Internet, environnement complexe pour agents situés

Laurent Magnin

Centre de Recherche Informatique de Montréal
550 rue Sherbrooke Ouest, bureau 100
Montréal (Québec) Canada H3A 1B9
laurent.magnin@crim.ca

RÉSUMÉ. *Les agents qui auront à circuler sur Internet seront plongés dans un environnement très complexe, très vaste et très dynamique. Des recherches spécifiques dans le domaine de l'intelligence artificielle située sont donc nécessaires afin de rendre ces agents adaptatifs. C'est ainsi que nous avons réalisé des agents génériques « GenA » indépendants des plates-formes d'exécution d'agent, par l'intermédiaire d'une interface, « GenAi ». Une seconde étape de notre projet consistera à offrir à ces agents une vue standardisée de leur environnement, par l'intermédiaire de bibliothèques dynamiques de fonctionnalités.*

ABSTRACT. *Internet will be a very complex, large and dynamic environment for agents. Some specific research about Situated Artificial Intelligence has to be done to allow such agents to be adaptive. That way we realised "GenA" generic agents that are able to run on different Agent Platforms by using a "GenAi" interface. A second step will be to provide to such agents an uniformed vision of their environment by the way of dynamic functions' libraries.*

MOTS-CLÉS : *environnement, agent situé, multi-plate-forme, agent adaptatif, Internet.*

KEY WORDS: *Environment, Situated Agent, Multiplatform, Adaptive Agent, Internet.*

Introduction

Agents purement logiciels[1] tels que décrits dans [GAS 98] et agents situés peuvent paraître antinomiques. Pourtant, il existe un domaine pour lequel les deux se rejoignent. Il s'agit d'un réseau tel qu'Internet [MA 99 ; MAE 99 ; MÜL 99]. En effet, comme nous allons le voir, un tel réseau offre des caractéristiques bien différentes d'une simple plate-forme multi-agent développée par une équipe et qui se trouve déployée sur un ordinateur unique, voire un réseau limité de machines. À l'inverse, Internet est, ou plutôt sera, un environnement ouvert, peuplé d'agents fort hétérogènes, de services divers, mais surtout de plates-formes d'exécution d'agents incompatibles. Nous utilisons ici le futur car il n'existe pas encore de véritable mise en place d'applications basées sur les agents[2] sur le Réseau [JOS 99 ; MA 99]. Si l'expansion d'Internet, au travers du Web, s'est faite grâce à l'introduction d'un langage commun tel qu'HTML, il est fort probable qu'une telle unification ne se fasse pas dans le contexte des agents. En effet, les difficultés techniques sont plus importantes dans ce domaine, mais surtout, il existe désormais de nombreux compétiteurs qui savent ce que peut rapporter le fait d'imposer un standard. D'où une vive compétition, ce qui promet plusieurs années avant qu'un des compétiteurs ne puisse s'imposer, si cela est possible.

Avant que n'arrive cette unification au travers d'un standard, imaginons plus précisément l'univers Internet tel qu'il sera du point de vue « agent ». Ces agents auront à vivre dans un monde dont le sous-sol est bien structuré, au travers de TCP-IP et autres protocoles. Un monde constitué d'une très vaste étendue d'ordinateurs pouvant apparaître et disparaître. Seuls certains de ces ordinateurs pouvent accueillir ces agents — par l'intermédiaire de plates-formes *ad hoc* —. Lesquels agents peuvent éventuellement se déplacer d'un serveur à un autre (agents mobiles [WON 99]). Malheureusement, ces différentes plates-formes sont incompatibles entre elles. De plus, cohabitent sur celles-ci des agents ayant tous une histoire plus ou moins longue, utilisant des protocoles et *modus operandi* différents. Enfin, il s'agit d'un univers qui ne s'arrête jamais, en perpétuelle évolution tant conjoncturelle que structurelle (nouvelles plates-formes agents, nouveaux protocoles, etc.).

