JDESK: Simulador de Eventos Discreto Basado en Web [1]

Inmaculada García García, Ramón Mollá Vayá
Dpto. de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
46022 Valencia
e-mail: ingarcia@dsic.upv.es
rmolla@dsic.upv.es

Resumen

JDESK es un simulador generalista de eventos discreto que permite simular y obtener resultados de un modelo implementado en código Java. Es un simulador basado en web, por lo que puede utilizarse desde cualquier navegador. El modelo de simulación obtenido puede ejecutarse de forma local en cualquier plataforma. JDESK es un simulador versátil y con tiempos de simulación bajos. Los modelos de simulación se implementan definiendo los componentes del modelo y su interconexión. Por la facilidad de creación y modificación del modelo simulado puede usarse como prototipador. JDESK permite simular cualquier modelo y no impone restricciones al modelo a simular. Permite definir cualquier tipo de comportamiento del sistema, por caprichoso que sea. El simulador incluye un asistente para principiantes, que permite crear modelos de forma intuitiva. Incluye también bibliotecas de componentes del modelo y de ejemplos de modelos completos, que permiten guiar al alumno en el proceso de creación del modelo del sistema a simular.

1. Introducción

La simulación es una herramienta poderosa en el proceso de aprendizaje. El aprendizaje basado en simulación se define como "aprender haciendo" [11]. Permite al estudiante seleccionar experiencias y obtener sus propias respuestas. La simulación basada en web no es un campo nuevo dentro de la simulación, sino la necesidad de adaptar el campo de la simulación a las nuevas tecnologías [3]. Gran parte de los simuladores basados en web son versiones web de otros simuladores existentes.

La simulación puede utilizarse para resolver cualquier problema que pueda modelarse como un sistema de eventos discretos [13]. Algunas áreas de aplicación son [1]: procesos de producción, ingeniería de la construcción, diseño de semiconductores, aplicaciones militares, procesos de transporte y distribución, procesos de gestión,…

La simulación basada en web está permanentemente disponible desde cualquier navegador, lo que facilita el acceso de los estudiantes desde los laboratorios o incluso desde su propia casa, lo cual es una ventaja respecto a la simulación tradicional. Algunos simuladores disponen de bibliotecas y repositorios de modelos propios o aportados por los usuarios, lo que permite a los estudiantes crear modelos más complejos de forma sencilla o aprender a modelar sistemas mediante ejemplos. El simulador no debe estar disponible en local, lo que facilita su posibilidad de utilización y permite que esté siempre actualizado.

Una ventaja importante de la simulación en web es que permite construir simulaciones distribuidas (Distributed Interactive Simulation [9]), donde varios usuarios interaccionan. Los estudiantes pueden crear equipos de trabajo con estudiantes en diferentes ubicaciones, lo que enriquece su aprendizaje.

Existe un gran número de simuladores de eventos discretos basados en Web [12], como JSIM, SILK o SIMJAVA. JSIM [16] [17] es un simulador basado en web escrito en Java de código fuente abierto [6] [7]. Permite definir el

[1] Este trabajo ha sido financiado por la OCYT de la Generalitat Valenciana bajo el proyecto de investigación CTIDIB/2002/344.

modelo de forma gráfica, conectando nodos de diferente tipo. Incluye animaciones durante la ejecución de la simulación. Permite definir el modelo de forma modular y reutilizar componentes. SILK [4] es un simulador generalista escrito en Java como una biblioteca. Los modelos se escriben directamente en Java utilizando una biblioteca de clases. Permite definir los modelos de forma gráfica y modular, aunque también de forma textual. SIMJAVA [5] es un simulador de eventos discreto basado en SIM++ de código abierto [8]. Permite representar los objetos de la animación como iconos animados. Una simulación es un conjunto de entidades ejecutándose independientemente y comunicándose mediante eventos.

Entre los simuladores no basados en web hay dos simuladores utilizados en docencia y con dos filosofías completamente diferentes, que los hacen especialmente interesantes por su diferente forma de descripción del modelo y operación. SMPL [14] es una librería de C, lo que permite utilizar todo el potencial de este lenguaje en la implementación de los modelos. Destaca por su velocidad de simulación, pero su gran inconveniente es que la definición del modelo es complicada y trabajosa, pues supone definir el modelo como la secuencia de todos los posibles eventos del sistema. Deben definirse los algoritmos de planificación, el control de tiempos,... Modificar y depurar el modelo es complicado. No permite modelar cualquier sistema: existe un valor límite del número de clientes y de Estaciones de Servicio (ES) [2] en el modelo. QNAP [18] es un simulador implementado como un lenguaje propio. Definir un modelo en este simulador es sencillo y rápido, pues únicamente deben definirse las ES del modelo, sus propiedades y su interconexión. Modificar y depurar el modelo es rápido y sencillo. Los algoritmos de planificación y el control de tiempos los realiza el simulador automáticamente. No impone limitaciones en el modelo a simular. Por contra, el tiempo de simulación es entre 4 y 6 veces mayor que SMPL.

