Una Arquitectura de Software para Soportar Agentes Móviles Inteligentes

Edgardo A. Belloni† - Marcelo Campo

ISISTAN - Facultad de Ciencias Exactas
Universidad Nacional del Centro de la Provincia de Buenos Aires
Campus Universitario - Paraje Arroyo Seco (7000) Tandil, Bs. As., Argentina
† También, Comisión de Investigaciones Científicas de la Provincia de Buenos Aires
e-mail: {ebelloni,mcampo}@exa.unicen.edu.ar

1 Introducción

Los sistemas de agentes móviles representan actualmente uno de los paradigmas mas prometedores para la programación de sistemas ampliamente distribuidos y heterogéneos. Este tipo de sistemas involucra dos conceptos básicos: **sitios** servidores de recursos y **agentes** que poseen la habilidad de trasladarse a los sitios para acceder a sus recursos, o contactarse con otros agentes [10]. La movilidad de los agentes, es decir su habilidad de migrar de un sitio a otro, constituye la principal diferencia que este enfoque tiene con respecto a otras alternativas existentes para el desarrollo de sistemas distribuidos, como los de remote procedure call o remote evaluation.

Las principales ventajas, identificadas en el uso de agentes móviles, radican en: la potencial reducción de costos de comunicación global mediante la migración de las unidades de computación a los datos; y la posibilidad de distribuir computaciones complejas en diferentes hosts, posiblemente heterogéneos. Adicionalmente, los agentes móviles, han sido reportados en la literatura [5] como de gran utilidad para el desarrollo de aplicaciones de comercio electrónico, recuperación de información distribuida, workflow y groupware, entre otras. Tales aplicaciones se benefician al utilizar agentes móviles al explotar sus potenciales capacidades de: procesamiento autónomo y asincrónico; adaptabilidad dinámica a los cambios de su ambiente de ejecución; y tolerancia a fallas.

Los lenguajes orientados a objetos poseen características que permiten satisfacer parte de los requerimientos de la programación orientada a agentes [8] y en particular de agentes móviles [11]. Un agente puede modelarse mediante un objeto cuyos métodos representan sus habilidades, y las variables de instancia su estado mental. De esta forma, un objeto encapsulará, en sus métodos capacidades comportamentales de un agente y su conocimiento estará dado por el estado de sus variables de instancia.

Actualmente, se ha desarrollado un número importante de sistemas que utilizan el lenguaje orientado a objetos Java como soporte de programación de agentes móviles. Ejemplos de estos sistemas de soporte basados en Java, son: Aglets [4], Odissey [6], y Voyager [9]. Las ventajas que Java provee a la tecnología de agentes móviles, son fundamentalmente: su soporte multiplataforma; y la ubiquidad de su máquina virtual, que facilita la diseminación de agentes móviles en Internet. Adicionalmente, Java presenta características no encontradas en otros lenguajes que permiten la implementación directa de agentes móviles [11]. Por ejemplo, Java facilita la migración del código del agente y su estado mediante mecanismos de serialización de objetos y carga dinámica de clases locales o remotas. Por otra parte, el soporte de comunicaciones en redes provisto por Java, incluye sockets, comunicaciones URL, y un protocolo para el acceso a objetos distribuidos denominado remote method invocation (RMI).

No obstante, y a pesar de las bondades que los lenguajes orientados a objetos presentan para la programación orientada a agentes en general, y Java en particular para agentes móviles, estos lenguajes adolecen, sin embargo, de limitaciones. Estas limitaciones se evidencian a la hora de tratar con las actitudes mentales de los agentes debido a que se deben implementar distintos algoritmos de inferencia sobre las variables de instancia que representan el conocimiento del agente. Estos algoritmos son, en general, costosos de implementar y poco flexibles a cambios [2].

