Simulación Social Basada en Agentes

Candelaria Sansores, Juan Pavón

Dep. Sistemas Informáticos y Programación
Universidad Complutense Madrid
Madrid, 28040
csansores@fdi.ucm.es, jpavon@sip.ucm.es

Resumen

Una de las aplicaciones de la tecnología de agentes es la simulación de fenómenos sociales. Con este propósito se han desarrollado varias herramientas de simulación basada en agentes. Sin embargo, aunque el paradigma de agente debería facilitar, por sus características intencionales y sociales, la realización de modelos para simulación social, la utilización de las herramientas actualmente disponibles requiere un gran conocimiento práctico de programación, por lo cual son poco apropiadas para los sociólogos. Ello hace necesario proporcionar nuevas herramientas de modelado, más fácilmente adaptables a las necesidades de los sociólogos, con conocimientos más básicos de programación. Asimismo, otro requisito que surge habitualmente en este tipo de sistemas es la escalabilidad, tanto en lo que se refiere a la edición de los modelos de simulación como a su ejecución. Para abordar estos problemas se propone desarrollar varios entornos gráficos de simulación, orientados a dominios de aplicación específicos. Para su realización se plantea aplicar un entorno de desarrollo de sistemas multi-agente abierto y flexible, INGENIAS, que proporciona conceptos, métodos y herramientas para el modelado de este tipo de sistemas y la generación de código sobre múltiples plataformas.

Palabras clave: Modelado basado en agentes, Simulación Social, Agentes, Sistemas Multi-Agente, INGENIAS.

1. Introducción

Una herramienta habitual en las ciencias sociales consiste en representar los sistemas sociales como sociedades de agentes, ejecutar estos modelos y analizar su comportamiento. En este contexto, los agentes son sistemas de software autónomos orientados a describir el comportamiento de las entidades sociales estudiadas (tanto individuos como organizaciones). Una ventaja de esta técnica, conocida como Simulación Social Basada en Agentes (o *Agent Based Social Simulation*, ABSS), es la capacidad para estimar la plausibilidad del comportamiento de los agentes, la forma en que interactúan, y los efectos de ese comportamiento e interacción.

En [Axtell2000] se plantean varios motivos por los cuales emplear la computación basada en agentes en las ciencias sociales, y se argumenta que dependiendo de la complejidad del modelo social, existen tres usos distintos de la simulación basada en agentes: modelos con agentes como simulaciones clásicas (cuando los modelos pueden formularse y resolverse completamente), agentes artificiales como complemento a la formulación de teorías matemáticas (para modelos parcialmente resolubles), y computación con agentes como herramienta de análisis (cuando los modelos son intratables o probablemente sin solución). En este trabajo se tratan los dos últimos casos, por ser los que más interés tienen desde la perspectiva de agentes, ya que se trata de estudiar el comportamiento emergente de las sociedades de agentes, permitiendo la gestión de los modelos en

varios niveles, tanto individual (los agentes) como colectivo (la sociedad u organizaciones de agentes). Así, en el segundo caso, cuando los modelos matemáticos puedan formularse pero no resolverse completamente, los modelos basados en agentes pueden esclarecer significativamente la estructura de la solución, ilustrar las propiedades dinámicas del modelo, servir para probar la dependencia de los resultados de parámetros y suposiciones, y ser la fuente de numerosos ejemplos. El tercer caso surge porque existen clases de problemas importantes para los cuales, aunque es posible describir un modelo, hay algunos parámetros y configuraciones que no son estables o pueden cambiar de muchas maneras, lo cual hace extremadamente difícil el análisis. En estos casos, la simulación basada en agentes puede proporcionar más información sobre el comportamiento del sistema.

