Université de Nice - Sophia Antipolis

ORCCAD : un environnement de conception, de validation et d'exécution pour la robotique

Habilitation à Diriger des Recherches

présentée par

Daniel SIMON

Soutenue le 1er avril 1998 devant le jury composé de :

P. SANDER	Professeur à l'Université de Nice - Sophia Antipolis	Président
C. ANDRÉ	Professeur à l'Université de Nice - Sophia Antipolis	Rapporteur
M. BRADY	Professeur à l'Université d'Oxford	Rapporteur
J.P. ELLOY	Professeur à l'École Centrale de Nantes	Rapporteur
B. ESPIAU	Directeur de Recherche à l'Inria Rhône-Alpes	Rapporteur
M. SORINE	Directeur de Recherche à l'Inria Rocquencourt	Examinateur

Travaux réalisés à l'INRIA

La différence entre la théorie et la pratique est plus grande en pratique qu'en théorie

Les travaux exposés dans ce mémoire ont été initialisés dans le projet Robotique de l'IRISA et se sont poursuivis à l'INRIA Sophia-Antipolis dans les projets Prisme puis Icare. Ils sont nés du désir de mettre en oeuvre les travaux théoriques sur la commande des robots réalisés dans ces équipes et du constat de l'inadéquation des systèmes actuels de programmation de robots pour atteindre cet objectif. Ils ont contribué à donner naissance à ORCCAD, un ensemble de concepts, de méthodes et d'outils destinés à spécifier, programmer, vérifier et implanter des applications robotiques complexes. Ils ont servi de cadre à plusieurs thèses, à des collaborations avec d'autres projets INRIA et en particulier avec le projet Bip de l'U.R. Rhône-Alpes, ainsi qu'à quelques collaborations industrielles avec Aleph-Technologies et l'Ifremer.

Le document est organisé de la façon suivante : dans une première section nous dressons un panorama des architectures de contrôle/commande existant en robotique. La section suivante présente les principales idées ayant guidé notre démarche ainsi que les structures de l'architecture de contrôle/commande en ayant découlé. La robotique sous-marine est depuis plusieurs années notre terrain d'application favori : la section 3 donne un aperçu des activités menées dans ce domaine et donne en particulier un scénario de mission sous-marine en cours d'implantation à l'Ifremer. Cet exemple sert de support pour illustrer la suite du mémoire. Les méthodes utilisées pour spécifier et valider les actions de base de notre architecture, les *Tâches-Robot*, sont expliquées dans la section suivante. Ces actions de base sont ensuite composées de façon hiérarchique et structurée pour obtenir des actions validées de complexité croissante jusqu'à obtenir une mission complète comme illustré par la section 5. En conclusion, nous faisons le bilan des outils développés dans le cadre de ce projet et des actions de recherche à poursuivre. Ce document de synthèse est volontairement assez informel : les travaux réalisés sont essentiellement traités au travers d'un exemple et les détails techniques peuvent être trouvés dans les annexes constituées des principales et plus récentes publications concernant ORCCAD.

Remerciements

Merci à :

Peter Sander qui a accepté de présider ce jury après avoir vaillament supporté mon enseignement en plongée,

Charles André, pionnier des machines d'exécution pour Esterel, pour avoir accepté de rapporter sur ce travail,

Mike Brady, pour s'être intéressé à ce travail et avoir accepté de l'examiner,

Jean-Pierre Elloy pour toutes les discussions enrichissantes que nous avons pu avoir depuis tant d'années, et aussi pour le temps consacré à rapporter sur mes thésards,

Bernard Espiau qui a participé activement et dès le début aux travaux concernant Orccad. Merci pour son soutien et ses encouragements constants,

Michel Sorine est l'un des premiers à m'avoir acceuilli à l'Inria, je le remercie d'être encore présent aujourd'hui.

Sans pouvoir citer tout le monde de peur d'en oublier, merci également à tous les stagiaires, thésards, ingénieurs et chercheurs ayant participé ou soutenus les travaux autour d'Orccad, au sein du projet Icare et ailleurs.

Table des matières

Un robot mobile destiné à intervenir dans un environnement hostile (sous-marin, nucléaire ou spatial) est un parfait exemple de système critique. Cela signifie que pour un tel système toute opération de reprise, de reconfiguration ou autre, qui nécessite l'intervention directe d'un opérateur humain est toujours coûteuse, souvent difficile et parfois même tout simplement impossible. C'est pourquoi il faut le plus possible leur assurer une fiabilité aussi grande que possible et ceci avant leur lancement en opération. Plus précisément, une fois définie une mission, il apparaît nécessaire de vérifier que : (1) ses spécifications sont correctes, c'est-à-dire correspondent aux objectifs de l'utilisateur ; (2) sa programmation est conforme aux spécifications ; (3) les contraintes induites par les aspects liés à l'implantation temps-réel ne modifient pas son comportement. Ceci concerne principalement deux domaines, qui, de plus, ne doivent pas être considérés indépendamment : les aspects commande, modélisés par des équations différentielles en temps continu ou échantillonné et les aspects contrôle, représentés par des systèmes à événements discrets.

Les études de nature informatique entreprises au sein du projet Icare de l'Inria Sophia-Antipolis ont pour objectif de répondre aux problèmes posés par la programmation des systèmes robotiques avancés dont la spécificité est d'être des systèmes dynamiques généralement rapides, distribués, réactifs, multitâches et reconfigurables. Plus particulièrement, il s'est avéré nécessaire de développer une architecture de contrôle/commande et des outils destinés à la mise en oeuvre des algorithmes de commande de robots étudiés par ailleurs dans le projet.

1 Contexte de la recherche

Un contrôleur de robots doit être capable de traiter une grande diversité d'actions, allant par exemple de la commande d'une cellule flexible d'assemblage en milieu industriel au contrôle d'un véhicule autonome opérant en milieu hostile. La diversité des applications potentielles en robotique et le désir de mettre en oeuvre des algorithmes de commande sophistiqués impliquent une ouverture du système de programmation. D'un point de vue industriel, les contrôleurs actuels restent le plus souvent des machines fermées, dotées d'un langage limité ne permettant pas d'exécuter des actions complexes telles que des commandes référencées capteurs. Outre leurs limitations fonctionnelles, ils ne permettent pas de traiter correctement les aspects temps-réel sous-jacents à toute application robotique, même simple.

Depuis quelques années, de nombreuses propositions d'architecture de contrôle/commande de robots dits "intelligents" ont été faites. Des techniques allant de l'utilisation des sciences cognitives aux théories les plus modernes de l'Automatique sont utilisées à l'aide d'outils informatiques les plus divers. Cependant, alors qu'une architecture complète de commande de robots implique la mise en oeuvre de nombreux sous-systèmes, la plupart des développements ne se focalisent que sur certains d'entre eux. Certaines approches étudient principalement la décomposition fonctionnelle du système alors que d'autres se concentrent principalement sur les problèmes d'implantation. Des outils et des méthodes *ad hoc* sont alors utilisés pour compléter la conception du système, remettant ainsi en cause sa cohérence. Enfin, certains aspects tels que la vérification des programme et l'implantation de ceux ci sur l'architecture cible sont le plus souvent ignorés [33]. Certaines approches reposent sur le développement d'un système d'exploitation temps-réel spécifique tel que Chimera [83] et de systèmes de communication dédiés [47]. Des machines cibles puissantes ont été construites [39] et des bibliothèques de programmes spécifiques à la robotique ont été développées [7]. Ces approches fournissent effectivement des outils d'exécution efficaces mais laissent généralement l'utilisateur sans outils de programmation de haut niveau et surtout sans *méthodologie d'utilisation.*

D'un autre coté, la programmation de "haut niveau" des systèmes robotisés est traditionnellement la chasse gardée des systèmes à base de connaissances et utilise des techniques de raisonnement et de planification. Généralement, ces systèmes purement cognitifs ignorent résolument les bases de la mécanique, de l'automatique et de l'informatique temps-réel. Ces approches conduisent à des systèmes peu efficaces à l'exécution, ayant des difficultés à prendre en compte les incertitudes du monde réel où évoluent les robots. Parmi les approches plus systématiques, dont Hasemann donne un assez large panorama [34], l'architecture hiérarchisée RCS (Real-time Control System) [3] a été proposées comme

standard en vue de promouvoir une méthodologie de développement des programmes et de permettre la réutilisabilité du logiciel. Dans cette architecture très rigide les possibilités de communications "horizontales" entre flots de données sont très restreintes, rendant quasiment impossible la construction de boucles de commande référencées capteurs efficaces en temps réel. De fait, le modèle de référence subit de fortes distorsions en fonction des applications réalisées et perd donc son aspect de standard.

A l'opposé de cette approche hiérarchisée à l'extrême se trouve l'approche "behavioriste" popularisée par Brooks [15]. Ici, les comportements élémentaires du robot sont simplement superposés(i.e. fonctionnent en permanence en parallèle), les conflits entre couches étant résolus au moyen de portes inhibitrices. Il n'y a pas de niveau de supervision global et le comportement du robot est rapidement difficile à prédire et à analyser [26]. En pratique, ces robots sont limités à des comportement spécifiques et répétitifs [16] où le nombre de comportements fonctionnant en parallèle est en fait très réduit. De plus, les ressources informatiques du système sont sous-utilisées : ajouter un processeur chaque fois que l'on veut ajouter un nouveau comportement ne parait pas réaliste en particulier pour des robots mobiles autonomes, souvent limités en poids, en volume et en énergie embarquée ([50], [9]). Des améliorations à l'approche purement behavioriste ont pu être apportées en ajoutant un superviseur au dessus de l'architecture des "behaviors", permettant de n'activer à un moment donné que les couches nécessaires en fonction de l'avancement de la mission ([11]).

Cette approche particulière permet d'introduire les approches "hybrides", tentant de remédier aux défauts des deux systèmes précédents en combinant leurs meilleurs aspects. Le nombre de couches de logiciel est limité : les couches "basses" traitent des aspects commande, utilisant le plus souvent des boucles fermées capteurs/actionneurs en temps échantillonné (qui n'est qu'un abstraction du temps continu idéal), les couches plus "hautes" traitant d'aspects plus abstraits : planification, traitements d'exceptions, changements de modes de marche..., s'exprimant sous forme de réactions à des événements. Ce type de démarche permet de mieux prendre en compte l'aspect fondamentalement *hybride* d'un système robotique et le besoin de *réactivité* vis à vis des incertitudes de l'environnement, réactivité pouvant s'exprimer et être traitée dans les différentes couches avec des formalismes et outils différents.

Il est remarquable de constater que ces approches ont été principalement développées par des chercheurs confrontés à la dure *réalité expérimentale*, souvent dans le monde des véhicules autonomes, en particulier sous-marins [11, 35, 49]. Dans ces approches le premier soin est apporté à la définition des actions de bases et des procédures de récupération ainsi qu'à leur validation en général par simulation puis tests prudents. Il s'agit effectivement de systèmes critiques où le risque de perte totale n'est pas a exclure et où il est impensable de mettre à l'eau un système valant quelques Mecus sans avoir pris quelques précautions élémentaires.

L'architecture de contrôle/commande mise en oeuvre dans le système ORCCAD (Open Robot Controller Computer Aided Design) développé actuellement au sein du projet Icare appartient à cette famille.

2 ORCCAD : Buts et Structures

2.1 Buts

ORCCAD est un ensemble de méthodes et d'outils permettant de spécifier, structurer, vérifier et programmer des systèmes robotiques devant effectuer des missions complexes, éventuellement en milieu mal connu ou peu structuré.

En dehors de l'étude de l'existant, ce travail a débuté par l'analyse des besoins, et est basé sur les fondements suivants :

- La plupart des actions pouvant être réalisées par des robots peuvent être spécifiée dans le formalisme de l'Automatique et peuvent être réalisée efficacement en temps-réel par des commandes en boucle fermée. Nous croyons fermement que la théorie de la commande doit être utilisée aussi loin que possible dans ce domaine. Cette affirmation s'appuie sur des formalismes solides développés par la communauté automaticienne et en particulier dans le projet Icare : théorie

des *fonctions de tâche* pour la commande de manipulateurs rigides et de systèmes holonomes, *feedback instationnaire* pour les robots non holonomes, utilisation de capteurs *extéroceptifs* pour contrôler le robot dans son environnement.
- Le système robotique, i.e. le contrôleur et son environnement de programmation, doit être accessible à plusieurs types d'utilisateurs de compétences très différentes. En particulier, l'utilisateur du système n'est généralement ni automaticien ni informaticien. Il doit avoir accès à une bibliothèque d'actions de haut niveau, lui permettant de se concentrer sur la spécification de son application. Ces actions doivent être conçues et validées par un automaticien à l'aide d'outils de développement et de test adéquats.
- Les performances finales du système dépendent en grande partie d'une implantation efficace des programmes temps-réel. La plus grande attention doit être portée à l'organisation des calculs sur la machine cible. Il est souhaitable de disposer de méthodes de synthèse et d'outils de vérification à ce niveau.
- La commande de robots nécessite l'usage d'algorithmes complexes, longs à programmer et à tester. Le logiciel produit doit être fiable et si possible formellement vérifiable, en particulier dans le cas de robot autonomes en environnement hostile. L'utilisation de méthodes de conception orientées objets et de langages synchrones peuvent aider à répondre a ces problèmes.

2.2 Méthodologie de conception

Une première réflexion avait abouti à la définition d'une architecture de contrôleur à 3 niveaux [13, 12] :
- Le niveau *application* est accessible à l'utilisateur du système robotique. Une application est vue comme une séquence d'actions de haut niveau associée à un ensemble pré-défini de traitements d'exceptions. Le langage que nous utilisons actuellement pour spécifier une application et effectuer des vérifications de propriété du programme est le langage synchrone Esterel [14]. A noter qu'à ce niveau les actions lancent des processus informatiques qui ne peuvent être considérées comme étant de durée nulle, ce qui a motivé la première implantation de l'instruction de manipulation de tâches asynchrones *exec*.
- Le niveau *commande* est celui où sont définies les actions élémentaires sous forme de *Tâches-Robot* (TR). Ce concept clé dans notre approche, apparu dans [13], a été étudié et formalisé dans la thèse d'È. Coste-Manière [23, 24] et plus tard raffiné dans celle de K. Kapellos [27]. Sa structure sera formalisée et détaillée plus loin. Il s'agit de spécifier un objet informatique combinant dans un même programme une partie algorithmique (loi de commande en temps échantillonné) et une partie logique réactive spécifiant les pré-conditions, exceptions et post-conditions de la TR. Ce comportement logique est également codé en Esterel. D'un point de vue implantation, une TR est réalisée par l'exécution essentiellement périodique d'un réseau de tâches temps-réel communicantes, les *Tâches-Module* (TM).
- Le niveau *système* contient les outils nécessaires à l'exécution des tâches composant une application : O.S. temps-réel, protocoles de synchronisation entre tâches, drivers spécifiques.... Les ports de synchronisation utilisés pour la communication entre TMs proviennent d'études antérieures (thèse de M. Mejia [44]) concernant un bus de terrain temps-réel et permettent de synchroniser les tâches temps-réel de façon plus ou moins lâche [45], le but ultime étant de rendre la spécification des communications indépendante de la machine d'exécution.

En fait, les premiers exemples d'utilisation du système (e.g. [75]) ont mis en évidence le fait que la TR est encore une action de trop bas niveau pour être efficacement manipulée par l'utilisateur final du système, en particulier au niveau de l'expression de leur séquencement en Esterel, de la définition des traitements d'exception et de la vérification du programme produit. Une structure intermédiaire entre le niveau des TRs et celui de l'application a été donc été introduite dans la thèse de K. Kapellos [36] : la *Procédure-Robot* (PrR), obtenue par composition hiérarchique de TRs, de PrRs et de traitements d'exceptions préalablement définis, programmés et validés. Ces combinaisons s'effectuent de façon structurée et hiérarchisée, offrant ainsi à l'utilisateur du niveau application des

actions validées de complexité variable, les plus simples étant les TRs elles mêmes et les plus complexes codant une mission complète du système ([36], [76]).

2.3 Tâches, procédures et traitements d'exceptions

D'un point de vue plus formel, une TR est constituée de la spécification complète ([72]) :
- D'une loi de commande entièrement paramétrée, de structure invariante pendant toute la durée de la TR ;
- D'un comportement logique pré-défini gérant l'activité de cette loi de commande.

Les événements manipulés par ce comportement logique sont eux mêmes typés en trois classes :
- les pré-conditions, dont la satisfaction permet l'activation effective de la loi de commande ;
- des exceptions de types 1, 2 ou 3 ;
- les post-conditions émises lors de la terminaison de la TR.

Les pré et post-conditions peuvent être utilisées dans un but de pure synchronisation, ou bien encore être associées à des processus de mesures protégés par des chiens de garde temporels.

Les exceptions de type 1 sont traitées localement dans la TR, les types 2 sont traitées au niveau de la PrR englobante, les exceptions de type 3 sont fatales pour l'application.

Cet ensemble de signaux constituent la *vue externe*, purement événementielle, de la TR qui est utilisée pour composer hiérarchiquement les TRs en procédures de complexité variable jusqu'à obtention d'une mission complète ([36], annexe C).

