Simulación basada en Agentes Reactivos

Andrés Díaz Pace – Federico Trilnik - Marcelo Campo - Alejandro Clausse
Universidad Nacional del Centro de la Provincia de Buenos Aires
Facultad de Ciencias Exactas - ISISTAN
Campus Universitario – Paraje Arroyo Seco – (7000) Tandil – Buenos Aires – Argentina
Email: { adiaz, ftrilnik, mcampo, clausse } @exa.unicen.edu.ar

Abstract

Los métodos de modelado analítico clásicos definen un modelo matemático (determinístico o estocástico) que describe la dinámica del sistema, y luego existe una resolución numérica del conjunto de ecuaciones planteado. Este enfoque, da lógicamente, mayor importancia a la precisión en los resultados y a la eficiencia en tiempo. Pero en general, no son adecuadamente atendidas cuestiones relativas a como diseñar una arquitectura de simulación, que pueda ser adaptada ante posibles cambios en la especificación del problema, y que sus componentes puedan ser reutilizados en otros modelos de simulación similares.
En este trabajo se propone un *framework* (Bubble), para construir modelos de simulación de procesos colectivos utilizando un enfoque multiagente. El objetivo del *framework* es permitir definir y organizar conjuntos de agentes reactivos, que interaccionen entre ellos a través de eventos, dando lugar a determinados comportamientos colectivos que tienen aplicación en la simulación de procesos. Se ha desarrollado como ejemplo base, una simulación de flujo de fluidos en 2 fases, donde se analizan las perspectivas del enfoque.

1. Simulación e Ingeniería de Software

La simulación aparece como una alternativa para el estudio de distintas situaciones: problemas de optimización, comportamientos transitorios en modelos dinámicos, problemas de la vida real que contienen ciertos elementos de incerteza, etc. En este campo han surgido distintos enfoques para llevar a cabo estas simulaciones, que utilizan la computadora como herramienta de cálculo. Estos enfoques se han ocupado principalmente de cuestiones tales como eficiencia en tiempo y precisión en los resultados. Sin embargo, no se ha dado tanta importancia a aspectos relacionados con el proceso de diseño y escritura de software de simulación.

En la práctica, existen muchos problemas (por ejemplo el funcionamiento de una planta industrial con flujo de fluídos) que involucran gran cantidad de parámetros, restricciones y optimizaciones, que varían con el tiempo, lo cuál lleva a problemas difíciles de manejar con las técnicas de simulación clásicas. Por otra parte, existen situaciones donde es costoso, peligroso y hasta imposible realizar experimentos sobre sistemas reales. Por esta razón, se hace necesario un estudio de alternativas de simulación para estos procesos, a fin de evaluar resultados, tomar decisiones y llevar a cabo acciones de control.

Para el modelado, es común que en primera instancia se haga una abstracción bastante simple del problema, que luego se va complicando con el agregado de nuevas características. No obstante, no resulta sencillo diseñar una arquitectura de software, que pueda ser adaptada ante posibles cambios en la especificación del problema, y que sus componentes puedan ser reutilizados en otros modelos de simulación similares.

A menudo se emplea una gran cantidad de esfuerzo construyendo software experimental para simular distintos fenómenos, pero el producto resulta pobre desde el punto de vista de la

producción de software reusable y flexible. Estos modelos computacionales "hechos a medida", suelen presentar problemas tales como:

- Duplicación del esfuerzo de desarrollo.
- Software muy específico, donde se complica realizar variaciones en los requerimientos.
- Dificultad para comparar resultados y reproducir situaciones.
- Complejidad de uso del software, para usuarios sin conocimientos específicos.
- Problemas para realizar un diseño de alto nivel de los componentes esenciales del modelo.
- Decisiones de diseño que no están bien documentadas, lo cual perjudica la comunicación entre grupos de trabajo.

En los últimos años, se ha propuesto una nueva variante de simulación computacional denominada simulación multiagente [Drogoul92], que se encuentra relacionada con el surgimiento de nuevas tecnologías en ingeniería de software, como la orientación a objetos y el concepto de agentes [Meyer97][Shoham93]. También resulta valioso desde el punto de vista de simulación, la introducción del concepto de "vida artificial" [Langton91][Gutowitz95].

La combinación de estas tecnologías conduce naturalmente a la construcción de un *framework* para simulación, como una alternativa para solucionar los problemas descriptos anteriormente. Un *framework* permite reusar tanto diseño como código [Johnson97].

En este trabajo se presenta un *framework* (Bubble) implementado en Java, para construir modelos de simulación de sistemas complejos. Los objetivos centrales de Bubble son la provisión de un soporte reusable que permita:

- Realizar distintos tipos de simulaciones, variando la geometría del modelo.
- Proporcionar extensibilidad y flexibilidad, para permitir agregar fácilmente nuevos componentes y remover o mejorar los ya existentes.
- Contar con componentes reusables en otros modelos de simulación similares.
- Modelar un determinado problema a diferentes niveles de abstracción.
- Integrar al modelo de simulación, herramientas de visualización, recolección de datos, inspección, etc.

