Extensión del UML para soportar el Modelado de Sistemas y Aplicaciones de Agentes de Software Móviles

Edgardo A. Belloni

ISISTAN - Facultad de Ciencias Exactas
Universidad Nacional del Centro de la Provincia de Buenos Aires
Campus Universitario - Paraje Arroyo Seco (B7001BBO) - Tandil, Bs. As., Argentina
E-mail: ebelloni@exa.unicen.edu.ar

Resumen. En este artículo se presenta un proyecto de investigación (MAM-UML) cuyo principal objetivo es el de mitigar la carencia de abstracciones y notaciones apropiadas que presenta el lenguaje de modelado de sistemas *Unified Modeling Language* (UML), en su versión estándar, respecto de la especificación de sistemas y aplicaciones basadas en agentes de software móviles. Se introducen las motivaciones para el proyecto y se describe el contexto en el que está inmerso, así como también, se enumeran las actividades de investigación y desarrollo que comprende. Adicionalmente se introduce, a modo de propuesta preliminar, un conjunto de vistas y modelos que capturan diferentes conceptos y características identificadas como relevantes para el modelado de agentes móviles durante la etapa de diseño de su desarrollo. Estos modelos prescriben la extensión del UML estándar, a través de los mecanismos que dicho lenguaje provee para ello.

1. Introducción

Actualmente, la Ingeniería de Software Orientada a Agentes – *Agent Oriented Software Engineering* o AOSE [22][21] – puede considerarse como un nuevo y razonable enfoque alternativo para el desarrollo de aplicaciones basadas en tecnología de Internet. Los desarrolladores de este tipo de aplicaciones pueden tratar de forma natural la inherente dinamicidad, compleja interactividad y heterogeneidad que las identifica, al concebirlas e implementarlas en términos de agentes de software, caracterizados estos por atributos tales como: autonomía, habilidad social, movilidad, reactividad y pro-actividad [9]. Ciertamente, la movilidad de los agentes – es decir, la capacidad de los agentes de software de poder trasladarse (migrar) de forma (potencialmente) autónoma a través de diferentes localizaciones en una red y actuando en representación de un usuario [20][4][10] – ha sido reconocida como una de las características que más beneficios promueve en el desarrollo de aplicaciones ampliamente distribuidas en entornos computacionales abiertos y dinámicos, como es el caso de las aplicaciones *web*. La movilidad de un agente involucra la transferencia de su código, datos y eventualmente su estado de ejecución, hacia los ambientes computacionales o sitios donde se localizan los recursos (entidades no autónomas, como por ejemplo: archivos, objetos, bases de datos, etc.) que el agente necesita acceder, o donde se localizan otros agentes a los cuales el agente necesita contactar e interactuar. Las principales ventajas, identificadas en el uso de agentes móviles [10], 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 [13] como de gran utilidad para el desarrollo de aplicaciones en diferentes dominios de sistemas basados en tecnología *web*, tales como: el comercio electrónico, la recuperación de información distribuida, asistentes personales y sistemas de *workflow* y *groupware*, entre otros. Dichas aplicaciones se benefician al utilizar agentes móviles al explotar sus potenciales capacidades de: reducción de la carga de la red; procesamiento autónomo y asincrónico; adaptabilidad dinámica a los cambios de su ambiente de ejecución; y tolerancia a fallas.

Desde fines de la década de los noventa, se han desarrollado numerosas y diferentes metodologías y técnicas de modelado orientadas a agentes [7][21][6]. Sin embargo, ninguna de estas puede considerarse aún como una metodología completa, capaz de soportar tanto el análisis como el diseño de sistemas multi-agente. Adicionalmente, se han realizado muy pocas contribuciones respecto de la definición de conceptos y notaciones para el modelado de agentes móviles, restringiendo consecuentemente aún más la utilidad de las técnicas y metodologías de AOSE existentes, para el desarrollo de aplicaciones en Internet. De hecho, hasta el presente, los trabajos realizados en el área de agentes móviles se han enfocado principalmente en el desarrollo de numerosos y diferentes plataformas o *frameworks* (denominados genéricamente como sistemas de agentes móviles) para el soporte tecnológico de la movilidad de los agentes de software [17]. Se ha invertido un gran esfuerzo en el desarrollo de tales infraestructuras, tanto en proyectos desarrollados dentro de la comunidad académica como así también dentro del ámbito comercial. Sin embargo, y a pesar de tales esfuerzos, existe hoy día muy poca evidencia acerca de un enfoque ingenieril serio y sistemático para el desarrollo de aplicaciones basadas en agentes móviles [7][11][6]. Hasta ahora, el enfoque usual para conducir este tipo de desarrollo se ha orientado principalmente a la etapa de implementación; las

