Revista Ciencia e Ingeniería. Edición Especial: "Jornada de Modelado y Simulación" pp. 85-94. ISSN 1316-7081

Galatea: una historia de modelado y simulación

Galatea: a modelling and simulation story

Uzcátegui, Mayerlin[12*]; Dávila, Jacinto[1] y Tucci, Kay[12]
[1]Centro de Simulación y Modelos. Facultad de Ingeniería;
[2]Laboratorio SUMA. Facultad de Ciencias,
Universidad de Los Andes,
Mérida 5101, Venezuela.
*maye@ula.ve

Resumen

Galatea es una plataforma libre de código abierto para simulación de sistemas multi-agente que incorpora estrategias de simulación bien conocidas con la que cualquier modelista o simulista puede ensayar esas estrategias en problemas de simulación de sistemas complejos. La historia de Galatea comienza mucho antes que se planteara formalmente el proyecto con ese nombre. En 1993, nuestro muy joven Centro de Simulación y Modelos, CeSiMo, propone un proyecto para explorar la reimplementación de la plataforma de simulación Glider sobre una plataforma orientada a objetos dando origen a un prototipo experimental. El problema del cambio estructural, inspirado por investigaciones en economía, se había convertido entonces en uno de los objetivos de investigación fundamentales del CeSiMo y vendría a dictar también la pauta para Galatea. La noción de agente hizo su aparición en algunos reportes internos en los que se enfatizaba su importancia para modelar sistemas complejos como una economía nacional. En 1998 se planteó la posibilidad de integrar Glider con herramientas de inteligencia artificial para modelar agentes. En el 2000, un proyecto vendría a combinar aquel prototipo de 1993, con una teoría de agentes basada en lógica computacional que se planeaba integrar en una nueva teoría de simulación de sistemas multi-agentes. Allí nació Galatea. El logro fundamental para el proyecto, sin embargo, llegaría con las aplicaciones. En 2004, Galatea fue incorporada al banco de pruebas de un proyecto en biocomplejidad. Las lecciones aprendidas desde entonces, los aportes particulares del proyecto, así como los desaciertos y caminos futuros, son discutidos en este documento.

Palabras Clave: Software, simulación, agentes

Abstract

Galatea is free and open source software for multi-agent, DEVS simulations. The platform integrates well-known, and some experimental simulation strategies to allow any modelist or simulist to use them to solve problems involving simulations of complex systems. The history of Galatea precedes its naming. In 1993, our beginning Centre for Simulation and Modelling, CeSiMo, started a project to migrate the simulation system Glider to a object-oriented platform leading to a functional prototype. The problem of Structural Change, identified by the economists, had become a research objective for CeSiMo and it also did for Galatea. The notion of agent appeared in some internal reports amid the insistence upon its importance as an abstract tool to model systems as complex as a national economy. In 1998, there was a proposal to integrate Glider with some Artificial Intelligence developments to model agents. In 2000, that proposal became a project objective to combine the 1993 prototype with the agent modelling proposal to produce a multi-agent simulation theory that serve as the specification of a new simulation platform. At that time, Galatea was born. However, the real achievements started to be obtained in 2004, when Galatea was considered, among other tools, as a test-bed for a biocomplexity international project. The lessons learnt ever since, other achievements, strategic mistakes and future developments are discussed in this paper.

Keywords: Software, simulation, agents

1 Introducción

Un rastreo en Internet o en servicios electrónicos bibliográficos de los términos "software" y "simulación" devuelve millones de referencias[1]. Suponer que hay cientos de miles de sistemas de software para simulación no parece una exageración. No obstante, ese orden de magnitud en la cantidad de soluciones es todavía ínfimo comparado con la cantidad de problemas diferentes sobre los que se considera útil la simulación, sin mencionar que muchos de esos desarrollos no están al alcance de cualquiera. Así que parece seguro asumir que hay mucho trabajo de desarrollo pendiente y eso, en sí mismo, permitiría justificar un nuevo proyecto. Desde luego que hace 10 años, la situación era mucho peor. Pero la justificación básica para emprender el proyecto Galatea no fue esa solamente. La meta primera fue contar con una pieza de software integral, es decir, que incorporara todas las estrategias de simulación que se propusieran, especialmente de origen local, y con la cual cualquier modelista o simulista pudiera ensayar esas estrategias sobre problemas de simulación de sistemas complejos.

Ese ambicioso objetivo se ha venido cumpliendo paulatinamente. Galatea es una plataforma para simulación de eventos discretos, DEVS, con una semántica basada a una red de nodos como metáfora del sistema a simular, la misma semántica matríz del sistema Glider (Domingo y Hernández, 1985; Domingo, 1988; Domingo et al., 1993; Domingo, 1995; Domingo et al., 1996), el proyecto paterno local, que fuera formalizada como parte del nuevo proyecto (Dávila y Tucci, 2002; Dávila, Uzcátegui y Tucci, 2005) y luego generalizada para re-acomodar la simulación continua, la simulación de sistemas multi-agentes (Moreno et al., 2007) y la simulación distribuída (Dávila et al., 2005), simulación con autómatas celulares (Uzcátegui et al., 2000; Quintero et al., 2004) y simulación con modelos explícitos de espacios urbanos o arquitectónicos (Laffaille et al., 2005). En Galatea, no sólo se ha formalizado la semántica Glider, sino que se ha reimplementado desde cero el software, sobre plataformas que, a nuestro juicio, facilitan extensiones futuras en muy diversas direcciones gracias a su popularidad, apertura y distribución.

[1] En Google, los términos *software* y *simulación*, en ese orden, dan aproximadamente 1.200.000 resultados (0,33 segundos), mientras que *simulation software*, da Aproximadamente 32.800.000 resultados (0,32 segundos). En `Yahoo` son 4,090,025 para la primera y 98,500,936 para la segunda. En `Bing`, son 241.000 y 23.500.000 respectivamente. El metabuscador `Monstercrawler` devuelve 81 y 70. `MetaCrawler` da 79 y 69. De nuevo en Google, *software*, *simulación* y *agentes* sube la colecta a aproximadamente 3.430.000 resultados (0,37 segundos). Mientras que la combinación *Agents*, *simulation* y *software* da aproximadamente 1.610.000 resultados (0,29 segundos).

