La replicación en la simulación social basada en agentes. El caso de SDML y RePast

Replication in multiagent based social simulation. The case of SDML and RePast

BRECHAS

El objetivo del presente artículo es justificar la necesidad de la replicación como metodología para alcanzar confiabilidad y rigor en la simulación social basada en agentes (ABSS). En él se comparan las características de dos conocidas plataformas de simulación, paradigmas del modelado declarativo e imperativo, SDML y RePast. Finalmente se presenta un marco de descripción de modelos para su replicación. Este marco está basado en la experiencia adquirida modelando y replicando modelos sociales aplicados a la gestión de recursos hídricos y a las finanzas, con diferentes plataformas de simulación como Swarm, RePast o SDML.

The objective in this paper is to justify the necessity of the replication as a methodology to reach trustworthiness and rigor in agent based social simulation (ABSS). Here the characteristics of two well-known platforms of simulation are compared; paradigms of declarative and imperative modeled one, SDML and RePast. Finally a model description framework is presented for its replication. This framework is based on acquired experience, modeling and retorting social models, applied to the management of hydric resources and to finances with different simulation platforms like Swarm, RePast or SDML.

José Manuel Galán *
Adolfo López Paredes * *
Cesáreo Hernández Iglesias
Javier Pajares Gutiérrez * * *

La replicación en la simulación social basada en agentes. El caso de SDML y RePast

¿Qué grado de confiabilidad nos dan los modelos de simulación social basada en agentes (*Agent based social simulation*, ABSS)? ¿Se puede garantizar la consistencia interna de un modelo? ¿Qué lugar ocupa la replicación en este proceso? ¿Qué problemas podemos encontrarnos al replicar? ¿Es posible replicar modelos en diferentes plataformas y en diferentes paradigmas de modelado? ¿Tiene alguna ventaja? ¿Es necesario un método estandarizado para describir modelos y experimentos?

El presente artículo tiene como objeto responder a las preguntas formuladas en el párrafo anterior y a otras preguntas derivadas. En las secciones siguientes discutiremos por qué la replicación es tan importante en ABSS: es la principal metodología para garantizar el rigor científico y demostrar la consistencia interna de los modelos.

El núcleo del trabajo es el marco de descripción que se presenta bajo el subtítulo "Hacia un marco estandarizado para la replicación". Este marco está basado en las lecciones aprendidas replicando modelos con diferentes plataformas y lenguajes de simulación: Swarm, RePast y SDML.

* Universidad de Burgos. Correo electrónico: jmgalan@ubu.es

** Universidad de Valladolid. Correo electrónico: adlo@eis.uva.es

*** Los autores pertenecen al grupo de investigación INSISOC (Ingeniería de los Sistemas Sociales) con sede en la Escuela Técnica Superior de Ingenieros Industriales de Valladolid (ETSII). http://www.insisoc.org

En particular, en este artículo presentamos una perspectiva y comparación general entre RePast y SDML, debido a su representatividad como paradigmas del modelado imperativo y declarativo.

Un resultado esperado de este trabajo es la contribución de otros colegas fruto de su experiencia en la comparación de los resultados de sus modelos con distintas plataformas de simulación.

■ Replicación de modelos en ABSS

La construcción y la simulación computacional de modelos basados en agentes constituyen el fundamento de la simulación social basada en agentes. Edmonds (2000) propone seis pasos en el desarrollo de un modelo basado en agentes (véase la figura 1):

1. Observar un determinado fenómeno social y abstraerlo en un modelo conceptual con los aspectos más relevantes, expresado en lenguaje natural.
2. Programar un Sistema Multiagente (MAS, *Multi-Agent System*) que capture el modelo propuesto.[1]
3. Simular y depurar el sistema multiagente.
4. Analizar los resultados mediante técnicas estadísticas normalmente.
5. Interpretar los resultados.
6. Aplicar los conocimientos aprendidos al fenómeno social objeto de estudio.

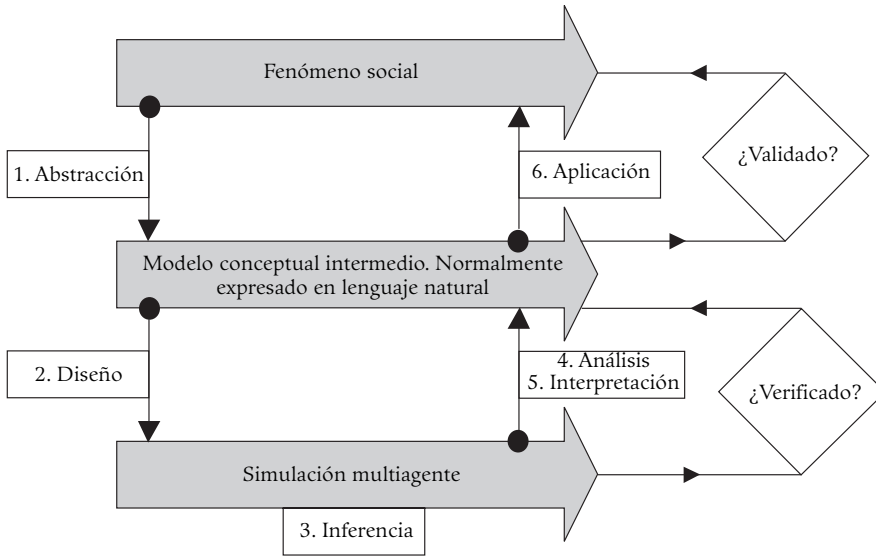
El modelado basado en agentes permite una representación computacional más realista e intuitiva de los fenómenos sociales que los métodos tradicionales (López-Paredes, 2003). Alcanzar el necesario rigor metodológico exigible a todo trabajo científico implica satisfacer tres objetivos en el desarrollo y programación de un modelo basado en agentes (Axelrod, 1997):

[1] No se debe fusionar con el paso anterior, pues ello provoca un problema de identificación entre modelo y programa informático.

- Verificación.[2] El programa debe implementar correctamente el modelo. Este es un objetivo evidente, pero no siempre fácil de demostrar.
- Funcionalidad. Facilidad para ejecutar el programa, interpretar los resultados y entender cómo funciona.
- Extensibilidad. Posibilidad de que usuarios futuros adapten el programa.

Numerosos autores proponen adoptar un lenguaje común para la simulación basada en agentes. Trabajos como los de Marietto *et al*. (2002) y Edmonds (2002) pretenden identificar las características para una plataforma ideal común.

FIGURA 1 DESARROLLO DE UN PROCESO DE SIMULACIÓN SOCIAL BASADA EN AGENTES. ADAPTADO DE EDMONDS (2000)



[2] A pesar de que en el artículo original Axelrod denomina a este objetivo *internal validity*, nosotros preferimos utilizar el término *verificación* por ser de uso más general en el campo de la simulación. En el presente artículo distinguimos entre el concepto de verificación de un modelo, definido anteriormente, y el concepto de validación de un modelo entendido como el proceso de evaluar la bondad de ajuste del modelo a las características de los referentes empíricos que intenta modelar.

Obviando la cuestión de si será o no será posible crear tal plataforma algún día, proponemos con este trabajo un objetivo de más fácil realización, y quizás inicialmente más fructífero para la comunidad, pues no implica la renuncia al uso de las herramientas por cada investigador. Consideramos fundamental establecer "un lenguaje común" de descripción de los modelos y los programas basados en agentes aplicados en las ciencias sociales.

