INTEGRACIÓN DE MODELOS ECOSIMPRO EN SIMULADORES DISTRIBUIDOS BASADOS EN HLA

Ferran Draper Pascual
Departament de Telecomunicació i Enginyeria de Sistemas, Escola Tècnica Superior d'Enginyeria
Universitat Autònoma de Barcelona . Ferran.Draper@gmail.com

Resumen

Este artículo presenta una metodología para alcanzar un elevado grado de automatización en la integración de modelos de simulación de sistemas continuos desarrollados con EcosimPro en un simulador distribuido bajo el estándar de simulación distribuida High Level Architecture (HLA).

Palabras Clave: Simulación distribuida, simulación de sistemas continuos, HLA, EcosimPro.

1 INTRODUCCIÓN

El objetivo del trabajo es desarrollar una aplicación que facilite el enlace de modelos de simulación EcosimPro [3] en un contexto de simulación distribuida HLA [2]. El principal propósito ha sido estudiar hasta qué punto se puede automatizar el enlace de modelos EcosimPro con un simulador distribuido, de forma que un usuario final lo pueda adaptar de forma transparente y añadiendo las mínimas modificaciones. Consecuentemente, se ha desarrollado un procedimiento que marca las pautas que ha de seguir el usuario para utilizar sus modelos, especificando dónde debe añadir el código adicional para implementar la funcionalidad dependiente del modelo y automatizando el resto de pasos necesarios.

El ejemplo sobre el que se ha trabajado es un juego de billar, donde cada modelo EcosimPro simula el movimiento de una bola de billar, con unas posiciones y velocidades iniciales determinadas al azar, y donde se envían a través de la red los movimientos de cada bola para determinar si hay alguna colisión entre las bolas o con los límites de la mesa de billar.

Las dos secciones siguientes presentan brevemente las tecnologías HLA y EcosimPro. En la sección 4 se exponen los detalles de la estructura de la federación HLA implementada. La sección 5 describe el procedimiento para integrar los modelos EcosimPro. En la sección 6 se comentan las particularidades del modelo de las bolas de billar. Finalmente, en la sección 7 se presentan algunas conclusiones.

2 ESTÁNDAR DE SIMULACIÓN DISTRIBUIDA HLA

El estándar HLA [2] es una arquitectura de propósito general desarrollada por la *Defense Modeling and Simulation Office* (DMSO) con el propósito de soportar la reutilización e interoperabilidad entre el gran número de simulaciones desarrolladas y mantenidas por el *US Department of Defense* (DoD). HLA está definido por el estándar IEEE 1516 [4].

HLA está formado por tres componentes: *Reglas de la Federación*, *Especificación de la Interfaz* y *Object Model Template* (OMT). En este trabajo sólo comentaremos brevemente la especificación de la interfaz. El software *Run-Time Infrastructure* (RTI) implementa dicha Especificación de la Interfaz y es el elemento más tangible de HLA. El RTI proporciona los servicios necesarios en una simulación distribuida HLA. Dos conceptos importantes en un simulador HLA son los de *Federado* y *Federación*. Un federado es cualquiera de los componentes que participan en una simulación distribuida. En este trabajo los federados son simuladores construidos a partir de componentes EcosimPro. Una federación es el conjunto de federados que integran una simulación distribuida. En una federación podemos encontrar entre 0 y n federados.

Los federados pueden unirse o abandonar una federación en cualquier momento, y comparten información con otros federados de la misma federación mediante *Objetos* e *Interacciones* HLA. Los objetos HLA son estructuras para representar información relativa a entidades que perduran en el tiempo, mientras que las interacciones representan eventos puntuales. Por ejemplo, en una simulación distribuida de entrenamiento de aviadores, la información de interés para la federación relativa a los aviones y radares se representaría mediante objetos. La información relativa al impacto de un misil con un avión se representaría mediante una interacción. Es importante no confundir el concepto de objeto HLA con el concepto convencional de objeto en el contexto de programación o modelado orientado a objetos.