2 La historia de Galatea

La historia de Galatea, como suele suceder, comienza mucho antes que se planteara formalmente el proyecto con ese nombre. En 1993, nuestro muy joven Centro de Simulación, CeSiMo, propone un proyecto para explorar la re-implementación de Glider sobre una plataforma orientada a objetos (Tonella et al., 1993). Ese proyecto se convierte en un prototipo de Glider sobre C++ (Tucci, 1993) que incluyó un diseño de las estructuras de datos básicas para simulación de eventos discretos, DEVS, tales como objetos y clases, considerados ya entonces como fundamentales para un simulador del cambio estructural (Terán, 1994; Domingo et al., 1996; Palm, 1999; Domingo et al., 1996; Domingo y Tonella, 2000). El problema del cambio estructural, inspirado por investigaciones en economía, se había convertido en uno de los objetivos de investigación fundamentales del CeSiMo (Domingo et al., 1995; Domingo y Tonella, 2000) y vendría a dictar también la pauta para Galatea. En varios seminarios y cursos del centro se discutió la necesidad de una conceptualización alternativa que permitiera representar la dinámica de sistemas que contienen modelos parciales de ellos mismos, es decir, los sistemas endomórficos (Zeigler, 1990). Además, debería manejar conocimiento alternativo y tener capacidad para cambiar la estructura del propio sistema. La noción de agente, también llamados actores, hizo su aparición en algunos reportes en los que se enfatizaba su importancia para modelar sistemas complejos como una economía nacional[2]. A finales de los 90s, se planteó la posibilidad de integrar Glider con herramientas de inteligencia artificial para modelar agentes (Dávila, 1997). En el 2000, un proyecto (Uzcátegui, 2002; Dávila y Uzcátegui, 2002) vendría a combinar aquel prototipo de 1993, con una teoría de agentes basada en lógica computacional desarrollada en (Dávila, 1997) y que se planeaba integrar en una nueva teoría de simulación de sistemas multi-agentes (Dávila, Uzcátegui y Tucci, 2005). Allí nació Galatea.

El logro fundamental para el proyecto, sin embargo, llegaría con las aplicaciones. En 2004, Galatea fue incorporada al banco de prueba de un proyecto en biocomplejidad (Biocomplexity Group, 2002; Quintero et al., 2004; Moreno et al., 2007), con el propósito de servir como herramienta base para el desarrollo de un simulador de un sistema biocomplejo. Para atender el proyecto, se desarrolló una versión de Galatea con un sistema de agentes totalmente desarrollado en Java y el simulador fue integrado con otra herramien-

[2] Por ejemplo en el Modelo del Proceso de Apertura Petrolera. `http://cesimo.ing.ula.ve/MAPEV`.

ta de desarrollo local: el SpaSim (Moreno et al., 2002), que es una librería para construir modelos espaciales con autómatas celulares. Galatea y su teoría de agentes representados en una forma de lógica computacional fueron particularmente útiles para atender las especificidades de un modelo de simulación de una reserva forestal venezolana. En particular, la carencia de datos precisos del comportamiento histórico de los agentes fue compensada con reglas de conducta que pueden ser informalmente validadas por conocedores del sistema real, aún cuando no tengan mucha experiencia en lenguajes de programación de computadores (Moreno et al., 2007; Acevedo et al., 2008).

3 La arquitectura de la plataforma Galatea

El camino desde el diseño hasta la implementación ha sido tortuoso. Incluso partiendo de una especificación matemática, que supone una cierta claridad de los objetivos de desarrollo, convocar el esfuerzo de los programadores no ha sido fácil. La elección del sistema base: la plataforma Java, siempre rodeada de polémica, ha quedado ampliamente justificada. Necesitábamos un lenguaje de nivel medio-bajo (en el eje entre la máquina y el usuario), que nos aislara de las especificidades del hardware particular, pero que nos permitiera suficiente flexibilidad para la indispensable optimización del simulador, a bajo nivel, sobre una máquina (en este caso una máquina virtual) o muchas máquinas en una red. Otras virtudes que se le suelen atribuir a Java: como portabilidad y seguridad, han contribuido pero no han sido esenciales. En cambio es su condición de lenguaje matriz con sintaxis C, lo que ha resultado más útil como base para el desarrollo de otro lenguaje, esta vez específico para simulación, al que denominamos lenguaje Galatea. La sintaxis del lenguaje Galatea es, de hecho, una mezcla de la sintaxis Glider, las estructuras básicas de Java reemplazando a Pascal (que fue el lenguaje matriz de Glider) y las reglas de conducta de los agentes escritas en los lenguajes de programación lógica Actilog (Dávila, 2003) y OpenLog (Dávila, 1999). La compleja semántica de Galatea establece que el código Java sea compilado y ejecutado por el motor de simulación de eventos discretos, mientras que las reglas de los agentes son interpretadas por un motor de inferencia implementado sobre una máquina Prolog (Dávila y Uzcátegui, 2004).

Esa combinación de lenguajes y el nivel de desarrollo ha hecho más difícil convocar el apoyo de nuevos programadores, al punto que ya está institucionalizado el proceso de formación previo a la realización de cada trabajo académico. Los programadores reciben instrucción en Java y en Prolog bien enfocada en torno a las aplicaciones y a una colección de ejemplos: la modeloteca, originalmente herencia Glider, que se ha convertido en la herramienta instruccional fundamental.

La modeloteca es una colección de ejemplos con modelos de simulación de diversos sistemas. Se la puede consultar directamente en el repositorio Galatea, bajo los directorios `demos` y `contrib`[3]. La primera implementación de algunos ejemplos migrados manualmente desde Glider se muestra en (Uzcátegui, 2002). En (Vivas, 2008) se refiere un esfuerzo por generalizar el concepto de la modeloteca en torno a la idea de plantillas de modelos o metamodelos, mientras que en (Laffaille, 2005), el concepto de metamodelo incluye, además de esas plantillas para generar modelos, a toda la librería con software de soporte para el modelado y simulación de un tipo específico de sistemas, en ese caso modelos que describen movilidad en espacios urbanos o arquitectónicos. Allí nació gSpace, la librería originalmente usada para modelos de espacios urbanos y simulación de desalojos (Laffaille, 2005; Laffaille et al., 2005; Laffaille et al., 2008; Molina, 2010), pero que también ha sido aplicada recientemente al modelado y simulación de la movilidad vehicular (Pérez, 2010).

