where the Automation_System class is composed of a Process_System class and a Control_System class. Associated operations represent the services provided by these two classes (physical transformations for the first and informational processing for the second). The dependency association between the Automation_System and Requirements classes represents the refinement relationship that must exist between these two classes. The Object Constraint Language (OCL) enables the expression of constraints applied on objects or associations of objects, and is

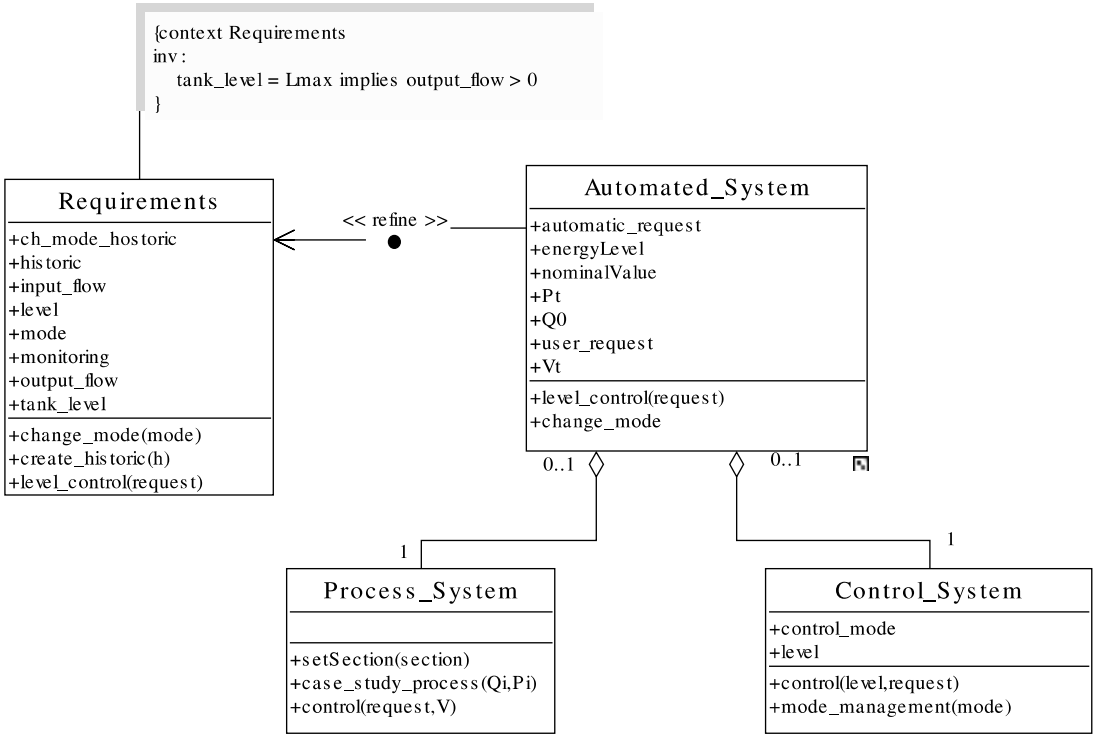



**Fig. A1.** UML class diagram of specifications according to Fusaoka's predicate.

used in the case study to express the invariant properties of the system.

A further specification should enrich this static representation with additional details related to the dynamic behaviour of the objects. Unfortunately, even if many formalisms are available (information exchanges between objects can then be described using a collaboration diagram, scheduling of operations within a class can be described by a sequence diagram, control operation behaviour can be described through a State-Transition diagram, etc.), the refinement rationale, which is the accurate way to guarantee the consistency the specification process, is barely supported by UML.

Available online at www.sciencedirect.com
ScienceDirect
Control Engineering Practice 15 (2007) 595–614
CONTROL ENGINEERING PRACTICE
www.elsevier.com/locate/conengprac

# Supervisory synthesis for product-driven automation and its application to a flexible assembly cell

Jean-François Pétin*, David Gouyon, Gérard Morel

*CRAN-UMR 7039, Nancy-University/CNRS, BP239, 54506 Vandoeuvre lès Nancy Cedex, France*

Received 2 May 2006; accepted 23 October 2006
Available online 11 December 2006

**Abstract**

A make-to-order business model requires manufacturing control to face customised product variability and traceability. Considering the product as central to the automation rationale provides each product occurrence with informational and decisional capabilities. This paper considers a formal framework for such product-driven automation within the context of the Supervisory Control Theory. Two interoperable classes of supervisors are synthesised. Product supervisors control routings through the manufacturing system according to customised product specifications and resource capabilities, and resource supervisors manage the execution of operations required by product supervisors. A case study, based on a flexible assembly cell, illustrates the approach and opens issues for industrial practice.


*Keywords:* Control system synthesis; Flexible automation; Supervisory control; Reconfiguration; Manufacturing plant control

## 1. Introduction

The introduction of Information Technology in manufacturing systems gives manufacturers an opportunity to promote make-to-order business models and mass customisation of products (Da Silveira, Borenstein, & Fogliatto, 2001).

Facing this wide range of customer needs requires manufacturing control systems to have the ability to adapt to variable demands in terms of product specifications or intrinsic system changes (Koren, 2005).

To this end, a flexible control system may embed a maximum set of switchable control policies that have been designed to cover the various product manufacturing routings and the functional redundancies between the machines. Even if the IEC61499 standard provides a framework to handle control execution paths, this approach increases the control complexity (Ollero et al., 2002), in terms of the number of control states, which naturally arises as the number of machines and product variability increase.

To avoid this combinatorial explosion, challenges for modern automation are to provide methods and tools capable of designing and implementing dynamically reconfigurable control systems. It means that the control policy is not fully predefined but includes observation abilities to detect when a reconfiguration is needed, and decisional abilities based on synthesis (Qiu, Wysk, & Xu, 2003) or scheduling algorithms (Henry, Zamai, & Jacomino, 2004) to automatically compute a new control configuration (Brennan, Zhang, Xu, & Norrie, 2002).

Considering that the control of elementary operations performed by the manufacturing machines seems to remain somehow constant through time, dynamic reconfiguration required by product variability is supposed to be limited to the "on the fly" generation of product routing control. Similar to the concept of the virtual production line (Qiu et al., 2003), this approach makes the product active in the scheduling and the execution of its manufacturing operations, and relies on a clear separation between machine control activities and product control.

This paper aims at proposing a formal modelling framework based on the Supervisory Control Theory (SCT) (Ramadge & Wonham, 1987) for a modular synthesis of such product-driven reconfigurable control systems.

*Corresponding author. Tel.: +33 383 68 44 38; fax: +33 383 68 44 43.
*E-mail address:* jean-francois.petin@cran.uhp-nancy.fr (J.-F. Pétin).

0967-0661/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.conengprac.2006.10.013

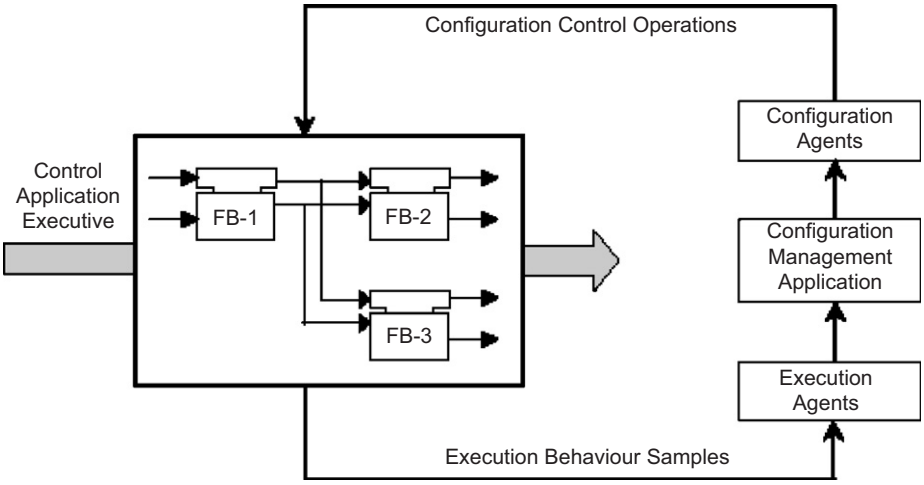



Fig. 1. Reconfiguration framework (Brennan et al., 2002).

Section 2 introduces the dynamic reconfiguration of control systems and briefly presents the SCT theory for their synthesis. Section 3 defines a synthesis framework for product-driven automation which ensures consistency between resource and product supervisors, which respectively control the execution of manufacturing operations and product routings. Section 4 details the different synthesis processes and the underlying models of resources, the manufacturing system and product specifications. This approach is illustrated in Section 5 using a case study based on the flexible assembly cell of the AIP-PRIMECA[1] Lorraine (AIPL) university workshop. Operational aspects of the supervisors synthesised for the case study are discussed in Section 6. Conclusions and open issues for future research are discussed in Section 7.

## 2. Problem definition

### 2.1. Dynamic reconfiguration of control

Mass customisation relates to the ability to provide individually designed products and services to every customer through high process flexibility (Da Silveira et al., 2001).

From the process point of view, machine flexibility is the capability of machines to perform different operations. Taking into account functional redundancies between machines, routing flexibility is the ability of manufacturing a given set of part types using one or more routes through the machines (Tsubone & Horikawa, 1999).

Therefore, product customisation, machine and routing flexibility impact the manufacturing control so that its online dynamic reconfiguration should make it possible to fit any process rescheduling or any customised product definition.

According to Brennan et al. (2002), implementing a dynamic reconfiguration of control requires a configuration loop involving informational, decisional and operational activities (Fig. 1) for:

- monitoring, diagnosis or even prognosis (execution agent in Fig. 1) in order to produce information about the control environment and to define when and where a reconfiguration is required,
- decision-making to define the most appropriate control policy (called Configuration Management Application in Fig. 1),
- operational execution of the reconfigured control actions (Configuration agents in Fig. 1).

Monitoring (Vogrig, Baracos, Lhoste, Morel, & Salzemann, 1987; Zamaï, Chaillet-Subias, & Combacau, 1998), diagnosis (Toguyeni, Craye, & Sekhri, 2006) or even prognosis of manufacturing systems have been widely explored by Discrete Event System scientists and provide today material for identifying degradations or failure modes where control reconfiguration would be required (Berruet, Toguyeni, Elkattabi, & Craye, 2000). Moreover, infotronics technology such as RFID tags embedded on the products enables individual identification of product occurrences that open a way towards the customisation of control rules for each product occurrence (McFarlane, Sarma, Chirn, & Ashton, 2002).

Operational aspects of control reconfiguration are generally concerned with class C interoperability[2] issues (Iung, Neunreuther, & Morel, 2001; Staroswiecki and Bayart, 1996) defined as the ability for automation components to be replaced by other ones offering similar services (class A maps the ability to exchange information while class B characterises the ability to cooperate for the execution of a given service). Indeed, operational reconfiguration has to manage switching from an obsolete control

[1] http://www.aipl.uhp-nancy.fr/lorraine/.

[2] SEMATECH: *Device interoperability guideline for sensors, actuators and controllers*, (1995), Technology Transfer Standard 94102567A-STD, http://www.sematech.org.

strategy to a new targeted configuration which can be obtained by tuning the component parameters or by replacing some of the control components.

These interoperability challenges lead to modular control systems promoting the use of standardised and reusable components on the shelves (Vyatkin, Chistensen, & Martinez Lastra, 2005). From a syntactic point of view, IEC 61499[3] provides a function block structure where the clear separation between execution control and algorithm description facilitates dynamic reconfiguration. From a semantic point of view, interoperability requires the standardisation of component interfaces and behaviour as well as the definition of a collaboration protocol (Endsley, Almeida, & Tilbury, 2006; Newman et al., 2000; Wright & Bourne, 1988).

The informational (monitoring, diagnosis) and operational issues of control reconfiguration are beyond the scope of this paper, which focuses on decisional activities required to define the most appropriate control policy. In the case of dynamically reconfigurable systems, two different methods of designing manufacturing control can be mentioned.

In the first method, the set of control policies required to master product variability are designed and pre-integrated in the control rules (Berruet et al., 2000; Toguyeni et al., 2006). It means that all possible manufacturing trajectories have been modelled including the various product manufacturing routings and the functional redundancies between the machines. In this case, reconfiguration requires the tuning or the selection of already designed control rules. The main benefit is the flexibility of control policies but these approaches often suffer from an inevitable state explosion problem related to the number of machines and product variability (Muhl, Charpentier, & Chaxel, 2003).

In the second method, the control policy, which is the most appropriate for taking into account product and manufacturing system features, is "on the fly" and automatically generated (Qiu et al., 2003). This approach avoids the state explosion problem encountered in the previous approaches but presents some limits when mixing various product lots on the same manufacturing system. Moreover, it requires using synthesis techniques (Lauzon, Mills, & Benhabib, 1997, Lennartson, Tittus, Fabian, & Hellgren, 1998) which enable automatic and on the fly generation of control rules.

### 2.2. Control synthesis

Among candidate approaches for control synthesis, such as techniques using Petri Nets (Achour, Rezg, & Xie, 2004; Basile, Carbone, & Chiacchio, 2006) or synchronous languages (Marchand, Bournai, Le Borgne, & Le Guernic, 2000), SCT (Ramadge & Wonham, 1987) has been proved to be an efficient and computer-aided framework. This theory provides a formal framework for DES analysis and synthesis based on two main concepts: the process to be automated (called a *generator* or a *plant*) and the supervisory controller (called a *supervisor*).

An automaton of the following form describes the process model:

$$G = (X_g, \Sigma_g, \alpha_g, x_{0,g}, X_{m,g}),$$

where $X_g$ is a set of states $x_g$, $\Sigma_g$ is a non-empty set of event labels called an alphabet, $\alpha_g$ is a transition function described by state transitions, $x_{0,g} \in X_g$ is the initial state, and $X_{m,g} \subseteq X_g$ is the set of *marked* (terminal) *states*. The model is assumed to generate spontaneously controllable and uncontrollable events. Controllable events can be disabled while uncontrollable events cannot be prevented and are permanently enabled.

The supervisor can affect the behaviour of the process model by enabling or disabling controllable events to maintain the process in a space of acceptable states for a given specification. An automaton of the following form describes the supervisor:

$$S = (X_s, \Sigma_s, \alpha_s, x_{0,s}, X_{m,s}),$$

where $X_s$ is a set of states $x_s$, $\Sigma_s$ is the alphabet used by $G$, $\alpha_s$ is a transition function, $x_{0,s}$ is the initial state, and $X_{m,s}$ is the set of *marked states*.

Synthesis algorithms (Kumar, Garg, & Marcus, 1991; Wonham & Ramadge, 1987) automatically generate the optimal supervisor, which enables the maximal set of events that do not contradict the goal specifications. These specifications define the enabled sequences of events that belong to automaton $G$. Considering the generator model as the physically possible sequences, and the goal specifications as the legal sequences, the synthesised supervisor is expected to inhibit the process behaviour so that only desirable sequences are generated. Synthesis algorithms involve the synchronous composition of process and specification models and the elimination of strongly and weakly forbidden states.