aplicaciones son implementadas directamente – de forma *ad-hoc* – seleccionando y utilizando una determinada plataforma de agentes móviles, siguiendo con poco rigor o en nada alguna técnica de modelado o metodología para su diseño. Este enfoque produce, consecuentemente, especificaciones tanto incompletas como inadecuadas respecto de los aspectos relevantes de los agentes móviles.

En el presente, el *Unified Modeling Language* (UML) [18][16] esta siendo reconocido como el vehículo más adecuado para soportar el modelado de agentes, dado que si bien el UML representa un estándar para la especificación de sistemas de software basados en el paradigma de orientación a objetos, existen similitudes significativas entre este paradigma y el de orientación a agentes [21]. Adicionalmente, el UML provee mecanismos para su extensión permitiendo la incorporación de nuevos elementos al lenguaje. En este sentido, diferentes organizaciones de regulación de estándares – principalmente, la FIPA (*Foundation of Intelligent Physical Agents*) [3] y el OMG (*The Object Management Group*) [16] – han estado impulsando, tanto en la comunidad de investigadores como en la de desarrolladores de aplicaciones, la presentación de propuestas para la extensión del UML de manera de soportar conceptos propios de los agentes de software en general. Sólo recientemente, en los últimos dos años, se han iniciado algunos pocos trabajos intentando mitigar la carencia de conceptos y notaciones apropiadas en el UML básico o estándar, de manera de capturar las abstracciones relevantes de los agentes móviles durante las etapas de análisis y diseño de su desarrollo. Estos trabajos han propuesto extensiones a un tipo particular de diagrama de UML para el modelado de características sumamente básicas y en un nivel alto de abstracción, entre las cuales se pueden citar como más representativas las definidas en [11][15][1] y [12]. No obstante, y a pesar de que estos trabajos representan contribuciones valorables, aún constituyen enfoques muy abstractos e incompletos para el modelado de aplicaciones basadas en agentes móviles. Como señala Mouratidis entre otros, en [14], es necesario aún explorar cómo dar respuestas, a través del modelado, a cuestiones tales como: ¿Por qué un agente migra de una localización particular a otra?; ¿Hacia dónde y cuándo migra un agente?.

En este artículo se presenta un proyecto (MAM-UML) cuyo objetivo esencial es el de aportar de mejoras al UML estándar, para su adecuación en el desarrollo de software en el contexto del paradigma de agentes móviles, definiendo un conjunto coherente de vistas y modelos que integre la línea de las contribuciones mencionadas en el área. Esto incluye la incorporación de nuevas abstracciones y mecanismos para dar respuestas a los siguientes interrogantes, entre otros, que surgen también al modelar sistemas y aplicaciones de agentes móviles y fueran reconocidos a partir de experiencias de desarrollo propias [2]:

- ¿Cuáles son los tipos de agentes y recursos involucrados en la aplicación?; ¿A quién representa un agente (persona, organización u otro agente)?; ¿Cuáles son los roles que un agente puede interpretar en la aplicación?; ¿Cuáles son las plataformas o sistemas de agentes móviles involucrados, que brindan servicios y localizaciones para la ejecución de los agentes? y ¿Cómo estos sistemas se organizan en federaciones en el contexto de la aplicación?.
- El tipo de movilidad que un agente utiliza, es ¿Fuerte o débil?. Es decir, ¿Qué se transfiere ante la migración de un agente: su código, datos o estado de ejecución?. Adicionalmente, dicha migración es ¿Pro-activa o Reactiva?. Es decir, ¿Es el agente quien determina cuándo y hacia dónde migrar, o esto es responsabilidad de otra entidad?.
- ¿Cómo se definen las relaciones estructurales que un agente mantiene con los recursos que utiliza y comparte con otros agentes? y ¿Cómo se resuelven estos vínculos estructurales entre agentes y recursos, ante la migración de un agente de una localización a otra?.
- ¿Cuál es el patrón de estados y actividades que sigue un agente móvil durante su ciclo de vida? y ¿Cuáles actividades son realizadas en cuáles de las localizaciones?.
- La comunicación que se establece entre los agentes de una aplicación, es: ¿Sincrónica o asincrónica?; ¿Local o remota?; ¿Directa o indirecta?. Si es directa, ¿Esta se da de a pares, o entre un agente y un grupo?. Y si es indirecta, ¿Cuál es el mecanismo de intermediación utilizado?.