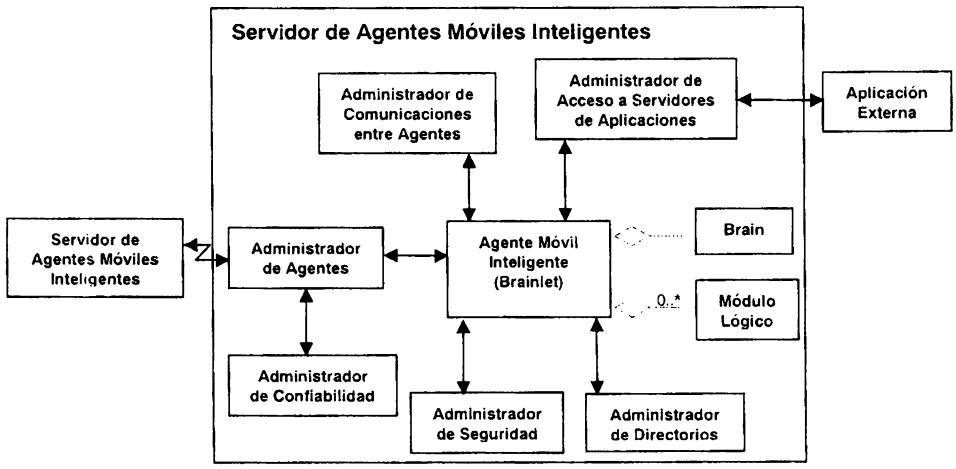
Por otra parte, los lenguajes lógicos permiten representar actitudes mentales en forma declarativa por medio de cláusulas lógicas. Estas cláusulas son interpretadas por algoritmos deductivos que, en el contexto de la programación de agentes, dan origen a razonamientos dependientes del conocimiento propio de los agentes. Desafortunadamente, los lenguajes lógicos presentan deficiencias en la programación de agentes debido a su imposibilidad de encapsular información y tratar con conocimiento privado [2].

En este artículo se presenta un enfoque para soportar el diseño y programación de agentes móviles inteligentes. Este enfoque es materializado esencialmente por una arquitectura de software basada en agentes móviles desarrollados en JavaLog [1][2], un lenguaje que integra los paradigmas de programación lógica y de orientación a objetos.

La siguiente sección describe la arquitectura de software básica necesaria para materializar el enfoque. Se caracterizan los diferentes componentes de esta arquitectura, y se describe cómo los agentes móviles integran el paradigma de programación lógica al de objetos. En la sección 3, se resumen los trabajos experimentales que se están llevando a cabo para evaluar la efectividad del enfoque. Por último, en la sección 4, se delinean algunas conclusiones.

2 Infraestructura para el soporte de Agentes Móviles Inteligentes

La siguiente figura presenta una arquitectura de software para el soporte de agentes móviles inteligentes. Esta se basa esencialmente en la arquitectura genérica de agentes móviles basados en Java descripta en [12]. El diagrama utiliza notación UML [7].



Descripción de Componentes

En esta arquitectura se reconocen diferentes componentes principales: un administrador de agentes; un administrador de comunicaciones entre agentes; un administrador de seguridad; un administrador de confiabilidad; un portal servidor de aplicaciones; y un administrador de directorios.

El administrador de agentes es responsable por la recepción de agentes para su ejecución en el host local, y del envío de agentes para ser ejecutados en hosts remotos. Antes de efectuar la migración de un agente, este administrador lo serializa junto con su estado. Luego, delega el envío de esta representación serializada del agente al administrador de confiabilidad, el cual asegura que el agente es recibido por el administrador de agentes correspondiente en el host destino. Al recibirlo, esta componente reconstruye el agente y los objetos que el referencia, a partir de su representación serializada, creando su contexto de ejecución.

El administrador de seguridad es responsable por la autenticación del agente antes de que este comience su ejecución. La máquina virtual Java subyacente invocará automáticamente a este componente para autorizar cualquier intento de utilizar recursos del sistema por parte de los agentes.

El administrador de comunicaciones entre agentes facilita la comunicación entre agentes móviles dispersos en diferentes hosts. Los agentes pueden utilizar el administrador de directorios para identificar la localización de un servidor de aplicaciones y luego migrar al host en el cual este servidor esta localizado. El portal servidor de aplicaciones representa para los agentes un punto de acceso a aplicaciones externas al sistema de agentes móviles, residentes en el host local, como por ejemplo acceso a servidores de bases de datos.

Esta arquitectura prescribe que los agentes constituyen objetos java compuestos, con capacidades de movilidad, persistencia, y comunicación con otros agentes. Cada agente ejecuta en su propio hilo de ejecución y esta implementado con un estilo de programación basado en call-backs, es decir basado en el modelo de delegación de eventos de Java. Durante su ciclo de vida, un agente recibe diferentes categorías de eventos en respuesta a sus acciones. Por ejemplo, si el agente migra hacia otro host ocurrirá un evento de movilidad justo antes y justo después de la migración. En este caso, son invocados los métodos call-back correspondientes (beforeDispatch y afterArrival). De esta manera, cada evento permite al agente determinar cómo reaccionar.

Brainlets: Agentes móviles que integran objetos y lógica

Los agentes móviles descriptos son implementados con JavaLog [1][2], un lenguaje que integra los paradigmas de orientación a objetos (Java) y de programación lógica (Prolog).