In the field of SCT, the benefit of modularity in avoiding the state space explosion generated by the synthesis algorithms has been widely demonstrated (De Queiroz & Cury, 2002; Endsley et al., 2006; Vahidi, Fabian, & Lennartson, 2006). Several extensions of the original framework have been proposed, such as:

- modular *supervisors*, with a distributed or hierarchical decomposition (Gohari & Wonham, 1998) and
- modular *generators* or *plants* based on a structured model of the process (Yoo & Lafortune, 2002),
- mixed approaches, where both *generator* and *supervisor* are modular (Chafik & Niel, 2000).

### 2.3. Dynamic reconfiguration using synthesis

The proposed approach aims at modelling a reconfigurable control system able to master the product variability induced by mass-customisation.

[3]International Electrotechnical Commission: Function *blocks, Part 1 – Architecture*, IEC PAS 61499-1, Geneva (2000).

It clearly appears that off-line designing of all control policies will at least be very difficult for complex customised products or even impossible if new product specifications occur. This reinforces the benefit of synthesis techniques for handling customised product control. However, considering that resource (or machine) elementary operation control, i.e. management of a manufacturing operation performed by a given resource, seems to remain somehow constant through time, "on the fly" synthesis may be limited to the routing of the product through the manufacturing system. However, to keep the flexibility of off-line design and to be able to adjust the product routing according to resource availability, objective is to synthesise a product supervisor which defines all the admissible manufacturing routes for a given customised product occurrence, and which initiates operations controlled by the resource supervisors.

This approach is part of the product-driven automation concept.

## 3. Synthesis framework for product-driven automation

### 3.1. The product-driven automation framework

The main hypotheses of product-driven automation consist in providing the product with information, decision and communication capabilities in order to make the product active in the scheduling and the execution of its manufacturing operations. In this sense, the IMS[4] community, especially in the area of Holonic Manufacturing Systems (Valckenaers 2001) promotes conceptual architectures, such as PROSA, which tend towards providing the manufactured product with intelligent behaviour. Based on these conceptual guidelines, this paper focuses on the design of a product-driven distributed control system, shown in Fig. 2, which is based on the cooperation between:

- product supervisors which control the manufacturing routes according to a scheduled list of operations the product has to undergo; these supervisors are specific for each product occurrence in order to take into account their customisation and
- resource supervisors which ensure correct execution of transport and transformation operations and provide the product controllers with accurate reports; control flexibility relies on tuning call parameters of the functional objects which coordinate and control the elementary operations.

The definition of these supervisors are founded, on the one hand, on the modelling of the manufacturing system capabilities which describe the system topology and the manufacturing operations performed by each resource, and, on the other hand, on the modelling of product requirements in terms of the operations it has to undergo.

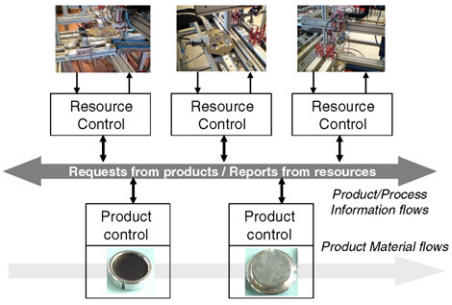



Fig. 2. Product-driven control architecture.

A unified modelling framework is required to facilitate joint design of product and resource controllers. According to the Discrete Event Systems (DES) control theory (Cassandras & Lafortune, 1999), the design of control systems consists in defining the (unknown) control rules of the (known) dynamics of a physical system that satisfy some (known) behavioural goals while satisfying the predicate (Fusaoka, Seki, & Takahashi, 1983):

$$\textit{Dynamics} \wedge \textit{Unknown Control Rules} \supset \textit{Goal}. \quad (1)$$

Note that the $\supset$ logic operator has the same meaning, according to Fusaoka's interpretation, as the implication operator ($\Rightarrow$). It means that satisfying behavioural properties of process and control models implies satisfying behavioural properties of the goal. In the context of product-driven automation, this predicate can be refined into two consistent interpretations.

The first interpretation relates to the product:

$$\textit{Manufacturing system capabilities} \wedge \textit{Unknown product control rules} \supset \textit{Product manufacturing plan}, \quad (2)$$

where the routing control is defined from the product manufacturing plans given in terms of the orderly operations to be applied to the product, and from the manufacturing resource capabilities.

The second interpretation relates to the resources and is used to provide a modular control for the manufacturing resources according to:

$$\textit{Resource dynamics} \wedge \textit{Unknown resource control rules} \supset \textit{Resource capabilities}, \quad (3)$$

where control rules are designed from resource capabilities given in terms of expected operation behaviour and resource dynamics in terms of physically acceptable states.

### 3.2. The product-driven synthesis framework

The framework presented in this paper applies SCT theory and its modular extensions in a structured

[4]Intelligent Manufacturing Systems international initiative, http://www.ims.org/.

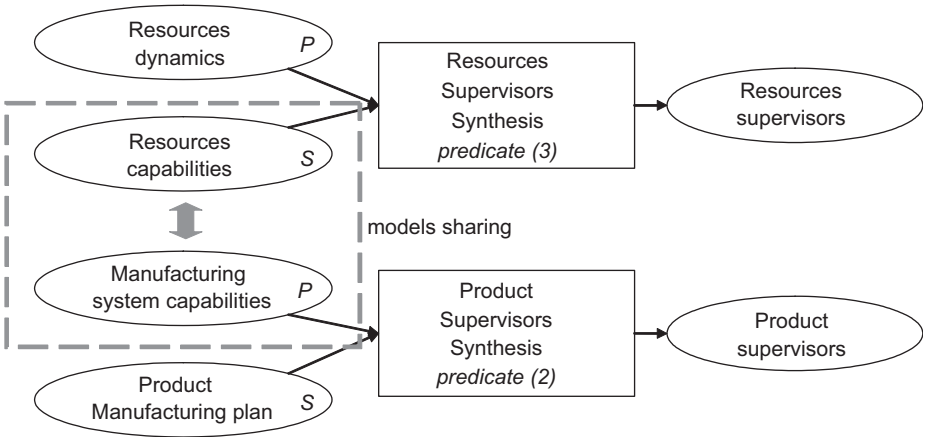



Fig. 3. The product-driven synthesis framework.

modelling method to synthesize product and resource supervisors of a product-driven automation according to predicates (2) and (3). Correspondence of predicates (1), (2) and (3) with SCT notation is respectively the following: *goal*, *product manufacturing plan*, and *resource capabilities* are models of SCT specification (*S*), *dynamics*, *manufacturing system capabilities*, and *resource dynamics* are models of SCT plant or generator (*P*) while *unknown control rules*, *unknown product control rules*, and *Unknown resource control rules* are supervisors to be synthesised.

Predicate (2) and (3) lead to execute two separate synthesis processes in order to obtain product and resource supervisors (Fig. 3) which cooperate. The product supervisor has to request a resource supervisor to initiate an operation; if the resource is able to answer positively, the resource supervisor can execute this operation according to internal constraints of the resource, and emits a report when the operation is over. Product and resource supervisor communication is assumed to occur when products are physically connected to resources. Moreover, only one product can be processed at a time by a given resource, and initiating operations on the same resource by two different product supervisors is physically avoided.

A necessary condition for interoperability requires standardisation of the supervisor interfaces based on request/report semantics and on a shared concept of generic manufacturing operations (transport and shape transformation) independent from their execution by a given resource. From a SCT point of view, this interface standardisation is ensured by sharing:

- a common modelling alphabet composed of request and report events related to a pre-fixed set of manufacturing operations that are defined independently from the location where they are executed and
- consistent models (dashed square of Fig. 3) of the resource capabilities to be used in predicate (3) and of the manufacturing system capabilities to be used in predicate (2).

Note that this interface standardisation means that alphabet events are interpreted in terms of inputs and outputs of the product and resource control. Consequently, controllability of these events depends on the environmental context of a supervisor. For example, a request event for executing an operation is seen by a product supervisor as a controllable event while it is seen as uncontrollable by the resource supervisor. As a result, the concept of global controllability and uncontrollability of events is absent from the synthesis processes. This corresponds in fact to an input/output interpretation of SCT theory as proposed by Balemi, Hoffmann, Gyugyi, Wong-Toi, and Franklin (1993).

## 4. Supervisor syntheses for product-driven automation

### 4.1. Product supervisor synthesis

In order to synthesise product supervisors, i.e. the unknown control rules of predicate (2), models of product manufacturing plans (specifications) and of manufacturing system capabilities are needed.

#### 4.1.1. Product specifications

The product designer elaborates product manufacturing plans which will generate the expected features of the products. From a control theory point of view, they can be represented in terms of an orderly set of manufacturing operations which the product has to undergo. This information can be represented using an automaton (Fig. 4), which represents the logical sequence between the morphological and/or spatial states of the product. Transition between two states is triggered when a report about product states occurs (*RP OPk,* where *k* is the number of *OP* operation performed on the product).

This model is a specification of the orderly states which characterise the product all along the manufacturing process (as given by reports) and does not control the requests to be sent to the resources. This logical sequence may present some flexible trajectories in case of

non-orderly product transformations (second automaton in Fig. 4).

*4.1.2. Manufacturing system modelling*

To model the manufacturing system capabilities, resource generic models and a composition operator based on the cell topology are proposed.

*4.1.2.1. Models of resource capabilities.* From a control point of view, the description of the resource capabilities is limited to the enumeration of the several operations a resource is able to perform. The alphabet of these models is composed of operation requests (*RQ OPk,i*) and reports (*RP OPk*) where $k$ is the number of operations and $i$ the number of resources. The language defined by these models details the admissible sequences of requests and reports.

According to Vogrig et al. (1987), Wright and Bourne (1988) and Lauzon et al. (1997), a manufacturing device can be described by three generic states: *working*, *idle*, and *down*. This generic model is extended by splitting the *working* state into several states which correspond to the operations the resource is able to perform. Note that the dysfunction states (*down*) of the machine are not considered to facilitate the presentation of the approach, since these states would only have as principal consequence the size of the models.

To systematise the modelling, generic models of resources are gathered using a systemic classification in terms of shape, time, or space, respectively associated to morphological transformation, storage, and transport systems.

The first two kinds of systems are modelled using an $n+1$ states automaton (Fig. 5a), where $n$ represents the number of performed operations. The resource state moves from *idle* to *OPk* ($k \in \{0, n\}$) when the operation *OPk* is requested by the product. A move back to the *idle* state occurs when the associated report is emitted. For synthesis purposes, the *idle* state is noted as a marked state. This means that *idle* is considered as the legal end of a sequence and corresponds to a steady state.

Transport systems have been modelled in a different way because their states represent the different reachable locations, and are all considered as steady states. Consequently, the previous model has been modified to introduce several marked idle states associated to locations of the manufacturing resources (Fig. 5b). Representation with a single idle state would have been adopted if the transport system had a reference position from which every move was initiated. Note that these resource models are only used to compose a system capability model from which the possible routings for one product occurrence are generated. Consequently, multiple product management is not considered for these models.

*4.1.2.2. A model of manufacturing system capabilities.* The transformation and storage (shape and time) resources have to cooperate with transport (space) resources in order to ensure transformation and routings of the product. Interactions between these resources are represented by a model of manufacturing system capabilities which takes into account cell topology. Classical synchronous composition, which leads to a Cartesian product, is too permissive to obtain this model because it does not identify the localisation of the transformation resources (places that can be reached by the transport system, i.e. states of the space model).

Therefore, an operator which fuses transformation and transport automata is introduced:

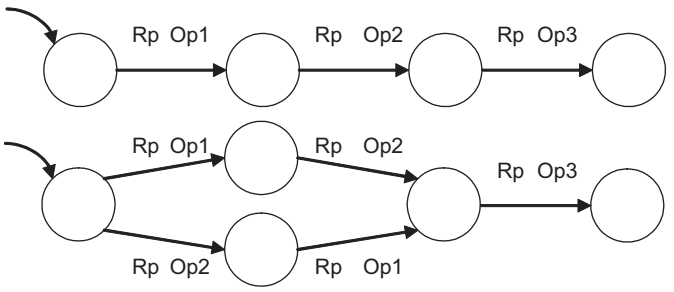



Fig. 4. Product specifications for control issues.

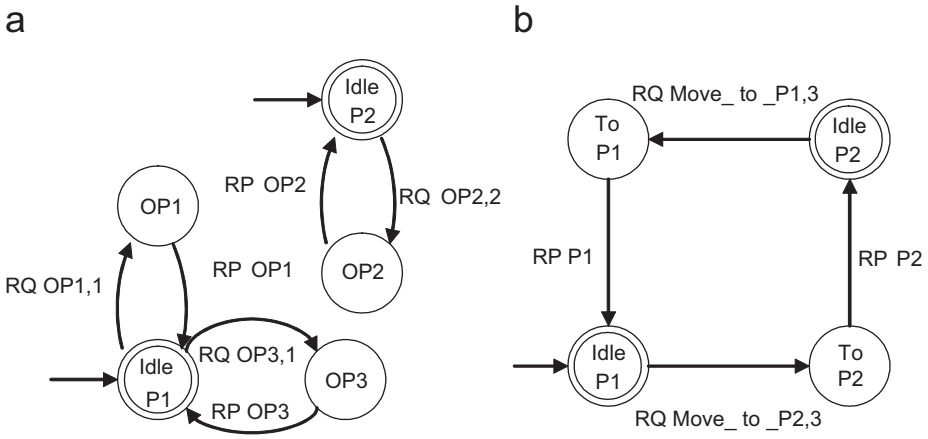



Fig. 5. Examples of resource generic models. (a) shape or time systems and (b) space system.

- states of input automata are merged according to state equivalence classes which associate operation capabilities with their spatial localisations and
- the two transition functions—associated to the two input automata—are joint and applied on their respective state classes.

The fusion operator is defined on two automata G1 and G2:

$G1 = (X1, \Sigma 1, \alpha 1, x1_0, X1_m)$ and

$G2 = (X2, \Sigma 2, \alpha 2, x2_0, X2_m)$,

with $\Sigma 1 \cap \Sigma 2 = \varepsilon$ and $X1 \cap X2 = \varnothing$.

To start, the initial automata are enriched by defining the function *sit*: $X1 \cup X2 \to \Re^3 \times F$, which associates a state $x \in X1 \cup X2$ with spatial coordinates $(x, y, z)$ of the product belonging to $\Re^3$ and with in-progress operations belonging to the set $F = \{op_1, op_2, \ldots, op_n, 0\}$ of available operations.

The *sit* function allows the definition of an equivalence relationship in such a way that $x1_i$ and $x2_j$ will be considered as equivalent if $\exists\,(x1_i, x2_j) \in X1 \times X2 / sit(x1_i) = sit(x2_j)$.