La siguiente sección describe el UML estándar y sus mecanismos de extensión. La sección 3 presenta el proyecto en términos generales, describiendo el contexto en el que está inmerso, sus objetivos y las actividades que comprende. Luego, en la sección 4, se introduce un conjunto de vistas y modelos que capturan diferentes conceptos y características tanto técnicas como funcionales, identificadas como necesarias para el modelado de agentes móviles. Finalmente, se presentan las conclusiones y se formulan algunos trabajos futuros.

2. UML: Un Lenguaje Estándar y Extensible para el Modelado de Sistemas

El *Unified Modeling Language* [18][16] es un lenguaje de modelado que representa, actualmente, un estándar para la visualización, especificación, construcción y documentación de los diferentes elementos que componen un sistema de software. No debe considerarse al UML como una metodología de desarrollo de

software sino como un lenguaje que provee un conjunto de abstracciones que conforman diferentes diagramas para modelar los diferentes aspectos (estáticos, dinámicos, de implementación, etc.) de un sistema. UML propone capturar dichos aspectos en diferentes modelos. Cada una de estos modelos incluye un conjunto de diagramas (diagrama de clases, de secuencia de mensajes, etc.), los cuales son representaciones gráficas de un conjunto de elementos o abstracciones (clases, relaciones estructurales entre clases, mensajes, objetos, etc.), y tienen como finalidad visualizar cierta porción de un sistema desde una perspectiva dada. Las categorías de modelos soportados por UML son las siguientes:

- **Modelos Estáticos.** Incluyen diagramas de: clases y objetos. Un diagrama de clases representa a un conjunto de clases, sus interfaces (agrupan operaciones), y las relaciones estructurales (asociaciones, agregaciones, composiciones y generalizaciones) que se dan entre dichas clases, así como también relaciones de realización y dependencia entre clases e interfaces. Estos diagramas pueden incluir paquetes (agrupan clases) representando, en general, subsistemas. Los diagramas de objetos representan instancias de clases (objetos) y *links* (instancias de relaciones). UML define también una notación para expresar colaboraciones genéricas en los diagramas de clases, para ser utilizadas en conjunción con paquetes parametrizables - generalmente definidos en diagramas aparte - en la especificación de patrones de diseño o incluso *O-O frameworks*.
- **Modelos Dinámicos.** Incluyen diagramas de: interacción, transición de estados y actividad. Un diagrama de interacción, ya sea de secuencia de mensajes o de colaboraciones, describe la interacción (el pasaje de mensajes) entre objetos para satisfacer una funcionalidad particular del sistema. Los diagramas de transición de estados representan máquinas de estados que incluyen elementos tales como: estados, transiciones, eventos y actividades. Finalmente, un diagrama de actividad describe el flujo de actividades que son necesarias desarrollar para satisfacer cierta funcionalidad.
- **Modelo de Casos de Uso.** Especifica las acciones que un sistema puede realizar desde un punto de vista externo. Incluye a los diagramas de casos de uso, que describen la funcionalidad requerida para el sistema junto con los actores que lo estimulan y con los cuales se comunica.
- **Modelos de Implementación.** Incluyen diagramas de: componentes y *deployment*. Los componentes representan agregados físicos de clases, que incluyen por ejemplo: los ejecutables y bibliotecas de vínculo dinámico. El diagrama de componentes muestra cómo las clases del diagrama de clases se corresponden con componentes físicos del sistema. El diagrama de *deployment* describe la configuración del *hardware* donde ejecutan los componentes físicos del sistema, representando la distribución de dichos componentes sobre diferentes plataformas o nodos, junto con los protocolos de comunicación que se establecen entre dichos nodos.
- **Modelo de Restricciones.** UML se complementa con un lenguaje declarativo denominado *Object Constraint Language* (OCL) [19]. OCL puede ser utilizado para especificar restricciones sobre un modelo o especificación UML, para incluir por ejemplo: invariantes, *pre* y *post*-condiciones de operaciones, y restricciones en la navegación de una red de objetos.