En la ABSS el modelo es un sistema multi-agente (SMA), un conjunto de agentes autónomos que operan en paralelo y que se comunican unos con otros. Las propiedades individuales de los agentes que describen su comportamiento e interacciones son conocidas como *propiedades elementales*, y las propiedades que emergen en un nivel colectivo más alto son conocidas como *propiedades emergentes*. La *emergencia* es un aspecto importante para el estudio de sistemas sociales complejos, pues se dice que el comportamiento global emerge de los agentes y sus interacciones cuando se ejecuta la simulación, los macro patrones y los procesos que emergen son entonces comparados con los patrones de la sociedad que, por otro lado, se ha observado empíricamente. Así, se puede argumentar que los SMA permiten la exploración de la relación micro-a-macro[1] [Coleman1990]. Es importante notar la sutil diferencia entre los SMA y la ABSS apuntada por [Conte1998]: "sí el campo de los SMA puede caracterizarse como el estudio de [o implementación de] sociedades de agentes autónomos artificiales, la simulación social basada en agentes puede definirse como el estudio de sociedades artificiales de agentes autónomos".

Junto con la aceptación y el interés creciente en la ABSS como herramienta de simulación, van apareciendo varias librerías para el desarrollo de sistemas de simulación basada en agentes. REPAST [RePast2004] es un conjunto de librerías Java que permiten a los desarrolladores crear entornos de simulación, crear agentes en redes sociales, recopilar datos de las simulaciones automáticamente, y construir interfaces de usuarios fácilmente. Sus características y diseño son similares a SWARM [Swarm2004], una de las primeras librerías para el modelado basado en agentes. Otra librería similar es ASCAPE [Ascape2000].

Estas librerías tienen grandes ventajas para los modeladores, aunque también tienen algunas limitaciones, ya que requieren que los modeladores cuenten con un gran conocimiento práctico de los lenguajes de programación hacia los cuales están orientadas, Java en la mayoría de los casos. Pero generalmente, los modeladores son sociólogos o economistas con conocimiento de programación básico, así que se debería apoyar su actividad con herramientas para especificar modelos de simulación en un lenguaje declarativo de alto nivel en lugar de hacerlo con lenguajes de programación de bajo nivel.

Esta idea aparece en SDML [SDML1997], un leguaje de modelado basado en SmallTalk. A diferencia de ASCAPE y REPAST, no requiere que los usuarios sean expertos en el lenguaje de programación subyacente, pero requiere que los usuarios aprendan a usar una interfaz compleja que puede ser tan difícil de dominar como todo un lenguaje de programación.

Otro intento de facilitar el modelado de este tipo de sistemas es SeSAm (*Shell for Simulated Agent Systems*) [Kluegl et al.2004], que proporciona un editor de comportamientos basado en máquinas de estado finito, similar a los diagramas de estado de UML. Sin embargo, al final hay que conocer un conjunto amplio de primitivas del entorno y el manejo de autómatas no es evidente, en general, para quienes no tengan una formación informática.

Otro requisito importante para la ABSS es la escalabilidad, tanto en la definición como en la ejecución de los modelos. Comparado con un sistema de simulación tradicional, generalmente basados en modelos matemáticos o estadísticos, un sistema de simulación basado en SMA con frecuencia requiere de una gran cantidad de recursos computacionales para su ejecución, ya que cada agente suele corresponderse con un hilo de ejecución. Al experimentar con las distintas plataformas existentes se puede comprobar que generalmente los modelos que soportan siguen

[1] *Las micro simulaciones son simulaciones basadas en propiedades de unidades de bajo nivel como por ejemplo individuos, a diferencia de las macro simulaciones (como la variedad de la dinámica de sistemas), la cual intenta modelar directamente macro fenómenos emergentes.*

estando limitados a un número reducido de agentes. Y en sistemas sociales el tamaño del modelo suele ser importante, ya que la variación en el número de agentes que interactúan puede afectar sensiblemente los principales resultados de la simulación. Empíricamente [Cioffi-Revilla2002] se ha mostrado que el tamaño es dependiente del tiempo en numerosos sistemas sociales (por ejemplo, centros urbanos) y que el grupo o tamaño del sistema realmente es importante para sistemas y procesos que involucran una acción colectiva y, similarmente, fenómenos sociales significativos. Por lo tanto, no hay fundamentos teóricos para afirmar qué un modelo con el mismo número de agentes que individuos observados en un sistema social de gran tamaño se comportará de la misma forma que un modelo con unos pocos agentes.

