Análisis de DIS y VRML para un Entorno de Simulación Distribuida Aplicada a la Navegación de Sistemas Autonómos Móviles.

M. Sánchez Alonso, J.M. García Carrasco
Dept. Ingeniería y Tecnología de Computadores

Antonio F. Gómez Skarmeta, H. Martínez Barberá
Dept. Informática, Inteligencia Artificial y Electrónica

Universidad de Murcia
Apartado Correos 4021, 30001 Murcia
Telf: 968-364607, Fax: 968-364151
E-mail: {skarmeta,msanchez}@dif.um.es

Abstract

In this paper we present the problems associated to the design of simulator for complex models, and the solutions that the Distributed Simulations Environments offer to this kind of activities. We propose the development of tool that consider not only the use of the Distributed Interactive Simulation protocol (DIS) but also its interacction with VRML, and the semi-automatic generation of distributed simulators starting from a simulation language. In order to analyse its possibilities we present the application of approach to the simulation of the movement of an autonomous robots, where different distributed components as the navigation, or the environment interact through a virtual world.

1. Introducción

En la actualidad, el campo sobre el que se desarrollan las simulaciones se ha ampliado de manera considerable, al entrar en juego nuevas tecnologías y protocolos de comunicación en Redes de computadoras, así como por la aplicación de nuevos conceptos de arquitecturas y el desarrollo de nuevos lenguajes de programación, tanto de propósito general como específico. El resultado es que lo que hace varios años hubiéramos dejado por imposible, ya tiene una solución. No es descabellado pensar en simular un conjunto de moléculas interactuando entre sí, a la vez que se tiene en cuenta el comportamiento de cada átomo que las componen, o simular en tiempo real el desplazamiento de un vehículo por la superficie de otro planeta, o el comportamiento de una planta industrial al completo y con todo lujo de detalles. Todo ello se debe al desarrollo de los medios físicos de comunicación y a las técnicas para emplearlos.

Uno de los ámbitos de aplicación de las nuevas tecnologías y tendencias de las comunicaciones es la mejora de los resultados en las simulaciones, gracias al cambio de concepto sobre como y donde debe ser tratada la información que compone la base de toda simulación. También se ha experimentado un gran avance en lo referente a los métodos de representación, sustituyendo los ejes de coordenadas en dos o tres dimensiones, por escenarios en los que la información se representa transmitiendo al usuario una cantidad de información adicional, que con otros métodos no es perceptible. En cuanto al como y donde tratar la información, son criterios que se han modificado debido a que actualmente, se puede conseguir de manera fiable que un conjunto de equipos manejen la misma información y sean capaces de intercambiar resultados, casi simultáneamente. Vamos a presentar, someramente, las técnicas y métodos más utilizados para conseguir estos logros.

En lo referente a la visualización de resultados en los procesos de simulación, lo que encontramos a la orden del día es la utilización de lo que se conviene en denominar "Mundos Virtuales". Realmente se trata de escenarios tridimensionales, visualizados en un *display* concreto, pero que se generan como combinación de procesos que se ejecutan en varias máquinas diferentes conectadas mediante un Red de comunicaciones, y por ello pueden ser tan complejos y completos como se desee. Ni que decir tiene que esto no hubiera sido posible sin el avance correspondiente en el desarrollo del *hardware* de aceleración de gráficos, que los dotan de mucho realismo. Al paso que se ha desarrollado el *hardware*, ha habido una avance sustancial en el *software* asociado, en forma de librerías de control de dicho *hardware*, y posteriormente con la aparición de lenguajes de programación específicos para tales propósitos como es el denominado VRML (*Virtual Reality Modeling Language*) [1] se ha culminado el proceso, posibilitando a cualquier usuario el manejo eficiente y sencillo de las funciones gráficas, para crear sus propios "Mundos Virtuales".