Nous voyons bien que dans un tel contexte, il est nécessaire que ce soit aux agents de s'adapter au réseau, et non l'inverse. Nous allons voir dans le reste de cet article les pistes que nous proposons pour cela.

---

[1] Nous ne parlons pas ici d'agents situés dans un environnement simulé [MAG 96].

[2] Si l'on considère de vrais agents, et non des filtres destinés à trier du courrier électronique…

Des plates-formes hétérogènes

Le principal obstacle concernant l'introduction d'applications basées sur les agents semble concerner leur « intelligence ». C'est ainsi que sont menés de nombreux travaux concernant les protocoles de communication, les modes de coopération, etc.

Si cette « intelligence » est nécessaire, encore faudrait-il au préalable que ces agents puissent « s'exécuter » sur des ordinateurs du réseau au travers de plates-formes *ad hoc*. Malheureusement, les plates-formes qui voient le jour (Aglets, Voyager, Grasshopper, etc.), bien que de plus en plus souvent basées sur Java, restent incompatibles entre elles [AGL 99 ; VOY 99 ; GRA 99]. Plus exactement un agent de l'une des plates-formes ne peut s'exécuter sur une des autres. Si cela ne pose pas de problème dans un univers « fermé », il en est tout autrement dans le cas d'Internet. En effet, que faire si l'on veut, par exemple, concevoir un agent destiné à trouver sur le réseau un billet d'avion aux meilleures conditions entre Montréal et Paris et que les serveurs d'Air France et d'Air Canada imposent de recevoir[3] des agents devant s'exécuter sur des plates-formes incompatibles entre elles ? Dans ce cas, il semble bien difficile d'imposer par nous même une modification d'au moins un de ces deux serveurs… Et nous ne pensons pas qu'apparaisse bientôt une norme telle que Fipa ou Masif [FIP 99 ; OMG 99] capable de s'imposer à ces différents « serveurs d'agents[4] ». Devrions-nous dès lors prévoir deux types d'agents, voire plus si d'autres serveurs de compagnies aériennes sont eux aussi incompatibles ?

Une façon de contourner cette difficulté serait de prévoir un type d'agent « universel », capable de s'exécuter sur nombre de plates-formes incompatibles entre elles. Cependant, techniquement cela n'est pas envisageable. En effet, il serait nécessaire que cet agent 1) soit construit suivant un modèle d'héritage multiple et 2) soit reconnu comme étant un agent natif par ces plates-formes ; deux conditions nécessitant un accès au code source de ces différentes plates-formes. Ce qui est impossible d'un point de vue commercial : seules les méthodes « publiques » d'accès aux fonctionnalités offertes par ces plates-formes sont fournies.

N'existerait-il donc pas de solution à cette incompatibilité ? Si, elle existe. Pour cela il faut changer notre façon de considérer nos agents logiciels, en les voyant comme des agents situés dans un environnement particulier, en l'occurrence un

---

[3] Un des intérêts particulier à Internet concernant la technologie agent se trouve dans le concept d'agent mobile [WON 99]. En effet, un agent s'exécutant en local sur un serveur (de compagnie aérienne par exemple) permet à celui-ci d'exercer un contrôle plus serré sur les transactions effectuées et une non-divulgation de celles-ci à l'extérieur, renforçant ainsi la sécurité. De plus, un agent effectuant des transactions en local peut permettre d'accroître l'efficacité du service en limitant les échanges sur le réseau.

