CoSARC : Une approche globale pour le développement de contrôleurs de robots

R. Passama
LIRMM Départements Robotique et Informatique
161 rue Ada, 34392 Montpellier cedex 5, France
passama@lirmm.fr
Encadrants : D. Andreu, C. Dony, T.Libourel

1 Introduction

Comme tous les systèmes logiciels actuels, les contrôleurs de robots évoluent et deviennent de plus en plus complexes. Cette complexité s'exprime de différentes façons : plus d'autonomies décisionnelle et opérationnelle sont requises ; plus de capteurs et d'actionneurs sont embarqués afin d'élargir les potentialités des robots ; plus de traitements et de commandes de natures diverses sont utilisés etc. Ce constat entraîne une difficulté toujours accrue au niveau de la conception et du développement de tels systèmes d'autant plus que la demande exige qualité et réutilisabilité.

Le besoin en termes de méthodes et outils adaptés est évident, comme en témoigne l'intensité actuelle des recherches sur ce sujet. Dans ce contexte, le but de nos travaux est d'intégrer les avancées du génie logiciel à composants [1] aux méthodes de conception et de développement des contrôleurs logiciels de robots [3] [4]. Pour réaliser ceci, nos travaux pluridisciplinaires ont consisté à étudier les propositions académiques et industrielles autour de deux domaines de recherche principaux. Le premier est le génie logiciel à objets et composants, qui fournit depuis plusieurs années des méthodes et outils informatiques visant à améliorer la qualité et la réutilisabilité du code et des spécifications des logiciels. Le deuxième domaine abordé est celui des architectures de contrôle mixtes en robotique, dont l'étude a fait émerger, entre autres, le concept de décomposition des architectures logicielles des contrôleurs en couches hiérarchisées, chaque couche étant responsable d'un niveau de délibération et de réaction dans l'architecture.

2 la méthodologie COSARC

Nous proposons à partir de ces études une méthodologie de développement de contrôleurs logiciels de robots, nommée CoSARC (acronyme anglais pour Component-based Software Architecture of Robot Controllers). Cette méthodologie s'appuie sur une architecture de contrôle générique s'inspirant de celles proposées dans la littérature. Cette architecture sert de patron au développeur d'architectures de contrôle auquel nous proposons un langage de modélisation et de programmation à composants pour effectuer le cycle complet de développement.

2.1 Architecture générique

L'architecture générique proposée s'inspire des travaux menés sur la conception d'architectures de contrôle mixtes. Elle décrit une solution, générique et adaptable, de structuration des architectures des contrôleurs logiciel de robots. Elle propose un découpage (en entités) qui favorise la lisibilité des architectures logicielles tant pendant le développement qu'à l'exécution et la réutilisation des composants à partir desquelles elles sont décrites. Cette architecture générique repose sur le concept de *Ressource*. Une *Ressource* est une entité responsable du contrôle d'une partie identifiée de la partie opérative d'un robot, qui peut être contrôlée de façon indépendante et dans différents modes de fonctionnement. Par exemple pour un robot manipulateur mobile, nous pouvons identifier trois *Ressources* : le manipulateur (contrôle indépendant du bras mécanique), le mobile (contrôle indépendant du véhicule) et le manipulateur mobile (contrôle du bras mécanique et du véhicule couplés, prenant en compte tous leurs degrés de liberté).

Chaque *Ressource* est décomposée en un ensemble de composants en interactions (i.e. une *Ressource* est donc considérée comme une architecture) : un ensemble de *Commandes* chargées de générer des données actionneurs à partir de données capteurs (réalisation d'un asservissement) ; un ensemble de *Perceptions* chargées de générer des données de haut niveau d'abstraction à partir de données capteurs ; un ensemble de *Générateurs d'évènements* chargés de générer des événements diffusés vers les niveaux supérieurs de l'architecture ; un ensemble d'*Actions* représentant les actions atomiques qu'est capable de contrôler la Ressource, responsables de la commutation et de la reconfiguration des *Commandes* via des événements diffusés par certains *Générateurs d'événements* ; un ensemble de *Modes* qui représentent les modes de fonctionnement d'une *Ressource* (téléopération, autonomie, coopération), chaque *Mode* s'appuyant sur un ensemble d'*Actions* et de *Générateurs d'événements* afin de réaliser les ordres provenant des niveaux supérieurs ; un unique *Superviseur de Ressource* qui est chargé de la commutation des *Modes* au sein d'une *Ressource* en fonction des ordres provenant de la couche supérieure.

L'architecture d'un contrôleur, quant à elle, est composée d'un ensemble de *Ressources* (sous-architectures),

d'un ensemble de *Contrôleurs d'entrées/sorties* responsables de l'échantillonnage des données capteurs et de l'affectation des données de commande aux actionneurs (ainsi qu'éventuellement d'autres traitements sur ces données), ainsi qu'un *Superviseur Global* qui se charge d'activer/désactiver les *Ressources* en fonction d'ordres provenant d'un opérateur humain. Le *Superviseur Global* est à l'heure actuelle le niveau de supervision le plus haut dans l'architecture, il traduit les ordres provenant d'un opérateur humain en une série d'ordres qu'il envoie aux *Ressources*, et qui peuvent être exécutés en parallèle ou en séquence en fonction de la nature même de ces ordres.