Equivalence classes are noted with a dot above equivalence class name, i.e. the equivalence class for $x$ is noted $\dot{x}$.

Then all $x2_j \in X2$ (resp. $x1_i \in X1$) which satisfy the equivalence relationship for each $x1_i \in X1$ (resp. $x2_j \in X2$) are sought. This defines state equivalence classes noted $\dot{x}1_i$ (resp. $\dot{x}2_j$) that merge $x1_i$ and $x2_j$ such as

$$\dot{x}1_i = \begin{cases} \{x1_i, x2_j\} & \text{if } \exists\ x2_j \in X2 / sit(x1_i) = sit(x2_j), \\ x1_i & \text{else,} \end{cases}$$

$$\dot{x}2_j = \begin{cases} \{x1_i, x2_j\} & \text{if } \exists\ x1_i \in X1 / sit(x1_i) = sit(x2_j), \\ x2_j & \text{else.} \end{cases}$$

The resulting automaton $G = (X, \Sigma, \alpha, \dot{x}_0, X_m)$ is then defined by

- $X = \{\dot{x}1_i\} \cup \{\dot{x}2_j\}$
- $\Sigma = \Sigma 1 \cup \Sigma 2$ *with* $\Sigma 1 \cap \Sigma 2 = \varnothing$;
- for $\dot{x} \in X$, $\sigma \in \Sigma$, $\alpha : X \times \Sigma \to X$
- $\alpha(\dot{x}, \sigma) =$ tpb 6pt
$$\begin{cases} \alpha(\dot{x}1_i, \sigma_k) = \dot{x}1_p \quad \text{for } \sigma_k \in \Sigma 1 \\ \quad \text{such as } \alpha 1(x1_i, \sigma_k) = x1_p, \\ \alpha(\dot{x}2_j, \sigma_h) = \dot{x}2_q \quad \text{for } \sigma_h \in \Sigma 2 \\ \quad \text{such as } \alpha 2(x2_j, \sigma_h) = x2_q, \end{cases}$$
- $\dot{x}_0 = (\dot{x}1_0)$
- $X_m = \{\dot{x}1_{m,i} / x1_{m,i} \in X1_m\} \cup \{\dot{x}2_{m,j} / x2_{m,j} \in X2_m\}$

The fusion operator is iteratively applied to fuse all transformation and transport models and to obtain a manufacturing system capabilities model. In the first iteration, G1 corresponds to the transport model while G2 corresponds to a transformation model. Then, for all following iterations, G1 is the previous fusion result, and G2 is the resource model to be merged.

Fig. 6 shows the model of a manufacturing system, which results from applying the fusion operator on two transformation resources (R1 is able to perform Operations 1 and 3 while R2 is limited to Operation 2) and one transport resource between them. Models of these resources conform to those in Fig. 5.

#### 4.1.3. Synthesis of product supervisors

Subsequently, classical synthesis algorithms can be applied to generate a supervisor that controls the alternate routes of a given product within the cell. The result is the most permissive supervisor which represents an exhaustive set of manufacturing trajectories which are acceptable to satisfy the product occurrence specifications. Optimality in terms of control performance or supervisor size is not sought during this phase which aims at defining all acceptable routings even if some are obviously of no interest.

This automatic synthesis of product routings is illustrated using the case study in the next section.

### 4.2. Resource supervisor synthesis

The synthesis of resource supervisors combines the object-oriented automation rationales and the modular synthesis techniques. Modularity criteria are not only driven by state-space explosion issues (De Queiroz & Cury, 2002), but must be justified by the structure of the physical process itself (Gouyon, Pétin, & Gouin, 2004).

To this end, structured modelling is proposed to start with the elementary actions, which can be executed by a resource using actuators. These actions are progressively coordinated in a bottom-up manner to perform actions that are more complex. Applying this structured modelling to the synthesis process gives rise to a modular and iterative synthesis method in which modular models of specification and plant are used to synthesise a hierarchy of coordinated supervisors.

The resource *specification* model is split into several sub-models associated to each module function; it mainly relates to elementary action specifications or coordination specifications.

*The generator* or *plant* model is also split into several sub-models. The sub-models of the lower layer represent the physically admissible behaviour of the actuators and are defined by the designer. The *generator* sub-model of a layer $n$ refers to the behavioural description of the layer $n-1$ supervisors considered as the *plant* for a layer $n$ coordination supervisor. This *generator* or *plant* model for the layer $n$ supervisor is automatically given by (Fig. 7):

- the projection of the layer $n-1$ supervisors to keep only the observable events from the given layer $n$,

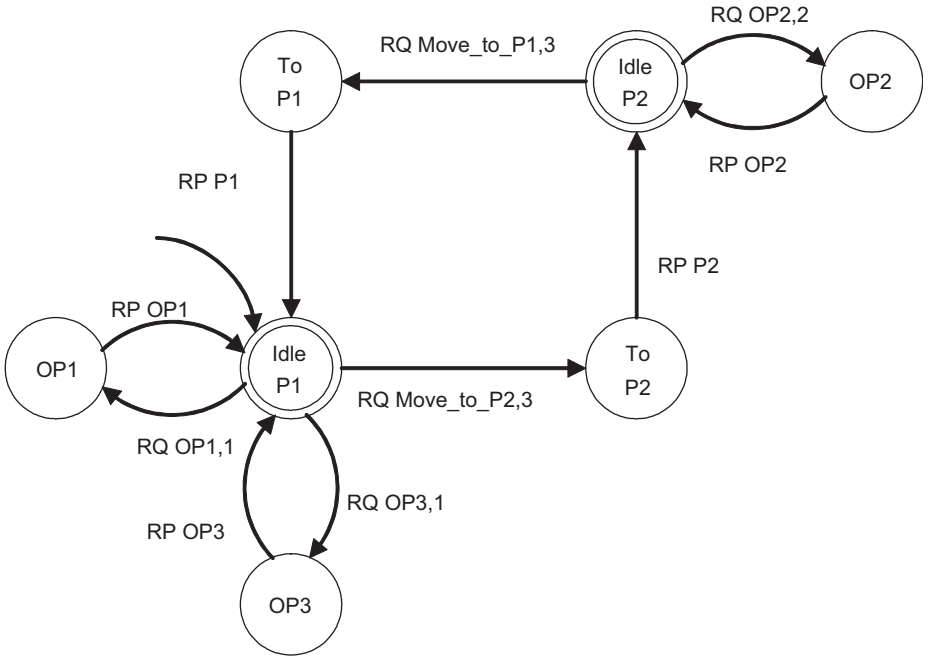



Fig. 6. Result of fusion operation.

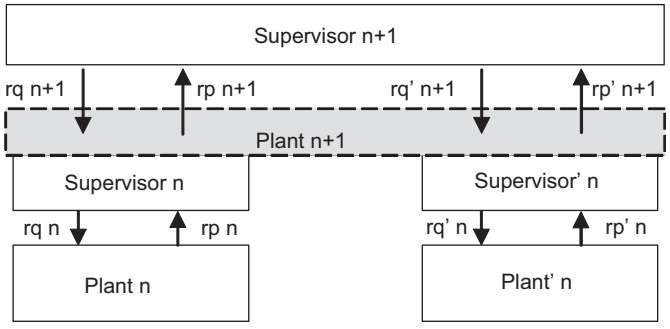



Fig. 7. Iterative definition of plant models.

- enriching the alphabet of each projection by self-loops with the union of the alphabets involved in the other projections; the aim is to have a common alphabet for all projections,
- the controllability modification of each projected model, i.e. controllable events of layer $n$ become uncontrollable and uncontrollable events become controllable and finally,
- the synchronous product of the above projections (they do not share events).

As already shown for product and resource supervisors, controllability of events is not global for all resource supervisors but is variable according to the layer where the synthesis is applied; a controllable event (i.e., seen as an *output*) for a layer $n$ supervisor will be uncontrollable (i.e., seen as an *input*) for its layer $n+1$ coordination supervisor (Fig. 7).

This iterative way of reasoning leads to a modular control synthesis that can be summarised by Fig. 8, which shows the various stages of synthesis, projection, and composition of supervisory controllers.

### 4.3. Discussion

Using coordinated supervisors cannot really be proved to be deadlock free. However, in order to limit the risk of deadlocks, the alphabets of supervisors of a given layer are disjoints. Consequently, two supervisors which belong to the same layer can be proved, according to Leduc (2002), to be deadlock free (no events are exchanged or shared between them). However, the approach does not ensure that a supervisor at a layer $n+1$ is never absorbed by a strongly connected component where some events are disabled and consequently may lead to blocking in some supervisors at layer $n$. Open issues in the framework of modular specifications should be addressed to avoid this type of deadlock, such as the prefix-closure of the alphabets (Jiang & Kumar, 2000). Hierarchical coordination is then applied until the highest layer is reached, where operations as seen by the product are processed.

Sharing resources by several products is assumed to be physically non-conflicting in the sense that only one product can initiate an operation on a given resource. The instantiation mechanism applied to the operation requests clarifies this situation on the models by ensuring that a request from a given product supervisor can only be processed by a unique resource. When a product supervisor initiates on a given resource (i) an operation (OPk) associated with the event *RQ_OPk,i*, the resource moves from the *idle* state to a *working* state and does not process any request until it returns to the *idle* state.

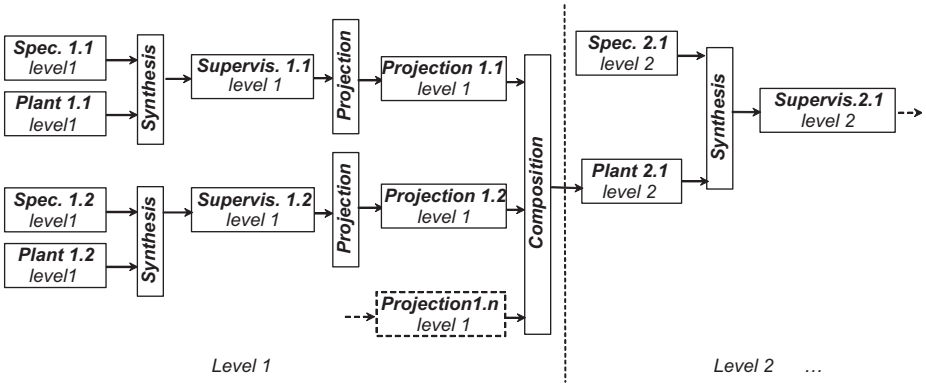



Fig. 8. Iterative synthesis method.

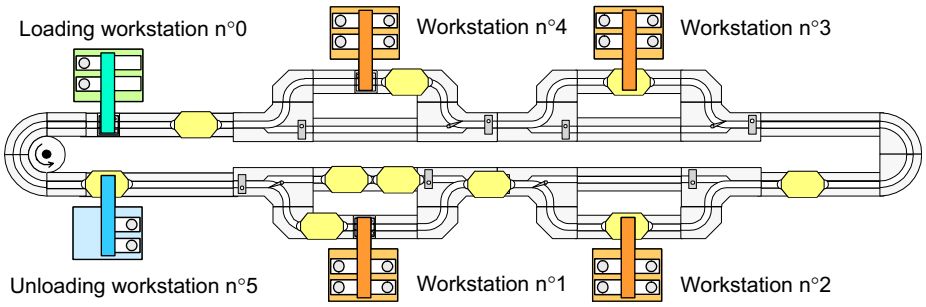



Fig. 9. AIPL Flexible assembly cell.

## 5. Application to the case study

### 5.1. Presentation of the case study

The concept of product-driven automation is illustrated in this paper using a *Flexible Assembly Cell* case study. This cell involves six workstations which are interconnected via a conveyor: one station for pallet loading, four similar assembly stations, and one station for pallet unloading (Fig. 9). Six different product families can be assembled (Fig. 10). Each workstation is able to perform from 1 to 4 assembly operations and involves a vacuum generator and three air cylinders to handle parts and products. Pallets are equipped with RFID tags, which correspond to the informational part of product controllers. A restriction is made so that each product will only go on one pallet during its assembly. Workstations are equipped with a Programmable Logic Controller (PLC), which implements the resource controller, and with two short distance RFID tag reader/writers. One is located before the workstation by-pass and the second is located in the working area of the station.

Applying the product-driven concept to the automation of this assembly cell leads to the following expected behaviour:

- When a product occurrence is planned to be manufactured, a model of the possible routings within the cell must be established, synthesised from product specification and the assembly workstation capabilities. This model represents a sequenced list of product states that takes into account the different admissible trajectories within the cell. Transitions between states correspond to reports which are received from the workstations or requests for transport or assembly operations. This model is embedded in RFID tags.
- Before each workstation, the RFID tag of the product is read. If a transition exiting from the active state corresponds to a transport request to a given workstation, the pallet is sent to the resource and a transport report is written on the RFID tag. If not, the pallet goes to the next workstation.
- When a product is present in the work area of a given workstation, the RFID tag is read. Transition exiting from the active state is interpreted by the resource controller as an operation request. The resource controller then executes the necessary actions to perform the requested operation. A report is emitted when the operation is correctly finished.
- Each time the RFID tag is written, a remote decision centre computes the new active state and the associated request of the routing model. Note that, if several solutions exist for a given operation (several machine are able to perform it), optimisation criteria such as workstation charge or routing time could be applied to select the requested resource.

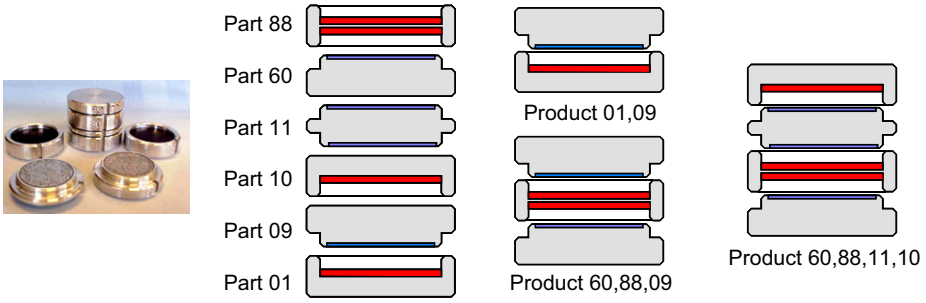



Fig. 10. Product types.

