

Réutilisation intelligente de programmes de vision en environnement distribué

Sabine Moisan¹ et Wided Lejouad-Chaari²

¹ Inria, Sophia Antipolis,
Projet Orion,
2004, route des Lucioles BP.93,
06902 Sophia Antipolis Cedex, France.
Tél : int+ 33 4 92 96 78 47, Fax : int+ 33 4 92 96 79 39
Sabine.Moisan@sophia.inria.fr

² ENSI Tunis,
Campus Universitaire La Manouba,
2064 La Manouba Tunis, Tunisie.
Wided.Chaari@ensi.rnu.tn

Résumé Les outils du domaine de la distribution offrent des opportunités techniques pour le partage d'informations et de connaissances. Nous utilisons ces moyens pour faciliter l'accès et la diffusion des programmes produits par la recherche. C'est l'objectif du *pilotage de programmes distribué* présenté dans cet article. Parmi les multiples applications, on peut citer le partage de bibliothèques de programmes entre équipes en vue de comparaison de résultats sur des jeux de test, l'accès à des programmes issus de diverses équipes d'un même domaine par des étudiants dans ce domaine, ou l'utilisation dans des domaines applicatifs (médecine, biologie, agronomie) de programmes complexes (en vision en particulier). Nous avons en parallèle commencé le développement d'une base de connaissances sur des programmes de vision développés à l'ENSI de Tunis et proposé un serveur Web pour la consultation et la modification à distance de cette base. Ces deux expériences sont complémentaires : la base de connaissances permet de valider et d'expérimenter les outils de pilotage et le serveur. L'article présente ces expériences et les perspectives attendues.

Mots clés pilotage de programmes, systèmes distribués, serveur Web, vision.

1 Introduction

Les outils du domaine de la distribution offrent des opportunités techniques pour partager informations et connaissances. Ces outils sont souvent utilisés pour diffuser les travaux de recherches des laboratoires, par exemple via des articles en ligne. Mais on peut aussi envisager leur utilisation pour faciliter l'accès et la diffusion des programmes, c'est l'objectif du *pilotage de programmes distribué* présenté dans cet article

En effet, de plus en plus d'applications en grandeur réelle impliquent des participants inter ou intra-entreprises disséminés sur divers sites. Les avancées de la technologie des réseaux, alliées aux techniques de pilotage de programmes, permettent d'envisager l'accès collaboratif à des programmes distants et le partage de programmes entre différentes disciplines. Parmi les multiples applications, on peut citer le partage de bibliothèques de programmes entre équipes de développement ou de recherche, en vue de comparaison de résultats sur des jeux de test, l'accès à des programmes issus de diverses équipes d'un même domaine par des étudiants dans ce domaine, ou l'utilisation dans des domaines applicatifs (médecine, biologie, agronomie) de programmes complexes (de traitement et d'analyse d'images par exemple). Le pilotage de programmes utilise des techniques d'*intelligence artificielle* pour permettre ces partages et ces échanges sans que le source des programmes ne soit diffusé. Ceci rend possible le seul partage de la compétence sur l'utilisation des programmes et favorise l'accès à des programmes complexes, même pour des utilisateurs novices.

Nous proposons d'étudier des outils soit pour la consultation distante de connaissances sur l'utilisation de programmes, soit pour le lancement d'une requête de traitement à distance avec des données (par exemple des images) locales, soit, à plus long terme, pour la construction collaborative de bibliothèques de programmes partagées.

2 Pilotage de programmes

Le but du *pilotage de programmes* est double. D'une part, il vise à automatiser l'utilisation de logiciels complexes, comme des bibliothèques de traitement d'images, grâce à des systèmes capables de planifier des traitements et d'en contrôler l'exécution. Un système de pilotage est un système à base de connaissances qui assure la sélection et l'enchaînement de programmes dans différentes configurations, grâce au raisonnement de son moteur et aux connaissances contenues dans la base (comme montré figure 1). Ceci permet la réutilisation « intelligente » [14] de programmes, sans modifier leur code et rend ces programmes accessibles à des utilisateurs non spécialistes, peu familiers des algorithmes et des techniques implantés dans les programmes. Le pilotage de programmes s'applique particulièrement bien aux domaines comme le traitement d'images ou du signal qui peuvent être utilisés dans des domaines très divers.

