Méthodes et outils informatiques pour la commande locale ou distante de systèmes réactifs

Philippe Le Parc

université de bretagne occidentale

Habilitation à Diriger des Recherches

Version finale

Méthodes et outils informatiques pour la commande locale ou distante de systèmes réactifs

Philippe Le Parc

EA2215 - Projet **L**angages et **I**nterfaces pour **M**achines **I**ntelligentes (LIMI)

Département Informatique
U.F.R. des Sciences et Techniques
Université de Bretagne Occidentale
20, avenue Le Gorgeu
CS 93837
29238 BREST cedex 3
FRANCE

Soutenue le 8 Juillet 2004 devant un jury composé de :

MM.:	Marcel LE FLOCH	Président
	Thierry DIVOUX	Rapporteur
	Eric GRESSIER SOUDAN	Rapporteur
	Nicolas HALBWACHS	Rapporteur
	Nicolas GUYOMARD	Examinateur
	Lionel MARCÉ	Examinateur

à Christel,
à Claire, François et Sophie.

Table des matières

Table des figures

Remerciements

Mon travail de chercheur de troisième cycle, puis d'enseignant chercheur s'est déroulé à l'Université de Bretagne Occidentale dans le cadre du laboratoire Langages et Interfaces pour Machines Intelligentes (LIMI), aujourd'hui intégré dans l'Equipe d'Accueil (EA) en Informatique 2215. Merci donc au Professeur Lionel Marcé pour m'y avoir accueilli, encadré et soutenu tout au long de ces treize années de travail.

Merci à M. Thierry Divoux, Professeur au CRAN de Nancy, à M. Eric Gressier, Professeur au CNAM de Paris et à M. Nicolas Halbwachs, Directeur de Recherches à l'Imag de Grenoble pour avoir bien voulu rapporter sur mon Habilitation à Diriger des Recherches.

Merci également à M. Marc Guyomard, Professeur à l'ENSSAT de Lannion et à M. Marcel Le Floch, Professeur à l'UBO de Brest pour avoir accepté de participer au jury.

Ce travail, comme le montre la bibliographie, n'aurait pas été possible sans collaboration avec Dominique L'Her, Pascal Ogor, Jean-Luc Scharbarg et Jean Vareille. Qu'ils soient ici remerciés pour leur aide, leur compétence et leur complicité. Dominique nous a quitté, mais son souvenir reste présent.

Que les membres passés et présents du LIMI et de l'EA2215 recoivent également mes remerciements pour offrir un cadre de travail agréable et une solidarité de tous les instants. Merci également aux personnels techniques et aux membres du secrétariat pour leur aide et leur disponibilité.

Introduction

Ce document synthétise près de treize années de travail en tant que chercheur de troisième cycle, puis d'enseignant chercheur au sein du département d'Informatique de l'UFR des Sciences et Techniques de l'Université de Bretagne Occidentale (UBO) de Brest.

Au cours de ces années, j'ai essayé d'exercer une activité équilibrée entre les tâches d'enseignement, d'administration et de recherche. Cet équilibre a été parfois difficile à trouver, en raison de la mutation profonde de l'enseignement d'informatique à l'UBO, avec la création d'un Institut Universitaire Professionnalisé (IUP) Ingénierie Informatique, à moyens humains presque constants. Néanmoins, l'équipe de recherche à laquelle je contribue, est passée du statut de Jeune Equipe au statut d'Equipe d'Accueil, grâce à la qualité du travail de ses membres.

Mon activité de recherche ne s'est jamais arrêtée et j'ai régulièrement publié mes travaux que ce soit dans des revues ou lors de congrès, au niveau international et national. J'ai ainsi apporté des contributions dans deux domaines connexes : la modélisation et la validation de langages métier de l'automatisme et le contrôle à distance de systèmes mécaniques sur des réseaux de communication non fiables.

Ces deux thématiques se rejoignent sur de nombreux points :

- Le domaine d'application, c'est à dire le contrôle de systèmes en environnement de production industriel. D'un côté, on a cherché à assurer une meilleure qualité, de la spécification à l'implantation en passant par la vérification des applications. De l'autre en proposant l'utilisation des nouvelles technologies de communication pour améliorer la productivité globale de l'entreprise. Ces travaux se placent donc tout naturellement dans le domaine dit des "systèmes réactifs" par opposition aux "systèmes transformationnels" ou aux "systèmes interactifs" [BB91].

- L'utilisation et l'introduction des méthodes et des outils de l'informaticien auprès de la communauté des automaticiens. Dans mon premier domaine de recherche, par l'intermédiaire du monde dit "synchrone" et dans le deuxième par la mise en oeuvre des technologies autour de l'Internet. Dans les deux cas, les technologies informatiques se sont placées au service d'autres disciplines tout en essayant de faire ressortir les points difficiles et de proposer des solutions innovantes.

- Le souci permanent de construire des applications sûres et de valider, si possible de manière formelle, les résultats obtenus. Cet aspect est surtout visible dans le cadre de la première activité de recherche, mais des travaux, non encore achevés, ont également porté sur la validation de l'architecture logicielle proposée pour le contrôle à distance. Ils sont directement issus des connaissances et de l'expérience acquise dans le premier thème.

Ce travail a été accompagné à la fois par d'autres collègues mais aussi à travers l'encadrement de nombreux étudiants, de la licence au doctorat.

Parallèlement à cette activité de recherche, j'ai assuré un enseignement à tous les niveaux universitaires en essayant d'être le plus pédagogique et le plus rigoureux possible. J'ai également apporté mon énergie au niveau administratif, en m'occupant en autre des stages en entreprise pour les formations professionnalisées et également en développant les relations internationales du département et de l'IUP informatique.

Le chapitre 1 de ce document présente un résumé de ces treize dernières années au niveau enseignement, administration et recherche. Dans le chapitre 2, ce dernier point est développé. Le chapitre 3 se veut prospectif et essaye donc de tracer quelques grandes lignes directrices, principalement au niveau recherche.

Chapitre 1

Présentation générale

1.1 Etat civil

Philippe LE PARC
Né le 16 août 1968 à Lorient (Morbihan)
Marié - 3 enfants (Claire, François et Sophie)

Adresse professionnelle :
Université de Bretagne Occidentale
Département d'Informatique
20 av. Victor Le Gorgeu
CS 93837 - 29238 Brest Cedex 3
Téléphone : 02.98.01.69.60
Fax : 02.98.01.62.52
Mail : Philippe.Le-Parc@univ-brest.fr
Page Web : http://www.ea2215.univ-brest.fr

1.2 Etudes supérieures

- Juin 1986 : baccalauréat C - mention Assez Bien - Lycée Dupuy de Lôme - Lorient.
- Septembre 1986 - juin 1987 : première année Institut National des Sciences Appliquées (INSA) de Rennes.
- Septembre 1987 - juin 1988 : deuxième année INSA Rennes.
- Septembre 1988 - juin 1989 : troisième année INSA Rennes. Admission dans le département d'informatique.
- Septembre 1989 - Juin 1990 : quatrième année INSA Rennes. Stage de 6 semaines chez Siemens AG à Munich (Allemagne).
- Septembre 1990 - Juin 1991 : obtention du titre d'ingénieur de l'INSA Rennes et DEA d'Informatique de l'Université de Rennes I. Stage de DEA effectué sous la direction du Professeur Lionel Marcé à l'Université de Bretagne Occidentale (UBO) de Brest, en relation avec l'entreprise CEGELEC. Ce stage avait pour thème l'étude d'un langage pivot public pour les langages de l'automatisme.
- Septembre 1991 - Janvier 1994 : thèse de doctorat, de l'Université de Rennes 1, sur le thème : "Apport de la méthodologie synchrone pour la définition et l'utilisation du langage Grafcet". Cette thèse a été réalisée dans la cadre du laboratoire Langages et Interfaces pour Machines

Intelligentes (LIMI) de l'UBO, dirigé par le Professeur Lionel Marcé. Ce travail a reçu une mention très honorable de la part du jury, qui était composé de :

MM.:	Jean-Pierre BANÂTRE	Président
	François PRUNET	Rapporteur
	Olivier ROUX	Rapporteur
	Jean-Bernard KOECHLIN	Examinateur
	Paul LE GUERNIC	Examinateur
	Jean-Jacques LESAGE	Examinateur
	Lionel MARCÉ	Examinateur

1.3 Service national

– Février 1994 - Janvier 1995 : Service militaire effectué en tant que scientifique du contingent au sein de l'Ecole Nationale Supérieure des Ingénieurs de Etudes et Techniques d'Armement (ENSIETA) sous la responsabilité de M. Jean-Luc Fleureau.

Au cours de ces douze mois, j'ai enseigné le langage Grafcet aux élèves ingénieurs et participé aux travaux pratiques d'automatisme. En parallèle, j'ai travaillé sur le projet de robotique mobile Marc'h.

1.4 Carrière universitaire

– Recrutement en juin 1994 par la commission de spécialistes $27^{ème}$ section.

– Prise de fonctions en février 1995 en tant que Maître de Conférences $2^{ème}$ classe.

– Promotion en septembre 1999 en tant que Maître de Conférences $1^{ère}$ classe. Promotion obtenue au niveau national (CNU 27).

1.5 Résumé des activités d'enseignement

Mes enseignements se sont déroulés majoritairement au sein du département d'Informatique de l'UFR Sciences et Techniques de l'UBO. Ils ont principalement porté sur les langages informatiques, sur les systèmes d'exploitation et les réseaux. J'ai pu enseigner à tous les niveaux universitaires, du DEUG au DESS et, dès le début de ma carrière, j'ai eu la charge d'enseignements complets (CM, TD et TP).

Actuellement, je suis responsable de l'Unité d'Enseignement (UE) "Compilation" en Maîtrise Informatique, de l'UE "Systèmes Réactifs et Intelligents" en Maîtrise Informatique, de l'élément constitutif (EC) "Réseaux" de l'UE "Systèmes d'Exploitation et Réseaux" en Licence, de l'EC "Systèmes d'Exploitation et Réseaux" de l'UE "Option Informatique" en Deug MIAS/SM/STPI $2^{ème}$ année et de l'EC "Langages Synchrones" de l'UE "Systèmes, Réseaux et Temps Réel" en DESS Informatique et Automatisation de la Production (DESS IAP).

En 1999-2000, j'ai également enseigné la compilation en IUP Ingéniérie Informatique, niveau licence.

La table 1.5 présente, de manière synthétique, les différents enseignements auxquels j'ai pris part avec une répartition Cours Magistraux (CM), Travaux Dirigés (TD) et Travaux Pratiques (TP). Hors encadrements et suivis de stages, je me suis efforcé d'effectuer un volume d'enseignement proche du service normal statutaire (192h équivalent TD).

Matière	1995-1996	1996-1997	1997-1998	1998-1999	1999-2000	2000-2001	2001-2002	2002-2003	2003-2004
Langage Pascal									
	TD : 18	TD : 18	TD : 18	TD : 18					
Deug 1^{ere} année	TP : 32	TP : 32	TP : 32	TP : 32					
Systèmes d'Exploitation	CM : 10	CM : 20	CM : 20	CM : 20	CM : 20	CM : 20	CM : 20	CM : 10	CM : 10
et Réseaux	TD : 10	TD : 10	TD : 10	TD : 10	TD : 10	TD : 10	TD : 10	TD : 10	
Deug 2^{eme} anée	TP : 20	TP : 20	TP : 10	TP : 20	TP : 20	TP : 20	TP : 20	TP : 10	TP : 20
Algorithmique									
	TD : 12	TD : 12	TD : 12						
Licence	TP : 8	TP : 8	TP : 8						
Réseaux								CM : 8	CM : 8
						TD : 8	TD : 8	TD : 16	TD : 16
Licence						TP : 16	TP : 16	TP : 24	TP : 24
Compilation				CM : 20	CM : 20	CM : 20	CM : 20	CM : 20	CM : 20
		TD : 20	TD : 20	TD : 20	TD : 20	TD : 20	TD : 40	TD : 20	TD : 20
Maitrise		TP : 40	TP : 40	TP : 40	TP : 20	TP : 20	TP : 20	TP : 40	TP : 40
Systèmes Réactifs et		CM : 6	CM : 9	CM : 10	CM : 9	CM : 10	CM : 10	CM : 10	CM : 10
Intelligents		TD : 8	TD : 9	TD : 10	TD : 10	TD : 9	TD : 9	TD : 8	TD : 8
Maitrise		TP : 10	TP : 10	TP : 10	TP : 14	TP : 10	TP : 10	TP : 8	TP : 14
Langages Synchrones		CM : 2	CM : 6	CM : 6	CM : 6	CM : 6	CM : 6	CM : 6	CM : 6
		TD : 4	TD : 2				TD : 2	TD : 2	TD : 2
DESS		TP : 8	TP : 6	TP : 6	TP : 6	TP : 6	TP : 6	TP : 6	TP : 6
Compilation									
					TD : 30				
IUP2 infomatique					TP : 14				
Total (éq. TD)	95	192,67	194,17	214	201,83	192,33	201	195,67	194,33
Encadrement de projets									
	2	3	4	4	8	4	5	5	3
tous niveaux confondus									
Suivi de stages									
DESS infomatique		8	8	8	10	12	9		
IUP Informatique				10	21	34	33	20	11

1.6 Résumé des activités administratives

1.6.1 Commission de spécialistes

Je suis membre de la commission de spécialistes, $27^{ème}$ section, de l'UBO depuis l'année universitaire 1999-2000, dans un premier temps en tant que membre suppléant puis en tant que membre titulaire. Depuis 2002, je suis assesseur de cette commission.

1.6.2 UFR Sciences et Techniques

Je suis membre élu, rang B, du Conseil d'Administration de l'UFR Sciences et Techniques depuis mai 2003.

1.6.3 Département Informatique

Je suis membre élu, rang B, du Conseil du Département d'informatique.

Je suis co-responsable pour la rédaction de la demande d'habilitation du Master Professionnel en informatique dans le cadre de la réforme LMD : animation de groupes de travail, synthèse, rédaction de documents, réunions avec les autres départements de l'UFR Sciences...

1.6.4 DESS Informatique

De septembre 1992 à septembre 2002, j'ai été responsable adjoint de DESS Informatique et Automatisation de la Production (IAP), en charge plus particulièrement des projets de fin d'études (sur 8 semaines) et des stages de fin d'études (4 à 6 mois en entreprise). Pendant cette période, j'ai donc eu à gérer ces deux points pour plus d'une centaine d'étudiants. J'ai personnellement suivi en entreprise 55 étudiants.

De plus, j'ai participé au renouvellement de l'habilitation, aux réunions de jury ainsi qu'à la rédaction de plaquettes de présentation de la formation.

J'ai pris en charge également la gestion des anciens étudiants du DESS : enquètes sur les débouchés, annuaire des anciens, liste de diffusion et repas annuel. Actuellement, un contact régulier est conservé avec 120 anciens étudiants (sur les 170 diplomés) et en 2002, plus de 60 étudiants ont participé au repas annuel.

1.6.5 IUP Informatique

De septembre 1998 à septembre 2002, j'ai été responsable des stages à l'IUP Informatique. Au cours de cette période, plus de 450 stages ont été effectués sous ma responsabilité et j'en ai personnellement suivi 129 depuis septembre 1998.

Mes tâches principales étaient les suivantes : information aux étudiants, promotion de l'IUP auprès des entreprises, diffusion des offres de stage, mise en place d'un modèle pour l'évaluation, organisation des visites en entreprises et des soutenances de stage.

Cette fonction de responsable des stages m'a également amené à faire partie du conseil de perfectionnement de l'IUP ainsi que des jurys des trois années.

1.6.6 Relations internationales

Depuis 1997, et cela sous mon impulsion, le département et l'IUP d'informatique entretiennent des relations internationales, dans le cadre Socrates, avec quatre universités européennes, Aalborg Universitet au Danemark, Oulu Polytechnic en Finlande, l'Université de Liège en Belgique et la Fachhochschule Ostfriesland/Oldenburg/Whilemshafen (FHOOW) d'Emden en Allemagne :

- Départ d'étudiants : 36 étudiants, en période pratique d'études d'une durée de 3 mois, 28 étudiants pour un semestre d'études complet d'une durée de 5 mois.

- Accueil d'étudiants : 7, pour des stages en laboratoire.
- Stages industriels à l'étranger : 2 séjours de 6 mois chez Fujitsu Londres et 4 séjours pour un total de 16 mois dans l'entreprise Videra Oy d'Oulu.

Personnellement, j'ai effectué cinq missions de mobilité enseignante Socrates (rencontres avec des enseignants et enseignements en anglais pour 8h) : Emden (Allemagne) 1999, Oulu (Finlande) 2000, Esbjerg/Aalborg (Danemark) 2001, Emden (Allemagne) 2002, Oulu (Finlande) 2004.

Réciproquement, j'ai favorisé la venue de collègues étrangers : le professeur Wolfgang Mauersberger (3 mois en 1998, 1 semaine en 1999, 1 mois en 2000, 1 semaine en 2002, 1 mois en 2003), le professeur Rolf Socher Ambrosius (5 semaines en 1999), le professeur Uffe Kock Will (2 semaines en 2001) et le professeur Craig Smith (1 semaine en 2003).

J'ai également participé à trois missions relations internationales pour le compte de l'UFR Sciences et de l'UBO (Groningen (Pays-Bas) en 1998, Emden (Allemagne) en 2001, Belgique/Pays-bas/Allemagne en 2003) et organisé en juin 1998, la visite au sein de l'UFR Sciences et Techniques de 11 universités partenaires.

Enfin, lors de la venue des experts de Comité Nationale d'Evaluation (CNE), en janvier 2003, j'ai été invité comme " personne ressource " aux entretiens concernant les relations internationales au niveau de l'UFR Sciences et Techniques et au niveau de l'Université.

Je suis membre de la cellule Relations Internationales de l'UFR Sciences et Techniques depuis sa création (2000).

Dans ce cadre des échanges internationaux, j'ai proposé au niveau Master IUP 2^{me} année, une option internationale composée d'un semestre d'études et d'un stage à l'étranger. Cette proposition s'appuie sur les contacts actuels et si elle est acceptée dans le cadre de la réforme du LMD, cinq étudiants devraient la suivre dès la rentrée 2004-2005.

1.6.7 Promotion Enseignement et Recherche

- Fête de la Science 2001 : démonstration de l'application de télé-contrôle à distance de caméras motorisées dans le cadre de l'aquarium d'Océanopolis.
- Fête de l'Internet 2002 : démonstration de l'application de télé-contrôle à distance de caméras motorisées dans le cadre de l'aquarium d'Océanopolis. Des présentations ont eu lieu dans trois écoles de la région brestoise auprès d'enfants de Classes Préparatoires, avec possibilité pour les enfants de poser des questions, en direct, au soigneur d'Océanopolis présent dans la manchotière.
- Fête de l'Internet 2002 : présentation et démonstration du réseau Internet aux membres de l'Université du Temps Libre de Brest. Les auditeurs ont pu ensuite "surfer" sur le réseau en étant accompagnés par des étudiants en informatique.
- Journées du Conseil Général du Finistère : conférence invitée sur les possibilités de contrôle à distance dans le cadre d'une journée d'information en direction des élus du département du Finistère.
- Salon InfoSup Morbihan 2000, 2001 et 2002 : promotion des études en informatique dispensées sur l'UBO (sur deux jours).

1.7 Résumé des activités de recherche

1.7.1 Contexte des activité de Recherche - EA2215 - Laboratoire LIMI

Mon activité de recherche s'est déroulée à l'Université de Bretagne Occidentale dans le laboratoire Langages et Interfaces pour Machines Intelligentes dirigé par le professeur Lionel Marcé. Ce laboratoire a dans un premier temps fait partie de la Jeune Equipe 2212, puis de l'Equipe

d'Accueil en informatique 2215. Cette dernière est composée actuellement, pour la partie UBO, d'environ 19 personnes.

A mon arrivée à Brest, le laboratoire LIMI ne comptait qu'un membre, le professeur Lionel Marcé. En 2002, il était composé d'un professeur et de huit maîtres de conférences. Il est maintenant intégré dans l'EA 2215 de l'UBO et a donné naissance à plusieurs actions de recherche dans différentes directions, robotique, téléproductique, vérification de circuits, ordonnancement temps-réel et bio-informatique avec comme point commun l'étude de méthodes et outils formels pour ces thèmes.

1.7.2 Thématiques de recherche

(Cette partie est développée dans les chapitres 2.1 et 2.2)

Au cours de ces dernières années, j'ai travaillé dans deux thématiques de recherche. La première (de 1991 à 1999) s'intéressait à la modélisation et à la validation de langages de l'automatisme, tel que le langage Grafcet, en utilisant des méthodes originaires du monde des langages synchrones. La deuxième (de 1999 à aujourd'hui) s'intéresse à la téléproductique, c'est-à-dire au contrôle à distance de systèmes mécaniques en utilisant comme support de communication un réseau sans qualité de service de type réseau Internet.

Ce changement de thématique opéré en 1999, correspond à la fois à la fin d'une action de recherche au niveau du laboratoire ainsi qu'à ma volonté de m'orienter vers des préoccupations différentes. Ma participation au projet téléproductique s'inscrit dans mon envie de travailler dans le domaine du contrôle de systèmes automatisés, de production ou non. De plus, l'utilisation d'un modèle de type Guide d'Etudes des Modes et Marche et d'Arrêt (GEMMA) pour la téléproductique, découlait naturellement de mes recherches précédentes. En outre, mon intérêt pour la validation de programmes et la vérification de propriétés s'est trouvé renforcé à travers des actions en direction de la validation de l'architecture logicielle proposée pour le contrôle à distance de machines.

Ces deux axes de recherche ont donné lieu à de nombreuses publications, dont la liste est dressée au chapitre 1.8.

1.7.3 Encadrement d'étudiants

L'encadrement d'étudiant est un moment privilégié, à la fois pour partager ses connaissances, mais également pour progresser dans sa thématique de recherche. L'UBO ne possédant pas de Diplôme d'Etudes Approfondies (DEA) en Informatique, il est alors très difficile d'encadrer des étudiants se destinant à des études doctorales en informatique. J'ai néanmoins essayé de participer au mieux des possibilités qui m'étaient offertes afin d'assurer le suivi et l'accompagnement d'étudiants. La création d'un Master Recherche à l'UBO, co-habilité avec l'Université de Rennes 1, devrait offrir de nouvelles facilités d'encadrement dès la rentrée 2004.

1.7.3.1 Etudiants en thèse

J'ai participé à l'encadrement de deux étudiants en thèse de doctorat :

– Dominique L'Her (pour 25%) : *Modélisation du Grafcet temporisé et vérification de propriétés temporelles.*

  Le sujet de M^{lle} Dominique l'Her portait sur la vérification du langage Grafcet à l'aide des automates temporisés [L'H97]. Il découlait directement de ma propre thèse qui dans son chapitre 6 propose des pistes pour la validation formelle de programmes Grafcet. L'encadrement principal a été assuré par le professeur Lionel Marcé et la thèse était financé par une bourse du Ministère de la Recherche.

  La thèse a été soutenue le 23 septembre 1997 devant un jury composé de : B. Arnaldi, N. Halbwachs, G. Vidal-Naquet, P. Le Guernic, L. Marcé et F. Prunet.

Résumé : "Le grafcet est un langage graphique dédié à la modélisation du comportement de la partie commande d'un système automatisé. Pour la plupart des applications modélisées, certains comportements ne doivent jamais être rencontrés car ils conduisent à des situations aberrantes voire dangereuses. Vérifier des propriétés sur des programmes grafcet est donc nécessaire. A cette fin, nous avons modélisé le grafcet dans des formalismes auxquels un outil de vérification est attaché.

Lorsque ce travail a débuté, le temps n'était pas pris en compte de manière quantitative. Pourtant au niveau du grafcet, il joue un rôle important à travers les temporisations : il est souvent utilisé pour synchroniser différentes parties opératives. Sa prise en compte doit permettre d'une part le calcul de contraintes temporelles, de temps de cycle d'autre part de vérifier des propriétés "classiques" (ne comportant pas de temps) sur des grafcets comportant des temporisations.

Tout d'abord nous avons essayé d'introduire le temps dans les modélisations qui avaient déjà été données pour le grafcet au sein de l'équipe : le langage SIGNAL, les systèmes de transitions. L'hypothèse suivante a alors été faite : un pas de cycle dans le cas de SIGNAL, le franchissement d'une transition dans celui des systèmes de transitions représente une unité de temps. Ceci nous a permis de vérifier des propriétes temporelles lorsque le grafcet ne comportait pas plus de deux temporisations et avec de petits délais. En effet cette modélisation du temps conduit à une croissance exponentielle du nombre d'états.

Aussi nous nous sommes tourné vers des formalismes qui dès leur définition prennent en compte le temps physique. Nous avons ainsi modélisé le grafcet grâce aux automates temporisés, exprimé les propriétés en logique TCTL et utilisé le model checker KRONOS. Grâce à cette méthode, nous avons pu vérifié des propriétés sur des applications de taille moyenne comme par exemple la cellule de production KORSO."

Mlle Dominique L'Her a été recrutée comme Maître de Conférences à l'Université de Bretagne Occidentale à l'issue de son doctorat.

– Pascal Ogor (pour 60%) : *Une architecture générique pour la supervision sûre à distance de machines de production avec Internet.*

Le sujet de M. Pascal Ogor portait sur la définition d'une architecture générique pour la supervision sûre à distance de machines de production en utilisant comme support de communication, un réseau non fiable tel que le réseau Internet [Ogo01]. Personnellement j'ai accompagné M. Pascal Ogor sur les aspects informatiques de son travail. Il a été également encadré par M. Jean Vareille (partie automatique/mécanique) et par le professeur Lionel Marcé. Cette thèse s'est déroulée en co-tutelle avec la FHOOW d'Emden en Allemagne et était finacée par une bourse régionale.

La thèse a été soutenue le 19 décembre 2001 devant un jury composé de : G. Burel, T. Divoux, G. Louis, L. Marcé, P. Le Parc, W. Mauersberger, E. Rutten et J. Vareille.

Résumé : "L'émergence d'Internet comme standard pour les applications et les protocoles dans le monde industriel nécessite une évolution des outils actuellement utilisés. Au contraire des solutions utilisant les réseaux dédiés, les communications sur Internet se caractérisent par l'absence de garantie de la qualité de service. Aussi ce travail poursuit l'objectif de proposer des outils permettant d'assurer la sûreté de fonctionnement des machines de production en tenant compte de l'incertitude sur la communication entre l'opérateur et le système automatisé.

Pour résoudre ce problème, notre travail se base sur l'implantation d'un comportement autonome guidant les réactions du système automatisé en cas de dysfonctionnements du réseau. Cette spécification est réalisée à l'aide d'un automate à états finis modélisant les modes de marches de l'application et intégrant une nouvelle dimension liée à la qualité de la communication : le GEMMA-Q. Cet outil est intégré dans une plate-forme logicielle "SATURNE" qui, utilisant les technologies *web*, fournit une architecture communicante

entre des opérateurs et une machine de production. Ces éléments sont par la suite testés sur différentes plates-formes matérielles."

M. Pascal Ogor travaille aujourd'hui à Stuttgart (Allemagne) pour la société Génericom.

- Depuis 1^{er} janvier 2004, j'encadre de M. Loic Plassart (pour 70%), ingénieur dans la société LivBag (société du groupe AutoLiv, leader mondial dans le domaine de la sécurité automobile) sur le thème : *Optimisation du pilotage d'une chaîne d'assemblage automatisée, par la mise en oeuvre d'une couche d'abstraction logicielle.*

  L'objectif de ce travail, dans un premier temps, est de modéliser les différents composants d'une chaîne automatisée de production (postes de travail, réseau de terrain, poste de commande et de supervision, environnement de traçabilité) afin de mieux comprendre le fonctionnement global du système et de pouvoir réduire au minimum les temps de cycle de production des pièces. Dans un deuxième temps, des propositions de solutions seront modélisées, validées puis intégrées dans la chaîne de production. Enfin, une méthodologie de développement sera définie afin d'appliquer les résultats obtenus dans ce travail à la conception générale des chaînes de production de la société LivBag.

1.7.3.2 Etudiant en DEA

J'ai encadré, en 2001-2002, M. J. Sadouki, du DEA Sciences de la Matière, de l'Information et du Vivant (SMIV) de l'UBO sur le thème : *modélisation et vérification d'une architecture logicielle pour la téléproductique.* Il s'agissait de valider l'architecture logicielle que nous proposons dans le cadre de l'e-productique avec l'outil Spin/Promela. Ce travail a commencé par l'étude de l'architecture logicielle puis de différentes méthodes permettant la validation de propriétés. L'outil Spin a alors été choisi et une partie de l'architecture a été décrite en utilisant le langage Promela. Quelques propriétés ont alors pu être vérifiées.

1.7.3.3 Etudiants de DESS

Depuis 8 ans, je suis régulièrement 2 à 3 étudiants de DESS IAP de l'UBO dans le cadre de leur projet de fin d'études. Durant leur deux mois de présence, ces étudiants participent aux développements technologiques nécessaires à l'activité de recherche du laboratoire.

Voici quelques sujets proposés au cours de ces 8 années.

- Réalisation d'un traducteur du langage à relais en langage Signal - 1992/1993 - J. Bouloc : ce travail a permis de concevoir un traducteur de programmes en langage à relais vers un ensemble d'équations synchrones Signal en respectant une sémantique d'exécution ligne par ligne.

- Preuve de grafcets à l'aide des équations 3 états de Signal - 1992/1993 - B. Quéguineur : le but de ce projet était de construire un environnement permettant la génération, à partir d'un programme Grafcet traduit en Signal, des équations dynamiques polynomiales correspondantes et surtout de propriétés à vérifier. Ces propriétés étaient exprimées à haut niveau en terme "grafcet" puis traduites sous formes de formules vérifiables par l'outil Sigali.

- Réalisation d'un traducteur Grafcetvers Lustre - 1992/1993 - D. Gourtay : ce projet a permis de réaliser un compilateur de programmes Grafcet en Lustre. La partie analyse de programme source réalisée pour le traducteur Grafcet-Signal a été reprise intégralement et complêtée par un module de génération des équations Lustre.

- Portage sous Windows NT4.0 de la chaîne de développement Grafcet du LIMI - 1996/1997 - A. Cordier : la chaîne de développement Grafcet a été conçue dans un environnement Unix. Elle comporte un éditeur de programmes, un compilateur Grafcet-Signal, un compilateur Grafcet-Automates à Etats. L'objet de ce projet était de porter l'ensemble des composants dans un environnement Windows NT4.0 pour une utilisation et une diffusion plus large.

- Un prototype d'éditeur de Grafcet en Java - 1998-1999 - D. Gajan : l'objectif de ce projet était de récrire notre éditeur de programmes Grafcet en Java afin d'assurer une meilleure portabilité et pérennité. La version précédente ayant été écrit dans l'environnement Tcl/Tk.

- Diffusion interactive d'images video - 2000/2001 - G. Cotten : nous avons expérimenté à travers ce projet l'environnement Java Media Framework (JMF) pour la diffusion d'images sur Internet afin de remplacer l'outil Webvideo que nous utilisions. De bons résultats ont été obtenus au niveau du laboratoire et nous permettaient de régler dynamiquement le flux vidéo. Malheureusement, les tests à plus grande échelle n'ont pas été concluants en raison de l'usage du protocole RTP.

- Statistiques réseau par étude de fichiers de logs - 2001/2002 - S. Troadec : notre environnement de téléproductique produit des fichiers de données permettant de caractériser chaque connexion réalisée. Ce travail a porté d'une part sur l'extraction des informations dans les fichiers de traces et d'autre part en la réalisation de statistiques simples (nombre de connexions, durée moyenne, origine, qualité réseau moyenne etc.).

- Nouvelles technologies réseau pour le Projet Saturne - 2002/2003 - N. Ceotto : ce projet a essayé de trouver une solution au transport des informations de notre plateforme d'e-productique au travers des pare-feu (firewall) de l'internet. Les solutions n'étaient déployables que sur des versions de plate-forme java postérieures à celle que nous utilisions à cette date. Elles devraient pouvoir être maintenant utilisables (passage du JDK 1.1.8 au JDK 1.4 réalisé au dernier trimestre 2003).

1.7.3.4 Etudiants de Maîtrise

Depuis 5 ans, les étudiants de Maîtrise informatique doivent effectuer un Travail d'Etudes et de Recherche (TER) au cours de leur scolarité. Chaque année, j'encadre 2 à 3 étudiants. Contrairement aux projets de DESS où la finalité est technologique, les sujets proposés aux étudiants de maîtrise se basent d'abord sur une étude bibliographique puis, généralement, sur l'exploration d'un domaine donné à travers l'utilisation d'une méthode ou d'un logiciel spécifique.