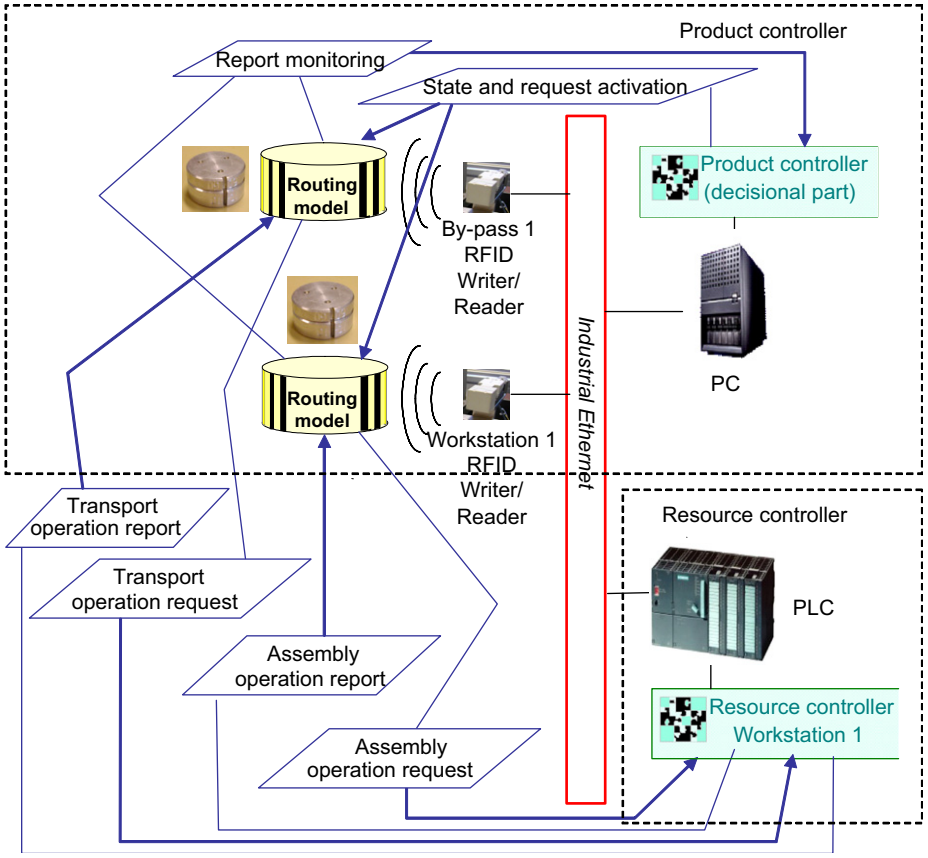



Fig. 11. Product-driven technical architecture.

The technical architecture of the product-driven control of the assembly cell is given in Fig. 11 for the workstation 1.

This technical implementation aspect is not further detailed in the paper which focuses on modelling and synthesis issues for product-driven automation.

### 5.2. *Product supervisor synthesis*

The alphabet used in this application is given in Table 1; for the product, requests are considered as controllable events while reports are uncontrollable.

Let us consider a 01–09 type product; it is composed of an assembly of 01 and 09 parts. According to Section 3.1, product specification is given by Fig. 12, where reports acknowledge the end of parts 01 and 09 assembly operations and the end of the unloading (RP 99).

Resource models are designed by adapting the generic models of Section 4.2.1 to the following capabilities (Fig. 13):

- workstations 0, 2, and 4 are able to assemble 88 and 01 parts, 09 parts, 11 and 09 parts respectively,
- workstation 5 is in charge of unloading the product out of the cell (operation noted 99),
- workstations 1 and 3 are unused and
- the transport system is a single conveyor that allows moves from any workstation to any other.

Table 1
Alphabet for product supervisor synthesis

| Label | Significance |
|---|---|
| RQ *N*,*i* | Request for assembling a part of type *N* by resource i |
| RP *N* | Report about assembling a part of type *N* |
| RQ pos *i* | Request for moving to position of resource *i* |
| RP pos *i* | Report about moving to position of resource *i* |


Fig. 12. Product manufacturing plan for 01–09 product (*specification*).

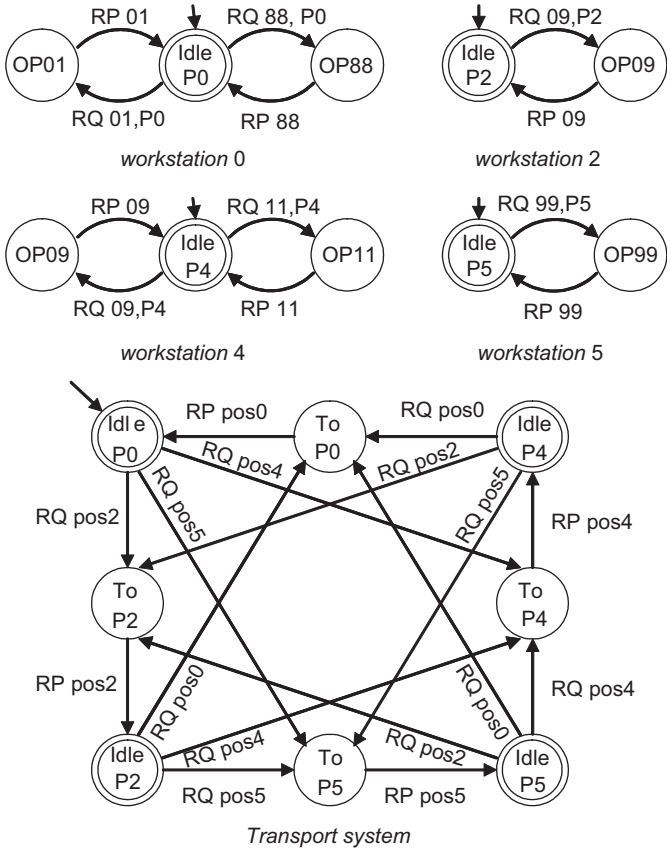



Fig. 13. Workstations and transport models.

The model of the assembly cell capabilities is designed by applying the fusion operator of the Section 4.1.2 to these resource models (Fig. 14). It describes the set of plant admissible behaviours without control specification.

A 01–09 product supervisor can be generated using the TCT[5] tool for synthesis procedures, from the automata of Fig. 12 (as *Specification*) and Fig. 14 (as *Plant*) (Fig. 15).

[5]Operations on automata (product, projection, synthesis) are made using the TCT software tool developed at the University of Toronto (http://odin.control.toronto.edu/DES/).

It defines all acceptable routings of the 01–09 product within the assembly cell according to the different assembly operations that this product has to undergo (i.e. according to the product specification presented in Fig. 12). In other words, it represents the maximal set of alternate routes for a given product within the cell which all lead to the expected manufactured product.

### 5.3. Resource supervisor synthesis

Application to the assembly cell of the iterative synthesis method shown in Fig. 8 leads to three hierarchical layers for the resource supervisors: layer one concerns the actuators (air cylinders and a vaccum generator), layer two concerns the *pick and place* function (involving vertical air cylinders and a vaccum generator) and *move* function (involving two horizontal air cylinders, and the third layer supplies the part manipulation function.

Details are given below for each layer of the iterative synthesis method (complete for the lowest one and partial for the higher ones), knowing that paper length and supervisor size inhibit the complete representation of all the models used and supervisors synthesised in this case study.

#### 5.3.1. Layer 1 supervisors

The workstations of the AIPL assembly cell are composed of air cylinders and a vacuum generator with their associated magnetic sensors to pick up, move, and finally assemble the parts.

The *generator* (or *plant*) generic model of double-acting air cylinders with their control valve is composed of two marked states in which the observable device behaviour is steady (a cylinder in a pushed or retracted position) and by two passing states, which represent the evolution from one steady state to another (Fig. 16). A complete list of events is provided in Table 2. The vacuum generator is modelled using two states: *sucking on* and *off*.

Actuator *specifications* are then given in a modular way. For the generic model of air cylinders, two automata describe rules that filter or validate some pushing and retracting requests according to the current state of the device (first left automaton in Fig. 17).

For example, after an initialisation phase considering uncontrollable events, a pushing order (*Pord*) is generated only after a pushing request (*rqP*), and there is always a retracting order (*Rord*) between two pushing orders (*Pord*). Two other automata describe the filtering of the observations. For example, a pushed position report (*rpR*) is generated only after a pushed position sensor event (*Ppos*) if a prior pushing order (*Pord*) is sent; a retracting order (Rord) invalidates this sequence (first right automaton in Fig. 17).

The complete specification of the air-cylinder control module results from the synchronous product of these four automata (21 states, 68 transitions). The same approach is used for the specification of the vacuum system (8 states, 25 transitions). Note that these specifications are not natural

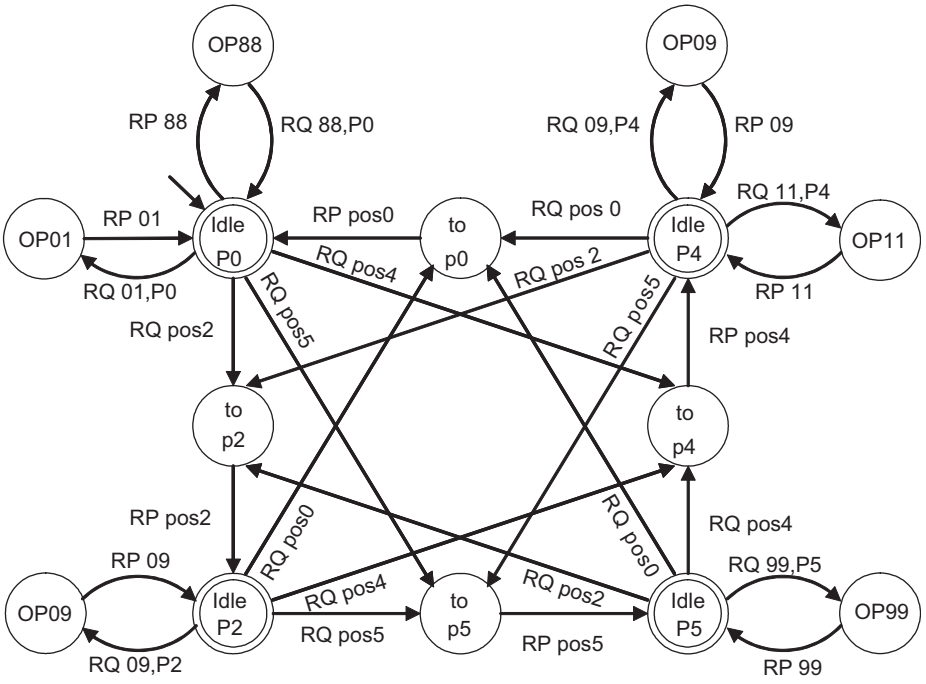



Fig. 14. The model of manufacturing system capabilities (*plant*).

at all and can be considered as a major difficulty of the synthesis process as further discussed in Gouyon et al. (2004).

Subsequently, the TCT tool is used to compute the supremal controllable sublanguage that defines the most permissive supervisor (Fig. 18), as is usually done in the SCT synthesis framework.

The most permissive supervisor is determined for each actuator (air cylinders and vacuum generator) of the workstation. Note that if similar actuators are involved, previous alphabets are prefixed to be disjointed.

### 5.3.2. Coordination supervisors

At layer 2, two main functions manage the manipulator *move* from a position to another one and the part *pick & place*. Both functions result from coordination of the actuators (two air cylinders for moving and one air cylinder and vacuum generator for picking and placing).

*Generator* models used for this synthesis phase are automatically generated from the actuator supervisors following the mechanism presented in Section 4.2: projection of actuator supervisors and synchronous composition. Fig. 19 shows the projection of the air-cylinder supervisor, which only preserves report and request events (orders sent to the valve and sensor data have been eliminated), used in the synchronous composition. Note that the generic air-cylinder supervisor is instantiated to each actuator by renaming the projected events (for example by adding an 'h' for the horizontal cylinder) to create a specific alphabet for each instance.

The *specification* model describes the actuator coordination rules. Fig. 20 shows an example of the coordination specification for the *pick & place* function (see Table 3). The TCT tool is then used to generate the *pick & place* supervisor (20 states, 63 transitions) and the *move* supervisor (138 states, 558 transitions).

Finally, the highest hierarchical layer (layer 3) is in charge of coordinating the *pick & place* and *move* functions to carry out a given assembly using part manipulation from one position to another. For example, assembling part 01 means moving to the place where 01 is stored, picking the part, moving back to the pallet and then placing the part on the semi-finished product.

*The generator* at this last layer is obtained by a projection and a synchronous composition of the *pick & place* and *move* supervisors.

The *specification* includes functional (sequence of moving and picking) and safety (no moves when picking) properties. According to predicates (2) and (3), the highest layer of resource specification makes the link between available operation requests and reports and the resource elementary actions which must be initiated to answer them. Projection of this specification by only preserving events that belong to the alphabet of the product supervisors describes the resource capabilities in term of operation requests and reports and is consistent with the *generator* model that is used in Section 4.1 to synthesise product supervisors. In other words, operation *OPi* that appears in the generator model in Section 3 is further detailed in several elementary states in the resource specification model while preserving the same entering and exiting events for the refined sequence.

Fig. 21 shows an example of such a specification for the Workstation 0 where operations *01* and *88* are supported.

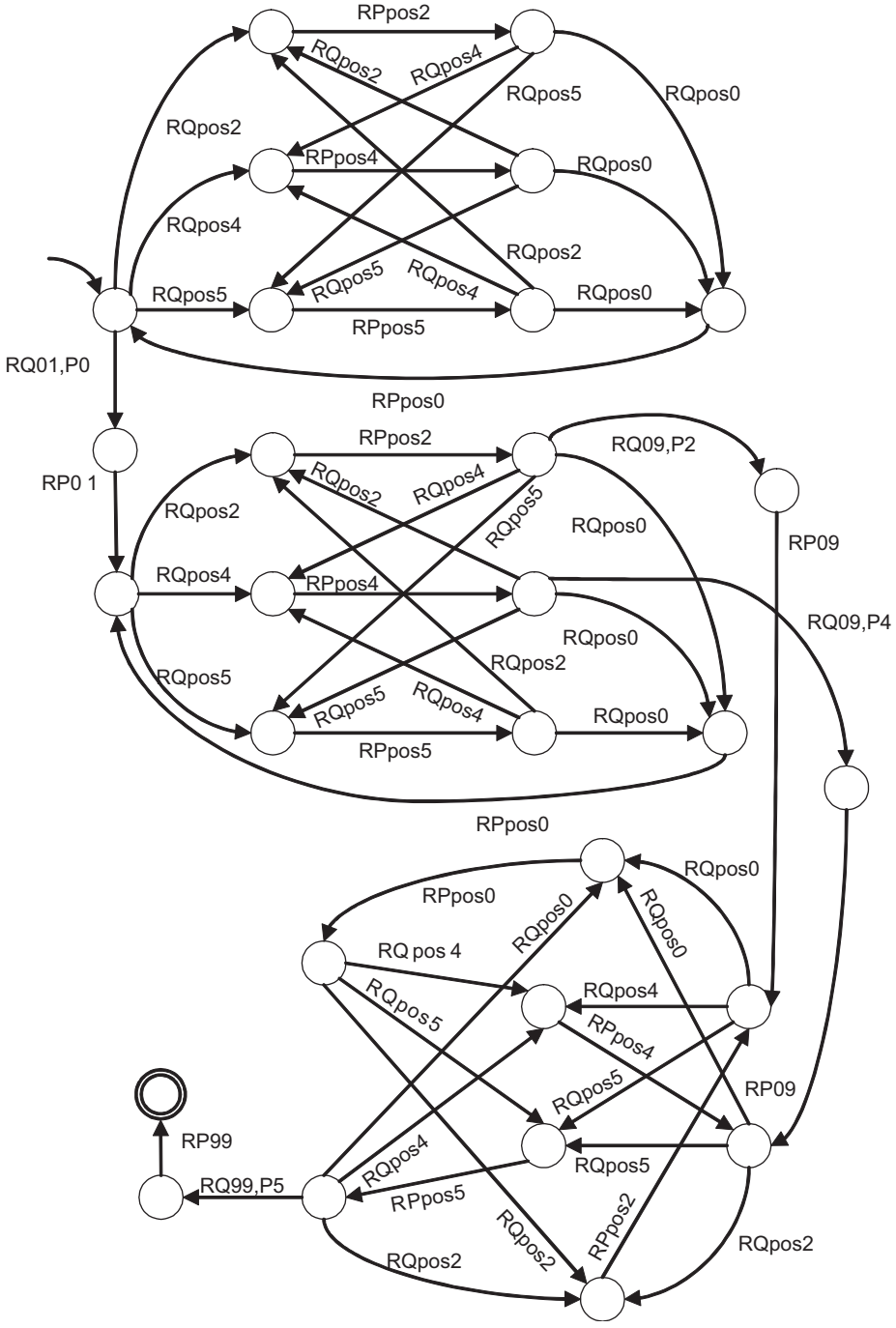



Fig. 15. 01–09 product supervisor.