En lo referente al manejo de información y su intercambio por los distintos elementos que van a realizar la simulación, también se plantean serios avances en los últimos años. Aunque el concepto de Sistema Distribuido es algo antiguo en la Informática, la estandarización de ciertos aspectos de los sistemas distribuidos es algo más actual (téngase en cuenta que dichos sistemas se han desarrollado a la zaga del avance de las redes de comunicaciones), en concreto, lo que se refiere al

intercambio de información para un propósito específico y de una manera determinada. Como ejemplo en lo referente a simulaciones, encontramos SIMNET un estándar para simulaciones distribuidas desarrollado a principio de 1985 bajo los auspicios de DARPA, cuya primera y principal aplicación fue con fines militares, aunque ello no era óbice para que pudiera haber servido para otros menesteres. Con posterioridad y siguiendo este camino se desarrolla el estándar IEEE 1278.1 llamado DIS (*Distributed Interactive Simulation*) en 1995 [2], que como su nombre indica es válido para simulaciones distribuidas interactivas. Ambos SIMNET y DIS, se entienden como protocolos que forman parte de la pila de protocolos que conforma la arquitectura de la red que se utiliza, aunque el segundo se apoya en TCP/IP y el primero no [3]. El hecho de ser protocolos de comunicaciones, va a tener una incidencia decisiva en su aplicación, ya que esto mismo los va ha hacer independientes del soporte de Red que exista para constituir el Sistema Distribuido.

Como último aspecto a comentar en esta introducción es interesante indicar los esfuerzos que se vienen desarrollando últimamente para llevar a cabo el proyecto INTERNET 2, que representa la utilización de nuevas tecnologías de transmisión y nuevos servicios con un índice de prestaciones mucho más elevado que el de la "vieja" INTERNET. Es precisamente en este entorno en el que la aplicación de protocolos como DIS para simulaciones distribuidas interactivas produce el mejor rendimiento, así como facilita la utilización de entornos gráficos desarrollados con lenguajes tales como VRML. En los puntos que veremos a continuación, exploraremos la relación entre Modelos y Simuladores, viendo cuales son los condicionantes que hacen que una simulación sea factible de llevarse a cabo y en caso contrario las posibles soluciones. Posteriormente presentaremos una alternativa para el desarrollo de simuladores, que contempla los problemas anteriores y su resolución mediante los métodos, técnicas y tecnologías actuales y más novedosas. Finalizando con un ejemplo de aplicación basado en los pasos a seguir para conseguir, de esta manera, la simulación distribuida e interactiva del control y movimiento un robot con ruedas, que se entiende como un sistema relativamente complejo

2. Modelos y Simuladores

Tal y como se plantea en la actualidad la simulación de procesos, se hace necesario manejar unos modelos que representen dichos procesos y que son computacionalmente complejos, bien sea por la precisión requerida, el número de parámetros y variables a tener en cuenta, o simplemente por la propia estructura del proceso en cuestión. Es obvio que esta problemática no será aplicable a todos los modelos que se desarrollan, pero existe un gran número de ellos, que condicionados por factores externos al sistema a modelar (p.e. obtención de resultados en tiempo real, ajuste de parámetros de funcionamiento no conocidos a priori, etc.), requerirán unas características de simulación particulares que influyen a la hora de convertir el modelo teórico en un modelo computacional. También deberán ser tenidas en cuenta las herramientas y lenguajes de soporte que se vayan a manejar para la consecución de dicho objetivo, ya que la elección de éstas condicionará, en gran medida, el conocimiento que sobre el comportamiento del sistema nos proporciona la simulación [4].

Partiendo de la base, que el tema que nos ocupa es precisamente el de procesos a simular, que pareciendo sencillos inicialmente, puede resultar una tarea compleja su planteamiento en un modelo, nuestro objetivo es obtener herramientas de simulación para modelos complicados, que sean capaces de operar aún en el caso de que haya que tener en cuenta un buen número de condicionantes externos durante el proceso de simulación. Para ello, no se pretenden obtener planteamientos o técnicas novedosas sobre la simulación de modelos, sino alternativas al proceso de simulación que faciliten la obtención de resultados, de una forma agradable y flexible para el usuario, y a su vez no compliquen excesivamente la tarea de construcción de la aplicación encargada de la simulación, facilitando la mayor parte de las decisiones de diseño que sean necesarias llevar a cabo. Para hablar de cuestiones precisas, puede ser interesante tomar como ejemplo las simulaciones de procesos en las que el objetivo es conseguir que el simulador funcione en tiempo real.

