GALATEA: Plataforma de Simulación de Sistemas Multi-Agentes

GALATEA: Multi-Agents Simulation Platform

Mayerlin Uzcategui[12*], Kay Tucci[12], Jacinto Dávila[2],
[1]SUMA, Facultad de Ciencias, ULA.
[2]CeSiMo, Facultad de Ingeniería, ULA,
Mérida,5101, Venezuela.
*maye@ula.ve

Resumen

GALATEA (GLIDER with Autonomous Logic-based Agents, TEmporal reasoning and Abduction), es un software de simulación que propone integrar, en una misma plataforma computacional, las herramientas conceptuales y concretas para simulación de: eventos discretos, sistemas continuos y sistemas multi-agentes; de forma distribuida e interactiva. Es decir, proporcionar un ambiente de modelado y simulación de agentes que permite explorar alternativas para su integración con un formalismo general de modelado y simulación de eventos discretos.

Palabras Clave: Simulación, Sistemas Multi-Agentes

Abstract

GALATEA (GLIDER with Autonomous Logic-based Agents, TEmporal reasoning and Abduction), is a simulation software proposed to integrate the conceptual and concreted tools in a unique simulation platform in order to simulate: discrete events, continuous, combined and multi-agents systems; in distributed and interactive way. We offer a multi-agents modeling and simulation environment to study different scenarios to integrate agents with discrete event model and simulation formalism.

Key words: Simulation, Multi-Agent Systems.

1 Introducción

La simulación ha sido utilizada para conocer el comportamiento de sistemas o procesos y de esta forma comprender su comportamiento y realizar inferencias sobre el mismo. Tradicionalmente, en sistemas que tienen entidades "pensantes" dichas entidades son modeladas, en primera aproximación, por medio de distribuciones estadísticas. Tal es el caso del modelado clásico de una cola de un banco, donde el tipo de operación bancaria a realizar por un cliente se determina por medio de una distribución de probabilidad como una manera de manejar la enorme complejidad de una caracterización detallada del comportamiento humano. La otra manera es utilizar estructuras de decisiones para modelar el comportamiento de cada entidad pensante. Esta aproximación fue tradicionalmente considerada inabordable. En este caso, cada condición de la estructura de decisión a su vez puede depender de una función de probabilidad. Una nueva aproximación para la representación de entidades pensantes consiste en utilizar algún método de Inteligencia Artificial, como la llamada Tecnología de Agentes (Information Society Technologies, 2005). Este tipo de representación permite modelar sistemas a través de entidades autónomas con conocimiento independiente a la unidad de cómputo utilizado como mecanismo de razonamiento. Además, cada agente puede "percibir su ambiente, a través de sensores y actuar sobre su entorno a través de efectores" (Russell y Norvig, 1995). Nuestro trabajo está enfocado a esta última aproximación.

Para ello hemos desarrollado una teoría que pretende servir como la especificación formal de un Sistema Multi-Agente que podría ser simulado por GALATEA. Con este proyecto estamos extendiendo un lenguaje de simulación de larga trayectoria, GLIDER (Domingo, 1988; Domingo y col., 1996) con las abstracciones necesarias para permitir a los modelistas representar sistemas en los que se

desenvuelven entidades pensantes, los agentes, que percibe y actúan sobre su ambiente.

Uno de los objetivos fundamentales de GALATEA es completar el enriquecimiento de la sintaxis y la semántica GLIDER para permitir la descripción de agentes y de sistemas multi-agentes. Los agentes representan, en el modelo de simulación, a aquellas entidades que en el sistema modelado pueden percibir su ambiente, tienen sus propias metas y creencias y actúan, siguiendo esas creencias, para alcanzar esas metas, posiblemente cambiando el ambiente durante ese proceso.

Enriquecer a GLIDER de esa manera requiere más que un conjunto adicional, ad hoc, de constructos lingüísticos. Hemos extendido la teoría de simulación para explicar la conducta de esos objetos especializados: los agentes, bastándonos en las formalizaciones de la conducta de agentes presentadas los trabajos en Inteligencia Artificial que citamos al comienzo de esta sección.

GALATEA esta concebido como una colección estructurada de objetos que intercambian mensajes. El intercambio y procesamiento de mensajes está asociado a la ocurrencia de eventos que corresponden a cambios en ciertas sub-estructuras, llamadas nodos. Modelar un sistema consiste en en describir una red de nodos, cada uno con una descripción de su conducta y de cómo, cuándo y con cuáles otros nodos intercambiará mensajes.

