Simulation Distribuée Interactive sous MADKIT

Fabien Michel* — **Pierre Bommel**** — **Jacques Ferber***

** LIRMM Université Montpellier II - CNRS*
161, rue Ada - 34392 Montpellier Cedex 5 - France
{fmichel,ferber}@lirmm.fr

*** CIRAD*
73, rue Jean François Breton
34398 Monpellier Cedex 5 France
bommel@cirad.fr

RÉSUMÉ. *Sur la base d'un problème pratique portant sur la réalisation d'une application de simulation distribuée interactive, il s'agit pour nous d'évaluer la plate-forme* MADKIT *en tant que support de développement de Systèmes Multi-Agents (SMA) distribués. Inspirée du monde des jeux vidéo en réseau, l'application que nous présentons permet à plusieurs utilisateurs connectés ensemble d'interagir simultanément sur une unique simulation.*

ABSTRACT. *Through a practical problem that relies on the realization of a distributed interactive simulation (DIS), we want to evaluate the* MADKIT *multi-agent platform as a toolkit for building distributed multi-agents systems (MAS). Inspired by the world of networked video games, the application which we present allows several connected users to simultaneously interact on one single simulation.*

MOTS-CLÉS : *simulation interactive distribuée, jeux vidéo en réseau, SMA*

KEYWORDS: *distributed interactive simulation, networked video games, MAS*

1. Introduction

Inspirée du monde des jeux vidéo, l'application que nous présentons permet à plusieurs utilisateurs connectés en réseau d'interagir sur une même simulation. Une telle réalisation impose de nombreuses contraintes et constitue un véritable test pour une plate-forme multi-agents distribuée comme MADKIT[GUT 01, MIC 01]. Cette implémentation nous a ainsi permis d'évaluer MADKIT en tant que support de développement de SMA distribués.

2. Description de la Simulation Interactive : *"les Abeilles"*

Cette simulation est simple, il s'agit d'un essaim d'abeilles dirigé par une reine se déplaçant de façon aléatoire. Outre la possibilité pour l'utilisateur de paramétrer la simulation (taille et couleur des essaims lancés), celui-ci peut contrôler le déplacement d'une reine en lui imposant une politique de déplacement définie ou directement à l'aide de la souris. Il peut par ailleurs filtrer sa perception de l'environnement en choisissant de ne voir que les essaims d'une couleur.

Hormis les agents simulés, l'application utilise 5 types d'agent. L'agent *Scheduler* est chargé de l'ordonnancement de l'ensemble des agents de l'application. Les agents de type *Controller* contrôlent le déplacement d'une reine. Les agents *Epiphyt*[1] espionnent le système et envoient les informations receuillies aux agents de type *Projectionist* chargés de l'affichage. Enfin, les agents de type *InputListener* écoutent les interactions utilisateurs et les transmettent aux agents concernés (de type *Controller*).

La plate-forme MADKIT, basée sur un modèle organisationnel, gère la distribution de manière transparente en assurant la cohérence de la structure organisationnelle (partagée par tous les sites connectés) et l'acheminement des messages entre les agents locaux ou non. Ainsi sur la base d'une organisation comprenant l'ensemble des agents, la distribution consiste ici à répartir les agents suivant les besoins.

3. Première Solution : Architecture Client / Serveur

Inspirée du système X-Window, cette technique consiste à exécuter la simulation sur une seule machine. La visualisation et le contrôle se font à distance sur les machines clientes. Pour réaliser cette version, il suffit de placer les agents de type *InputListener* et *Projectionist* sur les machines clientes comme le montre la figure 1. Avec cette architecture, il n'y a pas d'ambiguïté sur l'état du monde. Cependant, le trafic réseau lié aux informations d'affichage est trop important pour ce type d'application.

---

1. Un épiphyte est une plante qui croît fixée sur d'autres plantes, mais sans les parasiter. Selon le même principe, [GIR 94] propose de développer des agents épiphytes pour évaluer des SMA. Il recommande de placer des sondes sur les agents du modèle, afin d'étudier leur comportement.

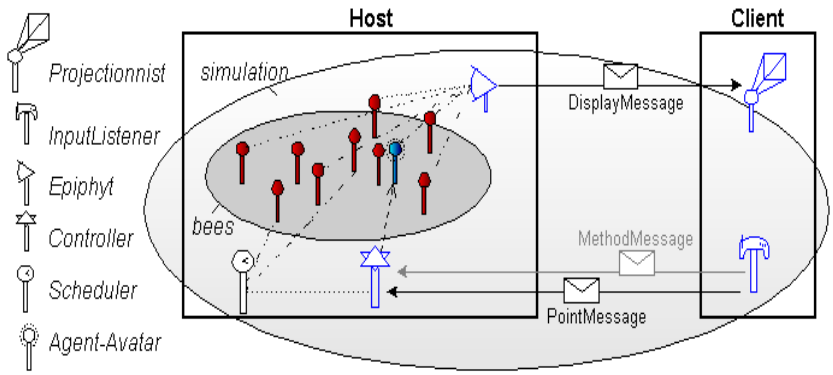

Figure 1. *Visualisation distribuée et contrôle à distance (Un agent avatar est ici un agent jouable : une reine). Host et Client sont deux noyaux* MADKIT *situés sur des machines distantes. La présence de plusieurs clients est par ailleurs possible.*

4. Deuxième Solution : Architecture *Dupliqua*

L'architecture par duplication (*dupliqua*) a été développée pour pallier aux inconvénients de la précédente en limitant le nombre de messages échangés. Très utilisée dans le monde des jeux vidéo, l'idée est que chaque machine héberge une copie de la simulation dans son ensemble[FIT 00, VEL 98]. Seules les interactions utilisateurs sont alors échangées de manière à ce que la simulation reste cohérente. Ce type d'architecture entraîne cependant des divergences entre les mondes simulés. Il existe par exemple une inévitable latence qui ne permet pas aux événements utilisateur d'être pris en compte au même moment sur l'ensemble des machines.