*epr* and *epo* denote picked and placed reports, *dpr* and *dpo* picking and placing requests, *ei* represents the manipulator position and *di* the request for reaching a manipulator position.

The consistency between product and resource supervisor synthesis can then be demonstrated by proving that the resource specification model in Fig. 21 can be projected to obtain an associated resource model such as the ones presented in Fig. 13 (see Table 4):

- events kept by the projection are requests *rq01* and *rq88* as well as reports *rp01* and *rp88*,
- the two marked states are merged,
- the *move* and *pick & place* sequences are replaced by a single state, meaning that the resource is executing the required assembly operation.

The last layer of the resource supervisor can then be synthesised using the TCT tool and corresponds to an

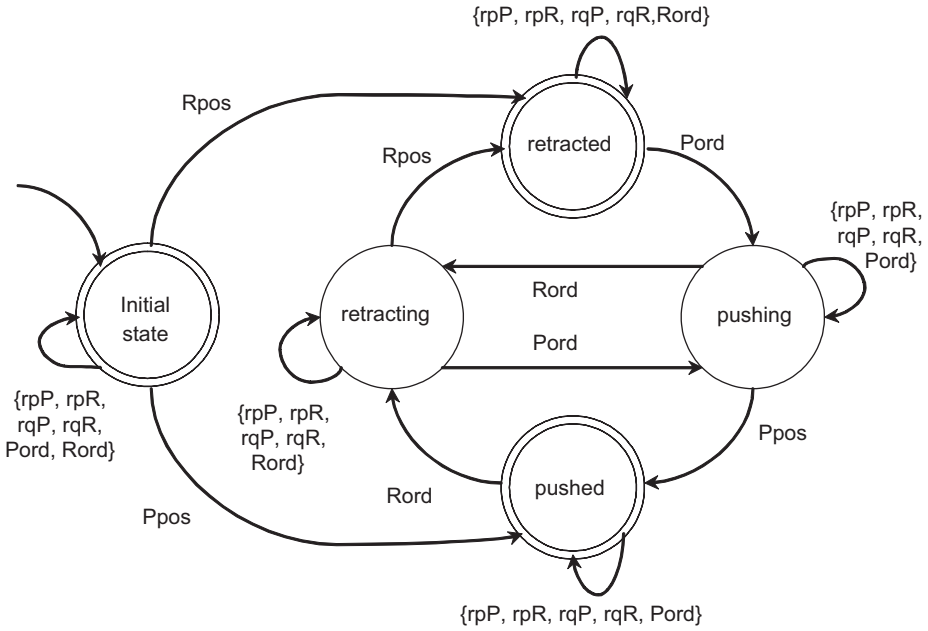



Fig. 16. Air cylinder *generator* (or *plant*) model.

Table 2
Events for Figs. 16–18

| Label | Significance | Controllability |
|---|---|---|
| rqR | Retracting request | Uncontrollable |
| rqP | Pushing request | Uncontrollable |
| rpR | Retracted position report | Controllable |
| rpP | Pushed position report | Controllable |
| Rord | Retracting order | Controllable |
| Pord | Pushing order | Controllable |
| Rpos | Retracted position sensor | Uncontrollable |
| Ppos | Pushed position sensor | Uncontrollable |

automaton (38 states, 43 transitions) that synchronises the actions required to carry out the assembly operations.

The final resulting control architecture is composed by three hierarchical layers: layer 1 involves 5 actuator supervisors (4 for the air cylinders and 1 for the vacuum generator), layer 2 involves one supervisor for part picking/placing and one supervisor for manipulator moves, and layer 3 supervisor coordinates the two layer 2 supervisors for part assembly.

## 6. Implementation issues

Implementation of product and resource controllers requires transforming the supervisors synthesised in Sections 3 and 4 to introduce deterministic choices. Indeed, there is a clear interpretation gap between the roles a supervisor is assumed to play within the SCT modelling framework and the roles a controller has to play within current practices in real-time control systems (Zaytoon & Carre-Menetrier, 2001).

Within the SCT framework, the process (generator) is assumed to generate events in a spontaneous manner. Therefore, the only way for the supervisor to affect the behaviour of the process is to enable or to disable the controllable events. Moreover, this supervisor is said to be a maximally permissive supervisor, meaning that it includes all legal process sequences for a given specification without providing choice criteria between two legal sequences of controllable events. However, a reactive control system is expected to force some events to occur, not only to enable and disable some of them. It is often based on a set of predetermined evolution rules which calculate the appropriate outputs (controllable events) to be applied to the process system according to its current state, given as inputs (uncontrollable events).

The gap between interpretations of the *supervisor* in SCT and the *controller* in the forcing events approaches (Marikar, Rotsein, & Macchietto, 1998) can be bridged by an input–output interpretation of the SCT controllable and uncontrollable events (Balemi et al., 1993). Such an interpretation can be direct (Nourelfath & Niel, 2004) if the supervisor sequences contain at least one controllable event between two uncontrollable events. This means that the supervisor reacts to a given input by emitting an output before a new input occurs. If this hypothesis is not verified, interpretation is indirect since it requires translating a supervisor into a controller according to a set of rules (Fabian & Hellgren, 1998; Marikar et al., 1998).

### 6.1. Implementation of resource supervisors

Translation from resource supervisors to deterministic resource controllers provided with an input-output interpretation is based on a priority allocation mechanism. The

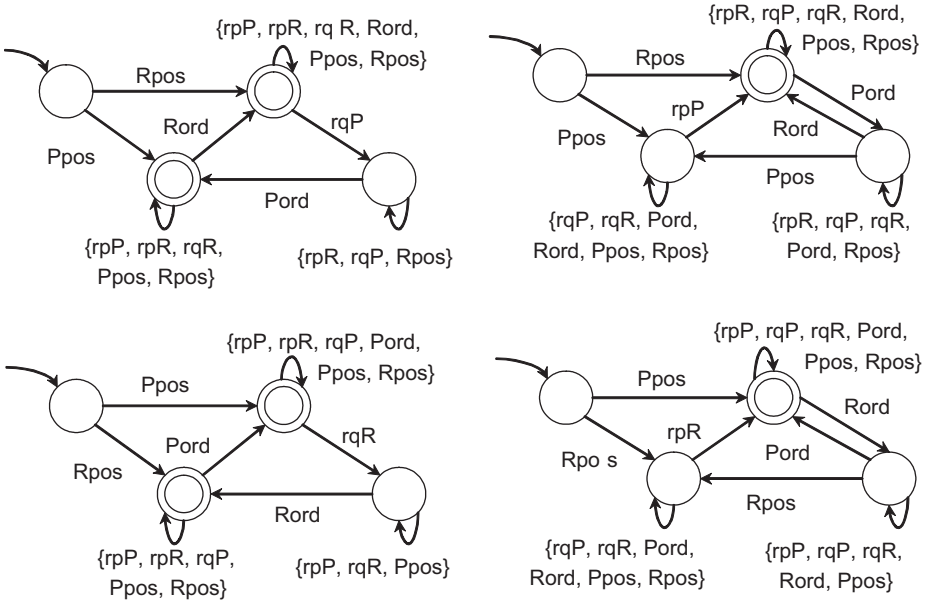



Fig. 17. Air cylinder specifications.

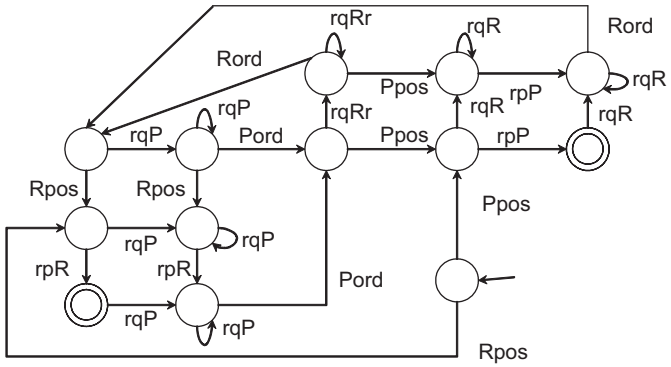



Fig. 18. The air cylinder supervisor.

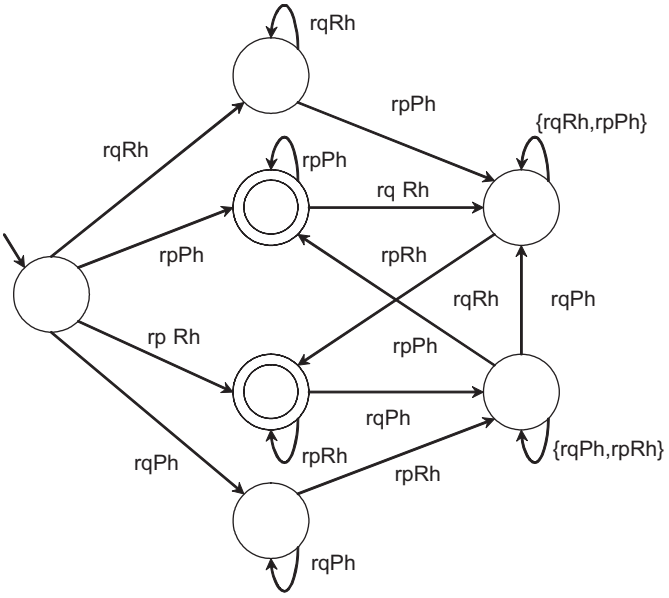



Fig. 19. Projection of the air-cylinder supervisor.

objective is to enable two transitions $t1$ and $t2$ to exit from the same given state S1 if and only if the two events associated with $t1$ and $t2$ are uncontrollable. This strong hypothesis is justified by the fact that:

- two controllable transitions exiting from the same state mean that two actions (outputs) are acceptable for the supervisor, but one of them needs to be forced by the controller,
- controllable and uncontrollable transitions exiting together from the same state mean that the actions associated with the controllable event (output) may be triggered or not depending on the sampling period in which the uncontrollable event is seen.

In the first case, priority will be given to the event that belongs to an alphabet of a higher-layer supervisor (in Fig. 22, *report* and *request* have higher priority than *action* and *observation*). In the second case, priority depends on event controllability, uncontrollable events having higher priority than controllable events (in Fig. 22, *observation* has higher priority than *action,* and likewise for *request* and *report*). If two events have the same hierarchical layer and the same controllability, allocating priority refers to a design choice.

SCT events are interpreted as inputs and outputs. Inputs are associated to uncontrollable events while outputs are associated to controllable events. More precisely, uncontrollable events are interpreted as rising edges of Boolean variables that trigger transitions. Controllable events of the supervisor are interpreted as rising edges that

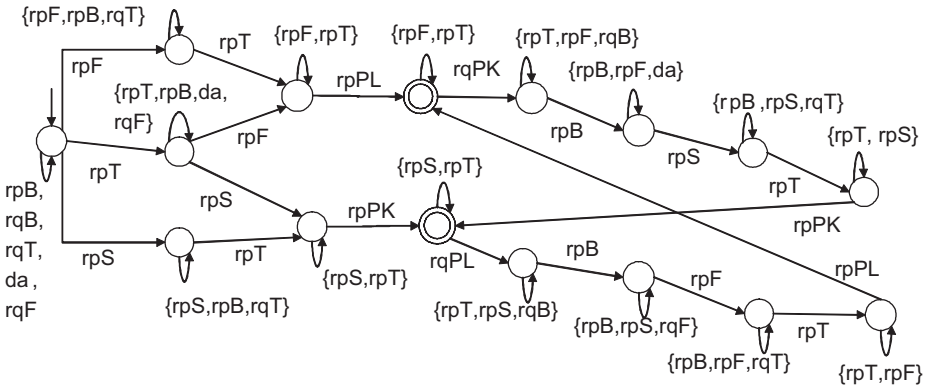



Fig. 20. *Pick & place* specification model.

Table 3
Events for Fig. 20

| Label | Significance | Controllability |
|---|---|---|
| rqS | Sucking request to vacuum cups | Controllable |
| rqF | Freeing request for vacuum cups | Controllable |
| rpS | Sucked Part report | Uncontrollable |
| rpF | Free Part report | Uncontrollable |
| rqT | Rise request (go to top) | Controllable |
| rqB | Go down request (go to bottom) | Controllable |
| rpT | "At the Top" report | Uncontrollable |
| rpB | "At the Bottom" report | Uncontrollable |
| rqPK | Pick request | Uncontrollable |
| rqPL | Place request | Uncontrollable |
| rpPK | Part picked | Controllable |
| rpPL | Part placed | Controllable |

activate transitions toward states in which outputs are produced and maintained until these states are deactivated. These coding rules are based on algebraic equations that synchronously activate ($Ai_t$) and deactivate ($Di_t$) a state ($Si_t$) in accordance with

$$Si_{t+1} = Ai_t \vee (Si_t \wedge \neg Di_t).$$

These algebraic equations can then be encoded into IEC 61131-3[6] PLC standard programming languages such as Ladder Diagram (LD) or Structured Text (ST). Each supervisor of the control hierarchy gives rise to an implementation module called Function Block (FB). Plugging these equations into FB requires using an execution algorithm that ensures equation scheduling. The most frequently used algorithm, *without stability search*, is based, initially, on the evaluation of the transitions that can be triggered, then on the calculation of the newly reached situation, and finally on the activation of the associated outputs. Fig. 23 shows an example of a supervisor implementation in LD language.

[6]Int. Electrotechnical Commission, *IEC 6113-3 Programmable controllers, Part 3 programming languages*, 2000.