2. Bubble: Un framework para simulación utilizando Agentes Reactivos

El objetivo de Bubble es permitir definir y organizar conjuntos de agentes reactivos para modelar un problema complejo, y obtener una simulación cuya evolución se obtenga a partir de la interacción entre estos agentes.

Los agentes reactivos constituyen las unidades básicas de la simulación, y se caracterizan por un estado interno y un conjunto de tareas a ejecutar. También pueden existir agentes que contengan grupos de agentes en su interior, con lo cuál pueden realizarse modelamientos a distintos niveles de abstracción utilizando estructuras jerárquicas de agentes.

El mecanismo de interacción entre los agentes reactivos se realiza a través de eventos, que los agentes emiten y/o reciben. Los agentes poseen sensores asociados, que se pueden registrar por los distintos eventos de interés que deseen recibir. Existe un mecanismo de invocación implícita, que permite que los agentes sean notificados de los eventos a través de sus sensores [Shaw96].

El comportamiento de los agentes se define a través de un conjunto de tareas, cada una de las cuáles se organiza según un esquema precondición-acción. Una tarea define una serie de acciones a ejecutar por un agente cuando se satisfacen ciertas precondiciones (relacionadas con el estado interno del agente y los eventos recibidos) . También se pueden definir tareas compuestas que encapsulen una determinada red de tareas ya existentes, esto permite componer y relacionar diferentes comportamientos.

El *framework* permite también, el agregado de agentes encargados de otros aspectos de la simulación, como por ejemplo visualización y recolección de información estadística.

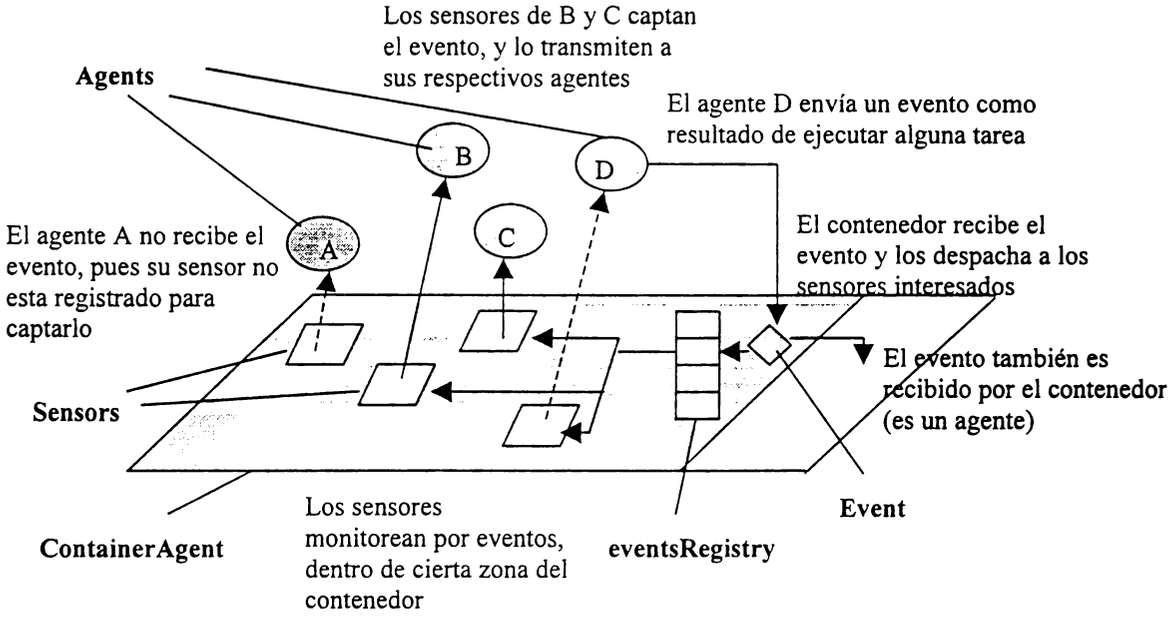

Fig. 1 - Esquema conceptual de la arquitectura

3. Una aplicación: Flujo de fluidos en 2 fases

A fin de aplicar las ideas del *framework*, se ha desarrollado como ejemplo base, una simulación de flujo de fluidos en 2 fases [Herrero95]. En este problema, el modelo está compuesto por un fluido continuo, el cuál contiene un fluido disperso en su interior. Debido a las propiedades de estos fluidos (como por ejemplo la densidad), el fluido disperso forma burbujas.