Obviamente, la simulación en tiempo real es una cuestión ya conocida, ampliamente estudiada y en la que el problema fundamental no es la complejidad de los algoritmos que se utilicen para representar el modelo, sino la potencia del computador sobre el que se ejecuta la simulación, que condiciona aspectos fundamentales como son la visualización de resultados en tiempo real, la posibilidad de modificar parámetros de ejecución en función de los resultados parciales que se obtienen, y si la escala de tiempo simulado se corresponde con la realidad. Para solucionar estos problemas se han venido utilizando métodos tradicionales como son los optimizadores de código, las librerías y aceleradores de gráficos, y por supuesto la mejora de las prestaciones de los equipos a utilizar. El problema es que se puede llegar al límite actual en estos campos y aún así no haber conseguido los objetivos de la simulación.

La solución pasa entonces, por aplicar métodos no tan tradicionales en los que podemos apoyarnos para lograr los objetivos deseados. A *grosso modo* podríamos centrar el problema fundamental en la potencia del sistema en que se

ejecuta la simulación, para solucionarlo sería conveniente aumentarla de forma que no tengamos que actuar físicamente sobre él, y esto nos lleva directamente al uso de los Sistemas Distribuidos y de las Redes de Comunicaciones asociadas. Resuelta esta circunstancia, como efecto colateral surge el inconveniente de la utilización de aplicaciones sobre Sistemas Distribuidos y Redes, en los que el programador debe preocuparse de que el usuario no sea consciente de la existencia de la distribución de recursos, y de que observe únicamente los resultados de dicha distribución. Obsérvese que las simulaciones en tiempo real son firmes candidatos a utilizar este tipo de sistemas, ya que se puede repartir la potencia de cálculo entre las diferentes CPU´s que conformen el Sistema Distribuido, utilizando la Red de comunicación para intercambiar información entre los distintos procesos que se ejecutan [5].

Todo este planteamiento conlleva una serie de problemas de diseño a la hora de realizar la aplicación que ejecutará la simulación. Aunque existen herramientas para trabajar sobre Sistemas Distribuidos son de propósito general, y por lo tanto no suelen adaptarse a las necesidades requeridas, al igual que para cada modelo habrá que realizar una aplicación completa que refleje sus características particulares. Se impone, por tanto, la necesidad de establecer las bases para el desarrollo de un Generador de Simuladores, que integre las prestaciones en el uso de los Sistemas Distribuidos con el desarrollo, de forma semiautomática, del código asociado al simulador del modelo a realizar, sea cual sea dicho modelo; sin necesidad de que el programador conozca los protocolos de comunicaciones que se manejan en la Red o las políticas de asignación de recursos que emplea el Sistema Distribuido sobre el que se ejecuta la simulación. Por supuesto, en cualquier caso todo ello sería aplicable a un sistema *Stand-alone* como caso particular de Sistema Distribuido de un solo componente.

3. Generación de Simuladores

Para conseguir los anteriores objetivos será necesario analizar el problema de la construcción de simuladores a partir de un modelo determinado. Independientemente de las técnicas de desarrollo estructurado que se puedan aplicar, habrá un primer problema a resolver que dependerá de la naturaleza del método de simulación, ya que ésta se ve condicionada dependiendo de sí lo que se quiere es modelar un conjunto de eventos discretos en el tiempo, o bien si es una simulación continua en el mismo. El primer caso implica modelos restrictivos en cuanto a los sistemas que se quieren representar y para aplicaciones bastante concretas. El segundo tipo es más generalista y representa la mayoría de las simulaciones que se tratan de realizar, a su vez es más complejo en cuanto a la construcción del correspondiente simulador. En resumen, trataremos las simulaciones continuas, intentado encontrar los patrones comunes de los simuladores que las representan.