Voici quelques sujets proposés au cours de ces 5 années.

- Construction d'un compilateur Grafcet vers PL7.2 - 1997/1998 - P. Calvez : il s'agissait, à partir du format textuel du Grafcet défini au cours de mon travail de thèse, de générer l'équivalent pour les automates de type Télémécanique. Un prototype de compilateur a été construit permettant de générer certains grafcets.

- Spin : outil de validation de systèmes répartis - 1999/2000 - M. Prigent : ce travail avait pour objectif d'étudier l'outil de validation SPIN : installation, prise en main, évaluation sur de petits exemples.

- Etude du logiciel de vérification Uppaal - 2001/2002 - N. Ceotto : le but de ce TER était de prendre en main l'outil Uppaal et de l'appliquer à la validation de l'architecture logicielle que nous proposons pour la téléproductique. Seule une modélisation à haut niveau a pu être réalisée, mais elle a montré les potentialités de cet outil.

- Capteurs réseaux dans le cadre de la téléproductique - 2001/2002 - S. Debatisse : dans nos applications d'e-productique, nous mesurons en permanence la qualité du réseau à l'aide d'un système cyclique : envoie d'une requète, attente d'une réponse, mesure du délai, pause de X ms. L'objectif de ce travail était d'une part d'étudier l'impact de la valeur de X et d'autre part d'étudier l'intérêt d'un envoi régulier de requètes (en non plus après retour et pause). Les résultats obtenus n'ont pas montré de différences importantes, en terme d'appréciation et donc d'information pour l'utilisateur.

1.7.4 Logiciels

Trois environnements logiciels ont été créés à partir de mes travaux de recherche.

1.7.4.1 Chaîne de développement orientée Grafcet

Elle est composée d'un éditeur de programmes Grafcet, d'un compilateur Grafcet vers le langage synchrone Signal et d'un simulateur, cette chaîne a permis de valider les travaux menés sur la sémantique du Grafcet. Enrichie par différents travaux elle permet également d'effectuer des validations sur les programmes en proposant des traductions soit sous forme de systèmes de transitions, soit sous forme d'automates temporisés. Cette chaîne de développement a été utilisée pour la simulation et la validation de la cellule Korso. Elle est également utilisée dans le cadre de l'option Systèmes Réactifs et Intelligents en Maîtrise Informatique. Elle a été principalement développée en langage C pour la partie calculs, et en TCL/TK pour la partie graphique.

1.7.4.2 Projet SATURNE (Software Architecture for Teleoperation over an UnpRedictable NEtwork)

Ce logiciel, composé d'une partie serveur, d'une partie client et d'un ensemble de modules à chargement dynamique permet la prise de contrôle à distance de systèmes matériels en utilisant comme support de communication le réseau Internet. Ce logiciel est utilisé pour les manipulations du laboratoire (bras manipulateur Ericc, machine de prototypage rapide ISEL, robot mobile Khepera, caméra motorisée EVI-D31). Il a été développé en langage Java.

1.7.4.3 LogAnalyser

Lors de l'utilisation de l'environnement Saturne, des traces de connexions sont enregistrées en permanence : date, durée, origine, commandes effectuées, qualité du réseau. Pour l'application déployée à Océanopolis, ces traces représentent un volume d'environ 850 Mo (près de 18 000 connexions entre août 2001 et février 2004). Afin de faciliter l'analyse de ces fichiers, l'outil LogAnalyser permet d'extraire les informations importantes et de les stocker dans une base de données de type SQL. Ces données sont ensuite traitées à l'aide de requètes incluses dans des scripts PhP.

1.7.5 Contrats

Les travaux sur la téléproductique ont été financés à travers un Programme de Recherche d'Intérêt Régional (PRIR) qui a permis l'acquisition du fraiseuse de prototypage rapide. De plus, les travaux de thèse de M. Pascal Ogor ont été financés par une bourse régionale, ceux de Mlle Dominique L'Her l'étant à travers une bourse ministérielle.

L'environnement SATURNE a fait l'objet d'un soutien de la part du Conseil Général du Finistère pour son déploiement dans deux bassins de l'aquarium d'Océanopolis (Brest). En particulier, une caméra motorisée aérienne (Pan/Tilt/Zoom) a été installée dans la manchotière d'Océanopolis, et une autre sous-marine dans le bassin des requins puis dans l'aquarium tropical (www.oceanopolis.com - en fonctionnement depuis le 16 août 2001). J'ai été en charge de ce projet à 90%. A cette occasion, un transfert de technologie a été réalisé en direction de la société Irvi-Progénéris.

Le travail de thèse de M. Loic Plassart fait également l'objet d'un contrat de 3 années avec la société LivBag (01/01/2004 au 31/12/2007).

1.7.6 Participation à des groupes de travail

J'ai pris part à différents groupes de travail nationaux en y réalisant des présentations scientifiques et en prenant part aux débats :

– Le groupe Langages Synchrones animé par Mr Albert Benveniste.

- Le groupe Grafcet de l'AFCET, animé par Mr Jean-Jacques Lesage, qui s'est ensuite transformé en groupe COSED (Commande Opérationnelle des Systèmes à Evénements Discrets) du club EEA.
- Le groupe Systèmes de Communications Industrielles animé par Mr Thierry Divoux, qui a participé de manière importante à l'Action Spécifique 01 du CNRS intitulée "Réseaux de Communications et Automatique". Pour ce groupe, j'ai entre autre participé à l'occasion d'un séminaire sur Brest.

Actuellement, j'ai participé au RTP 55 intitulé "Systèmes de Contrôle-Commande et Réseaux de Communication" animé par Mr Thierry Divoux ainsi qu'à l'AS 198 "Impact des NTIC en automatisation", animé par Mr Jean-François Petin, du RTP 47 "STIC et production coopérative médiatisée".

1.7.7 Activités diverses pour le laboratoire de recherche

1.7.7.1 Administration système

Depuis mon arrivée au sein du projet LIMI, en 1991, et jusqu'à début 2003, j'ai été en charge de l'administration des stations de travail (type Sun, sous Unix) pour l'ensemble de l'équipe. Depuis l'installation de la première machine jusqu'à la configuration actuelle (8 machines en réseaux), cette responsabilité m'a permis de suivre l'évolution des technologies et d'appréhender le métier, ingrat, d'ingénieur système. J'ai toujours essayé d'avoir comme objectif, la disponibilité maximale des ordinateurs associée à la fiabilité de fonctionnement et à la pérennisation des équipements matériels et logiciels.

1.7.7.2 Gestion financière

Je suis responsable depuis plusieurs années de la gestion financière du laboratoire LIMI et maintenant de l'EA 2215 - UBO. Cette tâche, très administrative, m'a permis de mieux comprendre le fonctionnement financier d'une université telle que l'UBO.

1.7.7.3 Organisation de congrès

J'ai participé à l'organisation du premier congrès Modélisation des Systèmes Réactifs qui s'est tenu à Brest en mars 1996.

J'ai également participé à l'organisation du séminaire "Synchronous Languages" qui s'est déroulé à Roscoff en 1997.

1.8 Publications

Au cours de ces années de recherche, j'ai publié mes travaux, soit comme auteur principal, soit comme co-auteur à 44 reprises dont 8 fois dans la cadre de revues internationales ou nationales.

1.8.1 Revues Internationales (5)

[1] L. Marcé et P. Le Parc. Defining the semantics of languages for programmable controllers with synchronous processes. *Control Engineering Practice*, 1(1):79–84, 1993.

[2] P. Le Parc, B. Queguineur et L. Marcé. Two proof methods for the grafcet language. *Annual Review in Automatic Programming*, 18:61–65, 1994.

[3] D. L'her, P. Le Parc et L. Marcé. Proving sequential function chart programs using automata. Dans *Workshop on implementing automata*, numéro 1660, page 149 - 163. LNCS, 1999.

[4] P. Le Parc, D. L'her, J.L. Scharbarg et L. Marcé. Grafcet revisited with a synchronous dataflow language. *Revue IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Human*, 29(4), Aout 1999.

[5] D. L'Her, P. Le Parc et L. Marcé. Proving sequential function chart programs using automata. *Theoretical Computer Science*, (267):141–155, Septembre 2001.

1.8.2 Revues Françaises (2 + 1 en révision)

[1] P. Le Parc, D. L'her, J.L. Scharbarg et L. Marcé. Le Grafcet revisité à l'aide d'un langage synchrone flot de données. *Technique et Science Informatiques*, 98(1), Janvier 1998.

[2] D. L'her, P. Le Parc et L. Marcé. Modélisation et vérification du Grafcet temporisé. *JESA : Journal Européen des Systèmes Automatisés*, 33(5-6):651 683, Juillet 1999.

[3] P. Le Parc, J. Vareille et L. Marcé. E-productique ou contrôle et supervision distante de systèmes mécaniques sur internet. *JESA : Journal Européen des Systèmes Automatisés*, 2004. en révision.

1.8.3 Conferences internationales avec comité de lecture et actes (16)

[1] L. Marcé et P. Le Parc. Defining the semantics of languages for programmable controllers synchronous processes. Dans *Proceedings of the* 18^{th} *IFAC/IFIP workshop on Real-Time Programming*, Bruges - Belgique, Juin 1992.

[2] P. Le Parc et L. Marcé. Synchronous definition of grafcet with signal. Dans *International Conference on Systems, Man and Cybernetics*, pages 675–680, Le Touquet France, Octobre 1993. IEEE-SMC.

[3] P. Le Parc, B. Queguineur et L. Marcé. Two proof methods for the Grafcet language. Dans *Workshop on real-time programming*, Lac de Constance, Suisse, Juin 1994. IFAC-IFIP.

[4] M. Arnoux, J.L. Scharbarg, P. Le Parc et L. Marcé. Simulation du fonctionnement d'une usine à béton. Dans *Symp. Automatisation de processus mixtes: les systèmes dynamiques hybrides*, Bruxelles, Belgique, Novembre 1994. SEE-IBRA.

[5] D. L'her, P. Le Parc et L. Marcé. Modeling and proving grafcets with transitions systems. Dans *2nd AMAST workshop on real-time systems*, Bordeaux, Juin 1995. AMAST.

[6] L. Marcé, D. L'her et P. Le Parc. Vers l'amélioration de la qualité des modèles des systèmes de Dans *Congrès international de Génie Industriel*, Montreal, Canada, Octobre 1995.

[7] J.L. Fleureau, P. Le Parc et L. Marcé. Signal: a language for low level implementation. Dans *Southeastcon'95:visualizing the future*, Raleigh NorthCarolina, USA, Mars 1995. IEEE.

[8] L. Marcé, D. L'her et P. Le Parc. Modelling and verification of temporized grafcets. Dans *CESA*, Lille, Juin 1996. IEEE.

[9] D. L'her, J.L. Scharbarg, P. Le Parc, J. Vareille et L. Marcé. Specification and verification of the korso production cell. Dans *INCOM98*, Nancy, France, Juin 1998. IFAC, IEEE.

[10] P. Ogor, P. Le Parc, J. Vareille et Lionel Marcé. Remote control of robotics systems on the web. Dans *NETTIES 200*, Oulu, Finlande, 23-25 mars 2000.

[11] P. Ogor, P. Le Parc, J. Vareille et L. Marcé. Controlling remote systems on the web. Dans *Proceedings of the ACIS* 1^{st} *International conference on Software Engineering Applied to Networking and Parallel/Distributed Computing*, Reims, France, 18-21 mai 2000.

[12] P. Ogor, P. Le Parc, J. Vareille et L. Marcé. Control a robot on internet. Dans 6^{th} *IFAC "Low Cost on Automation" Symposium*, Berlin, Allemagne, 8-9 octobre 2001.

[13] J. Vareille P. Le Parc, P. Ogor et L. Marcé. Web based remote control of mechanical systems. Dans *International Conference on Software Telecommunications and Computer Networks (IEEE - SoftCom2002)*, Split, Croatie, Octobre 2002.

[14] J. Vareille, P. Le Parc et L. Marcé. Le contrôle réactif par internet et son application aux machines de production. Dans *CPI'2003*, Meknes, Maroc, Octobre 2003.

[15] P. Le Parc, J. Vareille et L. Marcé. Web remote control of machine-tools the whole world within less than one half-second. Dans *ISR'2004 : International Symposium on Robotics*, Paris, France, Mars 2004.

[16] P. Le Parc, J. Vareille et L. Marcé. Web based remote control of mechanical systems. Dans *1st IFAC Symposium on Telematics Applications in Automation and Robotics*, Helsinki, Finlande, Juin 2004.

1.8.4 Conferences nationales avec comité de lecture et actes (11)

[1] L. Marcé et P. Le Parc. Modélisation de la sémantique du GRAFCET à l'aide de processus synchrones. Dans *Actes du congrès GRAFCET'92*, Mars 92.

[2] P. Le Parc et L. Marcé. Définition synchrone du Grafcet à l'aide du langage Signal. Dans *Real-time systems 94*, Paris, Janvier 1994. RTS.

[3] D. L'her P. Le Parc L. Marcé. Modélisation et vérification du grafcet. Dans *MSR'96, Modélisation des Systèmes Réactifs*, Brest, Mars 1996.

[4] D. L'her, P. Le Parc et L. Marcé. Proving sequential function chart programs using automata. Dans *Workshop on implementing automata*, Rouen, France, Sept 1998. WIA.

[5] D. L'her P. Le Parc, J.L. Scharbarg et L. Marcé. Modélisation et vérification de la cellule korso par une boite Dans *ADPM'98*, Reims, France, Mars 1998. AFCET, SEE.

[6] P. Le Parc, R. Querrec, P. Chevaillier, J. Tisseau et L. Marcé. Un environnement de développement pour la conception de systèmes automatisés de production. Dans *2ème congrès sur la Modélisation des Systèmes Réactifs, MSR'99*, Cachan, France, Mars 1999.

[7] D. L'her, P. Le Parc et L. Marcé. Une étude des relations Grafcet langages synchrones flots de données. Dans *Journée SEE sur les nouvelles percées dans les langages pour l'automatique*, Amiens, Novembre 1999.

[8] P. Le Parc P. Ogor, J. Vareille et L. Marcé. Vers les machines outils et les robots communicants. Dans *Séminaire Objets Communicants SOC'2001 - SEE/RNRT*, Grenoble, France, Octobre 2001.

[9] J. Vareille, P. Le Parc et L. Marcé. Vers les machines de production communicantes. Dans *Conférence ISA-SEE "le contrôle de la production dans l'entreprise réactive de l'ère Internet"*, Nice, France, Mars 2002.

[10] J. Vareille, P. Le Parc et L. Marcé. Démonstration à distance des travaux de recherche du limi dans le Dans *JJCR'16*, Lyon, France, Septembre 2002.

[11] J. Vareille, P. Le Parc, L. Marcé, S. Barré, A. Mamoune et J. Le Guen. Contrôle actif de machines de production par internet. Dans *CFM'2003*, Nice, France, Septembre 2003.

1.8.5 Présentation dans des groupes de travail nationaux (6)

[1] P. Le Parc et L. Marcé. Modélisation des opérateurs de temporisation en Signal. Dans *Groupe Grafcet de l'AFCET*, Paris, France, Avril 1993.

[2] P. Le Parc et L. Marcé. Outils du laboratoire LIMI pour le Grafcet. Dans *Groupe Grafcet de l'AFCET*, Paris, France, Décembre 1993.

[3] D. L'Her, P. Le Parc et L. Marcé. Une méthode pour la vérification du Grafcet. Dans *Groupe Grafcet de l'AFCET*, Paris, France, Juillet 1997.

[5] P. Le Parc, J. Vareille et L. Marcé. Démonstration des plateformes de télé-productique du laboratoire LIMI. Dans *Groupe SCI - Systèmes de Communications Industrielles* , Toulouse, France, Juillet 2000.

[6] P. Le Parc, J. Vareille et L. Marcé. Gestions des modes de marche et d'arrêt d'un système téléopéré. Dans *AS01 - Réseaux de communication et Automatique*, Paris, France, Mars 2002.

1.8.6 Manuscript de thèse (1)

[1] P. Le Parc. Apport de la méthodologie synchrone pour la définition et l'utilisation du langage Grafcet. Thèse de Doctorat, Université de Rennes 1, Janvier 1994.

1.8.7 Divers (2)

[1] S. Thiébaux, P. Le Parc et L. Kervella. La programmation temps-réel - l'approche synchrone, Janvier 91. Etude bibliographique - DEA Rennes.

[2] P Le Parc. Etude du langage GRAFCET et de ses relations avec le langage SIGNAL, Septembre 91. Rapport de stage - DEA Informatique Rennes.

Chapitre 2

Activités de recherche

Mon activité de recherche se situe dans le domaine de l'informatique, une informatique orientée vers le contrôle et la commande de systèmes industriels ou non. Cette thématique regroupe, au sens large, la productique, la robotique, les systèmes temps réels, les systèmes réactifs, les automatismes industriels, soit en résumé, les disciplines s'intéressant aux systèmes en mouvement ou effectuant des mouvements sous le contrôle d'un environnement informatique.

Le travail dans un tel domaine requiert l'utilisation de diverses compétences. Certaines sont typiquement dans le champ disciplinaire de l'informatique comme la modélisation de systèmes, la validation de programmes, le génie logiciel ou les réseaux de communication et d'autres qui n'y sont pas telles que la mécanique, l'électronique, l'automatique. Cette mixité est très intéressante et sans devenir un expert des dernières matières précédement citées, elle permet un enrichissement intellectuel permanent et la découverte de méthodes, modèles, outils et techniques autres que ceux habituellement connus et maîtrisés par les informaticiens.

Dès le début de mon travail de DEA, puis au cours de ma thèse et ensuite, j'ai participé à des groupes de recherche se situant ainsi à la frontière de l'informatique et de l'automatique :

- Le "Centre Coopératif de Génie Automatique" regroupant les sociétes April, Cégelec, EMR, Sygral et 3IP ainsi que deux établissements d'enseignement supérieur (ISMCM et UBO/LIMI) et dont l'un des objectifs était de définir un langage, informatique, permettant l'échange d'information entre automates programmables industriels.

- Le groupe de travail Grafcet de l'Afcet, depuis transformé depuis en groupe COSED (Commande Opérationnelle des Systèmes à Evénements Discrets), dont la problématique générale était d'une part la promotion de la méthode Grafcet et d'autre part l'enrichissement de ce formalisme en particulier au niveau de la validation de programme. De nombreuses méthodes, issues des travaux de la communauté informatique ont alors été largement introduites par nous-mêmes et utilisées.

- Le groupe C^2A Synchrone (C^2A signifiant d'ailleurs Collaboration CAO Automatique), qui a permis de fédérer les travaux autour des trois langages synchrones que sont Esterel, Lustre et Signal afin de les renforcer et qu'ils aient aujourd'hui une place incontrournable dès que l'on parle de programmation sûre de systèmes réactifs.

  On peut noter que la première phrase de la première publication de ce groupe [Ben89], évoque les problèmes "que rencontrent les industriels de l'automatique". Il s'agissait là, pour des acteurs fortement impliqués dans la communauté informatique, de proposer et d'inventer des solutions permettant d'améliorer les processus de conception de logiciel pour ce domaine.

Ces contacts ont permis de mener à bien ma première activité de recherche, décrite ci-après dans la section 2.1, qui a cherché à proposer des solutions pour la modélisation et la vérification

des langages de l'automatisme (et en particulier du Grafcet), en utilisant les techniques synchrones et les avancées en matière de validation de programmes.

Ma deuxième activité de recherche (section 2.2) se situe elle aussi à l'interface de plusieurs disciplines : outre les disciplines citées précédemment, le contrôle à distance de systèmes mécaniques sur des réseaux sans qualité de service fait également intervenir des compétences dans le domaine des réseaux de communication. Ce travail a été engagé à travers plusieurs groupes de réflexion nationaux :

- Le groupe Téléproductique Bretagne, qui a essayé de présenter différentes expériences de contrôle distant de machines de production industrielles présentes dans plusieurs établissements bretons : Insa de Rennes, Supélec Rennes et l'Université de Brest.
- Le groupe Systèmes de Communication Industrielle (SCI) qui s'intéressait à la fois aux nouvelles technologies de communication, filaires ou non, pour les milieux industriels et aux nouvelles applications induites, afin de mieux en définir les enjeux, les contours et les potentialités. Ce groupe s'est ensuite intégré à l'Action Spécifique 01 (AS01).
- L'Action Spécifique 01 du département STIC du CNRS était intitulée "Auto-Telecom : réseaux de communication et automatique". Elle proposait deux grandes thématiques, l'une sur les aspects de qualité de service dans les réseaux et l'autre sur les applications distribuées utilisant ces réseaux de communication. Mon travail s'intégrait plus particulièrement dans cette deuxième partie, tout en regardant de manière attentive, les résultats produits par la première.

Aujourd'hui de nombreuses équipes au niveau national travaillent dans ce domaine et sont regroupées en particulier à travers deux Réseaux Thématiques Pluridisciplinaires du département STIC du CNRS, le RTP 55 "Systèmes commandés en réseaux" et le RTP 47 "STIC et production ccopérative médiatisée" ainsi que dans la section Automatique du Club EEA qui a d'ailleurs organisé en mars 2004 des journées d'études "Automatique et Informatique". Citons également de RNRT "Métrologie pour l'Internet" et l'AS 48 "Métrologie des réseaux de l'Internet".

Au niveau européen, la lecture des directions prioritaires du programme "Information Society Technologies" et de la devise générale "*IST vision : anywhere anytime natural access to IST services for all*", montrent tout l'intérêt des travaux que nous menons qui visent à utiliser les technologies actuelles pour permettre l'utilisation de systèmes dynamiques à distance. Bien évidement, des travaux similaires sont menés de part le monde entier.

La suite de ce chapitre présente donc tout d'abord les recherches menées autour de la modélisation des langages de l'automatique puis celles autour de la commande distante de systèmes sur des réseaux sans qualité de service.

2.1 Modélisation et vérification du Grafcet

Cette première thématique a pris naissance suite à une étude [LP91] proposée par un ensemble de constructeurs de solutions d'automatismes [1] qui souhaitaient pouvoir échanger des programmes d'un Automate Programmable Industriel (API) à un autre. Au niveau syntaxique, la définition d'un langage pivot commun et la construction de traducteurs peut permettre de tels échanges. Par contre au niveau sémantique, il est nécessaire que les différents API aient la même interprétation d'un programme donné afin de pouvoir assurer le même comportement lors d'une exécution.

Un exemple simple est celui du langage à relais. Ce langage, issu des relais électromagnétiques utilisés dans l'industrie manufacturière avant l'avènement des API, est un des langages métiers de l'automatisme. Il permet la spécification et la réalisation d'équations logiques booléennes reliant des capteurs (relais schématisés par des barres verticales) et des actionneurs (bobines schématisées par des parenthèses).

1. regroupés dans le CCGA

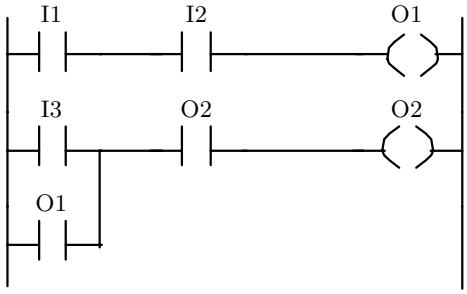

FIG. 2.1 - *Exemple de programme décrit en langage à relais*

Sur la figure 2.1 est présenté un exemple très simple de programme écrit en langage à relais. Néanmoins, deux interprétations de ce programme sont possibles.

L'interprétation dite horizontale (ou ligne), qui cherche à calculer chaque sortie en évaluant les lignes de manière séquentielle. On peut alors traduire le schéma par ces deux équations :

$$\begin{cases} O1_t & \leftarrow & I1_{t-1} \; and \; I2_{t-1} \\ O2_t & \leftarrow & (I3_{t-1} \; or \; O1_t) \; and \; O2_{t-1} \end{cases}$$

Aux valeurs des entrées $I1$, $I2$ ainsi qu'aux valeurs des sorties $O1$ et $O2$ sont ajoutés des indices temporels qui permettent de mieux comprendre comment se déroule l'évaluation des équations. En effet, le fonctionnement de base d'un API est cyclique : acquisition des entrées, traitement, émission des nouvelles valeurs. Dans les équations, l'indice $t-1$ correspond aux valeurs des entrées lors de l'acquisition et l'indice t aux valeurs des sorties qui vont être émises. On remarquera que dans la deuxième équation, deux valeurs d'entrée sont utilisées en partie droite. A $O1$ est affecté l'indice t qui correspond donc la valeur d'$O1$ calculée dans la première équation. Par contre, à $O2$ est associé l'indice $t-1$: on utilise la valeur précédente pour calculer la nouvelle valeur.

Dans le cadre de la deuxième interprétation du langage à relais, interprétation dite verticale, l'évaluation ne se fait plus ligne par ligne, mais zone après zone. Sur la figure 2.1, la partie gauche, composée uniquement de relais correspond à la zone de tests et la partie droite, composée uniquement de bobines, à la zone d'action. On peut donc traduire ainsi le schéma :

$$\begin{cases} O1_t & \leftarrow & I1_{t-1} \; and \; I2_{t-1} \\ O2_t & \leftarrow & (I3_{t-1} \; or \; O1_{t-1}) \; and \; O2_{t-1} \end{cases}$$

Dans ce cas, pour la deuxième équation, la référence temporelle prise pour $O1$ est $t-1$.

Pratiquement, si l'on suppose que les entrées $O1$ et $O2$ avaient une valeur fausse à l'instant $t-1$ et que les entrées $I1$, $I2$, $I3$ sont fausses à l'instant t, on obtient une valeur fausse pour $O2$ à l'instant t pour la première interprétation et vraie pour la deuxième ! On voit donc bien tout l'importance de ne pas seulement échanger les schémas et de ne pas rester au niveau syntaxique. Il est nécessaire de connaître et de pouvoir exprimer de manière claire les différentes interprétations d'un langage donné.

Le travail mené a justement porté sur ce point. Il ne s'agissait pas pour nous de choisir entre telle ou telle interprétation, mais de proposer des méthodes permettant de les spécifier finement. Il nous fallait donc un formalisme basé sur une sémantique rigoureuse et permettant d'exprimer des notions temporelles. Les formalismes synchrones, et en particulier Signal, répondaient bien à ces contraintes.

2.1.1 Langages synchrones

2.1.1.1 Présentation et hypothèses synchrones

Les formalismes synchrones [BB91] sont apparus en France dans les années 80. Ils sont principalement dus aux travaux de 3 groupes de recherche : Irisa/Inria Rennes pour Signal [LGG90]

sous la conduite de P. Le Guernic, Imag Grenoble pour Lustre [HCRP91] sous la conduite de N. Halbwachs et P. Caspi et Inria Sophia antipolis pour Esterel [BDS91] sous la conduite de G. Berry.

Les concepteurs de ces langages ont souhaité proposer des réponses à la problématique des systèmes réactifs, c'est-à-dire des systèmes dont l'évolution est assujettie à leur environnement. On peut citer par exemple les systèmes de contrôle-commande dans le transport aérien, les systèmes temps réel etc..

Deux hypothèses ont été posées :

- Les concepteurs des langages synchrones ont considéré que l'exécution des tâches se faisait de façon instantanée. Le temps physique est remplacé par un temps logique. Bien sûr, cette hypothèse nécessite une machine infiniment rapide donc purement imaginaire. Mais, s'il est possible de montrer que l'exécution d'une tâche se fait en un temps borné, inférieur aux contraintes temporelles souhaitées, cette hypothèse devient tout à fait réaliste.

- La deuxième hypothèse retenue a été de considérer la simultanéité (synchronisme) des événements d'entrée. Dans la plupart des autres formalismes existants, l'environnement et le traitement des données se fait se manière asynchrone, ce qui entraîne un ordonnancement des variables source d'incertitude. Dans l'approche synchrone, ces problèmes sont rejetés au niveau d'un moniteur asynchrone/synchrone, qui permet à l'utilisateur de se concentrer uniquement sur les aspects importants de son application.

Autour des langages synchrones un ensemble d'outils a été créé allant d'environnements de développement à des générateurs de code vers différentes machines cibles, tout en passant par des outils permettant la vérification de programmes [BCE+02]. L'ensemble des travaux s'est déroulé dans un cadre scientifique ayant pour objectif la conception de chaînes de développement complètes, de la spécification à l'exécution, reposant sur des principes sémantiques forts.

Parmi les trois formalismes cités précédemment, nous avons choisi d'utiliser Signal. Comparativement à Esterel, il offre une approche flots de données similaire à ce que l'on peut trouver dans le cadre des langages de Lustre, il possède, grâce à son mécanisme d'horloge plus complexe, un pouvoir d'expression un peu plus riche. Néanmoins, nous avons mené des travaux de modélisation du Grafcet avec Lustre et Esterel, mais ils ont été moins poussés et moins valorisés.

2.1.1.2 Le langage Signal

Le noyau d'instructions de Signal est composé de quatre opérateurs de base. Les deux premiers sont des opérateurs monochrones, c'est-à-dire qu'ils ne se réfèrent qu'à un seul index temporel ; les deux suivants sont qualifiés de polychrones.

- Les opérateurs fonctionnels : ce sont les opérateurs classiques arithmétiques ou booléens : *c := f(a*$_1$*, ..., a*$_n$*)*. Ils imposent le synchronisme des signaux utilisés à un instant logique donné t.

$$\forall t > 0, \quad c_t = f(a_{1\,t}, ..., a_{n\,t})$$

- Le retard : noté $, il permet de retarder un signal d'un instant logique.

  b := a$1 init i signifie que le signal b prend à l'instant t, la valeur que possédait a la dernière fois que celui-ci était présent (instant $t-1$). a et b sont des signaux synchrones.

$$\forall t > 1, \quad \begin{cases} b_t & = a_{t-1} \\ b_0 & = i \end{cases}$$

  La valeur i correspond à la valeur initiale du signal retardé b.

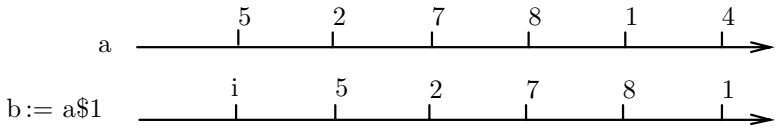

FIG. 2.2 - *Chronogramme temporel du retard*

– Le filtre : il sert à réaliser le sous-échantillonnage d'un signal en fonction d'un signal booléen. Sa syntaxe est la suivante : $c := a$ *when* b, et signifie que le signal c sera présent quand le signal a est présent et que le booléen b possède une valeur vraie. Aucune relation d'horloge n'est imposée entre a et b.

$$\forall t > 0, \quad c_t = \begin{cases} a_t & si\ a_t \neq \perp\ et\ b_t = vrai \\ \perp & sinon \end{cases}$$

Le symbole $\perp$ correspond à l'absence du signal.

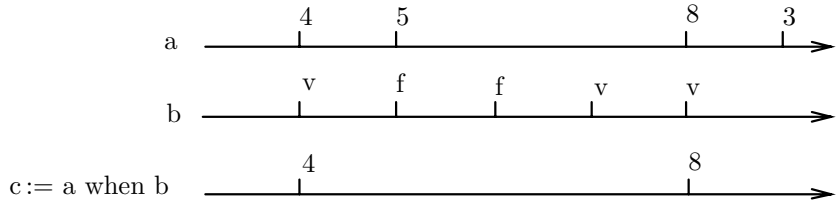

FIG. 2.3 - *Chronogramme temporel du filtre*

– La fusion : noté $c := a$ *default* b, elle a la sémantique suivante : c prend la valeur de a si a est présent, sinon, il prend la valeur de b si b est présent. C'est donc une fonction de sur-échantillonnage qui est proposée par cet opérateur.

$$\forall t > 0, \quad c_t = \begin{cases} a_t & si\ a_t \neq \perp \\ b_t & sinon \end{cases}$$

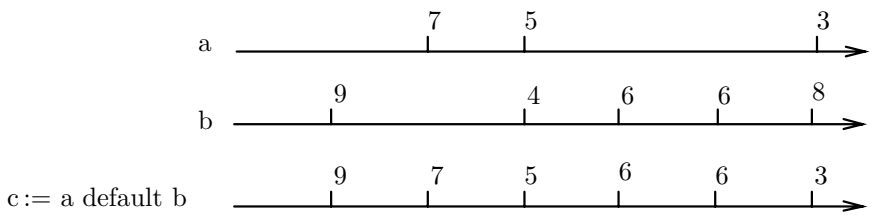

FIG. 2.4 - *Chronogramme temporel de la fusion*

Un processus Signal résulte donc de la composition parallèle (notée |) d'équations écrites à partir des opérateurs définis ci-dessus.