Si bien el efecto Torre de Babel derivado de la multiplicidad de lenguajes de implementación puede ser peligroso, como señala Bruun (2002), nuestra experiencia en la programación con diferentes herramientas (plataformas y lenguajes) pone de manifiesto que la heterogeneidad y multiplicidad son una fuente de riqueza para la comunidad científica.

Si el lenguaje de implementación es sólo un medio para codificar un modelo, y no un fin en sí mismo, disponer de varias plataformas alternativas daría mayor alcance y rigor a los resultados de un modelo que hubieran sido replicados con diferentes plataformas de simulación.

La descripción del modelo y de la implementación en un lenguaje de simulación de acuerdo con unos criterios comúnmente aceptados facilitaría la tarea de replicación como medio de verificación. Más aún, elevaría el rango y la consideración de este tipo de trabajos, tan necesarios, por otro lado, para corroborar los cada vez más complejos resultados obtenidos con la simulación basada en agentes. De acuerdo con Edmonds y Hales (2003), podemos afirmar que "una simulación no replicada es una simulación de poca confianza".

La replicación no solamente aporta confiabilidad a los resultados logrados con un modelo, sino que si ésta se produce en diferentes lenguajes de programación, como sugiere Tesfatsion (2002), o incluso en diferentes paradigmas de modelado (Moss *et al*., 1997), habrá mayor conocimiento del sistema objeto de estudio.

Sin duda, no siempre es posible, al menos en términos del binomio esfuerzo-beneficio, implementar un modelo en cualquier plataforma. En general, el enfoque del problema viene condicionado por el conocimiento que se tiene de él. Mientras que en un enfoque procedimental se programan los procesos para determinar los estados, en un enfoque declarativo son los estados los que determinan los procesos. Por tanto, dependiendo del conocimiento del que parte el investigador y del fenómeno social por modelar, será uno u otro el más conveniente (véase Edmonds *et al*., 1996). Llevado a un caso límite, no tiene sentido para un científico social

desarrollar un modelo puramente lógico en un lenguaje de propósito general, pues necesitaría implementar un motor de inferencia adecuado, cuando dispone de lenguajes lógicos concebidos específicamente para dicho propósito.

Lamentablemente, pese a que con frecuencia surgen nuevos modelos con la metodología multiagente, no son muchos los que se replican. Y eso a pesar de que es este proceso el que puede dar a la simulación social la credibilidad y el rigor de otras ciencias. En palabras de Axelrod (1997), la replicación es uno de los sellos de la ciencia acumulativa. Notables excepciones a esta tendencia las constituyen los trabajos de Axtell *et al*. (1996), uno de los primeros ejemplos en replicar, alinear y acoplar diferentes modelos de simulación social, y el de Moss (2000), con sus "canonical task environments" (véase la tabla 1 que muestra algunos modelos basados en agentes que han sido replicados).

TABLA 1

Modelo	*Autor*	*Año*	*Publicado en*	*Replicado por*	*En*
Model of speculative learning agents	Duffy, J.	2001	*Journal of Economic Dynamics and Control*	Rouchier, J.	M2M Workshop
Evolution of Cooperation without Reciprocity	Riolo, R.,Cohen, M. D. y Axelrod, R.	2001	*Nature*, 411: 441-443	Edmosnds, B. y Hales, D.	M2M Workshop
Conway's Game of Life,	John Horton Conway	1970	*Scientific American*, 223 (octubre, 1970): 120-123	Axtell, R., Axelrod, R., J. M. Epstein y M. D. Cohen	Computational and Mathematical Organization Theory, vol. 1, núm. 2, pp. 123-141.
Cohen, March and Olsen's Garbage Can	Cohen, M. D., March, J. G. y Olsen, J.	1972	*Administrative Science Quarterly*, 17, 1-25.	Axtell, R., Axelrod, R., J. M. Epstein y M. D. Cohen	Computational and Mathematical Organization Theory, vol. 1, núm. 2, pp. 123-141.

TABLA 1 CONTINUACIÓN

Modelo	*Autor*	*Año*	*Publicado en*	*Replicado por*	*En*
Schelling's Tipping Model	Schelling, T.	1978	*Micromotives and Macrobehavior*, Nueva York, W. W. Norton.	Axtell, R., Axelrod, R., J. M. Epstein y M. D. Cohen	Computational and Mathematical Organization Theory, vol. 1, núm. 2, pp. 123-141.
Axelrod's Evolution of Prisoner's Dilemma Strategies	Axelrod, R.	1987	*Genetic Algorithms and Simulated Annealing*.	Axtell, R., Axelrod, R., J. M. Epstein y M. D. Cohen	Computational and Mathematical Organization Theory, vol. 1, núm. 2, pp. 123-141.
March's Organizational Code	March, J. G.	1991	*Organizational Science*, 2, 71-87.	Axtell, R., Axelrod, R., J. M. Epstein y M. D. Cohen	Computational and Mathematical Organization Theory, vol. 1, núm. 2, pp. 123-141.
Alvin and Foley's Decentralized Market	Alvin, P., & Foley. D.	1992	*Journal of Economic Behavior and Organization*, 18, 27-51.	Axtell, R., Axelrod, R., J. M. Epstein y M. D. Cohen	Computational and Mathematical Organization Theory, vol. 1, núm. 2, pp. 123-141.
Kauffman, Macready and Dickenson's NK Patch Model	Kauffman, S., Macready, W. G., Dickinson, E.	1995	Santa Fe Institute Working Paper, 94-06-031.	Axtell, R., Axelrod, R., J. M. Epstein y M. D. Cohen	Computational and Mathematical Organization Theory, vol. 1, núm. 2, pp. 123-141.
Riolo's Prisoner's Dilemma Tag Model	Riolo, R.	1997	Santa Fe Institute Working Paper, 97-02-016.	Axtell, R., Axelrod, R., J. M. Epstein y M. D. Cohen	Computational and Mathematical Organization Theory, vol. 1, núm. 2, pp. 123-141.
Artificial Stock Market	Arthur, W. B., Holland, J. H., LeBaron, B., Palmer, R., y Taylor, P.	1997	*The Economy as an Evolving Complex System II*.	Ehrentreich	Computational Economics 0209001, Economics Working Paper Archive at WUSTL.

Las razones de la falta de modelos replicados hay que buscarlas en la dificultad de replicar, y presenta diversos problemas. Con base en nuestra experiencia y en la adquirida por otros autores, se puede considerar que los principales problemas que un investigador encuentra a la hora de replicar un experimento realizado sobre un modelo basado en agentes en simulación social son los siguientes:

1. En general se utiliza la aproximación basada en agentes en modelos complejos, lo que dificulta su correcto entendimiento, su descripción y, por tanto, su replicación.

2. Pequeñas diferencias en cualquier parte del modelo, *a priori* sin importancia, son capaces de evitar incluso la equivalencia distribucional entre los modelos. Es común en este tipo de modelos la aparición de efectos de dependencia histórica (ej. Pajares *et al.*, 2003) y las consecuentes desviaciones de los resultados a partir de los factores aleatorios, por lo que sólo un análisis detallado de sus trayectorias y multitud de simulaciones son capaces de determinar la equivalencia de los modelos.