Nuestro objetivo final es disponer de una familia de lenguajes, apoyada por una única plataforma de computación, que nos permita modelar y simular sistemas multi-agentes. Disponer de lenguajes de diversa naturaleza en la misma plataforma es una contribución importante a un enfoque interdisciplinario para modelado y simulación de sistemas.

2 Descripción de la Plataforma

GALATEA integra conceptos y herramientas que permiten simular sistemas bajo los enfoques distribuido, interactivo, continuo, discreto y combinado. Además, incorporamos soporte para el modelado y la simulación eficiente de sistemas multi-agentes. GALATEA este conformada por:

1. Una familia de lenguajes, y sus respectivos compiladores o interpretadores, que permiten simular sistemas multi-agentes. Los lenguajes son: Galatea-Java (una versión del lenguaje original de Glider, pero con sintaxis Java en lugar de la correspondiente Pascal que se usaba en Glider), Openlog/Actilog, una colección de lenguajes de programación lógica de agentes (Dávila, 2003; Dávila, 1999) y un lenguaje elemental para ecuaciones diferenciales (para modelos continuos).
2. Un simulador DEVS (extendido para lidiar con agentes).
3. Un ambiente para la construcción de modelos.
4. Una colección de bibliotecas de apoyo con METAMODELOS, plantillas genéricas que pueden ser convertida en una gama de modelos específicos. El ejemplo en la sección siguiente muestra una de ellas (gSpace).
5. Una colección de modelos de ejemplos reunidos en la MODELOTECA.

La propia plataforma de GALATEA ha sido desarrollada en Java (Sun Microsystem Inc, 1994) y puede ser ejecutada desde diversos ambientes que soportan aplicaciones Java (la hemos probado con la plataforma de Sun Microsystems, SableVM y GNU Java Classpath).

En (Dávila, 1997) se muestra una especificación detallada de los lenguajes. En (Uzcategui, 2002) se muestran los detalles de diseño e implementación de funcional del simulador. Y en (Ramos, 2006) se muestran los detalles de diseño e implementación del prototipo funcional para la interfaz gráfica del ambiente de desarrollo que llamamos GIDE.

En GALATEA se propone una reformulación de la manera tradicional de manejar la relación entre los componentes de simulación con miras a la simulación interactiva. Por ello es necesario pensar en varios programas ejecutándose en computadoras de diverso hardware y distribuidas geográficamente.

Como mencionamos anteriormente, los adelantos en Simulación Interactiva han permitido el desarrollo de un marco de referencia estándar que facilita la integración de múltiples componentes de simulación. Este marco de referencia, denominado HLA (DoD DMSO, 1995), nos sirve como referencia para el diseño de la plataforma.

Fig. 1. Arquitectura de la plataforma de simulación GALATEA

La Fig. 1 muestra la estructura de la federación HLA, usada en GALATEA, dividido en sus componentes funcionales principales: La Simulación, o el programa de simulación en ejecución, compuesto por varios objetos llamados federados, con los cuáles se puede conectar con otras simulación o con otros tipos de estructuras (pasivas, como las bases de datos, o activas, como los agentes).

La interacción entre esos federados es programada a través de lo que llamamos la Interfaz, un componente que se integra en la Simulación, pero que contiene un código específico para sistema simulado (y que se anota como parte del texto fuente del modelo correspondiente, en la sección Interface).

El segundo componente funcional de la arquitectura es la Infraestructura de Tiempo de Ejecución (RTI: *Runtime Infrastructure*). Esta infraestructura proporciona un grupo de servicios de propósito general que soportan las interacciones entre los componentes de simulación. Todas las interacciones entre los componentes de la simulación se llevan a cabo a través del RTI. En GALATEA el RTI es provisto por los servicios subyacentes en Java, para la versión mono-procesador. Pero también tenemos una versión de GALATEA que opera sobre una plataforma distribuida sobre FIPAOS (FIPA-OS, 2003).