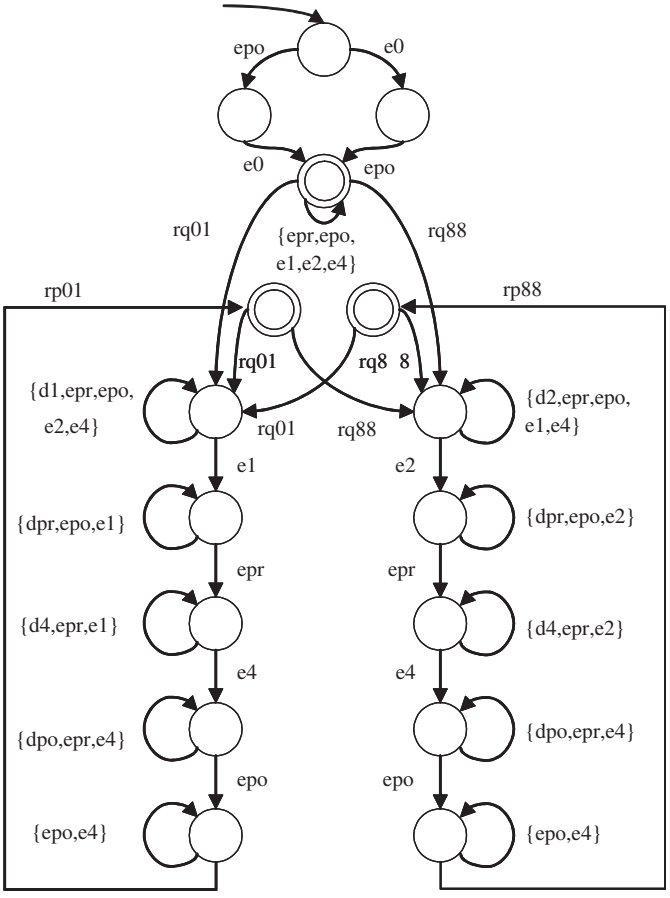



Fig. 21. Highest-layer specification.

To facilitate the implementation of the resource supervisors, a dedicated software tool (TCT2LD) has been developed:

- to read the supervisors description files given as input by the TCT tool,
- to provide an automatic (if possible) or interactive way of allocating priorities to transform the supervisors into controllers and

Table 4
Events for Fig. 21

| Label | Significance | Controllabilty |
|---|---|---|
| rp01 | Assembling 01 report | Controllable |
| rp88 | Assembling 88 report | Controllable |
| rq01 | Assembling 01 request | Uncontrollable |
| rq88 | Assembling 88 request | Uncontrollable |
| epo | Product placing report | Uncontrollable |
| epr | Product picking report | Uncontrollable |
| dpo | Product placing request | Controllable |
| dpr | Product picking request | Controllable |
| e0 | Position 0 reached | Uncontrollable |
| e1 | Position 1 reached | Uncontrollable |
| e2 | Position 2 reached | Uncontrollable |
| e4 | Position 4 reached | Uncontrollable |
| d1 | Position 1 request | Controllable |
| d2 | Position 2 request | Controllable |
| d4 | Position 4 request | Controllable |

- to generate LD programs for downloading into PLCs.

Using this set of tools (TCT and TCT2LD), the resource controllers of the assembly cell have been successfully implemented and tested.

### 6.2. Implementation of product supervisors

Product supervisors represent all acceptable routings within the plant in such a way that the product specification is established. Bridging the gap between product supervisor and product controller requires removing indeterministic situations. This happens when there are two transitions exiting from the current product state. In this case, the product controller has to make a decision to select one of the acceptable manufacturing trajectories and

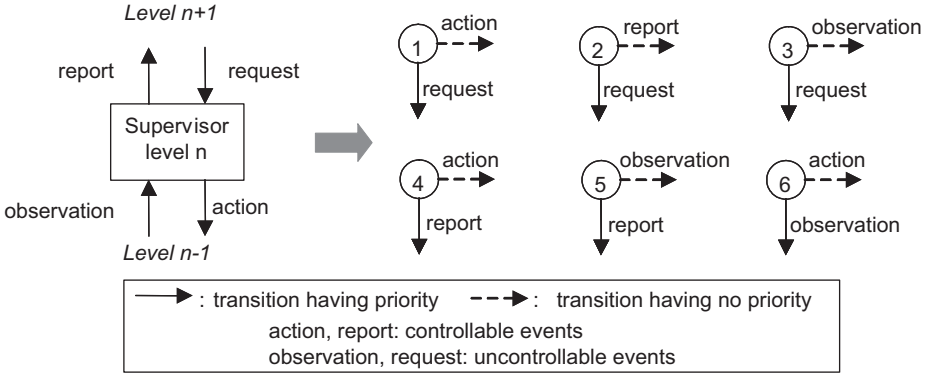



Fig. 22. Priority allocation.

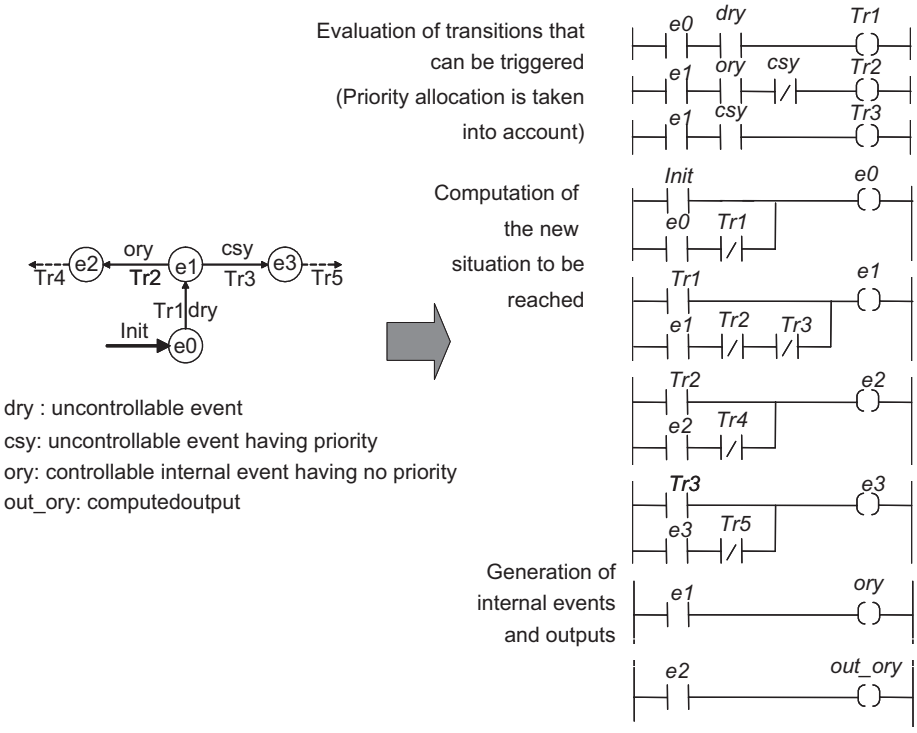



Fig. 23. From supervisor to Ladder Diagram.

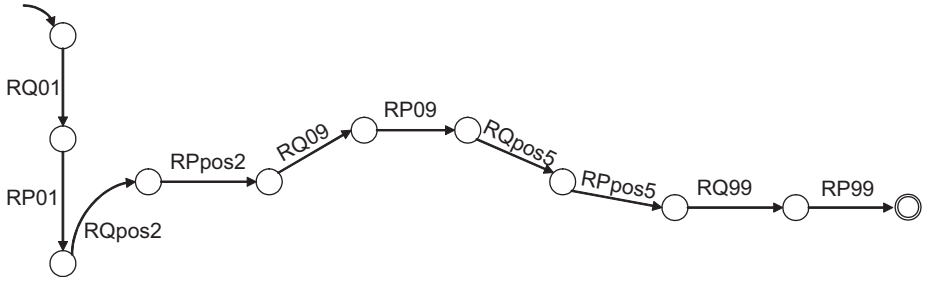



Fig. 24. Selected manufacturing route.

then call for the corresponding resources. If the other cases, the product controller applies the only admissible sequence, as proposed by the supervisor.

Consequently, solving the determinism problem requires making a decision for choosing one of the admissible sequences according to external criteria such as resource performance, resource availability, transport time to resource or status of the resource. This problem is similar to path research within an automaton and solutions have been proposed using static or dynamic costs associated with each transition (Marchand, Boivineau, & Lafortune, 2000), or optimisation algorithms, such as Dijkstra's algorithm (Dijkstra, 1959). Static cost will help in defining the chosen trajectory before production while dynamic costs will help in defining in real time the accurate trajectories.

This technique has been applied by defining transition costs as the product of the time to move from one manufacturing state to another and the quantity of product inside the resource buffers (space between by-pass and workstation). When an indeterministic situation occurs, the optimisation algorithm is executed to choose the next state among all admissible states. Fig. 24 shows an example of a manufacturing route that results from successive dynamic choices based on the product supervisor given in Fig. 15.

The product supervisor is then considered as the informational part of the product controllers. This part is coded into RFID tags thanks to arrays including the name and activation of product states as well as the name and the triggering state of transitions. The decisional part of the product controller is coded in Java and implemented in a remote computer.

## 7. Conclusions and open issues

This work is part of a research project on product-driven automation for business-to-manufacturing purposes (Morel, Panetto, Zaremba, & Mayer, 2003). This paper focuses on the design and implementation of a product-driven control system. On-the-fly reconfiguration is the main property, which is addressed to face variability of customised products and justifies the use of automatic synthesis techniques. The main objective is to provide an iterative modeling and synthesis method within the context of SCT to ensure the interoperability between product controllers which manage product routings within the manufacturing systems, and resource controllers which manage the execution of manufacturing operations.

However, even if the application of the approach to the case study has been successful, some limits of SCT modelling and synthesis must be mentioned.

The size of the generated supervisors is the major problem of synthesis algorithms, even if used for a pedagogical case study. Modularity is a classical way to handle this problem. For example, the modular and iterative synthesis of resource controllers of the assembly cell leads to a supervisor with a maximum of 138 states and 558 transitions. This can be considered as huge for human analysis of the result but can be easily implemented on PLCs with the use of tools such as TCT2LD. However, when the routing complexity increases, implementation of product supervisors could be a real difficulty. Splitting the product *specification* into sub-plans and isolating *plant* islands are open issues that could introduce modularity in the product supervisor synthesis.

The robustness of the synthesis algorithms is questionable (Gouyon et al., 2004); indeed, on the one hand, the resulting supervisor strongly depends on the way the *plant* and *specifications* have been modelled; on the other hand, human based modelling activities can give rise to a wide range of models that more or less cover the initial needs. Consequently, the modelling phase remains a major difficulty for the synthesis, especially when using finite state machines. This justifies further developments, such as synthesis based on the refinement of algebric equations (Roussel, Faure, Lesage, & Medina, 2004) or based on scheduling algorithms (Henry et al., 2004).

As addressed by Qiu et al. (2003), the above-mentioned problems could put into question the application of synthesis techniques for MES industrial applications but also opens onto complementary experiments.

## References

Achour, Z., Rezg, N., & Xie, X. (2004). Supervisory control of marked graphs with partial observations. *International Journal of Production Research*, *42*(14), 2827–2838.

Balemi, S., Hoffmann, G. J., Gyugyi, P., Wong-Toi, H., & Franklin, G. F. (1993). Supervisory control of a rapid thermal multiprocessor. *IEEE Transactions on Automatic Control*, *38*, 7.

Basile, F., Carbone, C., & Chiacchio, P. (2006). Simulation and analysis of discrete-event control systems based on Petri nets using PNetLab. *Control Engineering Practice*, *15*(2), 241–259.

Berruet, P., Toguyeni, A. K. A., Elkattabi, S., & Craye, E. (2000). Toward an implementation of recovery procedures for flexible manufacturing systems supervision. *Computers in Industry*, *43*(3), 227–236.

Brennan, R. W., Zhang, X., Xu, Y., & Norrie, D. H. (2002). A reconfigurable concurrent function block model and its implementation in real-time java. *Journal of Integrated Computer-Aided Engineering*, *9*, 263–279.

Cassandras, C. G., & Lafortune, S. (1999). *Introduction to discrete event systems*. Dordrecht: Kluwer Academic.

Chafik, S., & Niel, E. (2000). Hierarchical-decentralized solutions of supervisory control. In *Proceedings of the third international symposium on mathematical modelling*. Vienna, Austria.

Da Silveira, G., Borenstein, D., & Fogliatto, F. S. (2001). Mass customization: Literature review and research directions. *International Journal of Production Economics*, *72*, 1–13.

De Queiroz, M. H., & Cury, J. E. R. (2002). Synthesis and implementation of local modular supervisory control for a manufacturing cell. In *Proceedings of the sixth international Workshop on discrete event systems* (pp. 377–382). Zaragoza, Spain.

Dijkstra, E. W. (1959). A note on two problems in connexion with graph. *Numerische Matematik*, *1*, 269–271.

Endsley, E. W., Almeida, E. E., & Tilbury, D. M. (2006). Modular finite state machines: Development and application to reconfigurable manufacturing cell controller generation. *Control Engineering Practice*, *14*, 1127–1142.

Fabian, M., & Hellgren, A. (1998). PLC-based implementation of supervisory control for discrete event systems. In *Proceedings of the 37th IEEE conference on decision and control*. Tampa, Florida.

Fusaoka, A., Seki, H., & Takahashi, K. (1983). A description and reasoning of plant controllers in temporal logic. In *Proceedings of the eighth international joint conference on artificial intelligence* (pp. 405–408). Karlsruhe, Germany.

Gohari, P., & Wonham, W. M. (1998). Hierarchical supervisory control of discrete-event systems. In *Proceedings of the fourth IFAC workshop on discrete event systems*. Cagliari, Italy.

Gouyon, D., Pétin, J.-F., & Gouin, A. (2004). Pragmatic approach for modular control synthesis and implementation. *International Journal of Production Research*, *42*(14), 2839–2858.

Henry, S., Zamai, E., & Jacomino, M. (2004). Real time reconfiguration of manufacturing systems. In *Proceedings of the IEEE conference on systems, man and cybernetics*. The Hague, Netherlands.

Iung, B., Neunreuther, E., & Morel, G. (2001). Engineering process of integrated-distributed shop floor architecture based on interoperable field components. *International Journal of Computer Integrated Manufacturing*, *14*(3), 246–262.

Jiang, S., & Kumar, R. (2000). Decentralized control of discrete event systems with specializations to local control and concurrent systems. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, *30*(5), 653–660.

Koren, Y. (2005). Reconfigurable manufacturing and beyond. In *Proceedings of the CIRP third international conference on reconfigurable manufacturing*. Ann Arbor, MI, USA.

Kumar, R., Garg, V., & Marcus, S. L. (1991). On controllability and normality of discrete event dynamical systems. *Systems & Control Letters*, *17*, 157–168.