D'autre part, le pilotage permet une capitalisation et une structuration des compétences sur l'utilisation des programmes. Il offre ainsi un outil de gestion organisationnelle des programmes d'une équipe, en complément des outils de génie logiciel usuels. En effet, à chaque programme est associée dans la base de connaissances une description lisible et structurée, précisant son nom, son auteur, son rôle, ses entrées, ses paramètres, ses sorties, ses effets, etc. Ceci, ajouté à des informations sur les combinaisons usuelles des programmes entre

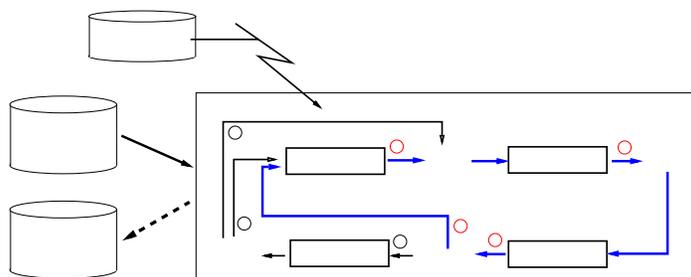


FIG. 1. Les étapes du raisonnement d'un système de pilotage. La *planification* construit un(e partie de) plan(1), l'*exécution* réalise l'appel effectif aux programmes du plan qui produisent des résultats (2), transmis à l'*évaluation* pour jugement (3). Si le jugement est positif (4), la planification continue. Si des erreurs ont été détectées (5), la *réparation* décide des actions de correction appropriées, ce qui peut relancer soit la planification (6) soit l'exécution (7).

eux, constitue à la fois une source de documentation et une mémoire d'entreprise sur les programmes, qui est de plus exploitable par un SBC pour automatiser le pilotage de ces programmes. Le processus de pilotage de programmes consiste à utiliser ces connaissances pour sélectionner les meilleurs programmes, lancer leur exécution en fonction de la requête (donc des données) de l'utilisateur et les enchaîner correctement.

Les techniques des systèmes à base de connaissances (SBC) répondent bien à ce double objectif de capitalisation et d'opérationnalisation, car elles donnent les moyens d'exprimer et de structurer les connaissances nécessaires sur l'utilisation des programmes de manière à la fois lisible, pérenne et opérationnelle.

Nous avons déjà réalisé plusieurs systèmes de pilotage de programmes dans différents domaines [3,15] grâce à une plate-forme logicielle qui offre un support complet et flexible pour construire des SBC [11]. L'existence de cette plate-forme facilite notre objectif de distribution, car elle permet d'identifier et d'isoler facilement les constituants à distribuer.

Avec l'extension du Web, de nombreux projets ont vu le jour pour faciliter la distribution de bases de connaissances et d'ontologies [6,7,5,4] ou de méthodes de résolution de problèmes distribués [2]. Cependant, peu de recherches se sont intéressées aux problèmes spécifiques du pilotage distribué, qui combine les problèmes de migration de données, de connaissances et de programmes exécutables, ainsi que le partage de connaissances (citons les travaux passés sur PowerTask [13]). Certains travaux sur les « data-intensive decision flow » [10] commencent à aborder des problèmes proches mais en se focalisant sur les aspects middleware et données, plutôt que sur la modélisation des connaissances. Ces travaux sont complémentaires des nôtres.

2.1 Distribution du pilotage de programmes

Dans un environnement distribué il devient nécessaire d'une part, pour des utilisateurs distants de pouvoir utiliser les facilités d'une couche de pilotage au dessus des programmes pour traiter leurs données et, d'autre part, pour des experts de consulter les informations sur les programmes existants, voire pour introduire de nouveaux programmes dans un SBC (ainsi

que les connaissances associées pour les utiliser). Si des programmes liés à une même problématique sont développés dans des équipes disséminées, le pilotage distribué peut permettre une résolution coopérative de problème, dans laquelle chaque étape représente le savoir-faire d'une équipe. Il permet aussi à des chercheurs qui souhaitent comparer leurs programmes de confronter les expériences et d'enrichir les résultats de recherche.