3. Existen ambigüedades en las descripciones de los modelos. Un típico ejemplo de ambigüedad en la descripción de los modelos es la falta de especificación de la resolución de conflictos y empates.

4. Existencia de ambigüedades en la descripción de los experimentos y resultados. Es común la no especificación de todos los parámetros que han generado unos resultados específicos.

5. Descripciones incompletas de los modelos.

6. Descripciones completas pero erróneas, posibles apariciones de inconsistencias entre el modelo descrito y la implementación.

7. El peligro de los errores de punto flotante (Polhill *et al.*, 2003).

8. Los modelos de simulación multiagente además pueden ser contados desde diferentes puntos de vista; desde el punto de vista de un actor, en orden cronológico, desde un punto de vista global… Es necesario especificar el punto de vista empleado para el análisis de los resultados para que el replicador tome el mismo punto de vista a la hora de validar los modelos.

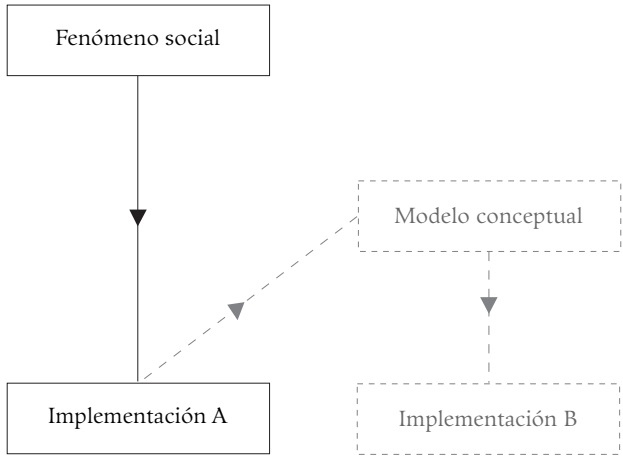
9. El posible efecto de malos generadores de números seudoaleatorios. Los generadores de números seudoaleatorios pueden variar en la longitud del ciclo de valores que producen antes de empezar a repetir la secuencia. A la hora de replicar un modelo conviene que la plataforma o el lenguaje en los que estén implementados tanto el modelo original como el replicado tengan generadores de números

seudoaleatorios con ciclos suficientemente largos como para evitar tendencias. También es importante cerciorarse de que estos números sólo se inicialicen una sola vez en cada simulación para evitar también el efecto de tendencias.

10. En los resultados de los modelos basados en agentes, con frecuencia se presentan exclusivamente modelos, experimentos y resultados. A la hora de replicar un modelo es de gran ayuda un apartado que explique la causalidad de los resultados; así se evita la posible equivalencia de los modelos por la casual cancelación de efectos.

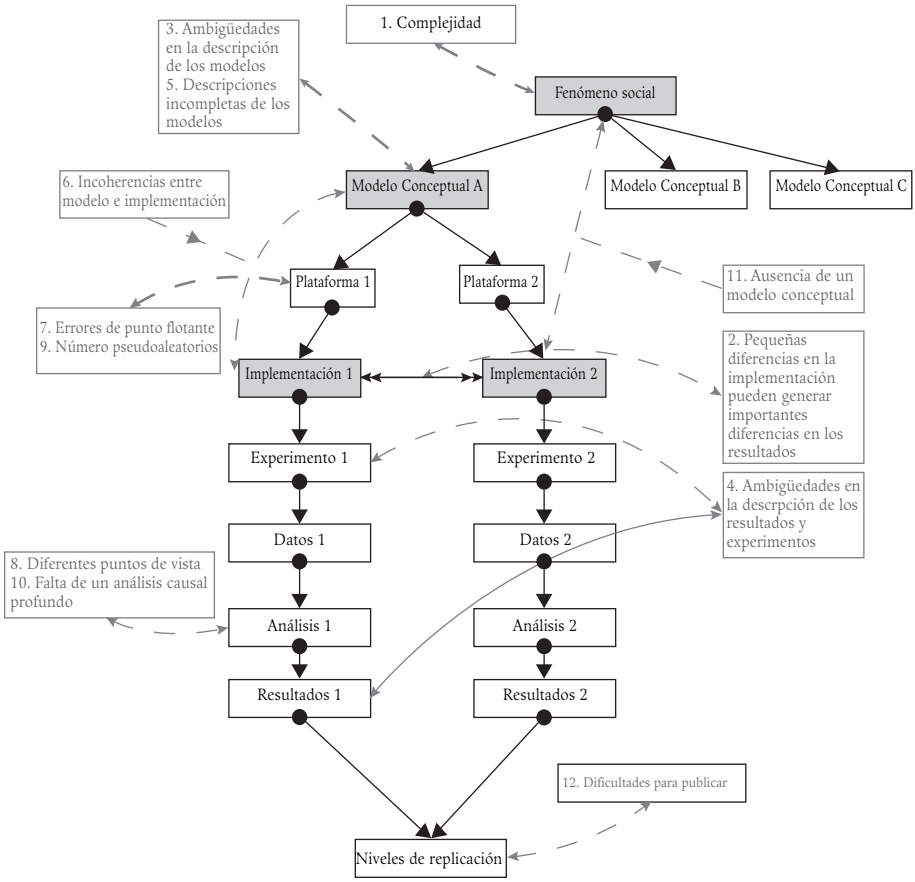
11. A veces la descripción de los modelos está ligada a una arquitectura específica, confundiéndose modelo e implementación. En estos casos el replicador se enfrenta al desafío de intuir el modelo conceptual que se sobrentiende en el modelo original.

FIGURA 2 A VECES EL MODELO CONCEPTUAL SÓLO SE REPRESENTA IMPLÍCITAMENTE A TRAVÉS DE LA IMPLEMENTACIÓN



12. Además, a toda esta problemática propia que posee la replicación hemos de unir la baja rentabilidad académica de los trabajos de este tipo. La falta de valoración por parte de los revisores de las revistas científicas no estimula su desarrollo. Este tipo de trabajos lógicamente no aportan originalidad, pero aportan o refutan credibilidad a los trabajos ya realizados y dan rigor a la metodología.

FIGURA 3 PERSPECTIVA GENERAL DE LOS PROBLEMAS TÍPICOS DE UN PROCESO DE REPLICACIÓN.



SDML versus *RePast*

Hemos manifestado la riqueza que proporciona la heterogeneidad de lenguajes de programación de modelos basados en agentes. En la actualidad hay muchas plataformas orientadas a la simulación social basada en agentes, cada una con sus objetivos y características: SWARM, RePast, SDML, Starlogo, ASCAPE, etc. Además existen

grupos de investigadores que utilizan Java y otros lenguajes de propósito general para implementar sus modelos.

En textos recientes podemos encontrar análisis comparativos entre CORMAS, Swarm y MadKit (Marietto *et al*., 2002), o perspectivas generales de la plataformas de simulación basadas en agentes (López-Paredes, 2001; Gilbert y Bankes, 2002). No obstante, no existe ninguna comparación entre dos que desde nuestra experiencia pueden ser las más firmes candidatas a alcanzar un mayor nivel de desarrollo por prestaciones y ser totalmente complementarias, que no sustitutivas: SDML y RePast.