Pour corriger ces incohérences, la technique ici utilisée consiste à prendre un des noyaux MADKIT pour référentiel (le *master*) : Il détient alors *la vérité* sur le monde. Les autres noyaux distants sont appelés *dupliqua*. Toutes les *S* secondes, l'agent *Epiphyt*, placé sur le noyau *master*, envoie la position de toutes les reines à tous les agents de type *Controller* sur l'ensemble des machines. Si la divergence dépasse un certain seuil, les agents *Controller* rectifient la position des abeilles en conséquence. La figure 2 illustre ce fonctionnement pour trois noyaux.

5. Conclusion

En assurant de manière transparente que tous les noyaux partagent la même organisation, la plate-forme MADKIT permet de réaliser rapidement des applications distribuées. Grâce à la structure organisationnelle, l'agentification des composants a pour principal intérêt de permettre à ces derniers de communiquer (notamment par broad-

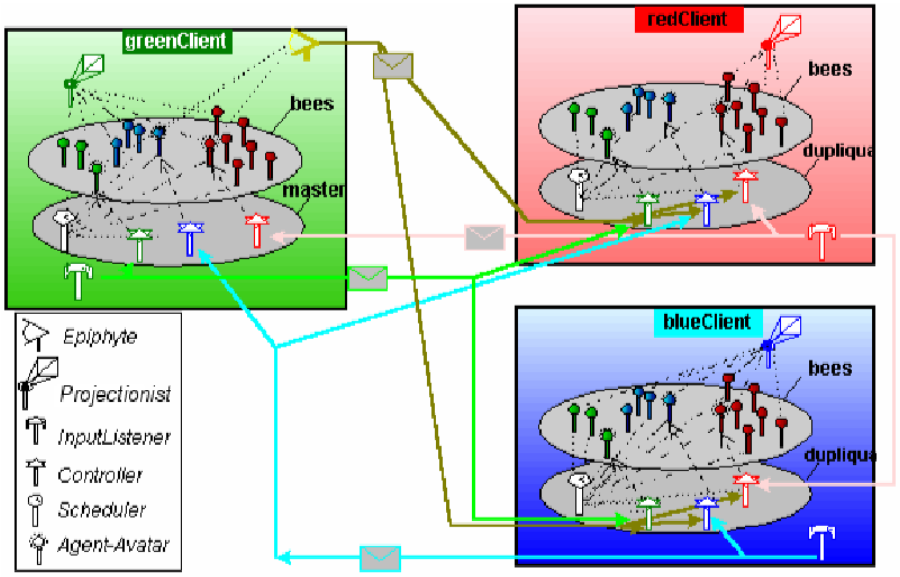

Figure 2. *Architecture Master-Dupliqua avec correction. Le greenClient est le Master, les deux autres sont les Dupliquas.*

cast sur un groupe) avec tous les agents présents dans l'organisation sans se soucier de leur localisation physique. En affranchissant ainsi les contraintes techniques liées à la distribution, MADKIT permet de se focaliser sur les problèmes d'algorithmique distribuée et de profiter au mieux des principaux atouts de la plate-forme (hétérogénéité et modèle organisationnel). Ce qui nous a permis, dans le cadre de notre application, de considérer les SMA comme une technologie logicielle distribuée simple et efficace.

6. Bibliographie

[FIT 00] FITCH C., « Multiplayer Technology Today », *http ://www.gamasutra.com/ features/ 20000120/fitch*$_0$4.htm, 20$January$2000.

[GIR 94] GIROUX S., PACHET F., PAQUETTE G., « Système d'information épiphyte : espionnage des interactions entre agents », *Deuxièmes Journées Francophones IAD&SMA JFIADSMA 94*, Voiron, 9-11 May 1994, page 211.

[GUT 01] GUTKNECHT O., MICHEL F., FERBER J., « Integrating tools and infrastructure for generic multi-agent systems », *Proceedings of the* 5^{th} *International Conference on Autonomous Agents AA'2001*, Montreal, June 2001.

[MIC 01] MICHEL F., FERBER J., GUTKNECHT O., « Generic Simulation Tools Based on MAS Organization », *Proceedings of the* 10^{th} *European Workshop on Modelling Autonomous Agents in a Multi Agent World MAMAAW'2001*, Annecy, France, 2-4 May 2001.

[VEL 98] VELLON M., MARPLE K., MITCHELL D., DRUCKER S., « The Architecture of a Distributed Virtual Worlds System », *Web site de Microsoft, http :// www.research.microsoft.com/vwg/papers/oousenix.htm,* 1998.