Los federados de un simulador distribuido normalmente se encontrarán en ordenadores diferentes conectados a través de Internet. A diferencia de otras arquitecturas distribuidas como, por ejemplo, CORBA [5], en HLA la interacción entre los componentes distribuidos está normalmente ligada al tiempo. La federación empieza en tiempo 0 al ser creada por un federado (el primero que se une a ella), y avanza en el tiempo a través de peticiones de sus federados. La Figura 1 ilustra el ciclo típico que cumplimenta un federado cuando se une a una federación. Al unirse debe publicar/suscribir los objetos e interacciones HLA de los que enviará/recibirá actualizaciones. A continuación declarará sus políticas de gestión del tiempo y registrará sus objetos locales (cada una de las instancias de los objetos HLA publicados por el federado). Entonces puede empezar la simulación. En cada ciclo de simulación, los federados envían y/o reciben actualizaciones y hacen peticiones para que avance el tiempo de la federación hasta el instante deseado. Cuando abandonan la federación, los federados deben desregistrar sus objetos locales y eliminarlos, deshacer sus políticas de gestión del tiempo, abandonar la federación y eliminarla en caso de ser el último federado participante. Toda esta funcionalidad se implementa a través de peticiones al RTI y sus correspondientes *callbacks***,** llamadas procedentes del RTI en respuesta a un evento. En la Figura 1 se puede apreciar de forma resumida el ciclo de vida típico de un federado. Las peticiones al RTI aparecen representadas en negrita y las respuestas del RTI en azul.

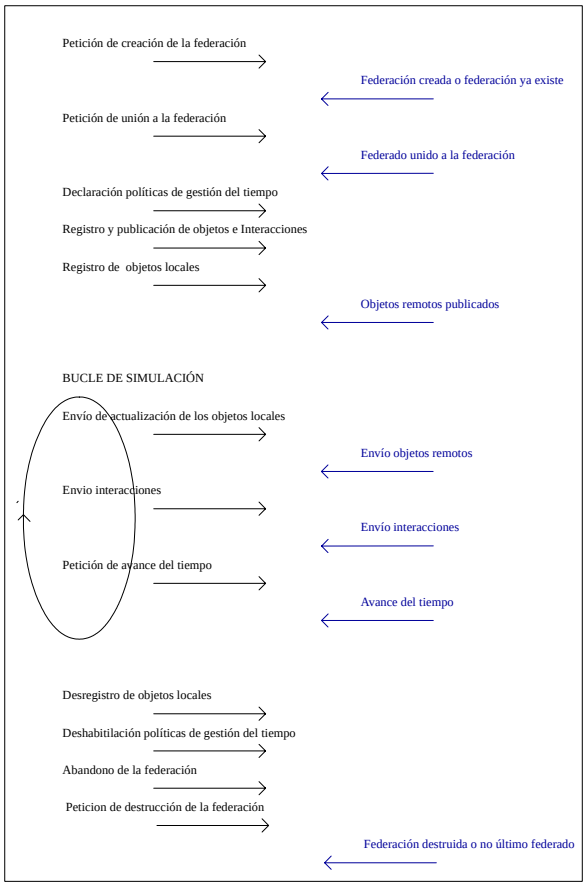

Figura 1: ciclo de vida de un federado

3 ENTORNO DE MODELADO Y SIMULACIÓN ECOSIMPRO

EcosimPro [3] es una herramienta de modelado y simulación de sistemas continuos y discretos. Los modelos EcosimPro se denominan componentes y pueden representar la dinámica de un sistema mediante ecuaciones algebraico-diferenciales, ecuaciones en diferencias y/o eventos discretos. El modelo matemático de un componente, a fin de poder ser reutilizado en diferentes contextos, no predefine causalidad computacional en sus ecuaciones. Cada componente puede tener asociado uno o varios modelos matemáticos con la causalidad computacional adecuada para su resolución numérica dependiendo del contexto de reutilización [1]. Estos modelos matemáticos se llaman particiones. A partir de ellos se puede definir casos de simulación o experimentos.

Los modelos se pueden utilizar para la simulación directamente con EcosimPro, pero también pueden ser exportados en forma de clases C++ para su reutilización en otras aplicaciones. Esta posibilidad es la que se utiliza en este trabajo para integrar modelos EcosimPro en simuladores distribuidos. En este caso, utilizamos directamente la clase C++ que codifica la partición, ya que definimos el experimento en el código de los federados.

4 DESCRIPCIÓN DE LA FEDERACIÓN ECOSIMPRO

4.1 ESTRUCTURA GENERAL

Básicamente, y tal como se puede apreciar en la Figura 2, los federados implementados disponen de una interfaz llamada *CRTIInterface* que implementa toda la funcionalidad HLA. Esta clase funciona a modo de interfaz y gestiona las comunicaciones con el RTI. Esta gestión es realizada mediante unas clases especiales de la arquitectura HLA llamadas *embajadores* (embajador del RTI para la comunicación federado/RTI y embajador del federado para la comunicación RTI/federado).