Los elementos presentes en los diferentes diagramas de UML tienen una semántica bien definida por el lenguaje. Dicha semántica esta representada por un *meta-modelo*, el cual es un modelo de los elementos provistos por UML. Es decir, cada elemento definido en un modelo de UML es una instancia de un concepto (clase o tipo) existente en el meta-modelo. Concretamente, los conceptos definidos en el meta-modelo de UML están especificados utilizando una técnica semi-formal basada en tres vistas complementarias:

- ***Abstract Syntax* (UML + Lenguaje Natural).** La sintaxis abstracta define las construcciones o conceptos existentes en el lenguaje y las relaciones entre ellas de una manera independiente de la notación (representación gráfica). La notación del lenguaje (sintaxis concreta) es definida luego haciendo corresponder los elementos notacionales a las construcciones definidas en la sintaxis abstracta. Dicha sintaxis abstracta es presentada en diagramas de clase UML mostrando las meta-clases que definen las construcciones y sus relaciones. Adicionalmente, se provee una pequeña descripción en lenguaje natural de las construcciones definidas.
- ***Well-Formedness Rules* (OCL).** Estas reglas describen la semántica estática de UML. La semántica estática de un lenguaje define cómo deben conectarse las instancias de las construcciones de manera que el resultado sea significativo. La semántica estática de cada meta-clase UML, excepto por las restricciones de ordenamiento y multiplicidad, se define como un conjunto de invariantes sobre una instancia de la meta-clase. Estos invariantes deben satisfacerse para que la construcción sea significativa. Cada invariante es definido como una expresión OCL.
- ***Semantics* (Lenguaje Natural).** Finalmente, se define el significado de las construcciones bien formadas. La semántica es descripta en lenguaje natural.

Por otra parte, UML ha sido diseñado para ser extensible dado que provee diferentes mecanismos que permiten introducir, si fuera necesario, nuevos elementos de modelado para dominios específicos, tales como: aplicaciones *web* o basadas en agentes, procesos de desarrollo de software, etc. Dichos mecanismos pueden ser utilizados, de manera aislada o conjunta, para definir nuevos elementos con distinta semántica, propiedades, y notación gráfica que los elementos de modelado del UML estándar. Los mecanismos predefinidos por UML para su extensión, son los siguientes:

- ***Stereotypes*.** Son utilizados para crear un nuevo elemento a partir de uno ya existente, asignándole una nueva semántica, generalmente más específica que la del original. Un estereotipo (*stereotype*) puede entenderse como un meta-tipo que define propiedades (a través de *tagged values*), restricciones adicionales (a través de *constraints*) e incluso una nueva representación gráfica (icono), a un elemento del UML estándar.

- ***Tagged Values*.** Cada elemento del UML posee su propio conjunto de propiedades, por ejemplo, las clases tienen un nombre, atributos y operaciones. Un *tagged value* puede entenderse como un meta-dato, que permite especificar una nueva categoría de propiedad que puede ser asignada a un elemento de UML.
- ***Constraints*.** Son utilizadas para introducir restricciones sobre uno o más elementos, refinando su semántica. Una restricción puede ser definida por medio de una explicación textual informal o utilizando OCL.

La definición de un conjunto coherente de extensiones a UML utilizando estos mecanismos, para un propósito y dominio específico, constituye un *profile*. Por ejemplo, la versión 1.4 del UML [16] incluye un *profile* para el modelado de procesos de desarrollo de software. Adicionalmente, es necesario destacar que es posible extender el meta-modelo de UML definiendo explícitamente nuevas meta-clases, siguiendo la técnica de especificación descripta previamente.