Como esperábamos, esa combinación de varios lenguajes ha significado un desafío particularmente difícil para el desarrollo de los traductores y compiladores. El primer esfuerzo de codificación del traductor de la sintaxis del Glider a Galatea se realizó en (Vargas, 2003) y se profundizó luego en (Ramos, 2006). Pero la lección principal en esa dirección es que el desarrollo de los traductores es un trabajo delicado que debe ser asumido por un grupo de consulta permanente. En Galatea se le denomina la comisión de sintaxis y se reúne periódicamente para acordar cambios a la sintaxis, desde luego, siempre conectados con la semántica de toda la plataforma. Todavía no se cuenta con un traductor completo para el lenguaje Galatea, pero se está bastante cerca del desarrollo del compilador que traduce a código Java y ya existe el interpretador de los lenguajes de programación lógica sobre Prolog. Además, se ha explorado la posibilidad de traducir todos los códigos al lenguaje base Java. En (Amaya 2003) se desarrolló un traductor automático de código Prolog a Java, ProgtoJav (Amaya y Dávila, 2009), con el cual se podría desplegar toda la plataforma sobre un sólo sistema base optimizado para ejecución. Una conclusión de ese trabajo, sin embargo, es que la plataforma mixta, que "corre" sobre las máquinas virtuales de Java y Prolog, es suficiente para soportar aplicaciones de mediana envergadura.

4 Modelos de sistemas dinámicos.

Un aspecto que continúa siendo muy novedoso en Galatea es la plataforma multi-agente. El simulador con-

[3] El repositorio Galatea funciona con el sistema de control de versiones *Subversion* y se le puede revisar en línea desde `http://galatea.sourceforge.net` y `http://galatea.svn.sourceforge.net/viewvc/galatea/`

vencional DEVS ha sido integrado con una capacidad para lanzar procesos o corutinas para cada agente en la simulación. Cada uno de esos procesos o corutinas se ocupa de la dinámica interna de cada agente y se tiene más de una forma de implementarla. Una implementación pura en Java con un motor de inferencia muy elemental, apropiado para agentes reactivos, se empleó en (Ablan et al., 2003; Quintero et al., 2004; Moreno et al., 2007). Pero también se cuenta con una implementación que emplea la máquina Prolog para proveer a cada agente de un motor de inferencia que puede usar como sistema de planificación de sus acciones. Esta es una implementación que puede admitir agentes reactivos, pero también agentes proactivos que planifican sus acciones en procura de metas de alto nivel (Dávila, 2011).

La plataforma multi-agente se ha prestado para experimentos innovadores de simulación. En (Gómez, 2002; Gómez, 2005), el simulador es transformado en una máquina distribuida, con un procesador real al servicio de cada agente en la realización de simulaciones multi-agentes. Ese esfuerzo se ha convertido en una buena carta a jugar cuando quiera que algún sistema a simular exija mayores recursos de cómputo en virtud de su complejidad. Siempre será posible distribuir a los agentes entre varios procesadores, sin mencionar que el propio simulador DEVS podría aprovechar cálculos en paralelo. De hecho, en (Díaz, 2002), Galatea se conecta con Grass, un conocido sistema de información geográfica libre. Ese trabajo fue un primer esfuerzo en la dirección a integrar la simulación con la gestión de información geográfica. El simulador se conecta, mediante un API de consulta y actualización, con la base de datos geográfica que gestiona el Grass.

Algunos de nuestros experimentos con sistemas multi-agentes precedieron a Galatea y fueron implementados sobre Glider. En (Contreras, 1999), se modeló un sistema de transporte público local con usuarios y vehículos, mientras que en (Ferrer, 2003), el concepto de agente fue incorporado a los modelos de simulación de una empresa de desarrollo de software y de una institución educativa en los cuáles las creencias de los agentes, incluyendo sus ontologías, fueron explícitamente representados y probado en Prolog. También antes de tener la capacidad multi-agente, Galatea fue usada en (Lopez, 2003) para implementar un alineador de secuencias de ADN usando un mecanismo de optimización basado en un algoritmo genético. Esos experimentos contribuyeron a mostrar la necesidad y a probar la factibilidad de una plataforma integrada para simulación.

Otra aproximación al modelado multi-agente fue posible incluso con Galatea antes de que tuviese incorporado el simulador multi-agentes. En (Grisolía, 2003) se codificó, en Java y para el simulador DEVS de Galatea, una versión completa de un modelo de simulación de la economía de Venezuela, original del Prof. Carlos Domingo, quién lo codificó en Glider; convirtiéndolo en un juego de simulación. Este modelo-juego permite que los humanos representen a los actores de la economía mientras participan en la evolución del sistema económico en rondas financieras anuales. Ese trabajo se hizo al mismo tiempo que el desarrollo complementario y soporte para internet del modelo juego (Dávila, 2003) el cuál demostró cuan fácil es implementar la plataforma cliente-servidor para simulación de juegos sobre Internet con Galatea.

Aún más significativo para la tecnología multi-agente es la variedad de posibilidades para la codificación de agentes. En la semántica original Glider, como demuestran aquellos ejemplos, es posible concebir representaciones de los humanos que realizan acciones o se desplazan por el sistema simulado. Incluso es posible representar conceptos esenciales en las organizaciones humanas tales como "proceso" y "servicio" que guardan una relación natural con los agentes. La siguiente sección elabora al respecto.

4.1 Modelos de la burocracia

La descripción explícita de los agentes puede ser una herramienta muy útil para abordar la complejidad al nivel de detalle apropiado en ciertos sistemas. Sin embargo, no es la única forma de dar cuenta de conductas humanas en un modelo de simulación. La simulación tradicional DEVS (Zeigler, 1976; Zeigler et al., 2000) permite, sin más, modelar un proceso como una serie parcialmente ordenada y posiblemente recurrente o concurrente de actividades caracterizadas como servicios. Un servicio es, en términos simples, una cola de solicitantes que son servidos por uno o más agentes desde una taquilla, durante ciertos tiempos determinables.

Esa conceptualización está imbuida en la semántica de Galatea, de manera que un servicio puede ser fácilmente representado en un modelo de simulación como se muestra en la Fig. 1 y un proceso, de la misma manera,

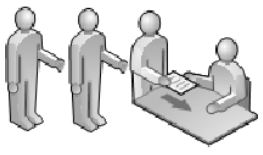

Fig. 1. Un servicio es una cola y una taquilla.

como un agregado de servicios, como en la Fig. 2.