[4] Utilisons désormais ce terme « serveur agent » pour désigner une instance de plate-forme multi-agent en fonctionnement sur un ordinateur spécifique, et de ce fait capable d'accueillir des agents. À noter qu'un ordinateur peut contenir plusieurs serveurs agents au même instant, ceux-ci s'exécutant sur des machines virtuelles Java différentes.

serveur d'agent sur Internet. Ce changement d'approche impose que ce soit à l'agent de s'adapter. Et ce non pas en fonction du réseau dans sa globalité, mais à la plate-forme multi-agent sur laquelle il veut se rendre. C'est ainsi que nous avons développé au *centre de recherche informatique de Montréal* (Crim) un modèle d'agent générique[5] « GenA » capable de s'exécuter sur diverses plates-formes multi-agents commerciales, par l'intermédiaire d'une interface « GenAi » spécifique à chacune de ces plates-formes [MAG 99].

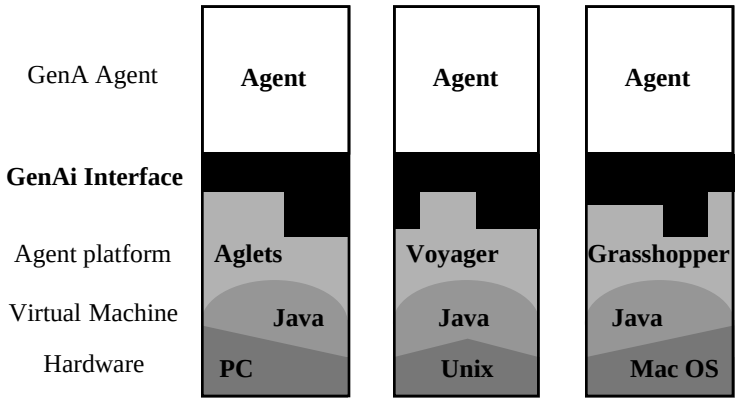

Figure 1. *Interface GenAI : couches logicielles*

Une implémentation fonctionnelle de cette interface GenAi a été réalisée au Crim. Pour l'instant, voici les trois plates-formes visées :

1. ASDK, Aglets Software Development Kit [AGL 99], développé par le laboratoire *IBM Tokyo Research Laboratory* ;
2. Voyager [VOY 99], produit commercial de la compagnie *ObjectSpace* ;
3. Grasshopper [GRA 99], produit commercial de la compagnie allemande *IKV++*.

Ainsi nos agents GenA sont-ils capables de s'exécuter sur ces trois plates-formes sans modification de leur code, peuvent communiquer les uns avec les autres indépendamment de leur plate-forme hôte, et peuvent même se déplacer d'un serveur d'agent à un autre *a priori* incompatible… Il ne nous semble pas ici nécessaire de faire une liste des avantages que cela représente. Toutefois, ceux-ci sont nombreux et très importants, tant du point de vue technique que commercial. Afin d'en faire la démonstration, une application d'agence de voyage virtuelle basée sur ces agents telle que présentée dans [MER 98] est en cours de réalisation.

---

[5] « Générique » au sens de non spécifique (à une plate-forme).

Cependant, concevoir et implémenter une interface telle que GenAi n'est pas suffisante. Il faut en effet que les agents puissent en tirer parti. Pour cela, il sera nécessaire d'offrir des serveurs spécifiques, capables par exemple de faire fonctionner en parallèle plusieurs plates-formes de types différents afin de faciliter la migration d'agents de l'une à l'autre. Il faudra également prévoir un mécanisme permettant à un agent de connaître la nature d'un serveur afin de pouvoir adopter l'interface idoine. Enfin, comme nous l'avons déjà dit, ce réseau sera en permanente évolution. De ce fait, la prise en compte de nouvelles plates-formes devra être effectuée, ainsi qu'une gestion des différentes versions de ces plates-formes ; à chaque version pouvant correspondre une interface GenAi spécifique.