Por esta razón, en el caso de este tipo de sistemas se hace necesario disponer de medios para escalar los modelos y no limitar el tamaño de la simulación. Más aún, los agentes que comprenden estos modelos deben ser capaces de usar los recursos computacionales distribuidos, primero, porque los agentes con frecuencia requieren una gran cantidad de recursos de computación, y segundo, por que el conjunto de datos requerido por la simulación podría estar distribuido geográficamente. De esta manera, una simulación distribuida puede ser deseable, mejorando la validación[2] del modelo. En este sentido, la computación en Grid [Foster y Kesselman1998] juega un papel importante para que esto suceda, proporcionando la tecnología para acceder a una gran cantidad de recursos distribuidos, y suministrando los medios para construir y ejecutar sistemas complejos, como las aplicaciones de simulación social distribuida.

La sección 2 presenta una propuesta de ABSS que cubre los requisitos de escalabilidad y modelado en un lenguaje de alto nivel. Esta propuesta está basada, por una parte, en la utilización de los métodos y herramientas de INGENIAS para el modelado y desarrollo de SMA, mediante su adaptación a dominios concretos de estudio social, y por otra en la definición de un conjunto de componentes de soporte para una infraestructura de computación en Grid. La sección 3 plantea la adopción de una solución de computación Grid para la ejecución de sistemas de agentes a gran escala. La sección 4 describe las actividades en este entorno que el modelador debe realizar para desplegar un modelo de sistema social en la infraestructura de la computación en Grid. Finalmente, en la sección 5 se discuten los beneficios de este planteamiento y se describe el estado actual de este trabajo.

[2] *La validación es uno de los pasos de un framework de investigación genérico para los estudios de simulación [Gilbert y Troitzsch1996], la cual valida si la simulación es un buen modelo del objetivo estudiado. Un modelo que refleja el comportamiento del objetivo es considerado válido.*

2. Un entorno para la ABSS

El objetivo de esta propuesta es proporcionar un entorno que permita el modelado, implementación, y despliegue de sistemas ABSS en un entorno de computación en grid, y que libere a los modeladores, en lo posible, de la tarea de programación. En la Fig. 1 se muestran los elementos principales del entorno: un editor de modelos que tiene que estar adaptado a un dominio concreto de estudio social, generadores de código para obtener un sistema de agentes que pueda ejecutarse en la plataforma de simulación, que podría realizarse sobre una infraestructura de computación en grid.

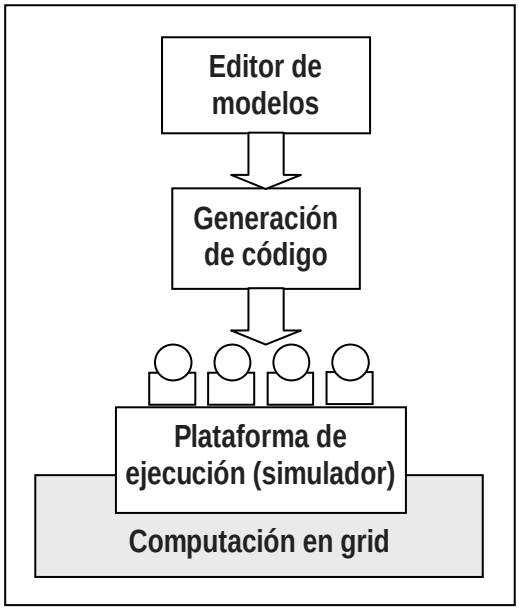

Figura 1. Elementos del entorno de ABSS

Las herramientas de modelado y generación de código están basadas en el INGENIAS Development Kit [IDK2004]. El modelador (por ejemplo, un sociólogo) define el modelo con el editor gráfico de INGENIAS previamente personalizado para el dominio particular donde se

vaya a trabajar. A continuación genera el código de los agentes para la plataforma de simulación, y ejecutará la simulación para obtener los resultados.