Tomando como base de partida el modelo de un sistema que varía continuamente en el tiempo, lo que se puede deducir de una primera observación es que, sea cual sea el sistema modelado, para su simulación será necesario establecer unas fases por las que pase el proceso. En primer lugar será necesario establecer las condiciones iniciales y de contorno del proceso de simulación, estableciendo el valor de las constantes a utilizar y los parámetros, que presumiblemente no deben variar durante el experimento. Una vez alcanzada y rebasada esta fase, la siguiente consistirá en aplicar las funciones que definen el modelo con estos valores iniciales y aumentando el valor del tiempo. Por último, una vez alcanzado el tiempo establecido se puede, si es necesario, actuar sobre los valores obtenidos. El ejemplo más típico es la simulación de una masa que cuelga de un resorte, en la que el sistema se modela mediante las ecuaciones que gobiernan el equilibrio dinámico, siendo los valores iniciales la constante del resorte y el valor de la masa.

Para el ejemplo dado, el proceso de simulación consistirá en la resolución de las ecuaciones del modelo mediante rutinas de integración por aproximación numérica, así para cada instante de tiempo se obtiene un valor que depende del instante anterior. Configurando adecuadamente la simulación se pueden obtener los resultados de cada instante simulado y por último su representación gráfica que representa el comportamiento del sistema en el tiempo. Lo que a primera vista parece sencillo, se convierte en un problema de cierta envergadura sino se toman una serie de decisiones de diseño adecuadas, tales como que rutina de integración elegir, o si los resultados de la simulación deben ser visualizados al final del proceso o durante la realización del mismo. Seguramente durante el propio desarrollo se presentarán más dificultades, pero se concluye que las anteriores son las más importantes y las más decisivas para la construcción del simulador.

Debido a que la simulación se puede descomponer en varias fases, la primera y la última claramente declarativas y la segunda de ejecución, no es descabellado establecer una analogía entre una simulación y un programa en un lenguaje estructurado; de hecho hace tiempo que existen lenguajes específicos para expresar modelos y realizar su simulación, pero en general no permiten construir cualquier tipo de simulador. Buscando algo un tanto más generalista, se llega a la conclusión de que hay que establecer un Lenguaje Constructor de Simuladores, bien partiendo de cero o bien aprovechando los trabajos ya desarrollados en este campo. De cualquier manera, se deberá elegir este lenguaje para que no sea restrictivo de cara a la simulación o a los modelos, sino que por el

contrario, enriquezca y abra el abanico de posibilidades en todos los aspectos de importancia del proceso que nos ocupa.

Una vez establecido el Lenguaje Constructor, será necesario adaptarlo para que sea capaz de actuar en diversos sistemas. Retomando la problemática del apartado anterior, deberá tener capacidad para funcionar en Sistemas Distribuidos, lo que permite realizar simulaciones complejas sin limitación de recursos. El principal problema al que nos enfrentamos ahora, será como vamos a realizar la simulación distribuida si partimos de un modelo monolítico. En primer lugar, a la hora de establecer el modelo se debe utilizar una filosofía distinta, al saber que se va a simular sobre un Sistema Distribuido. Deberemos establecer los elementos del sistema a modelar que interactúan entre sí, con el objeto de que el modelo se convierta en un conjunto de submodelos que intercambian información entre ellos para cada instante de la simulación. De esta forma, habremos obtenido el conjunto de tareas que posteriormente se pueden distribuir en el sistema. En segundo lugar, será necesario establecer el mecanismo de recogida de la información de todas los submodelos para su posterior visualización, que preferentemente debería realizarse en tiempo real, y para ello surgirán nuevas tareas que serán susceptibles de ser distribuidas a su vez.