Cette composition s'effectue grâce à des opérateurs ([25]) exprimant différentes formes de synchronisation : séquence, parallélisme, conditionnelle...De plus, des traitements de récupération sont définis en cas d'échec de l'exécution d'une des TRs utilisées, afin de prendre en compte l'environnement d'exécution de la PrR.

Plus précisément, une PrR est définie par :
- un comportement réactif similaire à celui de la TR : pré-conditions, post-conditions, exceptions de type 2 ou 3 ;
- un programme principal qui correspond à l'exécution nominale de l'application. Ce programme est exprimé en termes de composition de TR (ou de PrR préalablement définies) ;
- un ensemble de traitements de récupération associés aux exceptions de type 2 susceptibles d'être soulevées par les TR utilisées dans le programme principal ou par un observateur local à la PrR. Ces traitements de récupération sont exprimés de la même façon que le programme principal.

Cette forte structuration du logiciel de contrôle/commande dans le modèle d'ORCCAD résumée dans la figure 1, permet de formaliser les traitements d'exceptions à effectuer et est également à la base de l'utilisation de méthodes de vérifications formelles.

Les actions de bases (les TRs) sont exécutées au niveau fonctionnel, où l'activité est essentiellement de nature calculatoire (exécution périodique d'une loi de commande). Le comportement logique défini à ce niveau est très spécifique du robot utilisé. Les exceptions de type 1 sont traitées à l'intérieur de la TR par modification de paramètres, par exemple pour passer "en douceur" une singularité d'un bras manipulateur ([37]).

L'activité des PrRs est de nature logique et permet de coordonner les actions de base. Les exceptions de type 2 sont traitées par la PrR englobant l'action émettrice : son traitement peut par exemple provoquer l'arrêt d'une TR en cours d'exécution et l'activation d'une TR de secours. Dans le cas où plusieurs sous-systèmes coopèrent l'émission d'un signal peut également aboutir à une action de synchronisation entre deux activités s'exécutant en parallèle ([80]).

L'émission d'une exception de type 3 correspond à une situation où la mission ne peut plus être exécutée, par exemple suite à une panne irrécupérable d'une ressource physique, et doit amener à une mise en sécurité du système au travers d'une procédure prédéfinie. Remarquons qu'arrêter un robot ou un processus physique en général *n'est pas* arrêter son contrôleur, il suffit de penser par exemple à ce que peut être une procédure d'arrêt d'urgence d'aéronef pour s'en convaincre...

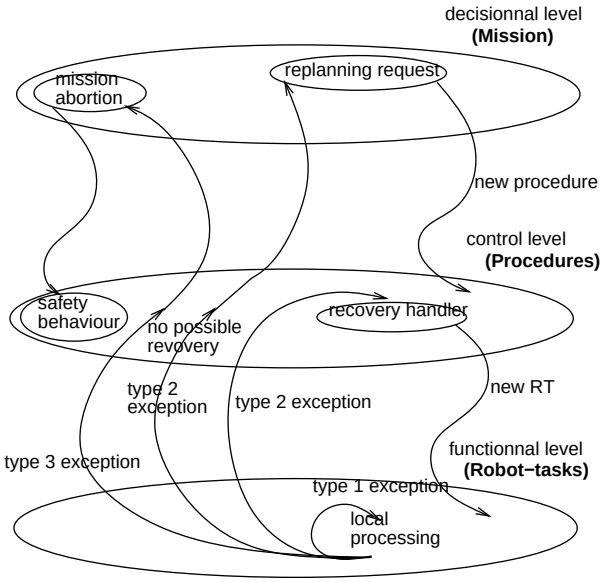


FIG. 1 – Traitement d'exceptions dans Orccad

Idéalement les niveaux "fonctionnels" et "contrôle" devraient être coiffés d'un niveau "décisionnel" permettant de replanifier partiellement la mission en cas d'échec (sans notion de contraintes de type temps-réel) : en l'absence d'un planificateur de tâches réellement réactif et fiable ce niveau n'est pas actuellement implanté. Le niveau mission est en fait constitué de la PrR la plus englobante, codée et vérifiée "à la main".

Ces concepts et les outils associés ont été (au moins partiellement) utilisés dans plusieurs applications utilisant des robots de type très différents : bras manipulateurs ([19],[36]), robot mobile à roues ([51]), train virtuel des véhicules du projet Praxitèle ([38]), programmation de mission de véhicule sous-marin ([75]). Les concepts, méthodes et outils développés dans ORCCAD vont maintenant être illustrés au travers d'un exemple de commande d'un système de manipulation sous-marin, développé au cours du projet Esprit BRA Union ([54]). Cet exemple est par ailleurs décrit en détail en annexe E.

3 Contexte applicatif : la Robotique Sous-Marine

La robotique sous-marine s'est révélée être un domaine d'application riche de problèmes scientifiques intéressant aussi bien des aspects théoriques de commande que de mise en oeuvre. La commande des sous-marins munis d'ailes portantes pose des problèmes analogues à ceux posés par les robots mobiles à roues (non holonomes). Si la vision est très limitée, de nouveaux types de capteurs (sondeurs, sonars...) peuvent être utilisés pour contrôler le robot dans son environnement. La réalisation de véhicules autonomes libres efficaces et fiables suppose l'utilisation d'une approche système globale et nécessite l'existence de systèmes informatiques extrêmement fiables. Le désir de nos partenaires de réaliser des systèmes opérationnels à des horizons raisonnables est très motivant. Il s'agit d'un domaine en expansion où bien souvent la robotisation est incontournable.

Nos actions dans ce domaine passent essentiellement par des collaborations avec le centre Ifremer de Toulon, en particulier avec le laboratoire de Robotique et d'Intelligence Artificielle. La création récente du groupement Robotique Sous-marine Méditerrannée auquel nous participons devrait permettre d'élargir les possibilités de coopération. D'un point de vue moins local, c'est également une occasion de participer à des actions internationales au travers de projets européens Mast et Esprit et d'une collaboration avec la NSF.

3.1 Commande de Véhicules Sous-marins Autonomes

A la suite d'une première convention de recherche sur la faisabilité de la commande de sous-marin en mode suivi de fond, nous avons participé au projet européen MAST "Advanced Research System for Unmanned Autonomous Underwater Vehicle". Il s'agit d'un projet d'études des technologies clés à développer en vue de le réalisation de véhicules autonomes à long rayon d'action. Nous y étions plus particulièrement chargé d'études d'architecture, avec l'objectif d'appréhender d'une façon cohérente la trilogie architecture de perception-architecture d'actionneurs-commande. Ce sujet a servi de cadre à la thèse d'A. Santos ([65]).

Ce travail a d'abord été l'occasion d'acquérir une bonne connaissance de l'état de l'art en modélisation d'engins sous-marins, très bien résumé dans la première partie de la thèse d'A. Santos, et a donné lieu à l'élaboration de modèles de simulation de véhicules sous-marins libres.

D'autre part, dans le thème "suivi automatique de fond à altitude relative constante", un certain nombre de véhicules et d'architectures d'actionneurs typiques ont été analysés, conduisant à l'élaboration des lois de commande correspondantes.

- Les véhicules disposant d'actionneurs de type hélice dans toutes les directions jouissent de bonnes propriétés de commandabilité. L'approche par "fonction de tâche" a été appliquée de manière à permettre la prise en compte simultanée de tâches de suivi de fond et d'évitement d'obstacle ([74],[58],[59]). Une première approche est basée sur la minimisation d'un critère quadratique faisant intervenir les fonctions primaires de suivi de fond et d'évitement d'obstacle. Une seconde permet de passer de manière continue d'une fonction de tâche à l'autre ([60]). Ces commandes ont été mises au point en simulation et testées sur le véhicule expérimental VORTEX à l'Ifremer. L'annexe G résume ces travaux.
- Les véhicules munis d'ailes portantes à l'avant et à l'arrière présentent des directions d'évolution interdites et sont donc de type non-holonome. L'approche par fonction de tâche n'est plus directement utilisable pour synthétiser la loi de commande. Deux lois de commande en vitesse ont été étudiées, l'une basée sur une synthèse LQ calculée à partir du linéarisé tangent du modèle d'interaction véhicule/environnement et l'autre, plus robuste, est dérivée d'une fonction de Lyapounov. Un détecteur de saut sur l'altitude du véhicule permet de franchir les ruptures importantes dans le profil du fond ([61],[63]).
- Les véhicules munis uniquement d'ailes à l'arrière sont à la fois non-holonomes et à non-minimum de phase, et ne permettent pas de découpler les mouvements de tangage et de dérapage. Une commande non-linéaire, incluant la dynamique du véhicule, a été développée par migration d'une commande linéaire LQG/LTR vers une commande LQ utilisant un reconstructeur d'état non-linéaire. Cette commande, testée en simulation, est robuste face aux incertitudes des paramètres du modèle ([62]).

3.2 Un site expérimental pour ORCCAD

En dehors de ces activités proches de l'Automatique classique, il est rapidement apparu que la robotique sous-marine constitue un excellent terrain expérimental pour ORCCAD. On y trouve une assez grande variété de tâches à réaliser : navigation en suivi de trajectoire, navigation en évitement d'obstacles, suivi de fond ou de murs, asservissement au point fixe, manipulation en mode automatique ou téléopéré..., en utilisant des capteurs visuels ou acoustiques. Le véhicule étant libre et soumis à des perturbations (courant, ombilical...), il doit être commandé de façon permanente, les transitions entre TRs doivent donc s'exécuter en un temps minimal compatible avec la dynamique du processus. Enfin, s'agissant de robots opérant en milieu hostile, il est préférable de vérifier, à l'aide de méthodes formelles ou de simulations réalistes, que son comportement sera celui spécifié en particulier en ce qui concerne les propriétés critiques pour sa survie.

Les premières expériences menées en collaboration avec l'Ifremer ont permis de tester des missions simples avec le véhicule expérimental Vortex ([75]). Ce genre d'expérience a en particulier montré la difficulté de réaliser une mission complexe par utilisation directe de TRs dans un programme ESTEREL et a partiellement motivé le développement de la notion de procédure.

Dans le cadre du projet Union le véhicule a été équipé d'un bras manipulateur ; le site expérimental actuellement utilisé est décrit par la figure 2.

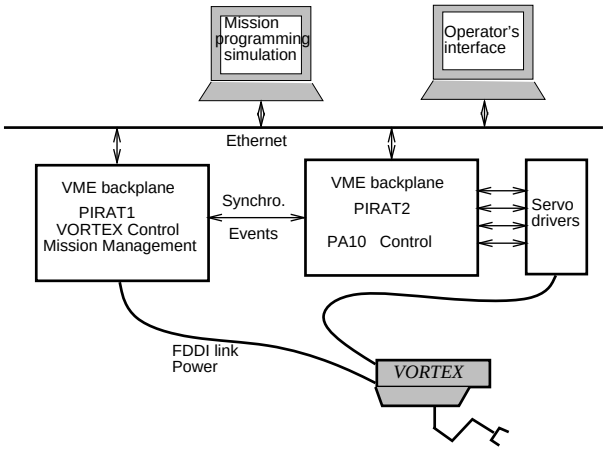


FIG. 2 – Site expérimental de l'Ifremer

Vortex est un petit ROV (Remotely Operated Vehicle) développé par l'Ifremer en tant que support de recherche en contrôle/commande d'engins sous-marins, du point de vue algorithmique de commande et architectures informatiques. Il est équipé de six propulseurs à hélice (quatre dans le plan horizontal et deux verticaux) permettant de commander tous ses axes sauf en roulis. Il est doté d'un ensemble classique de capteurs permettant de mesurer la plus grande partie de son état interne mais également de capteurs extéroceptifs (caméra CCD et ceinture de sondeurs acoustiques) permettant de réaliser des tâches référencées capteur telles que le suivi automatique de murs. Le bras électrique qui lui est attaché est doté de 7 degrés de liberté.

Pour des raisons historiques et pratiques, le véhicule et le bras possèdent chacun leur propre contrôleur temps réel (bacs VME), chargés du calcul des lois de commande. Le contrôleur logique chargé de la coordination des diverses TRs et PrRs réside sur l'un de ces bacs. Seuls des signaux de coordination sont échangés entre les deux bacs par une connexion dédiée. Ces contrôleurs sont d'autre part reliés aux consoles de programmation et de supervision par un réseau Ethernet.

Jusqu'à présent, les applications d'Orccad concernent des robots mobiles ou des bras manipulateurs fonctionnant de façon automatique. Dans le cadre du projet Esprit BRA UNION (Underwater Intelligent Operation and Navigation), nous nous proposons d'appliquer et d'adapter notre système pour la programmation de robots fonctionnant en mode mixte automatique/téléopéré. Le thème du projet est le contrôle/commande de systèmes d'intervention constitués d'un véhicule porteur associé à un ou plusieurs bras manipulateurs. Le type de tâche à exécuter est typiquement la réalisation d'une tâche de manipulation en mode téléopéré avec stabilisation automatique du porteur face à une structure immergée. La présence de plusieurs activités s'exécutant en parallèle (commande du bras, commande du véhicule, gestion de l'interface opérateur) pose de nouveaux problèmes de synchronisation. L'établissement de procédures fiables d'enchaînement de tâches se pose à nouveau de façon cruciale. Enfin, dans ce cas le flot de données entre l'opérateur et le contrôleur est beaucoup plus riche que dans le cas des robots autonomes. En fait, on trouve deux types de liaison entre le contrôleur et la station opérateur : i)une ou des liaisons haut débit permettant de transférer les trajectoires émises par le bras maître et de récupérer des informations sensorielles de type images ou retour d'effort ; ii) une liaison faible débit permettant de forcer les modes de marche du contrôleur et de récupérer des informations concernant l'état d'avancement de la mission (figure3). La structure même de l'architecture de contrôle n'est pas remise en cause.

Pour tester la méthodologie de programmation et de vérification d'ORCCAD nous avons imaginé le scénario de mission décrit figure 4.

Cette mission simule l'inspection d'un tuyau vertical immergé dans la piscine de l'Ifremer. Sa description en langage naturel est la suivante :

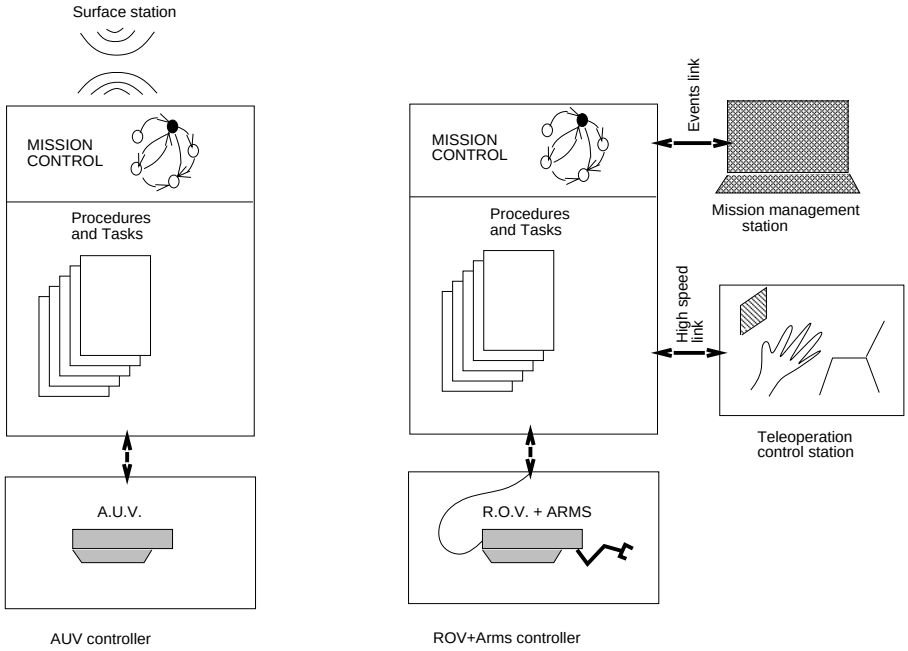


FIG. 3 – Architecture de commande de systèmes autonomes et de systèmes téléopérés

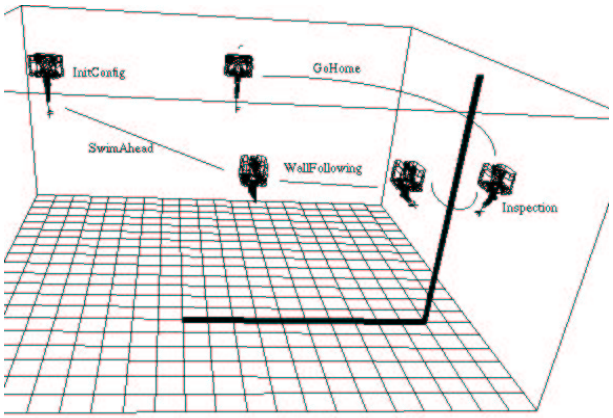


FIG. 4 – Scénario de mission d'inspection sous-marine

Après la phase de mise à l'eau et d'initialisation, le système est maintenu au point fixe jusqu'à ce que le bras soit passé de sa position initiale étendue à sa position de sécurité replié. Il est alors bloqué par ses freins pendant que le véhicule navigue à un cap et une profondeur prédéterminés, puis suivre les murs de la piscine jusqu'à atteindre son lieu de travail, en vue du tuyau à inspecter. Vortex est alors stabilisé à proximité du tuyau pour permettre au bras d'en inspecter certaines zones, d'abord en mode téléopéré puis en mode automatique. Lorsque l'opération est terminée le véhicule revient à sa position de départ, le bras étant à nouveau bloqué en position repliée.