Considerando la diversidad de dominios de problemas sociales existentes, pensamos que en lugar de proporcionar una herramienta demasiado general para la implementación de sistemas ABSS, una manera de simplificar el desarrollo de modelos es proporcionar un entorno fácilmente adaptable a diferentes fenómenos sociales de diferentes dominios. Esto implica que podríamos tener editores de modelos específicos para cada dominio y diferentes correspondencias a plataformas de ejecución. El IDK está construido en base a la especificación de meta-modelos y proporciona APIs para gestionarlos y poder generar los editores, herramientas de verificación, y módulos de generación de código. Por lo tanto, solamente es necesario especificar meta-modelos para cada dominio social y las correspondencias a la infraestructura de ejecución. La tarea de especificación y generación de los editores específicos del dominio y generadores de código no es responsabilidad del modelador, sino del desarrollador de la plataforma ABSS, quién debe conocer los meta-modelos de INGENIAS y contar con experiencia en programación Java para personalizar los editores y las herramientas de generación de código (soportada por un *plugin* de Eclipse para el IDK). Esta tarea se realiza una sola vez para un dominio de problema específico, y varios modeladores pueden beneficiarse de estas herramientas.

El IDK (ver Fig. 2) está basado en los principios de la metodología INGENIAS, la cual considera la definición de un conjunto de meta-modelos que describen los elementos que forman un SMA desde varios puntos de vista, y que permiten establecer un lenguaje de especificación de SMA. A éstos se añaden para simulación dos nuevos meta-modelos:

- El meta-modelo *Timer*, para especificar pasos regulares de tiempo y permitir simular el ciclo de percepción-reacción de los agentes que actúan por el paso del tiempo. Este meta-modelo gestiona el paso del tiempo en el modelo durante la simulación.
- El meta-modelo *Space*, que describe la distribución espacial de los agentes en el modelo, esto es, las relaciones espaciales simples como 2D, 3D, Grids hexagonales, torus hexagonales, etc.

La especificación de un SMA en INGENIAS consiste en la definición, control, y gestión del estado mental de cada agente, las interacciones de los agentes, la organización del SMA, el entorno, y las tareas y objetivos asignadas a cada agente. El modelado del SMA es facilitado por el editor gráfico y las herramientas de verificación. Como complemento a estas herramientas hay un proceso genérico para parametrizar e instanciar un SMA a una plataforma específica con el componente generador de código. Otro componente esencial del entorno es el habilitador de los agentes para el Grid (*Agent-Grid Enabler*) una extensión al generador de código de INGENIAS que actúa como middleware para la infraestructura GT3.

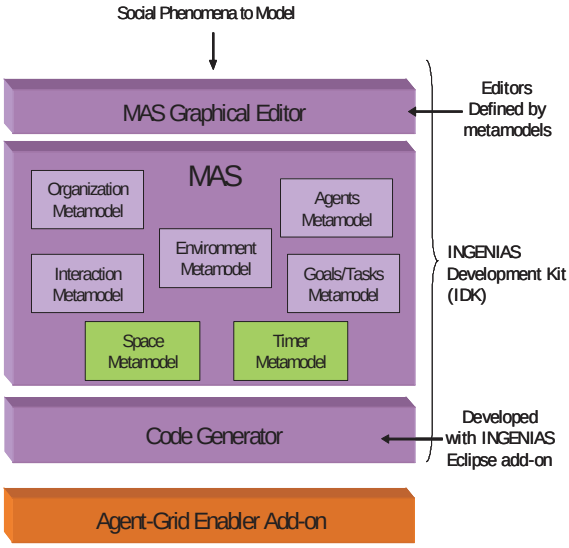

Figura 2. Componentes para el Modelado

3. ABSS en un entorno de computación en grid

Los grid computacionales son entornos distribuidos mundialmente que se enfocan en la infraestructura, herramientas, y aplicaciones para la compartición confiable y segura de recursos a gran escala. El Grid proporciona a los SMA un entorno colaborativo distribuido escalable, robusto y seguro. Ambas tecnologías parecen beneficiarse una de la otra, sin embargo, este trabajo se centra en explotar los beneficios que la tecnología Grid puede proporcionar a la ejecución de SMA a gran escala. Esto requiere la definición de una arquitectura para la ejecución de SMA en un entorno Grid, lo cual