La Fig. 3 ilustra, con un ejemplo de un sistema real, la posibilidad de describir un proceso burocrático institucional [4],para otorgar subvenciones

[4] El modelo fue diseñado a partir de las especificaciones suministradas por M. Ruíz e I. Vivas, encargadas por varios años de

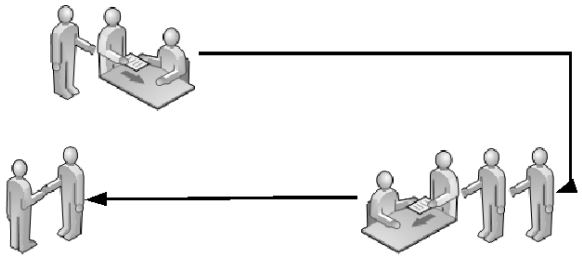

Fig. 2. El proceso es una composición de servicios.

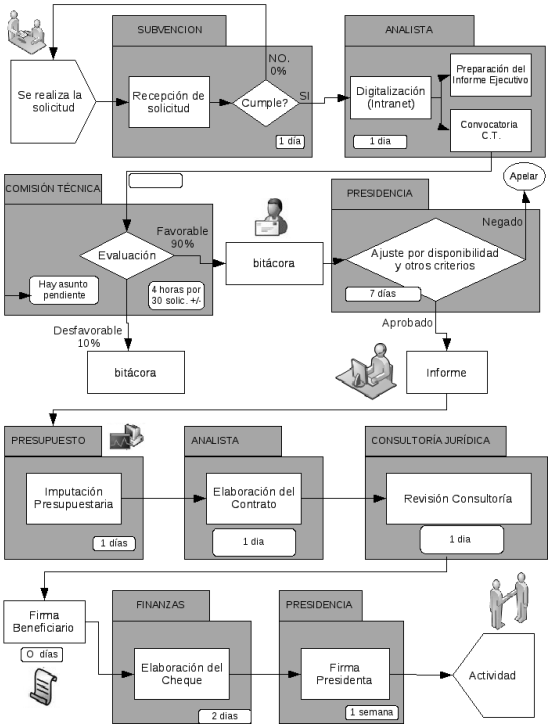

Fig. 3. Un Proceso Burocrático.

(ayudas económicas) a investigadores y proyectos de investigación. La descripción corresponde a una especie de diagrama de flujo que usa símbolos que tienen una semántica específica en Glider mostrada en (GLIDER Development Group, 2000) y otros que corresponden a los tradicionales diagramas de flujo. La única variante es la utilización de unos recuadros con etiquetas que identifican a cada agente y que incluyen sus respectivas acciones o decisiones. Cada agente es responsable de un servicio y el conjunto completo, con sus direcciones de flujo bien definidas, constituyen un proceso burocrático.

El paquete `contrib.burocracia`, en el repositorio Galatea, contiene la codificación de un modelo elemental, pero completamente funcional (simulable) de ese proceso real. Con ayuda del código en `contrib.gGui`, es posible visualizar el comportamiento de ese sistema siguiendo el proceso en cada servicio de varias maneras. Es posible reportar, como se muestra en la Fig. 4, el

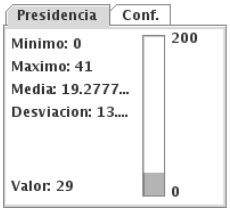

Fig. 4. Ocupación de un servicio.

nivel de ocupación de cada servicio, con las estadísticas de sus colas. Resulta, además, sumamente conveniente para los tomadores de decisiones conocer la "inclinación hacia la estabilidad" de cada servicio, graficando el comportamiento de las colas a lo largo del tiempo, como se muestra en la Fig. 5. Las gráficas de "inclinación

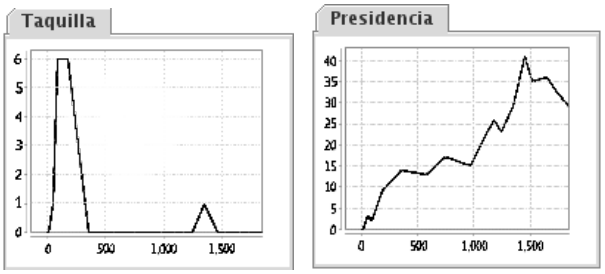

Fig. 5. Inclinación hacia la estabilidad. Taquilla es sostenible. Presidencia no.

hacia la estabilidad" permiten saber si la configuración actual del sistema modelado se inclina hacia una conducta estable o deriva sin remedio hacia una situación insostenible en la que las colas de los servicios se llenarán y estos colapsarán.

La Fig. 6 muestra un ejemplo de interfaz gráfica para el modelo `contrib.burocracia`, donde, además de los reportes de niveles y de las tendencias de las colas de los servicios, se le ofrece al usuario información adicional del proceso de simulación.

5 Modelos de cambio estructural

Todos los proyectos descritos hasta este punto, a pesar de su enorme valor práctico, no plantearon mayor desafío a la expresividad de los lenguajes de simulación existentes. El cambio estructural (Domingo y Tonella, 2000) ha planteado ese desafío. Se trata de cómo modelar sis-

ese proceso en Fundacite Mérida. El modelo fue implementado por J. Dávila y L. Andrade. La interfaz gráfica se le debe a N. Vicuña quien adaptó el software del JfreeChart a Galatea.

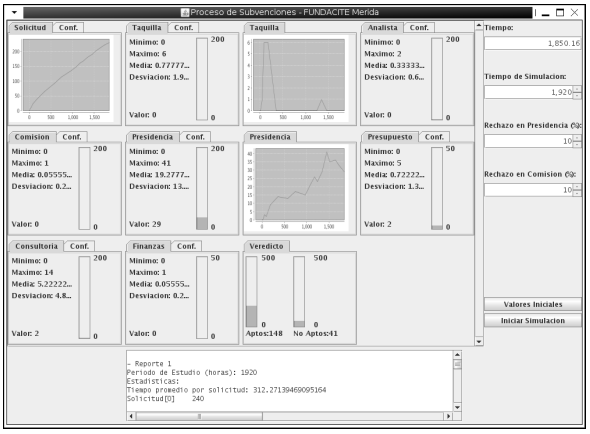

Fig. 6. Interfaz al Usuario del Modelo del Proceso Burocrático.

temas que combinan más de una dinámica tradicionalmente descrita por algún modelo matemático bien definido. En economía es común observar que un modelo matemático se agota cuando el sistema alcanza ciertos bordes de los dominios de las variables que le ha incorporado el modelista. Se habla entonces de que el sistema cambia de estructura y pasa a necesitar un nuevo modelo matemático con una suerte de vigencia por tramos (en las trayectorias del modelo) a partir del punto de quiebre del modelo anterior. La solución ingenua consiste en proveer una metarepresentación que organiza los modelos y los "habilita" de acuerdo al tramo por el que transcurre el sistema simulado. No sólo es esta una solución muy rígida en términos matemáticos, pues obliga al modelista a anticipar todas las estructuras posibles y sus interconexiones, sino que además, puede conducir a implementaciones muy pesadas en términos computacionales.