De plus, des macro-opérateurs ont été construits, comme par exemple, l'opérateur *synchro* qui permet de spécifier les relations temporelles existant entre différents signaux. Il sert donc plus à exprimer des contraintes qu'a définir des instructions.

2.1.2 Le Grafcet : un modèle et un langage

Le Grafcet [AFC77] (pour GRAphe Fonctionnel de Commande Etapes/Transitions) a été créé dans le début des années 70 par le groupe Systèmes Logiques de l'Association Française de Cybernétique Economique et Technique (Afcet) qui cherchait alors un formalisme permettant la spécification du cahier des charges d'un automatisme. En effet, plusieurs méthodes étaient utilisées dans le monde industriel (graphe d'états, réseaux de Petri, organigramme, etc.) à cet effet, mais aucune n'était normalisée ce qui rendait l'échange d'informations entre partenaires d'un même projet difficile.

Le succès du Grafcet (enseignement dès le début des années 80, normalisation nationale en 1982 et internationale en 1987) est du en grande partie à l'utilisation d'une représentation graphique simple associée à un pouvoir d'expression important. En effet, il est possible d'exprimer les structures de contrôle classiques de l'informatique (séquence, conditionnelle et itération) ainsi que le parallélisme. Une demande forte est alors apparue pour que le Grafcet passe du statut de formalisme de spécification au statut de langage de programmation de haut niveau, utilisable dans les automates programmables industriels.

Dans la suite de ce document, nous nous restreindrons à présenter le Grafcet "simple", c'est-à-dire sans étapes puits ou source, sans macro-étape et sans ordres de forçage. Ces différentes constructions particulières étant décrites et modélisées dans ma thèse [LP94]. De plus, le terme "Grafcet" désignera le formalisme et le terme "grafcet", un programme écrit en Grafcet.

2.1.2.1 Le formalisme Grafcet : syntaxe, postulats et règles

Le formalisme Grafcet est défini par une syntaxe, deux postulats temporels et cinq règles de fonctionnement.

- Syntaxe :

  le Grafcet est basé sur deux entités principales :

  - les étapes qui représentent les états du système et auxquelles sont associées les actions qui doivent être réalisées par l'automatisme commandé. Elles sont symbolisées par un carré et sont repérées par un numéro. Les étapes sont considérées soient comme actives (leurs actions sont alors exécutées) soit comme inactives. L'ensemble des étapes actives à un instant donné constitue la situation du Grafcet. Les étapes actives à l'initialisation sont dites initiales et sont représentées par un double carré.
  - Les transitions qui permettent de contrôler le passage d'un état à un autre en fonction de la valeur de la réceptivité qui leur est associée. Les réceptivités sont des fonctions booléennes composées à partir des valeurs des capteurs, des variables internes et des valeurs d'activité d'étapes. Elles sont représentées graphiquement par des traits horizontaux sur des liens et comportent en partie gauche un numéro de référence et en partie droite la réceptivité associée.

  Une description Grafcet correspond donc à un enchaînement d'étapes portant des actions et de transitions conditionnant le passage d'une étape à une autre ou plus généralement d'un groupe d'étapes à un autre. Les réceptivités sont naturellement liés aux actions portées par les étapes qui les entourent, et d'une manière générale, une étape sera désactivée lorsque les effets de l'action qu'elle porte auront engendré un changement de valeur d'un capteur.

  La figure 2.5 présente les différentes structures de contrôle exprimables à l'aide des étapes et des transitions. Par souci de simplification, les actions et les réceptivités n'y sont pas indiquées.

  A ce stade, on peut légitimement remarquer que le formalisme Grafcet ressemble fortement au réseaux de Petri (RdP). En terme de comportement, il correspond, à quelques nuances près, à la classe des réseaux Petri Interprétés.

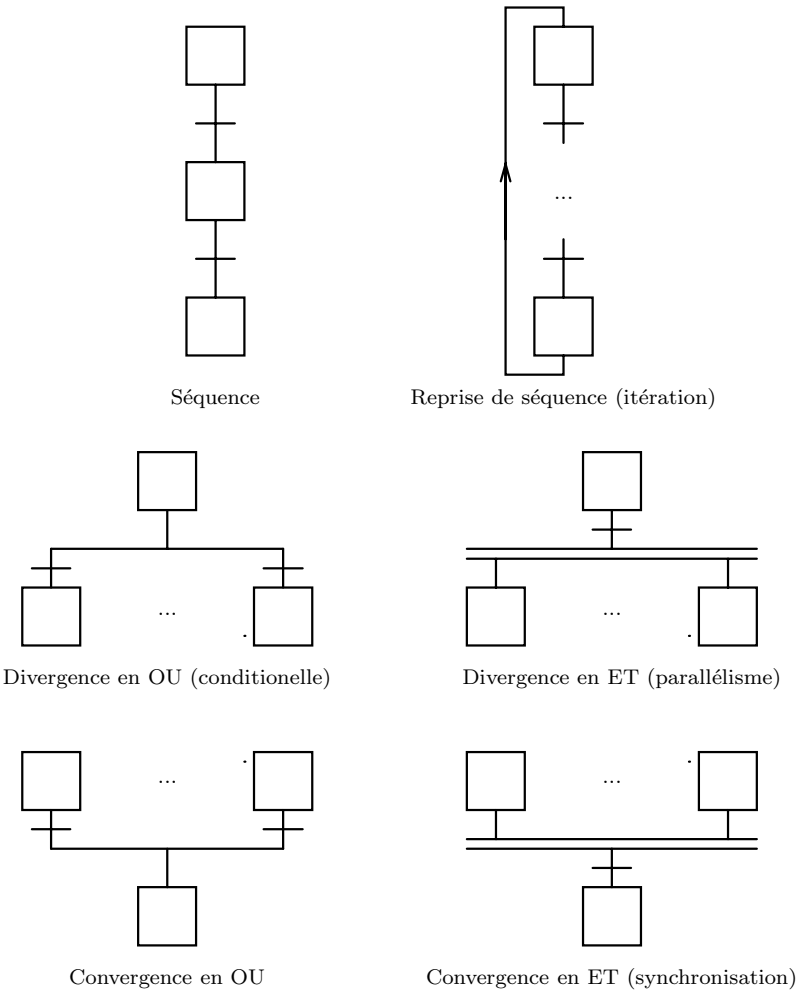

FIG. 2.5 - *Les différentes structures de contrôle en Grafcet*

– Postulats :

Lors de la définition du modèle Grafcet, deux postulats ont été retenus :

- Postulat 1 : Le modèle Grafcet postule que tous les événements sont pris en compte dès leur occurrence et pour toutes leurs incidences.
- Postulat 2 : Dans le modèle Grafcet, la causalité est considérée à temps nul.

De ces deux postulats, on peut conclure que, théoriquement, un Grafcet est attentif à toute modification de son environnement et qu'il peut y réagir instantanément. Ceci suppose donc l'utilisation de machines infiniment rapides à la fois pour l'acquisition, le traitement et l'émission des ordres. On voit bien là que les créateurs du modèle Grafcet ont pris ces hypothèses afin de dégager le concepteur d'une application des contingences matérielles pour qu'il puisse se concentrer uniquement sur les aspects logiques de son application. Ces postulats seront valides dès lors que l'application contrôlée est plus beaucoup plus lente que le système de contrôle.

On retrouve ici des hypothèses et une philosophie (oubli des problèmes d'implémentation au profit des problèmes de logique) très proche de celle revendiquée quelques années plus tard par les langages synchrones (voir la section 2.1.1.1).

Il est aussi nécessaire de préciser que les auteurs du Grafcet avaient une vision, que l'on pourrait qualifier d'asynchrone, des entrées : deux entrées ne peuvent changer de valeur simultanément.

– Règles de fonctionnement :

  La syntaxe étant précisée, le cadre d'évolution étant défini, les auteurs du Grafcet ont proposé cinq règles de fonctionnement pour le modèle Grafcet.

  – Règle 1 : A l'initialisation, seules les étapes initiales sont actives. Ceci traduit en général un comportement au repos.
  – Règle 2 : Une transition est dite validée si toutes les étapes amonts qui lui sont reliées sont actives. Une transition est dite franchissable lorsqu'elle est validée et que la réceptivité qui lui est associée est vraie.
  – Règle 3 : Une transition franchissable est immédiatement franchie. Les étapes immédiatement suivantes sont alors activées et les étapes immédiatement précédentes désactivées. Les activations et désactivations se font simultanément.
  – Règle 4 : Si dans un grafcet, plusieurs transitions sont simultanément franchissables, elles sont alors simultanément franchies.
  – Règle 5 : Si une étape est simultanément activée et désactivée, elle reste active. On donne donc priorité à l'activation.

A travers ces règles, on voit que le contrôle d'évolution se situe bien au niveau des transitions, qui doivent être validées avant d'être franchies lorsque les réceptivités associées sont vraies. On notera également que les évolutions se déroulent en parallèle : activation/désactivation et franchissements multiples simultanés.

Ces règles de fonctionnement montrent également deux différences essentielles avec les réseaux de Petri.

– En RdP, le marquage est numérique, c'est-à- dire que le franchissement d'une transition entraîne la suppression d'un nombre x de jetons dans la place amont et l'ajout de y jetons dans la place aval. En Grafcet, il y a désactivation de l'étape amont et activation de l'étape aval : on a donc un marquage booléen.
– En RdP, si plusieurs transitions sont franchissables, un choix non déterministe est effectué pour décider quelle transition devra être tirée. En Grafcet, toute les transitions franchissables le sont. L'exemple de la figure 2.6 montre les conséquence de ces choix. Dans le cas de gauche (RdP), les deux transitions sont tirables et seule l'une d'entre elle le sera. On arrivera donc à un marquage soit de la place 5, soit de la place 6. En Grafcet, il résultera l'activation de l'étape 5 et de l'étape 6 : la structure conditionnelle n'est donc pas exclusive et peut donc entraîner un parallélisme dit faible. Ce parallélisme faible n'étant d'ailleurs pas recommandé en Grafcet.

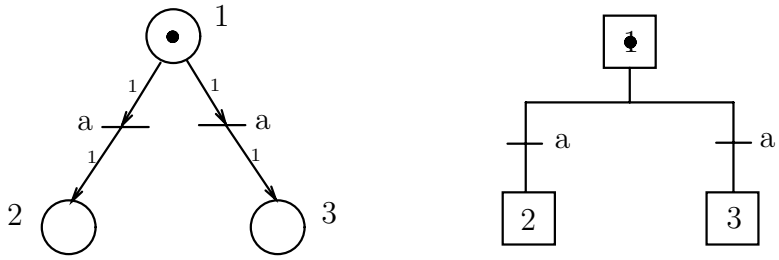

FIG. 2.6 - *Réseau de Petri et Grafcet : différences d'évolution*

2.1.2.2 Le langage Grafcet : interprétations

Le modèle Grafcet s'est rapidement imposé, du fait de sa ressemblance avec les méthodes existantes (en particulier les automates à états et les RdP) et de son aspect graphique. Assez vite, les concepteurs d'applications d'automatismes ont souhaité passer du niveau modélisation au niveau programmation et ont alors essayé d'implanter leurs modèles de conception à travers un langage : le langage Grafcet.

Plusieurs interprétations des règles de fonctionnement et des postulats ont alors vu le jour. Nous ne présenterons ici que les deux interprétations principales, celle dite sans recherche de stabilité (SRS) et celle dite avec recherche de stabilité (ARS).

– Interprétation SRS : c'est la plus souvent rencontrée. Elle respecte l'algorithme suivant :

```
Acquisition des entrées
Calcul des transitions franchissables
Franchissement des transitions franchissables
Mise à jour des actions
```

– Interprétation ARS : elle suit l'algorithme ci-après :

```
Acquisition des entrées
Tant que la situation est instable faire
|     Calcul des transitions franchissables
|     Franchissement des transitions franchissables
Refaire
```

Ces deux interprétations diffèrent sur la notion de stabilité. La première considère que la mise à jour des actions, c'est-à-dire le lancement des ordres associés aux étapes, doit être réalisée après un pas de calcul, alors que la seconde considère que la mise à jour des actions ne se fait que lorsqu'une situation stable est atteinte, c'est-à-dire que lorsqu'aucune transition ne peut plus être franchie.

L'algorithme SRS est sans nul doute l'algorithme le plus simple à mettre en oeuvre et celui qui garantit des temps d'exécution relativement constants. Par contre, il ne respecte pas le postulat 1 car toutes les incidences des événements d'entrée ne sont pas pris en compte. De plus, en général, les transitions en aval d'une étape portent une réceptivité avec une valeur vraie, lorsque les effets produits par les actions associés à l'étape ont eu lieu. Si après occurrence d'un événement, une étape devient active et que sa transition aval porte une réceptivité de valeur vrai, il est inutile d'exécuter les actions associées uniquement pour un cycle. Néanmoins, le risque posé par l'interprétation ARS est de ne jamais atteindre de situation stable, ce qui entraîne un arrêt d'évolution du programme et donc du contrôle. Il faut également noter que dans le cas ARS, le temps de calcul d'une nouvelle situation dépend du nombre d'itérations nécessaire pour l'atteindre : il n'est donc pas constant.

Dans le cadre de notre travail de recherche, nous n'avons volontairement pas pris parti pour l'une ou l'autre des interprétations, mais nous avons essayé de montrer que ces interprétations pouvaient bénéficier des apports des formalismes synchrones au niveau de la modélisation et de l'expression du modèle mathématique sous-jacent ainsi que des outils développés autour des langages synchrones et en particulier des outils de preuve.

2.1.3 Modélisation du Grafcet en Signal

2.1.3.1 Définitions et notations

Les définitions données ici pour le Grafcet sont largement inspirées de celles proposées pour les Réseaux de Petri. Nous nous intéressons au Grafcet sans ordre de forçage ni ordre externe (action). Une description complète peut être trouvée dans [LP94].

Définition 2.1
Une *structure de grafcet* est un quintuplet SG = <$\mathcal{E}$, $\mathcal{E}_0$, $\mathcal{T}$, Amont, Aval> tel que :
- $\mathcal{E}$ est un ensemble fini d'étapes repérées par des numéros deux à deux différents.
- $\mathcal{E}_0$ est un sous-ensemble de $\mathcal{E}$ d'étapes initiales.
- $\mathcal{T}$ est un ensemble fini de transitions repérés par des numéros deux à deux différents.
- Amont : $\mathcal{E} \times \mathcal{T} \rightarrow \mathbf{B}$ est l'application d'incidence avant où **B** représente l'ensemble des booléens **B** = {vrai, faux}.
- Aval : $\mathcal{E} \times \mathcal{T} \rightarrow \mathbf{B}$ est l'application d'incidence arrière.

Une structure de grafcet correspond à la partie graphique du Grafcet. Les applications Amont et Aval, qui peuvent être représentées sous forme de matrices, définissent pour chaque transition l'ensemble des étapes précédentes et suivantes : Amont(ϵ, τ) = vrai signifie que l'étape ϵ précède la transition τ ; Aval(ϵ, τ) = vrai indique que ϵ suit τ.

Définition 2.2
Une *évolution simple* d'un grafcet correspond au franchissement simultané de toutes les transitions franchissables (règle 4).
Une étape pouvant être au même instant activée et désactivée, la priorité est donnée à l'activation (règle 5).
Pour une étape donné ϵ, les conséquences d'une évolution simple sont les suivantes :
- Si $\exists\tau \in \mathcal{T}$, tel que Aval($\epsilon$, τ) = vrai et τ franchissable alors l'étape ϵ est activée : S(ϵ) = vrai.
- Si $\exists\tau \in \mathcal{T}$, tel que Amont($\epsilon$, τ) = vrai et τ franchissable, si $\forall\tau' \in \mathcal{T}$, tel que Aval($\epsilon$, τ') = vrai et τ' non franchissable, alors l'étape ϵ est désactivée : S(ϵ) = faux.
- Sinon S(ϵ) conserve sa valeur précédente.

La première règle correspond à l'activation de l'étape et la deuxième à sa désactivation.
Dans la suite de ce document, outre les définitions précédentes, nous utiliserons les notations :

E_i	étape de numéro *i*.
$S(E_i)$	état de E_i accessible à l'utilisateur : vrai (active) ou fausse (inactive).
T_i	transition de numéro *i*.
$R(T_i)$	valeur de la réceptivité associée à T_i.
$T(E_i)$	ensemble des transitions précédant E_i : $\{\tau \in \mathcal{T}$ / Aval(E_i, τ) = vrai$\}$
$T_j(E_i)$	$j^{ème}$ transition de $T(E_i)$.
$(E_i)T$	ensemble des transitions suivant E_i : $\{\tau \in \mathcal{T}$ / Amont(E_i, τ) = vrai$\}$
$(E_i)T_j$	$j^{ème}$ transition de $(E_i)T$.
$E(T_j)$	ensemble des étapes précédent T_j : $\{\epsilon \in \mathcal{E}$ / Amont(ϵ, T_j) = vrai$\}$.
$E_i(T_j)$	$i^{ème}$ étape de $E(T_j)$.

De plus, le symbole $\sum_{i=1}^{n}$ dénotera un 'ou' logique et $\prod_{i=1}^{n}$ un 'et' logique.

2.1.3.2 Interprétation SRS

Dans le cadre de l'interprétation SRS, un pas d'évolution correspond à une évolution simple telle que définie précédemment. A un instant t, réaliser un pas d'évolution simple correspond d'une part à l'acquisition d'un flot d'entrées, au calcul de la nouvelle situation et son émission vers le monde externe. Le modèle synchrone que nous utilisons impose que ces différentes opérations se déroulent en temps nul vis à vis de la partie opérative. L'instant $t + 1$ n'a pas de relation

temporelle avec l'instant t car nous utilisons un temps logique et non métrique. La seule relation existante entre t et $t+1$ est donc une relation de succession.

Modéliser l'interprétation SRS consiste en l'expression des règles permettant d'évaluer la situation à l'instant t en fonction de la situation à l'instant précédent $t-1$ et des valeurs des entrées aux instants t et $t-1$.

$$\mathrm{S}_t = \mathrm{F}\ (\mathrm{S}_{t-1},\ \mathrm{V}_t,\ \mathrm{V}_{t-1})$$

La situation d'un grafcet étant définie par l'état de chacune de ses étapes, cette équation se réécrit en :

$$\mathrm{S}_t(\mathrm{E}_i) = \mathrm{f}_i\ (\mathrm{S}_{t-1},\ \mathrm{V}_t,\ \mathrm{V}_{t-1})$$

bien que, en pratique, l'état d'une étape ne dépende que d'une partie de la situation du grafcet et que d'une partie des variables. Le calcul de la nouvelle situation correspond alors à l'évaluation des équations associées à chacune des étapes.

Equations d'évolution : Une étape ne pouvant être qu'active ou inactive, nous pouvons écrire :

$$\mathrm{S}_t(\mathrm{E}_i) = \text{vrai quand } \mathrm{E}_i \text{ reste active à } t \text{ ou quand } \mathrm{E}_i \text{ est activée,}$$
$$\text{autrement } \mathrm{S}_t(\mathrm{E}_i) = \text{faux}$$

ce qui s'exprime en Signal ainsi :

$$\mathrm{S}(\mathrm{E}_i) := \text{true when (”}\mathrm{E}_i \text{ reste active” or ”}\mathrm{E}_i \text{ est activée”) default false}$$

Le premier terme ”E_i reste active” modélise la non désactivation de l'étape i c'est-à-dire le fait que cette étape ne peut être désactivée si aucune transition aval n'est franchissable. Le deuxième terme ”E_i est activée” exprime son activation, c'est-à-dire les conséquences du franchissement d'une transition amont.

L'utilisation d'un ”ou logique” entre les deux expressions assure la règle 5 du Grafcet concernant l'activation et la désactivation simultanée d'une même étape.

L'expression ”E_i reste active” à la valeur vraie si E_i était active à l'instant précédent $t-1$ et qu'aucune des m transitions en aval n'est franchissable. En notant Franchissable(τ) la fonction définissant la condition de franchissement d'une transition τ validée par E_i alors :

$$\text{”}\mathrm{E}_i \text{ reste active”} := (\mathrm{S}(\mathrm{E}_i))\$1 \text{ and not } \sum_{j=1}^{m} \text{Franchissable}((\mathrm{E}_i)\mathrm{T}_j)$$

Si la transition T_j est précédée par q étapes E_k, le terme Franchissable($(\mathrm{E}_i)\mathrm{T}_j$) correspond à :

$$\text{Franchissable}((\mathrm{E}_i)\mathrm{T}_j) = \mathrm{R}((\mathrm{E}_i)\mathrm{T}_j) \text{ and } \prod_{k=1,k\neq i}^{q} \mathrm{S}(\mathrm{E}_k(\mathrm{T}_j))\$1$$

En effet, une transition n'est franchissable que si elle est validée (1^{er} terme de l'équation) et que toutes les étapes amont sont actives ($2^{ème}$ terme).

L'expression ”E_i est activée” prend la valeur vraie si au moins une des p transitions précédant l'étape E_i est franchissable ($\sum$), c'est-à-dire si elle est validée et que toutes les étapes amont sont actives ($\prod$) :

$$\text{”}\mathrm{E}_i \text{ est activée”} := \sum_{j=1}^{p} \left\{ \mathrm{R}(\mathrm{T}_j(\mathrm{E}_i)) \text{ and } \prod_{l=1}^{r} (\mathrm{S}(\mathrm{E}_l(\mathrm{T}_j(\mathrm{E}_i))))\$1 \right\}$$

A chaque étape du Grafcet, on peut donc associer une équation de la forme (1), instantiée par l'environnement local de l'étape considérée : transitions amont et aval, étapes amont. Ces différentes équations sont deux à deux indépendantes puisqu'elles ne sont basées que sur la situation antérieure et sur les valeurs des entrées. Le modèle synchrone que nous utilisons implique ensuite leur évaluation simultanée et immédiate.

Formation des réceptivités : La valeur d'une réceptivité $R(T_j)$ résulte de la composition des variables d'entrées de V (notées I_n) et des variables internes (X_i) par les opérateurs de OP. Chaque transition étant numérotée, le processus "formation des réceptivités" réalise l'association d'une transition et de sa valeur.

Quatre cas peuvent se produire suivant le type des variables (externes ou internes) et la façon de les utiliser :

- Variables d'entrée :
  $R(T_j) = ... \ I_n \ ...$ devient en Signal, $R(T_j) = ... \ I_n \ ...$
- Variables d'entrée sous forme événementielle :
  $R(T_j) = ... \uparrow I_n \ ...$ devient en Signal, $R(T_j) = ...$ (not(I_n\$1) and I_n) ...
  $R(T_j) = ... \downarrow I_n \ ...$ devient en Signal, $R(T_j) = ...$ (I_n\$1 and not(I_n)) ...
- Variables internes :
  $R(T_j) = ... \ X_i \ ...$ devient en Signal, $R(T_j) = ...$ ($S(E_i)$\$1) ...
  L'utilisation du retard est motivée par le fait que le calcul de la situation à l'instant t se fait en fonction de la situation à $t-1$.
- Variables internes sous forme événementielle :
  $R(T_j) = ... \uparrow X_i \ ...$ devient $R(T_j) = ...$ (not($S(E_i)$\$2) and ($S(E_i)$\$1)) ...
  $R(T_j) = ... \downarrow X_i \ ...$ devient $R(T_j) = ...$ (($S(E_i)$\$2) and not($S(E_i)$\$1)) ...

Ces équations font référence au passé par l'intermédiaire de l'opérateur \$. Ceci pose le problème de l'initialisation des variables externes et internes. Pour les premières, l'initiative en est laissée à l'utilisateur en fonction des spécifications de son application. Pour les secondes, nous avons posé que la situation d'un grafcet aux instant fictifs $t = -1$ et $t = -2$ était égale à la situation initiale. Les expressions $\uparrow X_i$ sont donc toujours fausses à $t = 0$.

Architecture générale : Nous avons précédemment défini d'une part une équation d'évolution par étape et d'autre part une équation de calcul de réceptivité par transition. Ces équations seront regroupées dans deux processus interconnectés comme le montre la figure 2.7. Cette séparation n'est pas fondamentalement nécessaire, car le compilateur remet toutes les équations au même niveau, mais autorise une programmation plus claire et plus compréhensible.

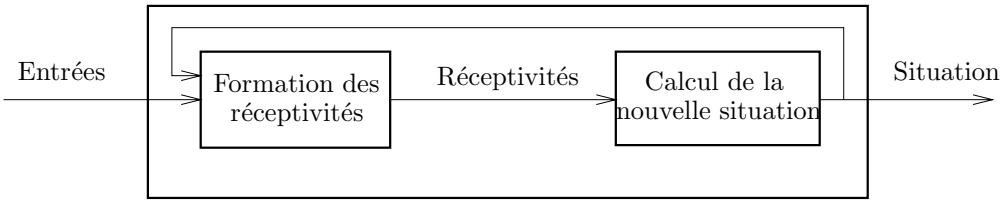

FIG. 2.7 - *Architecture générale SRS*

Le processus "formation des réceptivités" permet la définition des valeurs des réceptivités en fonction des entrées.

Le deuxième processus comporte les équations associées à chaque étape qui permettent de déterminer la situation atteinte par le grafcet après un pas d'évolution simple.

2.1.3.3 Interprétation ARS

La modélisation du Grafcet dans le cadre de l'interprétation ARS, repose d'une part sur la notion de stabilité et sur la notion d'évolution itérée dont les définitions sont données ci-après. Les

notions utilisées pour l'interprétation SRS sont réutilisées ici, les deux interprétations se basant sur les même modes d'évaluation des réceptivités et de franchissement des transitions.

Définition 2.3
Une situation sera dite *stable* si et seulement si aucune transition n'est franchissable sans modification du flot des entrées.

Définition 2.4
Une *évolution itérée* d'un grafcet est définie comme :
- une évolution simple avec acquisition d'entrées
- suivie d'une suite, éventuellement vide, d'évolutions simples sans acquisition d'entrées,

jusqu'à l'obtention d'une situation stable.

Une évolution itérée correspond naturellement à un pas d'évolution ARS. Sa définition implique l'utilisation des évolutions simples, donc de l'interprétation SRS. Celle-ci en constitue le sous-bassement nécessaire.

Nous utiliserons dans la suite les notions de *monde externe* et de *monde interne*. Le premier correspond au monde accessible par l'utilisateur : c'est le monde de la stabilité. Le second est le monde qui n'est accessible que par la machine : c'est le monde de l'instabilité. Il se modifie à chaque évolution simple alors que le monde externe ne change qu'à chaque évolution itérée.

Schématiquement, il est possible de représenter cette dualité monde externe/monde interne comme le propose [Lho93] (figure 2.8) :

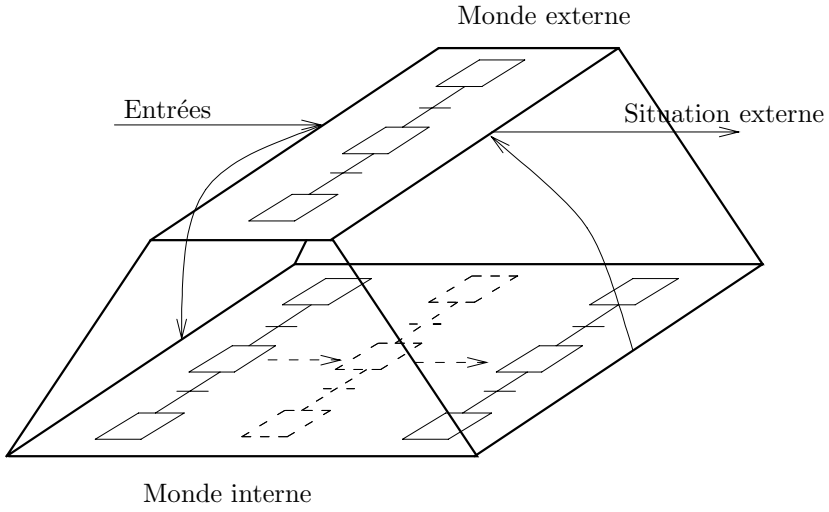

FIG. 2.8 - *Monde interne et monde externe*

D'après les hypothèses de synchronisme retenues, une évolution itérée qui correspond à l'acquisition des entrées, le calcul des évolutions simples nécessaires pour obtenir la stabilité et l'émission des résultats se déroule en temps nul pour l'utilisateur. L'instant $t+1$ est l'instant logique externe suivant l'instant t pendant lequel une nouvelle évolution itérée est réalisée.

Pour distinguer les différentes évolutions simples intervenant à t dans le calcul de la situation finale, nous les daterons ainsi : $t+\delta_1$, $t+\delta_2$... Ces instants sont donc des instants logiques internes successifs pendant lesquels seules des évolutions simples sont réalisées.

Dans la suite, nous utiliserons la notation $X(E_i)$ pour dénoter l'état d'une étape au niveau interne, $S(E_i)$ conservant sa signification originelle. Les valeurs de $X(E_i)$ et de $S(E_i)$ sont équivalentes uniquement lorsque le monde interne et le monde externe se synchronisent, c'est-à-dire lorsque le grafcet atteint une situation stable.

Architecture générale : L'architecture générale ARS présentée figure 2.9 décrit les différents processus utilisés ainsi que les flots de données.

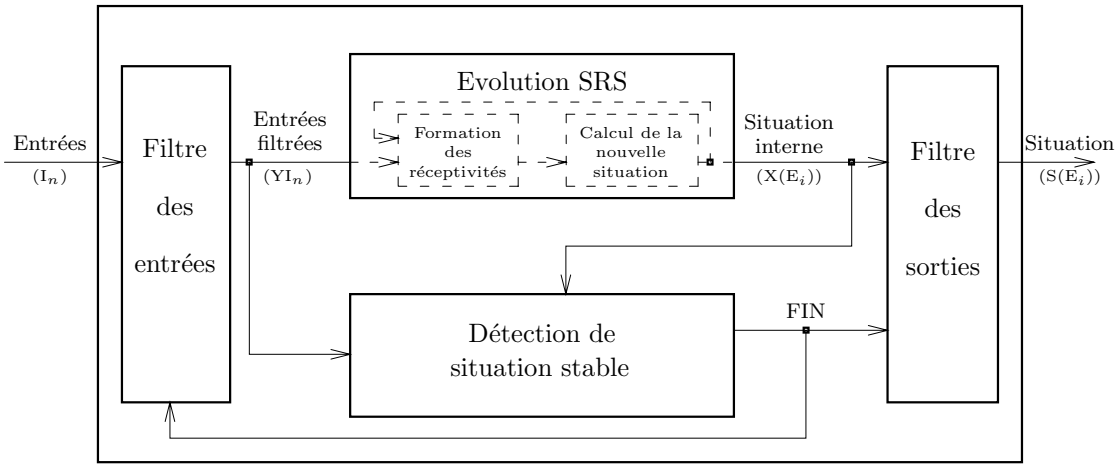

FIG. 2.9 - *Architecture générale ARS*

Comparativement à l'architecture SRS, on notera l'apparition de trois processus permettant :

- le filtrage des entrées, c'est-à-dire l'isolement du monde externe pour le monde interne. Il n'y a donc pas d'acquisition de valeurs lors de la recherche de stabilité.
- le calcul de la condition de stabilité nécessaire à la synchronisation entre les deux mondes.
- le filtrage des sorties, c'est-à-dire l'actualisation de la nouvelle situation au niveau externe.

Processus d'évolution SRS : Les équations utilisées dans le processus d'évolution SRS sont les mêmes que celles présentées dans le chapitre 2.1.3.2, à deux différences près :

- Les activités des étapes ne sont plus vues au niveau externe $S(E_i)$ mais au niveau interne $X(E_i)$. Cette remarque est valable aussi bien pour le calcul de la nouvelle situation que pour la formation des réceptivités. Il est important de noter d'ailleurs, que la référence à une étape dans une transition se fait donc au niveau interne et non externe. Ceci permet d'être sensible au passage même fugitif à travers une étape.
- Les entrées utilisées ne sont plus les entrées réelles mais les entrées dites filtrées, c'est-à-dire stabilisées le temps de la recherche de stabilité. Elles seront préfixées par la lettre Y. Les changements d'états des variables externes (fronts montants ou descendants) ne seront donc pris en compte que lors du premier pas d'évolution simple afin de respecter la fugitivité de ces informations.

Processus de détection de situation stable : Ce processus permet de déterminer si la situation atteinte est ou non stable. Il admet en entrée la situation à l'instant $t + \delta_n$ ainsi que les valeurs des entrées filtrées et produit un signal FIN caractérisant le type de la situation.