RePast (Recursive Porous Agent Simulation Toolkit) es un marco de programación creado por la Universidad de Chicago para el desarrollo de simulaciones basadas en agentes utilizando el lenguaje Java —y, por tanto, utilizando un lenguaje imperativo y multiplataforma—. Proporciona un conjunto de librerías para la creación, la ejecución, visualización y recogida de datos de una simulación basada en agentes (Collier, 2003).

RePast se puede considerar simplificadamente como una colección de agentes de cualquier tipo y un modelo que organiza y controla la ejecución del comportamiento de esos agentes de acuerdo con un programa (*schedule*). Un programa en RePast se comporta típicamente como un simulador de eventos discreto (*discrete event simulation*) en que la unidad mínima es conocida como *tick*; también puede comportarse como caso degenerado del anterior como simulador paso a paso (*time-stepped model*).

SDML (Strictly Declarative Modeling Language) es un lenguaje de programación declarativa con características orientadas a objetos y fundado en una lógica no monotónica y temporal como es KD45 (Konolige, 1992). En SDML se representa el conocimiento mediante bases de reglas y bases de datos, y se utilizan como mecanismos de razonamiento el encadenamiento hacia atrás y hacia delante. Se dota a los agentes de reglas que determinan su comportamiento y que pueden compartir con otros agentes debido a sus características orientadas a objetos. El ser basado en un fragmento de lógica SGAL (*strongly grounded autoepistemic logic*) permite la prueba formal de la totalidad del modelo construido.

Las características de ambas plataformas son herencia de los paradigmas de modelado que implementan. RePast es apto para implementar el enfoque procedimental, mientras que SDML está más dirigido al enfoque declarativo. La forma

típica de funcionar con estas plataformas en ambos casos es describir el problema que se estudiará y, a partir de ello, deducir nuevas consecuencias mediante la ejecución del programa.

En el caso de RePast, las consecuencias se deducen de la especificación de los agentes y del entorno como un conjunto de variables y métodos de acuerdo con los paradigmas de la OOP. Con ellos generalmente se implementa un proceso concreto conocido de antemano y se concluye con los estados que emergen de esa implementación.

En el caso de SDML se implementa un conjunto inicial de creencias de cada agente y de situaciones ciertas del entorno, y se implementa la descripción del comportamiento como un conjunto de implicaciones. Si un conjunto de situaciones son ciertas (antecedentes), entonces otro conjunto de situaciones también son ciertas (consecuentes). A cada implicación de este tipo se le llama regla. La ejecución de un programa en SDML es el encadenamiento de las reglas de que dispone cada agente mediante el motor de inferencia, y la generación consecutiva de situaciones ciertas. En este caso se puede decir que son los estados los que se implementan, y el proceso el que emerge (Moss *et al*., 1997). El programador de SDML identifica las entidades y relaciones que merece la pena representar y qué relaciones guardan entre sí dichas entidades, y el procedimiento de inferencia decide cómo convertir los hechos en "solución" del problema.

Siguiendo la estructura de análisis de plataformas de simulación multiagente propuesto por Marietto *et al*. (2002) y ampliándola en algunos puntos, presentamos un esquema de comparación entre RePast y SDML (véase la tabla 2).

Prestaciones tecnológicas

A. Técnicas de gestión de la programación. En general, RePast se puede considerar un simulador basado en eventos. Al igual que en Swarm, un programa en RePast se compone de una colección de agentes con un programa de eventos asociado. Un caso degenerado de este tipo de funcionamiento es el uso de RePast como simulador paso a paso.

En SDML un programa se puede considerar como una colección de agentes dotados con un conjunto de reglas lógicas asociadas con diferentes niveles de tiem-

pos. En SDML es el motor de inferencia (basado en un formalismo lógico) el encargado de disparar las reglas ciertas en cada instante de simulación. Se puede considerar como un simulador basado en eventos o como simulador paso a paso.

Prestaciones de dominio

A. Lanzamiento de agentes. En RePast los agentes pueden ser puestos en instancias como objetos en sus propios hilos, permitiendo el modelado de simulaciones distribuidas. En SDML los agentes se convierten en instancias de tipo objeto.

B. Gestión de errores intencionados. No disponible en ninguna de las dos plataformas.

C. Integración con entornos controlados y no controlados. No disponible en ninguna de las dos plataformas.

Prestaciones de desarrollo

A. Desarrollo de arquitecturas de agentes. En este punto encontramos diferencias notables entre ambas plataformas. En RePast no existen restricciones respecto a la arquitectura interna que han de tener los agentes y, si bien aún no están disponibles las librerías específicas que implementen funcionalidades como algoritmos genéticos y redes neuronales, se pueden utilizar cualesquiera otras que estén programadas en Java. Por otra parte, SDML posee amplias facilidades para implementar agentes cuya representación del conocimiento corresponda al formalismo de los sistemas de producción de reglas, pero presenta dificultades para implementar métodos algorítmicos de manera eficiente.

B. Métodos de gestión de la comunicación. El intercambio de mensajes en RePast es sincrónico. En SDML los agentes pueden disparar sus reglas de forma sincrónica o asíncrona, y en los modelos más complejos existen diferentes agentes que actúan de forma secuencial, mientras que otros procesan sus acciones en paralelo.

C. Abstracciones organizativas. Ni en RePast ni en SDML están definidos específicamente los conceptos de rol. Sin embargo, en cuanto al concepto de grupo, en SDML se pueden establecer consideraciones jerárquicas y de agregación y agrupa-

ción mediante la estructura de *container*. En RePast el concepto de grupo se puede asumir al de colección de agentes, pero tampoco está definido específicamente.

D. Gestión de sociedades múltiples. Tanto en RePast como en SDML es posible trabajar con múltiples sociedades jerárquicas.

E. Gestión de estructuras espaciales y redes. Una característica interesante es la capacidad para implementar las relaciones espaciales asociadas a los individuos o grupos en los sistemas sociales. RePast incorpora un amplio espectro de *grids*: de dos dimensiones, toroidales, con diferentes capacidades de ocupación, etc., en el paquete *space*. RePast posee, además, un paquete específico para construir simulaciones de redes (*network simulations*), y se está desarrollando el Evolver, un entorno de desarrollo rápido. En estos aspectos SDML presenta ciertas desventajas, puesto que no posee herramientas específicas para manejar estructuras espaciales, y es preciso emplear las matrices y listas.

Prestaciones para el análisis y la exploración

A. Posibilidad de observar/intervenir en eventos de comportamiento. Los eventos de comportamiento son aquellos a los que un observador externo puede acceder. RePast intenta desacoplar, siguiendo la filosofía Swarm, las acciones de observación de las acciones de ejecución propias del modelo. Así, por ejemplo, mediante la definición de los métodos buildModel() y buildDisplay(), pero esta distinción no está tan claramente definida como en aquél (el *observer* y el *model swarm*). RePast no cuenta con herramientas específicas para seguir este tipo de eventos, y ha de ser el programador quien los implemente si quiere observarlos. En SDML existe una ventana que permite seguir todas las reglas que se disparan como ciertas —algunas de ellas pueden ser *assumptions* generadas por el motor de inferencia que serán confirmadas o no durante el resto de la simulación—. Esta opción es muy útil para el programador, pues facilita la supervisión del motor de inferencia durante la simulación.