La migración de Glider a una plataforma orientada a objetos estuvo motivada, entre otras cosas, por la posibilidad de asociar código a los elementos cambiantes o dinámicos en una simulación. La plataforma base del Glider, el lenguaje Pascal, no incorporaba entonces, en ninguna de sus realizaciones, facilidades para asociar código a las estructuras de datos y esto impedía, por ejemplo, que los mensajes, usados en la semántica Glider para representar elementos que se desplazan por la red de nodos, tuvieran asociados instrucciones específicas para adecuar su conducta a contextos específicos de simulación, análogos a los tramos que se mencionaron antes. Era entonces una facilidad importante a procurar en dirección al cambio estructural.

Pero hubo también otra motivación importante para la migración, esta vez no solamente a la orientación por objetos, sino hacia la orientación por agentes. En aquellas primeras conversaciones con Carlos Domingo y Giorgio Tonella, se perfiló la posibilidad de que los agentes se convirtieran en "fuentes" de los cambios, tal como fue discutido y formalizado, en el contexto de la programación lógica y la inteligencia artificial en (Dávila, 1997; Dávila y Uzcátegui, 2004). El objetivo planteado fue explorar el uso de los agentes para generar el cambio estructural, a propósito de lo cual se plantearon algunos escenarios simplificados de estudio.

Con el ánimo de abreviar este presentación, permita el lector que, a la usanza tradicional en lógica, se use un ejemplo muy simple para ilustrar cómo Galatea ofrece los medios para codificar modelos en los que los agentes actúan como fuentes del cambio estructural.

5.1 El ejemplo del Gerente Bancario

Un modelo de las taquillas de un banco se ha convertido en un objeto instruccional tradicional en los cursos de modelado y simulación. Galatea lo hereda de Glider ya codificado como una pequeña red de nodos, con un nodo de entrada que representa la puerta del banco por la que arriban los clientes, representados como mensajes. Las taquillas del banco son representadas por un arreglo de nodos de tipo recurso, tal como los que usamos para representar servicios en el modelo de la burocracia, y la salida del banco es un simple nodo de salida. Los clientes se acumulan, siempre que la tasa de entrada supere a las de servicio, en la cola corrrespondiente de cada nodo recurso que representa una de las taquillas. El número de taquillas es, junto con otros parámetros de modelo, una decisión que toma el modelista al momento de codificar el modelo. Es decir, esa es la especificación de la estructura del sistema que permanece a lo largo de toda simulación en los modelos tradicionales.

Uno puede codificar un modelo flexible de un banco en el que se abran y cierren taquillas de acuerdo a las demandas de los clientes (es decir, la cantidad de clientes en el banco). Lo que se haría, normalmente, para lograr tal modelo es declarar una estructura máxima con todas las taquillas posibles en el banco y se ofrecería una manera de habilitar y deshabilitar taquillas según sea necesario. Esto es equivalente a programar un modelo de funciones matemáticas por tramos, en el que cada función se "activaría" en el tramo correspondiente para dar cuenta de la conducta de las variables dependientes, dado los valores que las independientes adquieren en ese tramo.

Lo que se propone acá es sutil pero profundamente diferente. Se propone que el propio modelo contenga el mecanismo, no para habilitar y deshabilitar subestructuras de la macro-estructura, sino para crear y eliminar directamente a las estructuras específicas. Galatea ofrece esta posibilidad gracias, en principio, a la plataforma subyacente orientada a los objetos, que permite crear y eliminar objetos con gran flexibilidad y robustez; pero también gracias a las previsiones para modelar agentes que se encargan de hacer el trabajo.

El modelo del Gerente, ubicado en el paquete `contrib.gerente` del repositorio Galatea, ilustra todo

esto. El modelo está definido en el nivel Galatea en el archivo `Gerente.g`, que se muestra en las figuras 7-9 y

```
TITLE Sistema simple de taquillas con gerente
NETWORK
  Entrada(I){
    IT = 10;
    SENDTO(Taquilla[MIN]);}
  Taquilla (R) [3]{
    RELEASE
    SENDTO(Salida);
    STAY = 45;}
  Salida (E){}
DECL
  STATISTICS ALLNODES;
INIT
  TSIM = 300;
  ACT(Entrada,0); END.
...
```

Fig. 7. El Gerente Bancario. Primera Parte: contiene a las secciones `TITLE`, `NETWORK`, `DECL` e `INIT`.

que da origen a las clases y objetos, en el nivel Java, que se describen en (Dávila, 2011). La conducta del modelo es idéntica a las descrita en el viejo modelo de las taquillas. De hecho, las líneas de código mostradas en la Fig. 7 corresponden exactamente con la forma tradicional de describir el banco y su sistema de taquillas.

En Galatea se incorpora, con las líneas de código de las figuras 8 y 9, el modelo un agente con capacidad

```
...
AGENTS
 Gerente {
  abd(cola_larga).
  abd(crear_taquilla).
  abd(taq_vacias).
  abd(elim_taquilla).
  abd(revisa_cola).
  observable(cola_larga).
  observable(taq_vacias).
  user_built(timing(_)).
  if timing(T)  then revisa_cola(T).
  if cola_larga then crear_taquilla.
  if taq_vacias then elim_taquilla.
  // código prolog para predicados user_built
  timing(T) :- gensym('', C), atom_number(C, T).
 }
...
```

Fig. 8. El Gerente Bancario. Segunda Parte (`AGENTS`).

para crear y eliminar taquillas de acuerdo a ciertas condiciones que se sugieren.