Con estos planteamientos, parece bastante claro que lo que se está definiendo es un sistema que integre tres problemas importantes en la actualidad como son los lenguajes de simulación, el intercambio de información en tiempo real a través de un Red de comunicaciones y la generación de gráficos. Estos tres aspectos pueden cubrirse utilizando técnicas muy actuales, en concreto los dos últimos, nos referimos al protocolo DIS (*Distributed Interactive Simulation*) para todo lo referente al intercambio de información entre los submodelos con el objeto de conseguir una simulación interactiva, y la lenguaje VRML (*Virtual Reality Modeling Language*) que permite generar mundo virtuales a partir de gráficos calculados en distintas máquinas conectadas en una Red, como se presentan en Fig. 1. Obviamente todo esto puede realizarse siempre y cuando se cuente con una Red de Altas Prestaciones que garantice el intercambio de información a alta velocidad, tal y como se plantea en el Proyecto INTERNET 2.

Quedaría por concretar el Lenguaje Constructor para construir los submodelos; existen varias opciones para elegir pero aún no están lo suficientemente maduras como para decantarse, lo que sí parece claro es que debe ser un lenguaje con la potencia de la orientación a objetos, en cuanto a las propiedades de jerarquía y herencia, factores fundamentales para el desarrollo de software, y por tanto del código final del simulador. Todos estos aspectos ligados proporcionan la estructura de lo que constituye un Generador de Simuladores para Sistemas Distribuidos, que sería una herramienta capaz de generar el código de los simuladores de cada submodelo definido, en el que se incluiría, de forma transparente al usuario, todo lo necesario para que las simulaciones de los submodelos se comuniquen entre sí, así como la información necesaria para construir el mundo virtual que sirva para la visualización de los resultados en tiempo real.

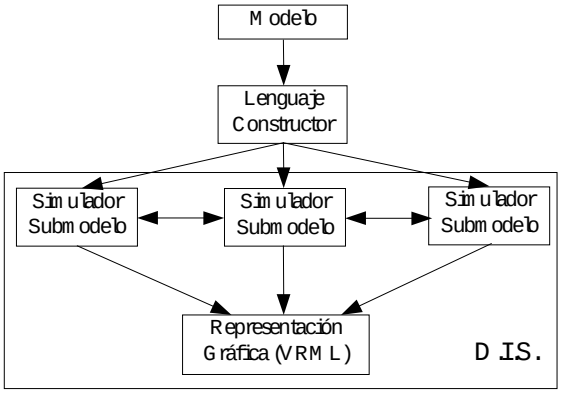

Figura 1

4. Un ejemplo de aplicación

Para comprobar como se desarrollan las ideas antes expuestas, podemos basarnos en un sistema, que precisamente se está utilizando como punto de partida para la construcción y verificación de un Generador de Simuladores para Sistemas Distribuidos. El sistema en cuestión es un robot móvil con guía de ultrasonidos para sortear obstáculos, realizando mapas de las trayectorias a seguir. La estructura de dicho sistema en la actualidad se ha realizado de forma distribuida, de manera que los tres componentes que la forman se intercambian información a través de una red de comunicaciones. Fundamentalmente existe un bloque de Control que es el que recoge los datos relativos a la gestión de los motores que impulsan al robot, en conjunción con los datos recibidos por el emisor/receptor de ultrasonidos y conociendo la posición, se establecen las próximas acciones de control a realizar. La arquitectura de dicho bloque se puede ver en la Fig. 2.