requiere diseñar e implementar nuevos mecanismos para la distribución segura de los agentes, descubrimiento de servicios, gestión de recursos, y balanceo de carga en el entorno Grid.

El Grid posibilita la creación de entornos colaborativos de resolución de problemas, a través de los cuales los científicos o usuarios finales pueden ejecutar aplicaciones complejas que usualmente requieren colaboración e interacción de una gran cantidad de recursos heterogéneos distribuidos [Foster et al.2001]. La computación en Grid tiene que ver con la compartición coordinada de recursos no sujetos a un control centralizado y con la resolución de problemas en organizaciones virtuales, dinámicas, multi-institucionales. Siendo una organización virtual un conjunto de individuos e instituciones definidos por reglas de compartición. Por lo tanto, la interoperabilidad en el Grid requiere de protocolos, interfaces, y políticas que sean no solamente abiertos y de propósito general sino también estándar. Los estándares permiten establecer acuerdos de compartición de recursos dinámicamente con cualquier participante interesado y así crear algo más que una plétora de sistemas distribuidos incompatibles o no interoperables.

Por otro lado, la definición de estos protocolos estándares es el problema más crítico al que se enfrenta la comunidad Grid hoy en día. Una de las propuestas más aceptadas es Globus Toolkit [Globus2001], un estándar *de facto* ampliamente utilizado, el cual está formado por un conjunto de servicios y librerías que permiten programar *Servicios Grid*, que facilitan la creación de infraestructura y aplicaciones Grid. Los servicios Grid que proporciona son: seguridad, gestión de recursos, información de acceso y gestión de datos.

Un servicio Grid es un servicio Web que sigue un conjunto de convenciones que le permiten una gestión controlada, resistente a fallos y segura, de servicios que mantienen estado. La especificación OGSA (Open Grid Service Architecture) [Foster et al.2002] pretende establecer la arquitectura común para aplicaciones basadas en el Grid, refinando el modelo de los servicios Web. OGSA define qué son los servicios Grid, de qué deben de ser capaces, y en qué tipos de tecnologías deben de basarse, aunque no proporciona una especificación técnica detallada. Por otro lado, OGSI (Open Grid Service Infrastructure) [Tuecke et al.2003] proporciona la especificación formal y técnica de los conceptos descritos en OGSA, incluyendo los servicios Grid. El GT3 está construido con la especificación OGSA y es una implementación de OGSI de código abierto.
Los componentes más relevantes del GT3 para este trabajo son: (i) la infraestructura central (Core Services) [Sandholm y Gawor2003], que proporciona entornos de alojamiento de servicios, infraestructura de seguridad (GSI), así como servicios a nivel de sistema como bitácora y gestión, (ii) índice de servicios, que proporciona información acerca de recursos disponibles y su estado dentro del Grid, y (iii) el GRAM (Globus Resource Allocation Manager), que simplifica el uso de sistemas remotos proporcionando una sola interfaz estándar para solicitar y usar recursos de sistemas remotos, éste componente es usado comúnmente para soportar aplicaciones de computación distribuida.

Una vez que el desarrollador haya modelado y generado el código de la simulación, el siguiente paso será desplegar el sistema como se muestra en la fig. 3., para esto será necesario utilizar los componentes proporcionados por el soporte ABSS que forma parte del entorno propuesto.