Además de todo lo dicho sobre cambio estructural, el modelo del agente es un excelente espacio para ilustrar porqué identificamos a Galatea con una familia de lenguajes. Como podrán ver, la Fig. 7 muestra el código tradicional Glider, con algunas modificaciones que explotan la sintaxis Java para manipular objetos (ver el código `Taquilla3.java` en el repositorio). Pero también se incluye código en lenguajes de programación lógica, en la sección AGENTS, para activar al agente gerente (Fig. 8); y la sección INTERFACE (Fig. 9), donde se define la interfaz entre los agentes y el ambiente para

```
...
INTERFACE
  public void revisa_cola(double t, Agent ag,
                          jpl.Integer at) {
    int clientes_en_banco = 0;
    for (int i=0; i<Taquilla3.taquilla.length;i++)
      clientes_en_banco =+
              Taquilla3.taquilla[i].getEl().ll();

    if (clientes_en_banco > 10)
      agente.inputs.add("cola_larga");
  }
  public void queja(double t, Agent ag) {
      System.err.println("#"+ag.agentType+
          ag.agentId + ": Cola larga!!");
  }
  public void crear_taquilla(double t, Agent ag) {
    if (Taquilla.mult > Taquilla.maxMult) {
      System.out.println("Banco lleno!!");
    else
      addNodeInstance(Taquilla); }
  }
  public void taq_vacias(double t, Agent ag) {
    for (int i=0, j=0; i<Taquilla.mult; i++)
      if (Taquilla3.taquilla[i].getEl().ll()+
          Taquilla3.taquilla[i].getIl().ll()==0) {
        j++;
        if (j > 1)
          ag.inputs.add("taq_vacias");
      }
  }
  public void elim_taquilla(double t, Agent ag) {
    int i = 0;
    while ((i < Taquilla.mult - 1) &
           (Taquilla3.taquilla[i].getIl().ll()+
            Taquilla3.taquilla[i].getEl().ll()>0))
      i++;
    if (Taquilla3.taquilla[i].getIl().ll()+
        Taquilla3.taquilla[i].getEl().ll()==0)
      delNodeInstance(Taquilla,i);
  }
```

Fig. 9. El Gerente Bancario. Tercera Parte (`INTERFACE`)

que el simulador pueda interpretar las acciones propuestas por los agentes sobre el sistema, está codificada en Java.

Estas extensiones permiten que un modelo tradicional de simulación, las taquillas de banco, incorpore elementos del cambio estructural. En este caso (ver los código de `contrib.gerente` en el repositorio Galatea), el banco que comienza funcionando con 3 taquillas, en ciertas condiciones de alta demanda (fijando el tiempo entre llegadas de personas en 1) y con suficiente tiempo (fijando el tiempo de simulación en 300), termina con 11 taquillas, es decir, 7 nuevas taquillas abiertas durante la simulación, como muestra la Fig. 10.

6 Desaciertos en el proyecto

Galatea, como todo servicio o producto tecnológico, debe ser evaluada por sus usuarios. No se entienda lo que sigue como un anticipo, ni mucho menos como un reemplazo de esa evaluación, a la que el proyecto se so-

```
Banco lleno!!
300.0        Scan (E)Salida[0]
300.0        Scan (R)Taquilla[10]
300.0        Scan (R)Taquilla[9]
300.0        Scan (R)Taquilla[8]
300.0        Scan (R)Taquilla[7]
300.0        Scan (R)Taquilla[6]
300.0        Scan (R)Taquilla[5]
300.0        Scan (R)Taquilla[4]
300.0        Scan (R)Taquilla[3]
300.0        Scan (R)Taquilla[2]
300.0        Scan (R)Taquilla[1]
300.0        Scan (R)Taquilla[0]
...
Sistema simple de tres taquillas con gerente
Time:              301.0
Time Stat:         301.0
Replication:         1
Date:              04/49/09 11:49:06
Elapsed time:      0h 1m 53.169s
```

Fig. 10. Fragmento del archivo de traza del modelo del Gerente. Notese que en el instante 300.0 hay 11 taquillas: `Taquilla[0] - Taquilla[10]`.

mete con humildad. Hay, sin embargo, una conjunto de decisiones que pueden estar afectando justamente la posibilidad de que Galatea, como Glider antes, estén al alcance para uso y evaluación por parte de una comunidad mayor de usuarios.

Con todos los interesantes desafíos técnicos que ha enfrentado el proyecto, las decisiones acerca del cómo aprovechar la fuerza de trabajo, siempre voluntaria, han privilegiado el desarrollo de la plataforma para llevarla al punto que otros programadores puedan usarla para producir simuladores. Los llamados usuarios finales, no expertos o poco conocedores de computación o simulación, no han sido atendidos lo suficiente. Se han hecho esfuerzos considerables, en (Cabral, 2001; Ramos, 2006), para dotar a Galatea de un ambiente gráfico de desarrollo con servicios visuales para apoyar a quienes tengan menos pericia en el desarrollo y codificación de modelos. Pero son desarrollos aún no completos y que, en todo caso, requerirán del usuario o usuaria una formación básica en computación, con lo que se excluye de su uso a una población, incluso con conocimientos científicos, que bien podrían favorecerse de la simulación para resolver problemas.

Es posible que una estrategia diferente, como la realización de ciertos simuladores con interfaces amigables a usuarios no expertos, pudiera potenciar la comunidad de usuarios y atraer la atención, no sólo hacia Galatea, sino hacia el uso de la simulación para enfrentar, entender y resolver problemas cotidianos.

7 Conclusión y caminos futuros.

Galatea ya sirve al objetivo planteado en su origen: contar con un sistema de software integral, es decir, que incorpora todas las estrategias de simulación bien conocidas y con el cual cualquier modelista o simulista puede ensayar esas estrategias sobre problemas de simulación de sistemas complejos. Hay, desde luego, un largo camino por recorrer con las aplicaciones, posiblemente considerando las lecciones sugeridas en la sección anterior. En particular, se plantea explorar soluciones de GeoSimulación para usuarios no expertos y la simulación para optimización.