Ainsi, l'étude de la distribution d'outils de pilotage de programmes s'avère intéressante afin de prendre en compte l'accroissement des capacités informatiques (connectivité via Internet et le Web), l'hétérogénéité (environnements de développements différents et données différentes) et l'accroissement des besoins (utilisation optimale des ressources et des compétences d'autrui). En traitement d'images l'utilisation de l'Internet a amené la mise en place de jeux de tests d'images communes. Le pilotage de programmes distribué offre, en plus de ce partage de données, la possibilité de partager des programmes, ce qui répond aux besoins actuels des chercheurs et des industriels.

Dans ce but, nous avons étudié différentes modalités de distribution d'un SBC. Un SBC se décompose classiquement en un moteur d'inférence (lui-même décomposable en sous-parties), une base de connaissances et des interfaces graphiques. Un système de pilotage comporte en plus les programmes à piloter et les données. Tous ces différents constituants peuvent être localisés sur différents sites. Différents cas de distribution sont donc à envisager : par exemple, un utilisateur final peut lancer une requête de traitement sur des programmes et des données locaux, avec un moteur et une base de connaissances distants. Chaque forme de distribution peut engendrer des problèmes différents dus à la complexité des données à transférer, à l'hétérogénéité des langages et des environnements de développement, à des besoins spécifiques de ressources (matérielles ou logicielles) pour l'exécution de certains programmes, à la gestion de la cohérence des connaissances d'usage sur les mêmes programmes en différents sites, etc. Par exemple, un moteur sur un site et une base de connaissances sur un autre site engendrent des accès coûteux [8].

Nous allons étudier des solutions (en nous appuyant, par exemple, sur des expériences passées [9]) qui rendent la distribution le plus possible transparente aux utilisateurs. Notre objectif à long terme est de proposer un outil de coopération et de partage qui permette à des équipes de mettre en commun des programmes et les connaissances sur leur utilisation et de bénéficier ainsi des développements d'autres équipes. Dans un premier temps, nous avons en parallèle démarré deux expériences complémentaires qui vont dans ce sens.

3 Premières expériences

Nous avons d'une part développé une base de connaissances sur les programmes de vision de la bibliothèque Grimage du Grift/Pina de l'ENSI Tunis. D'autre part, nous avons étudié un prototype de serveur pour la consultation et la modification à distance d'une telle base. Ces deux expériences sont complémentaires car la base de connaissances permet de valider et d'expérimenter à la fois les outils de pilotage et le serveur.

3.1 Une base de connaissances en vision

Une base de connaissances sur des programmes de traitement d'images de la bibliothèque *Grimage* développés à l'ENSI de Tunis a été réalisée [12]. Une partie de cette base est visualisée figure 2. Cette bibliothèque se compose de programmes de reconstruction 3D et d'indexation. Les programmes d'indexation basés sur le contenu (contour et niveaux de gris) permettent d'optimiser les recherches dans des bases de données d'images. Ils consistent en l'isolation d'un objet dans une image, sa description par des invariants et la quantification de ces descripteurs. Les traitements de reconstruction 3D se font à partir soit d'enveloppes de points (convexes ou non) soit de coupes parallèles. Ces traitements sont intéressants à piloter de par les différents types de données pris en compte et les différents algorithmes applicables.