Traditionnellement, lorsqu'un protocole est modifié ou lorsqu'une nouvelle fonctionnalité est proposée, un humain modifie la configuration informatique pour en tenir compte. Dans le cadre de nos agents, cela n'est plus possible. En effet, un certain nombre d'entre eux auront une vie autonome sur une longue durée. D'où la nécessité de les doter de mécanismes adaptatifs leurs permettant de survivre et de continuer à remplir leur mission malgré la variabilité de leur environnement ; dans le cas présent la modification ou l'ajout de nouvelles plates-formes multi-agents et interfaces GenAi. Par exemple, il serait de peu d'utilité d'introduire un nouveau type de plate-forme si aucun des agents déjà présents sur Internet n'était capable 1) de s'en apercevoir et 2) d'en tirer parti. Ces notions d'adaptation à un environnement en évolution afin de pouvoir survivre (dans notre cas pouvoir « s'exécuter » sur un serveur) et ainsi d'atteindre son but sont au cœur de l'intelligence artificielle située.

Pour clore l'aspect exécution des agents sur des « serveurs d'agents », signalons que ceux-ci devront exercer un certain contrôle sur leurs agents, pour des raisons de sécurité (afin par exemple de protéger des données sensibles en fonction d'un niveau d'accréditation accordé aux agents) et d'efficacité (pour éviter par exemple qu'un agent n'accapare tout le temps CPU). À l'inverse, les agents devront être capables de se « défendre » vis-à-vis de ces serveurs, afin par exemple de protéger leurs données (numéro de carte de crédit, liste de sites déjà explorés, etc.) ; de s'allouer suffisamment de temps CPU ; voire d'être capables de se relancer à partir d'une copie sur un autre serveur en cas de soudaine indisponibilité du premier. L'on entrevoit ici toutes les difficultés, mais également toutes les richesses induites par l'arrivée d'agents sur Internet.

Bibliothèques de fonctionnalités

Tenons désormais pour acquis le fait (essentiel) qu'un agent tel que de type GenA puisse s'exécuter de façon uniforme sur un grand nombre de serveurs d'agents à travers Internet. Cependant, l'environnement de tels agents ne se limitera pas à cet aspect. Il leur faudra en effet compter, ou plus exactement, tirer profit des autres agents qu'ils rencontreront, dans une optique de coopération propre aux

systèmes multi-agents. Et la difficulté spécifique à ce domaine est dès lors fortement accrue. En effet, il ne s'agit plus ici de concevoir un ensemble d'agents capables de coopérer au sein d'une application « unifiée », mais bien de concevoir des agents autonomes qui devront s'adapter à d'autres agents ayant été conçus et mis en place par des compagnies commerciales diverses et concurrentes.

Encore une fois, nous ne pensons pas qu'un standard d'échange entre agents puisse s'imposer rapidement au détriment de tous les autres, malgré des tentatives telles que KQML [HUH 99]. Au contraire, plusieurs protocoles et *modus operandi* vont cohabiter avec plus ou moins de bonheur. Ceci pour des raisons commerciales, mais également technologiques (il n'existe encore aucun modèle satisfaisant « d'interaction » entre agents, à la différence de langages opérationnels et complets tels que HTML et XML [GLU 99]). Comment dans ce cadre concevoir des agents un tant soit peu capables de s'intégrer dans des communautés diverses ? De nouveau, il nous semble déraisonnable d'imaginer un « agent universel », capable d'échanger et de coopérer avec nombre de type différents d'agents.

Notre approche est plus pragmatique. Avant même de penser à un modèle cognitif de haut niveau rendant l'agent capable de s'adapter suivant les circonstances, offrons lui la possibilité de tirer parti de fonctionnalités offertes par son environnement. Par exemple, si un agent se trouve en face d'un autre agent disposant comme seul mode de communication un dialecte spécifique, il serait intéressant que notre agent puisse disposer d'un « module » correspondant à ce dialecte. Tout comme il dispose d'une interface GenAi lui permettant de s'exécuter sur une plate-forme donnée. C'est donc ainsi que nous souhaitons constituer nos agents GenA, autour d'une bibliothèque de fonctionnalités téléchargeables dynamiquement. Bien entendu cela n'est qu'une première étape, laquelle nous semble toutefois indispensable. Restera dans un deuxième temps à concevoir les mécanismes 1) permettant un accès uniforme du point de vue de l'agent à ces fonctionnalités (par exemple appel par l'intermédiaire d'une méthode identique au différentes versions du module de communication), 2) faisant en sorte que ces modules soient chargés à bon escient et dynamiquement en fonction du contexte de l'agent et enfin 3) permettant à celui-ci de tirer parti de nouvelles fonctionnalités non prévues lors de sa conception.