B. Posibilidad de observar/intervenir en eventos cognitivos. Los eventos cognitivos son aquellos relacionados con las arquitecturas internas de los agentes. RePast posee herramientas tan potentes como las sondas (*probes*), elementos capaces de visualizar y modificar variables o ejecutar métodos internos del agente en cualquier instante de ejecución. SDML almacena en una base de datos ligada a cada

agente todos los eventos que se han desencadenado durante la simulación, ligados siempre al instante en que se han producido.

C. Gestión del análisis de datos. RePast posee herramientas que facilitan este requisito. Posee específicamente el paquete Analysis para esta función, lo que permite la definición de las fuentes de datos y relacionarlas con el tipo de gráfico específico que requiera el modelador. Además posee el paquete Gui que proporciona, entre otras, la posibilidad de tomar instantáneas de las simulaciones y hacer películas QuickTime®. Este es probablemente el mayor inconveniente de SDML: las capacidades gráficas son muy limitadas y lentifican la simulación.

D. Capacidades de depuración. RePast no posee ninguna herramienta específica para la depuración de errores, y ésta sólo se puede llevar a cabo mediante herramientas de desarrollo generales para la programación en Java, como JBuilder o Visual J++.

En los aspectos de depuración SDML presenta, en nuestra opinión, ventajas frente a RePast. SDML detecta de modo automático los conflictos entre reglas, y cualquier base de reglas en la que no existe una solución lógica consistente da como resultado un mensaje de error. Resulta fácil localizar las reglas fuente del error (Wallis & Moss 1994). El Debugger de SDML permite la simulación paso a paso como opción, y en cualquier caso el usuario puede observar para cada predicado de las cláusulas que conforman una regla los valores, o el tipo, o clase a que pertenecen. De esta forma se facilita la depuración de los errores debidos a incompatibilidades de tipos.

Además, SDML dispone del Querying, un espacio para realizar simulaciones limitadas que permiten probar y depurar las reglas que se van a incorporar a una base de reglas, y/o revisar aquellas que pensamos que no funcionan correctamente. El Querying se debe ligar a un instante de simulación (o *eternity*), y está disponible para cada agente del modelo, de forma que se puede emplear la base de datos creada en la última simulación para realizar pruebas. Las cláusulas que se introducen en el Querying serán simuladas, y el Debugger estará activo con todas sus funciones como en toda simulación.

TABLA 2

	RePast	*sdml*
	Información general	
Nombre completo	Recursive Porous Agent Simulation Toolkit	Strictly Declarative Modelling Language
Desarrollado por	The University of Chicago's Social Science Research Computing	Centre for Policy Modelling. Manchester Metropolitan University
Web	http://repast.sourceforge.net/	http://sdml.cfpm.org/
Lenguaje original	Java	Smalltalk
Paradigma de modelado	Imperativo	Declarativo
Características de POO	Sí	Sí
Extensibilidad	Características de POO	Herencia de módulos
Tiempo relativo de ejecución	Bajo	Alto
Lógica	No	FOSGAL (KD-45)
	Prestaciones tecnológicas	
Técnicas de gestión de la programación	Simulador basado en eventos	Simulador lógico basado en eventos
	Prestaciones de dominio	
Lanzamiento de agentes	Los agentes pueden ser lanzados como objetos e hilos	Los agentes pueden ser lanzados como objetos
Gestión de errores intencionados	No disponible	No disponible
Integración con entornos controlados y no controlados	No disponible	No disponible

TABLA 2 CONTINUACIÓN

	Prestaciones de desarrollo	
Desarrollo de arquitecturas de agentes	Arquitectura plana	Especialmente orientado para agentes lógicos
Métodos de gestión de la comunicación	Modo sincrónico	Modo sincrónico y asíncrono
Abstracciones organizativas	No trabaja ni con roles ni grupos	No trabaja con roles. Utiliza grupos a través de la estructura de container
Gestión de sociedades múltiples	Disponible	Disponible
Gestión de estructuras espaciales y redes	Fácilmente mediante paquetes espaciales y de redes	No muy desarrollada
	Prestaciones para el análisis y la exploración	
Observar/intervenir en eventos de comportamiento	No disponible formalmente	Acceso a las bases de datos y seguimiento de las reglas
Observar/intervenir en eventos cognitivos	Sondas (*probes*) los objetos	Acceso a todas las bases de reglas y datos durante la simulación
Gestión de análisis de datos	Paquetes específicos como Analysis o GUI	Herramientas pobres
Capacidades de depuración	No disponible	Detección automática de conflictos, Debbuger y Querying

■ Hacia un marco estandarizado para la replicación

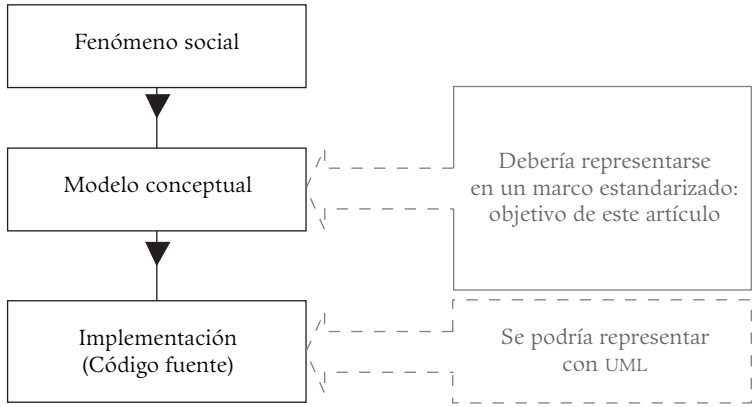
Como hemos visto en secciones anteriores, la replicación presenta multitud de posibles problemas que dificultan su realización. Muchos de estos problemas son consecuencia de las dificultades de descripción y especificación que conllevan los modelos basados en agentes y los experimentos llevados con ellos. En nuestra

opinión, el efecto Torre de Babel que existe en nuestra comunidad científica no es consecuencia de la multitud de plataformas existentes, sino de la falta de una estructura estándar de descripción de modelos, experimentos y resultados.

Existen diversos intentos desde la ciencia computacional de establecer marcos, lenguajes y herramientas para estandarizar el diseño, descripción, modelado y análisis de sistemas software basados en agentes (Iglesias *et al.*, 1999). Los informáticos (Kavi *et al.*, 2002; Odell *et al.*, 2000; Parunak y Odell, 2002) están intentado extender las herramientas utilizadas en la descripción de programas orientados a objetos como UML (*unified modelling language*), pero éstas no son una alternativa válida como marco para representar modelos sociales. En la simulación social existen algunas propuestas de marcos (Mentges, 1999; Simão y Pereira, 2003) e intentos de sistematización en la construcción de sociedades artificiales (Aguilera y López-Paredes, 2001).

UML es un lenguaje válido para representar lo que hemos venido llamando "implementación" (véase la figura 4). El uso de UML en la descripción de implementaciones multiagentes puede mejorar el entendimiento del modelo, por ejemplo, con los diagramas de actividad y de secuencia para representar el comportamiento de los agentes y sus interacciones (Oechslein *et al.*, 2001). Sin embargo, no es apropiado para capturar el modelo conceptual.