La stabilité étant définie au niveau des transitions, une équation est associée à chacune d'elles. Une transition ne sera plus franchissable soit si elle n'est plus validée, c'est-à-dire si au moins une des m étapes amont est inactive, soit si sa valeur deviendra fausse à l'instant $t + \delta_{n+1}$. On écrira :

$$\mathrm{F}_j := \text{when ((not } \prod_{k=1}^{m} \mathrm{X}(\mathrm{E}_k(\mathrm{T}_j))) \text{ or (not R'}(\mathrm{T}_j)) \text{)}$$

où F_j est un signal présent quand la transition n'est plus franchissable. Le terme R'(T_j) se calcule à l'aide des règles de traduction suivantes :

- Variables d'entrée :
  Si R(T_j) = ... I_n ... alors, R'(T_j) = ... I_n ...
- Variables d'entrée sous forme événementielle :
  Si R(T_j) = ... $\uparrow\mathrm{I}_n$ ... alors, R'(T_j) = ... false ...
  Si R(T_j) = ... $\downarrow\mathrm{I}_n$ ... alors, R'(T_j) = ... false ...
  La mise à faux de $\uparrow\mathrm{I}_n$ et de $\downarrow\mathrm{I}_n$ se justifie par le fait qu'un front (information de type événementiel) ne peut être consommé plusieurs fois successivement. R'(T_j) étant la représentation de la future valeur de la réceptivité associée à T_j, l'évaluation des fronts ne peut aboutir qu'à une valeur fausse.
- Variables internes :
  Si R(T_j) = ... X_i ... alors, R'(T_j) = ... X(E_i) ...
  Contrairement au processus d'évolution SRS, le calcul de stabilité prend en entrée la situation que l'on vient d'atteindre. Il n'y a donc pas lieu de retarder la valeur de l'activité de l'étape.
- Variables internes sous forme événementielle :
  Si R(T_j) = ... $\uparrow\mathrm{X}_i$ ... alors, R'(T_j) = ... X(E_i) and not X(E_i)\$1...
  Si R(T_j) = ... $\downarrow\mathrm{X}_i$ ... alors, R'(T_j) = ... not X(E_i) and X(E_i)\$1 ...
  Les variables internes pouvant évoluer, contrairement aux variables externes, leur front associé doit pouvoir être testé.

La situation atteinte sera déclarée stable lorsque tous les signaux F_j seront présents, soit :

$$\mathrm{FIN} := \text{true WHEN}_{j=1}^{q}\ \mathrm{F}_j \text{ default false}$$

avec $\text{WHEN}_{j=1}^{q}\ \mathrm{F}_j$ = when (F_1 when (F_2 when...))

Filtre des entrées : Ce premier filtre admet en entrée les valeurs fournies par l'environnement et le signal FIN dénotant le type de situation atteinte. Si cette dernière est stable, il faut alors autoriser l'acquisition de nouvelles entrées, sinon, les valeurs émises en direction des autres processus doivent être les dernières acquises.

En notant I_n la $\mathrm{n}^{ième}$ entrée, le processus émet une valeur YI_n définie comme suit :

$$\mathrm{YI}_n := \mathrm{I}_n \text{ default } \mathrm{YI}_n\$1$$

Bien sûr une telle équation est définie pour chaque entrée. Il reste maintenant à définir les contraintes permettant le respect du comportement souhaité. Deux ordres de synchronisation seront utilisés :

synchro {I_1, I_2, ..., I_p, when FIN\$1}
synchro {YI_1, YI_2, ..., YI_p, FIN}

Le premier permet d'imposer la lecture des entrées uniquement lorsque le signal FIN était vrai à l'instant précédent, c'est-à-dire que la situation atteinte était alors stable. Le signal FIN sera initialisé à la valeur vrai pour autoriser la première acquisition des entrées.

La deuxième instruction spécifie que les valeurs filtrées et le signal FIN sont synchrones, c'est-à-dire qu'elles évoluent au même rythme qui est d'ailleurs celui de l'horloge maximale.

Filtre des sorties : Ce filtre permet l'émission au niveau du monde externe de la situation lorsque celle-ci est stable. Pour chaque étape, une équation est donc définie :

$$S(E_i) = X(E_i) \text{ when FIN}$$

De plus une instruction de synchronisation est ajoutée pour spécifier les relations entre $X(E_i)$ et FIN :

$$\text{synchro } \{X(E_1), X(E_2), ..., X(E_n), \text{FIN}\}$$

Il y a donc synchronisme entre les valeurs internes de l'activité, le signal FIN et les entrées filtrées.

2.1.3.4 Actions et extensions du Grafcet

Actions : Les actions ont, elles aussi, été modélisées à l'aide du langage Signal. Qu'elles soient de type continu, conditionnel, mémorisé, impulsion, une action est traduite par une variable dont la valeur est calculée en fonction de l'activité des étapes et de son type.

Extensions du Grafcet : Les notions d'étapes sources et puits, de transitions source ou puit, de macro-étapes ou de forçage sont considérées comme des extensions du Grafcet qui permettent d'augmenter le pouvoir d'expression de ce langage.

Les étapes (resp. transitions) sources sont des étapes (resp. transitions) ne possédant pas de transitions (resp. étapes) en amont. Les étapes (resp. transitions) puits sont des étapes (resp. transitions) ne possédant pas de transitions (resp. étapes) en aval. Leur modélisation est donc triviale car elle revient à supprimer dans les équations précédentes, les termes inutiles.

Les macro-étapes sont des étapes auxquelles sont associées des extensions qu'elles remplacent. Elles permettent de concevoir un programme en adoptant une vision fonctionnelle. N'étant qu'une simplification graphique, elles se modélisent de manière évidente avec le travail précédent.

Le mécanisme dit de forçage, permet une décomposition hiérarchique d'un Grafcet. En effet, dans une application, plusieurs grafcets peuvent évoluer en parallèle et des dépendances hiérarchiques peuvent exister. En particulier, le grafcet gérant l'arrêt d'urgence doit pouvoir agir sur les autres grafcets. Une méthode de traduction par ajout de transitions sources et puits a été proposée et permet de ramener un ensemble de grafcets liés par des forçages à un grafcet simple. La hiérarchie des forçages, qui conditionne l'ordre d'évaluation des grafcets, est traduite de manière implicite en utilisant les relations temporelles logiques propres au langage Signal. De plus, le compilateur Signal permet de détecter automatiquement les cycles dans les hiérarchies de forçage sans avoir recours à des techniques spécifiques.

2.1.4 Validation du Grafcet

L'étape de modélisation du Grafcet en Signal étant achevée, elle ouvre la voie sur l'utilisation des outils du monde synchrone. Entre autre, la possibilité de simuler une programme dans l'une ou l'autre des interprétations, de générer des exécutifs temps réel [LPLSM99] et bien sûr de valider des applications.

Dans ce dernier domaine, nous avons étudié trois méthodes : la première est basée sur l'étude de l'outil de preuve associé à Signal, Sigali. La deuxième consiste à utiliser les systèmes de transition [LPQM94b]. La troisième repose sur les automates temporisés [LLPM99a][LLPM01]. Dans les

trois cas, nous avons essayé de valider des propriétés de sûreté ("les mauvaises choses n'arrivent pas") et des propriétés de vivacité ("les bonnes choses peuvent arriver"). Les vérifications étaient réalisées soit directement sur le modèle généré soit à l'aide d'observateurs dont l'atteignabilité étaient vérifiée (en particulier pour Sigali).

D'une manière plus générale, les méthodes utilisées classiquement pour la vérification de propriétés utilisent trois phases :

- Modélisation du programme dans un formalisme donné,
- Expression des propriétés à vérifier dans un formalisme particulier (généralement sous forme de formules de logique),
- Utilisation d'un outil permettant de valider les propriétés sur le programme modélisé.

2.1.4.1 Validation par les systèmes dynamiques polynomiaux : Signal-Sigali

Un programme Signal correspond à un ensemble d'équations reliant les signaux d'entrée à ceux de sortie. Tout programme ne manipulant que des informations booléennes peut-être encodé sur le corps des entiers modulo 3 ($\mathbf{Z}/3\mathbf{Z}$).

Pour les signaux, la valeur -1 signifie que le signal considéré est présent et qu'il possède une valeur fausse. La valeur 0 correspond à l'absence du signal et la valeur 1 indique la présence du signal avec une valeur vraie.

A chaque opérateur du langage, on associe alors une ou plusieurs équations de définition.

- Fonctions booléennes : $y := f(a_1, ..., a_n)$. Les différents signaux doivent posséder la même horloge ce qui conduit à l'équation suivante : $y^2 = a_1^2 = ... = a_n^2$.

  De plus une deuxième équation est associée en fonction de la valeur de f. Par exemple pour $y := a_1 \; or \; a_2$, on utilisera l'équation $y = a_1 a_2 (1 - (a_1 + a_2 + a_1 a_2))$.
- Opérateur $when$: $y := a_1 \; when \; a_2$ est représenté par l'équation : $y = a_1(-a_2 - a_2^2)$.
- Opérateur $default$: sur $\mathbf{Z}/3\mathbf{Z}$, $y := a_1 \; default \; a_2$ est traduit par : $y = a_1 + a_2 - a_1^2 a_2$.
- Opérateur de retard : $y = a\$1 \; init \; a_0$. Cet opérateur nécessite l'utilisation d'une mémoire notée ξ pour la mémorisation de la valeur précédente du signal a. En notant ξ', la valeur à l'instant courant et ξ_0 la valeur initiale alors, l'opérateur de retard se traduit par 3 équations : $y = a^2\xi$, $\xi' = a + (1 - a^2)\xi$ et $\xi_0 = a_0$.

En utilisant les représentations précédentes, tout programme peut être traduit et mis sous la forme d'un système d'équations dynamiques polynomiales du type suivant :

$$\begin{cases} P(X,Y) &= X' \quad (1) \\ Q(X,Y) &= 0 \quad (2) \\ Q_0(X) &= 0 \quad (3) \end{cases}$$

où X dénote l'ensemble des variables d'états qui proviennent de l'utilisation des signaux retardés (ξ). Y est composé des autres signaux ainsi que des horloges du programme.

L'équation (1) est appelée "équation d'évolution". Elle traduit le prochain état (X') en fonction de l'état courant et des autres signaux. L'équation (2) définit les contraintes sur X et Y. Enfin (3) est l'équation initiale modélisant l'état de départ du programme.

Un tel système est une représentation polynomiale d'un système de transitions. On a donc ainsi une représentation mathématique compacte de l'ensemble des états atteignables possibles. L'outil Sigali [Dut92] permet de manipuler ce système dynamique. La validation d'une propriété revient en général à modéliser les états vérifiant la propriété et ensuite, grâce à l'outil, à montrer que cet ensemble d'états est ou n'est pas atteignable à partir de l'état initial.

A titre d'exemple, si l'on souhaite vérifier que deux étapes (`i` et `j`) d'un grafcet peuvent être actives simultanément, on construira d'abord l'ensemble des états vérifiant cette propriété :

```
actives_i_et_j : gen([etat_i=1, etat_j=1, etat_fin=1]);
```

Dans cette formule, `etat_i` et `etat_j`, désignent les variables d'états associées aux étapes `i` et `j`. `etat_fin` indique la notion de situation stable. L'opérateur `gen` permet le calcul du polynôme représentant l'ensemble des états dans lesquels les trois variables précédemment décrites portent une valeur égale à `1`, ce qui traduit l'activité des étapes `i` et `j` dans une situation stable.

Ensuite, on vérifiera que cette état est atteignable à partir de l'état initial, en utilisant la fonction prédéfinie de Sigali, `accessible`, où `syst` décrit le système dynamique polynomial.

```
accessible(syst, actives_i_et_j);
```

Avec cette méthode et l'utilisation d'autres opérateurs de Sigali, il est relativement aisé de vérifier qu'une situation est atteignable, que des étapes sont activables ou désactivables simultanément, que le grafcet comporte ou non des situations instables etc..

Néanmoins, les temps de calculs sont généralement longs dès lors que les vérifications sont complexes et il n'est pas possible d'exprimer facilement des notions temporelles. Il faut noter par contre, que la représentation sous forme de polynômes est très compacte et est une solution à l'explosion combinatoire souvent rencontrée dans les méthodes basées sur les automates à états.

2.1.4.2 Validation par les systèmes de transition

Cette deuxième méthode utilise les systèmes de transitions [Arn92] et la notion de produit entre systèmes de transitions.

Définition 2.5

Un *système de transitions étiqueté* sur un alphabet A d'actions ou d'événements est un quadruplet $\mathcal{A} =< S, T, \alpha, \beta, \lambda >$ tel que :

- S est un ensemble fini d'états,
- T est un ensemble fini de transitions,
- $\alpha, \beta : T \rightarrow S$ sont des applications qui à chaque transition t associent son origine $\alpha(t)$ et son extrémité $\beta(t)$,
- $\lambda : T \rightarrow A$ est une application qui à toute transition t associe son étiquette $\lambda(t)$, définissant ainsi l'action ou l'événement qui provoque le franchissement de la transition.

Définition 2.6

Soient n systèmes de transitions $\mathcal{A}_i =< S_i, T_i, \alpha_i, \beta_i, \lambda_i >$ pour $i = 1, ..., n$. Le *produit libre* $\mathcal{A}_1 \times ... \times \mathcal{A}_n$ est le système de transitions $\mathcal{A} =< S, T, \alpha, \beta, \lambda >$ défini par :

$$
\begin{aligned}
S &= S_1 \times ... \times S_n \\
T &= T_1 \times ... \times T_n \\
\alpha(t_1, ...t_n) &=< \alpha_i(t_1), ..., \alpha_n(t_n) > \\
\beta(t_1, ...t_n) &=< \beta_i(t_1), ..., \beta_n(t_n) > \\
\lambda(t_1, ...t_n) &=< \lambda_i(t_1), ..., \lambda_n(t_n) >
\end{aligned}
$$

A partir de la notion de produit libre, il est possible de définir la notion de *produit de synchronisation*. Dans un produit libre, toutes les combinaisons entre transitions sont autorisées. Dans un produit de synchronisation, seules les transitions respectant une contrainte de synchronisation pourront être combinées. Dans le cas de la modélisation du Grafcet, on interdira par exemple qu'une transition portant l'étiquette a puisse se synchroniser avec une transition portant l'étiquette *non a*.

Avant de valider un grafcet, il est nécessaire de pouvoir modéliser son comportement à l'aide d'un système de transitions. Ce travail s'effectue en trois phases :

1. Construction des systèmes de transitions atomiques (STA) : ils modélisent, de manière locale, le franchissement d'une transition. Par exemple, dans le schéma 2.10, on montre la traduction d'une séquence et une divergence en ET.

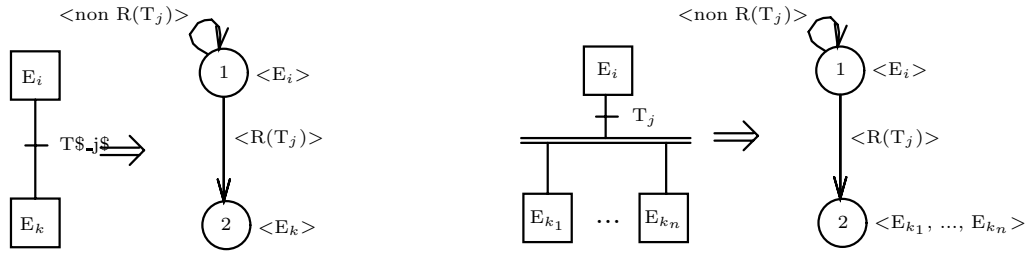

FIG. 2.10 - *Exemple de systèmes de transitions atomiques*

On associera donc un tel STA à chaque transition du grafcet que l'on souhaite modéliser. On notera sur la figure, l'étiquetage du système de transitions qui permet d'associer à chaque état la liste des étapes actives, et à chaque transition, la condition de franchissement.

2. Construction des systèmes de transitions de base (STB) : ils permettent de modéliser le comportement à partir d'une étape donnée. Pratiquement, si une étape e ne possède qu'une transition aval t, la STA de t correspond au STB de e. Par contre, si une étape possède plusieurs transitions aval t_1, t_2, ... t_n, son STB correspondra au produit de synchronisation des n STA associés à t_1, t_2, ... t_n. Un exemple est présenté, figure 2.11.

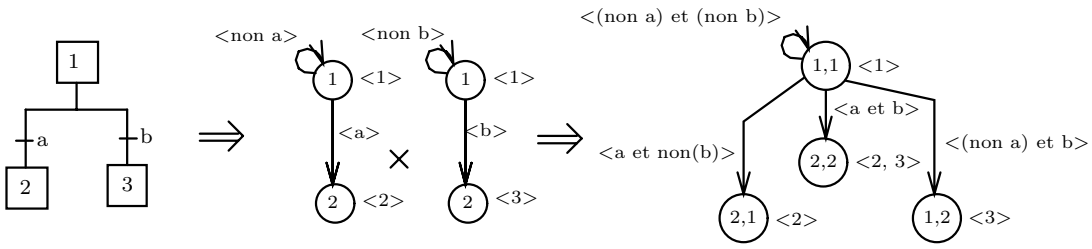

FIG. 2.11 - *Construction d'un système de transition de base*

Lors de la réalisation de ce produit, il est nécessaire de bien prendre en compte la valeur des étiquettes des transitions. En effet, si la réceptivité de la branche de droite du grafcet portait la valeur *non a*, la transition entre l'état (1,1) et (2,2) n'existerait pas.

3. Construction du système de transitions complet (STC) : il est construit de manière itérative à partir de la connaissance de la situation initiale. Par exemple, si l'on suppose que, dans le grafcet partiel de la figure 2.11, l'étape 1 est une étape initiale, le système de transitions complet partiel associé est celui présenté en partie droite de la figure. Une première itération doit permettre de le compléter en lui ajoutant les systèmes de transitions associés aux états (2,1), (1,2) et (2,2). Dans les deux premiers cas, il correspondent aux STB modélisant respectivement les étapes 2 et 3. Pour calculer le dernier cas (etat (2,2)), il suffit de construire le produit de synchronisation des STB modélisant la situation dans laquelle les étapes 2 et 3 sont actives et ainsi de suite. Dans le cas où plusieurs étapes sont initiales, on construira

un STC partiel correspondant au produit de synchronisation des étapes concernées et l'on utilisera ensuite le même procédé de construction.

Cette méthode permet la modélisation directe de l'interprétation Sans Recherche de Stabilité. Si l'on souhaite modéliser le comportement Avec Recherche de Stabilité, il faut et il suffit de calculer la fermeture transitive du graphe en tenant compte des contraintes sur les arcs du système de transitions. Lors de ce calcul, on peut être amené à ajouter un arc ayant le même état source et but : on détecte alors un cycle d'instabilité potentiel.

La modélisation ayant été effectuée, on obtient donc un système de transitions correspondant à un grafcet donné. Des propriétés pourront être validées en parcourant le système obtenu. Nous avons utilisé l'outil MEC qui propose quatre fonctions de base permettant de définir les états sources ou buts d'une transition et vice-versa. A partir de ces fonctions et des opérateurs ensemblistes, on peut construire les notions de successeur et prédécesseur et donc la notion d'accessibilité (potentielle ou inévitable) d'un ensemble d'états.

L'application de cette méthode sur un exemple concret permet alors de valider que des étapes peuvent ou ne peuvent pas être actives ou activées ou désactivées simultanément, qu'à partir d'une situation donnée, on peut ou non accéder à une autre situation, qu'une situation est une situation de blocage etc.. Ces validations peuvent être réalisées très simplement et très rapidement, mais cette méthode est souvent confrontée au problème de l'explosion combinatoire due au parallélisme implicite et explicite du langage Grafcet.

Nous avons utilisé la connaissance de l'environnement pour essayer de réduire la taille des systèmes de transitions. En effet, certains capteurs sont liés [2] et l'expression de leurs liens, dans le cadre des produits de synchronisation, amène à une diminution des états et des transitions.

2.1.4.3 Validation par les automates temporisés

Cette troisième méthode est sans doute la plus aboutie de celles que nous avons étudiées. Elle repose en grande partie sur le travail de D. L'Her [L'H97][LLPM99a][LLPM01] et permet la prise en compte de notions temporelles lors de la phase de validation d'un programme. La modélisation des programmes est effectuée à l'aide d'automates temporisés et les propriétés à valider sont exprimées sous forme de formules de la logique TCTL (Timed Computation Tree logic) et vérifiées à l'aide de Kronos [Yov93].

Définition 2.7

un **automate temporisé** est un quintuplet (S, s_{init}, H, A, Inv) où :

- S est un ensemble fini de sommets où s_{init} est le sommet initial.
- H est un ensemble fini d'horloges, variables réelles prenant leur valeur dans $\mathcal{R}^+$.
- A est l'ensemble des arcs de l'automate temporisé. Chaque arc est défini par un tuple de la forme (s_s, ψ, l, R, s_b). Les sommets s_s et s_b sont les sommets source et but respectivement de l'arc. ψ est la condition que doivent satisfaire les horloges de l'automate temporisé pour pouvoir franchir l'arc. l est une étiquette représentant une ou plusieurs actions. R est l'ensemble des horloges à remettre à zéro lorsque l'arc est franchi.
- $Inv : S \rightarrow \psi(\mathrm{H})$ associe à chaque sommet une contrainte temporelle appelée invariant. Tant que l'invariant d'un sommet est vrai, il est possible de rester dans ce sommet.

À l'initialisation, le système se trouve au sommet initial et toutes les horloges ont pour valeur 0.

Un automate temporisé est donc un automate auquel ont été rajoutées des contraintes temporelles sur ses arcs pour conditionner les franchissements et sur les états pour modéliser les durées d'activité.

2. En particulier les capteurs de position car un mobile ne peut-être à deux endroits simultanément.

Principe de la modélisation : Les évolutions du grafcet sont relatives aux entrées qui varient et au temps qui s'écoule. Pour modéliser à l'aide d'un automate temporisé, les évolutions d'un grafcet, il faut donc prendre en compte ces deux paramètres. Les informations portées par une transition d'un automate temporisé ne peuvent être que des contraintes temporelles, un sommet de l'automate représentera donc une situation Grafcet, une valeur des entrées et des temporisations. Les transitions correspondent à une modification des entrées ou à une évolution du temps induisant une modification des valeurs des temporisations. Si ces modifications entraînent une évolution de la situation Grafcet, l'état but représente la situation après évolution. Les invariants des sommets et les contraintes temporelles permettent d'exprimer les temporisations.

Sommet : Dans le cas général, un sommet de l'automate est défini par une situation du grafcet, une valuation des variables booléennes d'entrée et la valeur des temporisations apparaissant dans le grafcet. A une situation du grafcet peuvent donc correspondre plusieurs sommets de l'automate.

Horloge : Une horloge est associée à chaque étape apparaissant dans une temporisation. Ces horloges vont être gérées de manière à avoir pour valeur le temps depuis lequel l'étape associée est active ou inactive.

Invariant associé à un sommet : L'invariant associé à un sommet exprime la contrainte que doivent satisfaire les horloges pour qu'aucune temporisation ne change de valeur dans le sommet. Tout d'abord, nous cherchons les horloges pertinentes dans un sommet, c'est-à-dire celles associées à une étape utilisée comme référence d'une temporisation susceptible de changer de valeur. Elles correspondent aux horloges qui satisfont l'une des deux conditions suivantes :

- l'horloge h_i est associée à l'étape i, l'étape i est active et il existe une temporisation fausse ayant pour référence l'étape i: $t_j{=}t_{1j}/X_i/t_{2j}$. Cette temporisation est alors susceptible de devenir vraie. La contrainte associée à cette temporisation peut alors être écrite : $h_i{<}t_{1j}$. Cette expression peut être généralisée en remarquant que plusieurs temporisations ayant pour référence l'étape i peuvent être fausses au même instant. La contrainte associée s'ecrira alors: $h_i \leq \min\ t_{1j}$ pour $\{\ t_j{=}t_{1j}/X_i/t_{2j}$ avec t_j fausse $\}$

- l'horloge h_i est associée à l'étape i, l'étape i est inactive et il existe une temporisation vraie $t_j{=}t_{1j}/X_i/t_{2j}$ ayant pour référence l'étape i. Il est possible que cette temporisation devienne fausse. La contrainte associée à cette temporisation peut alors être écrite : $h_i{<}t_{2j}$. Cette expression peut être généralisée en remarquant que plusieurs temporisations ayant pour référence l'étape i peuvent être vraies au même instant. La contrainte associée s'écrira alors : $h_i \leq \min\ t_{2j}$ pour $\{\ t_j{=}t_{1j}/X_i/t_{2j}$ avec t_j vraie $\}$

Finalement, la contrainte associée à un sommet est la conjonction des contraintes associées aux horloges pertinentes si le sommet comporte au moins une horloge pertinente et *true* sinon.

Transition : Les arcs de l'automate correspondent à une modification des entrées et/ou à une évolution du temps amenant une modification des valeurs des temporisations. Une variable d'entrée peut voir sa valeur changée dans n'importe quel sommet. Seules les temporisations correspondant aux horloges pertinentes du sommet peuvent voir leur valeur modifiée.

- La contrainte temporelle associée à une transition indique si une ou plusieurs temporisations voient leurs valeurs modifiées. Lorsque le sommet source ne comporte aucune horloge pertinente, la transition n'est pas contrainte temporellement : sa garde est "true". Par contre si le sommet source comporte une ou plusieurs horloges pertinentes, alors la contrainte temporelle est une conjonction de propositions de la forme $h_i{=}t_i$ et $h_j{<}t_j$. La première forme correspond à un changement de valeur de la temporisation tandis que la deuxième indique que la temporisation reste inchangée.

- Les horloges dont les étapes de référence ont été activées ou désactivées lors de la transition sont mises à zéro de telle manière que la valeur de l'horloge représente toujours le temps depuis lequel l'étape est active, respectivement inactive.
- A partir d'une situation, de la valeur des entrées, de la valeur des fronts d'entrées, de la valeur des temporisations et de la valeur des fronts d'étapes, la nouvelle situation est calculée pour l'interprétation choisie et les temporisations sont mises à jour. Le sommet but est alors défini par la situation atteinte, les entrées et les temporisations mises à jour. Pour l'interprétation ARS, si une situation stable n'a pu être atteinte, un sommet représentant l'état instable est défini.

Les transitions de l'automate ainsi définies ne correspondent pas forcément au franchissement de transitions grafcet.

Construction : La construction de l'automate temporisé commence par la définition du sommet initial. Puis ce sommet est traité, c'est-à-dire qu'on calcule son invariant et les transitions qui en partent. De nouveaux sommets sont ainsi découverts. Le calcul se poursuit tant que tous les sommets n'ont pas été traités. Le nombre de sommets possibles étant fini, la terminaison de l'algorithme est assurée. Ensuite, il est possible d'exprimer des propriétés en logique TCTL et de le valider grâce à Kronos.

La logique TCTL : La logique TCTL (Timed Computation Tree Logic) [ACD90] [HNSS94] est une logique temporelle qui étend la logique arborescente CTL en y introduisant une variable : le temps. Elle utilise des symboles qui portent à la fois sur l'ensemble des exécutions ($\exists$: il existe une exécution, $\forall$: pour toutes les exécutions) et sur l'ensemble des états d'une exécution ($\mathcal{U}$: jusqu'à, $\Diamond$: il existe un état, $\Box$: pour tous les états). Pour introduire le temps explicitement dans la syntaxe, la portée des opérateurs temporels est bornée dans le temps : la formule $\forall\Box_{\leq 4} p$ signifie, par exemple, que sur toutes les exécutions du système, pour tous les états jusqu'à l'instant 4, la proposition p est vraie.

Quelques propriétés temporelles vérifiables : Grâce à l'utilisation de la logique TCTL, il est possible de vérifier les mêmes propriétés qu'avec les deux méthodes précédentes. Par exemple, pour montrer qu'une situation S1 est atteignable depuis une situation S0, il suffit de vérifier la formule : $S0 \Rightarrow \exists\Diamond\ S1$. De même, pour montrer qu'une situation est une situation de blocage, on utilisera la formule suivante : $(init \Rightarrow \exists\Diamond\ S) \wedge (init \Rightarrow \forall\Box\ (S \Rightarrow \forall\Box\ S))$. Cette expression signifie que depuis l'état initial, il existe un chemin menant vers un état S et qu'une fois l'état S atteint, on y reste.

TCTL, permet d'exprimer également des propriétés temporelles comme par exemple, pour vérifier qu'une étape d'un grafcet est active au plus t unités de temps on écrira $init \Rightarrow \exists\Diamond$ (S $\wedge$ (S $\Rightarrow$ S $\exists\mathcal{U}_{>t}$ S)) et on vérifiera que cette formule est fausse. On est également capable d'étudier le délai (minimum ou maximum) entre deux événements, par exemple l'activation et la désactivation d'une étape.

Optimisations Cette modélisation permet la validation de beaucoup de propriétés temporelles ou non. Elle a été appliquée [LSLP$^+$98] par exemple sur la cellule de production Korso [LL94]. Néanmoins des améliorations, en vue de diminuer la taille de l'automate construit et d'augmenter les performances ont été proposées par D. L'Her :

- Prise en compte de l'environnement : de manière similaire à la méthode basée sur les systèmes de transition, la prise en compte de l'environnement et donc des dépendances entre variables d'entrée du grafcet a été mise en oeuvre également avec les automates temporisés.
- Vérification modulaire : des travaux ont eu lieu sur la vérification modulaire pour essayer de valider des propriétés sur des parties de programme et non sur l'ensemble. Cette voie de recherche est difficile car elle ne permet de valider que des propriétés très locales.

– Nouvelle modélisation : une nouvelle modélisation a été proposée qui permet de regrouper des états en exprimant uniquement les variables d'entrée importantes pour un sommet donné et non toutes les combinaisons linéaires de variables d'entrée. Des gains en taille de plus de 30% ont été alors constatés.

2.1.5 Outils associés

L'ensemble de ces travaux de recherche a donné lieu à la réalisation d'outils permettant l'expérimentation et la validation des méthodes et techniques décrites précédemment. En premier lieu, un éditeur graphique de grafcet a été créé. Il permet de saisir un programme complet de manière simple et intuitive, tout en vérifiant que la règle de syntaxe du Grafcet (alternance étapes-transitions) soit respectée. Il autorise la mise en place d'actions sous le format normalisé (type de l'action, libellé de l'action, variable de feedback), l'ajout de macro-étapes et d'ordres de forçage.

Le programme, une fois saisi, peut être compilé en Signal : l'utilisateur a le choix de l'interprétation, SRS ou ARS. Lors de cette phase des vérifications syntaxiques et sémantiques sont effectuées afin de contrôler, entre autre, le respect de la syntaxe des réceptivités, le typage des variables etc..

A l'issue de la compilation Grafcet → Signal, le programme obtenu est compilé en C par le compilateur Signal de l'IRISA. Une redéfinition, automatique, des entrées-sorties permet alors l'utilisation d'un simulateur de programme en mode pas à pas ou continu.

Dans le cadre d'utilisation pédagogique en (option "Systèmes Réactifs et Intelligents" - Maîtrise Informatique - Université de Bretagne Occidentale), l'ensemble peut être relié à une cellule de production fictive, composée de tapis roulants et de bras manipulateurs permettant aux étudiants d'appliquer les notions vues lors des enseignements.

D'autres réalisation ont été effectuée afin de contrôler des maquettes réelles de robots (mini bras manipulateur FischerTechnik, robot LegoMindStorms) ou d'automatismes (maquette d'ascenseur).

La figure 2.12 présente la simulation réalisée pour l'application Korso [LL94]. Au premier plan, on visualise la cellule de production et en arrière-plan les différents grafcets gérant les équipements de la cellule. On notera que certaines étapes sont représentées en rouge, car elles correspondent aux étapes actives lors de la saisie de cette figure.

Du coté de la validation, un compilateur Grafcet → Systèmes de Transitions a également été construit. Il reprend la partie analyse du compilateur Grafcet → Signal, et la complète en permettant la construction, soit en SRS, soit en ARS, du système de transitions équivalent à un grafcet donné. Lors des calculs des produits de synchronisation, les contraintes sur les étiquettes sont analysés grâce à l'utilisation des diagrammes de décision binaires [Bry86].

Pour les automates temporisés, des compilateurs ont été construits pour l'une ou l'autre des modélisations proposées. Un outil nommé "Vérif" a également été développé : il propose de charger un grafcet, de le transformer sous forme d'un automate temporisé puis de valider des propriétés. L'intérêt de cet outil est que les propriétés sont exprimées non pas sous forme de formules de logique TCTL, mais sous forme de questions prédéfinies se rapportant directement au Grafcet. L'outil peut donc être utilisé par des non spécialistes. Il peut être enrichi facilement par des nouvelles questions dès lors que l'utilisateur sait leur associer une formule TCTL ad hoc.

