Chapitre 7

Programme de recherche

7.1 Axes de recherche

7.1.1 Le problème central des systèmes multi-agents : l'autonomie

Formalisation de l'autonomie

Les systèmes multi-agents constituent le dernier pas de l'évolution des logiciels vers des systèmes complexes à composants de plus en plus autonomes. Alors que depuis les débuts du domaine l'autonomie est considérée comme une propriété fondamentale de l'agent, le concept n'a jamais été défini de façon très précise. Comme le montre le thème de la récente conférence AAMAS'03 (`http://csce.uark.edu/~hexmoor/AAMAS-03/AAMAS-03-cfp.htm`), le sujet acquiert aujourd'hui une place prépondérante, et à juste titre il me semble.
Je considère ici que les agents sont plutôt des super-composants, de taille et de complexité supérieure à celle des composants. Si composants et agents se ressemblent sur bien des points, je pense que vouloir faire au niveau composant ce qu'on fait au niveau agent rend difficile le contrôle au niveau agent. Certes, on bénéficierait de tous les avantages de l'autonomie des composants, mais on perdrait du contrôle au niveau agent, ce qui rendrait le système encore plus difficilement contrôlable.
Mon sujet de thèse initial sur la formation de coalitions, une fois plongé dans un contexte de commerce électronique, a évolué vers la problématique plus fondamentale : que peut-on contrôler dans un système dans lequel les agents sont totalement autonomes ? Pour respecter l'autonomie, il faut que le protocole ne pose de contraintes sur des données observables (les communications) afin de les rendre vérifiables. On ne contrôle pas ce qui se passe dans l'agent, seulement ce qui en sort.
Cette élégante solution a cependant un inconvénient majeur : la vitesse de convergence est très lente. Plus d'autonomie, cela signifie certes plus de flexibilité, d'évolutivité, d'adaptativité, un système plus ouvert, mais aussi moins de contrôle.
Mon objectif est donc de parvenir à une formalisation de la notion d'autonomie afin d'être capable de fixer le degré d'autonomie en fonction du niveau de contrôle désiré. On rogne un peu sur l'autonomie pour gagner en contrôle.
La validation du système se ferait alors en deux temps :

- valider le protocole spécifié dans un certain formalisme (Réseau de Petri par exemple), en faisant l'hypothèse que l'agent respecte les contraintes spécifiées ;
- valider l'agent en environnement fermé (génie logiciel) :

  validation off-line : vérifier que le comportement de l'agent n'entre pas en conflit avec les spécifications du protocole.

  validation on-line : vérifier à l'exécution, d'une par observation des communications par le système, et d'autre part en incluant dans l'agent un "composant-conscience" qui contrôle qu'en interne, l'agent respecte bien les contraintes données.
- valider l'agent en environnement ouvert (commerce électronique, web-services) :

  validation on-line : vérifier à l'exécution, d'une par observation des communications par le système, et d'autre part en incluant dans l'agent un "composant-législateur" qui informe l'agent ce qu'est la norme, qui ne repose que sur les communications.

En environnement fermé, on a une autonomie un peu restreinte, mais avec plus de contrôle. En environnement fermé, l'autonomie est plus grande, mais avec moins de contrôle.
Toute la difficulté est alors de spécifier les protocoles et les composants "composant-conscience" et "composant-législateur" pour le rendre le moins intrusif possible.
De mon point de vue, l'autonomie intervient beaucoup plus dans la conception des interactions (et donc protocoles) que dans celle des agents.
Un groupe de travail francophone (avec industriels) sur le sujet est en cours de constitution.

L'autonomie au niveau interaction

Originellement, l'autonomie s'appliquait à un groupe de personnes qui se régissent par leurs propres lois. Transposé au monde des agents, l'autonomie au niveau interaction doit permettre aux agents de choisir (librairie de protocoles), adapter (paramétrage de protocoles), modifier (fusion de morceaux de protocoles) ou créer les protocoles qu'ils utilisent.
Plusieurs travaux exploratoires ont montré la complexité du sujet (travaux de Jean-Luc Koning et d'Alexandre Pauchet en France), mais l'intérêt me semble majeur : apporter de l'autonomie au niveau interaction introduira c'est certain un degré supplémentaire de flexibilité et d'adaptabilité.
Cet axe me semble fondamental pour tout ce qui est lié aux systèmes multi-agents : la coopération, l'interopérabilité, les interactions, les organisations, ...

7.1.2 Systèmes Multi-Agents et Génie Logiciel

Dans les Systèmes Multi-Agents, le génie logiciel occupe une place importante. Comme le souligne Jean-Pierre Briot, les agents sont le dernier pas de l'évolution de la programmation sur le plan de l'abstraction (intentions, plans), sur le plan du découplage (après les composants) et sur le plan de la liberté d'action (autonomie).
Les approches habituelles de génie logiciel pour les systèmes multi-agents adoptent la vision classique externe du système : les agents sont conçus, puis implémentés et enfin exécutés sans qu'ils interviennent dans le processus. Or, des capacités d'introspection pourraient permettre aux agents de s'observer en train de se construire, de s'exécuter, d'interagir, de s'organiser. Ils ne seraient plus alors une marionnette dans les mains de leur concepteur, mais un acteur de leur conception.
Cette approche aurait peut-être encore plus de succès en Extreme Programming, où l'entité développée est très liée à son concepteur/développeur, et où l'expérimentation et l'amélioration continuelle nécessite de nombreuses interactions entre l'humain et le logiciel.