3. El Proyecto MAM-UML: Contexto, Objetivos y Actividades

El presente proyecto, descripto aquí en términos generales y denominado *Modelado de Agentes Móviles con UML* (MAM-UML), es parte componente de una de las líneas de investigación definidas en el Instituto de Sistemas de Tandil (ISISTAN-UNICEN). Dicha línea de investigación, definida genéricamente con el título de *Modelado de Software con UML*, especifica como su objetivo primario el de aportar mejoras al UML estándar respecto del desarrollo de software con nuevas tecnologías, en particular en las áreas de desarrollo de aplicaciones basadas en agentes de software (AOSE) y orientadas a aspectos (*Aspect-Oriented Programming* - AOP). Dado este objetivo primario se trazan, para dicha línea de investigación, los siguientes objetivos específicos:

- Definición de un procedimiento riguroso y sistemático para la introducción de nuevos conceptos al UML estándar.
- Enriquecimiento de los modelos y diagramas provistos por UML, o la incorporación de nuevos, para la introducción de nuevos conceptos de modelado en el desarrollo de software con nuevas tecnologías. Documentación formal de las extensiones a UML que se propongan.
- Construcción de una herramienta CASE, o exploración acerca de la factibilidad de extender una ya existente, la cual soporte el meta-modelo del UML y permita la extensión del UML estándar para el desarrollo de software con nuevas tecnologías.

El proyecto MAM-UML instancia tales objetivos de manera específica en el paradigma de agentes de software móviles y prescribe, en general, las siguientes actividades de investigación y desarrollo para su consecución:

- Relevamiento bibliográfico sobre el paradigma de agentes móviles, en general, con el objeto de establecer el estado del arte en el área e identificar las abstracciones relevantes para el modelado de los sistemas y aplicaciones basadas en agentes móviles. Adicionalmente, se hace necesario también, un relevamiento bibliográfico sobre los trabajos realizados respecto de la extensión del UML y que contribuyen al modelado de agentes móviles.
- Análisis de las carencias del UML estándar para el modelado de sistemas y aplicaciones basadas en agentes móviles. Analizándose también, las contribuciones existentes para mitigar estas carencias.
- Extensión de los modelos y diagramas que UML provee, o definición de nuevos modelos, de manera de representar las abstracciones identificadas como relevantes y de las cuales el UML estándar carece, en el contexto del paradigma de agentes móviles. Esta actividad presupone también, la documentación de las extensiones propuestas definiendo un *profile* que incluya los s*tereotypes, tagged values y constraints* correspondientes.
- Validación de la efectividad de las extensiones al UML propuestas, por medio del modelado de diferentes escenarios y aplicaciones. A priori, los dominios de aplicación seleccionados incluyen: sistemas de *e-commerce*, asistentes personales, y aplicaciones de *Socialware* [5] en general.

4. Una Propuesta de Modelado de Agentes Móviles utilizando UML extendido

En este contexto, y a partir de la revisión del estado del arte en el área de agentes móviles y extensiones a UML propuestas, se han identificado diferentes abstracciones consideradas relevantes en el diseño de sistemas y aplicaciones de agentes móviles [2b]. El énfasis de esta revisión se orientó principalmente a la captura de las características tanto funcionales como técnicas de los sistemas de agentes móviles, explorándose también los diferentes mecanismos alternativos a tener en cuenta para su diseño e implementación [4][10][17]. Las abstracciones identificadas se organizan en un conjunto coherente de vistas y modelos, que constituye una propuesta preliminar de *profile* UML, actualmente en evaluación y refinamiento por medio del modelado de diferentes escenarios y aplicaciones simples. El conjunto de vistas, y los modelos que incluyen, se resume a continuación de manera informal.

- **Vista Organizacional**. Describe las entidades involucradas en la aplicación y sus relaciones estructurales (asociación, composición, etc.). Se utiliza un diagrama de clases para especificar: los tipos de agentes (*sterotypes: mobile* o *stationary*, derivados del elemento *active class* en el meta-modelo del UML estándar); los roles que estos agentes pueden interpretar

(*sterotype: role*, derivado de *class*); los recursos que manipulan y comparten (*sterotype: resource*, derivado de *class*); los sistemas de agentes (o entornos computacionales, *sterotype: agent system*, derivado de *package*); las localizaciones (o lugares, *sterotype: place*, derivado de *package*) que definen dichos entornos para la ejecución de los agentes; y las federaciones (o regiones, *sterotype: region*, derivado de *package*) que conforman los entornos computacionales. Se definen también, diferentes *tagged values* para la descripción de propiedades de los *sterotypes* mencionados, como por ejemplo: *location* (localización actual de un agente móvil o recurso) y *authority* (personas u organizaciones a las cuales representa un agente móvil, y se asocian también a localizaciones, entornos o regiones que las aceptan como válidas en el contexto de una aplicación), entre otros.