El tercer componente es la Interfaz General. Esta interfaz, independiente de su aplicación, mantiene una manera estándar de relacionar los componentes de simulación con el RTI. La interfaz es independiente de los requisitos para el modelado y del intercambio de datos entre los componentes de la simulación. En particular, en nuestro caso, la interfaz además de controlar la sincronización del tiempo global, debe proporcionar el conjunto de servicios necesarios para que el intercambio de datos entre los agentes y la simulación a la que están asociados fluya.

A continuación, describimos los componentes de la arquitectura que proponemos en la Fig. 1: el simulador y los agentes. Esta descripción incluye los detalles que han sido tomados en cuenta en la implementación.

3 El Simulador y los Agentes

Como mencionamos anteriormente el simulador de la plataforma GALATEA es una extensión del simulador GLIDER. La simulación de un sistema consiste en generar y activar una sucesión de eventos en el ambiente simulado. Por consiguiente, el simulador, esencialmente, activa dichos eventos y ejecuta las piezas de código asociadas con cada uno. Se puede argumentar que el disparo o activación de un evento consiste, precisamente, en la ejecución de esa pieza de código. Sin embargo, hay casos que no encajan con esta definición. Así, en GLIDER un evento es la activación de un nodo que puede o no implicar la ejecución de su código.

Una simulación de un modelo GLIDER es la activación de los eventos relacionados a los nodos y la posterior revisión de la red, buscando el código a ser ejecutado y probando sus pre-condiciones estructurales. Si estas pre-condiciones se alcanzan, el código debe ejecutarse. Para garantizar que los efectos (cambios) inducidos en cada evento se propaguen a través de todo el sistema, la activación de un nodo (la ocurrencia de un evento identificado con el nombre del nodo) puede activar nuevos eventos y además puede implicar cambios en el estado global del sistema o pases de mensajes.

Así, una revisión de la red conduce a la evolución de la simulación del sistema. Cuando un nodo se activa, su código se ejecuta y se inicia la revisión del resto de la red. Durante la revisión, aquellos nodos que no tienen una lista externa (EL, lista de espera para los mensajes, o cuya EL está vacía se ignoran. En estos nodos no se reflejan cambios debido a que no contienen ninguna entidad.

El proceso de revisión continúa visitando nodos (reflejando los cambios ocurridos en aquellos nodos que contienen entidades) hasta que se alcanza el nodo donde se inicio la revisión (el nodo activado) luego de completar un ciclo completo sin movimientos de mensajes.

Una vez que la revisión termina, el simulador busca otra entrada de la lista de evento futuros (FEL), para determinar el próximo evento en ser ejecutado. La FEL es, por supuesto, una colección de eventos: etiquetas de los nodos ordenadas según el tiempo fijado para su ejecución.

GALATEA, conserva esta estrategia general, excepto que, basado en la especificación proporcionada por la teoría de simulación multi-agentes (Dávila y col., 2005), proporciona la ejecución de un motor de inferencia asociado a cada agente. Todos estos motores se ejecutan concurrentemente con el simulador principal y su ejecución se intercala cuidadosamente, para intercambiar información y permitir la simulación de un sistema multi-agentes en forma efectiva. Los motores de inferencia proporcionan la simulación del ciclo percibir-razonar-actuar de cada agente mientras el simulador controla la evolución física del sistema.

Este enfoque en el que la simulación de un sistema se realiza a partir de los componentes distribuidos no es una idea nueva. De hecho, como mencionamos anteriormente, estamos aprovechándonos de una especificación existente para simulación distribuida, incorporando la simulación en una "federación". El estándar HLA especifica las condiciones mínimas que un conjunto de componentes debe cumplir para ser convertidos en una federación. En nuestro caso, el simulador y cada uno de los agentes es un federado. Ellos se agrupan a en una federación multi-agentes que usa una estrategia simple, centralizada, de manejo de tiempo para controlar su propia evolución. En particular, en GALATEA los intercambios de información son siempre actualizaciones de la lista de eventos futuros, FEL, (o conjunto de influencias como las llamamos en otras partes). Los agentes incorporan sus intenciones (influencias o acciones que quieren ejecutar) en la FEL y el simulador decide el desenlace de las mismas. De igual forma, el simulador determina y programa las percepciones apropiadas para cada agente en la FEL, y cada agente "decide" lo que quiere ver a través de los métodos de la interfaz.

Este algoritmo general para GALATEA, nos ha permitido especificar los detalles que deben regir el comportamiento del simulador. A continuación daremos una breve descripción del comportamiento del simulador.