7.1.3 Représentation des préférences

Durant ma thèse, j'ai eu besoin d'utiliser un formalisme de représentation des préférences, mais tous reposaient sur une hypothèse de transitivité plus ou moins forte. Estimant qu'une préférence est une comparaison locale entre deux options qui ne doit pas tenir compte des autres options, j'ai proposé un nouveau formalisme qui ne se base pas sur cette hypothèse. Ce formalisme permet notamment de calculer de façon naturelle la préférence d'un groupe.
Les préférences sont très utilisées en théorie de la décision, et ce nouveau formalisme s'avère prometteur pour définir de nouveaux opérateur. De même, en classification, on utilise un concept proche de celui de la préférence, avec les mêmes difficultés. J'ai décidé d'approfondir ma recherche dans ce domaine en reprenant contact avec un de mes rapporteur, Richard Émilion. Une collaboration tripartite avec Amal El Fallah est envisagée.

7.1.4 Systèmes large échelle et tolérance aux fautes

La puissance de calcul est de plus en plus disponible sous la forme de machines en réseau (grappe, grille). Les applications réparties large échelle – notamment les Systèmes Multi-Agents large échelle – utilisent un grand nombre de ressources sur des durées qui rendent la probabilité de panne non négligeable.
iquer tous les agents, il est nécessaire de choisir quels sont les plus critiques. Les approches existantes

consistaient à transmettre des indices de criticité de la couche SMA à la couche système, qui devait alors décider qui et comment répliquer les agents. Estimant qu'il s'agissait d'un goulet informationnel, j'ai cherché à définir plus précisément la criticité. J'en suis arrivé à la considérer comme les conséquences d'une faute en terme de Qualité de Service (QoS) ; la probabilité de faute est aussi prise en compte. Ce modèle est très général, mais prédire les conséquences de la faute est très dépendant de l'application. Un vaste chantier s'ouvre pour trouver des solutions réutilisables pour permettre à des applications large échelle de fonctionner le mieux possible du point de vue de l'utilisateur (la QoS du système), en prenant en compte ces possibilités de faute. Il semble nécessaire d'inclure dans le système des connaissances du concepteur de l'application, des préférences de l'utilisateur de l'application, des connaissances dynamiques de l'application (par observation en cours d'exécution), des connaissances dynamiques sur l'utilisation des ressources et d'autres connaissances. L'objectif est alors de permettre l'intégration de toutes ces connaissances pour prendre la meilleure décision.
Cette approche semble des plus prometteuses de par sa généralité, mais elle nécessite encore de la recherche pour être opérationnelle. L'utilisation d'agents pour superviser l'exécution de l'application permet une flexibilité et une richesse d'échange informationnelle qui devraient donner des meilleurs résultats que l'approche classique.

7.1.5 Simulation de phénomènes complexes

L'approche Système Multi-Agents

Les simulations utilisent généralement deux types de modèles : soit un modèles global sous forme d'équations qui doit être résolut mathématiquement avant d'être simulés ; soit des modèles locaux de comportement des éléments et de leurs interactions qui peuvent être implémanté directement. Indéniablement, la première approche est beaucoup plus rapide sur une machine isolée ; mais elle présente de nombreux inconvénients. 1) La puissance de calcul est aujourd'hui beaucoup plus disponible sous la forme d'un large réseau de machines (grappes, grilles, ...) que sous celle d'une unique ordinateur puissant. La possibilité de distribuer le calcul sur le réseau est alors un facteur primordial qui détermine la vitesse d'exécution. La deuxième approche étant distribuée par nature, elle peut reprendre l'avantage sur des topologies matérielles adéquates. 2) Dans l'approche globale, le modèle mathématique doit être calculable, ce qui contraint et bride les possibilités de modélisation. Dans les approches distribuées, il n'y a pas de recherche de solution globale ; la limite est alors la calculabilité en un temps raisonnable des modèles individuels et des interactions. 3) Une modification même mineure d'un système nécessite dans le premier cas de recommencer tout le calcul, après avoir avoir trouvé la solution du nouveau système. Au contraire, dans le deuxième cas, la modification, le remplacement ou la suppression d'éléments du système peut se faire sans arrêter le système, ce qui facilite les expérimentations.
L'approche distribuée permet d'analyser le comportement du système, à n'importe quel instant, à n'importe quel endroit, et à n'importe quelle échelle.
Les calculs par approche distribuée utilisant un grand nombre de ressources (processeurs, mémoire, réseau), et ayant une durée de vie imporante, il est nécessaire de prendre en compte la possibilité d'erreur et de crash d'un agent (voir tolérance aux fautes).