2.1.6 Conclusion

Ce travail sur la modélisation et la validation du Grafcet s'est étalé de 1991 à 1999. Il a donné lieu à de nombreuses publications et a permis de montrer tout l'intérêt des méthodes synchrones pour la définition de la sémantique des langages de l'automatisme ainsi que des outils de preuve dans ce domaine.

Les connaissances que j'ai pu acquérir au cours de ces années, tant sur le domaine que sur les méthodes et les outils sont réutilisées dans mon deuxième projet de recherche sur le contrôle à distance de systèmes mécaniques sur des réseaux de communications non fiables.

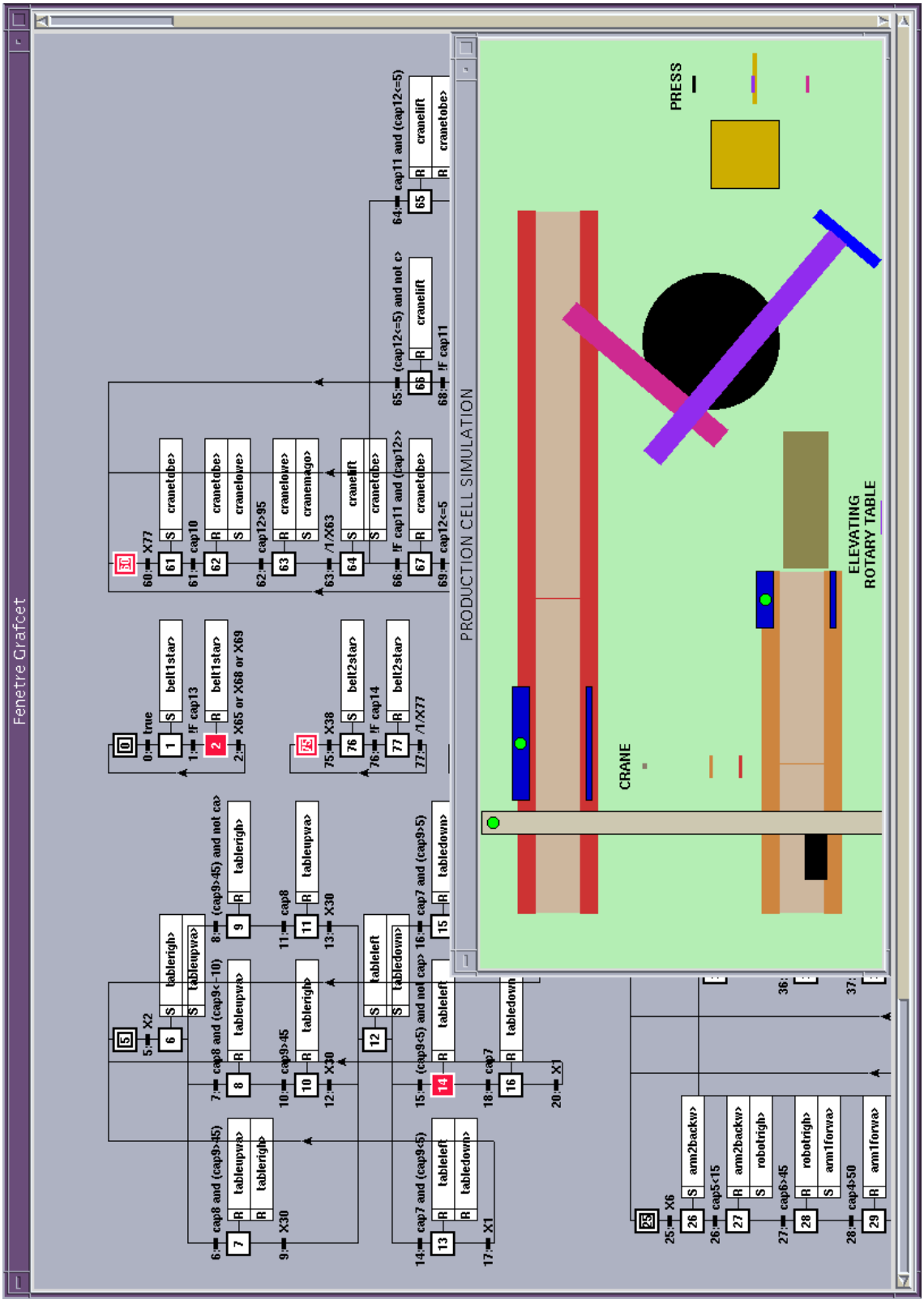

FIG. 2.12 - *Simulation de la cellule Korso*

2.2 Contrôle à distance sur un réseau sans qualité de service

Au cours des dix dernières années, un développement très important des réseaux informatiques et en particulier du réseau Internet a eu lieu. Cette progression s'est réalisée en raison du développement des infrastructures au niveau mondial, de la baisse des coûts des ordinateurs personnels, de la possibilité de raccordement des usagers en utilisant les réseaux téléphoniques et surtout des possibilités que représente ce réseau interconnectant des milliers d'utilisateurs potentiels (près d'un milliard prévus en 2004). L'Internet permet principalement l'échange d'informations grâce à des documents de type hypertexte, avec une interactivité relativement pauvre entre l'utilisateur, qui consulte, et les créateurs des documents. Néanmoins, l'Internet est également utilisé pour réaliser des échanges entre utilisateurs, soit de manière asynchrone (avec le courrier électronique par exemple) soit de manière synchrone (le chat ou clavardage en étant le meilleur exemple).

Notre travail se place dans ce dernier domaine. Nous essayons en effet d'évaluer les possibilités d'interaction et de contrôle de systèmes mécaniques au travers des réseaux de type Internet.

Les concepteurs d'automatismes industriels n'ont pas attendu l'avènement de l'Internet pour utiliser les réseaux dans leurs applications. Les réseaux de terrains, tels que Modbus, FIP, Telway ou Profibus existent depuis les années 1980 [Sch02]. Leur introduction a d'abord répondu aux besoins de déport des capteurs/actionneurs puis de conduite et de supervision des installations. Ces réseaux étant proches des applications et utilisés pour réaliser des boucles de contrôle-commande, ils possèdent donc des caractéristiques de type temps-réel dur afin d'assurer des délais de transport des informations faibles, vis-à-vis des temps de cycle des applications [KgRD02]. Ils sont de plus déterministes. Depuis quelques années, on observe un rapprochement avec les réseaux informatiques tant au niveau matériel que logiciel afin de diminuer les coûts, de profiter des avancées technologiques et des solutions ouvertes. En particulier on notera l'émergence de l'Ethernet industriel, bien que ce dernier ne soit pas un réseau déterministe. De nombreux travaux proposent néanmoins des solutions pour pallier ce problème [Vit01][GRD02][SGSB03] par l'implantation de protocoles spécifiques et par la gestion des configurations et des équipements réseau.

De plus, l'émergence de la dimension réseau se traduit par l'intégration de serveurs Web permettant la remontée et la présentation des données dans des solutions industrielles, comme par exemple à travers les coupleurs Ethernet ETZ 510 de Schneider Electric [RG03][BMG02] ou dans les caméras de la famille AXIS. Il faut néanmoins noter que dans le cadre industriel, les réseaux sont généralement sur-dimensionnés par rapport à la quantité d'information qu'ils véhiculent. Ils peuvent être alors vus comme des réseaux fiables et disponibles en permanence.

L'"e-productique[3]" a pour objectif d'aborder le contrôle à distance du côté de sa plus lointaine limite, très au delà des limites de l'entreprise et du réseau géré et contrôlé par ses employés. L'Internet est ici utilisé au sens large, en partant de l'hypothèse qu'il n'offre aucune qualité de service. Le système à contrôler se situe dans un lieu donné, l'utilisateur quant à lui peut se trouver n'importe où dans le monde dès lors qu'il possède un accès au réseau Internet. Il doit pouvoir envoyer des commandes et recevoir des informations de manière similaire à l'opérateur face à la machine.

Ce type de mode de commande fait partie, selon la classification proposée par Sheridan [She92], de la commande supervisée. Précisément, nous nous situons dans le quatrième cas de la figure 2.13. Les traits en pointillés indiquant une interaction faible et ceux pleins signifiant une interaction forte, nous introduisons donc le réseau Internet entre l'opérateur et l'ordinateur. C'est ce dernier qui est au contact de la machine et qui assure son contrôle permanent (boucles de régulation entre autres). L'opérateur va assigner des ordres de haut niveau et pouvoir visualiser leurs effets à travers l'affichage de valeurs numériques ou de retours visuels et sonores. Nous sommes donc dans un cas de commande supervisée avec des contraintes temps réel que l'on pourrait qualifier de molles.

3. En anglais, e-manufacturing.

Le cas 3, quant à lui, correspond également à une commande supervisée. Dans une telle configuration, c'est l'opérateur humain qui va agir directement sur le système. Des études ont été menées également dans cette configuration [Lel00] pour autoriser l'utilisateur à avoir un contrôle distant. Malheureusement, les contraintes de l'automatique imposent un contrôle régulier, et donc un réseau respectant de fortes contraintes temps réél, peu compatibles avec les caractéristiques du réseau Internet et donc il est nécessaire d'avoir recours à des techniques de compensation induisant des retards importants.

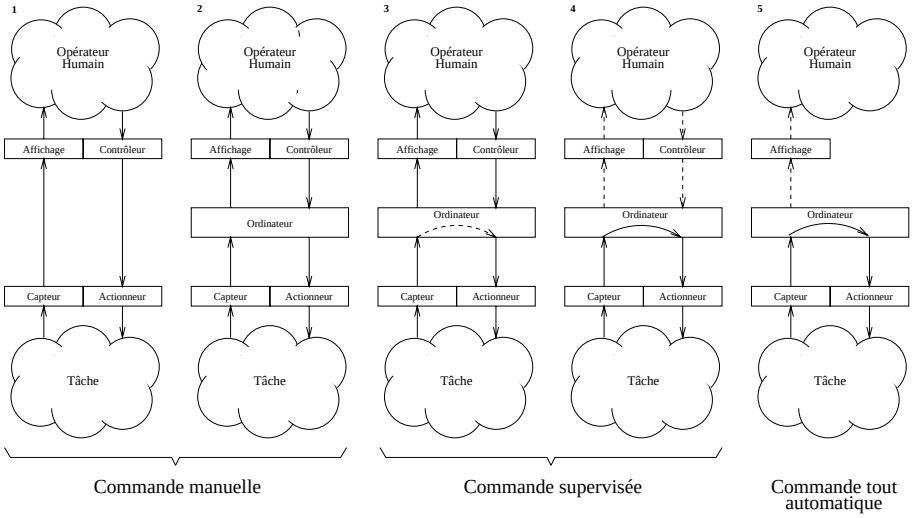

FIG. 2.13 - *Classes de contrôle selon Sheridan*

Les applications de l'e-productique sont potentiellement nombreuses. Nous les avons classées en quatre thèmes :

- **Formation :** que ce soit en formation initiale ou en formation professionnelle, l'apprentissage de la conduite de systèmes mécaniques se réalise aujourd'hui à la fois sur un plan théorique hors de l'atelier et sur un plan pratique face à la machine. Cette dernière phase pourrait tout à fait être réalisée à distance dès lors que le problème de mise en place des objets manipulés par la machine est résolu, par exemple par du personnel sur place.

  L'intérêt principal est la diminution des coûts. En effet, en formation professionnelle, on limiterait les déplacements des personnels du lieu de l'emploi au lieu de formation. Pour la formation initiale, on constate que les établissements possèdent différentes machines et que celles-ci ne sont que peu utilisées. L'accès distant et la mise en commun de moyens (voire la création de centres spécialisés) autoriserait alors une meilleure rentabilité du parc de machines. De plus, le développement de ces techniques permettrait la formation dans des pays n'ayant pas les ressources suffisantes pour investir dans des machines performantes mais possédant des infrastructures de communication standard.

- **Expertise :** certaines opérations nécessitent une connaissance particulièrement fine d'une machine dépassant les connaissances des opérateurs, comme par exemple des réglages particuliers ou des opérations spécifiques. Actuellement, un expert doit donc se déplacer pour réaliser ces opérations. On peut tout à fait imaginer que cet expert puisse opérer à distance, ce qui limiterait ses déplacements, augmenterait sa capacité d'intervention et donc aurait des conséquences finales sur le coût de la production.

- **Maintenance :** en terme de maintenance préventive, l'utilisation du technologies réseau est déjà opérationnelle. Des statistiques de fonctionnement sont remontées automatiquement et présentées graphiquement aux utilisateurs à travers de simples navigateurs Internet.

Mais, dans le cadre d'une maintenance corrective, il est souvent nécessaire qu'un technicien spécialiste se déplace afin de diagnostiquer la panne et de la réparer. L'introduction de l'e-productique, permetterait d'effectuer certains tests et donc une partie ou la totalité du diagnostic à distance, puis soit de réparer, soit d'expédier la pièce ad hoc sans déplacement obligatoire d'une personne. Il y a alors un gain financier, mais surtout un gain en terme de réactivité.

– **Production industrielle :** comparativement aux trois autres thèmes précédents, la mise en place d'une véritable production industrielle dans le cadre de l'e-productique reste plus hypothétique. Néanmoins les technologies de communication, les possibilités d'accès distants et de prise de contrôle faciliteront le travail et de plus en plus d'opérations seront réalisées de cette manière à l'avenir.

Nous ne sommes aujourd'hui qu'au début de l'e-productique et l'objectif final est d'être capable de réaliser les applications proposées ci-dessus en fournissant aux utilisateurs des systèmes à la fois simples, performants et sûrs. Dès aujourd'hui, on observe un intérêt grandissant autour de ce thème et l'arrivée de solutions et de produits industriels.

Nous utilisons volontairement ici le terme d'e-productique, pour distinguer notre approche de celles de MES (Manufacturing Execution System) [Boe00]. En effet, dans la classification traditionnelle des flux d'information qui traversent les industries manufacturières, des niveaux de point de vues sont discernés suivant à la fois les constantes de temps auxquelles peuvent et doivent réagir les personnes concernées et les rôles joués.

Ainsi, la stratégie globale qui permet d'organiser l'entreprise dispose de méthodes d'analyse et de décision pour lequels des logiciels existent, réunis sous l'acronyme ERP (Enterprise Ressource Planning), le niveau tactique qui est intermédiaire entre l'atelier et la direction stratégique s'est vu affecter l'acronyme MES. Il joue un rôle crucial de relayage d'informations entre les ordres ou directives et la réalité opérationelle sur le terrain. Cette dernière est celle qui nécessite le plus de réactivité, le plus de personnel en général et de compétences concrètes. C'est au niveau opérationnel que se situent les gisements de productivité les plus grands ainsi que les causes potentielles des plus grandes pertes.

Notre travail vise à montrer que les capacités de transmission d'information atteintes par les technologies Web, peuvent apporter des gains jusqu'au niveau opérationnel, où que soient situés le personnel et les machines. De ce fait, il s'inscrit à la charnière des niveaux opérationnel et tactique, comme une fonctionnalité supplémentaire du MES ou bien comme une fonctionnalité étendue des automatismes de terrain. L'évanouissement de la frontière entre le niveau tactique et le niveau opérationnel, explique l'apparition des notions de l'entreprise étendue, de l'usine transparente [Vie00].

Dans la suite de ce chapitre, nous commencerons d'abord par présenter quelques expérimentations de référence, qui nous permettront alors de définir les différentes caractéristiques que doit posséder un système d'e-productique. Ensuite, nous exposerons l'architecture logicielle que nous avons mise en place et les originalités de l'approche que nous avons menée. Ce chapitre se conclura par une présentation des applications développées et des résultats obtenus, en particulier en terme d'étude du réseau Internet.

2.2.1 Travaux de référence

Beaucoup de travaux ont eu lieu dans le domaine du contrôle et de la commande à distance. Ils répondaient en général aux problèmes liés au travail dans des zones soient dangereuses soit inaccessibles par l'Homme. Ne seront présentés ici que des expériences marquantes qui, même si elles ne se basent pas toutes sur l'utilisation d'Internet, ont apporté leur pierre dans la réflexion sur le domaine de l'e-productique.

2.2.1.1 De Goertz à Lindbergh !

Parmi les premiers travaux de contrôle à distance, citons ceux de R. Goertz dans le cadre du laboratoire Argonne aux Etats-Unis dans les années 50 [Goe52]. Il s'agissait de manipuler des éléments radioactifs placés dans une zone protégée. Ni l'informatique, ni les réseaux de communication n'étaient utilisés et l'on peut considérer que la manipulation se réalisait de manière synchrone avec des opérateurs proches des éléments à manipuler. De tels travaux ont également été menés au CEA par J. Vertut dans les années 70 [Esp01]. Même si l'on est loin ici de l'e-productique, on voit bien que ces recherches permettaient d'accéder à des environnements hostiles sans risque à travers l'utilisation de nouvelles technologies.

Quelques années plus tard, et encore aujourd'hui, des exemples très intéressants de contrôle à distance ont été réalisés dans le cadre de la conquête spatiale où l'un des problèmes rencontrés est lié aux délais de communication (3 secondes sur la Lune, plus de 3 heures aux limites du système solaire) qui sont, dans l'espace, principalement dus aux distances à parcourir. On voit bien ici qu'entre le moment où l'on produit un ordre et le moment où on le reçoit, l'information obtenue peut ne plus être valide et les ordres émis inutiles voire dangereux.

Dans le cadre des premières missions, en particulier sur la Lune, les stratégies de contrôle étaient de type "move and wait" : l'opérateur donnait un ordre au système, qui le réalisait. Une fois l'objectif atteint, le système s'arrêtait et attendait l'ordre suivant. Cette méthode garantissait un contrôle assez fin du système dès lors que l'on lui envoyait des "petits" ordres provoquant, par exemple, de faibles déplacements et que l'on vérifiait et analysait les données envoyées avant de fournir l'ordre suivant. Dans le cadre de la mission Surveyor de la Nasa, le temps moyen entre deux ordres était de 25 minutes, à comparer avec un délai de communication de 3 secondes. Cette faible réactivité a conduit les ingénieurs, en particulier dans le cadre des programmes en direction de la planète Mars, à modifier leurs stratégies et donc à donner plus d'autonomie aux véhicules d'exploration. Lors de la mission "Mars Pathfinder", les opérateurs fournissaient des buts à atteindre et non plus des séries d'ordres simples. L'opérateur est donc maintenant dans une position de superviseur : il est sorti de la boucle de contrôle du robot.

L'autre apport du télé-contrôle dans le domaine spatial concerne les procédures de reprise en cas de problèmes techniques. En effet, il n'est pas envisageable de pouvoir réparer un objet dans l'espace[4] et il faut donc, dans la mesure du possible être capable de fonctionner dans des modes dégradés. La panne la plus rédhibitoire étant de toute évidence la perte de communication avec l'engin spatial. Dans un tel cas, la mission s'arrête et tous les efforts sont consacrés au rétablissement de la communication, ce qui par exemple mis plus de six mois pour la sonde d'étude du Soleil Soho en 1999 [Van99]. La rupture de connexion ou sa dégradation doivent donc être prises en compte lors de la conception des systèmes, que ce soit du côté de l'utilisateur ou du système à contrôler.

Une autre expérience marquante dans le domaine du contrôle à distance est l'"Opération Lindbergh" [5] au cours de laquelle un médecin français a pu réaliser une opération chirurgicale sur une malade de l'hôpital de Strasbourg depuis un bureau de New-York, en septembre 2001 [Web01]. Au cours de cette première mondiale, le chirurgien disposait d'équipements permettant de commander un robot reproduisant fidèlement ses mouvements et d'informations à la fois visuelles et sonores montrant le champ opératoire. La communication entre les deux lieux de l'expérimentation était réalisée grâce à un réseau haut débit, utilisant la technologie ATM, avec un délai d'aller-retour garanti inférieur à 200 ms. En pratique les temps de communication moyen étaient de l'ordre de 150 ms. La bande passante réservée était de 10 Megaoctets. Ce projet a coûté plus de 2 millions d'Euros (dont la moitié pour le robot Zeus), mobilisé jusqu'à 80 personnes et nécessité plus de 3 ans de préparation. On est ici dans un cadre bien différent de celui de l'e-productique. En effet, l'un de nos objectifs est de proposer des solutions peu coûteuses à la fois en terme d'investissement logiciel et matériel ainsi qu'en terme de fonctionnement. De plus, les garanties

4. à l'exception des engins très proches de la Terre comme cela a été le cas pour le télescope Hubble en 1993, 1997, 1999 et 2002.

5. Du nom du premier aviateur à traverser l'atlantique de New-York à Paris.

de délai et de bande passante sont sans communes mesure avec celles que l'on peut rencontrer aujourd'hui dans l'Internet.

2.2.1.2 Expérimentation sur Internet

En parallèle des expériences évoquées dans la partie précédente, de nombreux travaux ont eu lieu autour du contrôle de systèmes mécaniques connectés sur l'Internet. Dans la suite, nous ne présenterons que trois expériences : *KhepOnTheWeb* [SF00] car cette démonstration est sans doute la première expérience de robotique mobile sur le Web, l'*Australian Telerobot* [TD97] qui fonctionne depuis 1996 et qui représente l'une des expérimentations les plus abouties et *Ariti* [OMKC00] pour son approche originale d'aide à l'utilisateur.

Outre ces trois projets, de nombreuses expérimentations [GS01] ont eu lieu depuis quelques années. Nous pouvons citer, sans vouloir être exhaustif, CyberCut [SW96], PumaPaint [Ste98], Telegarden [GS], TeleLabs [BCQ+01], les travaux du LIRMM [FAAdLR03], de l'INSA de Lyon [LPS+03], du CRAN [GML03] etc.. Toutes ont abordé le problème du contrôle à distance de systèmes sous un angle original en essayant de résoudre les problèmes inhérents à l'utilisation de l'Internet comme médium de communication. Les résultats sont tels qu'actuellement de nombreuses plateformes sont en cours de développement comme par exemple dans les AIP pour des besoins pédagogiques ou encore dans des entreprises telles que l'usine Schneider Electric de Merpins [TPL03].

– KhepOnTheWeb :

  Cette expérience a été menée d'octobre 1997 à juin 2000 par P. Saucy et F. Mondada à l'EPFL en Suisse [SF00]. Il s'agit d'un petit robot de type Khepera (diamètre 50 mm - figure 2.14 en haut à droite) sur lequel a été placée une mini caméra et devant circuler à travers un labyrinthe. Ce dernier possède entre autre un miroir permettant à l'utilisateur de voir le robot et une rampe, qui une fois gravie, donne une vue sur l'ensemble du labyrinthe (voir figure 2.14 en bas à droite).

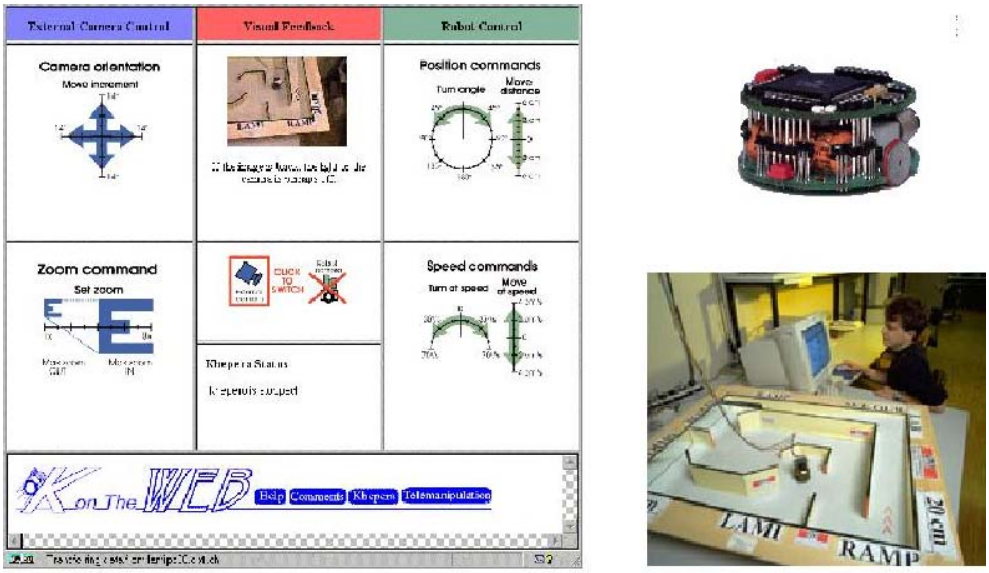

FIG. 2.14 - *L'interface de contrôle, le robot et labyrinthe*

Du point de vue logiciel, l'utilisateur disposait d'une interface (figure 2.14 partie gauche) lui permettant d'envoyer des ordres et de recevoir des images vidéos en temps réel provenant de la caméra embarquée sur Khepera et d'une caméra fixe dirigée vers le labyrinthe.

L'interface proposée était construite en langage HTML (Hypertext Markup Language) et comprenait des images cliquables. Les actions de l'utilisateur étaient transmises vers un serveur qui exécutait alors des programmes CGI (Common Gateway Interface). Ces derniers transmettaient les ordres directement au robot mobile par l'intermédiaire d'un lien filaire. D'un côté comme de l'autre du système, aucune intelligence n'était présente donc aucune autonomie de réaction possible lors d'occurrence d'événements non prévus.

En terme de résultats, cette expérience a démontré la faisabilité de l'accès et du contrôle distant et son intérêt puisque sur une année, plus de 27500 personnes se sont connectés sur le site et environ 45% d'entre eux ont effectué des mouvements avec le robot. Néanmoins les auteurs ont remarqué que peu de personnes essayaient de prendre le contrôle plusieurs fois de suite, sans doute par manque d'objectifs.

KhepOnTheWeb n'est plus accessible aujourd'hui, mais une démonstration en ligne d'un système proche peut être trouvée dans un musée virtuel de la robotique ouvert par les constructeurs des robots Khepera.

– **Australian Telerobot :**

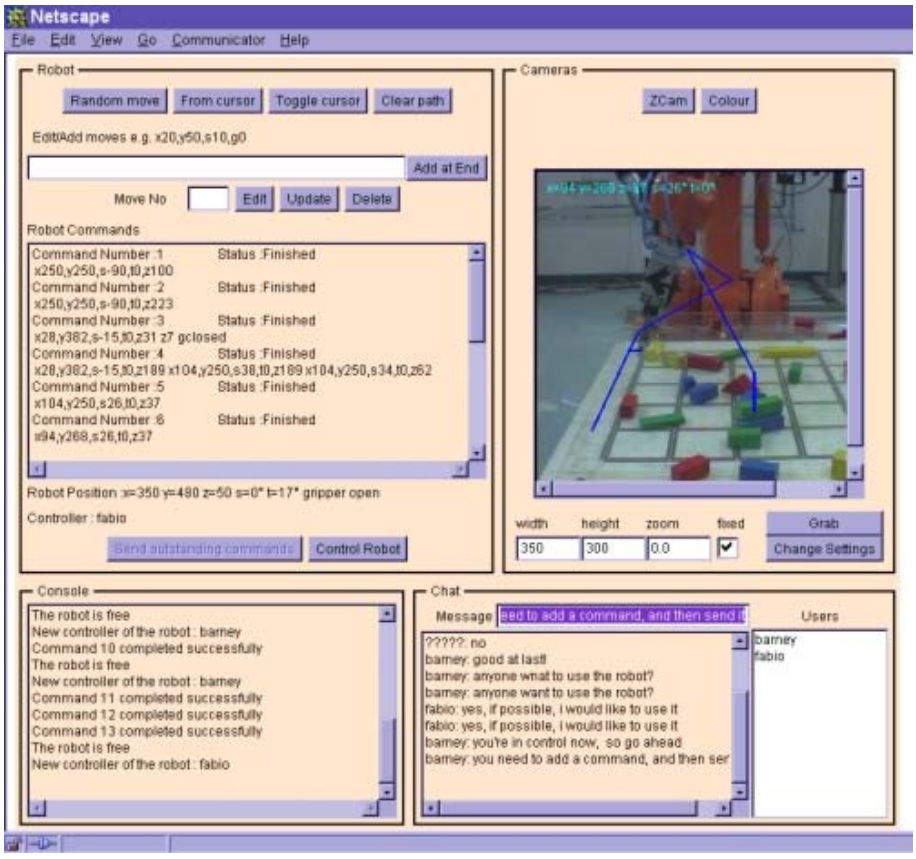

FIG. 2.15 - *L'interface de contrôle du telerobot australien*

Le Télérobot australien est accessible sur Internet depuis 1996 [TD97]. Il s'agit d'un bras manipulateur à 6 degrés de liberté qui, placé au dessus d'un table quadrillée, est capable de saisir des objets de forme et de couleurs variés afin de construire des assemblages. Au niveau technologique, la première version utilisait des scripts CGI, rapidement remplacés par l'utilisation d'applets Java offrant plus de possibilités en terme d'interface mais aussi une notion de connexion permanente à l'application serveur.

La figure 2.15 présente l'interface utilisateur. La zone intitulé "Robot" permet de saisir les ordres qui seront envoyés au bras manipulateur. La zone "Console" affiche un retour d'information après l'exécution des commandes. La zone "Chat" permet un éventuel dialogue entre utilisateurs connectés. Dans la partie "Cameras", l'internaute accède à l'une des deux caméras qui filment en temps-réel le robot. Cette interface retrace également les déplacements déjà effectués par l'utilisateur (traits bleus sur l'image vidéo). De plus, elle permet d'envoyer des ordres au robot en utilisant simplement le pointeur de la souris.

Très bien documentée, très facile d'utilisation, accessible quasiment en permanence avec des temps de réaction très faibles (inférieurs à la demi-seconde à partir du réseau de l'Université de Brest), cette expérience démontre de manière éclatante qu'il est possible d'agir à distance sur des objets lointains et de réaliser des opérations complexes.

– **Ariti :**

L'environnement Ariti a été développé par le laboratoire Cemif de l'Université d'Evry. Il s'agit d'un système à quatre degrés de liberté assurant la translation d'une plateforme dans un plan. Sur cette plateforme est installé un pic permettant de saisir des anneaux de polystyrène disposés verticalement sur des crochets (voir figure 2.16 à gauche). L'originalité et tout l'intérêt de ce travail résident dans l'utilisation de guides virtuels permettant d'aider l'utilisateur à la fois pour la saisie et la dépose des anneaux.

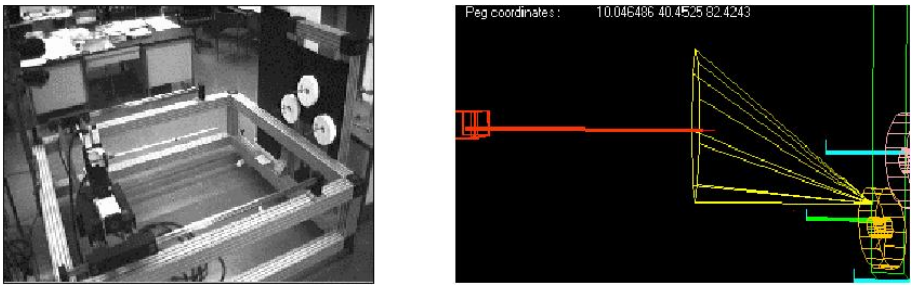

FIG. 2.16 - *Ariti : vision globale et guides virtuels*

En effet, lorsque l'on expérimente des systèmes téléopérés, le contrôle fin des outils est l'un des grands problèmes rencontrés. Les informations fournies, qu'elles soient sous forme de valeurs numériques ou d'images vidéo, ne permettent pas en général, une sensibilité suffisante pour effectuer des opérations précises. L'idée des guides virtuels est de créer des objets logiciels qui interdisent l'accès à certaines zones induisant de fait une réalisation correcte des opérations. Sur la partie droite de la figure 2.16, on peut remarquer d'une part le pic (à gauche), d'autre part les anneaux de polystyrène (à droite) et entre ces éléments un cone. Ce dernier permet de diriger le pic en un point précis d'un anneau afin de pouvoir le saisir au mieux et ainsi de pouvoir le décrocher et le repositionner par la suite. De plus, dans le cadre d'Ariti, les opérations sont d'abord effectuées virtuellement avant de l'être réellement. On parlera alors de réalité augmentée.

A partir de l'étude de ces expériences sur Internet, on peut construire une vue schématique du fonctionnement d'une application d'e-productique (figure 2.17). L'utilisateur, à travers son navigateur Internet, s'adresse à un serveur Web (étape 1) et télécharge une application sur son poste de travail comme par exemple une applet Java (étape 2). Une connexion est alors établie vers le serveur en charge de la gestion des machines cibles (étape 3) et après vérifications, l'utilisateur peut en prendre le contrôle. Parallèlement à l'étape 3, des connexions sont également établies vers des serveurs multimédia proposant des vues (vidéo, son) du système à contrôler.