El diagrama de secuencia de la Fig. 2 corresponde al proceso de ejecución en la plataforma GALATEA de una simulación que incluye agentes. Este diagrama muestra la interacción a lo largo del tiempo entre los objetos que participan, destacándose las actividades recursivas de activación y recorrido de la red de nodos y la activación de los agentes que participan en el sistema. Los componentes asociados al proceso de simulación son:

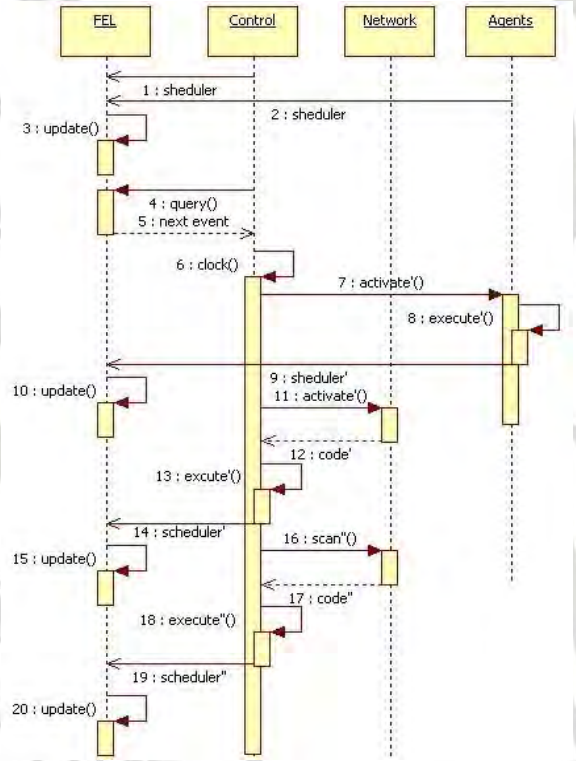


Fig. 2 Diagrama de secuencia para GALATEA

FEL: Contiene, cronológicamente ordenados, los eventos que deben "tener lugar" en la simulación de un sistema. Realiza las siguientes operaciones:

- incluye en la lista de eventos los eventos que recibe
- si se le solicita, entrega el evento que debe ser procesado

Control: Representa el controlador del proceso de simulación realizando las siguientes actividades:

- solicita la programación de los eventos que permiten el inicio del proceso de simulación. Este procesamiento incluye la activación de la red de nodos y de los agentes
- solicita el próximo evento y controla que se procese el evento
- programa los nuevos eventos

Network: Representa el conjunto de subsistemas (nodos) presentes en el sistema a simular y por lo tanto, tiene como función controlar la interacción y activación de los nodos. Debe cumplir las especificaciones y debe llevar a cabo las siguientes actividades:

- controla la activación del nodo actual
- controla la revisión de la red
- controla la activación de los agentes

Ginterface: Sirve de intermediario entre los agentes y el simulador.

Agents: Representa al grupo de agentes presentes en el sistema. Tiene la función de controlar las actividades propias del agente:

- razonar. Implica asimilar percepciones, observaciones, recuerdos, metas, creencias y preferencias para tomar decisiones sobre que afectarán su conducta
- observar. Percibir el ambiente
- actuar. Intentar modificar el ambiente de acuerdo a las decisiones tomadas

En el diagrama de colaboración de la figura se muestran los intercambios de datos disparados por los eventos que han sido activados en el sistema. En este diagrama, se evidencia que el componente Control es quien tiene a su cargo el simulador, y ordena las actividades que se llevan a cabo, mientras que el componente GInterface sirve de intermediario entre el simulador y los agentes. En el diagrama de la **Fig. 3** se destacan las siguientes colaboraciones:

