Paradigme de programmation par agents

David Levayer
Étudiant en maîtrise à l'Université
du Québec à Chicoutimi,
Élève ingénieur à Polytech Grenoble
Contact : david.levayer [at] gmail.com

Corentin Ricou
Étudiant en maîtrise à l'Université
du Québec à Chicoutimi,
Élève ingénieur à Polytech Grenoble
Contact : corentin.ricou [at] gmail.com

***Abstract*— Le document suivant est réalisé dans le cadre du cours "Patrons et modèles" de H. Mcheick à l'Université du Québec à Chicoutimi. La programmation par agents est un paradigme de programmation proposée il y a une quinzaine d'années. Elle vient palier les défauts de la programmation orientée-objet notamment pour le développement d'applications distribuées. Le développement d'agent est particulièrement pertinent si l'on souhaite disposer d'un réseau d'entités dynamiques et *context-aware*. Certains environnements profitent largement des caractéristiques particulières de ce type de programmation. Cet article présente brièvement le concept d'agent et propose une approche expérimentale basée sur le framework Jade. Il revient également sur les forces et les faiblesses de ce modèle et sur son intérêt en tant que patron architectural.**

***Index Terms*— Pattern, Object-oriented-programmation, Agent, Multi-agent system, Distributed application, Mobile, Adaptation, Jade**

I. Introduction

Depuis maintenant une quinzaine d'années [1], la programmation orientée-agent se démarque et s'affirme comme une évolution pertinente à la programmation orientée-objet. De part l'essor des moyens de communication et grâce à des capacités réseaux toujours plus élevées, le développement d'applications distribuées prend tout son sens. Il permet une meilleure répartition des tâches entre les individus d'un même réseau et offre de généralement un meilleur temps d'exécution. Les systèmes sont ainsi répartis sur plusieurs machines et interagissent entre eux : ils sont à la fois autonome et conscient de leur environnement. Ce dernier évolue au cours du temps et il est important de tenir compte de ces variables au cours de l'exécution.

Ce caractère dynamique et évolutif est au coeur de la programmation orientée-agent. Dans l'optique de faciliter le développement et l'intégration de tels systèmes, plusieurs *frameworks* ont été développés afin de supporter au mieux ce paradigme de programmation. Jade est l'un d'entre eux : ce framework développé en Java permet de faciliter et de standardiser le développement de système multi-agents. Dans un premier temps, cet article présente le contexte d'utilisation de la programmation orientée-agent avant de décrire plus en détails les caractéristiques de cette dernière. Nous présenterons ensuite le framework Jade avant de donner un prototype complet de système multi-agents. Une courte conclusion sera l'occasion de rappeler les différents avantages et les limites de cette approche.

II. Contexte d'utilisation

De nos jours, Internet est en constante évolution. Si le Web 2.0 favorise les interactions *Human-to-Human*, le Web 3.0 pourrait très bien être celui des interactions *Machine-to-Machine* [2]. L'essor des systèmes distribués et des intelligences artificielles requiert une évolution dans la façon de penser et de concevoir des systèmes à la fois dynamiques, intelligents et déployables à grande échelle. De ces besoins est né le concept d'agent : la création d'une entité autonome destinée à accomplir une tâche précise en fonction d'un environnement donné.

Même si ce concept prend tout son sens dans les systèmes distribués, on peut noter qu'il se démarque également dans d'autres contextes spécifiques. C'est notamment le cas lorsque le nombre de données à partager est très important ou lorsque la bande passante du réseau est faible. Dans les deux cas, l'envoi d'un agent "sur place" limite l'utilisation du réseau en privilégiant une résolution locale du problème. Les réseaux à grande latence (communication satellite par exemple) ou non-fiables (coupures fréquentes) sont également de bons exemples d'environnements où le déploiement d'agents possède un avantage certain. Enfin, on peut ajouter que l'utilisation d'agents peut être interessante lorsque les traitements réalisés ont une relation de causalité : l'agent évolue et se déplace dans un nouveau contexte en tenant compte des informations qu'il a accumulé lors de ces précédents traitements.