Un certain nombre de contraintes sont imposées par l'opérateur : dans toutes les phases de la mission, la détection d'une entrée d'eau doit provoquer l'arrêt des opérations et une remontée d'urgence du système. A tout moment, un rappel de l'opérateur provoque le retour du véhicule au point de départ. Le véhicule ne peut naviguer qu'avec le bras replié et bloqué en position de sécurité. Enfin, pendant la phase d'inspection, le bras doit être bloqué en cas de perte d'information visuelle jusqu'à ce que Vortex ait retrouvé sa position d'équilibre. L'ensemble des TRs et PrRs nécessaires à la réalisation de cette mission est résumé figure 5. Certaines d'entre elles vont maintenant être utilisées pour illustrer les méthodes et outils développés dans ORCCAD.

Procedures	Sub−Proc. and Tasks	Main Events	Comments
InitCruiseConfig	Vortex: StationKeeping Pa10: GoToPark	Parked	Using absolute position Arm folded and locked
ReachWorkingArea	Vortex: SwimAhead WallFollowing Pa10: GoToPark	WallDetected CornerDetected	Navigation Using US sensors
DoInspection	Vortex: KeepStableBase Pa10: Pa10_Working Pa10BrakesOff	StopKeepStable Operator's calls	Stabilization in workspace Inspection with arm tip Brakes are released
GoHome	Vortex: GoToPoint Pa10: GoToPark	WayPointReached	Come back to initial position
Emergency	Vortex: GoUp Pa10: GotoPark	Hardware failure Water leak Operator's interrupt	Emergency recovery behaviour
KeepStableBase	Vortex:KeepStableCamera KeepStableUS	TargetFound TargetLost	Visual servoing Distance towalls servoing
Pa10_Working	Pa10: Pa10_TT_JS Pa10_TT_SE3 Pa10_Teleop	JointLimits Operator's calls	Traj. tracking in joint space Traj. track. in operationnal space Teleoperation mode
GoToPark	Pa10: Pa10_TT_JS Pa10BrakesOn	ParkPosReached ArmLocked	Arm folded Arm locked by brakes

FIG. 5 – Liste des TRs et PrRs utilisées dans la mission d'inspection

4 Conception et Vérification de Tâches-Robot

Les TRs modélisent et implantent les actions de base d'un système robotique. Leur activité est dominée par l'exécution périodique d'une loi de commande et leur conception relève d'un utilisateur ayant des compétences d'Automaticien. Il s'agit de lui fournir des outils lui permettant de spécifier, valider et générer des TRs de façon conviviale.

Une méthodologie de construction de TRs a été développée dans le cadre du projet Esprit ARMS ([70]). Une première maquette d'Interface Homme-Machine (IHM) implantant cette méthodologie a été développée par K. Kapellos et E. Castillo au cours de leur thèse ([36],[19]), en collaboration avec la société Aleph-Technologies. En dehors de ces deux mémoires de thèse, ces travaux sont décrits dans [71], dont une version raccourcie figure en annexe B. Nous allons maintenant décrire l'utilisation de cette méthodologie au travers de l'un des exemples les plus classiques en robotique : la commande en poursuite de trajectoire dans l'espace opérationnel d'un bras manipulateur rigide.

De façon informelle, nous désirons contrôler la position et l'orientation du bras dans l'espace des configurations des corps rigides SE^3 , l'origine étant le repère de la base du bras liée au véhicule.

Les pré-conditions incluent une initialisation correcte de toutes les tâches temps-réel composant la TR. La TR doit être interrompue si une butée articulaire est atteinte. La durée de la TR est celle de la trajectoire de référence. Une exigence de performance peut s'exprimer par le fait que l'erreur de poursuite doit être inférieure à un seuil ϵ à la vitesse maximum du bras.

4.1 Spécification en temps continu

S'agissant d'un système non-linéaire, la loi de commande est généralement spécifiée en continu, puis discrétisée. Les bras manipulateurs rigides sont des systèmes holonomes pour lesquels le but à atteindre peut être spécifié sous la forme d'une fonction de tâche ([56]) et peut être réalisé avec la structure de commande générique :

$$\Gamma = -k\hat{M}\left(\widehat{\frac{\partial e}{\partial q}}\right)^{-1} G\left(\mu De + \widehat{\frac{\partial e}{\partial q}}\dot{q} + \widehat{\frac{\partial e}{\partial t}}\right) + \hat{N} - \hat{M}\left(\widehat{\frac{\partial e}{\partial q}}\right)^{-1}\hat{f} \qquad (1)$$

où q est le vecteur des coordonnées articulaires, e la fonction de tâche (dans ce cas un vecteur d'erreur exprimé dans SE^3), Γ est le vecteur des couples articulaires, $\hat{M}$ et $\hat{N}$ représentent le modèle dynamique du bras et k, μ, G, D sont des gains. Les "chapeaux" indiquent que des modèles plus ou moins complets des différents termes peuvent être utilisés. Le concepteur de TR peut sélectionner ces modèles dans des classes d'objets partiellement instanciés via l'IHM et dessiner sa loi de commande sous forme d'un bloc-diagramme (figure 6). Il lui reste à terminer d'instancier ces *modules* algorithmiques en précisant les valeurs numériques à affecter aux différents paramètres du système e.g. les paramètres de Denavit-Hartenberg décrivant la cinématique du bras et les paramètres inertiels des différents segments.

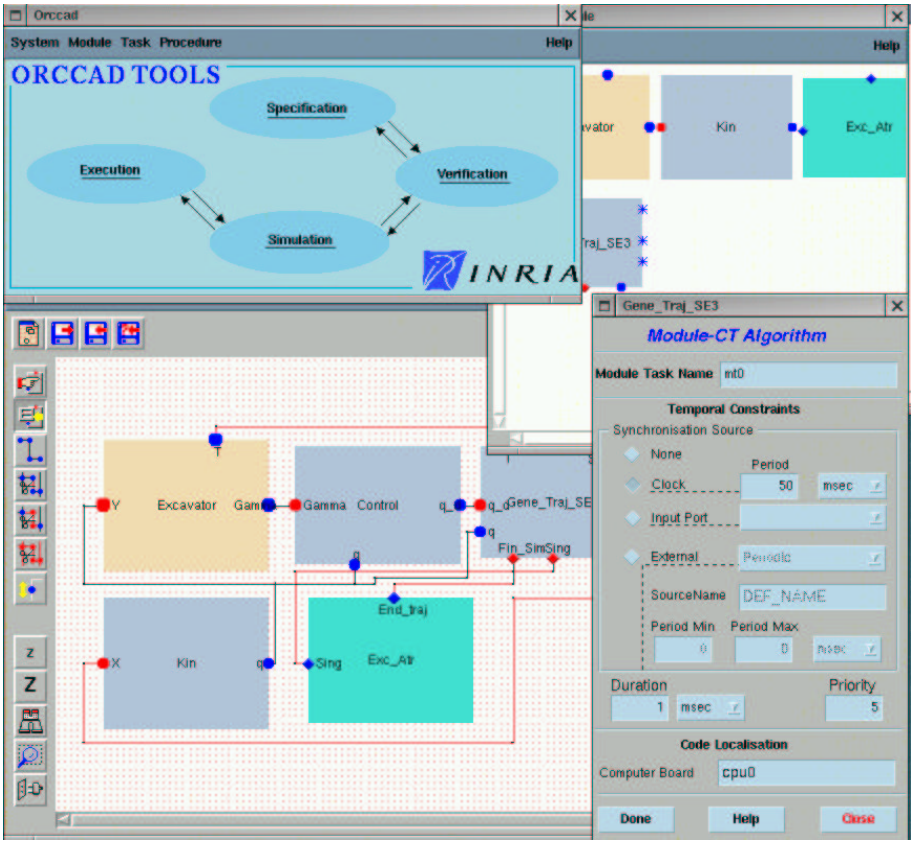


FIG. 6 – Interface Homme-Machine d'ORCCAD

En plus des modules algorithmiques sont également instanciés des *observateurs*, également périodiques, permettant de générer des événements en fonction d'observations mesurées dans le système. C'est par exemple le cas du module BUT-PA10 chargé de surveiller les butées articulaires : en cas d'approche d'une butée en dessous d'un seuil pré-défini, l'observateur émet un événement à destination

de l'*Automate Tâche-Robot* (ATR). L' ATR est réveillé par l'arrivée d'événements produits par les observateurs et émet en sortie des exceptions de différents types, ou signale la bonne fin de la TR en cas de terminaison nominale. Dans ce cas précis, il s'agirait d'une exception de type 2, aboutissant à l'arrêt de la TR en cours et à l'activation d'une autre TR choisie par la Procédure englobante.

Grâce au fort typage des événements manipulés par le comportement logique de la TR, le code ESTEREL de l'ATR est généré via une fenêtre spécialisée de l'IHM : le code ainsi généré a été prouvé être logiquement correct de façon générique ([36]).

L'ATR a un double rôle. Vu de l'intérieur de la TR, il gère l'état des tâches temps-réel qui la composent (création, activation, initialisation, suspension et destruction) et de ce point de vue joue un rôle un peu analogue à celui d'un régisseur d'ordonnancement [22], bien que nous ne fassions pas (pour le moment) de contrôle de charge du CPU. Il joue un rôle crucial lors des phases de transition entre TRs successives. Vu de l'extérieur de la TR, il présente la vue externe de son comportement logique sous forme d'un système à événements discrets exploité par la PrR englobante.

4.2 Addition de contraintes temporelles

Pour passer de cette description purement algorithmique a une représentation implantable sous forme d'un programme multitâches, il faut maintenant décorer les modules d'attributs temporels : les modules ainsi complètement instanciés deviennent des *Tâches-Module* (TM) dont la structure générale est donnée par la figure 7.

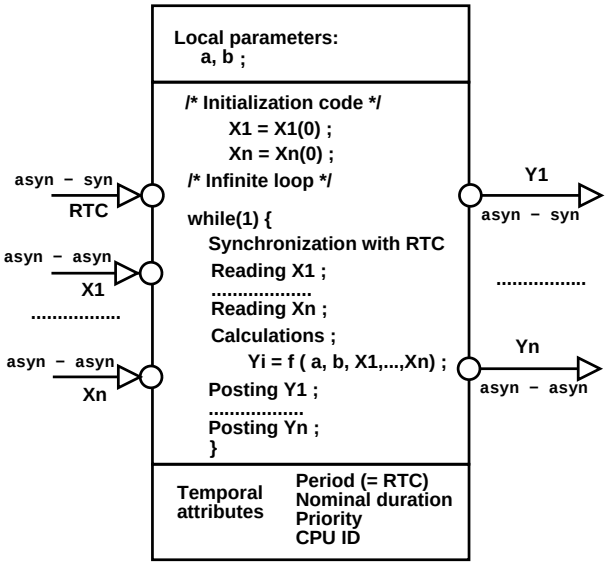


FIG. 7 – Structure d'une TM périodique

Une fois le code d'initialisation exécuté, elle se compose d'une boucle infinie dans laquelle tous les ports d'entrée sont lus dans l'ordre de déclaration, le code de calcul exécuté et les résultats écrits sur les ports de sortie dans l'ordre de déclaration. Les attributs temporels d'une TM sont :

- sa durée d'exécution présumée (à des fins de simulation et d'analyse d'ordonnançabilité) ;
- sa période (si une période explicite est déclarée) ;
- sa priorité ;
- son processeur d'appartenance ;
- le type de synchronisation utilisé par ses ports de communication unidirectionnels (producteur $\rightarrow$ consommateur).

Nous ne déclarons pas d'échéance (date de fin d'exécution au plus tard), qui est donc implicitement égale à la période. Ces attributs temporels vont permettre de gérer l'exécution des tâches temps-réel implantant les TMs sous le contrôle d'un système d'exploitation temps réel (RTOS).

Pourquoi les synchroniser ? A première vue, il parait plus simple de laisser les modules s'exécuter en

(pseudo) parallèle en espérant ainsi maximiser l'utilisation du ou des CPU. C'est une solution classique, implantée par exemple à l'aide de mécanismes de type client/serveur (e.g. [30]). C'est également l'approche la plus fréquemment utilisée dans les théories sur l'ordonnancement de tâches temps-réel ([32]). Malheureusement, cette approche ne tient pas compte des dépendances entre tâches pourtant primordiales quand aux performances de la boucle de commande.

Pour un algorithme de commande donné, les performances effectives vont dépendre de deux données temporelles essentielles : la période d'échantillonnage et la latence, i.e. le délai séparant l'instant d'une mesure de celui où la commande calculée à partir de cette mesure est envoyée aux actionneurs. Il ne sert pas à grand chose d'échantillonner très vite la loi de commande si la latence vaut plusieurs fois la cadence d'échantillonnage ([56]). Or, la latence dépend étroitement de l'ordre d'exécution des TMs qui doivent autant que possible respecter les dépendances de données entre modules.

Ne pas imposer de synchronisations sur les communications revient à laisser l'ordonnanceur décider de lui même l'ordre d'exécution des modules. Ceux ci vont s'exécuter dans l'ordre des priorités qui leur ont été affectées. Si des modules ont la même priorité, leur ordre d'exécution relatif sera déterminé par quelque mécanisme interne du RTOS, par exemple l'ordre dans lequel ils auront été saisis à l'écran ou l'ordre alphabétique des identificateurs...Le cas le pire (réellement vécu au cours de l'une des premières expérimentations du robot mobile Anis du projet Icare) est illustré par la figure 8 : toutes les tâches ayant la même priorité, l'ordre d'exécution des n modules composant l'algorithme de commande peut être complètement renversé. Même si la période d'échantillonnage de la commande est respectée, la valeur de la latence peut atteindre presque n fois celle de la période, conduisant à une instabilité de la commande. Soulignons que, tous les modules ayant été déclarés avec la même période dans cet algorithme, l'application pure et simple de la politique d'ordonnancement optimale pour un ordonnanceur à priorités fixes *Rate-Monotonic* [66] (la priorité est d'autant plus grande que la période est petite) peut aboutir à ce résultat.

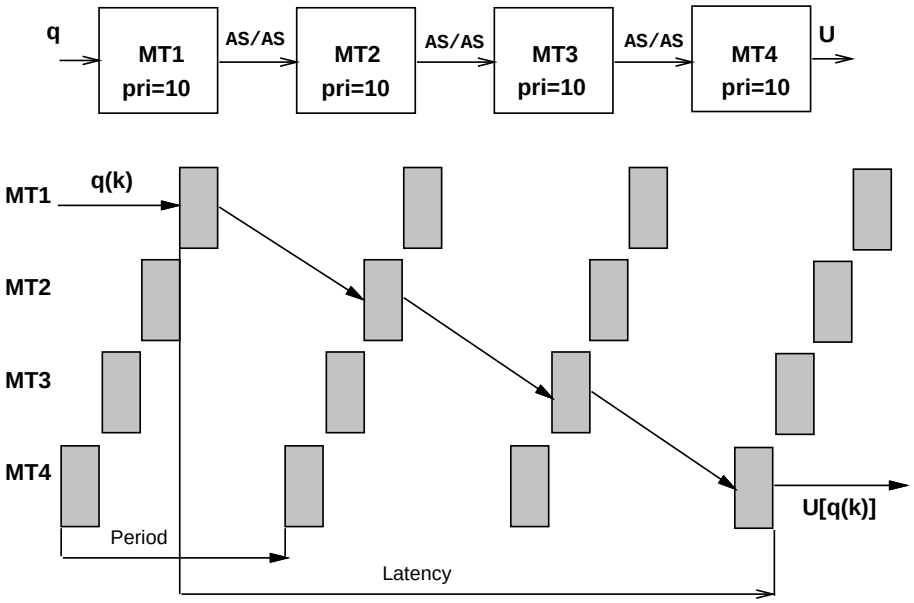


FIG. 8 – Un pipe-line inversé : a)Spécification - b)Ordre d'exécution

La structure d'un algorithme de commande en robotique ne se limite souvent pas à un seul "pipe-line" de modules comme dans l'exemple précédent, mais comporte souvent plusieurs chemins de calcul, plus ou moins imbriqués et pouvant s'exécuter à des rythmes différents. Ainsi, dans les figures 9 et 10, nous comparons en simulation deux algorithmes classiques pour la commande en poursuite de trajectoire articulaire de bras manipulateurs : un classique PID décentralisé à gains fixes et une commande dite "dynamique" (Computed Torque Control) où le module supplémentaire MOD.DYN calcule des modèles explicites de la matrice d'inertie M du bras et du vecteur de forces centrifuges, de Coriolis et de gravité N s'exerçant sur le robot (Cet exemple est détaillé en annexe B).