Habitualmente las comunicaciones del federado hacia el RTI se realizan a través del embajador del RTI, como por ejemplo la petición de creación de la federación, la petición de avance de tiempo o la declaración de un punto de sincronización, mientras

que el embajador del federado gestiona las respuestas del RTI. No obstante, esta separación en la gestión de las comunicaciones implica el uso de variables globales para acceder a los objetos HLA. Con el objetivo de evitar en nuestra implementación la utilización de variables globales, las peticiones y respuestas relativas a los objetos HLA se realizan en el embajador del federado. De esta forma se encapsulan los atributos y clases necesarias para los objetos en una sola clase.

El RTI provee dos clases que representan los embajadores, *RTIAmbassador*, el embajador del RTI, y *FederateAmbassador*, el embajador del federado. La segunda clase es abstracta; en una infraestructura HLA se tienen que diseñar las clases hijas de *FederateAmbassador*, de forma que éstas implementen los *callbacks* requeridos por el estándar HLA para especificar y definir su funcionalidad. En nuestra federación hemos diseñado la clase *CTrivialAmbassador*. Este embajador recibe las respuestas del RTI referidas a la gestión de tiempos y a la gestión de la federación. La clase *CEcosimProAmbassador*, heredera de esta última, implementa la funcionalidad referida a la gestión de los objetos HLA. Finalmente, la clase *CTemplateAmbassador*, heredera de la clase *CEcosimProAmbassador*, es una clase plantilla a modificar y completar por el usuario final a fin de adaptar la funcionalidad del federado a cada modelo de simulación en particular. En la Figura 2 se pueden observar las relaciones de herencia que hay entre estas clases.

Tal como hemos remarcado, la clase *RTIAmbassador* implementa las comunicaciones hacia el RTI, mientras que la clase final de las herederas de *FederateAmbassador*, *CtemplateAmbassador*, gestiona los *callbacks*, pero también las comunicaciones hacia el RTI relativas a los objetos. Adicionalmente, esta ultima clase realiza la comunicación con el modelo de simulación EcosimPro, a través de una librería dinámica o DLL creada a partir de las clases C++ del componente Ecosimpro. En la Figura 2 aparecen detalladamente las comunicaciones entre las clases de un federado.

4.2 FEDERADOS Y GESTIÓN DEL TIEMPO

En una federación HLA son soportadas una gran variedad de políticas de gestión del tiempo. El RTI provee un servicio opcional de gestión del tiempo para coordinar el intercambio de objetos e interacciones HLA entre los federados, de manera que estos eventos pueden ser asociados con un instante concreto en el tiempo. En general, los miembros de una federación pueden convivir con diferentes políticas de tiempos y con diferentes visiones del tiempo local. La única restricción es que el tiempo siempre debe avanzar. Esto significa que cualquier evento asociado a un instante de tiempo debe ser enviado antes de que el tiempo de la federación se haya avanzado hasta dicho instante. Así mismo, el *callback* relativo al evento será recibido por cada federado cuando su tiempo local coincida con ese instante de tiempo.