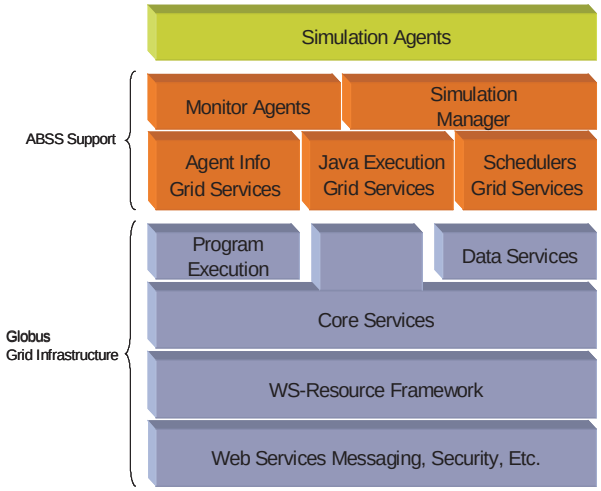

Figura 3. Infraestructura Grid para la ABSS

El soporte consta de cinco componentes, los cuales están clasificados en servicios de alto nivel (*gestor*, *monitor*) y servicios de bajo nivel (*planificador*, *ejecución Java*, *información de agentes*). El principal objetivo de estos componentes es la distribución, localización y gestión de agentes que pertenecen a un modelo de simulación en particular, para su ejecución distribuida en el Grid. Para lograr su objetivo los componentes ofrecerán (1) fácil y cómodo acceso al Grid para la ejecución de agentes, (2) mecanismos de seguridad, (3) facilidad de uso, ocultando la complejidad del entorno Grid a los

desarrolladores y mejorando la accesibilidad de los recursos a los usuarios finales, y (4) gestión y control del proceso de simulación.
El *gestor de la simulación* (*simulation manager*) permite la carga de agentes en el entorno Grid, despacha y delega la gestión de los agentes a los *planificadores* (*schedulers*), que en su momento los envían a los servicios de ejecución Java responsables en realidad de la ejecución. Los agentes son implementados en Java por su portabilidad en entornos distribuidos heterogéneos y por su actual optimización que ha mejorado el desempeño de las aplicaciones [Ghahramani y Pauley2003]. Los planificadores buscan recursos disponibles y adecuados requeridos por los agentes que serán ejecutados en nodos remotos, y gestionan la ejecución. Normalmente, la infraestructura Grid proporciona un servicio de registro que contiene información indexada de servicios y recursos disponibles y que se puede localizar a través de un método de descubrimiento de servicio, por lo tanto hacemos uso intensivo de esta característica para implementar los planificadores.

El servicio de *ejecución Java* (*Java Execution*) es un servicio de computación que ejecuta un agente inicializando para esto una máquina virtual Java (JVM) y proporciona interfaces para un monitoreo básico del estado de ejecución del agente. El servicio de *información del agente* (*Agent Information*) es responsable de la localización del agente, actúa como un registro de agentes que pertenecen a un proceso de simulación particular. Así, una vez que los agentes son inicializados, éstos se registran con el servicio de información de agentes para notificar y recibir notificaciones de otras localizaciones de otros agentes. Por último, los *monitores* (*monitors*) de la simulación tienen dos propósitos, el primero es presentar información del estado de ejecución de los agentes a los usuarios finales, el segundo es actuar como observadores que se comunican con los agentes de la simulación y con el entorno para recopilar datos y presentar resultados intermedios acerca del proceso simulado.

Como los SMA no tienen un control centralizado [Wooldridge y Jennings1995] el comportamiento global del sistema emerge de los agentes y sus interacciones basadas en sus propias acciones autónomas. Por lo tanto, no proporcionamos un proceso central que recoja información de cada agente y decida qué acción debe tomar cada uno, pues esto iría en contra de algunos fundamentos de los SMA. Un argumento similar ha sido establecido también por [Davidsson2001]. Así, nuestra propuesta asume que las simulaciones que se llevaran a cabo son simulaciones distribuidas dirigidas por el tiempo.