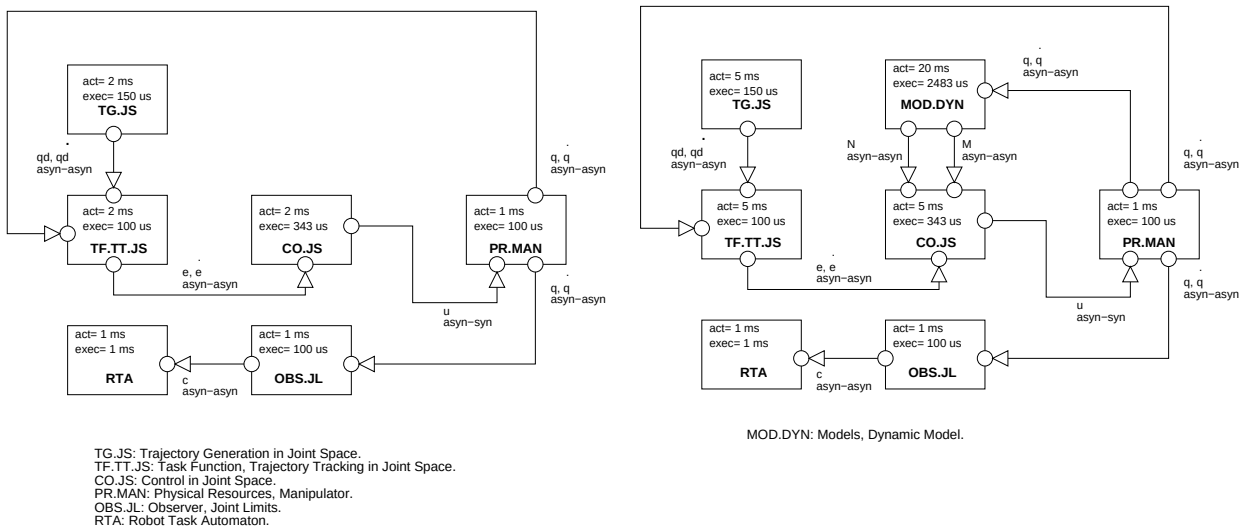


FIG. 9 – Bloc-diagrammes : a)PID à gains fixes - b)Commande dynamique

Conformément à ce que l'on attend de la théorie de la commande échantillonnée, on constate sur la figure 10a qu'une augmentation de la cadence d'échantillonnage permet de réduire l'erreur de poursuite, même sans modifier la valeur des gains. Sur la figure 10b, on constate que, comme on l'espérait sans que cela soit prouvable par la théorie, le calcul explicite du modèle dynamique du robot permet d'améliorer la performance de poursuite tout en diminuant substantiellement la taille des gains[1]. On constate également, sans trop de surprise, que le module MOD.DYN peut être exécuté à une cadence beaucoup plus lente que la boucle de commande principale avec très peu d'incidence sur la performance obtenue : ceci peut permettre d'optimiser l'utilisation de la puissance de calcul disponible, mais ce module s'exécutant à une cadence plus lente ne doit évidement pas se synchroniser avec les autres.

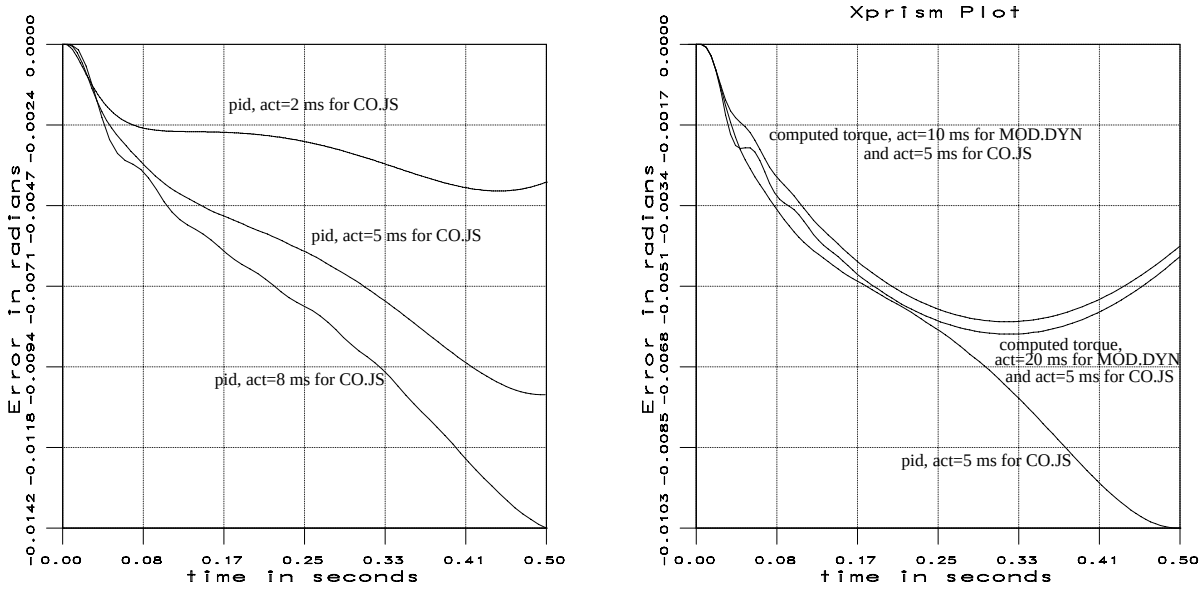


FIG. 10 – Erreurs de poursuite : a)PID avec différentes cadences - b)Comparaison PID/commande dynamique

Actuellement les ports d'ORCCAD peuvent mettre en oeuvre quatre types de synchronisation

[1]Résultat valable pour ce robot, cette trajectoire et cette vitesse ; s'agissant d'un système non-linéaire, on ne peut en tirer de conclusion générale...

différents entre TM productrice et TM consommatrice :

- ASYN/ASYN : la communication est uniquement fonctionnelle et ne bloque ni le producteur ni le consommateur ;
- SYN/SYN : c'est le rendez-vous classique ;
- ASYN/SYN : la TM productrice n'est jamais bloquée par la communication, la TM consommatrice soit lit une nouvelle donnée non encore consommée, soit est bloquée jusqu'à production d'une nouvelle donnée ;
- SYN/ASYN : un mécanisme symétrique du précédent, où la TM consommatrice n'est jamais bloquée et la TM productrice est bloquée jusqu'à la prochaine demande sauf si celle ci est apparue depuis la dernière lecture.

De façon a rendre le service de synchronisation transparent vis a vis de la machine cible (distribuée ou non), nous modélisons les synchronisations à l'aide des sémaphores privés répartis développés dans sa thèse par M. Mejia ([44]). Ces services sont obtenus en combinant, sur chaque paire de ports reliés, l'ordre des opérations *signaler* et *attendre* avec le type de sémaphore associé à chaque port (traditionnel, fugace ou figé). Ces protocoles de communication avaient été spécifiés et validés à l'aide du langage ESTEREL . En fonction de la machine cible, ils sont implantés à l'aide des outils temps réels adéquats : sémaphore classique pour une exécutif centralisé, sémaphore partagé si le contrôleur est réparti sur un bus de fond de panier, "sockets" si la répartition est faite par l'intermédiaire d'un réseau local.

4.3 Validation d'une Tâche-Robot

4.3.1 Vérification du squelette de synchronisation

L'activation d'une TR provoque l'exécution d'un réseau de TMs périodiques synchronisées. Le couplage entre ces tâches peut être plus ou moins lâche en fonction des protocoles de synchronisation utilisés. Les simulations de boucles de commande réalisées à l'aide de l'IHM ont montré l'utilité de calculs multicadences et l'importance de la façon de synchroniser les tâches de calcul quand aux performances des lois de commande obtenues ([19]). Cependant, une spécification maladroite du squelette de synchronisation de la TR ou des attributs temporels affectés aux TMs peut provoquer

- des blocages structurels, ne dépendant que de l'entrelacement des synchronisations, par exemple dus à une circularité de communication impliquant des rendez-vous ;
- des incohérences de spécification temporelle, dépendant de la valeur numérique des attributs temporels et pouvant également conduire à des comportements fautifs tels que le dépassement d'échéance.

Il est donc souhaitable de pouvoir effectuer une validation formelle des choix faits au moment de l'instanciation des TMs, avant simulation. Diverses techniques ont été abordées, en particulier l'utilisation de langages synchrones et d'automates temporisés. Des résultats significatifs ont été obtenus à l'aide de modèles des TMs sous forme de réseaux de Petri temporisés (RdP).

En fait, les modèles obtenus, tel que celui représenté figure 11, sont des graphes d'événement (une place n'a qu'une transition de sortie et une transition d'entrée, ce qui dénote un système déterministe), jouissant de propriétés structurelles intéressantes [46]. En particulier, le graphe est vivant (et donc exempt de blocage) si tout circuit orienté est marqué par au moins un jeton dans le marquage initial. La détection des blocages structurels s'obtient donc par l'analyse de la vivacité des invariants de places [73]. Par exemple, il est facile de voir sur la figure 11 que le graphe décrivant les trois tâches communicantes se bloque puisque le circuit $s_4 = \{p_{15}, p_{13}, p_{14}, p_{18}, p_8, p_{16}, p_5\}$ n'est pas marqué dans le marquage initial, correspondant à la fin de l'initialisation des TMs et au début de l'exécution de la loi de commande. Un logiciel écrit au cours de la thèse d'E. Castillo génère automatiquement la matrice d'incidence décrivant le RdP correspondant au réseau de TMs décrit par l'IHM d'ORCCAD et réalise l'analyse des invariants de places pour conclure à l'absence de blocages en temps polynômial [69].

Une extension du modèle des TMs à l'aide de réseaux de Petri temporisés permet de mettre en évidence certaines incohérences de spécification temporelle par recherche d'un régime stationnaire. En fait, comme c'est souvent le cas dans l'analyse de RdP de structure non triviale, il s'agit d'explorer

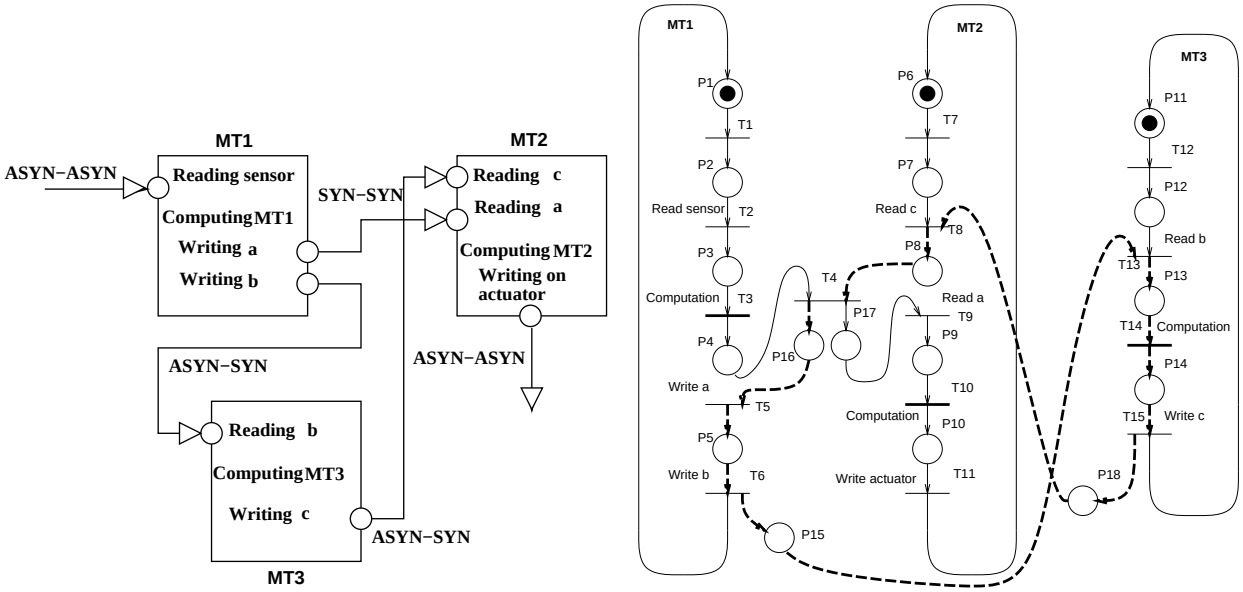


FIG. 11 – Système de trois TMs avec blocage structurel

le graphe d'atteignabilité jusqu'à trouver un régime stationnaire à moins d'être préalablement piégé dans un état puits. Enfin, cette technique se limite à la détection d'un cas particulier d'incohérence temporelle, celle où un module explicitement périodique dépasse son échéance. Nous verrons dans la conclusion que des techniques algébriques peuvent surmonter cette difficulté.

Enfin, moyennant quelques hypothèses peu contraignantes dans le cadre spécifique d'ORCCAD où l'activité des TRs est dominée par l'exécution de modules périodiques, une condition suffisante portant sur la nature et le nombre des ports de synchronisation permet de construire un squelette de synchronisation non seulement exempt de blocages et d'incohérences temporelles mais permettant également de respecter "au mieux" les exigences de la commande en boucle fermée en minimisant les latences de calcul entre mesures et commande sur les chemins critiques du graphe de calcul. La figure 12 montre la commande dynamique d'un bras manipulateur ainsi partiellement synchronisée ainsi que le modèle RdP correspondant, qui se trouve éclaté en plusieurs composantes connexes. Ces conditions sont très simples à vérifier et devraient donc facilement s'intégrer dans les méthodes de l'IHM (ces méthodes sont motivées et détaillées en annexe D).

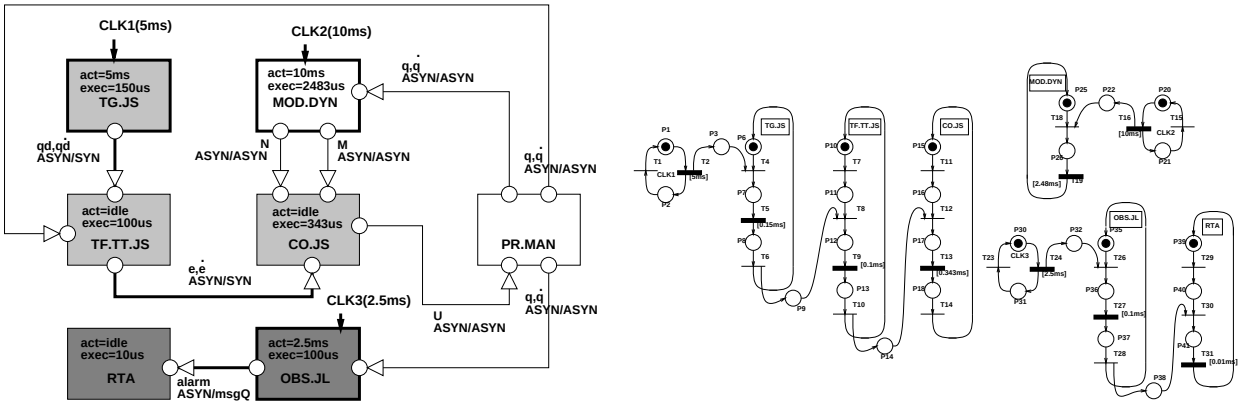


FIG. 12 – Une TR partiellement synchronisée et son modèle réseau de Petri

4.3.2 Vérification de performances par simulation

Les systèmes robotisés sont des systèmes dynamiques non-linéaires complexes, commandés en temps discret par des programmes multitâches souvent répartis et multicadencés. Il est généralement impossible de prédire par l'analyse des résultats quantitatifs sur les performances du système complet. La simulation reste pour cela un outil indispensable, encore faut il que ces simulations soient suffisamment *réalistes* pour rendre des résultats exploitables. Le logiciel Simparc, principalement réalisé par C. Astraudo et J.J. Borrelly dans le cadre du projet Esprit II ARMS [5], a été conçu dans ce but. Il permet de simuler en parallèle l'exécution des programmes sur la machine cible et l'évolution du processus commandé. L'analyse des résultats permet de valider une implantation particulière des programmes, de valider le choix de certains modèles utilisés dans la loi de commande, d'effectuer un pré-réglage de gains ou encore d'effectuer des choix d'architecture et de dimensionnement sur le processus lui même.

Les différents types d'objets manipulés par Simparc sont :

- Une description macroscopique du matériel, i.e. des processeurs, mémoires, bus, convertisseurs, horloges temps-réel..., auxquels peuvent être associés divers attributs ;
- Un modèle du système contrôlé écrit sous forme d'équations différentielles ordinaires. Notons que l'intégration de certaines classes du logiciel Dynamechs [43] dans le modèle de véhicule sous-marin développé en commun avec l'Ifremer nous permet de disposer d'un modèle dynamique de simulation relativement complet et générique de véhicule sous-marin libre muni de bras manipulateurs ;
- De modèles de capteurs à base de lancer de rayon, permettant de simuler des capteurs de distance tels que sondeur acoustique, sonar multi-faisceaux ou caméra ;
- De modèles d'environnement décrits sous forme de facettes ;
- D'un modèle minimal de RTOS (tâches préemptibles, sémaphores, mémoire partagée, file de message) permettant de simuler une structure de programme aussi proche que possible du programme téléchargeable finalement généré.

La figure 13 décrit ainsi l'architecture matérielle du contrôleur du système Vortex/Pa10, sur laquelle l'architecture logicielle de commande, spécifiée grâce à l'IHM d'ORCCAD peut être projetés à des fins de simulation.