JavaLog utiliza objetos para modelar las habilidades de los agentes y cláusulas lógicas para modelar el estado mental de dichos agentes. Las habilidades de los agentes pueden estar representadas por capacidades de acción

como por capacidades de comunicación con otros agentes, mientras que el estado mental puede presentarse en forma de creencias, intensiones y compromisos. JavaLog define al módulo como concepto básico de manipulación. En este sentido, un objeto y un método son vistos como módulos. Un objeto es un módulo que encapsula datos y un método es un objeto que encapsula comportamiento

Los agentes móviles construidos con JavaLog son denominados Brainlets. Esto se debe a que estos agentes encapsulan un objeto compuesto denominado *brain*. Este objeto es una instancia de un interprete de módulos lógicos que utiliza cláusulas Prolog. Este interprete está implementado en Java y permite utilizar objetos dentro de cláusulas lógicas, así como embeber módulos lógicos dentro de código Java [1][2]. Cada agente es una instancia de una clase que puede definir parte de sus métodos en Java y parte en Prolog. La definición de los métodos de una clase en JavaLog puede estar compuesta por varios módulos lógicos definidos en métodos y módulos lógicos referenciados por variables de instancia.

3 Trabajo Experimental

El enfoque presentado, una arquitectura para el soporte de agentes móviles inteligentes, representa una de las líneas actuales inmersas en el proyecto de investigación y desarrollo de agentes inteligentes en el dominio de socialware[1], del ISISTAN.

En este contexto, se están desarrollando diferentes sistemas de información, multiagente e integrados a la world wide web, para asistir en las distintas actividades sociales de los miembros de diferentes comunidades virtuales. Estas actividades incluyen, por ejemplo, comercio electrónico solidario, planificación de reuniones, y convocatorias u ofrecimientos de voluntarios a campañas solidarias.

Las aplicaciones descriptas pueden verse beneficiadas con el uso de agentes móviles representando a diferentes miembros de una comunidad virtual al desarrollar actividades sociales en comunidades remotas. Tales agentes podrían transportar las preferencias y restricciones de su representado, encapsulando estas actitudes mentales en forma declarativa.

4 Conclusiones

En este artículo fue presentado un enfoque para el desarrollo de agentes móviles. Este enfoque es materializado por una arquitectura de software que integra los paradigmas de programación lógica y de orientación a objetos para soportar el diseño y programación de agentes móviles inteligentes.

La efectividad del enfoque está siendo evaluada al ser aplicado en el desarrollo de sistemas multiagente en el dominio de socialware.

Referencias Bibliográficas

1. A. Amandi, A. Zunino and R. Iturregui. Multi-paradigm Languages Supporting Multi-agent Development. In Multi-Agent System Engineering, F. J. Garijo and M. Boman (Eds.). Lecture Notes in Computer Science, Vol. 1647. pp. 128–139. Springer-Verlag, Berlin - Heidelberg - New York, 1999.
2. A. Amandi, A. Zunino, R. Iturregui. JavaLog: una integración de objetos y lógica para la programación de agentes. V Congreso Argentino de Ciencias de la Computación. Tandil, Bs. As., Argentina. Octubre de 1999.
3. F. Hattori, T. Ohguro, M. Yokoo, S. Matsubara, and S. Yoshida. Socialware: Multiagent Systems for Supporting Network Communities. Communications of the ACM. Vol. 42, No. 3, pp. 55-61. March 1999.
4. D. Lange and M. Oshima. Programming and Deploying Java Mobile Agents with Aglets. Addisson-Wesley Longman, Reading Mass., 1998.
5. D. Lange. Seven Good Reasons for Mobile Agents. Communications of the ACM. Vol. 42, No. 3, pp. 88-90. March 1999.
6. Odissey White Paper. General Magic Corp. Cupertino, Calif., 1998.
7. J. Rumbaugh, I. Jacobson, and G. Booch. The Unified Modeling Language Reference Manual. Reading, Addison-Wesley, 1999.
8. Y. Shohan. Agent-Oriented Programming. Artificial Intelligence, 60(1), pp. 51-92, March 1993.
9. Voyager White Paper. ObjectSpace Corp. Dallas, Texas, 1998.
10. J. White. Mobile Agents. In Software Agents, J. Bradshaw (Ed.), pp. 437-472. MIT Press, 1997.
11. D. Wong, N. Paciorek and D. Moore. Java-based Mobile Agents. Communications of the ACM. Vol. 42 - No. 3, pp. 92–102, March 1999.

[1] Socialware es el nombre que fuera acuñado por F. Hattori entre otros [3], para caracterizar al tipo de sistemas multiagente que proveen mecanismos de asistencia a las actividades de sociedades virtuales.