III. Programmation par agents

A. Présentation générale

En informatique, un agent est une entité logicielle qui agit de façon autonome. S'il est mobile, il peut également se transférer d'un environnement à un autre par ses propres moyens. Il interagit alors avec les services fournis par

l'environnement et tente d'accomplir un but préalablement défini [3]. Malgré leur caractère autonome, les agents communiquent entre eux afin d'acquérir et de partager de l'information. Idéalement, un agent possède un seul but. Ce dernier peut néanmoins être divisé en une multitude de sous-objectifs. Fait important, un agent est par ailleurs soumis à un cycle de vie : il naît (instanciation), réalise sa tâche (vie de l'objet agent) et meurt (libération mémoire lorsque la tâche est accomplie). Cette vision d'un agent est très importante car elle est l'analogie d'un être humain. De la même façon que la modélisation objet est basé sur la représentation des objets réels qui nous entourent, le concept d'agent représente une personne réelle. Comme tout être humain, un agent est donc autonome, pensant et en constante interaction avec son environnement. Il prend des décisions afin d'accomplir au mieux la tâche qui lui est confiée.

B. Différence entre agent et agent mobile

Tous les agents ne sont pas des agents mobiles. Un agent "classique" s'exécute intégralement depuis l'environnement dans lequel il a été crée alors qu'un agent mobile peut être amener à se déplacer vers un autre environnement. Ainsi, un agent mobile "s'envoie" vers l'environnement cible puis est reconstruit (nouvelle instance) afin de s'exécuter en local (traitement réalisé sur la machine hôte). Pour accomplir ses objectifs et récolter des informations, un agent classique va quant à lui communiquer de manière accrue, notamment avec d'autres agents distants. Ces échanges prennent le plus souvent la forme de messages. On perd alors certains avantages propres aux agents mobiles, notamment l'économie de bande passante ou le caractère "hors-ligne" de certaines opérations. En revanche, on conserve les avantages propres à la programmation par agent (et à la programmation distribuée de manière plus large) : réduction du temps total d'exécution, réduction du temps de réponse (parallélisme), équilibrage des charges ou encore déploiement dynamique. Les figures Fig. 1 et Fig. 2 illustrent les schémas d'exécution de ces deux types d'agents. Le framework Jade présenté ci-après propose un modèle d'agent non-mobile. Lorsqu'un agent veut des informations sur un contexte distant, il envoie un message via le système de messagerie intégré au framework.

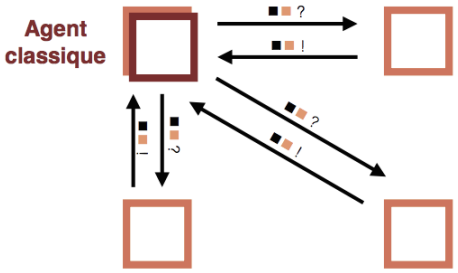

Fig. 1. Schéma d'exécution d'un agent classique

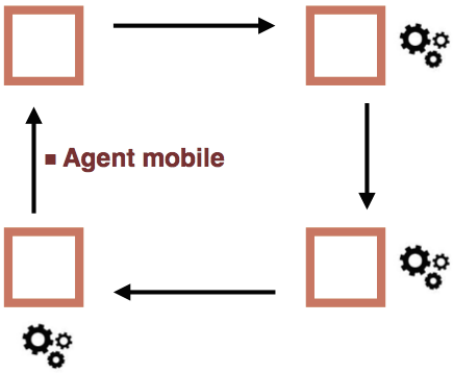

Fig. 2. Schéma d'exécution d'un agent mobile

C. Agents et patrons de conception

IV. Framework JADE

V. Prototype réalisé

VI. Conclusion

```
// Hello.java
import javax.swing.JApplet;
import java.awt.Graphics;

public class Hello extends JApplet {
   public void paintComponent(Graphics g) {
      g.drawString("Hello, world!", 65, 95);
   }
}
```

References

[1] T. Garneau and S. Deliste, *Programmation orientée-agent : évaluation comparative d'outils et environnements*, JFIADSMA, Lille, France, 28-30 octobre 2002.

[2] O. Boissier, *Multi-agent system*, support de cours, http://www.emse.fr/ boissier/enseignement/maop11/courses/introduction-4pp.pdf, consulté le 21 avr. 2015.

[3] R. Gray, D. Kotz, G. Cybenko and D. Rus, *Mobile agents: Motivations and state-of-the-art systems*, Tahyer School of Engineering, Dartmouth College, 2000.