2. Objetivos

JDESK [10] se ha diseñado para aunar las siguientes características, que le permiten adaptarse a las necesidades de la docencia, integrando lo mejor de cada simulador:

- Creación rápida y fácil del modelo de simulación, El modelo del sistema es una descripción de los elementos que lo componen, siguiendo el estilo de modelado de sistemas de QNAP.
- Fácil depuración y modificación, debido al estilo de modelado similar a QNAP.
- Multiplataforma, pues el modelo es código escrito en Java.
- El modelo debe estar implementado en un lenguaje fácil de aprender y que ofrezca la posibilidad de integrar elementos externos a la simulación. En JDESK el modelo de simulación está implementado en Java.
- Posibilidad de simular cualquier modelo, incluso con comportamientos no convencionales. SMPL permite la implementación de cualquier comportamiento del sistema, pero con un coste de implementación muy elevado. JDESK permite implementar comportamientos caprichosos de forma sencilla y modular.
- Creación modular del modelo de simulación:
  - Reutilización de componentes de otros modelos o del propio modelo, debido a la utilización de objetos de Java.
  - Soporte a simulación distribuida, permitiendo crear equipos de alumnos trabajando sobre un mismo modelo, por la utilización de Java como lenguaje de implementación del modelo.
  - Creación de un repositorio de ejemplos y componentes de modelos, mediante bibliotecas escritas en Java.
  - Compartir el conocimiento. Los modelos escritos por un usuario pueden ser utilizados por otros usuarios como parte del sistema simulado.

Los simuladores basados en web no reúnen todas las características requeridas para su utilización en docencia. Estos simuladores permiten definir los modelos a simular de una forma más o menos intuitiva, dependiendo del simulador en concreto. Los simuladores más complejos ofrecen completos manuales de usuario. Permiten simular sistemas con comportamientos predeterminados que pueden ser adecuados para simular la mayor parte de los sistemas. Si embargo, no ofrecen el marco para

simular de forma fácil e intuitiva sistemas con comportamientos caprichosos.

El objetivo principal JDESK fue crear un simulador apto para su uso en cualquier entorno, y especialmente en el campo de la docencia. Aúna todas las características deseables en un simulador dedicado a la docencia. En cuanto a la forma de operatividad y velocidad, el diseño de JDESK se basó en SMPL y QNAP. JDESK permite definir el modelo siguiendo el estilo marcado por QNAP (definiendo únicamente las ES y su interconexión), de forma fácil y rápida. La velocidad de simulación supera con creces la obtenida por SMPL en la mayor parte de los modelos simulados.

En los siguientes apartados se da una visión general de la forma interna de operación de JDESK y de su utilización por parte del usuario.

3. Características de JDESK

JDESK es un simulador generalista de eventos discretos orientado a eventos. Está implementado en Java, lo que lo hace especialmente apropiado para su ejecución vía web [15].

Permite crear modelos describiendo la topología del sistema y las características de cada uno de sus elementos. Los modelos, por tanto, son fáciles de depurar y modificar. Esta característica lo hace adecuado para cualquier etapa del proceso de simulación: como prototipador o para obtener resultados finales.

JDESK incluye un asistente que permite guiar al alumno en el proceso de creación del modelo. Para modelos con comportamientos usuales o para usuarios no expertos se recomienda el uso del asistente. El asistente está implementado como un Applet de Java.

Los modelos puede describirse utilizando orientación a objetos y metodologías top-down, lo que hace posible la reutilización de partes del modelo en el propio sistema simulado o la creación de bibliotecas de componentes de modelos de simulación. La posibilidad de crear bloques de simulación contenidos en bibliotecas permite reutilizar el esfuerzo de otras personas en la definición de modelos y, además, crear simulaciones distribuidas entre un equipo de personas, de forma que los módulos desarrollados por los diferentes componentes del equipo puedan enlazarse para obtener la simulación de un sistema complejo.