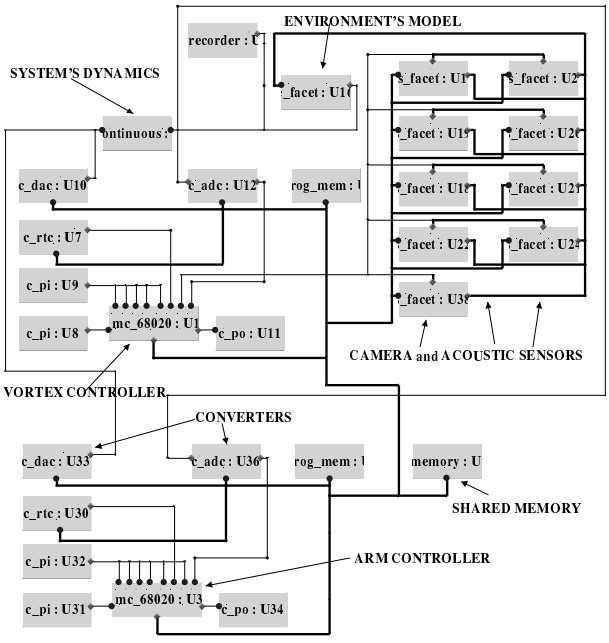


FIG. 13 – Schéma de simulation du contrôleur Vortex/Pa10

Dans une première phase, ces simulations peuvent permettre de valider la faisabilité d'un nouvel

algorithme de commande et de choisir des valeurs convenables des attributs temporels des TMs et de la taille des gains assurant le respect d'indices de performance requis par l'utilisateur. En l'absence de théorie sous-jacente, trouver des valeurs efficaces pour ces attributs temporels reste difficile et seul ce genre de simulations réalistes permettent d'orienter les choix du concepteur de TR. Les résultats de simulation permettent, sans que cela soit automatisé pour le moment, de dimensionner la machine cible et de donner de précieuses indications sur les possibilités du robot lui même e.g. de choisir une résolution convenable pour les convertisseurs de mesure analogiques/numériques. Dans une dernière phase, la simulation prend en compte un modèle d'architecture matérielle et logicielle identique à l'architecture du système réel de façon à valider une implantation particulière.

L'exemple de la figure 14 montre les angles de roulis et tangage de Vortex lors d'un mouvement rapide du bras Pa10, où les deux premiers axes tournent de 90 degrés en 3 secondes. Le bras est commandé en position articulaire alors que le véhicule est stabilisé dans un coin de la piscine grâce à une boucle de commande utilisant les sondeurs acoustiques latéraux. Compte tenu des limites du matériel disponible, les cadences d'échantillonnage ont été fixées à 100 ms pour le véhicule et 10 ms pour le bras. Les 4 sondeurs latéraux utilisés sont cadencés à 360 ms avec un décalage de 90 ms entre eux. Cette campagne de simulation a rapidement montré que la performance de stabilisation (taille de l'erreur transitoire) est principalement limitée par la faible cadence d'échantillonnage des capteurs acoustiques : une amélioration de cette performance passe donc d'abord par une refonte de l'électronique des sondeurs plutôt que par une augmentation de la puissance de calcul ce qui constitue une information précieuse pour le concepteur du système.

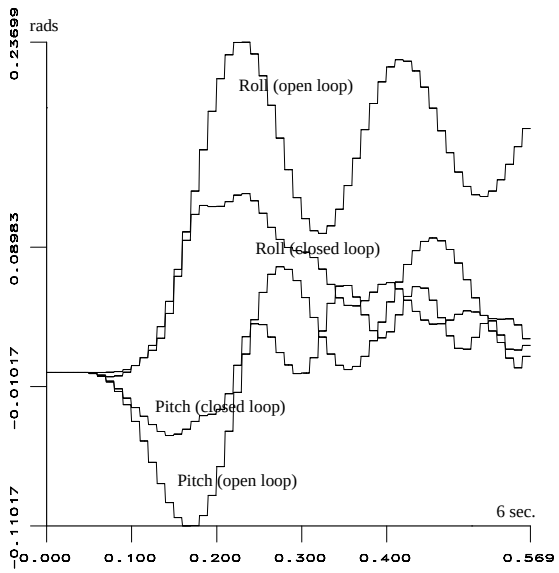


FIG. 14 – Angles de roulis et tangage de Vortex au cours d'un mouvement rapide du bras

Une fois que les TRs nécessaires à une application donnée ont été programmées et validées, elles peuvent être hiérarchiquement composées pour réaliser des actions de plus haut niveau jusqu'à obtenir une mission complète.

5 Conception et Vérification de Procédures

La description informelle de mission donnée en section 4 nous a naturellement conduit à la décomposer en cinq PrRs principales, qui seront elles mêmes composées pour obtenir le programme de la mission complète :

- INITCRUISECONFIG prépare le système pour sa mission d'inspection ;
- REACHWORKINGAREA permet au véhicule de rejoindre son site de travail : elle séquence une tâche de navigation à cap constant suivie d'une tâche de suivi automatique de murs jusqu'à

atteindre le site voulu ;
- DoInspection est la procédure de travail : l'extrémité du bras, en mode téléopéré puis automatique, inspecte certaines portions de tuyau pendant que la base doit rester activement stabilisée ;
- GoHome ramène le véhicule à son point de départ et le prépare à être sorti, le bras étant replié en position de sécurité ;
- Emergency traite les exceptions globales (type 3), en particulier la remontée d'urgence en cas de détection de fuite ou de dépassement de la profondeur maximum autorisée.

La procédure DoInspection nécessite une coordination fine entre le bras et le véhicule et nous servira de support pour illustrer la méthodologie de conception et de vérification des PrRs.

5.1 Conception d'une procédure

DoInspection est composée de deux sous-procédures principales : Pa10SafeSe3 gère les mouvements du bras dans différents modes (téléopéré ou automatique) tandis que KeepStableBase gère la stabilisation du véhicule : examinons d'abord cette dernière PrR plus en détail.

La procédure KeepStableBase Nous disposons de deux moyens pour stabiliser le véhicule dans son site de travail : l'asservissement visuel sur le tuyau à inspecter [55] et l'asservissement par rapport aux murs en utilisant les sondeurs latéraux [80]. La fréquence d'échantillonnage de l'asservissement visuel étant très supérieure à celle des capteurs acoustiques, nous supposons que cette loi de commande sera la plus robuste face aux perturbations apportées par les mouvements du bras sur le véhicule libre. Cependant, un mouvement trop rapide de Vortex peut aboutir à un décrochage de l'asservissement visuel : dans ce cas, on utilise les sondeurs pour re-stabiliser le véhicule jusqu'à remise en service de l'asservissement visuel. Cette situation, où deux ou plusieurs TRs concourent à la poursuite d'un but commun (ici, la stabilisation du véhicule), en étant activées en fonction du contexte et en étant exceptions l'une de l'autre, est caractéristique des procédures d'Orccad. La spécification de cette procédure est donnée par la figure 15. La TR principale KeepStableCamera s'exécute sauf en cas de perte du signal vidéo qui lève une exception de type 2. La TR secondaire KeepStableUs s'exécute alors, le contrôle étant redonné à la tâche primaire lorsque le signal vidéo est à nouveau disponible pour réaliser la stabilisation du véhicule en asservissement visuel.

```
Name : KeepStableBase
Pre-cond : OkInit[30ms], SignalIsValid[30ms]
Main programme :
      KeepStableCamera
T2 exceptions : TargetLost
      do
         KeepStableUs
      until targetFound
T3 exceptions : WaterLeak, MaxDepth
Post-cond : StopKeepStable [10ss]
```

Fig. 15 – Spécification de la PrR KeepStableBase

Dans cette spécification, l'utilisateur n'a pas à se préoccuper du mécanisme fin de gestion de la transition entre TRs successives qui est automatiquement généré. Ce mécanisme, décrit dans la figure 16, doit assurer :

- que l'exécution des deux lois de commande soient mutuellement exclusives
- que la transition soit la plus courte possible, idéalement de l'ordre de grandeur de la période d'échantillonnage

Il y a recouvrement entre la terminaison de la première TR et le début de la deuxième : la TR en cours de démarrage s'initialise pendant que celle en cours exécute une procédure de terminaison.

L'arrivée des pré-conditions de la deuxième TR provoque alors, via le programme de la PrR, l'arrêt de la première loi de commande immédiatement suivi du démarrage effectif de la deuxième. Ce processus peut être vérifié en observant la *vue abstraite* de l'automate de la procédure, réduit aux seuls signaux significatifs (figure 17). On peut en effet constater que l'événement signalant la fin d'une commande (préfixé par ?CmdStop) précède toujours l'événement d'activation (préfixé par !Activate) et que ces deux événements apparaissent dans une même transition de l'automate. Le comportement du code généré a d'autre part été validé grâce à l'outil d'analyse WinView présent dans l'environnement de mise au point de VxWorks ([79]).

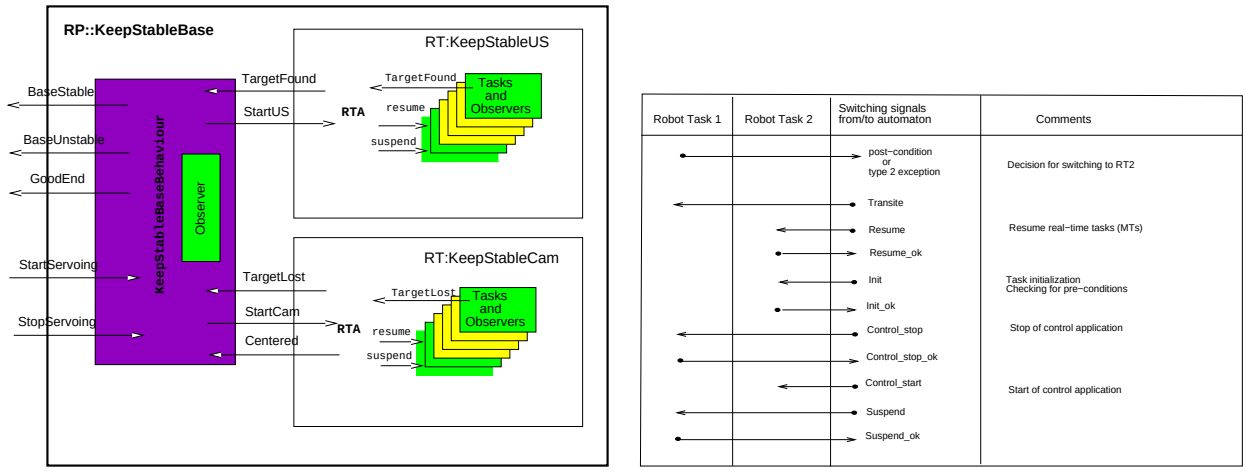


FIG. 16 – Mécanisme de transition entre TRs

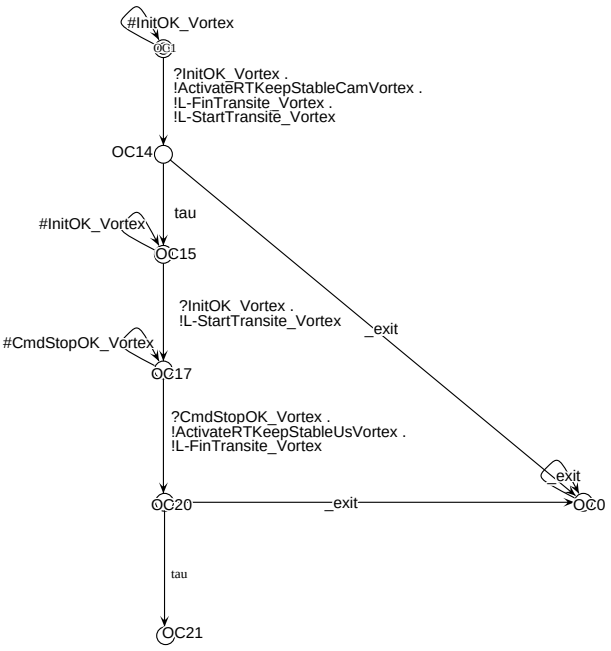
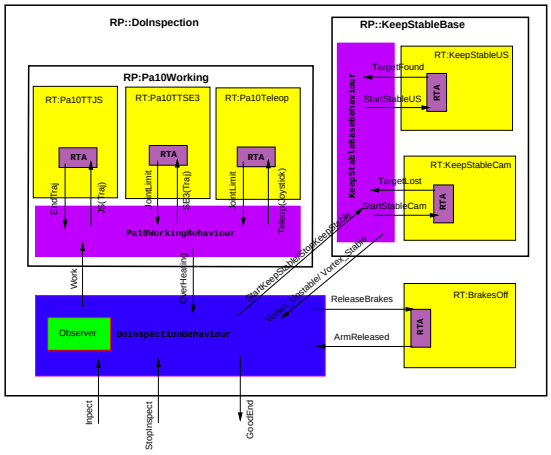


FIG. 17 – Vue abstraite d'une transition entre TRs

La gestion des différents modes de commande du bras est spécifiée d'une façon analogue dans la PrR PA10SAFEMOVESE3.

La procédure DOINSPECTION Il s'agit maintenant de synchroniser les commandes du bras et du véhicule porteur pour réaliser la tâche d'inspection de la structure sous-marine : les deux PrRs précédemment définies et validées vont devoir s'exécuter en parallèle. La spécification informelle est la suivante : une fois le véhicule stabilisé face à la structure en asservissement visuel, le bras inspecte deux

points de la structure avec à chaque fois une phase d'approche en commande articulaire puis une phase d'approche fine en commande dans l'espace opérationnel. En cas de décrochage de l'asservissement visuel, le bras doit être bloqué pendant que le porteur se re-stabilise grâce aux capteurs acoustiques, repasse en asservissement visuel dès que le signal vidéo est de nouveau utilisable et revient se centrer devant la structure. L'opération du bras ne peut reprendre qu'a ce moment là. La spécification de cette PrR est donnée figure 18. Cette spécification est écrite avec le prototype du langage "métier" MAESTRO, en cours de développement par N. Turro et È. Coste-Manière [84], et expanse le programme ESTEREL correspondant à la PrR.




```
PAR{
SEQ(do
        Pa10BrakesOn
    until  BaseStabilized;
    repeat 2 times
        Pa10SafeMoveJS;
        Pa10SafeMoveSe3;
    end repeat
    ;
    do
        Pa10SafeMoveJs
    until  ParkingPositionReached
    emit  InpectionOK
    ) ;
```

```
SEQ(do
        loop
            BaseSearchTarget;
            emitBaseStabilized
            KeepStableBase;
        end loop
        until  (InpectionOK)
    ) }
```

FIG. 18 – Spécification de la PrR DOINSPECTION

5.2 Vérification formelle d'une procédure

La spécification d'une PrR fait intervenir à la fois : (1) la composition logique d'événements typés et d'actions robotiques; (2) des caractéristiques temporelles qui expriment essentiellement des temporisations associées à des attentes d'événements. L'analyse complète d'une PrR nécessite donc la prise en compte de ces deux aspects (logique et temporelle).

Pour les aspects logiques l'approche la mieux appropriée car conduisant à des analyses globales est l'approche comportementale. Celle-ci suppose cependant que les caractéristiques temporelles quantitatives soient abstraites en signaux logiques. Une autre approche doit donc être utilisée pour l'analyse des aspects temporels liés à une PrR. Pour cela les techniques de "model-checking" symboliques, basées sur une modélisation des systèmes à l'aide d'automates étendus dans lesquels le temps est représenté explicitement, sont les plus appropriées. L'aspect vérification temporelle a été uniquement traité à l'U.R. Rhône-Alpes et les principaux résultats apparaissent dans l'annexe C.

De manière similaire à la TR, la spécification d'une PrR est automatiquement traduite en un programme ESTEREL de manière à pouvoir analyser l'automate obtenu par compilation avec MAUTO. Cependant, il n'est malheureusement plus possible ici d'assurer par construction la correction d'une PrR, son comportement n'étant plus aussi figé que celui de la TR. Seul l'utilisateur peut décider du bien fondé de son comportement. La vérification formelle d'une PrR est donc un processus interactif qui tient toute sa place durant la phase de spécification. Néanmoins, cette phase de vérification doit être le plus possible automatisée afin de faciliter la tâche de l'utilisateur non expert en vérification formelle. Pour cela nous avons identifié des classes de propriétés qui guident l'utilisateur dans sa démarche en lui proposant pour chacune de ces classes une méthodologie précise à appliquer. Ces différentes classes sont les suivantes :

- Propriété de **sûreté** : elle exprime le fait que toute exception fatale (type 3) est correctement prise en compte et qu'une occurrence de ce type d'événement aboutit bien à l'exécution de la procédure d'exception prévue. Pour la vérifier, nous construisons une action abstraite comme par exemple :

$$Erreur = /Waterleak\,?\ and\ not\ /Ascent\,!$$

spécifiant dans le cas de cet exemple précis que l'occurrence de l'événement signalant la détection d'une fuite non suivie de la mise en oeuvre de la procédure de remontée d'urgence est un comportement fautif. L'abstraction de la procédure de mission par ce critère est alors calculée par Mauto. La propriété est vérifiée par l'absence de l'action *Erreur* dans l'automate résultant.