4. Despliegue del sistema ABSS

Dada la estructura del entorno, el despliegue del sistema ABSS en el Grid puede resumirse en los siguientes pasos:

1. El modelador desarrolla un modelo de un sistema social con el editor gráfico del IDK adaptado al dominio de simulación particular y genera el código de los agentes para su ejecución en el entorno Grid.
2. El usuario se registra con el gestor de la simulación. El gestor autentifica al usuario y carga las configuraciones personales y credenciales como por ejemplo el certificado del usuario para obtener acceso a los servicios y recursos en el Grid.
3. El usuario configura los parámetros específicos de cada proceso de simulación a través del gestor de la simulación, el cual proporciona una interfaz gráfica para la carga y gestión de los agentes de la simulación.
4. Distribuir a los agentes en los nodos del Grid con el gestor, el cual invoca a los planificadores para servir a los agentes. El planificador busca los recursos disponibles en el índice de servicios y despacha a los agentes a los servicios de ejecución.
5. El usuario inicia el proceso de simulación y recupera los datos simulados para su procesamiento y evaluación con la ayuda de los monitores.

La seguridad del sistema ABSS se garantizará en todos los nodos del Grid a través de la delegación de certificados proxy que permiten implementar un mecanismo de registro único. El entorno proporciona todas las interfaces necesarias para interactuar en el entorno protegido, el cual soporta autenticación, autorización, y protección de mensajes a través de protocolos estándar como SSL/TLS y certificados estándar X.509.

5. Conclusiones

La utilización de agentes en simulación de fenómenos sociales puede ser más útil si se

considera que los usuarios finales, esto es, los sociólogos que definen modelos de simulación, no necesiten tener conocimientos específicos de programación. Para lograr este propósito es necesario que los entornos de definición y ejecución de modelos sean lo más cercanos posible a los conceptos que se utilizan en el dominio de aplicación concreto. Como los conceptos utilizados habitualmente en el desarrollo de sistemas multi-agente son próximos, nuestra propuesta es consideran un entorno de estas características, en concreto el IDK, y adaptarlo para el modelado y simulación basada en agentes de sistemas sociales complejos.

Además, para solucionar el problema de la escalabilidad, la simulación es soportada por herramientas que generan código para los agentes y los despliegan en un entorno de computación en grid. Los modeladores son apoyados en el detalle de implementación por un proceso automático. Los servicios añadidos de bajo nivel soportan la ejecución de la simulación en el entorno Grid, y los servicios de alto nivel facilitan el seguimiento y control de esas simulaciones y la recuperación de resultados.

En esta propuesta se consideran varios aspectos no considerados en conjunto por las herramientas de simulación basada en agentes actuales, que están más enfocadas a la tarea de programación y no toman en consideración la tarea del modelado conceptual. Nuestro propósito es precisamente facilitar el proceso de investigación social, desde el modelado conceptual hasta el análisis de la simulación, para lo cual aprovechamos los principios y herramientas asociados al proceso de desarrollo de INGENIAS.

Consideramos fundamental establecer un lenguaje y procedimiento común para describir modelos basados en agentes aplicados a las ciencias sociales, independientemente de la plataforma de implementación, principalmente porque es necesario en la comunidad científica como un medio de comunicación. También, pero no menos importante, hemos identificado la necesidad en muchos usuarios de poder replicar modelos en diferentes plataformas. La replicación contribuye a la fiabilidad de los resultados del modelo y a una mejor comprensión del sistema. Como [Axelrod1997] afirma, la replicación es una de las características distintivas de la ciencia acumulativa. Es necesaria para confirmar si los resultados obtenidos de una simulación dada son confiables en el sentido de que puedan ser reproducidos por terceros. Sin esta confirmación, es posible que algunos resultados publicados sean erróneos debido simplemente a errores de programación, a una mala representación de lo que realmente fue simulado, o errores en el análisis o reporte de los resultados.