- **Vista del Ciclo de Vida**. Describe el ciclo de vida de un agente móvil. Incluye dos modelos:

  Modelo Patrón de Ciclo de Vida. Especifica el patrón de estados que experimenta un agente móvil durante su ciclo de vida (activo, suspendido, desactivado, etc.) y los eventos o condiciones que provocan las transiciones entre estos estados. Actualmente, los patrones de ciclo de vida de agentes móviles más difundidos son: (i) el de proceso persistente, adoptado por Telescript [20] y sistemas (de agentes móviles) con movilidad fuerte similares; y (ii) el basado en tareas, adoptado por Aglets [13] y sistemas con movilidad débil similares. Se utiliza un diagrama de transición de estados (*statechart*) para especificar el patrón de ciclo de vida correspondiente. Adicionalmente, se definen *sterotypes* para especificar la acción de migración del agente móvil según el tipo de movilidad que utiliza (*sterotypes*: *strong move* y *weak move*, derivados de *action*). Por otra parte, si la movilidad es reactiva la acción de migración será provocada por un evento externo, en este sentido pueden utilizarse varios *statechart* que muestren la sincronización entre las entidades que deciden la migración y el agente que debe migrar en consecuencia.

  Modelo de Actividades. Describe las actividades realizadas por un agente móvil durante su ejecución (cuando se encuentra *activo*, un estado en el modelo anterior). Es factible detallar también: *dónde* (en que localización), *por qué* (objetivo o propósito que persigue) y *cuándo* (ocurrencia de un evento o situación) un agente móvil realiza una determinada actividad. Se utiliza un diagrama de actividades en los cuales diferentes *swimlanes* permiten particionar el diagrama en las diferentes localizaciones donde el agente ejecuta, y diferentes *constraints* asociadas a las actividades definen los "por qué" y los "cuándo".

- **Vista de Movilidad**. Incluye modelos para especificar la movilidad de código y estado de los agentes, así cómo también, los mecanismos para la administración de su espacio de datos.

  Modelo de Movilidad de Código y Estado. Describe hacia dónde migra un agente, por qué migra, cuándo lo hace y esencialmente qué (código, datos o estado de ejecución) se transfiere en la migración. Se utilizan diagramas de componentes y *deployment* especificando: los nodos donde residen los agentes móviles, entornos computacionales y lugares de ejecución involucrados (*sterotypes* correspondientes derivados de *component*). Los links entre nodos especifican el protocolo utilizado en la transferencia y cuplas navegando entre nodos representan agentes móviles migrando (*sterotype: mobility*, derivado de *dependency relationship*; con diferentes *tagged values* definidos para especificar: el agente móvil, el tipo de movilidad según qué se transfiere - fuerte o débil -, el tipo de movilidad según si es el propio agente quien decide migrar o no - pro-activa o reactiva –, y las localizaciones de origen y de destino). Diferentes *constraints* asociadas a la cuplas definen los "por qué" y los "cuándo". En el caso de movilidad de código también diferentes *constraints*, asociadas a la cuplas, definen los mecanismos utilizados según: la dirección de transferencia (*shipping* o *fetching*); la naturaleza del código a transferir (*previously installed*, *stand-alone* o *fragment*); y la sincronización involucrada (*synchronic* o *not synchronic*). Tales mecanismos representan patrones que pueden ser definidos en paquetes UML, por separado. Adicionalmente, pueden utilizarse diagramas de interacción para representar escenarios de navegación, junto con los *sterotypes, tagged values y constraints* mencionados.