- Propriété de **vivacité** : elle exprime le fait que la PrR peut toujours atteindre son but dans une exécution nominale (i.e. sans émission d'une exception de type 3). Elle est prouvée de façon analogue à la propriété de sûreté : un critère abstrait est construit avec le signal *Bonne_Fin* émis à la fin de toute exécution réussie d'une PrR. Le résultat de l'abstraction du comportement de la procédure par ce critère doit être équivalent à un automate contenant la seule action *Bonne_Fin*.
- **Détection de conflits** : Il s'agit de vérifier qu'au cours de l'évolution de la PrR il n'existe pas de conflits dans l'utilisation des ressources du système, qu'elles soient physiques ou informatiques. C'est en particulier l'exemple du paragraphe précédent concernant la transition entre TRs. La technique consiste à réduire l'automate de la PrR en ne laissant apparaître que les signaux significatifs et à observer visuellement le résultat.

Ces propriétés sont génériques et peuvent être vérifiées automatiquement au travers de l'IHM qui construit elle même les critères abstraits nécessaires. D'autres propriétés, plus spécifiques à une application donnée, peuvent également être vérifiées mais de façon moins systématique.

- **Conformité du comportement logique** Il s'agit de vérifier que le comportement programmé est bien cohérent avec la spécification donnée par l'utilisateur. Des vues abstraites peuvent être construites à différents niveaux d'abstraction (par exemple au niveau des TRs) et observées pour conclure à la conformité du comportement logique du programme avec la spécification de l'utilisateur. Un exemple en est donné dans la partie supérieure de la figure 19 où l'automate de la procédure DoInspection a été réduit en ne faisant apparaître que les actions KeepStableBaseUS et Pa10SafeMoveSe3. L'observation du résultat montre que ces actions ne s'exécutent jamais simultanément ce qui est bien le comportement voulu.
  Inversement, de telles vues abstraites peuvent servir d'aide à la spécification : des exemples en sont données en annexe C

Une aide à la construction de critères abstraits permettant de vérifier ces propriétés génériques est proposée dans l'interface graphique d'Orccad.

Enfin, ces techniques ne permettent de vérifier que des propriétés logiques des PrRs. Il est également intéressant d'évaluer par simulation le comportement du système commandé lors des transitions de tâches. Pour ce faire, les programmes correspondants aux diverses lois de commande ainsi que celui du contrôleur logique sont assemblés dans un programme de simulation Simparc. La partie inférieure de la figure 19 montre ainsi l'évolution simulée de la position du véhicule (au dessus) et du bras (en bas) pendant une exécution de DoInspection. On constate que la transition est suffisamment douce pour permettre la ré-acquisition du signal vidéo et que le bras n'évolue bien que lorsque le véhicule est centré en asservissement visuel devant la structure à inspecter.

6 Bilan et perspectives

Initialement, Orccad a été conçu pour la programmation de lois de commande pour bras manipulateurs rigides. Le robot AID du projet Icare a ainsi été programmé et contrôlé à l'aide de lois de commande issues de la théorie des fonctions de tâches telles que le suivi de trajectoire dans l'espace opérationnel et la commande référencée vision, en utilisant les classes de modules algorithmiques correspondant à ce type de robots ([19]). Le système a également été testé pour la programmation de robots très différents comme le robot mobile a roues Anis du projet Icare, le véhicule sous-marin Vortex de l'Ifremer et les véhicules du projet Praxitèle, ce qui illustre la *généricité des concepts* sur lesquels Orccad a été fondé. Ces expérimentations ont en particulier montré l'excellence de l'approche commande référencée capteurs pour la navigation de robots mobiles en environnement encombré [74]. La commande par retour d'état instationnaire a pu être testée en vraie grandeur. Enfin, la possibilité d'utiliser Esterel en tant que langage de programmation de missions a également été mise en évidence [75]. D'autres utilisations sont encore envisagées pour la programmation de la commande de robots

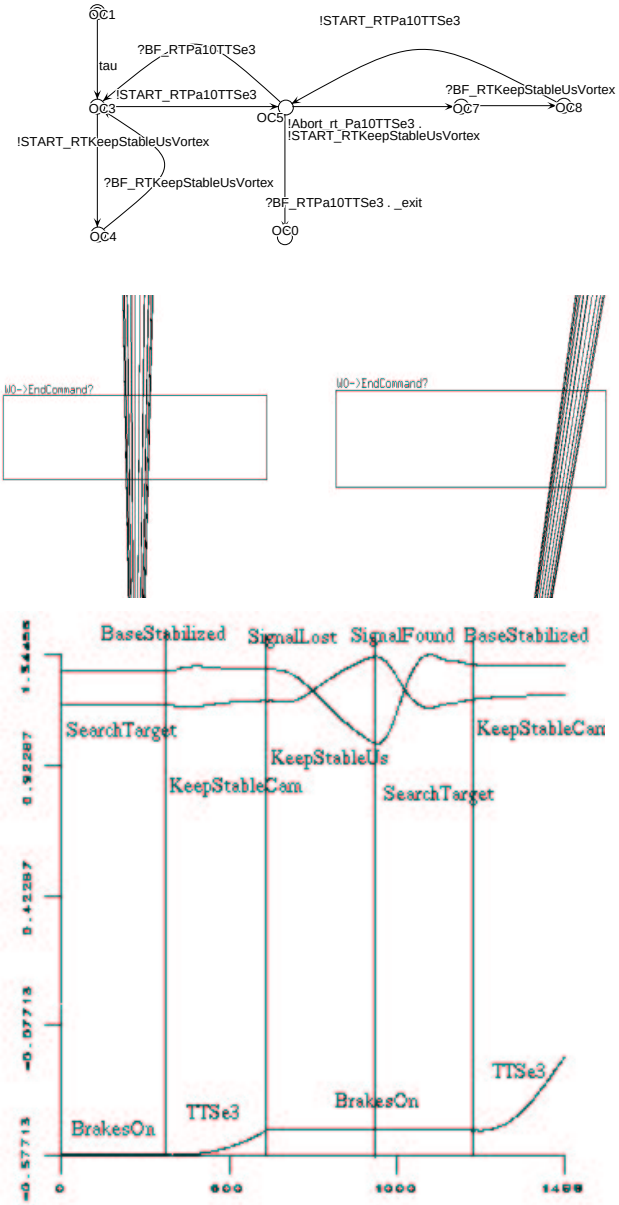


FIG. 19 – La PrR DoInspection : a) Vue abstraite au niveau des TRs - b)Simulation des transitions

bipèdes (projet BIP). Ces premières expériences, ainsi que le désir de s'adapter à de nouveaux terrains d'applications, ont cependant mis en évidence certaines faiblesses et manques du premier prototype.

Nouvelle version de l'IHM Sur la base des retours d'expériences accumulés durant ces expérimentations, une nouvelle version d'IHM a été développée par K. Kapellos et R. Pissard [81]. Cette interface intègre les outils suivants (figure 20) :

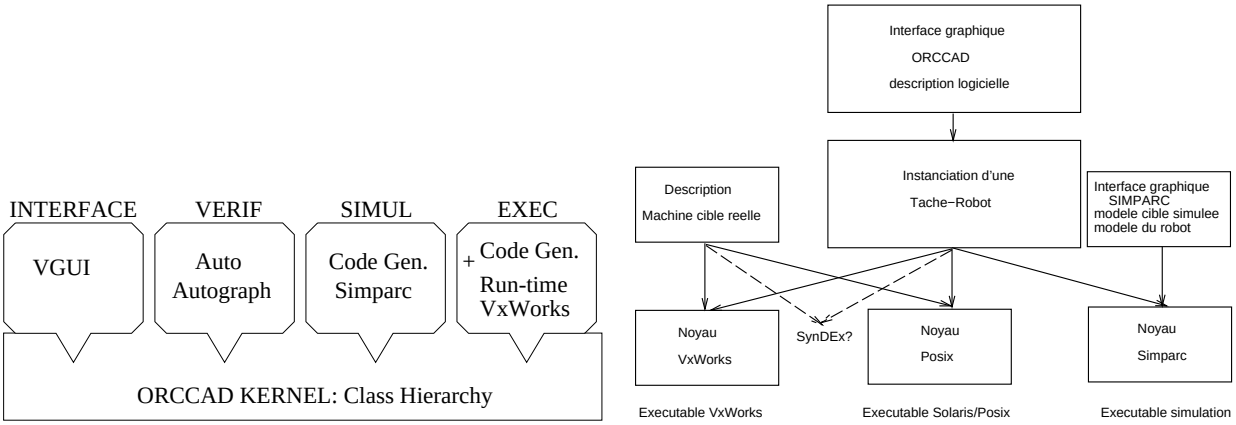


FIG. 20 – Structure de la nouvelle IHM d'ORCCAD

- Le **Noyau** est une librairie de classes C++ implantant la hiérarchie de classes proposée dans le modèle de la structure ORCCAD.
- L'**Interface graphique**, développée sous IlogViews, permet de créer des instances de la hiérarchie d'ORCCAD. Elle permet de créer des objets de complexité croissante, des modules de base jusqu'à des applications complètes. Les modules de vérification et de génération de code peuvent être activés au travers de cette interface.
- Le module de **Vérification**, utilisant Mauto et Autograph, permet de vérifier les propriétés logique du contrôleur après composition des actions élémentaires.
- Le module de **Simulation** permet de simuler le système robot+contrôleur à l'aide de Simparc. L'utilisation d'un modèle du système d'exploitation temps-réel permet de simuler une structure de code identique à celle effectivement générée pour la cible réelle.
- Le module d'**Exécution** génère le code exécutable sur l'architecture cible. Les cibles logicielles actuelles sont VxWorks et Solaris utilisant les "threads" Posix (cette dernière activité ayant été réalisée en collaboration avec le projet Robotvis). Enfin, nous envisageons une connexion avec SYNDEX, un logiciel d'aide à l'implantation optimisée multi-processeurs développé par le projet Sosso à l'Inria Rocquencourt, ce qui permettrait d'étendre le domaine de cibles d'ORCCAD vers des cibles telles que PC ou micro-contrôleurs.

Ces différents modules sont en cours d'intégration. Une première version du système, simple mais diffusable, est prévue pour le début de 1997.

Vérification du squelette de synchronisation Les modèles de TMs et de synchronisation développés sous forme de réseaux de Petri temporisés apparaissent être tout à fait satisfaisants pour conclure quand à l'absence de blocages structurels dans le réseau de tâches temps-réel communicantes, l'algorithme utilisé étant polynômial et, en pratique, très rapide. Il n'en est pas de même en ce qui concerne la recherche d'incohérence de spécification temporelle qui nécessite une exploration plus ou moins exhaustive du graphe d'atteignabilité. De plus, la technique utilisée nécessite une modification du RdP et ne permet de détecter qu'un cas fautif, celui où une TM dépasse son échéance temporelle. Enfin, le modèle actuel ne représente pas les notions de priorité et de préemption, empêchant ainsi d'analyser complètement et de construire automatiquement le diagramme d'exécution temporel du réseau de TMs périodiques. Cependant, des méthodes *algébriques* d'analyse de RdP temporisés apparaissent, comme par exemple l'algèbre sur le demi-anneau (max,+) [6].

En fait, ces RdPs temporisés sont des graphes d'événements, ayant un modèle *linéaire* dans l'algèbre (max,+) permettant de d'exploiter les équivalents de notion classique en théorie des systèmes telles que équations d'état, fonctions de transfert et boucles de rétroaction. De cette façon, il est possible d'analyser aussi bien les modes transitoires qu'asymptotiques de notre système multitâches périodiques et d'en évaluer les performances, e.g. en terme de taux de production des sorties, le tout avec des algorithmes en temps polynômial. Dans une étude en cours, en collaboration avec les projets Mistral et Sloop de l'Inria Sophia, le modèle est enrichi de façon à tenir compte les notions de priorité et de préemption entre tâches : les résultats préliminaires suggèrent qu'ainsi le diagramme d'exécution complet pourrait être analysé en temps polynômial, au moins lorsque les horloges sont toutes multiples d'une horloge de base.

Langage utilisateur Nous avons choisi Esterel en tant que langage de programmation d'applications, et nous l'utilisons également pour coder le comportement local des TRs. Ce langage s'est avéré bien adapté, en particulier pour coder les traitements d'exceptions nombreux dans nos programmes. Cependant, l'utilisation d'Esterel et des outils de vérification associés nécessite une expertise certaine de la part de l'utilisateur. Il est nécessaire de fournir à celui ci un langage plus naturel, utilisant des primitives pouvant être traduites en Esterel comme par exemple celui décrit dans [25]. Ces primitives peuvent combiner plusieurs TRs ainsi que des traitements d'exception dans une PrR spécifique du domaine d'application, fournissant à l'utilisateur des primitives réellement de haut niveau, vérifiées et fiabilisées. Parmi divers formalismes possibles (graphiques ou textuels) le langage MAESTRO actuellement développé par N. Turro et È. Coste-Manière correspond à ce besoin.

Vérification formelle Il nous paraît important de pouvoir procéder à un certain nombre de vérifications formelles sur le logiciel de commande d'un système devant opérer en milieu hostile et donc difficilement récupérable. En particulier, nous voulons vérifier que la spécification du programme respecte bien les désirs de l'utilisateur, spécialiste du domaine d'application, et dans un deuxième temps que le programme implanté respecte la spécification. L'utilisation d'Esterel comme langage de programmation du comportement logique des TRs, des PrRs et des missions permet de vérifier certaines propriétés cruciales telles que la vivacité et la sûreté dans un cadre formel unique. La structuration des traitements d'exception en trois types permet, dans une certaine mesure, d'automatiser la construction de critères abstraits permettant de vérifier ces propriétés à l'aide du logiciel d'analyse d'automates Mauto associé à Esterel. Des méthodes d'expression "naturelle" de critères abstraits de vérification restent à développer pour aider l'utilisateur non expert à vérifier les propriétés spécifiques de son application.

D'autre part, les vérifications actuellement réalisées sont purement logiques. Il peut être intéressant de vérifier des propriétés faisant intervenir le temps de façon explicite (prouver qu'une action se terminera dans un délai donné) ou encore faisant intervenir la valeur numérique de certaines variables d'état du système (le sous-marin aura-t-il toujours assez d'énergie à bord pour sortir de sous la banquise ?). Des expériences préliminaires encourageantes ont été menées dans ce but en liaison avec le projet Spectre de l'UR de Grenoble ([77]), avec comme objectif l'utilisation de théories et d'outils apparaissant pour la vérification de systèmes hybrides. Les méthodes utilisées restent cependant lourdes à mettre en oeuvre ([29]).

Répartition du contrôle Jusqu'à présent, dans les applications réalisées le code de contrôle rassemblant le comportement logique des TRs et PrRs est compilé en un seul automate global, permettant en particulier d'effectuer des vérifications formelles du comportement global au niveau le plus haut i.e. celui de la mission. Cependant, cet automate peut atteindre des tailles importantes (plusieurs centaines d'états et plusieurs milliers de transitions dans le cas de la mission décrite section 4), ce qui pose des problèmes pratiques de vérification même si ceux ci peuvent être partiellement résolus en utilisant des représentations sous forme de diagrammes de décision binaires (BDD). De plus, le contrôleur étant situé sur l'un des noeuds de l'architecture matérielle, une rupture de communication entre ce noeud et les autres laisserait les sous-systèmes distants totalement hors de contrôle. Répartir le code de contrôle sur les différents noeuds permettrait de maintenir la taille des automates dans des

tailles raisonnables et d'améliorer la survivabilité des sous-systèmes en leur conférant une autonomie minimale. Deux techniques possibles sont envisagées et font l'objet de tests sur l'architecture distribuée de l'Ifremer :

- spécifier et compiler séparément les contrôleurs correspondant aux différents noeuds et les faire communiquer de façon asynchrone, solution simple qui se paye par la perte de la possibilité de vérification formelle globale de la mission ;
- compiler globalement la mission et utiliser un outil de répartition automatique de code issu de théories émergentes sur la distribution de systèmes réactifs [18], permettant de préserver les possibilités de vérification au prix d'un accroissement significatif du débit d'événements entre les noeuds.

Synthèse du contrôle La vérification des propriétés logiques du programme de contrôle reste délicate à effectuer par un utilisateur non expert des outils disponibles. D'autre part, dès que la taille de l'application et de l'automate résultant devient grande, le risque d'oublier de vérifier une propriété importante est bien réel. Il serait plus séduisant de pouvoir synthétiser le contrôleur à partir de l'expression de propriétés à vérifier, si possible décrites d'une façon "naturelle". Grâce aux structures mises en place dans ORCCAD, les aspects liés au temps continu sont pris en compte par les lois de commande. À partir de la vue externe des TRs nous ne manipulons plus que des systèmes à événements discrets. Il peut alors être envisagé de synthétiser le contrôle par exemple en adaptant les techniques introduites par Ramadge et Wonham [52], en particulier dans leurs variétés utilisant un modèle de type "entrée-sortie" pour le système contrôlé [8]. Il ne resterait alors à vérifier que la spécification des propriétés modélisant les désirs de l'utilisateur...mais cette voie reste totalement à explorer.

Références

[1] C. Aiglon, C. Lavarenne, Y. Sorel et A. Vicard : "Utilisation de SynDEx pour le traitement d'images temps réel", Rapport de recherche INRIA no 2968, Septembre 1996.