FIGURA 4



Como hemos remarcado, debido a la naturaleza exploratoria de la simulación (Edmonds y Hales, 2003), no poseemos unos requerimientos funcionales del sistema *a priori*, sino que los resultados del funcionamiento son inesperados, lo que dificulta la verificación.

Sin embargo, el ejercicio de replicación de un modelo parte con ventaja frente al diseño y la exploración de un modelo original; posee los resultados de los experimentos realizados en el modelo original.

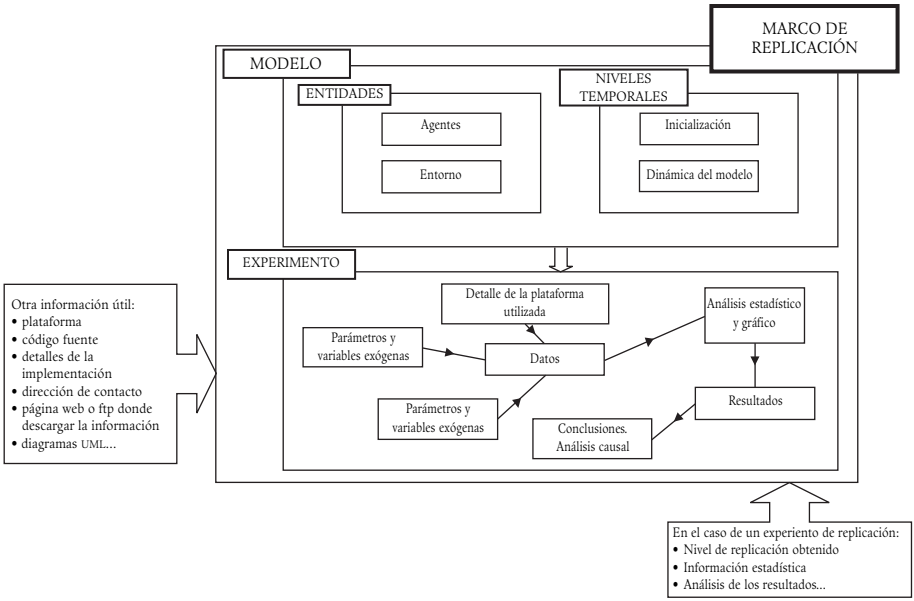
Por tanto, consideramos que el proceso de replicación de un modelo se puede llevar a cabo en dos etapas:

1. Reimplementación a partir de la descripción del modelo. Depuración mediante la comprobación de los aspectos especificados en la descripción y, si es posible, mediante el aislamiento de partes del modelo.

2. Replicación de los experimentos llevados a cabo en el modelo original. Análisis de las conclusiones de ambos modelos. En esta parte resulta tremendamente revelador el que los resultados sean diferentes en modo significativo, puesto que permite concluir una situación relevante, al menos uno de los modelos no está bien implementado (incluyendo aquí tendencias en los números seudoaleatorios, *floating point errors*…) o hay ambigüedades en la descripción del modelo que pueden provocar esas diferencias.

Como consecuencia, en nuestra propuesta del marco de replicación diferenciamos dos niveles de descripción necesarios en los ABSS, la descripción del modelo y la del experimento.

FIGURA 5 PERSPECTIVA GENERAL DEL MARCO DE REPLICACIÓN.



En la descripción del modelo en ABSS (siempre con alto nivel de detalle) podemos distinguir:

- Entidades: a) El entorno. b) Los agentes que participan en él.
- Niveles temporales: a) Inicialización. b) La dinámica de funcionamiento del modelo.

Para la descripción de las entidades del modelo proponemos el siguiente esquema:

1. Parámetros del modelo.
2. El entorno. Caracterización de manera completa y precisa.
   2.1. Tipo de entorno. Caracterización morfológica en lenguaje natural y gráfico.
      2.1.1. Entorno centralizado. Si los agentes tienen acceso a la misma estructura.

2.1.2. Entorno distribuido. Mediante redes o *grids*. Aquí es necesario especificar claramente el tipo de *grid*, su número de dimensiones, si es toroidal o queremos estudiar los "efectos esquina", si es multicapa, el número de agentes que admite por celda, si contiene recursos, la forma geométrica de la celda (hexagonal, triangular, cuadrada…).

2.1.3. Entorno híbrido. Combinación de los anteriores.

2.2. Acciones autónomas del entorno. Qué acciones realiza el entorno como respuesta al paso del tiempo (suponiendo el entorno dinámico), con independencia del efecto de las acciones de los agentes sobre él. Por ejemplo, la regeneración automática de recursos en el modelo de Sugarscape (Epstein y Axtell, 1996); la caracterización exhaustiva en lenguaje natural y, en el caso de algoritmos, la utilización de la especificación completa normalizada.

2.3. Posibles reacciones del entorno como respuesta a las acciones de los agentes. En general estas reacciones posiblemente pueden ser consideradas como la descripción de las acciones de los agentes y sus consecuencias en el entorno.

2.4. Otras consideraciones.

2.4.1. Si el entorno es un GIS, especificación de sus características.

2.4.2. Especificación de los posibles sistemas de resolución de incompatibilidades de actuación de agentes simultáneamente en el entorno.

3. Los agentes. Caracterización de manera completa y precisa.

3.1. Variables y constates que los caracterizan.

3.2. Arquitectura. En general, un agente dentro de un ABSS puede ser considerado como una estructura del tipo perceptores/sensores-mecanismo de razonamiento-efectores/actuadores. Los efectores son aquellas acciones que realiza el agente para captar información exterior a él; los efectores/actuadores son aquellas acciones que realizan los agentes con efecto en el exterior, bien sea la comunicación de un mensaje, un movimiento en el entorno (consideramos como actuadores las funciones motoras del agente) o una acción sobre otro agente. El mecanismo de razonamiento es la arquitectura propia del agente que le permite, a partir de los perceptores y de su "conocimiento"[3] adquirido, determinar qué acción tomar.

[3] Aquí tomamos la palabra "conocimiento" en sentido amplio para referirnos a cualquier mecanismo interno del agente que le permita tomar una decisión, incluyendo desde un sistema complejo con memoria, mecanismo de aprendizaje y sistema de decisión complejo hasta sistemas puramente probabilísticos de decisión.

3.2.1. Perceptores/sensores.

3.2.1.1. Accesibilidad. Especificación de qué partes del entorno son accesibles al apartado sensorial del agente y qué partes no lo son.

3.2.2. Actuadores.

3.2.2.1. Accesibilidad. Especificación de qué partes del entorno son accesibles al mecanismo actuador de los agentes.

3.2.3. Arquitectura interna. La arquitectura interna de los agentes determina su funcionamiento en gran medida y, por tanto, es una parte esencial en la replicación. Con frecuencia requiere el esfuerzo principal de descripción del modelo, ya que existen infinidad de posibilidades. Resulta difícil especificar aspectos relevantes de descripción generales para la multitud de arquitecturas posibles tan distintas como arquitecturas conectivistas, como una red neuronal, o sistemas basados en clasificadores o sistemas de producción (véase Ferber, 1999). Además, estos sistemas pueden estar dotados de mecanismos de aprendizaje y de memoria, en cuyo caso deben ser descritos igualmente. En cada caso el sistema debe quedar completamente definido sin ambigüedad con los sistemas de representación que le sean propios. La experiencia sugiere un esfuerzo extra en la definición completa de los algoritmos implementados en general y los algoritmos genéticos en particular, puesto que han sido fuente de los *bugs* más importantes detectados en trabajos de replicación (véase Ehrentreich, 2002; Edmonds y Hales, 2003). Otro aspecto esencial es la resolución de desempates y conflictos dentro de todos los procesos internos del agente (externos también), puesto que pueden tener una influencia crítica en los resultados.