A partir du patron proposé par cette architecture générique, les développeurs peuvent conduire l'analyse de toute architecture de contrôle de robot. Ils ont cependant alors besoin de formalismes et / ou langages pour modéliser et implanter cette architecture.

2.2 Langage à Composants

Le langage CoSARC est un formalisme de modélisation et un langage de programmation de haut niveau dédié à cette problématique. Il a été défini à partir de propositions déjà existantes au niveau des langages et modèles à composants en intégrant les besoins spécifiques liés au développement des contrôleurs logiciels de robots. Il propose aux développeurs de manipuler différentes catégories de composants.

Les *composants de représentation* sont introduits pour représenter les connaissances que le robot possède sur son environnement, sa partie opérative, sa mission etc. Ils seront utilisés pour représenter les entités à partir desquelles les décisions seront prises par les *composants de contrôle*.

Les *composants de Contrôle* sont introduits pour représenter les entités responsables des activités à partir desquelles est décidé de la réaction que le robot doit appliquer afin de réaliser sa mission, à différents niveaux d'abstraction. Ils seront utilisés pour représenter les différentes entités présentes dans l'architecture générique. Leur comportement réactif asynchrone est décrit par le biais des réseaux de Petri à Objets [2].

Les connections entre *composants de contrôle* sont réifiées à travers des composants appelés *connecteurs* qui encapsulent les protocoles d'interaction. Le protocole défini par un *connecteur* est décrit par un réseaux de Petri à Objets ce qui permet de donner une sémantique formelle à la composition des *composants de Contrôle*.

Enfin l'architecture globale d'un contrôleur, ou des sous-architectures comme celles relatives à chaque *Ressource*, est encapsulée dans un composant appelé *configuration*, ce qui permet de rendre réutilisable des assemblages complexes de *composants de contrôle*.

Après conception d'une *configuration*, le développeur peut décrire son déploiement via un ensemble de structures propres au langage CoSARC, qui permettent de représenter les noeuds de l'architecture matérielle du contrôleur, de réaliser le déploiement des *composants de contrôle* au sein de processus système en effectuant, de plus, l'ordonnancement des processus déployés chaque noeud.

3 Conclusion

La méthodologie proposée présente divers avantages. Tout d'abord l'objectif de réutilisabilité visé est atteint : tout composant est réutilisable et notamment les *connecteurs* réutilisables indépendamment des *composants de contrôle* eux-mêmes. L'intégration des réseaux de Petri à Objets donne une sémantique formelle à la composition des *composants de contrôle* via des *connecteurs*. Ainsi, le réseaux de Petri global de l'architecture d'un contrôleur peut être reconstitué à partir des réseaux de Petri des *composants de contrôle* et des *connecteurs*. Ceci donne au développeur la possibilité de détecter, dès la phase de modélisation, des aberrations (e.g. interblocages).

Le langage proposé contribue également à la traçabilité des composants pendant leur cycle de développement (modélisation et programmation) via la notion de raffinement. Le développeur modélise les caractéristiques structurelles et comportementales de chaque composant puis raffine le modèle en écrivant le code de chaque opération. Finalement des mécanismes de compilation permettent de traduire automatiquement l'ensemble des informations collectées afin de générer des composants exécutables au sein d'un environnement d'exécution spécifique. A titre d'exemple, le langage CoSARc propose de traduire les réseaux de Petri à Objets dans un format exécutable par un joueur intégré à cet environnement (ce qui résoud, entre autre, la difficile tâche de traduction des réseaux de Petri à Objets en code).

Notre équipe aborde actuellement, après la proposition de la méthodologie et du langage sous-jacent, la phase de validation et d'opérationalisation. Nous effectuons la conception et le développement d'un contrôleur de robot mobile entièrement basé autour de l'approche CoSARC. A plus long terme, nous développons l'atelier de génie logiciel et le middleware temps-réel au dessus duquel s'exécutent les composants écrits dans le langage CoSARC.

Références

[1] C. Szyperski, *Component Software : Beyond Object Oriented Programming*, Addison-Wesley, 1999.

[2] C. Sibertin-Blanc, High Level Petri Nets with Data Structure, *Proceedings of the 6th european workshop on Application and Theory of Petri Nets*, Espoo, Finland, 1985.

[3] J. D. Carbou, D. Andreu, P. Fraisse, Contrôle de robots autonomes basé sur les réseaux de Petri hybrides, *Conférence sur la modélisation des systèmes réactifs (MSR'01)*, Toulouse, France, 2001.

[4] F. Ingrand, Architectures Logicielles pour la Robotique Autonome, *Journées Nationales de la Recherche en Robotique JNRR'03*, Clermont-Ferrand, France, Octobre 2003.