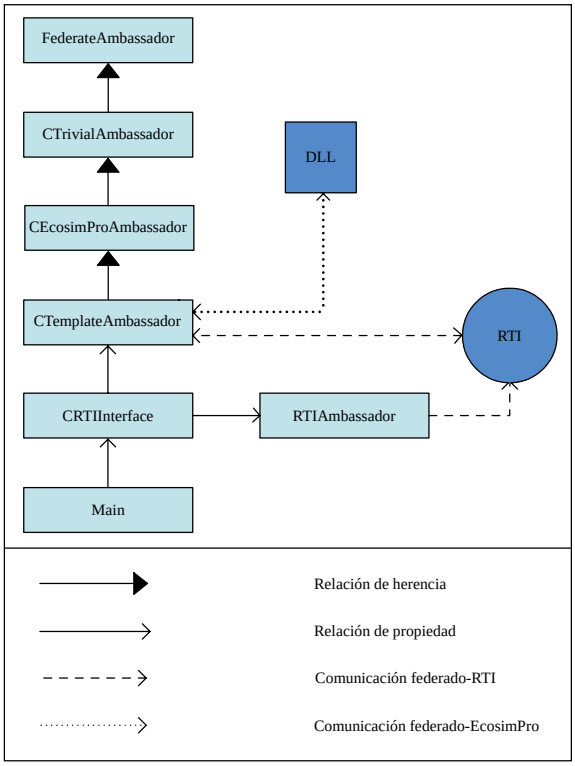

Figura 2: vista general de un federado

En una federación los federados pueden ser reguladores y/o restringidos. Que un federado sea regulador significa que es capaz de asociar un instante de tiempo a los eventos generados, mientras que un federado restringido es capaz de procesar esta clase de eventos. Los federados reguladores determinan hasta qué punto en el tiempo pueden avanzar los federados restringidos. Un federado que sea ni regulador ni restringido podrá generar y/o recibir eventos de este tipo, aunque ignorará la información referida al tiempo.

El tipo de política de tiempos utilizada por los federados determina como se ordenan los mensajes enviados entre ellos. En la federación diseñada todos los federados son reguladores y restringidos. Esto implica que todos los federados reciben los mensajes en el mismo orden y la simulación es repetible dadas las mismas condiciones iniciales. Esto añade una latencia adicional respecto a una federación donde los eventos no estén asociados al tiempo y puedan ser entregados en cualquier momento, pero es necesario en simulaciones donde interesa que las ejecuciones

sean deterministas (por ejemplo, para el análisis de sistemas).

En la federación EcosimPro, y a diferencia de lo que se puede encontrar en otras federaciones HLA, todos los federados avanzan en el tiempo de forma sincronizada, lo que significa que el tiempo local es el mismo en todos los federados. Esto es necesario ya que la salida del modelo de simulación local de cada federado depende de los resultados de los modelos de simulación remotos. En el ejemplo de las bolas de billar, una bola local puede colisionar con otra en cualquier instante, y por lo tanto debe conocer las posiciones de las bolas remotas en todo momento. El incremento utilizado para avanzar el tiempo de la federación es el mismo que se utiliza como intervalo de simulación para los modelos EcosimPro.

Inicialmente los federados EcosimPro avanzan en el tiempo con un intervalo de simulación prefijado. Dado un instante de tiempo t_n, los federados calcularán el estado alcanzado por su modelo de simulación local para el instante de tiempo t_{n+1}. Estos valores pueden ser considerados como no válidos por un federado al detectar que con el intervalo de simulación utilizado se han pasado por alto eventos que son consecuencia de las interacciones entre los federados. Por ejemplo, en la Figura 3 podemos ver el caso de las bolas de billar, donde se determinaría que en el instante t_n no hay colisión entre dos bolas, mientras que en el instante t_{n+1} el instante de colisión se ha sobrepasado. En este caso deberá encontrarse el instante de tiempo $t_{n'}$ y esto se consigue recalculando un nuevo estado a partir de t_n con un intervalo de simulación menor.

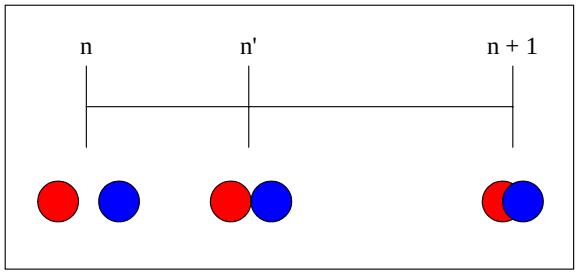

Figura 3: instante de colisión en la federación Billard

Para gestionar este problema se han diseñado dos tipos de objetos HLA que se describen a continuación.

4.3 OBJETOS E INTERACCIONES HLA

El problema anteriormente descrito se ha solucionado utilizando una presimulación que permitirá detectar aquellos eventos que obliguen a determinar el instante concreto en que ocurren (ver Figura 3).

Para ello se han definido dos objetos HLA que representan el mismo modelo de simulación EcosimPro. Uno de ellos está vinculado a un instante de tiempo ti+Δt calculado como la suma del tiempo local t_i más el intervalo de simulación Δt con el que se está trabajando. El otro objeto no tiene asociado ningún tiempo.