3.2.4. Relación e intercambio de mensajes con otros agentes. Aquí resulta interesante la utilización de diagramas de secuencia de UML, especificando si la transmisión es sincrónica o asíncrona.

Para la descripción de los niveles temporales proponemos el siguiente esquema:

4. Inicialización del proceso.

4.1. Creación del entorno. Especificación detallada de los algoritmos de generación del entorno, tanto si corresponden a un proceso estocástico o determinista.

4.2. Creación de los agentes. Especificación detallada de los algoritmos de inicialización de los agentes. Situación inicial de los agentes en el entorno.
4.3. Creación de una historia previa artificial (Warm up process en el Artificial Stock Market de Santa Fe, Arthur *et al.*, 1997).

5. Dinámica de funcionamiento del modelo.
5.1. Descripción en lenguaje natural y mediante diagramas de flujo la secuencia de acciones planificada y el mecanismo motor de la simulación.
5.2. Comunicación entre los agentes.

Descripción del experimento

6. Experimento.
6.1. Parámetros y variables exógenas utilizadas. Generador de números aleatorios. Tipo de variables (precisión).
6.2. Resultados.
6.3. Análisis estadístico.
6.4. Análisis gráfico.
6.5. *Causality*. Explicación del mecanismo por el cual el modelo produce los resultados (internos y externos) encontrados.
6.6. Plataforma utilizada, código fuente y detalles de la implementación. Dirección de contacto con el autor, dirección Web donde encontrar esta información… En realidad, estos aspectos no deberían ser estrictamente necesarios para la replicación del modelo, y con la información anterior debería ser suficiente, pero debido a la complejidad que conlleva la replicación de modelos siempre es interesante contar con información redundante.

Además consideramos que cuando se realice un trabajo de replicación sobre un modelo se incluya la siguiente información relevante.

7. En el caso de ser un artículo de replicación.
7.1. Nivel de replicación alcanzado, de acuerdo con Axelrod (1997), ordenado de condiciones más a menos restrictivas:
7.1.1. Identidad numérica, cuando se reproducen los resultados exactamente. Obviamente si el modelo posee, como es habitual, algún pro-

ceso estocástico, para probar este nivel de replicación ambos modelos deben utilizar el mismo generador de números seudoaleatorios y la misma semilla (aparte de tratar de la misma forma con los redondeos, truncamientos…).

7.1.2. Equivalencia distribucional, conseguida cuando los resultados no se pueden distinguir estadísticamente. Generalmente, mediante los tests de Kolmogorov-Smirnov, el estadístico Mann-Whitney-Wilcoxon u otro método no paramétrico para estudiar la igualdad de dos distribuciones independientes (véase, por ejemplo, Hogg y Craig, 1978).

7.1.3. Equivalencia relacional, en la cual los modelos presentan las mismas relaciones internas entre los resultados.

7.2. Test de hipótesis utilizado, número de simulaciones efectuadas, estadístico, nivel de rechazo, potencia del test…

7.3. Análisis causal de la simulación, tanto para comprobar la equivalencia, como para analizar las razones por las que no se ha producido.

■ Conclusiones

1. Necesidad de replicar los resultados de la simulación de un modelo. Validación y método científico (rigor). La replicación es el principal método para la validación de modelos de simulación social basados en agentes. Constituye un proceso necesario para garantizar rigor a los modelos y dar confiabilidad a sus resultados.

La importancia de la replicación en ABSS es consecuencia directa de los problemas con los que se trata. Con frecuencia se abandonan simplificaciones utilizadas en otro tipo de metodologías, como conceptos de convergencia, homogeneidad, equilibrio, y se representan sistemas complejos. Por tanto, se puede esperar que se produzcan resultados dinámicos y originales, haciendo difícil distinguir los errores de implementación de resultados válidos.

2. Beneficios adicionales de replicar los modelos ABSS en diferentes lenguajes de simulación. El propio ejercicio de replicar un modelo trae consigo otro tipo de ventajas asociadas. La replicación realizada en diferentes plataformas permite

tomar ventaja de las características específicas que las plataformas presentan, aumentando el entendimiento y la mejor exploración del modelo. Además se aumenta la universalidad del modelo, se hace más accesible a más comunidades científicas y se permite su ampliación o reutilización.

3. Limitaciones a la replicación de modelos. A pesar de todo lo dicho, lamentablemente replicar no es fácil y presenta multitud de problemas en todos los niveles de ejecución de una simulación. Muchos de ellos vienen heredados de las insuficientes, ambiguas o incompletas descripciones de los modelos y de los experimentos y resultados obtenidos con ellos.

Baja Rentabilidad. Además, una de las principales razones por las que estos tipos de trabajos no se publican con más frecuencia es la falta de valoración por parte de los revisores de las revistas científicas, y por la falta de eventos donde comparar modelos, diferentes implementaciones, etc. Se agradece la excepción llevada a cabo en el M2M Workshop en Marsella en 2003.[4] Este tipo de trabajos, por supuesto, no aportan originalidad en el acervo científico, pero aportan o refutan credibilidad a los trabajos ya realizados.

A través de una sucinta comparación entre SDML y RePast se han analizado las condiciones de uso y las características más relevantes de ambos lenguajes. Se ha puesto de manifiesto la diversidad de aproximaciones que absorbe la ABSS a través de los diferentes enfoques de programación y la dificultad que conlleva replicar modelos en diferentes plataformas.

4. Hacia un marco de referencia para la replicación. De la experiencia adquirida hemos propuesto unas pautas de descripción de los modelos basados en agentes que pretenden ser un paso más hacia la estandarización de los procesos de representación, y facilitar su entendimiento, su replicación y su reutilización. En él se enfatiza la necesidad de describir los modelos y los experimentos bajo diferentes puntos de vista, con alto nivel de detalle y sin ambigüedad.

[4] http://cfpm.org/m2m/

■ Agradecimientos

Este trabajo ha sido financiado por el Ministerio Español de Ciencia y Tecnología, a través del proyecto Ref.: BEC-2001-2108, titulado "La Investigación socioeconómica desde la inteligencia artificial: Modelos basados en agentes (contribuciones en memoria de Herbert Simon)".

Los autores agradecen las sugerencias y observaciones de los participantes en la ESSA'03 Conference sobre una versión preliminar del trabajo presentado (Galán *et al.*, 2003).

■ Bibliografía

AGUILERA A. y A. López-Paredes (2001), *Modelado multiagente de sistemas socioeconómicos. Una introducción al uso de la inteligencia artificial en la investigación social*, San Luis Potosí (México), El Colegio de San Luis.

ARTHUR, W. B., Holland, J. H., LeBaron, B., Palmer, R. y Taylor, P. (1997), *Asset Pricing Under Endogenous Expectations in an Artificial Stock Market. In The Economy as an Evolving Complex System II*, editado por W. B. Arthur, S. Durlauf y D. Lane, Addison-Wesley.