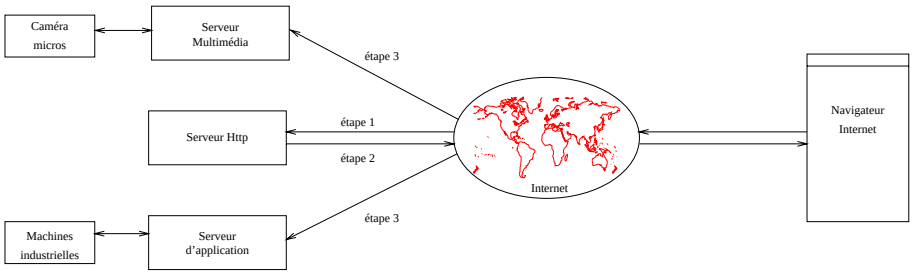

FIG. 2.17 - *Schéma générique d'un application d'e-productique*

2.2.2 Caractérisation d'un système d'e-productique

Les expériences décrites précédemment ainsi que les démonstrations que l'on peut trouver sur Internet abordent le problème de l'e-productique sous différents angles en se concentrant sur des aspects particuliers : faisabilité, interface utilisateur, aide à l'utilisateur, type de machine cible etc.. A partir de ces expériences, nous pouvons essayer de caractériser et de définir les bonnes propriétés que devrait proposer un tel système.

Un système d'e-productique est composé d'une partie cliente, accessible par l'utilisateur et d'une partie serveur, connectée au système que l'on souhaite contrôler. Ces éléments sont reliés par un système de communication qui permet d'échanger des informations sur un réseau, et dans notre cas, sur l'Internet. Plus généralement, plusieurs utilisateurs doivent pouvoir être connectés simultanément et doivent pouvoir prendre le contrôle de plusieurs machines.

Les fonctions que doit mettre en oeuvre un tel système sont les suivantes :

- Interface utilisateur:
  - Fonctions de base : la conception de l'Interface Homme Machine (IHM) un point clé dans la réussite ou l'échec de tout projet informatique [Cou90]. Dans le cadre de l'e-productique, elle doit être à la fois simple, intuitive et ergonomique et proposer au minimum la possibilité d'une part, de visualiser l'état du procédé que l'on souhaite contrôler et d'autre part, d'autoriser l'envoi d'ordres vers ce dernier. Concernant le premier point, l'état du procédé est obtenu par la fourniture de messages, l'affichage des valeurs numériques des capteurs ou à l'aide de retours d'informations multimédia (images vidéo et sons). Ces informations peuvent être brutes ou transformées au niveau du serveur. De telles transformations permettent alors de limiter le trafic sur le réseau et de faire ressortir d'éventuels problèmes ou pannes.

    L'envoi des commandes doit pouvoir se faire soit à partir de boutons présents dans l'IHM, de saisies au clavier ou bien à l'aide de la souris. Les ordres doivent être non ambigus et identiques à ceux que pourrait effectuer un opérateur qui serait en face à face avec la machine.

    Une contrainte technologique assez forte est l'utilisation des navigateurs Internet standard comme plateforme d'accueil de l'interface utilisateur. Elle est imposée par la volonté de pouvoir opérer depuis le plus grand nombre de systèmes sans nécessairement devoir installer des outils spécifiques. Cette contrainte est malheureusement limitante dans le choix des technologies et pose dans certain cas le problème de compatibilité entre navigateurs Internet.
  - Fonctions évoluées : deux exemples ont été présentés précédemment avec les guides d'Ariti et l'affichage des déplacements du Telerobot. Il s'agit à chaque fois d'enrichir l'information brute avec des informations virtuelles (réalité augmentée). Ces aides permettent à l'opérateur de mieux anticiper les mouvements et de garder une traces des

opérations effectuées. Ce domaine a largement été exploré dans la cadre d'applications de télé-robotique [EM88][Eve00] et les résultats obtenus devront être intégrés aux plateformes d'e-productique actuelles [HHV+00].

Une autre voie de recherche, mais semble-t-il peu explorée dans le domaine de l'e-productique, serait l'utilisation de la réalité virtuelle : elle permettrait d'obtenir une représentation en trois dimensions des scènes, autorisant l'opérateur à visualiser son travail non plus sous des angles prédéfinis de caméras ou grâce aux valeurs des capteurs [LPQC+99]. De plus, les ordres envoyés par l'opérateur pourraient être joués par le système virtuel en avance afin de pouvoir anticiper de mauvaises manipulations. L'utilisation de techniques d'affichage telle que VRML, permettent tout à fait d'utiliser la réalité virtuelle à l'intérieur de navigateur Internet [Gin04].

– Transport des messages

Les applications d'e-productique étant par nature réparties sur un réseau à grande échelle, elles reposent sur une couche logicielle permettant l'échange des messages entre les clients et le(s) serveur(s). Différents modèles et technologies peuvent alors être mis en oeuvre. Parmi les plus classiques, les modèles de type client-serveur [Com01] avec l'utilisation par exemple des sockets, les modèles de type objets répartis basés sur des ORB[6] tels que Corba ou les modèles de type composants communicants tels qu'on les trouve dans l'environnement .Net ou dans les serveurs J2EE[7] avec les EJB[8] [ICA03].

Quel que soit le choix retenu, le système de communication sous-jacent à l'application doit être le plus transparent possible pour le concepteur des applications, avec l'utilisation de messages de haut niveau et d'interfaces standard d'envoi et de réception. Côté utilisateur, le support de communication ne doit pas entraîner de délai de communication supplémentaire (autres que ceux inhérent spécifiquement au réseau utilisé), doit garantir l'acheminement des messages et leur intégrité. Il doit en outre être robuste.

– Prise en compte des caractéristiques réseau

L'Internet est un gigantesque réseau informatique interconnectant des centaines de milliers d'utilisateurs et de machines. Lorsque l'on émet un message sur cette toile mondiale, celui-ci est d'abord traité par la machine locale (au niveau du système d'exploitation), puis il est transmis à une succession de routeurs de marques, types et comportements différents. Il passe ainsi de machine en machine jusqu'à la machine destinataire et est enfin remis. L'acheminement nécessitera l'utilisation d'algorithme de résolution de noms, de routage voire de compression ou de cryptage. De plus ce message va se trouver confronté au trafic généré par d'autres utilisateurs et il peut être perdu, réémis, mis en file d'attente, routé par un chemin court ou long etc..

Actuellement, il n'existe pas de modèle de l'Internet [FP01] permettant de prédire le temps d'acheminement d'un message, malgré de nombreuses travaux dans ce domaine [FBTZ02]. Dans le cas général, on parlera donc d'un réseau sans qualité de service (QoS en anglais, Quality of Service). Contrairement donc aux réseaux industriels, qui eux peuvent garantir des temps d'acheminenent et de gigue, l'Internet doit donc être considéré comme une boite noire dans laquelle un message entre et peut ressortir. Cette absence de QoS peut apparaître comme incompatible avec la commande à distance de machines industrielles. C'est elle qui explique l'échec des essais de commande dans le cas 3 de la classification de Sheridan.

Néanmoins, si l'on se place dans un mode de type supervision active, le terme "active" signifiant que l'on est capable de transmettre des commandes au système, les expériences réalisées ont montré que le contrôle était possible. En effet, certains protocoles de communication comme TCP/IP garantissent l'acheminement des messages et grâce à des algorithmes

6. Object Request Broker.
7. Java to Enterprise Edition.
8. Enterprise Java Beans.

spécifiques, un flux relativement régulier des informations (gigue faible) et des délais de communication aller-retour relativement réduits (350 ms par exemple entre la France et le Japon).

Ces constatations ne doivent par faire oublier que, néanmoins, la communication sur le réseau Internet n'est pas fiable à 100% et que la qualité du réseau peut varier rapidement. Il faut donc, lorsque l'on construit une application d'e-productique avoir ces caractéristiques en mémoire. De nombreux travaux tendent à augmenter la QoS du réseau Internet à l'aide de différentes techniques. On peut citer la définition des protocoles RTP et RTCP [SCFJ03], des possibilités de reservation de bande passante avec RSVP [Whi97] ou IPv6 ou de l'amélioration des algorithmes de routage à l'aide de MPLS par exemple [DC02]. Notre travail voulant ce placer dans l'Internet au sens large, nous ne considérerons donc pas ces techniques et proposerons deux réponses permettant une prise en complet de la la QoS du réseau : l'utilisation de capteurs réseau pour mesurer en permanence l'état des connexions et la prise en compte de l'état du réseau dans le cadre d'un modèle de modes de marche.

– Gestion des utilisateurs, authentification, sécurité

  La gestion des utilisateurs est dépendante du domaine d'applications visé. En particulier dans le cas d'utilisation en maintenance ou expertise, le nombre potentiel d'utilisateurs est limité et, les applications étant de type industriel, des services d'authentification (nom et mot de passe) doivent être mis en place entre autre pour assurer la sécurité su système [SHS02]. De plus, on peut imaginer de restreindre la gamme des ordres autorisés en fonction des capacités et des compétences des utilisateurs. En cas d'accès simultanés des mécanismes de priorité doivent être mis en place en fonction de critères spécifiques, incluant entre autre la qualité du réseau de chacun des utilisateurs en concurrence.

  Dans le domaine de l'enseignement, l'authentification est nécessaire en cas d'utilisation asynchrone (auto-formation) avec des contraintes de durée ou d'objectifs à atteindre. Dans le cas d'un enseignement synchrone, élèves en présence d'enseignant, on pourrait souhaiter que l'enseignant puisse donner la main aux élèves et avoir un droit de préemption permanent. D'autres politiques peuvent également être mises en place pour permettre de s'adapter à des cas spécifiques, comme par exemple une file d'attente avec durées d'accès limitées.

  Outre la gestion de l'accès au système, il est bon de tenir à jour un journal des opérations effectuées afin de pouvoir rejouer des séquences d'ordres, mais également de savoir qui fait ou a fait quoi sur le système.

– Dialogue avec les systèmes contrôlés

  Le dialogue direct avec les machines est réalisé à partir du serveur d'application. Une liaison physique est donc assurée entre ces éléments, par exemple par l'intermédiaire d'une liaison série ou d'un bus de terrain. On considère généralement que cette liaison est fiable afin de ne pas compliquer outre mesure le système.

  Le serveur doit assurer à la fois la communication des ordres utilisateurs et de l'état du système. Il se charge également de la gestion des phases de démarrage et d'arrêt ainsi que du traitement des pannes éventuelles. En particulier, on peut considérer qu'il doit participer à la sécurité de fonctionnement du système, afin de ne pas provoquer de dégâts tant humains que matériels.

  D'un point de vue logiciel, il s'agit de construire un "pilote" de la machine visée. Celui-ci est très dépendant des caractéristiques de la machine et en particulier de son degré d'autonomie. Un même serveur doit pouvoir se charger de la gestion de plusieurs machines simultanément et doit pouvoir être facilement configurable, c'est-à-dire que l'on doit pouvoir à un instant donné accéder à un ensemble de machines puis y ajouter ou y enlever une ou plusieurs machines.

– Généricité des applications

C'est également un point relativement peu évoqué dans la littérature du domaine. Les applications sont conçues de manière ad hoc en fonction des systèmes à contrôler. Elles sont donc assez spécifiques et peu réutilisables alors que des services comme l'authentification, la gestion des utilisateurs, l'envoi et la réception de message, la surveillance du réseau sont des services communs à toute application. Les technologies comme XML ou comme MMS pourront également être utilisés afin de tendre vers une standardisation des messages échangés entres les différentes parties des applications.

Un système d'e-productique doit donc être conçu comme un noyau de services autour duquel gravitent des processus spécifiques aux machines commandées. Un premier ensemble de processus se situera du côté du serveur et assurera l'interface entre le serveur et la machine, et un deuxième ensemble, côté utilisateur assurera le dialogue avec l'opérateur [OLPVM00b].

– Documentation

Comme dans tout projet informatique, la documentation est une nécessité souvent négligée. Dans le cadre de l'e-productique elle est d'autant plus fondamentale, que l'opérateur agit à distance et ne peut en général pas disposer de manuels sous forme papier ni de contact direct avec d'autres personnes (auto-apprentissage, décalages horaires, langues...). Celle-ci se trouvera tout naturellement accessible sur l'Internet et outre les aspects propres à la commande devra expliquer les tenants et aboutissants d'un tel contrôle distant. A terme, les documentations devront pouvoir être intégrées dans des environnements tels que ceux proposés par le Réseau Universitaire des Centres d'Autoformation(RUCA) [RUC] ou dans des plateformes d'enseignement à distance type Rearsite [REA] ou WebCT [Web].

2.2.3 Apports dans le domaine

Comme indiqué dans la section précédente, un système d'e-productique doit rassembler de nombreux composants et de nombreuses technologies dans des domaines de compétences variés : interfaces homme machine, réalité virtuelle ou augmentée, réseau, développement de logiciels répartis, langages industriels, contrôle-commande, sécurité...

Dans notre travail, nous n'avons pu aborder tous ces points. Nous nous sommes concentrés sur ceux pour lesquels nous n'avons pas ou peu trouvé de réponses dans la littérature du domaine, c'est-à-dire la prise en compte des aléas réseau, tant au niveau de la conception que du fonctionnement des applications, et de la généricité des solutions à proposer. Dans la suite, nous présenterons l'architecture logicielle, la notion de capteurs réseau et l'utilisation du modèle GEMMA pour gérer les modes de marche en cas de problèmes de communication.

2.2.3.1 Architecture logicielle

L'architecture logicielle que nous avons proposée [OLPVM00b][VLPM03a], repose sur un ensemble de modules communicants entre eux à l'aide de messages. Ces modules peuvent être classés en trois catégories (voir également la figure 2.18) :

– Modules permanents (en gris foncé sur le schéma) : au nombre de trois, *Groom*, *Connection Manager* et *Device Manager*, ils sont présents en permanence et donc chargés dès le démarrage de l'application. Le premier constitue la porte d'entrée du système et est à l'écoute des demandes d'accès en provenance de clients potentiels. Lorsqu'un client se présente, il en réfère au *Connection Manager*, qui selon la politique de contrôle définie dans le pseudo-module *Control Algorithm* autorise ou non l'entrée dans le système. Ce dernier est configurable et chargé dynamiquement lors de l'éxecution. Le module *Device Manager* permet quand à lui, l'accès aux pilotes logiciels des machines via les *Tool Interface*.

– Modules dynamiques orientés clients (en blanc sur le schéma) : lorsqu'un internaute souhaite utiliser notre système, il va tout simplement charger une page Web dans son navigateur. Celle-ci contient une applet Java composé au minimum de quatre modules : *NI Sender*,

NI Receiver, *Ponger* et *Remote Client Manager*. Les deux premiers sont spécialisés dans l'envoi et la réception des messages. Le troisième permet d'évaluer la qualité du réseau en permanence. Le dernier fournit à l'utilisateur une interface homme machine de base permettant d'une part de recevoir des messages en provenance du serveur et d'autre part d'afficher les informations en provenance du *Ponger*.

De manière quasi-symétrique, après que la demande de connexion ait été acceptée, quatre processus sont créés côté serveur afin de gérer le nouvel arrivant : le *NI Sender* et le *NI Receiver* servent à la communication (processus identiques à ceux du client), le *Pinger* travaille en relation avec le *Ponger* côté client pour l'étude du réseau et le *Local Client Manager* réalise l'interface d'une part avec le *Connection Manager* pour les phases d'arrivée, de prise de contrôle et de départ et d'autre part avec le *Device Manager* pour la transmission des ordres et le retour d'information.

– Modules dynamiques spécifiques (en gris clair sur le schéma) : Ces modules fonctionnent par couple : *Tool interface* et *Tool GUI*. Le premier peut-être vu comme le pilote de la machine que l'on souhaiter contrôler. Connecté au *Device Manager*, il reçoit les ordres et renvoie l'état des capteurs ainsi que des informations d'état. Ces informations transitent à travers de l'ensemble du système pour être affichées, par l'intermédiaire du *Tool GUI* à l'utilisateur. C'est également ce module qui lui permet d'envoyer des commandes.

  Ces couples de modules sont spécifiques à une machine donnée et chargés dynamiquement à l'exécution en fonction de l'application considérée (modification d'un fichier de configuration). Plusieurs machines peuvent donc être commandées simultanément par le même utilisateur.

D'un point de vue technique, le prototype que nous avons développé a été écrit en Java, que ce soit du côté serveur ou du côté client (applets). La communication entre modules s'effectue à l'aide de files de messages point à point ("pipe") gérés de manière FIFO. La communication entre le client et le serveur s'effectue quand à elle à l'aide de messages transportés par l'intermédiaire de sockets en utilisant le protocole TCP/IP. Les messages respectent un format standard et contiennent dans leur entête, le nom de l'expéditeur, le nom du destinataire, la date d'émission ainsi que le type du message. Les modules sont gérés par des "thread Java" et fonctionnent de manière cyclique : attente d'un message, traitement du message, attente d'un message etc.. Une attention particulière est portée à la partie traitement afin que celle-ci soit la plus courte possible pour éviter l'engorgement du processeur. On observe d'ailleurs, que les différents modules passent leur temps à attendre des messages.

Nous n'avons pas choisi, en tout cas pour cette première version, d'utiliser d'environnements de construction d'applications répartis tels que Corba, J2EE ou autre. Ils nous paraissent en effet adaptés à notre problème, mais ils sont plus complexes à manipuler et surtout ne permettent pas de déterminer le comportement exact des applications, en particulier au niveau réseau.

Concernant le choix du protocole de communication [Com01], nous avons testé RTP/RTCP, UDP et TCP avant de choisir ce dernier. Lors des expérimentations avec RTP/RTCP, nous avons été confrontés à des problèmes de configuration de routeurs qui n'autorisaient pas les flux utilisant ce protocole. Comme nous nous plaçons dans le cadre de réseaux longue distance, sans aucune intervention possible sur les relais intermédiaires, l'utilisation de RTP/RTCP ne convenait pas. A terme cette décision devra bien évidement être réévaluée. Concernant le protocole UDP, une solution a été mise en oeuvre dès 1999, mais nous l'avons pas retenue en particulier en raison de pertes de paquets constatées dès les premières utilisations au delà des limites du laboratoire. En effet, ce protocole n'offre aucune garantie sur la remise des paquet et, en cas de congestion, les trames de type UDP peuvent être tout simplement détruites au niveau des routeurs sans que ni l'émetteur, ni le destinataire ne puissent s'en rendre compte [PJSB98]. Néanmoins, UDP apparait être un choix possible dans le cas de réseaux internes à une entreprise comme cela est proposé, entre autre, dans divers travaux sur l'Ethernet industriel [Tho][Vit01]. TCP, quant à lui, nous offre une garantie de remise des messages à l'utilisateur, ce qui est important dans le cas

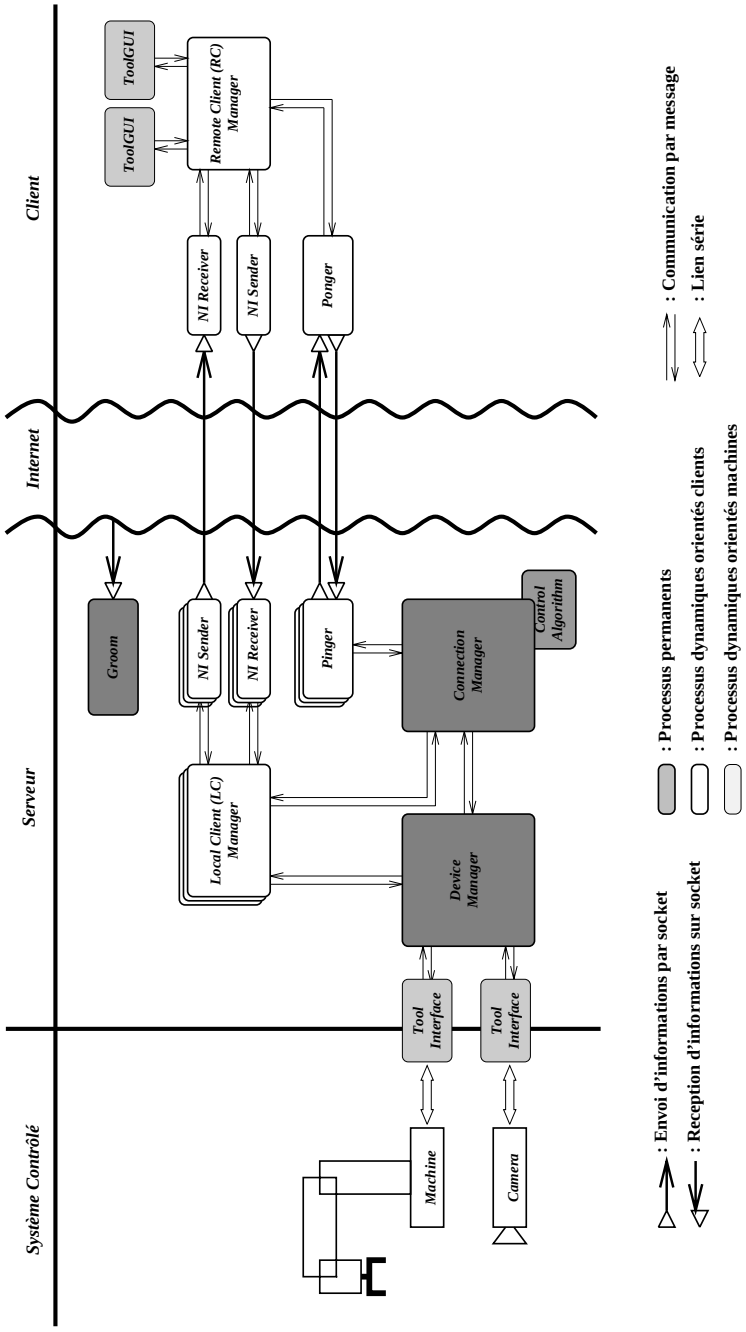

FIG. 2.18 - *Architecture logicielle Saturne*

d'une application de contrôle supervisé, même si cette fiabilité entraine des surcoûts en terme de transmission et retransmission de messages. Rappelons que nos contraintes ne sont pas de type temps réél dur et que la boucle de contrôle-commande se trouve, dans notre cas, au plus près de la machine : peu de messages sont donc échangés et le retard d'un message est moins pénalisant que sa perte dans le réseau.

2.2.3.2 Capteur réseau

Dans ce travail, le réseau Internet étant considéré comme un réseau sans qualité de service, les applications visées ayant quant à elles besoin de sécurité de fonctionnement, il est nécessaire de connaître en permanence l'état du réseau entre l'utilisateur et la machine afin de pouvoir assurer un contrôle maximal auprès des machines commandées. Une telle information est difficile à obtenir car elle met en jeu un ensemble complexe de composants permettant l'acheminement des messages. Par exemple, lorsqu'un internaute essaye d'accéder au télérobot australien depuis l'Université de Brest, pas moins de vingt-deux équipements (routeurs) sont utilisés. Ces équipements proviennent de fournisseurs différents, sont situés dans pays différents, répondent à des contraintes spécifiques. Les liens entre eux peuvent être de type paire torsadée de cuivre, satellitaire ou optique et bien sûr ils possèdent des caractéristiques (débit, gigue, bande passante) très différentes et variables. De plus un paquet acheminé à un instant t suivra un chemin donné alors qu'un autre paquet, acheminé à l'instant $t + 1$ pourra lui, suivre un chemin tout à fait différent. S'ajoute à ces paramètres, la charge du réseau qui est variable, à la fois dans le temps et l'espace.

On le voit bien, déterminer la qualité d'une connexion à un instant est une chose difficile [CLR00] et les outils disponibles, en grande majorité, travaillent soit dans des réseaux locaux, soit ne fournissent qu'une information statistique autour d'un équipement donné. Néanmoins, de nombreux travaux récents se penchent sur ce problème de métrologie de la qualité de service sur Internet [MET01][ML02][Owe03].

Comme dans le cas général il existe apparemment peu de modèles et peu de méthodes permettant d'évaluer les temps d'aller-retour d'une information, seul un système donnant une approximation est utilisable. Les processus *Pinger* et *Ponger* présentés précédemment permettent de l'obtenir. Après connexion d'un utilisateur, qu'il ait ou non le contrôle du système, un dialogue s'établit entre ces deux processus à l'aide de sockets TCP/IP : le *Pinger* envoie une demande (un *ping*) au *Ponger* qui lui répond immédiatement (par un *pong*). Le *Pinger* attend alors un temps donné puis réémet une demande vers le *Ponger* et ainsi de suite.

On utilise donc ici le même mécanisme que celui de la commande *ping*. Cette commande fait partie des commandes de supervision du protocole IP [9] et réalise des tests d'accessibilité d'une machine tout en fournissant une information de temps d'aller-retour (ou Round Time Trip - RTT) [Com01]. Nous n'avons pas choisi de l'utiliser directement, d'une part car nous souhaitons mesurer le temps d'aller-retour d'application à application et non au niveau de la couche IP et d'autre part en raison des filtrages de cette commande pour des raisons de sécurité. Néanmoins, des tests simples montrent une adéquation entre nos mesures et les temps observés par la commande *ping* ainsi qu'avec les résultats obtenus dans le projet Caida [HPMC02], qui évalue le réseau Internet par envoi massif de requètes *ping*.

Notre méthode permet deux types de mesure :

– Mesure statique : lorsque l'un des processus reçoit un message, il peut, en comparant les dates systèmes des messages émis et reçus précédemment, calculer le temps d'aller-retour du dernier échange et l'afficher. Cette information traduit donc une mesure du passé : elle peut sembler inutile ! Néanmoins, lorsque l'on regarde en détail le fonctionnement du protocole TCP/IP, on s'aperçoit que celui-ci possède des mécanismes de négociation qui permettent d'ajuster le transport des données en fonction des caractéristiques du réseau, entre l'émetteur et le récepteur. Il y a donc une certaine forme d'auto-régulation qui permet de limiter en partie la gigue propre au réseau. L'étude de nombreuses traces de communication dans

9. Plus précisément, *ping* est l'une des commandes du protocole ICMP (Internet Control Message Protocol).

le cas d'une application que nous avons déployée, montre en effet une relativement grande stabilité, en terme de délai, des échanges [LPVM04c] (voir également section 2.2.5).

A titre d'exemple, la figure 2.19 présente une trace de connexion enregistrée entre la France et le Japon par notre système de *Pinger/Ponger*. Elle a une durée de 1400 s et une valeur de RTT moyen de 349 ms. On remarquera que la plupart des valeurs sont comprises entre les deux lignes en pointillé +30% et -30% de la valeur moyenne.

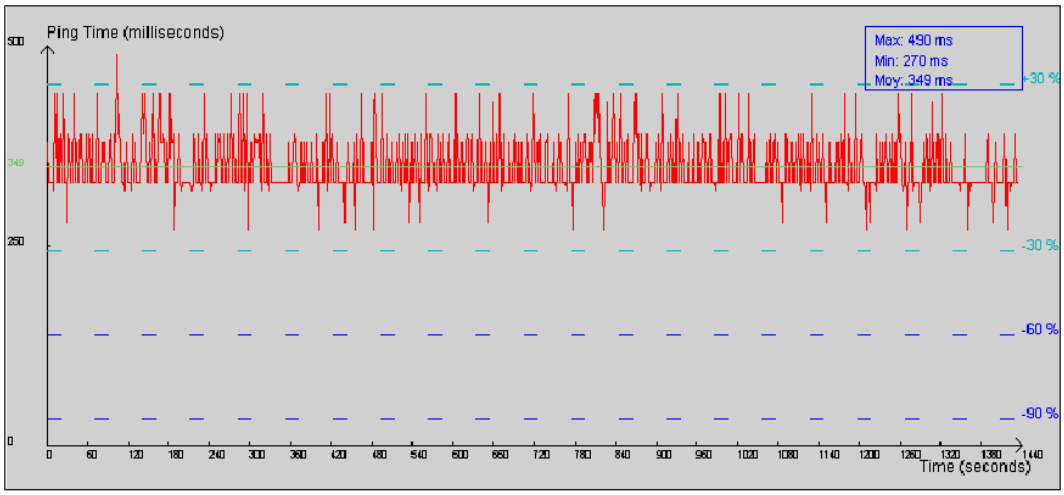

FIG. 2.19 - *Exemple d'une trace de connexion entre le Japon et la France*

Pour l'utilisateur, le tracé des valeurs des *ping/pong* sous forme d'un histogramme, lui permet alors d'avoir une estimation de la qualité de la communication et de prendre ou non certaines décisions en fonction de ce paramètre. Symétriquement, le serveur peut également refuser un utilisateur s'il juge sa qualité de connexion incompatible avec le contrôle des machines.

– Mesure dynamique : d'un côté client comme côté serveur, le mécanisme du *ping/pong* permet également d'obtenir des informations dynamiques. En effet, dès lors que l'un ou l'autre envoie un message, il peut mettre en route un ou plusieurs chiens de garde (timer) qui se déclencheront en cas de dépassement de valeurs limites. Ils assurent là aussi, une sécurité supplémentaire en cas d'aléas réseau. Pour l'utilisateur, on affichera également une barre de progression qui lui permet alors de visualiser le temps qui s'écoule entre émission et réception.

2.2.3.3 Modes de marche et d'arrêt

La notion de capteur réseau nous permet de caractériser la qualité d'une connexion de manière qualitative, plutôt que quantitative. Elle dépend, de plus, du type d'application visée et de son mode de fonctionnement : un temps d'aller-retour donné pourra être considéré comme bon dans le cas de machine à temps de réaction lent et mauvais pour des traitements rapides. Selon la qualité de communication, on peut également considérer des modes de fonctionnement différents. Dans un cas favorable, l'ensemble des commandes pourra être autorisé et les mouvements pourront s'effectuer à vitesse maximale. Dans un cas défavorable, on devra limiter certains ordres ou demander à l'utilisateur de respecter certaines contraintes. On voit ici se dessiner la notion de modes de fonctionnement reliés à la qualité du réseau.

Dans le domaine des automatismes, lorsque l'on conçoit une application, des notions similaires de modes de marche et d'arrêt existent. Elles traduisent, au premier niveau, les transitions entre les états d'arrêts (A) du système, les états de fonctionnement (F) et les états de défaillance (D). Un document de spécification normalisé a été défini vers la fin des années 70 pour prendre en compte ces notions : le GEMMA, acronyme de Guide d'Etude des Modes de Marche et d'Arrêt [ADE81][PM02] (voir figure 2.20).

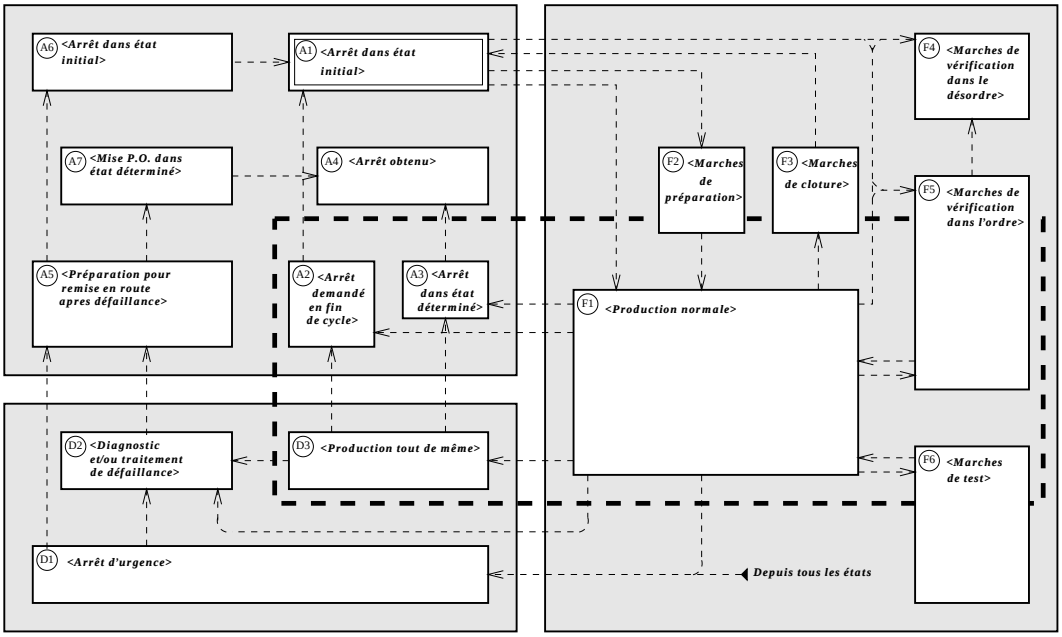

FIG. 2.20 - *Feuille de GEMMA*

Chaque grande catégorie (A, F, D) est subdivisée en sous-catégories traduisant différents modes de fonctionnement, comme par exemple des marches de test (case F6), des marches de remise en route après défaillance (case A5) ou des marches de diagnostic (case D2). Des transitions entre ces états doivent être instanciées afin de spécifier les conditions permettant le passage de l'un à l'autre. On notera un état particulier, D1, accessible depuis tous les états et modélisant l'arrêt d'urgence.