Para hacer evolucionar la simulación y estudiar distintas propiedades estadísticas del sistema, se define un conjunto de comportamientos para las burbujas: desplazamientos, coalescencias y rupturas. El resultado esperado será que las burbujas se moverán aleatoriamente, realizarán coalescencias y rupturas, hasta que el sistema alcance un estado de equilibrio. Se pueden llegar a modificar o complicar las reglas de comportamiento del modelo de simulación, agregando más parámetros y/o ecuaciones, a fin de capturar más acertadamente la física esencial involucrada en este tipo de problemas.

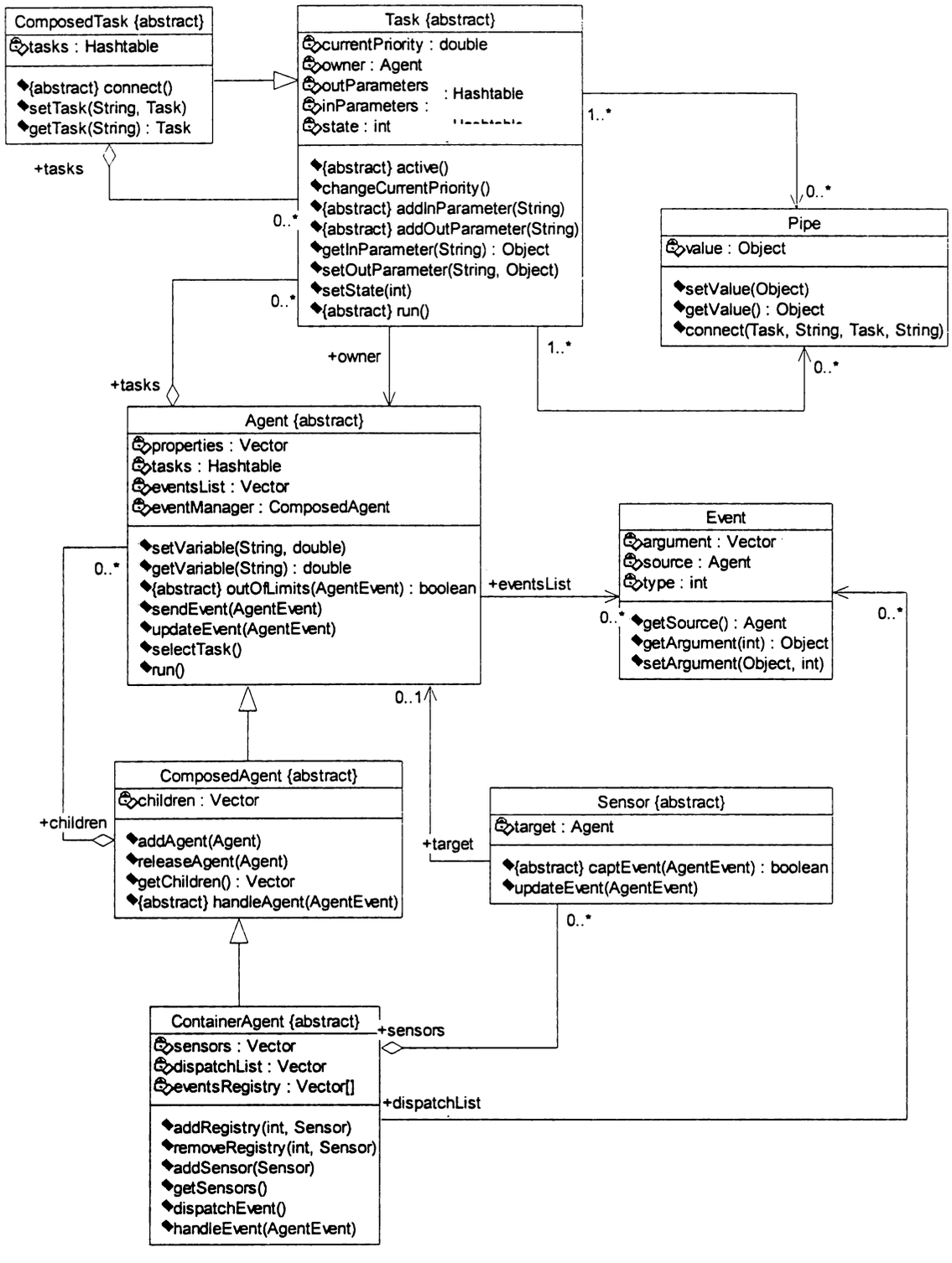

Figura - Diagrama de clases del *framework* Bubble

En la implementación del problema descripto (simulación de flujo de fluídos) utilizando la arquitectura provista por Bubble, el modelo de simulación está definido por un agente-ambiente

dividido en agentes-slices de fluido continuo, donde en cada uno de estos slices existen agentes-burbujas de fluido disperso.