Lauzon, S. C., Mills, J. K., & Benhabib, B. (1997). An implementation methodology for the supervisory control of flexible manufacturing workcells. *SME Journal of Manufacturing Systems*, *16*(1).

Leduc, R. J. (2002). Hierarchical interface-based supervisory control. *Doctoral Thesis*, Department of Electrical & Computer Engineering, University of Toronto.

Lennartson, B., Tittus, M., Fabian, M., & Hellgren, A. (1998). Specification structures for supervisory control. In *Proceedings of the fourth IFAC workshop on discrete event systems*. Cagliari, Italy.

Marchand, H., Boivineau, O., & Lafortune, S. (2000). On the synthesis of optimal schedulers in discrete event control problems with multiple goals. *SIAM Journal on Control and Optimization*, *39*(2), 512–532.

Marchand, H., Bournai, P., Le Borgne, M., & Le Guernic, P. (2000). Synthesis of discrete-event controllers based on the signal environment. *Discrete Event Dynamic Systems: Theory and Applications*, *10*, 325–346.

Marikar, M.T., Rotsein, G.E., & Macchietto, S. (1998). An integrated environment for the design of procedural controllers. In *Proceedings of the ninth IFAC/INCOM symposium*. Nancy, France.

McFarlane, D., Sarma, S., Chirn, J.L., & Ashton, K. (2002). The intelligent product in manufacturing control and management. In *Proceedings of the 15th triennal IFAC world congress*. Barcelone, Spain.

Morel, G., Panetto, H., Zaremba, M., & Mayer, F. (2003). Manufacturing enterprise control and management system engineering: Rationales and open issues. *IFAC Annual reviews in Control*.

Muhl, E., Charpentier, P., & Chaxel, F. (2003). Optimization of physical flows in an automotive manufacturing plant: Some experiment and issues. *Engineering Application of Artificial Intelligence*, *16*, 293–305.

Newman, W. S., Podgursji, A., Quinn, R. D., Merat, F. L., Branicky, M. S., Barendt, N. A., et al. (2000). Deign lessons for building agile manufacturing systems. *IEEE Transactions on Robotics and Automation*, *16*(3), 228–238.

Nourelfath, M., & Niel, E. (2004). Modular supervisory control of an experimental automated manufacturing system. *IFAC Control Engineering Practice*, *12*, 205–216.

Ollero, A., Morel, G., Bernus, P., Nof, S. Y., Sasiadek, J., Boverie, S., et al. (2002). From MEMS to Enterprise systems. *IFAC Annual Reviews in Control*, *26*(2), 151–162.

Qiu, R., Wysk, R., & Xu, Q. (2003). Extended structured adaptive supervisory control of shop-floor controls for an e-manufacturing system. *International Journal of Production Research*, *41*(8), 1605–1620.

Ramadge, P. J., & Wonham, W. M. (1987). Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, *25*(1).

Roussel, J. M., Faure, J. M., Lesage, J. J., & Medina, A. (2004). An algebraic approach for dependable logic control systems design. *International Journal of Production Research*, *42*(14).

Staroswiecki, M., & Bayart, M. (1996). Models and languages for the interoperability of smart instruments. *Automatica*, *32*(6), 859–873.

Toguyeni, A. K. A., Craye, E., & Sekhri, L. (2006). Study of the diagnosability of automated production systems based on functional graphs. *Mathematics and Computers in Simulation*, *70*(5–6), 377–393.

Tsubone, H., & Horikawa, M. (1999). A comparison between machine flexibility and routing flexibility. *The International Journal of Flexible Manufacturing Systems*, *11*, 83–101.

Vahidi, A., Fabian, M., & Lennartson, B. (2006). Efficient supervisory synthesis of large systems. *Control Engineering Practice*, *14*(10), 1157–1167.

Valckenaers, P. (Ed.) (2001). Holonic manufacturing systems [Special issue]. *Computer in Industry*, *46* (3), 233–331.

Vogrig, R., Baracos, P., Lhoste, P., Morel, G., & Salzemann, B. (1987). Flexible manufacturing shop. *Manufacturing Systems*, *16*(3).

Vyatkin, V. V., Chistensen, J. H., & Martinez Lastra, J. L. (2005). OOONEIDA: An open, object-oriented knowledge economy for intelligent distributed automation. *IEEE Transactions on Industrial Informatics*, *1*(1).

Wonham, W. M., & Ramadge, P. J. (1987). On the supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, *25*(3).

Wright, P. K., & Bourne, D. A. (1988). *Manufacturing intelligence*. Reading, MA: Addison-Wesley.

Yoo, T.-S., & Lafortune, S. (2002). A general architecture for decentralized supervisory control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, *12*.

Zaytoon, J., & Carre-menetrier, V. (2001). Synthesis of a correct control implementation for manufacturing systems. *International Journal of Production Research*, *39*, 329–345.

Zamaï, E., Chaillet-Subias, A., & Combacau, M. (1998). An architecture for control and monitoring of discrete events systems. *Computers in Industry*, *36*(1–2), 95–100.

1st IFAC Workshop on Dependable Control of Discrete Systems, Cachan, France, June 13-15 2007

# USING SYSML FOR IDENTIFICATION AND REFEFINEMENT OF MACHINERY SAFETY PROPERTIES

**Dominique EVROT[1+2], Jean-François PETIN[1], Gérard MOREL[1], Pascal LAMY[2]**

*(1) Université Henri Poincaré, CRAN, UMR 7039 CNRS-INPL-UHP*
*{dominique.evrot}{jean-francois.petin, gerard.morel}@cran.uhp-nancy.fr,*

*(2) INRS, BP 27 54501 Vandoeuvre lès Nancy cedex*
*pascal.lamy@inrs.fr*

Abstract: In the context of the development of systems subjected to strong dependability and safety properties, standards such as the IEC 61508 recommend the use of formal verification tools. In this way, conceptual and practical approaches related to computer sciences and automatic control, such as model checking, theorem proving, control synthesis, have been widely explored. However, in spite of the consensus that early phases of a system definition are the most important in ensuring that the target system will satisfy the user's requirements, most of these models and tools address the design and implementation phases where the identification and formalisation of system properties remain tricky. This machinery-dedicated paper combines system specification models supported by SysML to identify the system properties and architecture with model checker. This method is based on the refinement of system global requirements and their projection on the system components to formalise local properties to be proved by the model checker. A mechanical press case study illustrates this approach. 

Keywords: Requirements analysis, Safety analysis, SysML, System properties refinement, Machinery control.

## 1. INTRODUCTION

In machinery applications, safety functions can now be realized with safety Programmable Logic Controller (PLC). A bad implementation of this safety PLC can generate dysfunctions being able to cause losses of production or operator's hazards. In France, one of the role of the Institut National de Recherche et de Sécurité (INRS) is to anticipate future risk by acquiring new knowledge and converting current knowledge into practical know-how available to professional in charge of prevention.

Standards not dedicated to machinery sector, such as the safety-related IEC 61508[1] already strongly recommend the use of formal methods to control the complexity of software-intensive applications. Conceptual and practical approaches have been widely explored by organizations related to computer sciences and automatic control: examples are software verification by theorem proving (Abrial, 1996), model checking (Clarke *et al.*, 2000), or automatic synthesis (Cassandras and Lafortune, 1999). However, in spite of the consensus that early phases of a system definition are the most important in ensuring that the target system will satisfy the user's requirements, most of these models and tools address the design and implementation phases where the identification and formalisation of system properties remain tricky (Barragan *et* al, 2006). Over these later stages, many systems engineering and automation engineering practitioners (Johnson, 2004) consider that the time is ripe to formalize the earlier stages of specification.

This paper explores a machinery-dedicated combination of semi-formal models such as SysML, to identify, specify and refine system properties and architectures with formal models to be proved by a model checker. Second section presents the limits of Discrete Event System (DES) approaches for identification and specification of system properties

[1] IEC 61508, *Functional safety of electrical/electronic/ programmable electronic (E/E/PE) safety-related systems*

and highlights the lack of verification tools for SysML models. Section 3 proposes a combination of these approaches for identification, refinement and verification of safety properties. Section 4 illustrates this approach using a mechanical press case study. The final section provides some conclusions and open issues.

## 2. PROBLEM STATEMENT

Machinery safety functions that have been traditionally realized with electromechanical circuits are increasingly realized with programmable logic controllers. This is due to the increasing complexity of control command functions and the needs of maintenance and reconfiguration of industrial systems. To ensure the safety of the whole system, from operative part to man machine interface, including control software, it is necessary:

- to identify safety properties that formalize potentials risks and expected behaviors at a system level,
- to verify that the designed system or its models meet these properties.

Sections 2.1 and 2.2 show how system engineering oriented tools and discrete events systems oriented ones deal with these issues: safety properties identification and systems verification and validation.

### *2.1 System oriented approach*

Standards define system engineering as an "interdisciplinary collaborative approach to derive, evolve and verify a life cycle balanced system solution which satisfies customer expectations and meets public acceptability" (IEEE 1220[2]). This definition meets our aim that is to identify, verify and validate safety system requirements issued from an interdisciplinary approach. Some of system engineering tools, based on the Unified Modeling Language (UML[3]) and its extensions, cover the various aspect of system modeling, while other ones focus on a specific part such as requirements expression (DOORS[4]) or documentation traceability (Reqtify[5]).

UML and its extension UML2.0, is a system modeling toolbox that should address use cases identification and architectural, behavioral design of the system. These tools allow a multi-trades approach due to their high level of abstraction and its graphic notation. However, it seems that due to its origin and primary application being software-oriented, this language cannot be adequate for all aspects of systems engineering (hardware, software, man-machine interface…). The new profile SysML has been developed to meet system engineering community expectations. These new toolbox contains new diagrams, like the requirement diagram, that allows a most efficient system analysis and design. Even if UML provides a wide notation toolbox that covers the requirements in terms of expressiveness for industrial system specification and modelling (Panetto and Pétin, 2005), it suffers from a lack of formal semantics for validation and verification issues. Efforts have been made towards UML formalization (Laleau and Pollack, 2002) but the diagrams consistency and correctness verification is still an open issue.

### *2.2 DES oriented approach*

DES community developed tools and methods for the control systems verification to:

- prove model properties, such as analysis techniques for Petri Nets or Finite State Automata, theorem proving and model checking based on the building of the reachable states space of the control software (Clarke et al., 2000),
- to prove PLC programs properties by applying formal techniques to check the properties satisfied by controllers implemented using IEC 61131-3 programming languages (Roussel and Faure, 2002).

Even if these approaches have been proved to be efficient in the design and implementation phases, two common characteristics make them inadequate for the specification phase. First all these tools require the previous identification of software safety requirements that is not guided by the approaches and which remain tricky (Barragan *et* al, 2006). In addition, model checkers require the formalization using temporal logics of these requirements into safety properties. Second, expression of the underlying mathematical representation is limited to the modelling of system dynamics, and hardly covers the description of other system properties. More precisely, the decomposition of a complex system in several sub-systems, involving software, hardware and man-machine interface components, does not facilitate the traceablity and the refinement of global system properties through the sub-system decomposition, design, verification and validation. This aspect requires method that enables model refinement to capture very abstract requirements including safety properties and to project them into all system components.

Our objective is to bridge the gap between a necessary semi-formal approach, such as SysML, to capture and understand the global system requirements and DES model checkers that are efficient for validation and verification of safety properties.

[2] Standard for application and management of systems engineering

[3] OMG, Object Management Group, www.omg.org

[4] Telelogic, www.telelogic.com

[5] Tni-software, www.tni-software.org

## 3. PROPOSAL

It seems the best way is to couple system engineering languages for requirements identification and expression with tools for conception and verification of automated systems. The main difficulty is to trace requirements through these tools. The scope of this communication is to illustrate a method to refine safety system requirements and allocate them to obtain components architecture for the command software.

### *3.1 Requirements identification*

Requirements are the formalization of the client needs for one part and the formalization of provider limits or standards prescriptions for other ones. Identify them is the first step of the process, and we use the SysML requirements diagram to identify and to structure system requirements. Requirements diagram allow two kinds of link between requirements, a composition link and a derivation link. A composition link between two requirements means that the composed requirement is realized if and only if all the components requirements are realized. This link allows decreasing the abstraction level. A derivation link between two requirements means that, the derivate requirement existence is implied by the first requirement existence. There is no verification or realization consideration. This link allows adding new requirements at the same abstraction level. Identification of safety requirements is realized according to the ISO 12100-1[6] risk reduction method. Requirements are structured by using a refinement mechanism inspirited by the refinement present in the B method (Abrial, 1996; Evrot *et al*, 2006). When a requirement is defined, it can be formalized in a property in order to highlight the logical relation involved in the requirement. It is helpful for the projection of requirements on the different parts of the system. When the requirements are totally refined, when they are linked to components, then the properties are translated in a formal language.

### *3.2 Functions and components identification*

Components and functions are the concrete aspect of requirements. Requirements formalize a goal, functions are assembled to meet this goal and components realized functions. We use the SysML blocks diagram to represent the components architecture of the system; activities and sequences diagrams are used to represent the functions and the logical architecture of the system.

In manufacturing industry, reutilization takes an important part in system engineering. Requirements establishment is still the first step of the development but all oldest system in use bring knowledge about its realization in terms of components and functions. So the development process is driven both by the requirements refinement and identification and by the integration of knowledge of previous developments.

This is an iterative process to solve the Fusaoka's automation assertion (Fusaoka, 1983), which is here expressed in a system view (Pétin *et al.*, 2006):

$$\textit{Control Specifications} \wedge \textit{Process Specifications} \supset \textit{System Specifications} \quad (1)$$

The process starts with a definition of the system requirements. That allows proposing architecture for the operative or control part, or a logical architecture for the system, according to the imported knowledge. This architecture definition brings details needed for the requirements refinement, projection, analysis, and allocation. Then the cycle restart with more detailed requirements that lead to more detailed architectures.

### *3.3 Behavior design*

Requirements have been formalized in the requirements diagram and then components and functions have been identified and linked to requirements. Now we have to design the expected behavior of each components and functions. This design is realized with the state machine diagram of SysML or with specifics tools for the design of automation components. In this case SysML components diagram must be exported as an input for the design tools. Formal verification tools like UPPAAL[7] model checker can be used to ensure that the designed component, model or implementation verify the safety properties that have been formalized sooner from the requirements it is linked to. Safety properties must be translated in temporal logic language usually used by model checkers.

### *3.4 Traceability*

We have seen that system development deals with various notions between requirements capture phase and components specification phase. We have discussed about requirements, properties, components and functions. Use an object oriented language like UML is very helpful because all this notions can be considered as objects. From that assumption, we can formalize interactions between all these objects in a data model (fig. 1) that is adapted from the AFIS[8]'s data model. Data model is a support for the requirements traceability because of links that allow following the development process from the requirements to the components.

[6] ISO 12100-1 – Safety of machinery – Basics concepts, general principles for design. Part 1: Basic terminology, methodology.