En (Dávila, 2011) y en próximas publicaciones, por ejemplo, se explica cómo la simulación con agentes sirve a ejercicios de optimización (es decir, para mejorar la conducta de todo un sistema) por medio de la capacidad que tienen los agentes de corregir su conducta aprendiendo nuevas reglas durante la simulación. Este puede constituir un nuevo y más expresivo marco para problemas de optimización que requieran la forma genérica

$$\text{maximizar} \left(\sum_i C_i \right) \tag{1}$$

sujeto a

$$\forall_i \left(C_i = K_p T_i \text{ si } C_i < U_i \text{ y } C_i = K_c T_i \text{ si } C_i \geqslant U_i\right) \text{ y } \sum_i T_i \leqq T \text{ y } \forall_i (T_i \geqq T_i^0) \ , \tag{2}$$

en los cuales las restricciones tradicionales existentes en problemas de programación lineal son expresadas como reglas condicionales y los agentes en la simulación se encargan de procurar la solución.

Todas esas alternativas se constituyen en proyectos de largo aliento y por tanto, difíciles de organizar y realizar desde las condiciones locales de trabajo. La historia acá contada, sin embargo, es el mejor testimonio de que el trabajo es posible, valioso y reconfortante.

Agradecimientos

Este trabajo ha sido financiado parcialmente a través del proyecto C-1696-10-05-B del Consejo de Desarrollo Científico, Humanístico, Tecnológico y Artístico, CDCHTA, de la Universidad de Los Andes.

Referencias

Ablan M, Dávila J, Moreno N, Quintero R y Uzcátegui M, 2003, Agent modeling of the caparo forest reserve, En *ESMc'2003. European Simulation and Modelling Conference*, pp. 367–372, Naples, Italy, October 2003.

Acevedo MF, Callicott JB, Monticino M, Lyons D, Palomino J, Rosales J, Delgado L, Ablan M, Dávila J, Tonella G, Ramírez H y Vilanova E, 2008, Models of natural and human dynamics in forest landscapes: Cross-

site and cross-cultural synthesis, Geoforum, Vol. 39, No. 2, pp. 846-866.

Amaya JE, 2003, Integración de la programación declarativa y la programación orientada por objetos para desarrollar agentes inteligentes, Tesis de Maestría, Maestría en Computación, Universidad de Los Andes, Mérida, Venezuela.

Amaya JE y Dávila J, 2009, Mecanismos de traducción de prolog a java mediante progtojav, En *The Seventh Latin American and Caribbean Conference for Engineering and Technology*, pp. WE1-1 - WE1-10, San Cristobal, Venezuela.

Biocomplexity Group, 2002, Biocomplexity: Integrating models of natural and human dynamics in forest landscapes across scales and cultures, UNT-Yale-RICE-ULA-UNEG Universities. NFS Grant CNH BCS-0216722.

Cabral M, 2001 Prototipo del módulo GUI para la plataforma de simulación galatea, Tesis de pregrado, Ingeniería de Sistemas, Universidad de Los Andes, Mérida, Venezuela.

Contreras L, 1999, Simulación de sistemas de transporte basada en agentes, Tesis de pregrado, Ingeniería de Sistemas, Universidad de Los Andes, Mérida, Venezuela.

Dávila J, 1997, Agents in Logic Programming, Tesis de PhD, Imperial College of Science, Technology and Medicine, Londres, Reino Unido.

Dávila J, 1999, Openlog: A logic programming language based on abduction, En *PPDP'99. Principles and Practice of Declarative Programming*, Lecture Notes in Computer Science, Vol. 1702/1999, pp. 278-293, Paris, France.

Dávila J, 2003, Actilog: An agent activation language. Lecture Notes in Computer Science, Vol. 2562/2003, pp. 194-207

Dávila J, 2011, Lógica Práctica y Aprendizaje Computacional. Editorial Académica Española. ISBN: 978-3-8465-6233-8.

Dávila J, Gómez E, Laffaille K, Tucci K y Uzcátegui M, 2005, Multiagent distributed simulation with galatea, En *DS-RT '05: Proceedings of the 9th IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pp. 165-170, Washington, DC, USA, IEEE Computer Society.

Dávila J y Tucci K, 2002, Towards a logic-based, multiagent simulation theory, AMSE Special Issue 2000. Association for the advancement of Modelling & Simulation techniques in Enterprises, pp 37-51.

Dávila J y Uzcátegui M, 2002, Galatea: A multi-agent simulation platform. AMSE Special Issue 2000. Association for the advancement of Modelling & Simulation techniques in Enterprises, pp. 52-67.

Dávila J y Uzcátegui M, 2004, Gloria: An agent's executable specification. Collegium Logicum, Kurt Gödel Society, Vol. VIII, pp 35-44.

Dávila J, Uzcátegui M y Tucci K, 2005, A multi-agent theory for simulation, En *MSO'2005. Fifth IASTED International Conference on Modelling, Simulation and Optimization*, pp. 285-290, Oranjestad, Aruba.

Dávila MA, 2003, Una plataforma cliente-servidor para simulación con modelos juego, Tesis de pregrado, Ingeniería de Sistemas, Universidad de Los Andes, Mérida, Venezuela.

Díaz L, 2002, Integración de sistemas de información geográfica a la plataforma de simulación galatea, Tesis de pregrado, Ingeniería de Sistemas, Universidad de Los Andes, Mérida, Venezuela.

Domingo C, 1988, GLIDER, a network oriented simulation language for continuous and discrete event simulation, En *International Conference on Mathematical Models*, pp. 11-14, Madras, India.

Domingo C, 1995, Proyecto GLIDER e-122-92 informe 1992-1995, CDCHT, Universidad de Los Andes, Mérida, Venezuela.

Domingo C y Hernández M, 1985, Ideas básicas del lenguaje GLIDER, Instituto de Estadística Aplicada en Computación, Universidad de Los Andes, Mérida, Venezuela.

Domingo C, Jiménez T, Ramírez V, Sananes M, Terán O y Tonella G, 1996, Simulation of structural change. En *8th European Simulation Symposium (ESS'96)*, Editores Bruzzone A y Kerckhoffs E, pp. 112-117, Genova, Italia.

Domingo C, Sananes M y Tonella G, 1995, Problem solving by structural simulation, En *IASTED International Conference*, Editor Hamza MH, Acta Press, pp. 463-367, Anaheim, USA.

Domingo C y Tonella G, 2000, Towards a theory of structural modeling, Structural Change and Economic Dynamics, Vol. 11, No. 1-2, pp. 209-225.

Domingo C, Tonella G, Herbert H, Hernández M, Sananes M y Silva J, 1993, Use of object oriented programming ideas in a new simulation language, En *Summer Computer Simulation Conference*, pp. 137-142, Boston, USA.