Ainsi, nous aurons circonscrit l'environnement dans lequel l'agent devra s'adapter : celui-ci ne sera plus son environnement externe en tant que tel, mais une recomposition de cet environnement sous la forme d'une liste de fonctionnalités mises à plat et standardisées ; facilitant dès lors la conception des algorithmes (apprentissage, etc.) devant permettre à « l'intelligence » de l'agent de s'adapter à son environnement. À noter que notre approche ne se limite pas à modifier le mécanisme de perception des agents. En effet, les fonctionnalités offertes aux agents par l'intermédiaire de cette bibliothèque peuvent comporter des modules « cognitifs », par exemple, des modules de type « système expert ». Voire des

modules faisant directement appel à d'autres agents spécialisés (par exemple, à des agents qui, ayant un accès direct aux informations provenant des marchés financiers, sont dédiés au calcul du taux de change entre monnaies).

Pour arriver à mettre en place ces bibliothèques de fonctionnalités, nous comptons emprunter deux directions. D'une part, en se basant sur la méta-modélisation de nos agents [GUE 96]. Ce travail devant se faire en collaboration avec Houari Sahraoui du Crim [SAH 95]. D'autre part, en concevant un agent comme étant un système d'exploitation offrant des services en interne. Pour cela, Linux est un bon exemple. En effet, ce système d'exploitation permet de charger et de décharger dynamiquement des libraires, voire de gérer en même temps plusieurs versions d'une même libraire. À regarder également le développement actuel d'applications capables de se mettre automatiquement à jour, en fonction de la disponibilité de nouvelles versions présentes sur des serveurs Internet.

Conclusion

Nous avons vu que l'introduction d'agents logiciels au sein d'un réseau tel qu'Internet pose de nombreuses questions et difficultés nouvelles. Notamment du fait que ces agents auront nécessairement à s'adapter à un environnement très vaste, complexe, hétérogène, non propriétaire, en constante évolution, etc. Dès lors, de nombreux travaux de recherche devront être menés dans le cadre de l'intelligence artificielle située pour arriver à mettre en place des applications multi-agents inter-opérables sur Internet.

Un travail de ce type est en cours au sein du Crim. La première phase de ce projet a consisté à concevoir et implémenter une interface, GenAi. Cette interface permettant à nos agents génériques GenA de s'exécuter sur diverses plates-formes multi-agents *a priori* non compatibles ; à ce que des agents GenA présents sur deux plates-formes différentes puissent dialoguer ; voire à ce qu'un agent GenA puisse migrer d'une plate-forme à n'importe quelle autre. Pour l'instant seules trois de ces plates-formes ont été prises en compte. Cette approche a été validée au travers d'une application de démonstration basée sur une agence de voyage virtuelle.

Cette première étape franchie, nos agents auront encore à s'adapter aux modes d'interactions des agents déjà présents sur le réseaux ; ceux-ci étant encore une fois très hétérogènes. Pour résoudre ce problème, notre approche consiste à fournir à un agent GenA une bibliothèque de fonctionnalités, dépendante de son environnement mais transparente pour lui. Par exemple, la fonctionnalité « communication asynchrone » dépendra de la plate-forme supportant l'agent, mais respectera vis-à-vis de celui-ci une interface commune. Ainsi, ayant une vision standardisée de son environnement, il sera beaucoup plus facile à l'agent de s'y adapter.