El primero es utilizado para enviar el estado alcanzado por los modelos de simulación de cada federado en el instante ti+Δt, de forma que el tiempo de la federación pueda avanzar hasta dicho instante. El segundo se utiliza para enviar el estado obtenido como presimulación para decidir si los estados alcanzados con el intervalo de simulación Δt pueden ser considerados válidos. Es decir, a partir de los objetos recibidos se determina que durante dicho intervalo no ha ocurrido ningún evento significativo que obligue a recalcular el incremento Δt. En caso contrario, se pedirá a través de una interacción HLA que el intervalo de simulación Δt de la federación se reduzca. En el momento que se considere legal un determinado intervalo Δt (todos lo federados deben acordarlo), se autorizará a la federación a que el tiempo se avance hasta el instante ti+Δt. A partir de este instante se restaura el valor del intervalo de simulación al su valor original con el objetivo de agilizar la simulación.

4.4 UTILIZACIÓN DE LAS PARTICIONES ECOSIMPRO

La información contenida en los objetos e interacciones HLA se transmite en la federación como cadenas de caracteres [2]. Por otro lado, las particiones de los componentes EcosimPro, que se exportan como librerías DLL generadas a partir de su código fuente en C++, contienen variables que pueden ser de diferentes tipos de datos. Estos dos factores añaden una complejidad importante a la automatización del proceso de generación del código de un federado, ya que las clases que gestionan las simulaciones obtenidas de las particiones tienen que ser diseñadas para trabajar con un número desconocido de variables de varios tipos de datos también desconocidos. Además, los objetos e interacciones HLA que se han de poder intercambiar en forma de mensajes en la federación tienen que estar declarados en un archivo de configuración previamente a la ejecución de la misma. Adicionalmente, los objetos que se quieren compartir y cada uno de sus atributos deben ser declarados mediante mensajes al RTI antes de iniciar la simulación.

Para solucionar estas dificultades se ha creado una aplicación que analiza la partición utilizando su DLL y muestra las variables del componente EcosimPro. De esta manera se permite al usuario seleccionar las

variables de interés y generar automáticamente el archivo de configuración de los federados, donde se declaran los objetos HLA con sus atributos y las interacciones HLA. También se genera un archivo de configuración para los objetos EcosimPro donde se encuentran las variables de interés y sus tipos de datos. Estos ficheros de configuración son los que permiten en tiempo de simulación relacionar la información contenida en los objetos e interacciones HLA con las variables (y sus tipos de datos) de interés seleccionadas para cada modelo.

Esta gestión se automatiza a través de tres clases. Son necesarias tres clases para gestionar los objetos EcosimPro en la federación. La clase *CObjectManager* representa los identificadores de las variables de interés y sus tipos de dato. La clase *CEcosimProInterface* representa los modelos de simulación de cada federado presente en la federación. Cada uno de estos federados contiene una instancia de esta clase asociada a su modelo de simulación local y un vector de instancias asociadas a cada uno del resto de federados. Mediante los objetos *CObjectManager* se pueden relacionar las variables de interés del resto de federados con las del simulador local. Dichas variables de interés están almacenadas en cada objeto *CEcosimProInterface* como un vector de instancias de la clase *CEcosimProAttribute*. Esta última clase ha sido diseñada para soportar atributos de tipo real o entero, dependiendo del tipo de datos de la variable de interés en concreto. Es importante remarcar que estas tres clases se utilizan conjuntamente para maximizar la independencia del modelo de simulación del código propio del federado.

4.5 BUCLE DE SIMULACIÓN

Antes de entrar en el bucle de simulación los federados deben crear la federación (en caso de que ésta no exista), unirse a ella, declarar sus políticas de tiempo, los objetos e interacciones de las que publicarán y recibirán actualizaciones y registrar los objetos locales. A continuación, existe un punto de sincronización inicial donde los federados esperarán a los otros miembros de la federación antes de iniciar la simulación.

Para todo intervalo del bucle, cada federado enviará la presimulación de su objeto local, y decidirá en base a su propio objeto local y a los objetos remotos de los otros federados si los resultados de la simulación son legales para avanzar el tiempo. Si no se recibe ninguna petición para reducir el intervalo de simulación, cada federado enviará los valores del objeto que representa su modelo EcosimPro local y se avanzará el tiempo de la federación según el intervalo utilizado. Si algún federado necesita que se reduzca el intervalo, enviará una interacción para notificarlo. En este caso no se avanzará el tiempo de la federación, y se recalcularan los valores de los modelos con los nuevos intervalos. En el momento en que un federado que ha pedido la división del intervalo decida que estado alcanzado es válido, puede hacer una petición para restaurar el incremento original, pero si algún federado no ha resuelto aún los problemas que hayan causado la división del intervalo, puede bloquear esta restauración.