  Modelo de Administración de Recursos. Luego de migrar un agente móvil a un entorno computacional remoto, su espacio de datos (es decir, el conjunto de referencias a recursos accesibles por el agente) debe ser reorganizado. Este modelo especifica los tipos de recursos y de ligaduras (*bindings*) que se dan entre agentes y recursos. Adicionalmente, se especifican los mecanismos que describen cómo estas ligaduras son administradas cuando un agente móvil migra hacia otra localización. Se utiliza un diagrama de clases para especificar a los agentes y recursos (por medio de *sterotypes*) y sus *bindings*. Para el *sterotype* recurso pueden utilizarse *tagged values* para especificar sus tipos (*transferible* o *no transferible*; *libre* o *fijo*). Para especificar un *binding* puede utilizarse un *sterotype* de la relación de dependencia, definiendo un *tagged value* para especificar el tipo del *binding* (por valor, por tipo de recurso, o por identificador). Los mecanismos alternativos de administración del espacio de datos - que involucran la re-localización de recursos y re-organización de ligaduras - ante la migración de un agente a otro entorno remoto, pueden especificarse (en un alto nivel de abstracción) en dicho diagrama de clases como colaboraciones genéricas (patrones) entre los entornos computacionales de origen y destino, el agente y el recurso involucrados. El patrón específico (*Binding removal*, *By move*, *Network reference*, *By copy* o *Re-binding*) que define el mecanismo utilizado, se especifica en un paquete parametrizable aparte, incluyendo diagramas estáticos y dinámicos. Adicionalmente, pueden utilizarse diagramas de objetos para tomar instantáneas de agentes y *bindings* justo antes de la migración (en el entorno de origen) y después de esta (en el entorno destino), de manera de ilustrar el resultado esperable del mecanismo de administración a aplicar.

- **Vista de Comunicación**. Describe el diálogo que se da entre agentes así como también entre agentes y entornos, especificando sus interacciones, protocolos y mecanismos de coordinación utilizados (desde las perspectivas temporal y espacial). Se adopta para esta vista los modelos de interacción definidos por la metodología *MESSAGE/UML* y la notación de *AUML* para diagramas de secuencia, ambos reportados en [6], por su alcance y amplia difusión. No obstante, esta explorándose el enriquecimiento de estos modelos a partir de la definición de elementos adicionales que permitan especificar con más detalle aspectos, tales como: sincronicidad, localidad, destinatario e intermediación.

5. Conclusiones y Trabajos Futuros

En este artículo fue presentado un proyecto en realización, el cual involucra esencialmente el aporte de mejoras al UML estándar, para su adecuación en el desarrollo de software en el contexto del paradigma de agentes móviles. En este sentido, fue introducido un conjunto de modelos a modo de propuesta preliminar.

Dichos modelos capturan las abstracciones identificadas, hasta el momento, como relevantes en el modelado de características funcionales y técnicas de los sistemas y aplicaciones de agentes móviles, durante la etapa de diseño de su desarrollo.

La efectividad del enfoque está siendo evaluada por medio del desarrollo de especificaciones de diferentes escenarios y aplicaciones simples. A pesar de que la propuesta se presenta apropiada en el contexto de estos casos de estudio, se ha identificado la necesidad de la creación de nuevos modelos y vistas complementarias, como por ejemplo: un modelo describiendo la estructura interna de un agente (su conocimiento, creencias, deseos, intenciones y objetivos); un modelo de seguridad, para el modelado de mecanismos que prevengan la vulnerabilidad tanto de los entornos computacionales como de la integridad y privacidad de los agentes; un modelo de comunicación adicional que describa las interacciones que se dan entre un usuario y el agente (móvil) que lo representa, especificando los protocolos que involucran estas interacciones; y un modelo computacional para la ejecución de los agentes. Por otra parte, a partir del trabajo experimental descripto se espera poder definir un proceso iterativo de refinamiento de los modelos propuestos. Surge entonces un nuevo objetivo: integrar el proceso de refinamiento que se defina, adaptando una metodología ya existente. En este sentido, el *Unified Process* (*UP*) [8] parece ser el más adecuado de explorar dado que ha sido diseñado para ser configurable y utiliza a UML como notación.

Referencias Bibliográficas

[1].- Baumeister H., Koch N., Kosiuczenko P. and Wirsing M.: Extending Activity Diagrams to Model Mobile Systems. In Proceedings of Net ObjectsDays 2002 Conference, Erfurt, Germany, October 2002.

[2].- Belloni E. A. and Campo M.: BrainLets: Dynamic Inferential Capabilities for Agent-based Web Systems. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No. 13 (2001), pp. 108-114. ISSN: 1137-3601. © AEPIA.