[7] www.uppaal.com

[8] Association Française d'Ingénierie Système, www.afis.fr

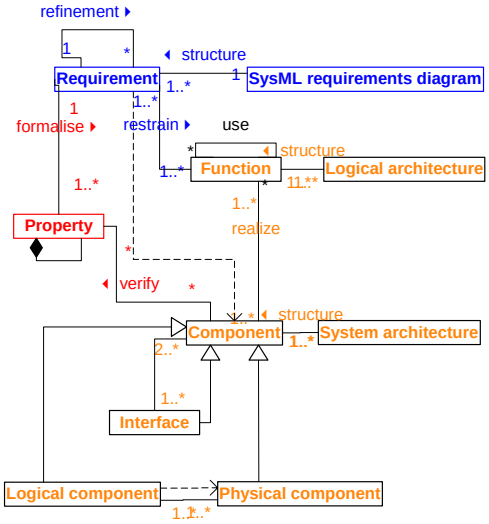



Fig. 1. Data model for the development process.

## 4. APPLICATION

In this paper, we consider a mechanical press that aims at stamping metal products with a tool in vertical movement. A crankshaft equipped with two sensors for top and bottom dead centres gives this vertical movement. A clutch, which is actuated by a pneumatic valve, ensures the transmission between motor and crankshaft. We will present a small part of the press development. Requirements identification lead to identify that a setting mode is needed to test various systems settings. Each identified requirement must be subject to a safety analysis, in order to identify risk and reduce it. Risk reduction requirements are modelled by a safety requirement put on the analysed requirement. In this case risk analysis identify the stamping movement as a hazard and a safety requirement is linked to the setting mode requirement. This safety requirement express that if operator is able to enter the working space, then the stamping movement must be shut down (fig. 2.).

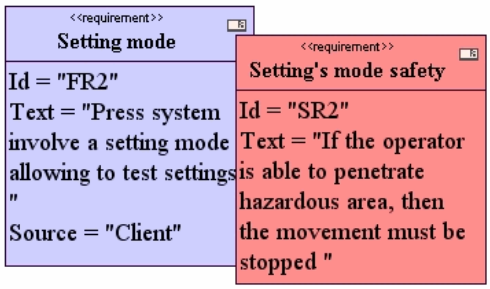



Fig. 2. SysML requirement diagram of setting mode

In the following, we will focus on the "setting mode safety" requirement refinement, allocation and verification. First, according to the data model, it must be formalized into safety property:

P1: *free access to the working area ⇒ press movement off* is linked to SR 2

P1 is a system property; each part of the press must satisfy it including control system, crankshaft, clutch, motor and operator. But without a more detailed expression of P1, we are not able to verify it. So the next step is the refinement of system architecture definition and then system requirements. In order to restrict dangerous area access for the operator, a two hands command is added in the system physical architecture. Requirements are detailed to ensure that actuation of two hands command prevent operator to access dangerous area. They precise where the two hand command must be placed, how it must be designed to prevent frauds, how it must be actuated etc. Use of a safety related standard is helpful to identify these requirements, in this example some of the following requirements are based on a standard dedicated to two hands commands:

- Two hands command control prevents each fraud allowing that both devices can be actuated without two hands.
- Two hands command is placed far enough from the dangerous area to prevent a penetration before the movement stop.
- Two hands command is activated only if both devices have been actuated in a 0,5 s interval.
- Two hands command is deactivated if one device is released.
- Two hands command is reactivated only if both devices have been released.
- Two hands command is deactivated if a fault is detected.

Requirements diagram illustrating refinement of requirements SR 2 is given figure 3. Composition links shows that if the six lowest requirements are satisfied then "two hands command" requirement is also satisfied. Requirements formalization and P1 projection can be done in three steps. First, formalization of requirement SR 2-1 that details how dangerous area's access is restricted. The associated safety property:

P2: *Two hands command not activated ⇒ free access to the working area*, is linked to SR 2-1

Second, we formalize requirements SR2.1.1to6 in properties. That is to highlight logical links involved in the text description of the requirements and to prepare properties projection on the different system's components. Properties are:

P3: *both devices actuated ⇒ two hands used*, is linked to SR 2-1-1
P4: *Two hands command activation ⇒ both devices have been released since the previous one*, is linked to SR 2-1-2
P5: *Two hands command's fault ⇒ bimanual control deactivation*, is linked to SR 2-1-3
P6: *Two hands command's activation ⇒ actuation of both devices 0,5s apart*, linked to SR 2-1-4
P7: *distance between two hands command and dangerous area > X meters* is linked to SR 2-1-5
P8: *Two hands command activated ⇒ both devices actuated*, is linked to SR 2-1-6

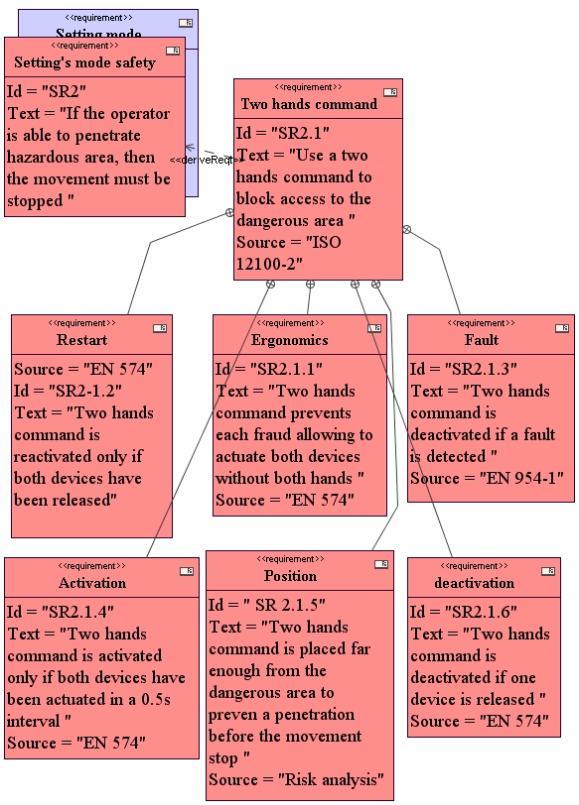



Fig. 3. Requirement diagram of two hands command

Third, these requirements are a more detailed view of the requirement SR 2-1. SR 2-1 is satisfied if and only if SR 2-1-1 to 6 are satisfied. Considering that system must satisfy P1 and P2, we can write:

P1 AND P2 become:
*P1': (Two hands command not activated⇒press movement off) AND P2,*
*With P2 = P3 AND P4 AND P5 AND P6 AND P7 AND P8*

So system must satisfy P1' and P3 and P4 and P5 and P6 and P7 and P8. Then with these more detailed requirements, we are able to define physical and logical architectures for the bimanual control. We choose a classical physical architecture with two devices, a cover, two switches for faults detections, and software.

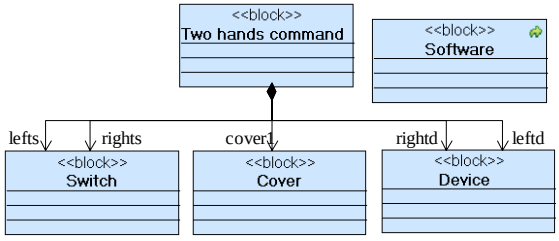



Fig. 4 Block definition diagram of two hands command physical architecture

Requirements allocation allows defining which component(s) realize(s) which requirement(s). Two hands command requirements allocation is given in table 1.

| Requirements | Components |
|---|---|
| SR 2 | All |
| SR 2-1-1 | Cover |
| SR 2-1-2 | Software / Device (l & r) |
| SR 2-1-3 | Software / Switch (l & r) |
| SR 2-1-4 | Software / Device (l & r) |
| SR 2-1-5 | Cover |
| SR 2-1-6 | Software / Device (l & r) |

Table 1. Requirements allocation

The movement part of requirement SR 2 is refined like the dangerous area access has been. That leads to refine properties that system must assume.

P9: *Clutch system in braking position⇒press movement off*
P10: *electromagnet not commanded ⇒ Clutch system in braking position.*
Then P1' become:
P1'': *Two hands command not activated ⇒ electromagnet not commanded AND P9 AND P10*

Finally if we project all these properties on the software, according to the requirements allocation, we obtain that software only satisfy partly P1'' and P4 and P5 and P6 and P8. Then a functional analysis shows that software assumes 4 functions describe in the following table.

| Requirements | Functions |
|---|---|
| SR 2 | SF1: braking decisions<br>SF4: clutch movement control |
| SR 2-1-2 | SF2: THC activation |
| SR 2-1-3 | SF3: THC fault computing |
| SR 2-1-4 | SF3: THC activation |
| SR 2-1-6 | SF3: THC activation |
| Clutch require. | SF4: clutch movement control |

Table 2: requirements constraints functions

These functions are implanted in 3 functional blocks:

- "TH command filter" component involving SF2 and SF3
- "Clutch filter" component involving SF4
- and "Setting's safety" involving SF1

So we obtain the following software architecture (fig. 5 and 6). The functional analysis leading to functions identification is realized using UML sequence and activity diagrams. Following definition of variables exchanges is allowed by analysis of these diagrams. Ports allow exchanging variables between blocks through provided and required interfaces. These interfaces are defined in a block diagram not represented here.

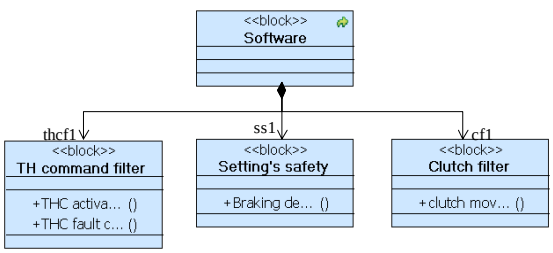



Fig. 5. Block definition diagram of software

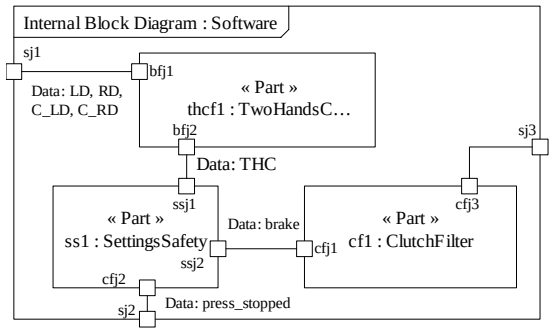



Fig. 6. Internal block diagram of software block

The final step is to project software properties on each functional block. Data model allow us to links properties with components, by using links between components and requirements, components and functions, functions and requirements, properties and requirements. We found that "Two Hands command Filter" block must satisfy P4, P5, P6, P8 and that "clutch filter" and "setting's safety" blocks must satisfy P1'' together. So properties are projected on components inputs and outputs:

P1'': *not THC* ⇒ *not clutch* become
P1.a: *not THC* ⇒ *brake* or *press stopped* and
P1.b: *brake or press stopped* ⇒ *not clutch*
P4: ↑THC ⇒ ↓RD and ↓LD since the previous ↑THD
P5: not((RD xor C_RD) and (LD XOR C_LD))⇒ not THC
P6: ↑BD ⇒ ↑RD and ↑LD for less than 0.5s
P8: THC ⇒ RD and LD

So we have identified safety properties that the software must satisfy. These properties must be expressed in the language used by the verification tool. Model checkers use temporal logic to express properties. Most of the previous properties can easily be expressed in this language, but P4 and P5 are more complicated. To express them it is necessary to used observatories. An observatory is a state machine that represents the different states defines in the property. Property P4 involves 3 states: the first one with THC active, the second one with THC inactive and both devices released and a third one, halfway, with THC inactive and only one device released. The property formalizes that activate THC is only possible if both devices have been released since the previous activation. So when state 3 is active (THC have just been deactivated), it is possible to reach state 1 only by activating state 2. So we add an extra state called "default" that can be reached from state 3 if THC is activated. P4 means "default" state can't be reached.

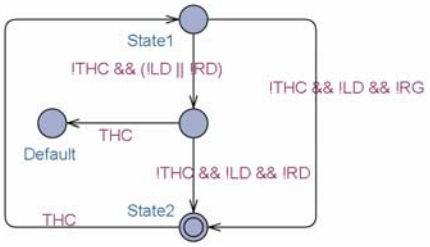



Fig. 7.State machine observatory

Figure 7 show the state machine designed with UPPAAL model checker to verify property P4.

## 5. CONCLUSION

Because safety properties for model checking are uneasy to identify and formalize *a posteriori*, we propose a method that include safety properties identification and formalisation during the design of the system. This method not only allows identifying safety properties for the control software that must be check, but also for the other subsystems. This point is very helpful in wide project because it provides to the subcontractors the properties their subsystem must satisfy. This method is based on the object-oriented language SysML that involves a wide ability for modelling. To use Non-formal language is necessary to apprehend a complex system, but we are aware that the development of the model is driven by a not proved refinement process. Works have been realized to combine UML non-formal language and B formal language (Laleau and Pollack, 2002), and so we are trying to complete this method with more formal method like the B method.

## 6. REFERENCES


Abrial J.R. (1996). *The B Book: Assigning Programs to Meanings*. Cambridge Univ. Press

Barragan S., Roth M., Faure J. M., (2006) Obtaining temporal and timed properties of logic controllers from fault tree analysis, in proc of*12th IFAC INCOM* vol 1 *pp 241-246* , St-Etienne, France, 17-19/05.

Cassandras C.G., Lafortune S. (1999). *Introduction to Discrete Event Systems*. Kluwer Academic Publisher, ISBN 0-7923-8609-4.

Clarke E.M., Grunberg O., Peled D.A. (2000) *Model Checking*. The MIT Press

Evrot D., Pétin J-F., Mery D., Formal specification of safe manufacturing machines using the B method: application to a mechanical press, *in proc. 12th IFAC INCOM*, vol 1 pp 277-282, St-Etienne, France, 17-19

Fusaoka A., Seki H., Takahashi K. (1983). A description and reasoning of plant controllers in temporal logic. *International Joint Conference on Artificial Intelligence.* pp 405-408. Karlsruhe, 8-12 aug. 1983.

Johnson T. L. (2004). Improving automation software dependability: a role for formal methods? In proc. of *11th IFAC/INCOM*, Salvador-Bahia, Brazil,4-7/04/04

Laleau R., Pollack F., (2002) Coming and going from UML to B: a proposal to support traceability in rigorous IS development. In Proc of *ZB'2002*, *LNCS vol. 2272*, pp 517–534, Grenoble, France.

Panetto H., Pétin J-F., (2003). Metamodelling of production systems process models using UML stereotypes. *International Journal of Internet and Entreprise Management*, vol. 3, n° 2, pp. 155-169, ISSN 1476-1300.

Pétin J-F., Morel G., Panetto H. (2006) Formal specification method for systems automation. *European Journal of Control*, vol. 01, ISSN 0947-3580.

Roussel J.-M., Faure J.-M., (2002). An algebraic approach for PLC programs verification, *in proceedings of WODES'02*, Zaragoza, Spain.