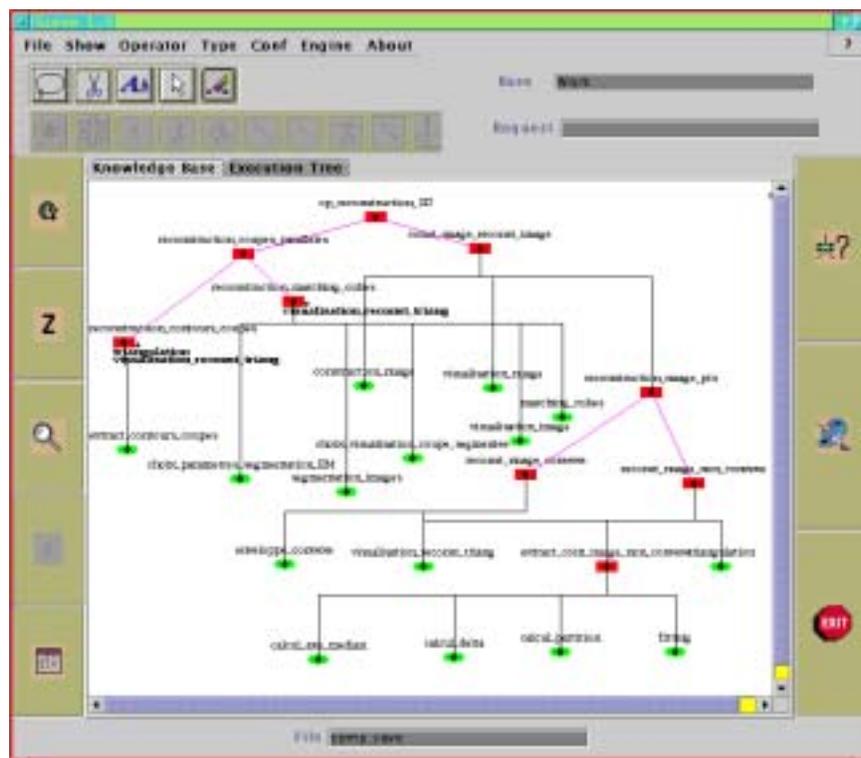


FIG. 2. Vue partielle de la base de connaissance Grimage. Les ovales représentent les programmes, les rectangles des décompositions : séquences (râteaux) ou choix (fourches)

La construction de la base de connaissances a consisté à identifier les différents programmes existants et leurs enchaînements, puis à les représenter par des *opérateurs* dans la base (respectivement primitifs pour les programmes et composites pour les enchaînements). Les opérateurs composites représentent des décompositions en sous-opérateurs (en séquence, alternatifs ou en parallèle). Aux opérateurs sont attachés des critères utiles au raisonnement du SBC (critères de sélection pour la planification, par exemple). Ces critères sont souvent intuitifs et difficiles à exprimer pour les experts, comme l'a montré cette expérience.

3.2 Serveur de partage de bases de connaissances

Nous avons aussi créé un prototype de serveur sur un site Web expérimental [1] donnant accès à des bases de connaissances en pilotage de programmes afin de mettre ces bases à disposition de la communauté scientifique grâce à une interface graphique et ainsi de partager les capacités d'utilisation de programmes. Les utilisateurs peuvent consulter ou enrichir l'expertise contenue dans une base via Internet de manière sécurisée.

Le serveur propose une architecture qui permet l'utilisation du prototype via Internet, que l'on soit sur navigateur (par utilisation d'Applets ou par des pages HTML) ou simplement à travers une application Java. L'architecture implantée est multi-niveaux avec trois niveaux principaux correspondants à trois composants : des esclaves (chargés de la gestion transparente des ressources réparties sur chaque site), un maître (offrant une vue globale de toutes les ressources et assurant la gestion des utilisateurs) et des clients (donnant une vue locale du système pour un utilisateur distant).

Le prototype permet à des groupes d'utilisateurs de partager des bases de connaissances de manière sécurisée et permet une administration partagée des bases de connaissances. Cependant, il ne permet pas encore l'exécution distante, c'est-à-dire de lancer un moteur d'inférences sur des données. La solution envisagée est de créer des objets Java pour le moteur et les programmes à piloter (binaires ou bibliothèques). Grâce à JNI (Java Native Interface) ces objets locaux seront manipulables à distance.

4 Conclusions

Les systèmes de pilotages distribués ouvrent des perspectives intéressantes, d'une part, pour des utilisateurs distants qui souhaitent utiliser les facilités de résolution optimale du pilotage pour traiter leurs données et, d'autre part, pour des experts et des programmeurs pour diffuser et partager des bibliothèques de programmes, sans surcoût prohibitif.