Pour prendre en compte des défauts de communication, il serait tentant de rajouter soit des transitions vers l'état D1 conditionnées par les valeurs du capteur réseau, soit d'ajouter spécifiquement un état D4 portant, par exemple, comme intitulé "Perte de connexion réseau" vers lequel, depuis tous les états, le système devrait évoluer en cas de problème de communication. Ces deux visions sont trop réductrices car elles ne font ressortir alors que deux états possibles pour la communication, bonne ou mauvaise.

Dans sa thèse [Ogo01], Pascal Ogor a proposé de subdiviser chaque case du GEMMA en autant de sous-cases que de niveaux de qualité de communication nécessaires, et ainsi de créer un GEMMA QoS. Chaque sous-case traduisant alors un mode de marche dans une qualité de communication donnée. Une proposition basée sur six niveaux (cinq + un) a été faite :

- Q1 : qualité "normale" : le contrôle exercé par l'opérateur est total et toutes les opérations sont autorisées.

- Q2 : qualité "détériorée sans perte de contrôle" : la communication n'est plus optimale, des contraintes peuvent être appliquées.

- Q3 : qualité "détériorée avec perte de contrôle" : dans ce mode, il serait souhaitable, soit que l'utilisateur arrête de lui-même son travail, soit que le contrôle, dans le cas d'une utilisation multi-utilisateurs, soit transmis vers un autre opérateur.
- Q4 : qualité "marche vers l'arrêt" : la qualité est alors trop faible pour que l'utilisateur continue à travailler. Il peut néanmoins être autorisé à effectuer certains ordres dans le but de mettre la machine en position de sécurité afin de faciliter une éventuelle reprise ultérieure du travail.
- Q5 : qualité "arrêt d'urgence" : la communication est rompue, le système doit de lui-même mettre en position de sécurité l'ensemble des machines contrôlée.
- Qz ; qualité "zéro utilisateur" : dans certains cas, comme par exemple des phases de prototypage rapide, un utilisateur peut vouloir lancer un processus de traitement long et ne pas rester connecté en permanence, mais venir contrôler par intermittence le bon déroulement des opérations.

Un GEMMA complet comporte 16 cases, chaque case pouvant instancier 6 niveaux de qualité réseau, on obtient potentiellement un GEMMA à 96 cases. Des propositions ont été faites pour réduire ce nombre, en fonction du type de comportement que les cases du Gemma décrivent. La figure 2.21 en présente une représentation en trois dimensions (par souci de simplification graphique, les liens reliant les états ne sont pas représentés).

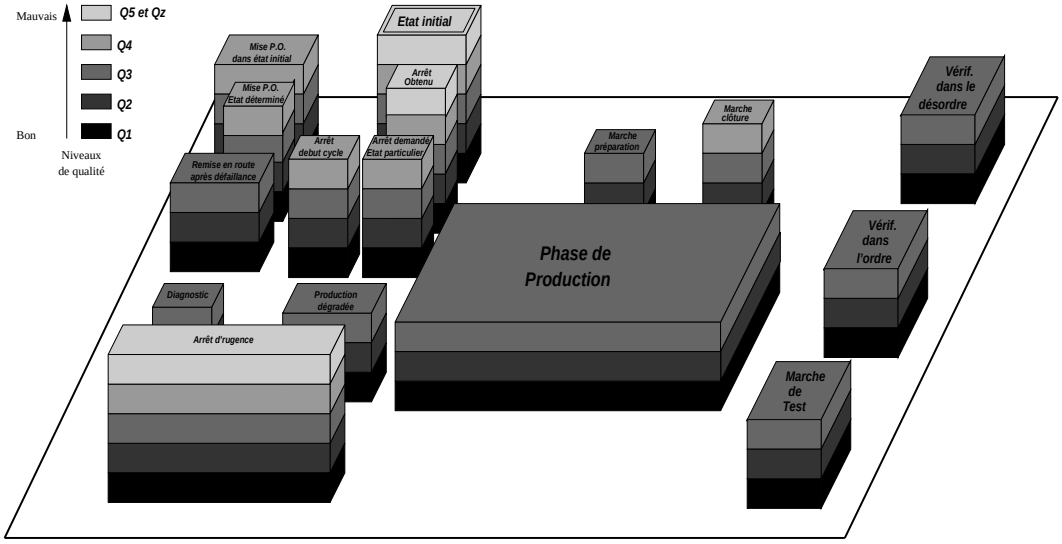

FIG. 2.21 - *Représentation 3D du GEMMA*

Pour chaque type d'application, pour chaque case du modèle GEMMA QoS, il est alors nécessaire de définir les sous-cases utiles, sachant qu'il n'est pas obligatoire d'instancier tous les niveaux de qualité. De plus il est nécessaire de définir les limites entre les différents niveaux en fonction des temps d'aller et retour et des performances attendues du systèmes à contrôler.

Des exemples concrets ont été construits en particulier pour le pilotage d'une caméra orientable ainsi que pour celui d'un bras robotisé (voir section 2.2.4). Le GEMMA est vu comme une machine à états finis et implémenté à l'aide du patron de conception orienté objet "State" contenant d'une part une définition des états et d'autre part un gestionnaire d'états. Pour chaque case, on instanciera un objet état en définissant une méthode "initialisation", une méthode "terminaison", une méthode "changement d'état" et une méthode "action". La méthode "changement

d'état" permet d'évaluer si l'on doit où non transiter à partir de l'état courant en fonction des valeurs associées aux transitions (valeurs des capteurs du procédé, pupitre utilisateur local ou distant, connexion/deconnexion des utilisateurs, qualité du réseau). La méthode "action" définit les opérations à effectuer dans l'état considéré. Le gestionnaire d'état permet quant à lui, l'activation des états et le franchissement des transitions.

L'objectif du GEMMA étant d'assurer la commande de la partie opérative, celui-ci est situé au plus près de la machine, c'est-à-dire inclus dans les processus *Tool interface* décrits précédement.

Ces exemples ont permis de montrer l'intérêt de cette approche, en particulier au niveau de la conception, car elle oblige le concepteur à réfléchir sur les comportements attendus de la machine en fonction des paramètres du réseau.

Néanmoins, il n'existe aujourd'hui pas d'outil automatique permettant de passer d'une spécification GEMMA (standard) à son implantation dans un calculateur. Par exemple dans le cadre d'une programmation en Grafcet d'un automatisme, le GEMMA est traduit en un grafcet de haut niveau capable de réaliser des actions de forçage/figeage (préemption) sur les grafcets correspondant aux divers modes de marche. Un grafcet spécifique est quand à lui construit pour traiter les cas d'arrêt d'urgence. Pour une implantation effective et automatisée du GEMMA QoS une telle démarche sera formalisée et mise en place. Elle définira les règles d'évolution entre les différents états du modèle en fonction des valeurs et des éventuelles priorités des transitions.

2.2.3.4 Synthèse

Dans notre travail, nous avons proposé une architecture logicielle pour l'e-productique : SATURNE pour "Software Architecture for Teleoperation over an UnpRedictable NEtwork". Elle repose sur un noyau de services de communication autour duquel gravitent des modules spécifiques aux machines visées. La prise en compte des aléas du réseau est réalisée à l'aide d'un capteur logiciel affichant ses informations, sous une forme statique et dynamique à l'utilisateur, et dont les valeurs peuvent être utilisées pour une gestion de type mode de marche.

Différentes applications ont été réalisées et sont présentées dans la suite de ce document.

2.2.4 Applications réalisées au laboratoire

Plusieurs applications ont été construites au laboratoire autour de SATURNE. Elles représentaient pour nous, à la fois un moyen de valider nos idées et de disposer de plateformes de démonstrations et de tests distantes.

La première machine pilotée a été une caméra de type SONY EVID-31 [LPOVM99]. Equipée d'un socle motorisé, elle peut effectuer des mouvements de rotation horizontale sur 200° et verticale sur 120°. Elle est de plus pourvue d'une focale variable. Dès 1999, elle a été mise en oeuvre et rendue accessible sur l'Internet, nous permettant alors de valider nos idées. Cette première expérience, même si elle ne s'applique pas à une machine de type industriel était nécessaire à la fois sur un plan technique, mais aussi car elle nous permettait de mettre en oeuvre une solution offrant un retour d'informations vidéo. L'applet de contrôle contient une image statique de la scène. Elle est cliquable et permet donc l'orientation de la caméra en fonction de ce que l'utilisateur souhaite regarder. Un ascenseur, en partie gauche permet de régler le facteur de zoom et des boutons prédéfinis peuvent permettre la visualisation de lieux précis. Aujourd'hui des solutions clés en main de ce type sont proposés par les fabricants de caméras, mais n'intègrent pas, en tous cas au niveau utilisateur, d'informations de qualité de service (mesure du RTT).

Nous nous sommes ensuite intéressés à la commande d'un bras manipulateur à 5 degrés de liberté de type Ericc permettant la saisie et le déplacement de petits objets. C'est le type même de robot d'apprentissage. Il est connecté au serveur par une liaison série RS232 et ses mouvements sont filmés en permanence par deux caméras. La première, de type EVID-31, donc motorisée et à focale variable, est placée en face du robot. La deuxième (de type AXIS 211) fournit une

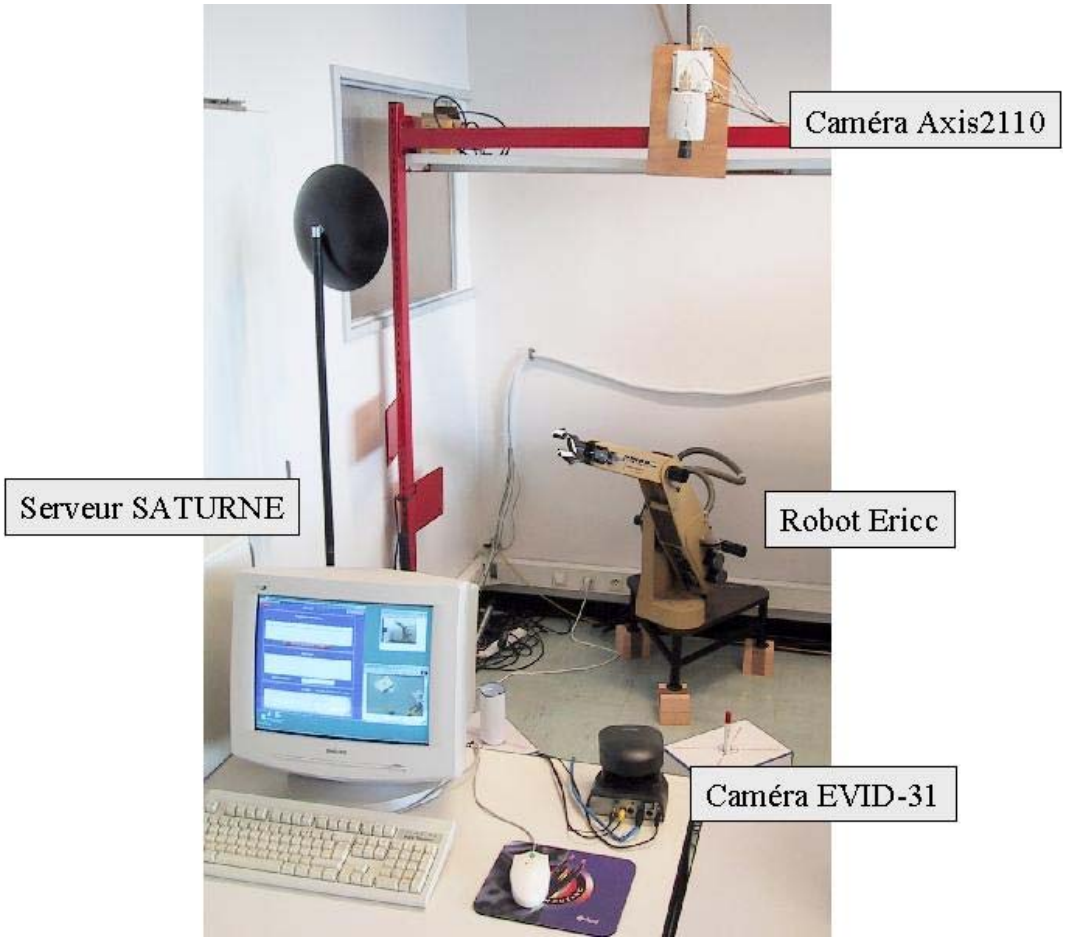

FIG. 2.22 - *Le robot Ericc et son environnement*

image vue du dessus. Elle est fixe, n'offre pas de possibilité de zoom mais contient un serveur Web intégré réalisant l'émission des images vidéo. Un module sonore, lié à la caméra AXIS est également présent. La photo 2.22 présente l'ensemble du système.

L'interface graphique utilisateur (voir figure 2.23) est composée de 4 grandes zones. En partie gauche, l'utilisateur accède aux images fournies en temps réel par la caméra Sony. La zone supérieure de la partie droite permet l'affichage des messages ainsi que des informations en provenance du capteur réseau. En dessous, se trouve la zone spécifique pour le contrôle du robot Ericc, puis la zone de contrôle de la caméra, qui est la reprise intégrale du travail présenté précédemment. On utilise ici le mécanisme de chargement dynamique et la notion de modules spécifiques à une machine donnée.

L'interface de contrôle du bras manipulateur propose deux modes de fonctionnement :

- Minitel : il correspond exactement à l'interface dont pourrait disposer un utilisateur local, avec entre autres un envoi caractère par caractère des ordres. De plus, deux boutons permettent d'envoyer un ordre d'arrêt simple ou d'arrêt d'urgence.
- Programme : il permet de saisir un programme complet (ou d'en copier un à partir d'une bibliothèque), de l'envoyer au robot Ericc avant de lancer son exécution.

Dans les deux cas, toutes les informations retournées par le bras manipulateur sont affichées

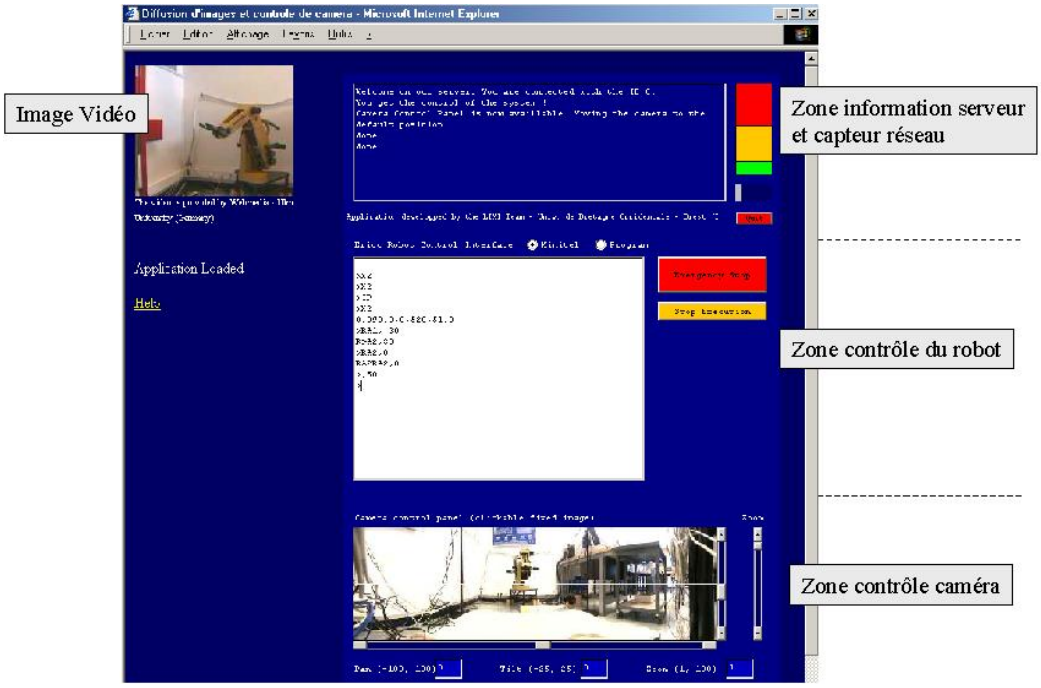

FIG. 2.23 - *Interface utilisateur pour le robot Ericc et la caméra EVID-31*

à l'utilisateur.

Un Gemma complet a été construit pour cet exemple. Il permet de modéliser le mode "Minitel" comme un mode de test (F4) et le mode "Programme" comme un mode de production normale (F1). Une prise de contrôle par un opérateur local entraîne la perte de contrôle de l'utilisateur distant et une transition vers l'état F6. Trois niveaux de qualité réseau ont été définis : Q1, Q2 et Q5. Une transition, en mode test, de la qualité Q1 vers la qualité Q2 entraine automatiquement une réduction de la vitesse de déplacement du bras. Une transition, en mode production normale, de Q1 vers Q2, interdit à l'utilisateur toute exécution de programme. Depuis tous les états, une transition vers Q5 entrane un arrêt en fin de cycle. L'arrêt d'urgence également été traité.

Le robot Ericc a été ainsi contrôlé à distance lors de diverses présentations scientifiques, au cours de séquences d'enseignement que ce soit au niveau local, national et international. Les délais observés, en général bien inférieurs à la demi-seconde entre l'envoi de l'ordre et le retour vidéo, ont permis une manipulation aisée et la réalisation de mouvements et de saisie d'objets.

Suivant les même principes, deux autres expérimentations ont été réalisées. La première autour d'un robot du type Khepera et la deuxième pour une machine de prototypage rapide de type ISEL. Dans les deux cas, seuls les processus spécifiques, *Tool Interface* et *Tool GUI* ont du être créés. Pour Khepera, moins d'une demi-journée de travail a été nécessaire afin d'obtenir une première version. La machine Isel, plus complexe, a nécessité cinq jours de travail à un étudiant qui, à son arrivée, ne connaissait rien de l'architecture SATURNE.

2.2.5 Projet Océanopolis et résultats sur la mesure du réseau

Initié en 1999 et en fonctionnement opérationnel depuis août 2001, le projet Océanopolis [LOIP01] a consisté à la mise en place de caméras, accessibles depuis l'Internet (connexion ADSL - 2048 Kbits descendants et 512 kbits montants), dans une manchotière et dans un aquarium (requins puis poissons tropicaux). Ce travail fait suite tout naturellement aux travaux effectués

autour des caméras SONY EVID-31, puisque qu'il y a eu réutilisation complète de la solution logicielle mise en oeuvre au laboratoire. Ce travail a été financé, au niveau matériel, par le Conseil Général du Finistère. Il a également reçu le soutien de la société Irvi-Progéneris qui a conçu l'ergonomie et le rendu graphique du site Web et a fait l'objet, ainsi que tout l'environnement Saturne, d'un dépôt de logiciel.

Pour les responsables d'Océanopolis, la mise en place de caméras filmant soit des manchots, soit des poissons présentait un triple intérêt :

- Dynamisation du site Web : la possibilité de visualiser des animaux en temps réel à l'aide d'une caméra pilotable en horizontal, vertical et focale permet d'offrir aux futurs visiteurs une partie du spectacle présent à Océanopolis. A l'époque de l'installation et d'ailleurs encore aujourd'hui, peu d'établissements de ce type permettaient de telles visites.

- Surveillance des animaux : les caméras sont également utilisées en interne pour la surveillance et le suivi des animaux, en particulier en période de ponte chez les manchots, les parents des nouveau-nés n'aimant pas être dérangés par les soigneurs.

- Activités pédagogiques : le système fonctionnant sur Internet, il est possible de faire visiter à un groupe distant, possédant un accès Web, les aquariums. Dans le cadre d'animations spécifiques, nous avons ainsi pu faire découvrir les manchots à des classes de primaire avec la participation d'un animateur auprès des animaux pouvant commenter et répondre aux questions (par le biais d'une liaison téléphonique).

La mise en place de ce système nous a également permis d'expérimenter notre logiciel en grandeur réelle. En effet, en deux ans, plus de 17 907 connexions ont été enregistrées en provenance du monde entier. Dans 93,68% des cas, les internautes ont effectué des mouvements de caméra. La date, l'origine, la durée, le nombre de d'ordres envoyés de chaque connexion ont été enregistrées, ainsi que le nombre et la valeur des mesures du capteur réseau. Une base de données est périodiquement enrichie avec les nouvelles valeurs et les données sont analysées à l'aide d'un logiciel conçu au laboratoire. Les statistiques présentées dans la suite de ce document ne se basent pas sur l'ensemble des connexions, mais sur celles ayant une durée minimum de 60s, soit 12 530 connexions, d'août 2001 à février 2004.

Les premiers retours de cette expérience sont les suivants :

- Fiabilité : le logiciel Saturne nous a donné entièrement satisfaction et seule une erreur minime au niveau de la gestion des utilisateurs a été mise en valeur au cours de ces deux années et demie de test. Par contre, l'environnement étant agressif (froid et humide ou sous-marin), des problèmes matériels ont affecté à la fois les caméras et les ordinateurs ce qui ne nous a pas permis d'obtenir une disponibilité à 100% du système. Nous avons également rencontré des problèmes lors des redémarrages programmés du PC : difficulté de reconnexion ADSL, pannes du serveur vidéo. Ces problèmes montrent bien que derrière l'aspect purement de conception d'une architecture logicielle pour l'e-productique, des techniques de tolérance aux fautes, de redondance doivent être mises en place. Le choix des environnements de déploiement, des logiciels connexes (serveur web, serveur multimédia...), des fournisseurs d'accès doit donc être réalisé avec le plus grand soin afin d'obtenir un ensemble efficace et efficient.

- Visibilité du site : il a été visité en majorité par des internautes français, qui représentent 8493 visites. Ensuite viennent les internautes néerlandais (1298 visites) et belges (804 visites). Les autres connexions (1935) sont issues du monde entier. Le site d'Océanopolis étant uniquement en langue française, les aquariums étant non éclairés la nuit, elles sont peu nombreuses en provenance de grands pays ayant un fort décalage horaire avec le nôtre.

- Ping/pong moyen : les données recueillies nous ont permis d'estimer des temps de ping/pong moyen. Ils sont indiqués dans le tableau 2.1:

	Global	France	Hors France	Modem 56k	Adsl
Nombre de connexions	12530	8493	4037	1468	2567
Durée Moyenne (s)	240	247	225	214	250
Ping/pong moyen (ms)	519	473	602	1023	214
Ecart type (ms)	140	140	35	669	29

TAB. 2.1 - *Mesures de temps d'aller-retour*

Ces résultats montrent, qu'en moyenne, le temps d'aller-retour d'une information entre l'internaute et le serveur est d'environ une demi seconde. Cela signifie, en moyenne toujours, que lorsqu'un utilisateur envoie une commande, celle-ci est reçu en environ 250 ms[10] et transmise immédiatement vers la partie opérative. Si l'on compare ces résultats à la situation d'un opérateur dans un atelier, ayant une vitesse de déplacement moyenne d'un mètre par seconde, tout opérateur se situant à plus de 25 cm sera plus lent que l'opérateur distant. Bien évidement, ce calcul ne prend pas en compte le temps nécessaire (à nouveau 250 ms) pour obtenir l'information qui va provoquer l'action de l'opérateur distant, mais néanmoins, ces temps moyens d'aller-retour constatés montrent les potentialités de réactivité de l'internaute.

Les deux colonnes de droite du tableau illustrent que les valeurs de ping/pong sont bien entendues très dépendantes du type de connexion utilisée. La colonne "Modem 56k" présente les résultats obtenus en regardant les traces en provenance d'utilisateurs de modem 56 Kbits. Le temps moyen de RTT est de l'ordre de la seconde. Comparativement, les possesseurs de connexion Adsl, ont une réactivité moyenne de 214 milli-secondes soit près de 5 fois plus rapide ! L'Adsl étant une technologie aujourd'hui largement déployée à un coût raisonnable, l'e-productique "réactive" devient donc facilement accessible.

On notera une dispersion des résultats pour les connexions par Modem avec un écart-type de 669 ms, alors que, bien que les caractéristiques des liaisons Adsl soit diverses (512 kbits, 1024 kbits ou plus), celui-ci reste faible. En effet, les messages de commande supervisé sont petits et les informations multi-media sont calibrées (période de rafraichissement, taille et qualité des images ...) afin d'éviter un engorgement des connexions.

D'autres mesures montrent également qu'il n'y a pas de rapport entre la distance physique des utilisateurs et leur distance "numérique". En effet, nous avons pu relever des temps d'aller-retour de l'ordre de 350 ms pour des utilisateurs situés au Japon ou de 420 ms pour des internautes australiens à comparer aux valeurs obtenues par exemple pour des utilisateurs français dotés d'un modem 56k.

– Stabilité des connexions :

La connaissance du ping/pong moyen n'est néanmoins pas suffisante pour tirer de grandes conclusions au niveau du fonctionnement du réseau Internet et des qualités de communication que l'on peut escompter. Il est important d'étudier également la stabilité des communications. La figure 2.24 présente ainsi deux calculs de pourcentage. L'histogramme "% ping au dessus de 100%" présente le pourcentage de mesures qui se trouvent entre 0 ms et 100% de la valeur moyenne. En moyenne, 96,64% des mesures sont dans cette bande. Si l'on s'intéresse à l'histogramme "% ping au dessus de 50%", on obtient un résultat de 91,48% des valeurs comprise entre 0 et 50% du ping moyen. La courbe "pourcentage de respect des 100%", indique que 28,8% des connexions respectent la contrainte des 100%, et la courbe de tendance polynomiale qui lui est associée indique, au cours du temps, une augmentation de cette valeur, qui traduit peut-être une amélioration globale du fonctionnement de l'Internet.

Ces résultats montrent que les mécanismes inhérents au protocole utilisé (TCP) de gestion des flux fonctionnent correctement et assurent un transport assez régulier des informations.

10. Cette valeur est une approximation, car on ne peut pas être sûr que les trames réseau prennent le même chemin à l'aller et au retour.

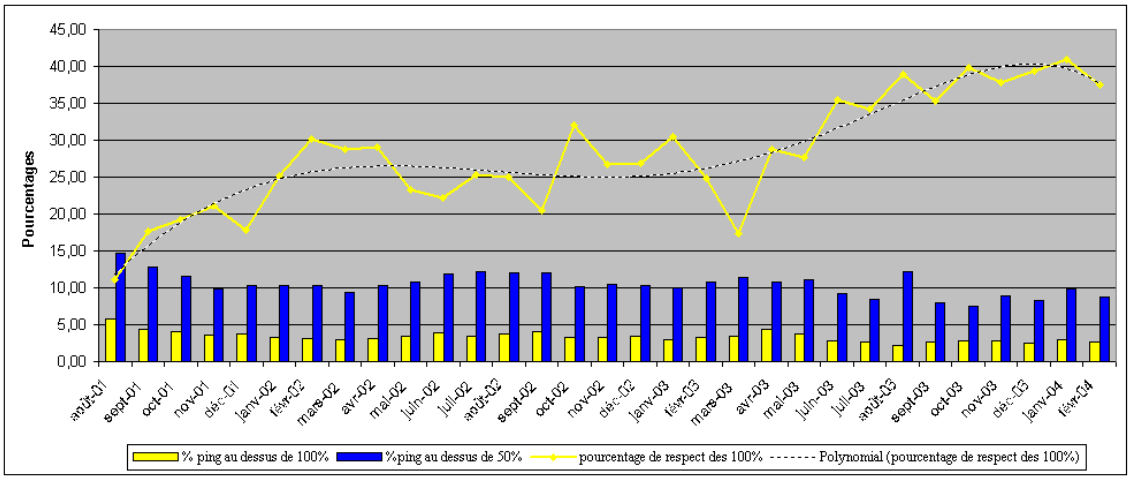

FIG. 2.24 - *Variations autour du Ping Moyen*

Ils ne doivent pas cacher néanmoins que lors de certaines connexions, malgré une bonne stabilité, des pics de *ping/pong* existent et peuvent, à certain moment faire croire à des ruptures de communication et que le protocole TCP, bien que nous garantissant certaines propriétés comme la remise des messages, reste sensible au problème d'engorgement des réseaux.

Notre objectif dans ce travail d'études des traces de connexion n'est pas de modéliser le réseau mais d'en retirer quelques notions qui peuvent nous aider dans le cadre de l'e-productique : des temps d'aller-retour tout à fait compatibles avec les applications visées et une stabilité de connexion mettant en valeur le travail sur les niveaux de qualité et l'utilisation du GEMMA QoS.

Chapitre 3

Conclusions et perspectives

3.1 Bilan de l'activité de recherche

Le travail sur la modélisation et la vérification du Grafcet ne fait plus partie aujourd'hui des thématiques de recherche de notre laboratoire. Il m'aura permis de découvrir le monde des automatismes ainsi que les modèles, méthodes et techniques qui y sont employés. J'ai pu également approfondir mes connaissances dans le domaine de la validation de programmes en étudiant différentes solutions. Plus généralement, ce travail a été l'un des premiers à essayer de faire un pont entre les langages de l'automatique et les langages synchrones et de valider des grafcets. Il a été suivi par d'autres travaux en particulier de thèse [Rou94][L'H97][Fra01] qui l'ont repris, complété et enrichi. Il a donné lieu à diverses publications scientifiques en particulier dans la revue Theoretical Computer Science.

Notre apport, dans le domaine de l'e-productique, se situe particulièrement dans la prise en compte de la qualité du réseau à travers la définition d'un capteur, la notion de GEMMA ainsi que l'affichage de zones de qualité à l'utilisateur. Cet aspect est assez peu traité dans la littérature du domaine. L'étude des connexions réseau, qui démontre que, dans certaines conditions, l'opérateur distant peut-être plus efficient que l'opérateur local, ouvre des portes très intéressantes vers de nouvelles applications, en particulier dans le domaine du télé-enseignement, et ne peut que nous encourager à poursuivre notre engagement dans cette voie.

3.2 Perspectives à moyen terme

Le travail réalisé est une première phase dans la construction d'un système d'e-productique complet. De grands chantiers restent ouverts et sont des voies possibles de recherches futures, pouvant donner lieu à d'éventuels travaux de thèse.

- Dialogue utilisateur :

  L'objectif du "dialogue utilisateur" est de fournir à ce dernier des informations suffisantes afin qu'il ait une représentation similaire à celle qu'il aurait en face du système qu'il souhaite contrôler. Actuellement, les retours d'information proposés sont basés d'une part sur des informations en provenance des capteurs proprioceptifs des systèmes mécaniques et d'autre part des informations de type multimedia (vidéos et sons). De plus l'utilisateur connait en permanence la qualité du réseau ce qui lui permet d'estimer le retard temporel entre ce qu'il perçoit et la réalité distante. Une bonne connaissance du système permet d'anticiper son évolution dans la plupart des cas, mais, en particulier en cas de dégradation de la qualité du réseau et donc d'augmentation du décalage temporel, le contrôle devient plus délicat à réaliser.

Trois grandes pistes devront être évaluées afin d'augmenter la qualité du retour d'information :

- Utilisation des techniques de réalité augmentée : appliquées entre autre sur le Telerobot australien ou dans l'environnement Ariti, ces techniques permettent d'ajouter de l'information sur les retours multimédia. Elles entrainent une meilleure compréhension des mouvements effectués et un guidage des actions de l'utilisateur à travers des zones autorisées ou non. Elles représentent donc une aide opérationelle en simplifiant le travail de l'opérateur.
- Utilisation de la réalité virtuelle : les techniques de réalité virtuelle permettent principalement l'immersion de l'utilisateur dans une scène virtuelle et son interaction avec cette dernière à l'aide d'interfaces à retour d'efforts. Dans le cas de l'e-productique, elles pourront être utilisées directement dans les phases d'apprentissage. En phase de production, un couplage fort entre le monde réél et le monde virtuel devra être réalisé, tout en tenant compte du décalage temporel, variable, introduit par le support de communication. Par certains côtés, on rejoindrait alors la réalité augmentée.
- Anticipation des mouvements : les ordres de commande étant produits par l'utilisateur, l'environnement mécanique étant connu, il est tout à fait envisageable de présenter à l'utilisateur le déroulement des actions légèrement en avance par rapport à la réalité effective. Cela autoriserait alors l'opérateur à réagir plus rapidement et de manière plus efficace, en particulier en phase de mise au point ou de tests de programmes. Cela reviendrait aussi à gommer l'écart temporel du au réseau, tout en n'oubliant pas que l'information transmise n'est qu'une projection de la réalité et que les aléas réseau sont toujours possibles.