Una decisión de diseño tomada, fue dividir el fluido continuo en partes (slices). Esto se basa en la suposición de que no hay influencia de comportamiento entre burbujas distantes, las burbujas reciben solo eventos de burbujas pertenecientes al mismo slice (o sea que están situadas en la misma subdivisión del fluido continuo). Así, se tiende a reducir el flujo de eventos en el sistema, aunque como contrapartida deben manejarse otras cuestiones como el traspaso de burbujas de un slice a otro. Otra ventaja de la división, es que mediante estas partes se puede representar ambientes con geometrías más complicadas. Inicialmente se utilizó una geometría 2D, comparando los resultados obtenidos con datos ya existentes de una simulación numérica en Fortran. Una vez observadas las perspectivas del enfoque, se realizó la extensión del problema a una geometría 3D.

4. Conclusiones y perspectivas

La simulación multiagente permite tratar los problemas de simulación desde un punto de vista novedoso, incorporando en este campo beneficios de la orientación a objetos tales como reusabilidad y flexibilidad de componentes En este sentido, Bubble explora las posibilidades de este enfoque.

La simulación realizada sobre fluídos con Bubble, arroja resultados coherentes con respecto al conocimiento que se tiene del problema físico en cuestión, y con tendencias similares a los resultados ya obtenidos de simulaciones numéricas existentes implementadas en Fortran. La extensión del problema de fluídos de 2D a 3D, es uno de los puntos donde se puede apreciar la flexibilidad de la arquitectura, ya que estas modificaciones aparentemente menores, presentan grandes inconvenientes dentro de un estilo procedural. Como continuación de este trabajo, se quiere utilizar Bubble para considerar nuevos aspectos del problema original (ignorados en una primera aproximación simplificada), y para tratar otros problemas más complejos.

Con respecto a la potencia de cálculo, como era de esperar, la simulaciones realizadas en Fortran resultaron más eficientes en tiempo que las realizadas con Bubble (implementado en Java). Sin embargo, este aspecto debe ser pesado contra otras características proporcionadas por el *framework*, como facilidades de modelado, flexibilidad, extensibilidad, etc.

El próximo paso dentro de esta línea, es extender la arquitectura básica de Bubble para soportar otras características tales como: concurrencia, distribución, razonamientos más elaborados en los agentes (capacidad cognitiva), herramientas de estadística, herramientas de visualización, etc. Actualmente se está aplicando Bubble para tratar problemas de otros dominios, entre los que se puede citar: un simulador de un mercado económico (que involucra compradores, vendedores y evolución de precios) y un simulador de fútbol (donde se estudian problemas de coordinación entre agentes).

Referencias

[Demazeau91] *From reactive to intentional agents*
In Decentralized Artificial Intelligence 2 (Y. Demazeau and J. P. Müller, eds.), pp. 3-14. Elsevier/North-Holland, Amsterdam. 1991.

[Drogoul92] *Multi-agent simulation as tool for modeling societies: Application to social differentiation in ant colonies,*
A. Drogoul and J. Ferber, Proc. Eur. Workshop Modelling Autonomous Agents Multi-Agent World, 4th,Rome, Italy, 1992.

[Ferber96] *Reactive Distributed Artificial Intelligence: Principles and Applications,*
J. Ferber, in "Foundations of Distributed Artificial Intelligence", Chapter 11, edited by G.M.P. O'Hare y N.R. Jennings. A Wiley-Interscience Publication. 1996.

[Gutowitz95] *Artificial-Life simulators and their application*
H. Gutowitz, Report written for the French goverment, 1995.

[Herrero95] *Geometrical automata for two phase flow simulation, Technical note*
V. Herrero, G. Guido-Lavalle, A. Clausse, Reprinted from Nuclear Engineering and Design 163 (1996) 117-124

[Johnson97] *Frameworks = (components + patterns)*
Communications of the ACM, 40(10), pp. 39-42, October 1997.

[Langton91] C. Langton, Preface (to artificial life ii). In C. Langton, C. Taylor, J.D. Farmer, and Rasmusse, editors. Artificial Life II, volume XI. Addison-Wesley, 1991.

[Meyer97] *Object-Oriented Software Construction,*
B. Meyer, Second Edition, published by Prentice-Hall, 1997.

[Shaw96] *Software Architecture, perspectives on an emerging discipline,Chapter 2: Architectural Styles*
M. Shaw and D. Garlan, published by Prentice-Hall, 1996.

[Shoham93] *Agent-oriented programming*
Shoham. Artificial Intelligence, 60(1):51-92, March 1993.

[Steels97] *The Artificial Life Roots of Artificial Intelligence,*
L. Steels, in "Artificial Life: an Overview" edited by C. Langton, MIT Press, 1997.

[Sycara98/2] *Multiagent Systems,*
K. Sycara, AI magazine Volume 19, No. 2