Toda esta gestión es bastante compleja ya que implica el uso de varios puntos de sincronización para asegurar que los objetos e interacciones HLA que no están asociados a un punto de tiempo se reciban en el momento y orden precisos. Adicionalmente, varios federados pueden pedir divisiones y restauraciones del tiempo en un mismo instante, y se debe garantizar que todas las peticiones se procesan. Las divisiones de tiempo, siempre que sean posibles (se puede reducir el intervalo hasta 0.1 segundos), son más prioritarias que las restauraciones, mientras que ninguna restauración es posible si algún federado indica lo contrario (bloquea la restauración).

Cuando se llegue al instante de tiempo marcado como final de la simulación, los federados deben eliminar sus objetos locales, deshacer sus políticas de tiempo y abandonar la federación (el último federado destruye la federación).

5 PROCEDIMIENTO DE AUTOMATIZACIÓN

Para generar un federado adaptado a un modelo EcosimPro, el usuario final debe seguir los siguientes pasos:

1) Inicializar las variables de entorno del sistema operativo COMPONENT y PARTITION con los valores correspondientes del modelo a integrar y la variable ECOSIM_PATH con la ruta de la aplicación EcosimPro.

2) Copiar los archivos C++ de la componente en el directorio correspondiente.

3) Ejecutar la aplicación proporcionada para generar automáticamente el código adicional necesario en las clases de la partición del componente para generar la librería DLL.

4) Ejecutar la aplicación proporcionada para generar la librería DLL de la partición.

5) Ejecutar la aplicación proporcionada para seleccionar las variables de interés y generar

automáticamente el archivo de configuración de EcosimPro y el de los federados.

6) Compilar el código del federado.

A partir de este punto, la aplicación funciona correctamente sin añadir ninguna funcionalidad adicional. Para cada intervalo de simulación, los federados calcularán, mediante su modelo de simulación EcosimPro, los valores de las variables de interés de su objeto local y enviarán al RTI la presimulación. A continuación, al no recibir ninguna petición de división del intervalo de simulación, enviarán su objeto local. Finalmente imprimirán por la salida estándar los valores de su objeto local y de los objetos remotos de los otros federados.

Tal como se ha comentado previamente, la federación dispone de las herramientas necesarias para reducir el intervalo de simulación en caso de que sea necesario. El programador deberá modificar la clase *CTemplateAmbassador* para adaptar la funcionalidad al simulador distribuido. Básicamente, se han de redefinir los siguientes métodos de dicha clase:

1) *Init*: método donde se inicializan los atributos de interés del modelo EcosimPro si es necesario (p.e. condiciones iniciales, valores de parámetros, etc).

2) *Divide*: método para decidir si el estado alcanzado con el intervalo de simulación es legal, hacer la petición de división del intervalo en caso contrario o restaurarlo cuando sea oportuno.

3) *Dividing*: método para bloquear una posible restauración en caso de que el federado decida que ésta no es posible.

6 EJEMPLO DE APLICACIÓN

6.1 FEDERACIÓN BILLARD

El modelo de simulación distribuida desarrollado representa una mesa de billar, donde cada federado participante tiene un modelo de simulación del movimiento de una bola de billar. Las bolas se inicializan con posiciones y velocidades iniciales determinadas al azar, y cuando colisionan con los bordes de la mesa o con otra bola de un federado remoto se recalculan sus velocidades. Si en un instante de tiempo t_n no existe colisión, y en el instante t_{n+1} se detecta que el instante de colisión se ha sobrepasado, el intervalo de la federación se reducirá a la mitad y se recalcularán los valores obtenidos con el nuevo intervalo. En la Figura 4 se puede observar una federación con dos participantes donde se repite la simulación por no considerarse válidos los resultados. Nótese que una vez detectadas dos trayectorias que llevan a colisión (instante t_{n+2} con intervalo 4t) el intervalo original 4t no se restaura hasta que el instante exacto de la colisión se ha detectado (instante t_{n+3} con intervalo t).