[2] R. Alami, R. Chatila and B. Espiau, "Designing an Intelligent Control Architecture for Autonomous Robots", *ICAR 93, Int. Conf. on Advanced Robotics*, Tokyo, November 1993.

[3] J.S. Albus, M. Juberts, S. Szabo, "RCS : A Reference Model Architecture for Intelligent Vehicle and Highway Systems", *ISATA 92*, Florence, Italy, June 1992.

[4] B. Armstrong-Helouvry, "Latency Analysis of Asynchronous Distributed Control", Internal Report, *Department of Electrical Engineering*, University of Wisconsin, Milwaukee, 1992

[5] C. Astraudo, J.J. Borrelly, "Simulation of Multiprocessor Robot Controllers", *Proc. IEEE Int. Conf. on Robotics and Automation*, Nice, May 1992

[6] F. Baccelli, G. Cohen, G.J. Olsder and J.P. Quadrat : *"Synchronization and Linearity"*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, 1992.

[7] P. Backes, S. Hayati, V. Hayward and K. Tso, " The Kali Multi-Arm Robot Programming and Control Environment" in *Proc. of the NASA Conference on Space Telerobotics*, G.Rodriguez and H. Seraji editors, JPL Publication 89-7, January 1989.

[8] S. Balemi, G.J. Hoffman, P. Gyugyi, H. Wong-Toi and G.F. Franklin : "Supervisory Control of a Rapid Thermal Microprocessor", *IEEE Trans. on Automatic Control*, vol. 38, no 7, July 1993.

[9] B. Banerdt e.a. : "INTERMARSNET, Phase A Study Report", *ESA Publication D/SCI(96)2*, April 1996.

[10] M. Barbeau, P. Freedman and D. Simon : "Synthesis of Safe and Live Robot-Tasks", Rapport de recherche n^o 145, *Université de Sherbrooke*, Février 1995.

[11] J.G. Bellingham, T.R. Consi, "State Configured Layered Control, *Proc. 1st Worshop on Mobile Robots for Subsea Environments*, pp 75-80, Monterey, October 1990.

[12] J.J. Borrelly and D. Simon, "Propositions d'Architecture de Contrôleur Ouvert pour la Robotique", *INRIA Research Report no 1304*, October 1990.

[13] J.J. Borrelly, D. Simon, V. Dupourqué et H. Poilvé : Architectures matérielles et logicielles des contrôleurs de robots, in *Techniques de la robotique, tome 1 : Architectures et commandes,* pages 195-253, Hermès, Paris, 1988.

[14] F. Boussinot and R. de Simone : "The Esterel Language", *Proc. of the IEEE*, vol. 79, n^o 9, September 1991.

[15] R.A. Brooks, "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, vol 2, no 1, pp 14-23, March 1986.

[16] R.A. Brooks : "Intelligence without representation", *Artificial Intelligence 47*, pp 139-159.

[17] R.B. Byrnes, "*The Rational Behavior Model : A Multi-Paradigm, Tri-Level Software Architecture for the Control of Autonomous Vehicles*, PhD Dissertation, Naval Postgraduate School, Monterey, USA, March 1993.

[18] P. Caspi, A. Girault, and D. Pilaud. Distributing reactive systems. In *7th Int. Conf. on Parallel and Distributed Computing Systems, PDCS'94,* Las Vegas, October 1994.

[19] E. Castillo : *Principes, outils et techniques de simulation, vérification et exécution d'actions robotiques*, Thèse de l'Institut Polytechnique National de Grenoble, Novembre 1994.

[20] J.B. Chen, B.S.R. Armstrong, R.S. Fearing and J.W. Burdick, "Satyr and the Nymph : Software Archetype for Real Time Robotics", *Proc. IEEE-ACM Joint Computer Conference*, Dallas, November 1988.

[21] R. David and H. Alla, *"Du GRAFCET aux réseaux de Petri"*, Editions Hermès, Paris, 1989.

[22] J. Delacroix, "Stabilité et régisseur d'ordonnancement en temps réel", *Technique et science informatiques*, vol. 13, no 2, pp223-250, 1994.

[23] E. Coste-Manière, *"Synchronisme et Asynchronisme dans la programmation des systèmes robotiques : apport du langage* ESTEREL *et de concepts objets"*, PhD Dissertation, Ecole des Mines de Paris, France, July 1991.

[24] E. Coste-Manière, B. Espiau and D. Simon, "Reactive Objects in a Task-Level Open Controller, *Proc. IEEE Conf. on Robotics and Automation*, pp 2732-2737, Nice, France, May 1992.

[25] E. Coste-Manière, B. Espiau and E. Rutten, "A Task-Level Robot Programming Language and its Reactive Execution", *Proc. IEEE Conf. on Robotics and Automation*, pp 2751-2756, Nice, France, May 1992

[26] J.L. Crowley, "Layered Control for Intelligent Robotic Devices", *IEEE Conf. Rob. and Aut., Workshop on Architectures for Intelligent Control Systems*, pp 55-60, Nice, France, May 1992.

[27] B. Espiau and K. Kapellos, "Robot Task Specification", Internal INRIA-Aleph Working Report, January 1992

[28] B. Espiau, D. Simon et K. Kapellos : "Formal verification of missions and tasks", *U.S./Portuguese workshop on undersea robotics and intelligent control*, Lisbonne, March 1995

[29] B. Espiau, K. Kapellos, M. Jourdan, D. Simon, " On the validation of robotics control systems Part I : High level specification and formal verification", *Rapport de Recherche no 2719*, INRIA, novembre 1995.

[30] S. Fleury, M. Herbb and R. Chatila : " Design of a modular architecture for autonomous robots", *IEEE Int. Conf. on Robotics and Automation*, San Diego, May 1994.

[31] A. Franck and R. McGhee, "Some Considerations Relating to the Design of Autopilots for Legged Vehicules", *Journal of Terramechanics*, vol. 6, no 1, March 1969.

[32] L. George, N. Rivierre and M. Spuri : "Preemptive and Non-Preemptive Real-Time UniProcessor Scheduling", Inria Research Report #2966, September 1996.

[33] S.Y. Harmon, "Architectures : Designers vs. Implementors", *IEEE Conf. Rob. and Aut., Workshop on Architectures for Intelligent Control Systems*, pp 1-6, Nice, France, May 1992.

[34] J.M. Hasemann : "Robot control architectures application requirements, approaches and technologies", in Proc. *SPIE Intelligent Robots and Computer Vision XIV*, Philadelphia, October 1995.

[35] A.J. Healey, R.B. McGhee e.a., "Mission Planning, Execution and Data Analysis for the NPS AUV II Autonomous Underwater Vehicle", *Proc. 1st Worshop on Mobile Robots for Subsea Environments*, pp 177-186, Monterey, October 1990

[36] K. Kapellos : *Environnement de programmation d'applications robotiques réactives*, Thèse de l'Ecole des Mines de Paris, Sophia-Antipolis, Novembre 1994.

[37] K. Kapellos and B. Espiau, "Implementation with Orccad of a Method for Smooth Singularity Crossing in a 6-DOF Manipulator", Inria Research Report no 2654, September 1995

[38] K. Kapellos, S. Abdou, M. Jourdan and B. Espiau : "Specification, Verification and Implementation of Tasks and Missions for an Autonomous Vehicle", *4th Int. Symp. on Experimental Robotics*, Stanford, 1995.

[39] J. Ish-Shalom, P. Kazanzides, " SPARTA : Multiple Signal Processors for High-Performance Robot Control", *IEEE Transactions on Robotics and Automation*, vol 5, no 5, October 1989.

[40] A. Jaritz and M. Spong : "An Experimental Comparison of Robust Control Algorithms on a Direct Drive Manipulator", *IEEE Trans. on Control Systems Technology*, vol. 4, no 6, November 1996.

[41] P.K. Khosla, "Choosing Sampling Rates for Robot Control", *Proc. IEEE Int. Conf. on Robotics and Automation*, 1987

[42] E. Mazer e.a., "ACT : A Robot Programming Environment", *Proc. IEEE Int. Conf. on Robotics and Automation*, Sacramento, April 1991.

[43] S. McMillan, D. Orin and B. McGhee, "Efficient Dynamic Simulation of an Underwater Vehicle with a Robotic Manipulator", *IEEE Trans. on Systems, Man and Cybernetics*, vol. 25, no 8, August 1995.

[44] M. Mejia Olvera : *Contribution à la Conception d'un Réseau Local Temps Réel pour la Robotique*, Thèse de Doctorat de l'Université de RennesI/IFSIC, Juin 1989.

[45] M. Mejia, D. Simon, P. Belmans and J.J. Borrelly, "Mécanismes de synchronisation dans un systeme robotique réparti", *Proc. Séminaire Franco-Brésilien sur les systèmes informatiques répartis*, Florianopolis, Brazil, September 89.

[46] T. Murata, "Petri Nets : Properties, Analysis and Applications", In *Proceedings of the IEEE*, Vol. 77, No. 4, April 1989.

[47] S. Narasimhan, D.M. Siegel and J.M. Hollerbach, "Condor : An Architecture for Controlling the Utah/MIT Dexterous Hand", *IEEE Trans. on Rob. and Aut.*, vol 5 no 5, October 1989.

[48] M. Nemani, T.C. Tsao and S. Hutchinson, "Multi-Rate Analysis and Design of Visual Feedback Digital Servo-Control System", *Journal of Dynamic Systems, Measurement and Control*, vol. 116, pp 45-55, March 1994.

[49] P. Oliveira, A. Pascoal, V. Silva and C. Silvestre : "Mission Control in the MARIUS AUV", Preprints of the *6th IARP workshop on Underwater Robotics*, Toulon, France, March1996, to appear in *Int. Journal of Systems Science.*

[50] A. Pascoal, "The AUV MARIUS : Mission Scenarios, Vehicle Design, Construction and Testing", *Proc. of the 2nd IARP Workshop on Mobile Robots for Subsea Environments*, Monterey, CA, May 1994.

[51] R. Pissard-Gibollet, P. Rives, K. Kapellos, J. Borrelly, "Real-Time Programming of a Mobile Robot Actions Using Advanced Control Techniques", in : *Proc. of 4th Int. Symposium on Experimental Robotics*, Stanford, California, USA, juin 1995.

[52] P.J. Ramadge and W.M. Wonham : "Supervisory control of a class of discrete event processes", *SIAM Journal of Control and Optimization*, vol. 25, pp 206-230, Jan. 1987.

[53] C. Ramamoorthy and G. Ho. "Performance evaluation of asynchronous concurrent systems using petri nets", *IEEE Trans. on Software Engineering*, 6(5) :440–449, September 1980.

[54] V. Rigaud e. a. : "UNderwater Intelligent Operation and Navigation : Main objectives and first results", *preprints of 6th IARP workshop on Underwater Robotics*, Toulon, March 1996

[55] P. Rives and J.J. Borrelly : "Visual Servoing Techniques Applied to an Underwater Vehicle", *Int. Conf. on Robotics and Automation*, Albuquerque, April 1997

[56] C. Samson, B. Espiau and M. Le Borgne : "Robot Control : the Task-Function Approach", *Clarendon Press, Oxford Science Publications*, U.K. , 1991.

[57] C. Samson and B. Espiau, "Application of the Task-Function Approach to Sensor-based Control of Robot Manipulators", *Proc. IFAC 1990*, Tallinn, CCCP, Aug. 1990

[58] A. Santos, D. Simon et V. Rigaud : "A sensor-based high-level control approach for autonomous underwater vehicles," *Proc. of Intelligent Robotics Systems IRS'94*, Grenoble, July 1994.

[59] A. Santos, D. Simon, and V. Rigaud, "Towards autonomous underwater vehicles (auvs) through sensor-based high level control," *Proc. OCEANS'94 OSATES - Brest (FRANCE)*, pp. 113–118,

[60] A. Santos, P. Rives, B. Espiau, and D. Simon, "Dealing in real time with a priori unknown environment on autonomous underwater vehicles," *IEEE Rob. & Aut.*, Nagoya, Sept. 1995.

[61] A. Santos, D. Simon, and V. Rigaud, "Sensor-based control of under-actuated autonomous underwater vehicles," *IFAC CAMS'95 - Trondheim (Norvège), Mai 1995.*

[62] A. Santos, D. Simon, R. Bitmead et V. Rigaud : "Bottom-referenced control of autonomous underwater vehicles", *U.S./Portuguese workshop on undersea robotics and intelligent control*, Lisbonne, March 1995

[63] A. Santos, D. Simon et V. Rigaud : "Bottom-referenced control of autonomous underwater vehicles", *ICAR'95*, Barcelone, Septembre 1995

[64] A. Santos, B. Espiau, P. Rives, D. Simon, V. Rigaud, "Sensor-Based Control of Holonomic Autonomous Underwater Vehicles", *Rapport de Recherche no 2609*, INRIA, juillet 1995.

[65] A. Santos, "*Contribution à la conception des sous-marins autonomes : architecture des actionneurs, architecture des capteurs d'altitude et commandes référencées capteurs*", thèse de doctorat, Ecole des Mines de Paris, Sophia-Antipolis, décembre 1995.

[66] L. Sha, R. Rajkumar and S. Sathaye, "Generalized Rate-Monotonic Scheduling Theory : A Framework for Developing Real-Time Systems", *Proceedings of the IEEE*, Special Issue on Real-Time Systems, vol. 82, n^o 1, January 1994.

[67] K.G. Shin and P. Ramanathan, "Real-Time Computing : A New Discipline of Computer Science and Engineering", *Proceedings of the IEEE*, Special Issue on Real-Time Systems, vol. 82, n^o 1, January 1994.

[68] K.G. Shin and X. Cui, "Computing Time Delay and its Effects on Real-Time Control Systems", *IEEE Trans. on Control Systems Technology*, Vol. 3, n^o 2, June 1995.

[69] M. Silva and J.M. Colom, "On the Computation of Structural Synchronic Invariants in P/T Nets", *Advances in PN'88, LNCS 340*, Springer-Verlag, 1989

[70] D. Simon and A. Joubert," ORCCAD : Towards an Open Robot Controller Computer Aided Design System", *INRIA Research Report no 1396*, February 1991.

[71] D. Simon, B. Espiau, E. Castillo and K. Kapellos, "Computer-Aided Design of a Generic Robot Controller Handling Reactivity and Real-Time Control Issues", INRIA Research Report no 1801, Novembre 1992.

[72] D. Simon, B. Espiau, E. Castillo and K. Kapellos, "Computer-Aided Design of a Generic Robot Controller Handling Reactivity and Real-Time Control Issues", *IEEE Transactions on Control Systems Technology*, vol 1, n^o 4, pp 213-229, December 1993.

[73] D. Simon, P.Freedman and E. Castillo, "Analysing the temporal behavior of real-time closed-loop robotic tasks", *IEEE Int. Conf. on Robotics and Automation*, San Diego, May 1994.

[74] D. Simon, A. Santos, V. Rigaud "A new sensor-based control approach for autonomous underwater vehicles", 2^{nd} *Workshop on Mobile Robots for Subsea Environment*, International Advanced Robotics Programme, Monterey, Mai 1994.

[75] D. Simon, E. Coste-Manière, R. Pissard, V. Rigaud, M. Perrier, A. Peuch : "A Reactive approach to underwater Vehicle control : the mixed Orccad/Pirrat programming of the Vortex vehicle", 2^{nd} *Workshop on Mobile Robots for Subsea Environment*, International Advanced Robotics Programme, Monterey, Mai 1994.

[76] D. Simon, K. Kapellos et B. Espiau : "Formal verification of missions and tasks : Application to underwater robotics", *ICAR'95*, Barcelone, Septembre 1995

[77] D. Simon, K. Kapellos, B. Espiau et M. Jourdan : "Formal verification of robotic missions and tasks", *2nd european workshop on real-time and hybrid systems*, Grenoble, Mai 1995

[78] D. Simon, E. Castillo, P. Freedman, "On the validation of robotics control systems Part II : Analysis of real-time closed-loop control tasks", *Rapport de Recherche no 2720*, INRIA, novembre 1995.

[79] D. Simon, K. Kapellos et A. Santos : "Smooth Task Switching Mechanisms", UNION project, Deliverable D2.5, Janvier 1996.

[80] D. Simon, K. Kapellos and B. Espiau, "Control laws, Tasks and Procedures with ORCCAD : Application to the Control of an Underwater Arm", *preprints of 6th IARP workshop on Underwater Robotics*, Toulon, March 1996 (to appear in *Int. Journal of Systems Science*)

[81] D. Simon, K. Kapellos, B. Espiau and R. Pissard-Gibollet : "Orccad : Software Engineering for Real-time Robotics, *A Technical Insight*", *ROBOTICA*, vol. 15, Part 1, pp 105-110, January-February 1997.

[82] D. Simon, E. Castillo and P. Freedman : "Design and Analysis of Synchronization for Real-time Closed-loop Control in Robotics", to appear in *IEEE Trans. on Control Systems Technology.*

[83] D.B. Stewart, D.E. Schmitz and P.K. Khosla, "CHIMERA II : A Real Time Multiprocessing Environment for Sensor Based Control", *Proc. IEEE International Symposium on Intelligent Control*, Albany , September 1989.