[2b].- Belloni E. A.: Sistemas y Aplicaciones de Agentes Móviles: Conceptos Básicos, Características Técnicas y Funcionales. Reporte Técnico. ISISTAN-UNICEN. Noviembre 2002.

[3].- FIPA. The Foundation of Intelligent Physical Agents. http://www.fipa.org

[4].- Fuggetta A., Picco G. P. and Vigna G.: Understanding Code Mobility. IEEE Transactions on Software Engineering, Vol. 24, No. 5, pp. 342-361, 1998.

[5].- Hattori F., Ohguro T., Yokoo M., Matsubara S. and Yoshida S.: Socialware. Multiagent Systems for Supporting Network Communities. Communications of the ACM. Vol. 42, No. 3, pp. 55-61. March 1999.

[6].- Huget M-P., Odell J. and Bauer B.: UML and Agents: Current Trends and Future Directions. Workshop on Agent-oriented methodologies. OOPSLA 2002, Seattle, USA, November 2002.

[7].- Iglesias C.A. and González J.C.: A Survey of Agent-Oriented Methodologies. In Proceedings of ATAL'98, LNAI No. 1555, pp. 317-330. Springer-Verlag, July 1998.

[8].- Jacobson I., Booch G. and Rumbaugh J.: The Unified Software Development Process. Addison-Wesley, 1999.

[9].- Jennings N. R.: On Agent-based Software Engineering. Artificial Intelligence, Vol. 117, No. 2, pp. 277-296, 2000

[10].- Karnik N. and Tripathi A. R.: Design Issues in Mobile Agent Programming Systems. IEEE Concurrency, Vol. 6, No. 3, July-September 1998.

[11].- Klein C., Rausch A., Sihling M. and Wen Z.: Extension of the Unified Modeling Language for mobile agents. In Siau K. and Halpin T. (Eds.): Unified Modeling Language. Systems Analysis, Design and Development Issues, chapter VIII. Idea Group Publishing, 2001.

[12].- Kosiuczenko P.: Sequence diagrams for mobility. In Spaccapietra S. (Ed.): 21 International Conference on Conceptual Modeling (ER2002). Springer-Verlag, October 2002.

[13].- Lange D. and Oshima M.: Programming and Deploying Java Mobile Agents with Aglets. Addisson-Wesley Longman, Reading Mass. 1998.

[14].- Mouratidis H., Odell J. and Manson G.: Extending the Unified Modeling Language to Model Mobile Agents. Workshop on Agent-oriented methodologies. OOPSLA 2002, Seattle, USA, November 2002.

[15].- Muscutariu F. and Gervais M-P.: On the modeling of mobile agent-based systems. In 3rd International Workshop on Mobile Agents for Telecommunication Applications (MATA'01), LNCS Vol. 2164, pp. 219-234. Springer-Verlag, August 2001.

[16].- OMG. The Object Management Group: Specification of the Unified Modeling Language (UML), version 1.4. September 2001. http://www.omg.org

[17].- Rodrigues Silva A., Romão A., Deugo D. and Da Silva M. M.: Towards a Reference Model for Surveying Mobile Agent Systems. In Automous Agents and Multi-Agent Systems, No. 4, pp. 187-231. Kluwer Academic Publishers, 2001.

[18].- Rumbaugh J., Jacobson I. and Booch G.: The Unified Modeling Language. Reference Manual. Addison-Wesley, 1999.

[19].- Warner J. and Klepper A.: The Object Constraint Language. Precise Modeling with UML. Addison-Wesley, 1998

[20].- White J.: Mobile Agents. In Software Agents, J. M. Bradshaw (Ed.), pp. 437-472. AAAI/MIT Press, 1997.

[21].- Wooldridge M. and Ciancarini P.: Agent-Oriented Software Engineering: The State of the Art. In Handbook of Software Engineering and Knowledge Engineering. World Scientific Publishing, 2001.

[22].- Zambonelli F., Jennings N. R., Omicini A. and Wooldridge M.: Agent-Oriented Engineering for Internet Applications. In Coordination of Internet Agents: Models, Technologies and Applications, chapter XIII. Springer-Verlag, 2000.