Remerciements

Nous tenons à remercier tout particulièrement le laboratoire universitaire Bell Canada (LUB), lequel par l'intermédiaire du Réseau de calcul et de modélisation mathématiques (rcm2), a financé en grande partie le travail de recherche concernant l'implémentation de l'interface GenAi.

Cette interface a été efficacement implémentée par El Hachemi Alikacem de l'équipe « Génie logiciel et ingénierie des connaissances » du Crim.

Nos remerciements vont également à Didier Naranin et à Nicolas Besson, stagiaires provenant du DESS GLA de l'université Paris VI, pour la réalisation de l'application de démonstration et de validation de GenAi.

Bibliographie

[AGL 99] « Aglets by IBM Tokyo Research Laboratory » http://www.trl.ibm.co.jp/aglets/

[DUR 99] Durfee, E. H. (1999). Distributed Problem Solving and Planning. *Multiagent Systems*. G. Weiss. Cambridge, London, The MIT Press, p. 121-164.

[FIP 99] Fipa (1999). « Foundation for Intelligent Physical Agents » http://www.fipa.org/

[GAS 98] Gasser, L. and J.-P. Briot. « Agents an Concurrent Objects: Briot-Gasser Interview » http://www.lis.uiuc.edu/~gasser/AgentsAndObjects-07.html

[GRA 99] « Grasshopper by IKV++ » http://www.ikv.de/products/grasshopper/index.html

[GLU 99] Glushko, R. J., J. M. Tenenbaum, et al., « An XML Framework for Agent-based E-commerce », *Communications of the ACM*, vol. 42, p. 106-114.

[GUE 96] Guessoum, Z., « Un environnement opérationnel de conception et de réalisation de systèmes multi-agents », Thèse de doctorat du Lip 6, Université de Paris VI, 1996.

[HUH 99] Huhns, M. N. and L. M. Stephens, « Multiagent Systems and Societies of Agents », *Multiagent Systems*, G. Weiss, Cambridge, London, The MIT Press.

[JOS 99] Joshi, A. and M. P. Singh (1999). « Multiagent Systems on the Net ». *Communications of the ACM*, vol. 42, p. 39-49.

[MA 99] Ma, M. « Agents in E-commerce », *Communications of the ACM*, vol. 42, p. 79-80.

[MAE 99] Maes, P., R. H. Guttman, et al., « Agents That Buy and Sell », *Communications of the ACM*, vol. 42, p. 81-91.

[MAG 96] Magnin L., « Modélisation et simulation de l'environnement dans les systèmes multi-agents », Thèse de doctorat, Laforia, Université de Paris VI.

[MAG 99] Magnin L., Alikacem E.H., « GenA : Multiplatform Generic Agents », MATA'99 Workshop, 6-8 octobre 1999, Ottawa, Canada.

[MER 98] Merlat, W., « Objets et agents pour Systèmes d'Information et de Simulation ». Thèse de doctorat du Lip 6, Université de Paris VI, 1998.

[MÜL 99] Müller, J. P. and M. Pischel, « Doing Business in the information Marketplace », *Actes d'Autonomous Agents*, Seattle, WA USA, ACM.

[OMG 99] OMG. « MASIF : Mobile Agent System Interoperability Facility » http://www.fokus.gmd.de/research/cc/ima/masif/index.html

[VOY 99] Voyager, « Voyager by ObjectSpace » http://www.objectspace.com/voyager/

[WON 99] Wong, D., N. Paciorek, et al., « Java-based Mobile Agents », *Communications of the ACM*, vol. 42, p. 79-80.

[WOO 99] Wooldridge, M., « Intelligent Agents », *Multiagent Systems*. G. Weiss. Cambridge, London, The MIT Press, p. 27-78.

[SAH 95] Sahraoui, A. H., « Application de la méta-modélisation à la génération des outils de conception et de mise en oeuvre des bases de données », Thèse de doctorat du Lip 6, Université de Paris VI, 1995.