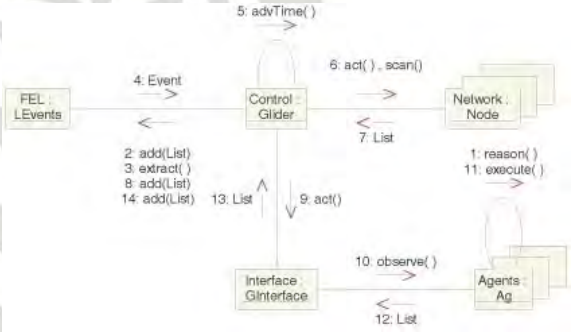


Fig. 3 Diagrama de colaboración para GALATEA

1. Se inicia el proceso de razonamiento de los agentes invocando el método reason(). Este proceso se mantiene activo durante la ejecución de la simulación y sólo es interrumpido momentáneamente mientras el agente actúa.
2. Solicita la incorporación de los eventos que inician el proceso de simulación haciendo uso del método add().
3. Solicita el evento a procesar a través del método extract().
4. Recibe un objeto tipo Event.
5. Extrae del evento el tiempo de activación y actualiza el reloj del simulador a través del método advTime().
6. Si el evento esta asociado a un agente extrae la lista de acciones que deben ejecutarse en este momento y las ejecuta, pero si el evento esta

asociado a un nodo se solicita la activación de dicho nodo a través del método act() y el recorrido de la red de nodos a través del método scan().

7. Recibe objetos List que contienen los eventos que deben programarse a causa de la activación del nodo y del recorrido de la red.
8. Solicita la incorporación de los nuevos eventos haciendo uso del método add().
9. Solicita la activación de los agentes a través del método act().
10. **GInterface**, a solicitud de **Control** invoca en cada agente el método observe().
11. **GInterface** desencadena en cada uno de los agentes la ejecución del método execute() que determina las acciones que ha de intentar llevar a cabo el agente de acuerdo al razonamiento alcanzado hasta el momento.
12. **GInterface** recibe de cada uno de los agentes que se ha activado un objeto List que contiene las influencias que deben programarse a causa del intento de los agentes de modificar su ambiente.
13. Recibe objetos List que contienen los eventos que deben programarse a causa de la activación de los agentes.
14. Solicita la incorporación de los nuevos eventos haciendo uso del método add().

Cabe destacar que el paso 6 refleja el proceso recursivo de revisión de la red, que se repite secuencialmente hasta asegurar que se han reflejado los cambios causados en el sistema por la activación del nodo.

Por otro lado, el proceso de activación de los agentes, que corresponde a los pasos 9-14, se ejecuta simultáneamente en todos los agentes presentes en el sistema y está fuertemente vinculado al proceso de razonamiento de cada agente, presentado en el paso 11, el cual esta ejecutándose continuamente y es sólo interrumpido un momento mientras se realiza la activación del agente.

El ciclo de la simulación se inicia en el paso 3 y abarca el resto de pasos, se repite hasta finalizar la simulación: cuando se alcanza el tiempo determinado para simular, cuando no quedan eventos o cuando el simulista lo indique.

Con esta descripción del comportamiento culminamos la presentación del simulador. Para la descripción formal de los agentes referimos al lector a (Uzcategui, 2002). En la siguiente sección mostraremos un ejemplo que los ilustra claramente. Para hacer que los agentes hagan lo que se espere de ellos, es preciso codificarles conocimiento operacional. Esto, a su vez, requiere de lenguajes de programación para agentes y por supuesto en nuestro caso también necesitamos un lenguaje que nos permita incorporarlos en la plataforma de simulación. Todos esos elementos han sido cubiertos en la implementación de GALATEA.

4 Estructura de un modelo GALATEA

La figura Fig. 4 muestras la estructura de un programa de GALATEA, el cual consta de seis secciones básicas:

	Encabezado
NETWORK	Descripción de la Red
AGENTS	Descripción de los Agentes
INTERFACE	Descripción de las relaciones entre los agentes y el ambiente
INIT	Iniciación de las variables y estructuras usadas en la simulación
DECL	Declaración de variables
END.	

Fig. 4 Estructura de un programa GALATEA

Encabezado: Esta sección contiene el título y la descripción del sistema, así como algunos comentarios.

Red: Esta sección, al igual que en GLIDER, empieza con la etiqueta NETWORK. Es donde se describen los nodos, las relaciones entre ellos y el código que guía la simulación. Cada nodo tiene dos partes: un título donde se definen el nombre, el tipo, la multiplicidad, los sucesores y las variables locales y 2) un código compuesto de constructos Java y GALATEA.

Agentes: Esta sección empieza con la etiqueta AGENTS y contiene descripción de cada tipo del agente.

Interface: Esta sección empieza con la etiqueta INTERFACE. En esta sección se incluye la descripción de las relaciones entre los agentes y el resto del sistema. Esta sección contiene el código Java que describe el comportamiento del simulador para permitir a los agentes percibir y actuar.