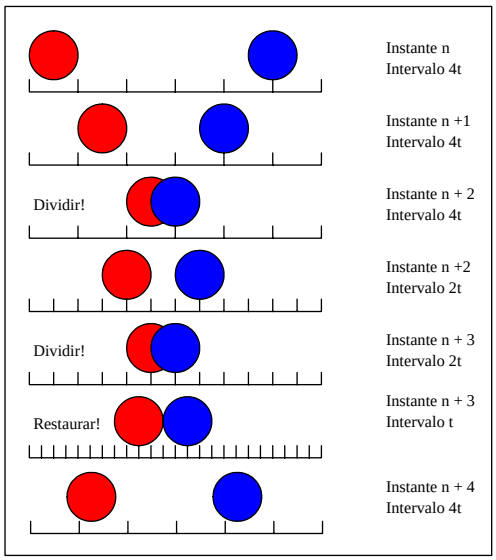

Figura 4: división del intervalo de simulación en la federación Billard

Esta funcionalidad se ha conseguido redefiniendo los métodos *Init*, *Divide* y *Dividing* de la clase *CTemplateAmbassador*. En *Init* se calculan las posiciones y velocidades iniciales de forma aleatoria. En el método *Divide* se calcula si hay alguna colisión y si ésta es precisa, tanto con los extremos de la mesa como con las bolas de los federados remotos enviadas a través de la presimulación. En caso necesario, desde este método se envían las interacciones para dividir o restaurar el intervalo de simulación. En *Dividing*, si al federado no le interesa que se restaure el intervalo de simulación, se bloquea una posible restauración del tiempo pedida por otro federado.

6.2 BILLARD OPENGL

Se ha añadido una capa OpenGL a la aplicación anterior, para apreciar los resultados obtenidos de una forma gráfica. En la figura 5 hay un ejemplo de ejecución con 4 federados.

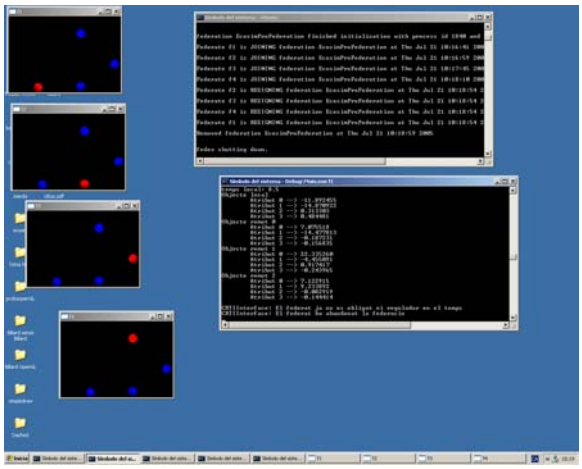

Figura 5: ejemplo ejecución de la aplicación Billard OpenGL

7 CONCLUSIONES

Con este trabajo hemos podido analizar las dificultades principales para integrar modelos EcosimPro en un simulador distribuido basado en la arquitectura HLA. Dadas las características de HLA y EcosimPro, la automatización de esta integración no es una tarea trivial debido tanto al problema de gestión del tiempo de simulación en lo relativo a las consecuencias de la interacción entre los simuladores que participan en la federación como a los problemas para compatibilizar la representación de los datos en la federación con su representación en los componentes EcosimPro. En este trabajo se ha creado una plataforma que ayuda a un usuario final a integrar sus modelos EcosimPro en una federación HLA. La automatización, tal como se ha descrito y siempre desde el enfoque adoptado, es posible hasta cierto punto. No obstante, las herramientas desarrolladas facilitan esta tarea.

Agradecimientos

Desearía agradecer el apoyo recibido por parte del Departamento de Telecomunicaciones e Ingeniería de Sistemas así como la confianza que han depositado en mi. También quiero agradecer el soporte del equipo de Empresarios Agrupados que ha desarrollado EcosimPro.

Referencias

[1] Cellier, F. and H. Elmqvist (1993), *Automated Formula Manipulation Supports Object-Oriented Continuous-System Modeling*. IEEE Control Systems.

[2] Defense Modeling and Simulation Office (2002), *RTI 1.3-Next Generation Programmer's Guide Version 5*, Department of Defense, US.

[3] EcosimPro (2004), *EcosimPro User Manual Version 3.3*, EA International, Spain.

[4] http://standards.ieee.org/catalog/olis/compsim.html

[5] OMG (2002), *The Common Object Request Broker: Architecture and Specification, Revision 3.0*. www.omg.org.