[84] N. Turro and È. Coste-Manière : "Mission programming language", UNION project, deliverable D2.2, Août 1996.

[85] L. Whitcomb and D. Koditschek, "Robot Control in a Message Passing Environment : Theoretical Questions and Preliminary Experiments", *Proc. IEEE Int. Conf. on Robotics and Automation*, USA, 1990.

A C.V. et Publications

Daniel SIMON

Né le 27 Février 1954 à Vesoul
Nationalité Française
Divorcé sans enfants.

Adresse professionnelle
INRIA - Projet ICARE
2004 route des Lucioles
B.P. 93
06902 SOPHIA-ANTIPOLIS Cedex
Tél : 04 93 65 77 69
e-mail : dsimon@sophia.inria.fr

Adresse personnelle
Les Hauts d'Antibes
Les Lilas B4
Bd. G. Appollinaire
06600 ANTIBES
Tél : 04 93 95 94 55

Formation

1980 : Docteur-Ingénieur, Faculté des Sciences et Techniques de Franche-Comté "Contribution à l'étude de modules de motricité hydrauliques pour robots d'assemblage".

1977 : Diplôme d'Etudes Approfondies, Mécanique non linéaire et chronométrie, Faculté des Sciences et Techniques de Franche-Comté.

1976 : Ingénieur de l'Ecole Nationale Supérieure de Chronométrie et Micromécanique, option Automatique, Besançon.

Expérience professionnelle

Février 1980 à Octobre 1981 : Ingénieur contractuel puis Chercheur 2 à l'Inria Rocquencourt (Projet de F. Germain). Réalisation d'asservissements pour le prototype de Cyclope.

Octobre 1981 à Septembre 1988 : Chercheur 2 puis CR1 à l'Inria-Rennes (Projet de B. Espiau). Etude et réalisation d'un robot hydraulique à hautes performances. Démarrage d'études sur les architectures de commande en robotique. Etude du réseau local temps réel Damnet.

Depuis Octobre 1988 : CR1 à l'Inria-Sophia dans les projets Prisme puis Icare. Participation au développement du logiciel de simulation Simparc. Conception et direction du développement de l'atelier de génie logiciel pour la robotique Orccad. Démarrage d'études et de collaborations dans le domaine de la robotique sous-marine.

Encadrement de thèses

P. Belmans : "Etude de la coopération et de la synchronisation dans les systèmes multirobots multicapteurs", *Université de Rennes I/IFSIC*, Septembre 1988.

M. Mejia : "Contribution à la conception d'un réseau local temps réel pour la robotique", *Université de Rennes I/IFSIC*, Juin 1989.

M.C. Séguillon : "Conception et implantation de la commande d'un robot hydraulique. Intégration dans un système multirobots multicapteurs", *Université de Rennes I*, Décembre 1989.

E. Castillo : "Principes, techniques et outils de simulation, vérification et exécution d'actions robotiques", *INPG Grenoble*, Novembre 1994.

A. Santos : "Contribution à la conception des sous-marins autonomes : architecture des actionneurs, architecture des capteurs d'altitude et commandes référencées capteurs", *École des Mines de Paris*, Décembre 1995.

Contrats industriels et collaborations scientifiques

Participation au projet Esprit II ARMS (Advanced Robot Manipulators System) : Spécifications d'architectures et d'outils de CAO pour les contrôleurs de robots (1989-1992).

Contrat de recherche en collaboration avec l'ENSMP et Aleph-technologies, en réponse à l'appel d'offre informatique du MRT : Développement de l'atelier de génie logiciel pour la robotique Orccad (1991-1993).

Contrat de recherche avec l'Ifremer : Modélisation et commande de véhicules autonomes sous-marins (faisabilité) 1991.

Contrat de recherche avec l'Ifremer : Programmation de véhicules sous-marins autonomes (1993).

Participation au projet européen MAST II Advanced System Research for Unmanned Autonomous Underwater Vehicles : Architectures de perception, de propulsion et de commande d'AUVs. (1992-1994).

Projet Esprit III BRA UNION (Underwater Intelligent Operation and Navigation) : Coordinateur du Workpackage "Mission Programming" (1994-1996).

Collaboration NSF/INRIA : projet "An Experimental Study of Software Architecture and Software Reuse for Control of Unmanned Underwater Vehicles" avec Naval Postgraduate School, Monterey, CA (Prof. R. McGhee), 1994-1996.

Collaboration CRIM/INRIA : "Modélisation et analyse temporelle de tâches temps-réel en robotique" (P. Freedman), 1993-1994-1996 (Projet soutenu par la Commission permanente de coopération franco-québecoise).

Projet AIOLI : "Automatisation des tâches d'Inspection et d'Opération sous-marines : Laboratoire d'essais et d'Instrumentations", financé par le Conseil Régional PACA, 1996-1997.

Action incitative TOLERE : "Code réparti tolérant aux pannes pour systèmes embarqués" en collaboration avec les projets Bip et Sosso, 1998-1999.

Enseignement

Cours et TP de CAO électronique au CERICS (Sophia), 1991-1992

Cours "Actionneurs pour la robotique", DEA Robotique et Vision, UNSA, 1991

Ecole d'été robotique du CIMPA (1992) : "Actionneurs et capteurs en robotique", "Architecture des contrôleurs de robot".

Essi 2 (Sophia) : "Actionneurs pour la robotique" 1994,1995,1996

Essi 3 (Sophia) : "Architecture des contrôleurs de robots" 1994,1996

Organisation du module d'option "Architectures logicielles et outils de programmation en robotique", DEA R&V, UNSA, 1993

"Le système Orccad", DEA R&V, UNSA, 1993,1994,1995,1996.

"Architectures de contrôle en robotique", DEA ARAVIS, 1997, 1998.

Divers

Participation au groupe de travail "Asynchrone/Synchrone" de l'action C2A du CNRS.

Représentant de l'Inria au Comité Directeur du Groupement Robotique Sous-marine Méditerrannée.

Articles de revue et chapitres de livre

- K. Kapellos, D. Simon, M. Jourdan and B. Espiau : "Task Level Specification and Formal Verification of Robotics Control Systems : State of the Art and Case Study", accepté par *Int. Journal of Systems Science.*
- The Orccad team : "The ORCCAD Architecture", *Int. Journal of Robotics Research*, special issue on Integrated Architectures for Robot Control and Programming, March 1998.
- D. Simon, K. Kapellos et B. Espiau : "Control laws, Tasks and Procedures with Orccad : Application to the Control of an Underwater Arm", a paraître dans *Int. Journal of Systems Science*, Vol 29 no 6, June 1998.
- D. Simon, E. Castillo and P. Freedman, "Design and Analysis of Synchronization for Real-time Closed-loop Control in Robotics", accepté par *IEEE Transactions on Control Systems Technology.*
- D. Simon, B. Espiau, K. Kapellos and R. Pissard-Gibollet, "ORCCAD : Software Engineering for Real-time Robotics :*A Technical Insight*", *Robotica* vol 15, pp 111-115, January 1997.
- V. Rigaud e.a. (Union team) "Union : Underwater intelligent operation and navigation" accepté par *IEEE Robotics and Automation Magazine*, Special Issue on Robotics & Automation in Europe : Projects funded by the Commission of the European Union, 1998.
- D. Simon, B. Espiau, E. Castillo and K. Kapellos, "Computer-Aided Design of a Generic Robot Controller Handling Reactivity and Real-Time Control Issues", *IEEE Transactions on Control Systems Technology*, Vol 1, $n^o 4$, pages 213-229, December 1993.
- J.J. Borrelly, D. Simon, V. Dupourqué et H. Poilvé : Architectures matérielles et logicielles des contrôleurs de robots, in *Techniques de la robotique, tome 1 : Architectures et commandes,* pages 195-253, Hermès, Paris, 1988.

Conférences

- D. Simon, K. Kapellos and B. Espiau : "Formalization of hybrid structures in robot controllers : the Orccad approach", INCOM'98, Nancy-Metz, June 1998.
- K. Kapellos, D. Simon, S. Granier and V. Rigaud : "Distributed Control of a Free-floating Underwater Manipulation System", *5th Int. Symposium on Experimental Robotics*, Barcelona, June 1997.
- D. Simon, B. Espiau, K. Kapellos, R. Pissard-Gibollet : "ORCCAD : un environnement de conception, de validation et d'éxécution pour les systèmes embarqués", Colloque Macsim sur le Contrôle des Processus, Toulon, Juin 1997.
- D. Simon, K. Kapellos and B. Espiau, "Design of Control Procedures for a Free-floating Underwater Manipulation System", *IEEE Int. Conference on Robotics and Automation*, Albuquerque, April 1997
- D. Simon, K. Kapellos et B. Espiau : "Control laws, Tasks and Procedures with Orccad : Application to the Control of an Underwater Arm", *6th IARP workshop on Underwater Robotics*, Toulon, Mars 1996
- A. Santos, D. Simon et V. Rigaud : "Bottom-referenced control of autonomous underwater vehicles", *ICAR'95*, Barcelone, Septembre 1995
- D. Simon, K. Kapellos et B. Espiau : "Formal verification of missions and tasks : Application to underwater robotics", *ICAR'95*, Barcelone, Septembre 1995
- D. Simon, K. Kapellos, B. Espiau et M. Jourdan : "Formal verification of robotic missions and tasks", *2nd european workshop on real-time and hybrid systems*, Grenoble, Mai 1995
- B. Espiau, D. Simon et K. Kapellos : "Formal verification of missions and tasks", *U.S./Portuguese workshop on undersea robotics and intelligent control*, Lisbonne, March 1995
- A. Santos, D. Simon, R. Bitmead et V. Rigaud : "Bottom-referenced control of autonomous

underwater vehicles", *U.S./Portuguese workshop on undersea robotics and intelligent control*, Lisbonne, March 1995
- A. Santos, D. Simon, and V. Rigaud, "Sensor-based control of under-actuated autonomous underwater vehicles," *IFAC CAMS'95* - Trondheim (Norvège), Mai 1995.
- A. Santos, P. Rives, B. Espiau, and D. Simon, "Dealing in real time with a priori unknown environment on autonomous underwater vehicles," *IEEE Int. Conf. on Robotics and Automation* - Nagoya (JAPON), Septembre 1995.
- A. Santos, D. Simon, and V. Rigaud, "Towards autonomous underwater vehicles (auvs) through sensor-based high level control," *Proc. OCEANS'94 OSATES - Brest (FRANCE)*, pp. 113–118,
- A. Santos, D. Simon et V. Rigaud : "A sensor-based high-level control approach for autonomous underwater vehicles (auvs)," *Proc. of Intelligent Robotics Systems IRS'94* - Grenoble (FRANCE),
- D. Simon, P. Freedman et E. Castillo : "Analysing the Temporal Behavior of Real-Time Closed-loop Robotic tasks", *IEEE Int. Conf. on Robotics and Automation*, San Diego, Mai 1994.
- D. Simon, A. Santos, V. Rigaud : " A new sensor-based control approach for autonomous underwater vehicles", 2^{nd} *Workshop on Mobile Robots for Subsea Environment*, International Advanced Robotics Programme, Monterey, Mai 1994.
- D. Simon, E. Coste-Manière, R. Pissard, V. Rigaud, M. Perrier, A. Peuch : "A Reactive approach to underwater Vehicle control : the mixed Orccad/Pirrat programming of the Vortex vehicle", 2^{nd} *Workshop on Mobile Robots for Subsea Environment*, International Advanced Robotics Programme, Monterey, Mai 1994.
- V. Rigaud, D. Simon, E. Coste-Manière, L. Marce, E. Lerest : "High Level Reactive and Intelligent Software for AUV's Mission Programming", *AUV'94*, Boston, 1994.
- V. Rigaud, L. Marce, D. Simon, E. Coste-Manière, M. Perrier, A. Peuch : "Versatile and Open Sussea Robot for Technical Experiment : Prototyping the Next AUV and ROV Generation, *ISOPE 94*, Osaka, April 1994.
- V. Rigaud, E. Coste-Manière, M. Perrier, A. Peuch, D. Simon : "Vortex : Versatile and Open Subsea Robot for Technical Experiment", *Proceedings of OCEANS'93*, Victoria, Canada, October 1993.
- M. Perrier, V. Rigaud, E. Coste-Manière, D. Simon, A. Peuch : "Vortex : a versatile testbed vehicle for control algorithms evaluation", 8^{th} *Int. Symp. on Unmanned Untethered Submersible Technology*, New Hampshire, USA, September 1993.
- E. Coste-Manière, B. Espiau and D. Simon, "Reactive Objects in a Task-Level Open Controller, *Proc. IEEE Conf. on Robotics and Automation*, pp 2732-2737, Nice, France, May 1992.
- M. Mejia, D. Simon, P. Belmans and J.J. Borrelly, "Mécanismes de synchronisation dans un systeme robotique réparti", *Proc. Séminaire Franco-Brésilien sur les systèmes informatiques répartis*, Florianopolis, Brazil, September 89.
- M. Mejia, D. Simon, J.J. Borrelly : "Un réseau local temps réel pour la robotique", *Colloque Francophone sur l'ingénierie des protocoles*, Bordeaux, Eyrolles, Septembre 1988.
- P. Belmans, J.J. Borrelly, M. Mejia, D. Simon : "A distributed system for robotic applications", 3^{rd} *IEEE Int. Symp. on Intelligent Control*, Arlington, Virginia, August 1988.
- J.J. Borrelly, D. Simon : "Specifications of a sublocal network for robotics", *Séminaire INRIA Réseaux locaux temps-réel*, Bandol, Avril 1986.

Rapports de recherche

- P. Freedman et D. Simon : “On the runtime co-existence of event-driven and time-driven behaviors for embedded control”, Rapport Technique CRIM, Janvier 1995.
- B. Espiau, M. Jourdan, K. Kapellos, D. Simon : “On the validation of robotics control systems Part I : High level specification and formal verification”, Rapport de Recherche Inria n^o 2719, Novembre 1995
- D. Simon, E. Castillo, P. Freedman : “On the validation of robotics control systems Part II : Analysis of real-time closed-loop control tasks”,Rapport de Recherche Inria n^o 2720, Novembre 1995
- A. Santos, B. Espiau, P. Rives, D. Simon, V. Rigaud : “Sensor-Based Control of Holonomic Autonomous Underwater Vehicles” Rapport de Recherche Inria n^o 2609, Juillet 1995
- P. Freedman et D. Simon. “Computing systems for robotics : where automatic control meets software engineering”.Rapport Technique CRIM 95/04, Montréal, April 1995.
- M. Barbeau, P. Freedman and D. Simon : ”Synthesis of Safe and Live Robot-Tasks”, Rapport de recherche n^o 145, Université de Sherbrooke, Février 1995.
- D. Simon, B. Espiau, E. Castillo and K. Kapellos, ”Computer-Aided Design of a Generic Robot Controller Handling Reactivity and Real-Time Control Issues”, INRIA Research Report no 1801, Décembre 1992.
- D. Simon, A. Joubert : ”ORCCAD : Towards an Open Robot Controller Computer Aided Design System”, Rapport de Recherche Inria n^o 1396, Février 1991.
- J.J. Borrelly et D. Simon : ”Propositions d’architecture de contrôleur ouvert pour la robotique”, Rapport de Recherche Inria n^o 1304, Octobre 1990.

B Computer-Aided Design of a Generic Robot Controller Handling Reactivity and Real-Time Control Issues

D. Simon, B. Espiau, E. Castillo et K. Kapellos

IEEE Transactions on Control Systems Technology, vol. 1, no 4, Décembre 1993

C Task level specification and formal verification of robotics control systems : state of the art and case study

K. Kapellos, D. Simon, M. Jourdan et B. Espiau

International Journal of Systems Science, vol. 30, no 11, Novembre 1999

D Design and Analysis of Synchronization for Real-time Closed-loop Control in Robotics

D. Simon, E. Castillo et P. Freedman

IEEE Transactions on Control Systems Technology, vol. 6, no 4, Juillet 1998

E Control laws, Tasks and Procedures with ORCCAD : Application to the Control of an Underwater Arm

D. Simon, K. Kapellos et B. Espiau

International Journal of Systems Science, vol. 29, no 10, Octobre 1998

F Orccad : Software Engineering for Real-time Robotics, *A Technical Insight*

D. Simon, B. Espiau, K. Kapellos et R. Pissard-Gibollet
Robotica, vol. 15, Décembre 1997

G Sensor-Based Control of Holonomic Autonomous Underwater Vehicles

Rapport de recherche INRIA no 2609, Juillet 1995

Résumé

ORCCAD(Open Robot Controller Computer Aided Design) est un ensemble de concepts, de méthodes et d'outils destinés à spécifier, programmer, vérifier et implanter des applications robotiques. Les actions de base sont modélisées par des *Tâches-Robot* combinant une loi de commande et un comportement logique réactif : elles font ainsi l'interface entre les aspects temps continu et discret. Ces actions sont ensuite composées de façon hiérarchique pour obtenir des actions de complexité croissante jusqu'à obtenir une application complète. Cette structuration rigoureuse du logiciel de contrôle/commande permet notamment d'utiliser des méthodes de vérification formelle chaque fois que possible. L'approche est illustrée au travers d'un exemple de mission de manipulation sous-marine.