Domingo C, Tonella G y Terán O, 1996, Generating scenarios by structural change, En *6th Annual Conference AI, Simulation and Planning in High Autonomy Systems*, pp. 331-335, La Jolla, USA.

Domingo C, Tonella G y Sananes M, 1996, GLIDER Reference Manual, CeSiMo–IEAC, Universidad de Los Andes, Mérida, Venezuela.

Ferrer M, 2003, Diseño de un sistema basado en tecnologías multiagente y del conocimiento para modelado y simulación de organizaciones, Tesis de Maestría, Maestría en Computación, Universidad de Los Andes, Mérida, Venezuela.

GLIDER Development Group, 2000, GLIDER Reference Manual, Versión 5.0, CeSiMo - IEAC, Universidad de Los Andes, Mérida, Venezuela.

Gómez E, 2002, Desarrollo de un prototipo del módulo

agente para la plataforma de simulación galatea, Tesis de pregrado, Ingeniería de Sistemas, Universidad de Los Andes, Mérida, Venezuela.
Gómez E, 2005, Diseño e implementación de la plataforma distribuible para el simulador multiagente galatea, Tesis de Maestría, Maestría en Modelado y Simulación, Universidad de Los Andes. Mérida. Venezuela.
Grisolía T, 2003, Un modelo juego para simulación de la economía venezolana orientado a agentes, Tesis de pregrado, Ingeniería de Sistemas, Universidad de Los Andes, Mérida, Venezuela.
Laffaille K, 2005, Gspaces meta-modelo para simular espacios urbanos y arquitectonicos basado en galatea, Tesis de maestría, Maestría en Modelado y Simulación, Universidad de Los Andes, Mérida, Venezuela.
Laffaille K, Tucci K, Uzcátegui M y Dávila J, 2005, Gspaces a meta-model for simulating agent mobility in urban or architectonic design, En *MSO'2005. Fifth IASTED International Conference on Modelling, Simulation and Optimization*, pp. 303-306, Oranjestad, Aruba.
Laffaille K, Tucci K, Uzcátegui M, Dávila J y Nava M, 2008, Simulación de desalojos en la ciudad universitaria de Caracas. aportes para el desarrollo de aplicaciones específicas para mitigar desastres en centros urbanos estratégicos, TodoArquitectura Revista Digital, Vol. 42, pp. 1851-1244.
López J, 2003, Computación evolutiva en el alineamiento de secuencias de adn. Trabajo de ascenso a la categoría de asistente, LCAR, Universidad Nacional Experimental del Táchira, San Cristóbal, Venezuela.
Molina J, 2010, Modelado y simulación mediante sistemas multiagente de la evacuación de personas de edificaciones de varios pisos en presencia de obstáculos. Tesis de pregrado. Ingeniería de Sistemas, Universidad de Los Andes, Mérida, Venezuela.
Moreno N, Ablan M y Tonella G, 2002, SpaSim: A software to simulate cellular automata models, En *IEMSs 2002, First Biennial Meeting of the International Environmental Modeling and Software Society*, Lugano, Switzerland.
Moreno N, Quintero R, Ablan M, Barros R, Dávila J, Ramírez H, Tonella G y Acevedo M, 2007, Biocomplexity of deforestation in the caparo tropical forest reserve in venezuela: an integrated multi-agent and cellular automata model. Environmental Modelling & Software, Vol. 22, No. 5, pp. 664-673.
Palm F, 1999, Simulación combinada discreta/continua orientada a objetos: Diseño para un lenguaje GLIDER orientado a objetos, Tesis de Maestría, Maestría en Matemática Aplicada a la Ingeniería, Universidad de Los Andes, Mérida, Venezuela.
Pérez CJD, 2010, Desarrollo de modelos de tráfico vehicular en galatea, Tesis de pregrado, Departamento de Física, Universidad de Los Andes, Mérida, Venezuela.
Quintero R, Barros R, Dávila J, Moreno N, Tonella G y Ablan M, 2004, A model of the biocomplexity of deforestation in tropical forest: Caparo case study, En *IEMSs 2004. International Environmental Modelling and Software Society*, Editores: Pahl C, Schmidt S y Jakeman T, Osnabrueck, Germany.
Ramos A, 2006, Gide: un ambiente de desarrollo integrado para la plataforma de simulación galatea, Tesis de Maestría, Maestría en Modelado y Simulación, Universidad de Los Andes. Mérida. Venezuela.
Terán O, 1994, Simulación de cambios estructurales y análisis de escenarios, Tesis de Maestría, Maestría en Estadística Aplicada, Universidad de Los Andes. Mérida. Venezuela.
Tonella G, Domingo C, Sananes M y Tucci K, 1993, El lenguaje GLIDER y la computación orientada hacia objeto, En *XLIII Convención Anual ASOVAC*, Mérida, Venezuela.
Tucci K, 1993, Prototipo del compilador GLIDER en C++, Tesis de pregrado, Ingeniería de Sistemas, Universidad de Los Andes. Mérida. Venezuela.
Uzcátegui M, 2002, Diseño de la plataforma de simulación de sistemas multi-agentes galatea, Tesis de Maestría, Maestría en Computación, Universidad de Los Andes. Mérida, Venezuela.
Uzcátegui M, Tucci K y Moreno N, 2000, Ejemplo de simulación espacial en galatea, En *II Taller Interdisciplinario de Sistemas Complejos (TISC 2000)*, Margarita, Venezuela.
Vargas Y, 2003, Traductor de modelos de simulación galatea a código java, Tesis de pregrado, Ingeniería de Sistemas, Universidad de Los Andes, Mérida, Venezuela.
Vivas A, 2008, La modeloteca. biblioteca de metamodelos para simulación basados en galatea. Tesis de Maestría, Maestría en Modelado y Simulación, Universidad de Los Andes, Mérida, Venezuela.
Zeigler BP, 1976, Theory of modelling and simulation, Interscience. Jhon Wiley& Sons, New York.
Zeigler BP, 1990, Object-oriented simulation with hierarchical, Modular models (Intelligent agents and endomorphic systems), Academic Press, Inc (Harcourt Brace Jovanovich, Publishers), Boston-Sydney.
Zeigler BP, Praehofer H y Kim TG, 2000, Theory of Modelling and Simulation, Academic Press, second edition.

Recibido: 30 de junio de 2011
Revisado: 13 de septiembre de 2011