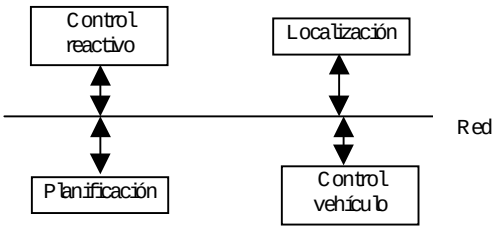

Figura 2

El Control reactivo se encarga de tomar las decisiones instantáneas deferentes a la navegación, la Planificación realiza la supervisión a alto nivel del desarrollo de los objetivos, la Localización se

encargará de construir un modelo del entorno y de ubicar al robot dentro de él, por último el Control del vehículo gobierna las señales recibidas de los sensores y las enviadas a los actuadores [6]. Para poder perfeccionar y depurar cada uno de los bloques, sin necesidad de poner en funcionamiento el robot, se puede sustituir a éste por un modelo que lo simule, este hecho tiene una ventaja adicional, y es que modificando el modelo, se puede probar el sistema con diferentes tipos de robot (p.e. aplicándolo a un robot con ruedas o a uno con patas, etc.), sin necesidad de disponer físicamente de él. La estructura del sistema completo de puede ver en la Fig.3.

El simulador que se ha desarrollado es capaz de tener en cuenta multitud de factores que son los que hacen que los resultados del modelo sean los esperados. Entre estos factores se pueden encontrar, la posible inclinación del terreno sobre el que se ejecutan las pruebas, los tiempos de aceleración (transitorios) de los servomecanismos que impulsan al robot hasta que alcanza la velocidad constante, posibles rozamientos del terreno con las ruedas, etc. El problema que surge es que, por una parte, la versión de este simulador se ha desarrollado específicamente y para ser ejecutado en una máquina única, por otra parte, la complejidad de los algoritmos y el número de parámetros a manejar es de bastante importancia, ya que la simulación del movimiento se realiza buscando el equilibrio dinámico del cuerpo del robot. Lo anterior, sumado a que el desarrollo del control está distribuido, lo hacen un firme candidato para ser el banco de pruebas de una herramienta como la planteada.

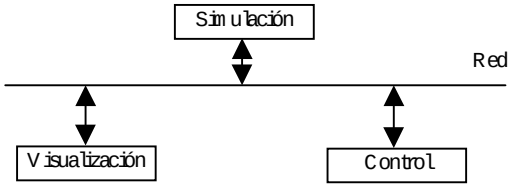

Figura 3

Los pasos a seguir, en lo que respecta a integrar el simulador de interacciones con el entorno y los elementos de control del robot, estarían en la línea de lo ya expuesto. En primer lugar, se plantea un esquema de visualización basado en un mundo virtual sobre el que se visualiza la imagen del robot, esta imagen se moverá en función de los resultados de las acciones de control y la simulación realizada en cada instante. En segundo lugar, se concreta una comunicación entre el bloque de control y el de simulación, con el objetivo de que cada decisión de control sea una entrada a tener en cuenta en el bloque de simulación, de igual manera los resultados de la simulación servirán como base para futuras acciones de control, y por supuesto todo ello se verá reflejado en el mundo virtual en el que se visualiza la escena. La relación entre los distintos elementos se puede ver en la Fig.4.

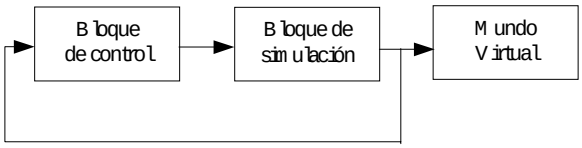

Figura 4

Según lo expuesto, la parte correspondiente al bloque de simulación es la que hay que distribuir. Debido a que en el momento actual no se ha optado por un Lenguaje Constructor de simuladores, se impone tomar las decisiones de diseño de forma manual, es decir, planteando el modelo como un conjunto de submodelos desde el principio. Para ello y teniendo en cuenta las simplificaciones que son necesarias, se plantea el bloque de simulación con, almenos, tres elementos que se pueden construir por separado y posteriormente comunicar entre sí, tal y como se observa en la Fig.5.