Finalmente, para lograr el rigor metodológico necesario demandado por cualquier trabajo científico tenemos que satisfacer los 3 objetivos principales de Axelrod en el desarrollo y programación de un modelo social basado en agentes: *validación interna*[3], *utilidad* (debe ser fácil ejecutar el modelo y analizar los resultados) y *extensibilidad* (debe ser posible adaptar y re-utilizar el modelo). Así, nuestro planteamiento esta comprometido en soportar estas cuestiones proporcionando herramientas adaptables que realizan la correcta transformación de los modelos conceptuales a código de programación.

Agradecimientos

Este trabajo ha sido desarrollado con el apoyo del Consejo Nacional de Ciencia y tecnología (CONACYT) de México y el proyecto TIC2002-04516-C03-03, financiado por el Ministerio de Ciencia y Tecnología (MCYT) de España.

Referencias

[Ascape2000] ASCAPE. http://www.brook.edu/es/dynamics/models/ascape (2000).

[Axelrod1997] Axelrod. "Advancing the Art of Simulation in the Social Sciences". Simulating Social Phenomena, Berling Springer. (1997).

[Axtell2000] Axtell. "Why Agents? On the Varied Motivations for Agent Computing in the Social Sciences." Working Paper No. 17. (2000).

[Cioffi-Revilla2002] Cioffi-Revilla. "Invariance and universality in social agent-based

[3] *La validación interna, también conocida como verificación (Gilbert y Troitzsch 1996) en el campo de la simulación, se enfoca a la exactitud de la transformación de la representación abstracta (el modelo conceptual) al código del programa (el modelo de simulación). Esto es, que el código del programa refleje fielmente el comportamiento que está implícito en la especificación del modelo conceptual (Axelrod 1997).*

simulations" Proc Natl Acad Sci U S A, 99 Suppl 3, pp. 7314-6. (2002).

[Coleman1990] Coleman. "Foundations of Social Theory". Cambridge, MA, Harvard University Press. (1990).

[Conte1998] Conte. "MAS and Social Simulation: A Suitable Commitment". Multi-Agent-Based Simulation, First International Workshop, MABS 1998Paris, France. (1998).

[Davidsson2001]Davidsson. "Applications - Multi Agent Based Simulation: Beyond Social Simulation" Lecture Notes in Computer Science, 01979, pp. 97-108. (2001).

[Foster, et al.1998] Foster, et al. "The Grid: Blueprint for a new Computing Infrastructure". San Francisco, Morgan Kaufmann Publishers. (1998).

[Foster, et al.2002] Foster, et al. "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration". Globus Project. (2002).

[Foster, et al.2001] Foster, et al. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations" International Journal of High Performance Computing Applications, 15, pp. (2001).

[Ghahramani, et al.2003] Ghahramani, et al. "Java in High Performance Environments", 36, pp. 111. (2003).

[Gilbert, et al.1996] Gilbert, et al. "Simulation for the Social Scientist". Buckingham, U.K., Open University Press. (1996).

[Globus2001] Globus. "The Globus Alliance". http://www.globus.org (2001).

[IDK2004] IDK. "INGENIAS Development Kit". http://grasia.fdi.ucm.es/ingenias (2004).

[Kluegl, et al.2004]Kluegl, et al. "From Simulated to Real Environments: How to Use SeSAm for Software Development". Lecture Notes in Computer Science, 2831. (2004).

[RePast2004] RePast. "RePast". http://repast.sourceforge.net (2004).

[Sandholm, et al.2003] Sandholm, et al. "Globus Toolkit 3 Core - A Grid Service Container Framework". (2003).

[SDML1997] SDML. "SDML: a Strictly Declarative Modelling Language". http://www.cpm.mmu.ac.uk/sdml (1997).

[Swarm2004] Swarm. "Swarm Wiki, the agent-based modelling resource". http://wiki.swarm.org (2004).

[Tuecke, et al.2003] Tuecke, et al. "Open Grid Service Infrastructure (OGSI) Version 1.0". Global Grid Forum. (2003).

[Wooldridge, et al.1995]Wooldridge, et al. "Intelligent Agents: Theory and Practice" Knowledge Engineering Review, 10, pp. 115--152. (1995).