Iniciación: Esta sección, al igual que en GLIDER, se indica con la etiqueta INIT, aquí se realiza la asignación de valores que se usarán a tiempo de ejecución de la simulación, como

- Los valores iniciales para los arreglos.
- Los valores iniciales para las capacidades de los nodos tipo
- Valores de funciones multivaluadas.
- Los valores de parámetros en mapas de frecuencia.
- Valores de arreglos iniciados desde las bases de datos.
- Activación inicial de nodos.

Declaración: Esta sección, al igual que en GLIDER, empieza con la etiqueta DECL y según el tipo de la declaración que se lleva a cabo es necesario usar la etiqueta:

- TYPE: Los tipos de datos definidos por los usuarios
- CONST: Constantes.
- VAR: Variables.
- NODES: Nodos definidos por los usuarios.

- MESSAGES: Mensajes.
- GFUNCTION: Funciones multivaluadas definidas por los usuarios.
- TABLES: Mapas de frecuencia.
- DBTABLES: Mapas provenientes de bases de datos.
- PROCEDURES: Procedimientos y funciones definidas por los usuarios.
- STATISTICS: Aplicación de estadísticas para los nodos y variables.

Como todo programa GLIDER debe culminar con la etiqueta END.

5 Ejemplo de Simulación de Sistemas Multi-Agentes-Móviles

Hoy en día, es posible desarrollar modelos computacionales usando técnicas de modelado y simulación que permiten comprobar la factibilidad y la eficiencia de los diseños arquitectónicos para la evacuación de agentes móviles en caso de catástrofe. Existen dos paradigmas clásicos para estos sistemas, el primero usa autómatas celulares (Biham y col., 1992; Nagel y Herrmann, 1993; Wolf y col., 1996) donde el espacio, los estados y el movimiento son disretos. El segundo usa espacio continuos, estados y reglas de movimiento (Helbing y col., 2000).

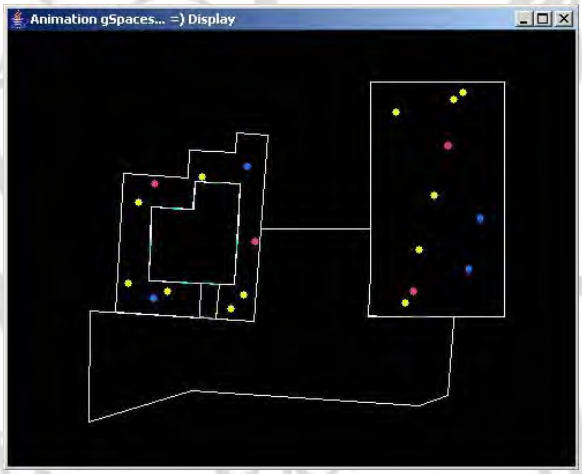
En la Fig. 5 Un modelo con agentes móviles, mostramos una versión simplificada del código de un ejemplo de simulación de desalojos en espacios urbanos, mostrado en detalle en (Dávila, Gómez y col., 2005), en el que se describe el comportamiento de un grupo de personas y un grupo de cucarachas dentro de un cuarto que tiene salida a un espacio abierto.