L'utilisation de ces techniques permettrait sans nul doute une meilleure perception et donc un meilleur contrôle à l'opérateur final. Néanmoins, il est difficile aujourd'hui de réellement quantifier l'apport de chacune d'entre elles et les moyens nécessaires à leur déploiement.

- Réseaux et adaptabilité : le capteur réseau que nous avons construit est simple et efficace. Il ne donne néanmoins qu'une information très parcellaire de la qualité du réseau en ne mesurant que les temps d'aller-retour de l'information. D'autres travaux [ML02] proposent des visions étendues de cette notion de capteur réseau qui ont pour but de mieux cerner les propriétés des connexions. Associé à des outils statistiques, en particulier pour des connexions ayant des caractéristiques et des origines connues (ce qui devrait être le cas pour des applications industrielles), l'indicateur de qualité réseau fournira alors une image plus précise de la qualité réseau instantanée et à venir.

  Cette information pourra alors être fourni, d'une part au Gemma QoS pour une gestion des modes de marche liée aux paramètres réseau, mais aussi aux différentes applications utilisées dans l'environnement. En particulier, on peut imaginer un impact direct sur les flux d'information multimédia permettant d'accroitre ou de diminuer dynamiquement la qualité de retransmission fournie à l'utilisateur. L'objectif est donc de permettre une meilleure immersion de l'opérateur tout en ne pénalisant pas l'envoi des commandes.

  De plus, il serait intéressant d'estimer l'impact des nouvelles technologies en matière de réseau de communication :

  - Le déployement d'IP V6, outre l'augmentation de l'espace d'adressage, devrait permettre de mettre en oeuvre des solutions de gestion de la qualité de service sur des réseaux à grande échelle. Seront-elles suffisantes pour garantir un transport fiable et constant de l'information ?
  - Les débits des réseaux ne cessent d'augmenter, mais parallèlement les volumes échangés ne cessent de progresser. Il est difficile de prédire aujourd'hui qui va l'emporter, si nous allons donc vers une amélioration globale du fonctionnement de l'Internet ou

vers une dégradation de ses performances. Dans tous les cas, connaître la qualité de la connexion est une information majeure pour toute personne souhaitant contrôler un système distant.

- L'émergence des réseaux sans fil et le développement des réseaux de téléphonie ouvrent la voie à une mobilité plus grande des utilisateurs et l'on peut imaginer à brève échéance des applications de contrôle à distance de systèmes mécaniques à partir d'un simple téléphone portable. Ces réseaux ont des caractéristiques spécifiques dont il faudra tenir compte. Dans ce domaine aussi, il sera important d'évaluer quels types d'applications d'e-productique seront utilisables et quelles précautions devront être prises afin de garantir un contrôle à distance fiable.

- Outils de génération automatique : l'architecture logicielle que nous proposons est basée sur un noyau de processus qui sont complétés par des modules spécifiques permettant le contrôle du ou des systèmes visés. Ces processus doivent pour l'heure être écrits manuellement et représentent donc une charge de travail qui peut être importante.

  Il nous apparaît nécessaire de concevoir des outils permettant d'automatiser cette phase, c'est-à-dire des outils qui, à partir d'une description à haut niveau, seront capables de générer les codes source à la fois pour les processus de la partie cliente et pour ceux de la partie serveur. Cette modélisation devra permettre de :

  - décrire les interfaces du système visé, c'est-à-dire entre autre la liste et surtout les caractéristiques des capteurs et des actionneurs accessibles,
  - expliciter le protocole de communication entre le calculateur de la machine à contrôler et le serveur d'e-productique,
  - proposer des messages standardisés pour l'échange des informations entre les deux parties de l'application (clients - serveur),
  - construire le Gemma QoS de et l'implanter directement dans les processus correspondants,
  - décrire l'interface utilisateur au plan graphique et comportemental.

  D'ores et déjà, ce travail a été en partie engagé : le noyau est stable et les modules spécifiques construits respectent tous la même architecture. De plus les choix de programmation, en particulier au niveau du Gemma, permettent une génération simple et rapide d'une partie importante du code.

  Depuis le début de ce projet, il existe une volonté forte de généricité de l'approche afin de pouvoir s'adapter à une grande variété d'applications et d'offrir une mise en oeuvre rapide. Cette approche devra être maintenue et renforcée dans l'optique de la création d'outils de génération automatique de code et elle devra s'appuyer sur les standards existants des différents domaines.

- Validation formelle : notre environnement Saturne, et d'une manière plus générale, l'ensemble des environnements d'e-productique, met en jeu des ensembles de processus séquentiels, communicants entre eux par l'intermédiaire de messages. Certains de ces processus sont réactifs aux actions de l'utilisateurs sur l'IHM, aux compte-rendus fournis par les systèmes mécaniques ou aux messages transitant sur le réseau. Les autres sont activés à la demande en fonction des tâches à réaliser. Côté serveur, comme côté utilisateur, le transport des messages peut être vu comme instantané alors qu'entre les deux parties de l'application, le temps de transport est plus ou moins long et en tous les cas inconstant. Ces environnements sont donc complexes.

  L'un de nos objectifs est de fournir un environnement d'e-productique sûr. L'application déployée à Océanopolis nous a démontré une certaine fiabilité, mais néanmoins celle-ci ne peut être aujourd'hui affirmée. Une vérification formelle de l'ensemble du système s'impose,

afin de garantir certaine propriétés, comme par exemple le fait qu'un message émis soit bien délivré à son destinataire ou qu'une rupture de la communication soit bien détectée dans les temps impartis par l'application.

La validation formelle impose tout d'abord une modélisation du système dans un formalisme ad hoc avant l'expression des propriétés et leur vérification. Des premiers essais ont été réalisés avec les environnements Spin/Promela [Hol97] ou Uppaal [BLL+98]. Ils ont d'ailleurs permis de détecter, en phase de modélisation, le non-traitement d'un message. Ces travaux devront bien évidement être poursuivis afin de mieux appréhender les potentialités des outils vis à vis de notre application et des propriétés à valider. De plus, la modélisation pouvant s'effectuer à plusieurs niveaux de détail et il n'est pas sûr qu'un même outil soit adapté à la fois à une description à gros, moyen ou petit grain.

La prise en compte des délais de communication nous imposera également l'utilisation de formalismes pouvant exprimer le temps ou du moins des bornes temporelles.

N'oublions pas également que notre système est composé d'un noyau complété par des processus écrits spécialement pour gérer une machine donnée. Il serait utile de pouvoir disposer de techniques de vérifications partielles afin de pouvoir valider les composants de manière autonomes avant de construire un ensemble complet.

Comparativement aux travaux sur la validation du Grafcet que nous avons mené, la validation d'environnements d'e-productique apparaît plus complexe, mais les outils proposés aujourd'hui ont des spectres plus larges et autorisent donc des traitements plus complets. Une approche multi-modèles semblent également être possible.

Ces différentes propositions se placent directement dans la suite du projet e-productique. Menées à terme elles renforceront la plateforme existante en proposant des mécanismes plus complets et plus fins au niveau de l'interface homme-machine, de la prise en compte du réseau, de la construction automatique ainsi que de la sûreté de fonctionnement.

Le travail engagé avec la société LivBag, sur l'étude de leurs chaînes de production et leur optimisation, me semble également une voie de recherche prometteuse. En effet, les premières études menées semblent montrer des gisements de productivité importants dans les applications de type manufacturière. Ces applications sont de plus en plus complexes car elles cherchent à offrir un maximum de services aux utilisateurs, parfois au détriment de leur tâche première de contrôle-commande de la ligne automatisée. Elles intègrent de plus en plus de composants informatiques comme des bases de données réparties ou des systèmes d'exploitation temps-réel, Elles sont distribuées et connectées aux réseaux de l'entreprise, doivent rester réactives et surtout assurer une fiabilité de production. Néanmoins, les méthodes de conception actuellement utilisées ont du mal à intégrer l'ensemble de ces contraintes.

Les méthodes et outils de l'informatique doivent permettre dans ce contexte, de modéliser les applications actuelles, de comprendre leur fonctionnement en détail et de trouver, soit de manière formelle soit par simulation, les noeuds qui, une fois libérés, permettraient d'augmenter la rentabilité globale du système. Partant de ces analyses, des méthodes de conception intégrant l'ensemble des paramètres pourront alors être proposées et des outils, allant de la spécification au déploiement sur site devront être construits.

Quelle(s) méthode(s) faut-il utiliser en phase d'analyse ? Quelle(s) méthode(s) de développement doit-on préconiser ? Comment intégrer ces propositions dans ce domaine professionnel ? Quels gains de productivité sont attendus ? Toutes ces questions restent aujourd'hui ouvertes et nécessiteront de mettre en commun les connaissances de diverses communautés afin de proposer des réponses adéquates.

3.3 Perspectives à long terme

A plus long terme, je souhaite conserver mon engagement dans le métier d'enseignant-chercheur : l'Enseignement et la Recherche sont deux disciplines intimement liées qui se nourrissent mutuellement. Maintenir une activité de recherche oblige à se tenir à jour des progrès à la fois sur les plans conceptuels et pratiques, et à anticiper les évolutions de la discipline. Ces connaissances doivent être alors présentées aux étudiants ce qui ne peut qu'enrichir leur formation. Par exemple lors de l'arrêt du travail sur le Grafcet et les langages synchrones, j'ai choisi de rejoindre le projet e-productique, parce qu'il me permettait de mieux comprendre les enjeux à venir ainsi que les mécanismes de l'Internet et qu'il nécessitait l'apprentissage de connaissances autour des applications réparties et du langage Java. Toutes ces notions font aujourd'hui parti du bagage nécessaire de l'informaticien.

De toute évidence, l'informatique a grandi au cours de ces dernières années et il est difficile d'en suivre toutes les évolutions. Le champ d'application qui m'a toujours plus particulièrement intéressé, est le contrôle de machines et de systèmes plus ou moins autonomes, connectés à un ordinateur. Mon souhait est de continuer à explorer au mieux ce type d'applications et à participer à la construction d'un pont entre plusieurs disciplines que sont l'informatique, les automatismes, la robotique, la productique... Il s'agit à la fois de comprendre les problèmes posés, les solutions apportées, les méthodes utilisées avec un regard d'informaticien, puis de comparer avec des problèmes similaires de la discipline afin d'enrichir les différentes communautés et de partager les connaissances.

L'enseignement de la recherche, et donc l'encadrement d'étudiants fait également partie de mes intérêts principaux. C'est un moment au cours duquel la communication et l'échange de savoirs mutuels sont fondamentaux et nous permettent d'oublier la rigueur de nos ordinateurs. C'est aussi en partie grâce aux étudiants, que les projets avancent et qu'ils deviennent ensuite réalité.

Bibliographie

[ACD90] R. Alur, C. Courcoubetis et D. Dill. Model-checking for real-time systems. Dans *5th annual IEEE symposium on Logics In Computer Science*, Juin 1990.

[ADE81] ADEPA. *GEMMA : guide d'études des modes de marches et arrêts*. Agence pour le Développement de la Productique Appliquée, 1981.

[AFC77] AFCET. Pour une représentation normalisée du cahier des charges d'un automatisme logique, Novembre 1977.

[Arn92] A. Arnold. *Systèmes de transitions finis et sémantique des processus communicants*. Masson, 1992.

[ASLPM94] M. Arnoux, J.L. Scharbarg, P. Le Parc et L. Marcé. Simulation du fonctionnement d'une usine à béton. Dans *Symp. Automatisation de processus mixtes: les systèmes dynamiques hybrides*, Bruxelles Belgique, Novembre 1994. SEE-IBRA.

[BB91] A. Benveniste et G. Berry. The synchronous approach to reactive and real-times systems. Dans *Proceedings of the IEEE*, volume 79, 9, Septembre 1991.

[BCE+02] A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. Le Guernic et R. de Simone. The Synchronous Languages Twelve Years Later. Dans *Proceedings of the IEEE, special issue on Embedded Systems*, 2002.

[BCQ+01] A. Bicchi, A. Coppelli, F. Quarto, L. Rizzo, F. Turchi et A. Balestino. Breaking the lab's walls telelaboratories at the university of pisa. Dans *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, pages 1903 – 1908, Seoul, Korea, Mai 2001.

[BDS91] F. Boussinot et R. De Simone. The ESTEREL language. Dans *Proceedings of the IEEE*, volume 79, 9, Septembre 1991.

[Ben89] A. Benveniste. Les langages synchrones : des noyaux logiciels pour la spécification et la conception des systèmes de contrôle-commande, Février 1989. Rapport 1 C2A.

[BLL+98] J. Bengtsson, K.G. Larsen, F. Larsson, P. Pettersson, W. Yi et C. Weise. New Generation of Uppaal. Dans *International Workshop on Software Tools for Technology Transfer*, Juillet 1998.

[BMG02] S. Barré, A. Mamoune et J. Le Guen. Les nouvelles technologies de l'information et de la communication (ntic) dans les systèmes d'automatismes. Dans *Colloque National sur la Recherche dans les IUT*, Le creusot - France, 2002.

[Boe00] J. CL. Boehm. MES : Manufacturing Execution System : le chaînon manquant entre planification et production, Décembre 2000. CETIM.

[Bry86] R.E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):pp 677–691, Août 1986.

[CLR00] M. Crovella, C. Lindemann et M. Reiser. Internet performance modeling: the state of the art at the turn of the century. *Performance Evaluation*, 42(2-3):91–108, Septembre 2000.

[Com01] D. Comer. *TCP/IP : architecture, protocoles, applications.* Dunod, 2001.

[Cou90] J. Coutaz. *Interfaces Homme-Ordinateur : conception et réalisation.* Dunod, 1990.

[DC02] R. Dragos et M. Collier. An mpls based architecture for differentiated web service. Dans *International Conference on Software Telecommunications and Computer Networks (IEEE - SoftCom2002)*, Split, Croatie, Octobre 2002.

[Dut92] B. Dutertre. *Specification et preuve de systemes dynamiques.* Thèse de Doctorat, Université de Rennes 1, 1992.

[EM88] P. Even et L. Marcé. Pyramide: an interactive tool for modelling of teleoperation environments. *IEEE International Workshop on Robotics and Remote Systems (IROS '88)*, pages 725–730, Novembre 1988.

[Esp01] B. Espiau. La robotique : histoire et perspectives. La science au présent - Encyclopedia Universalis, 2001.

[Eve00] P. Even. Modélisation 3d interactive de l'environnement de robots téléopérés. Mémoire d'Habilitation à Diriger des Recherches, Décembre 2000.

[FAAdLR03] P. Fraisse, C. Agniel, D. Andreu et J.A. Segovia de Los Rios. Teleoperations over ip network: Virtual puma robot. Dans *International Conference on Industrial Technology ICIT'03*, Maribor - Slovénie, Décembre 2003.

[FBTZ02] V. Firoiu, J. Le Boudec, D. Towsley et Z. Zhang. Theories and models for internet quality of service. *Proceedings of the IEEE, 2002.*, 90:1565–1591, Septembre 2002.

[FLPM95] J.L. Fleureau, P. Le Parc et L. Marcé. Signal: a language for low level implementation. Dans *Southeastcon'95:visualizing the future*, USA Raleigh North Carolina, Mars 1995. IEEE.

[FP01] S. Floyd et V. Paxson. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking*, 9:392–403, Août 2001.

[Fra01] F.F. Jimenénez Fraustro. *Conception sûre des automatismes industriels : modélisation synchrone de langages d'automates programmables de la norme CEI-61131-3.* Thèse de Doctorat, Université de Rennes 1, Mars 2001.

[Gin04] G. Gini. Robotics education, teleprogramming, telecontrol through the internet. Dans *ISR'2004 : International Symposium on Robotics*, Paris, France, Mars 2004.

[GML03] E. Gnaedinger, F. Michaut et F. Lepage. Implémentation d'une architecture de qds spécifique permettant l'adaptation d'une application de contrôle à distance d'un robot mobile. Dans *Quatrième Conférence Internationale sur l'Automatisation Industrielle*, Montréal, Canada, Juin 2003.

[Goe52] R. C. Goertz. Fundamentals of general-purpose remote manipulators. *Nucleonics*, 10-11:36–45, Novembre 1952.

[GRD02] J.P. Georges, E. Rondeau et T. Divoux. Evaluation of switched ethernet networks in an industrial context by using network calculus. Dans *4th International Workshop on Factory Communication Systems*, Västeras - Suède, Août 2002.

[GS] K. Goldberg et J. Santarromana. The telegarden. htttp://telegarden.aec.at/.

[GS01] K. Goldberg et R. Siegwart. *Beyond Webcams : an introduction to online robots*. The MIT Press, 2001.

[HCRP91] N. Halbwachs, P. Caspi, P. Raymond et D. Pilaud. Programmation et vérification des systèmes réactifs : le langage LUSTRE. *Technique et Science Informatiques*, 10(2):pp 139–158, 1991.

[HHV+00] P. Harmo, A. Halme, P. Virekoski, M. Halinen et H. Pitkänen. Etälä - virtual reality assisted telepresence system for remote control and maintenance. Dans *1st IFAC Conference on Mechatronic Systems*, pages 1075–1080, Darmstadt, Germany, Septembre 2000.

[HNSS94] T.A. Henzinger, X. Nicollin, J. Sifakis et Yovine S. Symbolic model-checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[Hol97] Gerard Holzmann. The model checker spin. *IEEE Transactions on Software Engineering*, 23(5):1–17, Mai 1997.

[HPMC02] B. Huffaker, D. J. Plummer, D. Moore et K. Claffy. Topology discovery by active probing. Dans *Symposium on Applications and the Internet (SAINT)*, Nara, Japan, Février 2002.

[ICA03] *ICAR2003 - 4ème école d'été sur les intergiciels et la construction d'applications réparties*. Projet Sardes - INRIA/IMAG-LSR, Août 2003.

[KgRD02] N. Krommenacker, J.P. georges, E. Rondeau et T. Divoux. Designing, modelling and evaluating switched ethernet networks in factory communication systems. Dans *1st International Workshop on Real-Time LANs in the Internet Age*, Vienne, Autriche, Juin 2002.

[Lel00] A. Lelevé. *Contribution à la télé-opération de robots en présence de délais de transmission variables*. Thèse de Doctorat, Université de Montpellier (Lirmm), Décembre 2000.

[LGG90] P. Le Guernic et T. Gautier. Data-flow to Von Neumann : the SIGNAL approach, may 1990. Rapport de recherche INRIA 1229.

[L'H97] D. L'Her. *Modélisation du Grafcet temporisé et vérification de propriétés temporelles*. Thèse de Doctorat, Université de Rennes 1, Septembre 1997.

[Lho93] P. Lhoste. Essai de formalisation des modèles internes/externes du GRAFCET, janvier 1993. Réunion du groupe GRAFCET de l'AFCET.

[LL94] C. Lewerentz et T. Lindner. A comparative study in formal specification and verification. Dans *Formal Development of Reactive Systems : Case Study Production Cell*, pages 1 – 54. Springer-Verlag LNCS 891, 94.

[LLPM95] D. L'her, P. Le Parc et L. Marcé. Modeling and proving grafcets with transitions systems. Dans *2nd AMAST workshop on real-time systems*, Bordeaux, Juin 1995. AMAST.

[LLPM96] D. L'her, P. Le Parc et L. Marcé. Modélisation et vérification du grafcet. Dans *Modélisation des systèmes réactifs*, Brest, Mars 1996. AFCET.

[LLPM98] D. L'her, P. Le Parc et L. Marcé. Proving sequential function chart programs using automata. Dans *Workshop on implementing automata*, Rouen, France, Septembre 1998. WIA.

[LLPM99a] D. L'her, P. Le Parc et L. Marcé. Modélisation et vérification du grafcet temporisé. *Journal Européen des Systèmes Automatisés*, 33(5-6):651 683, Juillet 1999.

[LLPM99b] D. L'her, P. Le Parc et L. Marcé. Proving sequential function chart programs using automata. Dans *Workshop on implementing automata*, pages 149 – 163. Springer Verlag LNCS 1660, 1999.

[LLPM99c] D. L'her, P. Le Parc et L. Marcé. Une étude des relations grafcet langages synchrones flots de données. Dans *Journée SEE sur les nouvelles percées dans les langages pour l'automatique*, Amiens, Novembre 1999.

[LLPM01] D. L'her, P. Le Parc et L. Marcé. Proving sequential function chart programs using automata. *Theoretical Computer Science 267*, pages 141–155, Septembre 2001.

[LLPSM98] D. L'her, P. Le Parc, J.L. Scharbarg et L. Marcé. modélisation et vérification de la cellule korso par une boite à outils grafcet. Dans *ADPM'98*, Reims, France, Mars 1998. AFCET, SEE.

[LOIP01] Laboratoire LIMI-EA2215, Oceanopolis et IRVI-Progeneris. The penguin project. http://www.oceanopolis.com/visite/visite.htm, 2001.

[LP91] P. Le Parc. Etude du langage GRAFCET et de ses relations avec le langage SIGNAL, Septembre 1991. Rapport de stage - DEA Informatique Rennes.

[LP94] P. Le Parc. *Apport de la méthodologie synchrone pour la définition et l'utilisation du langage Grafcet.* Thèse de Doctorat, Université de Rennes 1, Janvier 1994.

[LPLSM98] P. Le Parc, D. L'her, J.L. Scharbarg et L. Marcé. Le grafcet revisité à l'aide d'un langage synchrone flot de données. *Technique et Science Informatiques*, 98(1), Janvier 1998.

[LPLSM99] P. Le Parc, D. L'her, J.L. Scharbarg et L. Marcé. Grafcet revisited with a synchronous data-flow language. *Revue IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Human*, 29(4), Août 1999.

[LPM93] P. Le Parc et L. Marcé. Synchronous definition of grafcet with signal. Dans *International Conference on Systems, Man and Cybernetics*, pages 675–680, Le Touquet France, Octobre 1993. IEEE-SMC.

[LPM94] P. Le Parc et L. Marcé. Définition synchrone du grafcet à l'aide du langage signal. Dans *Real-time systems 94*, Paris, Janvier 1994. RTS.

[LPOVM99] P. Le Parc, P. Ogor, J. Vareille et L. Marcé. Robot control from the limi lab. http://similimi.univ-brest.fr, 1999.

[LPOVM02] P. Le Parc, P. Ogor, J. Vareille et L. Marcé. Web based remote control of mechanical systems. Dans *International Conference on Software Telecommunications and Computer Networks (IEEE - SoftCom2002)*, Split, Croatie, Octobre 2002.

[LPQC+99] P. Le Parc, R. Querrec, P. Chevaillier, J. Tisseau et L. Marcé. Un environnement de développement pour la conception de systèmes automatisés de production. Dans *2ème congrès sur la Modélisation des Systèmes Réactifs, MSR'99*, Cachan, France, Mars 1999.

[LPQM94a] P. Le Parc, B. Queguineur et L. Marcé. Two proof methods for the grafcet language. Dans *Workshop on real-time programming*, Lac de Constance Suisse, Juin 1994. IFAC-IFIP.

[LPQM94b] P. Le Parc, B. Quéguineur et L. Marcé. Two proof methods for the grafcet language. *Annual Review in Automatic Programming*, 18:61–65, 1994.

[LPS+03] A. Lelevé, P. Prévot, C. Subai, D. Noterman et M. Guillemot. Towards remote laboratory platforms with dynamic scenarios. Dans *7th world Multiconference on Systemics, Cybernetics and Informatics (SCI2003)*, Orlando - Floride, Juillet 2003.

[LPVM04a] P. Le Parc, J. Vareille et L. Marcé. E-productique ou contrôle et supervision distante de systèmes mécaniques sur internet. *JESA : Journal Européen des Systèmes Automatisés*, 2004. en révision.

[LPVM04b] P. Le Parc, J. Vareille et L. Marcé. Web based remote control of mechanical systems. Dans *1st IFAC Symposium on Telematics Applications in Automation and Robotics*, Helsinki, Finlande, Juin 2004.

[LPVM04c] P. Le Parc, J. Vareille et L. Marcé. Web remote control of machine-tools the whole world within less than one half-second. Dans *ISR'2004 : International Symposium on Robotics*, Paris, France, Mars 2004.

[LSLP+98] D. L'her, J.L. Scharbarg, P. Le Parc, J. Vareille et L. Marcé. Specification and verification of the korso production cell. Dans *INCOM98*, Nancy, France, Juin 1998. IFAC, IEEE.

[MET01] METROPOLIS. Projet RNRT 2001: Metrologie pour l'internet et ses services. http://www.telecom.gouv.fr/rnrt/projets/res_01_57.htm, 2001.

[ML02] F. Michaut et F. Lepage. A tool to monitor the network quality of service. Dans *IFIP-IEEE conference on Network Control and Engineering (CON'2002)*, Paris, France, Octobre 2002.

[MLLP95] L. Marcé, D. L'her et P. Le Parc. Vers l'amélioration de la qualité des modèles des systèmes de production: sémantique et validation. Dans *Congrès international de Génie Industriel*, Montreal, Octobre 1995.

[MLLP96] L. Marcé, D. L'her et P. Le Parc. Modelling and verification of temporized grafcets. Dans *CESA - Computing Engineering in Systems Application*, Lille, Juin 1996. IEEE.

[MLP92a] L. Marcé et P. Le Parc. Defining the semantics of languages for programmable controllers with the help of synchrnous processes. Dans *Proceedings of the* 18^{th} *IFAC/IFIP workshop on Real-Time Programming*, Bruges - Belgique, Juin 1992.

[MLP92b] L. Marcé et P. Le Parc. Modélisation de la sémantique du GRAFCET à l'aide de processus synchrones. Dans *Actes du congrès GRAFCET'92*, Mars 1992.

[MLP93] L. Marcé et P. Le Parc. Defining the semantics of languages for programmable controllers with synchronous processes. *Control Engineering Practice*, 1(1):79–84, Janvier 1993.

[Ogo01] P. Ogor. *Une architecture générique pour la supervision sûre à distance de machines de production avec Internet.* Thèse de Doctorat, Université de Bretagne Occidentale, Décembre 2001.

[OLPVM00a] P. Ogor, P. Le Parc, J. Vareille et L. Marcé. Controlling remote systems on the web. Dans *Proceedings of the ACIS* 1^{st} *International conference on Software Engineering Applied to Networking and Parallel/Distributed Computing*, Reims, France, Mai 2000.

[OLPVM00b] P. Ogor, P. Le Parc, J. Vareille et Lionel Marcé. Remote control of robotics systems on the web. Dans *NETTIES 200*, Oulu, Finlande, Mars 2000.

[OLPVM01] P. Ogor, P. Le Parc, J. Vareille et L. Marcé. Control a robot on internet. Dans 6^{th} *IFAC "Low Cost on Automation" Symposium*, Berlin, Allemagne, Octobre 2001.

[OMKC00] S. Otmane, M. Mallem, A. Kheddar et F. Chavand. Ariti : an augmented reality interface for teleoperation on the internet. Dans *Advanced Simulation Technologies Conference 2000 "High Performance Computing" HPC 2000*, pages 254 – 261, Wyndham City Center Hotel, Washington, D.C., USA, Avril 2000.

[OVLPM01] P. Ogor, J. Vareille, P. Le Parc et L. Marcé. Vers les machines outils et les robots communicants. Dans *Séminaire Objets Communicants SOC'2001 - SEE/RNRT*, Grenoble, France, Octobre 2001.

[Owe03] P. Owezarski. Métrologie de la qualité de service. Dans *Ecole d'été temps réel (ETR2003) - Systèmes, Réseaux et Applications*, Toulouse, France, Septembre 2003.

[PJSB98] M. Parris, K. Jeffay, D. Smith et J. Borgersen. A better-than-best-effort service for continuous media udp flows. Dans *Proceedings, Eighth Intl. Workshop in Network and Operating System Support for Digital Audio and Video*, pages 300–307, Cambridge, UK, Juillet 1998.

[PM02] E. Peulot et S. Moreno. *Le Gemma*. Editions Casteillas, 2002.

[REA] REARSITE. A propos de rearsite. http://listes.cru.fr/rs/fd/index.html.fr.

[RG03] N. Rafii et Y. Graton. Architecture de communication dans le domaine industriel basée sur ethernet. Dans *Université d'automne 2003*, Nantes et Angers, France, Novembre 2003.

[Rou94] J.M. Roussel. *Analyse de Grafcets par génbération logique de l'automate équivalent*. Thèse de Doctorat, Ecole Normale Supérieure de Cachan, Décembre 1994.

[RUC] RUCA. Réseau universitaire des centres d'autoformation. http://www-ruca.univ-lille1.fr/.

[SCFJ03] H. Schulzrinne, S. Casner, R. Frederick et V. Jacobson. Rfc 3550 : Rtp: A transport protocol for real-time applications, Juillet 2003. Remplace la RFC 1889.

[Sch02] Schneider Electric. Les bus et réseaux de terrain en automatisme industriel. *Inter-Sections*, Novembre 2002.

[SF00] P. Saucy et F.Mondada. Khepontheweb: Open access to a mobile robot on the internet. *IEEE robotics and automation magazine*, pages 41–47, mars 2000.

[SGSB03] E. Simon, E. Gressier-Soudan et J. Berthelin. Avoid lan switches ip routers provide a better alternative for a real-time communication system. Dans *2nd International Workshop on Real-Time LANs in the Internet Age*, Porto - Portugal, Juillet 2003.

[She92] T.B. Sheridan. *Telerobotics, automation and human supervisory control*. The MIT Press, 1992.

[SHS02] L. Strandén, J. Hedberg et H. Sivencrona. Machine control via internet - a holistic approach. Technical report, Swedish National Testing and Research Institute, 2002.

[Ste98] M.R. Stein. Painting on the world wide web : the pumapaint project. Dans *In Proceeding of the IEEE IROS'98 Workshop on Robots on the Web*, pages Victoria, Canada, Octobre 1998.

[SW96] C.S. Smith et P.K. Wright. Cybercut: A world wide web based design to fabrication tool. Journal of Manufacturing Systems, Janvier 1996.

[TD97] K. Taylor et B. Dalton. Issues in internet telerobotics. *FSR'97 International Conference on Field and Service Robots*, 8-10 Décembre 1997.

[Tho] G. Thomas. Tcp and udp for control. The Online Industrial Ethernet Book.

[TLPK91] S. Thiébaux, P. Le Parc et L. Kervella. La programmation temps-réel - l'approche synchrone, Janvier 1991. Etude bibliographique - DEA Rennes.

[TPL03] TPLINE. Les travaux pratiques de sciences industrielles sur des systèmes en fonctionnement en temps réel. http://www.tpline.net/contact/tp/tp_01.php, 2003.

[Van99] F.C. Vandenbussche. Soho's recovery - an unprecedented success story. ESA Bulletin, mars 1999.

[Vie00] J. Vieille. Intégration production - entreprise : La norme ansi/isa s95. Dans *10eme journées des CPIM de France*, Paris, France, Septembre 2000.

[Vit01] S. Vitturi. On the use of ethernet at low level of factory communication systems. *Computer Standards and Interfaces*, 23:267–277, 2001.

[VLPM02a] J. Vareille, P. Le Parc et L. Marcé. Démonstration à distance des travaux de recherche du limi dans le domaine de la téléproductique. Dans *JJCR'16*, Lyon, France, Septembre 2002.

[VLPM02b] J. Vareille, P. Le Parc et L. Marcé. Vers les machines de production communicantes. Dans *Conférence ISA-SEE "le contrôle de la production dans l'entreprise réactive de l'ère Internet"*, Nice, France, Mars 2002.

[VLPM03a] J. Vareille, P. Le Parc et L. Marcé. Le contrôle réactif par internet et son application aux machines de production. Dans *CPI'2003*, Meknes, Maroc, Octobre 2003.

[VLPM$^+$03b] J. Vareille, P. Le Parc, L. Marcé, S. Barré, A. Mamoune et J. Le Guen. Contôle actif de machines de production par internet. Dans *CFM'2003*, Nice, France, Septembre 2003.

[Web] WebCT. Webct's flexible e-learning solutions. http://www.webct.com/.

[Web01] WebSurg. Operation lindbergh : first transatlantic robot-assisted operation. http://www.websurg.com/lindbergh/index.cfm, Septembre 2001.

[Whi97] P. White. Rsvp and integrated services in the internet : A tutorial. *IEEE Communication Magazine*, pages 100–106, Mai 1997.

[Yov93] S. Yovine. *Méthode et Outils pour la Vérification Symbolique de Systèmes Temporisés*. Thèse de Doctorat, Institut National Polytechnique de Grenoble, Mai 93.