Simulation multi-échelle

La modélisation d'un système se fait à plusieurs échelles ; selon le point de vue choisi, les structures ne sont pas les mêmes. L'utilisation simultanée de modèles à des échelles différentes – un modèle multi-échelle – présente plusieurs avantages. 1) Lors de la modélisation, il est difficile de se concentrer sur tous les niveaux en même temps. Le fait d'avoir plusieurs modèles à différentes échelles permet de se focaliser sur une certaine partie du modèle, de la raffiner, puis de se tourner sur une autre, sans perte ni de cohérence, ni d'information. 2) En simulation, il est parfois trop coûteux de simuler toutes les éléments de niveau le plus bas. Or, certaines parties du modèle n'ont qu'une faible influence sur la partie étudiée ; ou le fait de considérer un modèle de niveau supérieur (moins précis) ne produit pas un résultat moins plausible. 3) Le modèle d'un niveau $n+1$ commet généralement une erreur par rapport au niveau n. Il se peut aussi que le niveau $n+1$ ne soit pas connu. On peut alors contruire le niveau $n+1$ en apprenant par observation le niveau n, ce qui lui permet de se corriger ou de se construire. La phase d'apprentissage peut se faire en début de simulation uniquement, ou de temps en temps en cours de simulation : les niveaux $n+1$ et n

coexistent seulement le temps nécessaire.
La simulation multi-échelle facilite la modélisation (ce qui concerne une grande partie de l'informatique), permet de simuler des systèmes beaucoup plus complexes, et autorise l'adaptation dynamique des modèles. La simulation de systèmes complexes me semble un angle d'attaque prometteur pour la compréhension du génome (on sait aujourd'hui simuler les interactions rigides, mais pas les niveaux au-dessus) ou pour la simultation de phénomènes physiques et sociaux.
Ce thème est à la rencontre de plusieurs de mes passions : les systèmes multi-agents d'une part, et les fondements de la Nature (la mécanique quantique) et de la Vie (la cellule et l'ADN). Mon postdoc porte sur la tolérance aux fautes de systèmes multi-agents large échelle, ce qui s'applique tout à fait à la simulation en physique des particules et en biologie cellulaire.

7.1.6 Systèmes Multi-Agents et Traitement Automatique de la Langue

L'augmentation extraordinaire de la quantité d'information disponible sous forme électronique nécessite et autorise des outils d'extraction d'information qui accèdent au niveau sémantique. Pour cela, de nombreux outils ont été développé, chacun s'attaquant à une phase spécifique du traitement.
Leur utilisation séquentielle pose cependant de nombreux problèmes. 1) La transmission de l'information d'une étape à l'autre n'est pas standardisée, ce qui nécessite une traduction - parfois avec perte - de la sortie d'un outil vers l'entrée du suivant. 2) Le traitement est organisé de façon séquentielle ; les données traversent les outils de traitement, comme dans l'approche fonctionnelle, alors que l'approche objet a montré tout son intérêt. 3) La principale difficulté est que le traitement séquentiel des données est très sensible aux erreurs, surtout si celles-ci ont lieu dans les premières étapes ; leur propagation peut engendrer des erreurs sur des structures plus grandes, comme par exemple le rôle d'un point. Même si ces erreurs sont détectées à des étapes ultérieures, il est très difficile de corriger les effets négatifs produits.
Je pense qu'une approche Système Multi-Agents (SMA) permettrait d'aider à répondre à ces problèmes. Un SMA est un ensemble d'agents qui s'exécutent en parallèle en collaborant pour la réalisation d'une tâche, en interagissant via l'échange de messages de haut niveau (actes du langages). Un agent est un processus (ou un thread) qui a en charge une donnée (SMA objet) ou un service (SMA fonctionnel), et qui possède des capacités d'intelligence artificielle comme le raisonnement, l'apprentissage (pour améliorer son comportement, apprendre de nouveaux concepts, ...).
Ces approches ont déjà été tentées, mais leur succès relatif peut être expliqué. Les approches SMA fonctionnel ne permettent pas une gestion fine des erreurs, notamment parce qu'il est difficile d'attacher des informations aux données, et d'y faire des traitements spécificiques, comme par exemple la dépendance causale entre connaissances. Les approches SMA objet ont des difficultés à réutiliser les outils existants, ce qui donne des résultats mitigés. Enfin, les approches réactives, pour lesquelles les agents ont un état simple et réagissent aux entrées en produisant une sortie par une fonction de calcul ne permettent pas non plus la réutilisabilité, mais de plus, leur convergence est très lente.
Pour ma part, je pense qu'une approche hybride, mélangeant agents-donnée et agents-service permettrait de bénéficier des avantages des deux approches. Une sélection statique ou dynamique des agents serait alors possible en observant la dynamique du système.