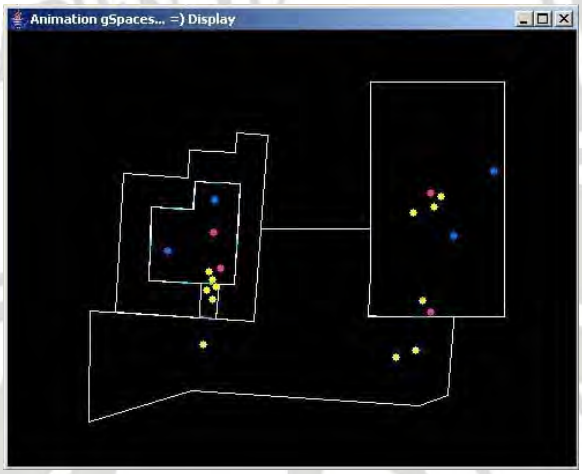
```
Title
Un espacio simple con dos tipos de agentes moviles
Network
 cucaracha (I){
  setAgent(cerebroCucaracha);
  x = unif(0,6);
  y = unif(0,4);
  vx = unif(-0.1,0.1);
  vy = unif(-0.1,0.1);
  sendto(cuarto);
 }
 persona (I){
  setAgent(cerebroPersona);
  vx = unif(0.2,0.5);
  vy = 0;
  sendto(cuarto.salida);
 }
 cuarto (S){
  it(0.1);
  resolution(1);
  addWall(0,0,6,0);
  addWall(6,0,6,4)
  addWall(6,4,0,4)
  addWall(0,4,0,1);
  addDoor(0,0,0,1,salida);
  move(){
   Cell c;
   double densidad;

   if (mess.getName == "persona" ){
    c = cuarto.getCell(mess);
    densidad =
    c.getnAg("persona")/c.getArea();
        velocidad =
    vxNew*vxNew + vyNew*vyNew;
        vmax = 4/Math.pow(densidad,3);
        if (velocidad > vmax) {
         vxNew=vxNew*vMax/velocidad;
         vyNew=vyNew*vMax/velocidad;
        }
   }
  }

  salida (E){
  }
Agents
  cerebroCucaracha{
  Goals ....
  Beliefs ....
  }
  cerebroPersona{
  Goals ....
  Beliefs ....
  }
Interface
  desplazamiento(vx,vy){
   cuarto.move();
  }
  Cucaracha.perceive();
  Persona.perceive();
End
```

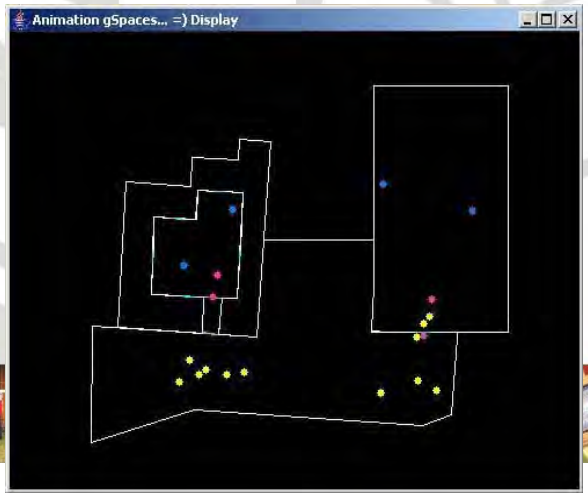
Fig. 5 Un modelo con agentes móviles

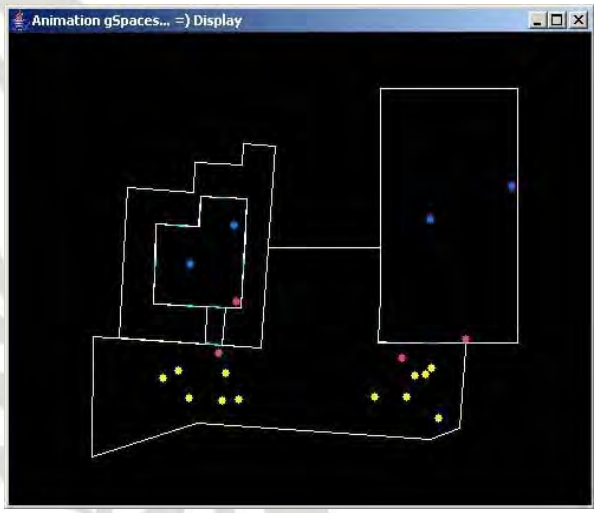
En este ejemplo usaremos la librería gSpace de GALATEA, que combina las características de los paradigmas discreto y continuo en un módulo hibrido y flexible para el desarrollo y simulación de sistemas con entidades que se mueven. En gSpace los espacios arquitectónicos o urbanos se dividen en subredes tipo Space y los agentes móviles son mensajes que tienen asociados atributos como: posición (x, y), velocidad (vx, vy).

Con gSpace (Laffaille y col, 2005), GALATEA puede producir salidas gráficas animadas para cada instante de tiempo de simulación, construídas sobre una representación visual como las usadas por las herramientas CAD, tales como las siguientes:









6 Conclusiones y Futuros Trabajos

GALATEA es una plataforma de simulación de sistemas multi-agentes que permite describir sistemas haciendo uso del modelado discreto, continuo, combinado, distribuido e interactivo.