AXELROD, R. (1997), "Advancing the Art of Simulation in the Social Sciences", en R.Conte, R.Hegselmann and P.Terna (eds.), *Simulating Social Phenomena*.

AXTELL, R., Axelrod R., J. M. Epstein y M. D. Cohen (1996), "Aligning Simulation Models: A Case Study and Results", *Computational and Mathematical Organization Theory*, vol. 1, núm. 2, pp. 123-141.

BRUUN, C. (2000), "Prospects for an Economics Framework for Swarm", en F. Luna y A. Perrone (eds.), *Agent-Based Methods in Economics and Finance: Simulations in Swarm*.

COLLIER, N. (2003), "RePast: An Extensible Framework for Agent Simulation", <http://repast.sourceforge.net/>

EDMONDS, B. (2000), "The Use of Models-making MABS more informative", en Moss, S. y Davidson, P. (eds.), *Multi Agent Based Simulation 2000, Lecture Notes in Artificial Intelligence*, 1979:15-32.

EDMONDS, B. (2002), "Towards an Ideal Social Simulation Language", 3rd Inter-

national Workshop on Multi-Agent Based Simulation (MABS'02) at AMMAS'02, Boloña, 15-16, julio.

EDMONDS, B. y Hales D. (2003), "Replication, Replication and Replication-Some Hard lessons from Model Alignment", en International Workshop Model to Model, Marcella.

EDMONDS, B., Moss, S. y Wallis, S. (1996), "Logic, Reasoning and A Programming Language for Simulating Economic and Business Processes with Artificially Intelligent Agents", en Ein-Dor, Phillip (ed.), *Artificial Intelligence in Economics and Management*, Boston, Kluwer Academic Publishers, pp. 221-230.

EHRENTREICH, N. (2002), "The Santa Fe Artificial Stock Market Re-Examined-Suggested Corrections. Computational Economics 0209001", Economics Working Paper Archive at WUSTL.

EPSTEIN, J. M. y Axtell, R. (1996), *Growing Artificial Societies. Social Science from the Bottom Up*, Cambridge, MIT Press.

FERBER, J. (1999), *Multi-Agent System: An Introduction to Distributed Artificial Intelligence*, Harlow, Addison Wesley Longman.

GALÁN, J. M., Downing, T., López-Paredes, A., Warwick, C. (2003), "Rigour and reliability in agent-based social simulation through replication", en First European Social Simulation Association Conference, ESSA'03, Groningen, 18-21 de septiembre.

GILBERT, N. y Bankes, S. (2002), "Platforms and methods for agent-based modeling", *Proceedings of the National Academy of Sciences USA*, vol. 99, Sup. 3, 7197-7198, mayo 14.

HOGG, R. V. y Craig A. T. (1978), *Introduction to Mathematical Statistics*, Collier Macmillan International Editions, cuarta edición.

IGLESIAS, C., Garijo, M. y González J. C. (1999), "A Survey of Agent-Oriented Methodologies", *Proceedings of the 5th International Workshop on Intelligent Agents V: Agent Theories, Architectures, and Languages (ATAL-98)*.

KAVI, K., Aborizka, M. y Kung D. (2002), "A Framework for Designing, Modeling and Analyzing Agent Based Software Systems", en Fifth International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'02).

KONOLIGE, K. (1992), "Autoepistemic Logic", en Gabbay, D. M., Hogger, C. J. y Robinson, J. A. (eds.), *Handbook of Logic in Artificial Intelligence and Logic Programming*, vol III, Oxford, Clarendon Press.

López-Paredes, A. (2001), *Análisis e ingeniería de las instituciones económicas. Una metodología basada en agentes*, Bilbao (España), Servicio de Publicaciones de la Universidad del País Vasco.

López-Paredes, A. (2003), *Ingeniería de sistemas sociales. Diseño, modelado y simulación de sociedades artificiales de agentes*, Valladolid (España), Servicio de Publicaciones de la Universidad de Valladolid (en prensa).

Marietto, M. B., David, N., Sichman, J. S. y Coelho, H. (2002), "Requirements Analysis of Agent-Based Simulation Platforms: State of the Art and New Prospects", en Proceedings 3rd. International Workshop on Multi-Agent Based Simulation (mabs'02), Boloña (Italia).

Mentges, E. (1999), "Concepts for an agent-based framework for interdisciplinary social science simulation", *Journal of Artificial Societies and Social Simulation*, vol. 2, núm. 2, <http://www.soc.surrey.ac.uk/JASS/2/2/4.html>.

Moss, S. (2000), *Canonical Task Environments for Social Simulation, Computational and Mathematical Organisation Theory*, vol. 6, núm. 3, pp. 249-275.

Moss, S., Edmonds, B. y Wallis, S. (1997), "Validation and Verification of Computational Models with Multiple Cognitive Agents", Discussion Papers 97-25, Manchester Metropolitan University, Centre for Policy Modelling.

Odell, J., Parunak, H. V. D. y Bauer, B. (2000), "Extending uml for Agents", en G. Wagner, Y. Lesperance y E. Yu (eds.), *Proceedings of the Agent Oriented Information Systems Workshop (aois) at the 17th National Conference on Artificial Intelligence*, pp. 3-17, Austin, Texas.

Oechslein, C., Klügl, F., Herrler, R. y Puppe F. (2001), "uml for Behavior-Oriented Multi-agent Simulations", en Dunin-Keplicz, Nawarecki (eds.), *From Theory to Practice in Multi-Agent Systems*, *Second International Workshop of Central and Eastern Europe on Multi-Agent Systems*, ceemas 2001 Lecture Notes in Computer Science 2296, Springer, pp. 217-226.

Pajares J., López-Paredes A., Hernández, C. (2003), *Industry as an Organisation of Agents: Innovation and R&D Management. Journal of Artificial Societies and Social Simulation*, vol. 6, núm. 2, <http://jasss.soc.surrey.ac.uk/6/2/7.html>

Parunak, H. V. D. y Odell, J. (2002), "Representing Social Structures in uml", en M. Wooldridge, G. Weiss y P. Ciancarini (eds.), *Agent-Oriented Software Engineering ii*, Berlín, Springer-Verlag, pp. 1-16.

Polhill, J. G., Izquierdo, L. R. y Gotts, N. M. (2003), "What every agent-based

modeller should know about floating point arithmetic", Submitted to Environmental Modelling & Software.

Simão, J. y Pereira, L. M. (2003), "Ethos: A mas Framework for Modelling Human Social Behavior and Culture", 4th Workshop on Agent-Based Simulation, Montpellier, april.

Suleiman, R., Troitzsch, K. G. y Gilbert, N. (eds.) (2000), *Tools and Techniques for Social Science Simulation*, Physica-Verlag.

Tesfatsion, L. (2002), "Agent-Based Computational Economics: Growing Economies from the Bottom Up", *Artificial Life*, vol. 8, núm. 1, pp. 55-82.

Wallis, S. y Moss, S. (1994), "Efficient Forward Chaining for Declarative Rules in a Multi-Agent Modelling Language", Center for Policy Modelling (cpm), Report núm. 004, <http://cfpm.org/cpmrep04.html>