Au vu des résultats encourageants des premières expériences, nous allons continuer nos travaux pour aborder la distribution des système de pilotage dans son ensemble. Dans un premier temps, nous allons améliorer la base de connaissances de test et le serveur Web. La base doit prendre en compte tous les programmes de Grimage disponibles et refléter leurs dernières versions. En effet, son contenu doit suivre les évolutions des programmes pour continuer à remplir ses rôles de capitalisation et d'utilisation opérationnelle. Le site Web actuel répond aux besoins de partage de connaissances en termes de parcours (*browsing*) et de modification de bases de connaissances, mais il reste à implanter l'exécution effective à distance de programmes sur des données images fournies par un utilisateur et à améliorer les performances et l'ergonomie.

A plus long terme, notre objectif est de proposer un outil de coopération et de partage qui permette à des équipes de mettre en commun, outre des données, des programmes et leurs connaissances d'usage au travers de serveurs de pilotage. Nous visons une architecture générique pour créer des serveurs de pilotage distribués adaptables selon les besoins des applications.

Références

1. T. ben Salah. Conception et réalisation d'un outil sécurisé pour la gestion des ressources distribuées sur internet : application au pilotage de programmes. Rapport de stage de 3^{ème} année, ENSI Tunis, juin 2000.
2. V.R. Benjamins, B. Wielinga, J. Wielmaker, and D. Fensel. Brokering Problem Solving Knowledge at the Internet. In *EKAW'99, European Knowledge Acquisition Workshop*, volume 1621 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, May 1999.
3. V. Clément and M. Thonnat. A Knowledge-based Approach to the Integration of Image Processing Procedures. *Computer Vision, Graphics and Image*, 57(2) :166–184, March 1993.
4. R. Dieng. *IEEE Intelligent Systems Special Issue on Knowledge Management and the Internet*. IEEE, May-June 2000.
5. J. Domingue. Tadzebao and WebOnto : Discussing, Browsing, and Editing Ontologies on the Web. In *11th Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, April 1998.
6. J. Euzenat. Corporate Memory through Cooperative Creation of Knowledge Bases and Hyper-documents. In M. Musen B. Gaines, editor, *10th Knowledge Acquisition Workshop KAW'96*, volume 36, pages 1–18, Banff, Canada, November 1996.
7. A. Farquhar, R. Fikes, and J. Rice. The Ontolingua Server : a Tool for Collaborative Ontology Construction. *International Journal of Human-Computer Studies*, 46 :707–727, 1997.
8. W. Lejouad. *Etude et application des techniques de distribution pour un générateur de systèmes à base de connaissances*. Thèse, université de Nice, Nice, juillet 1994.
9. W. Lejouad-Chaâri and F. Mouria-Béji. High-Level Communication Protocol in a Distributed Multiagent System. In Angel Pasqual del Pobil Jose Mira and Moonis Ali, editors, *IEA-AIE'98 International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems*, Lecture Notes in Artificial Intelligence 1415, Benicasim, Espagne, June 1998. Springer-Verlag.
10. F. Llirbat, R. Hull, B. Kumar, J. Su, G. Zhou, and G. Dong. Optimization Techniques for Data-intensive Decision Flows. In *Int. Conference on Data Engineering (ICDE)*, San Diego, February 2000.
11. S. Moisan. Réutilisation et générateurs de systèmes à base de connaissances : le *framework* BLOCKS. *TSI*, 20(4) :529–553, 2001.
12. A. Ouali. Vers la conception d'une base de connaissances pour le pilotage de programmes d'analyse d'images. Rapport de stage de 3^{ème} année, ENSI Tunis, juin 2000.
13. T. Parmentier, D. Ziébelin, and F. Rechenmann. Environnement de résolution de problèmes distribué. In *RFIA'98, 11ème congrès Reconnaissance des formes et intelligence artificielle*, pages II-265–274, janvier 1998.
14. M. Thonnat and S. Moisan. What can Program Supervision Do for Software Reuse? *IEE Proceedings - Software. Special Issue on Knowledge Modelling for Software Components Reuse*, 147(5) :179–185, October 2000.
15. M. Thonnat, S. Moisan, and M. Crubézy. Experience in Integrating Image Processing Programs. In *International Conference on Computer Vision Systems*, Las Palmas, Canary Islands, January 1999.