En este documento hemos resumido la teoría matemática que soporta la plataforma GALATEA, la cual esta siendo usada como la especificación formal de la implementación de GALATEA.

Además presentamos la estructura general de la plataforma GALATEA para simulación de sistemas multi-agentes. GALATEA como una extensión DEVS de la plataforma GLIDER que como su predecesor soporta el modelado y la simulación de propósito general. Al introducir agentes GALATEA también soporta descripciones más expresivas de sistemas en donde la toma de decisión, la actuación y los cambios en las estructuras son los elementos clave. Nosotros pretendemos completar el desarrollo de GALATEA, implementando los intérpretes lógicos para cada uno de los lenguajes lógicos usados para programar a los agentes.

7 Agradecimientos

Queremos agradecer a los profesores Carlos Domingo, Marta Sananes por su colaboración con este trabajo y queremos reconocer el apoyo logístico y económico




brindado por las dependencias de la Universidad de Los Andes: CeSiMo y SUMA. Así como también el co-financiamiento de parte del CDCHT-ULA bajo el proyecto I-886-05-02-A.

Referencias

Biham O, Middleton AA, Levine D. 1992. Self-organization and dynamical transition in traffic-flow models. Phys. Rev. A. 46.

Dávila J. 2003. Actilog: An agent activation language.In PADL2003. Practical Aspects of Declarative Languages, 5th International Symposium. 2562. LNCS. New Orleans, USA.

Dávila, JA. 1997. Agents in logic programming. Ph.D. dissertation. Imperial College of Science, Technology and Medicine. London, UK.

Dávila JA. 1999. Openlog: A logic programming language based on abduction. In PPDP'99 International Conference on Principles and Practice of Declarative Programming. Lecture Notes in Computer Science 1702. Springer. Paris, France.

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

Dávila J, Gómez E, Laffaille K, Tucci K, Uzcátegui M. 2005. Multiagent distributed simulation with galatea. In The 9-th IEEE Internacional Symposium on Distributed Simulation and Real Time Applications (DS-RT'2005). 165–170pp. IEEE. Montreal, Canada.

Dávila J, Uzcátegui M, Tucci K. 2005. A multi-agent theory for simulation. In The Fifth IASTED International Conference on Modelling, Simulation, and Optimization (MSO'2005). 285–290pp. The International Association of Science and Technology for Development (IASTED). Acta Press. Oranjestad, Aruba.

DoD DMSO. 1995. High Level Architecture (HLA). http://www.dmso.mil.

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

Domingo C, Tonella G, Sananes M. 1996. GLIDER Reference Manual. CESIMO–IEAC. Universidad de Los Andes. Mérida, Venezuela. CeSiMo IT-9608.

FIPA-OS. 2003. FIPA Open Source. http://fipa-os.sourceforge.net/.

GALATEA Group. Galatea, plataforma de simulación de sistemas multiagentes. http://galatea.sourceforge.net/.

Helbing D, Farkas I, Vicsek T. 2000. Simulating dynamical features of escape panic. Nature. 407. 487–490pp.

Information Society Technologies. 2005. Agentlink - european co-ordination action for agent-based computing, http://www.agentlink.org/.

Nagel K, Herrmann JH. 1993. Deterministic models for traffic jams. Physica A. 199.

Laffaille K, Tucci K Uzcátegui M and Dávila J. gSpace: a meta-model for simulating agent mobility in urban or arquitectonic designs. The Fifth IASTED Internacional Conference on Modelling, Simulation and Optimization (MSO 2005), 303-306pp. Oranjestad, Aruba. Agosto, 2005.

Ramos, A. 2006. GIDE. Un Ambiente de Desarrollo Integrado para la Plataforma de simulación GALATEA. Tesis de Maestría. Postgrado en Modelado y Simulación, Universidad de Los Andes. Mérida, Venezuela.

Russell SJ, Norvig P. 1995. Artificial Intelligence: A Modern Approach. Prentice Hall, Inc.

Sun Microsystems Inc. 1994. The source for the java technology. http://java.sun.com.

Uzcátegui, M. 2002. Diseño de la plataforma de simulación de sistemas multi-agentes GALATEA. Tesis de Maestría. Postgrado en Computación, Universidad de Los Andes. Mérida, Venezuela.

Wolf DE, Schreckenberg M, Bachem AE. 1996. Traffic and Granular Flow. World Scientific.