El bloque de control proporciona el valor de las fuerzas que van a actuar sobre el cuerpo del robot en cada instante, por ello sus valores se pasan al subsistema que calcula el rozamiento lateral (en el sentido de los ejes X e Y), y al que proporciona información sobre el hundimiento en el terreno y la pendiente que debe rebasar. En función de esos valores, se calcula la posición del centro de gravedad del robot y se le facilita al sistema de control para que realice nuevos ajustes. En el modelo no se contemplan los problemas asociados a la inercia del movimiento, ya que en el robot, la velocidad de desplazamiento y la distancia de los puntos de apoyo al centro de gravedad los hacen despreciables.

En este ejemplo, es necesario que el simulador sea lo más realista posible, ya que el bloque de control se implementa como un sistema basado en reglas, por lo que una vez determinados los parámetros de funcionamiento, debe ser posible sustituir el simulador por el robot real, sin necesidad de realizar ajustes posteriores, ya que sino no tendría sentido haber realizado el esfuerzo de generar una simulación.

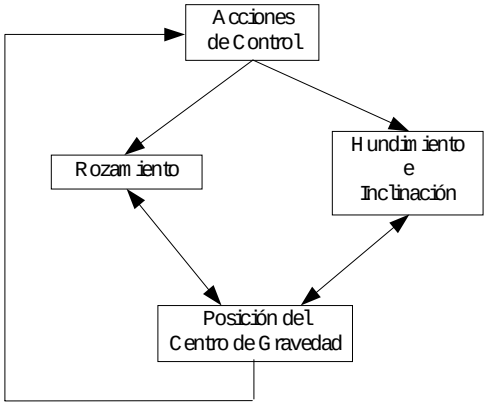

Figura 5

5. Conclusiones

A tenor de lo expuesto, puede verse la importancia que tiene, para cierto tipo de tareas, conseguir una óptima distribución de los recursos a manejar. Como es obvio, el objetivo es conseguir que los sistemas diseñados sean lo más versátiles posible, para ello es imprescindible utilizar en su desarrollo especificaciones estandarizadas y ampliamente aceptadas por la mayoría de los usuarios. Por otra parte, debido a la importancia que cobra el papel de las simulaciones en la realización de Proyectos de alto nivel, se presenta como necesario el desarrollo de una aplicación que facilite la construcción de dichos simuladores, minimizando el tiempo y los esfuerzos necesarios para su puesta en funcionamiento, ya que a menos que la propia simulación sea el objetivo del Proyecto, se suele utilizar como una herramienta de trabajo.

El estado del Proyecto de desarrollo de un Generador de Simuladores para Sistemas Distribuido, se haya en la fase de elección del Lenguaje Constructor al que hay que dotar de la capacidad de generar submodelos que sean capaces de funcionar de forma distribuida, para lo que se cuenta con los estándares citados anteriormente. De igual forma que se está en proceso de realización de un interfaz gráfico para el diseño de dichos modelos, que facilite el uso del Generador de Simuladores, aun en el caso de que el usuario no tenga conocimientos de manejo del Lenguaje Constructor.

Referencias

[1] A. Ames, D. Nadeau J. Moreland. "*The VRML 2.0 sourcebook*". John Wiley & Sons. New York. 1997. ISBN 0-471-16507.

[2] IEEE Standard Board. "*Distributed Interactive Simulation/Aplication protocol*". IEEE Standard 1278.1. IEEE Inc. 1995.

[3] J. Locke. "*An Introduction to the Internet Networking Environment and SIMNET/DIS*". Naval Postgraduated School. Monterey, CA, USA, 1995.

[4] P. Martini, M. Ruemekasten, J. Toelle. "*Tolerant Synchronization for Distributed of Interconnected Computer Network*". 11th Workshop on Parallel and Distributed Simulation (PAD'97). 1997

[5] A. S. Tanenbaum. "*Distributed Operating Systems*". Prentice-Hall. London, 1995. ISBN 0-13-143934-0

[6] A. Gómez Skarmeta, H. Martínez Barberá, P. García López. "*Una Arquitectura de Agentes Difusos para Robots Autónomos*", VIII Congreso Español de Lógica y Tecnología Fuzzy (ESTYLF'98). Pamplona, España, Sep. 1998.