El simulador incluye una serie de comportamientos predefinidos de sistemas, de forma que cualquier sistema convencional pueda modelarse fácil y rápidamente. También ofrece al usuario la posibilidad de definir comportamientos más sofisticados; por ejemplo, políticas de planificación no convencionales que incluso varíen dependiendo de las características del cliente actual o del estado del sistema. El control de la simulación puede realizarse en cualquier parte del sistema, dependiendo de cualquier evento. Las características del modelo, e incluso su topología, pueden variar dinámicamente en tiempo de simulación. Pueden crearse o destruirse ES dinámicamente, cambiar su interconexión o sus características. Permite la utilización de características avanzadas de modelos como semáforos, recursos o creación de hijos. En resumen, JDESK ofrece el marco para simular cualquier sistema que el usuario sea capaz de definir.

Las características del simulador lo hacen especialmente adecuado para realizar simulaciones en tiempo real, pues el usuario puede conocer en todo momento que es lo que está ocurriendo en el sistema. Esta facilidad también permite depurar sistemas complejos o demostrar como funciona el sistema simulado a los alumnos.

En JDESK los modelos se definen escribiendo de forma textual el modelo del sistema a simular en Java. El asistente de JDESK automatiza el proceso de escritura del modelo, pero el resultado siempre es un fichero java. Si el usuario ha definido el modelo de forma correcta, el simulador permite obtener el fichero Java del modelo para ejecutar la simulación en local. Las ventajas de escribir el modelo en Java para el proceso de aprendizaje son las siguientes:

- El estudiante no debe aprender un lenguaje específico para la simulación.
- Pueden utilizarse funciones de Java en la simulación no pertenecientes al simulador. El estudiante puede utilizar objetos implementados para otros propósitos en la implementación del modelo.
- La principal ventaja de utilizar Java como lenguaje de implementación del modelo es que el código implementado en Java es

multiplataforma. Para su utilización en enseñanza esta característica tiene grandes ventajas, pues funciona independientemente de la ubicación de los alumnos. El profesor puede implementar el modelo a simular y tener la certeza de poder usarlo para demostrar a los alumnos el funcionamiento del modelo simulado, independientemente de que tenga que usarlo en diferentes plataformas. No es necesario volver a acceder al simulador.

4. Funcionamiento del Núcleo de JDESK

Un modelo en JDESK está basado en dos entidades básicas: ES y clientes. Estas son las estructuras con las que el programador construye los modelos de simulación. Una ES modela una parte del sistema real que proporciona un determinado servicio. Está compuesta de una serie de servidores y una cola de clientes esperando ser servidos. Toda la información del modelo la contienen las ES. Los clientes son elementos de la simulación que viajan a través del sistema, cambiando su estado y el de las ES que atraviesan. Los clientes se pueden generar automáticamente mediante fuentes o bien el programador puede crearlos explícitamente.

La dinámica del sistema la modelan las ES que componen el modelo y la cola de eventos.

La cola de eventos es una estructura interna de JDESK y no visible al usuario. Está compuesta de clientes. Los clientes sirven de soporte para modelar los eventos del sistema. En JDESK sólo hay un posible evento: un cliente debe abandonar la ES donde está recibiendo servicio porque ha finalizado su tiempo de servicio. La cola de eventos está ordenada por el tiempo de finalización de servicio de los clientes.

La simulación comienza porque un cliente entra en una ES (figura 1 a). Si la ES tiene un servidor libre, comienza a recibir servicio (figura 1 b). Se genera un evento de salida del cliente al cabo de un tiempo determinado (figura 1 c).

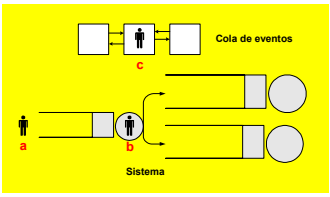

Figura 1. El cliente entra en la ES

Hasta este momento la simulación ha estado controlada por la ES donde ha entrado el cliente y ahora el control pasa a la cola de eventos. La cola de eventos ejecuta el siguiente evento, eliminándolo de la cola (figura 2 a). Sólo hay un posible evento: el fin del tiempo de servicio de un cliente. La cola de eventos invoca a la ES correspondiente para que libere al cliente. El cliente deja la ES y entra en otra ES o sale del sistema. El control de la simulación pasa a la ES destino. En la ES destino vuelve a comenzar el proceso. Si a la llegada de un cliente a la ES, no hay servidores libres, se inserta en cola de espera y no se genera ningún evento (figura 2 b).

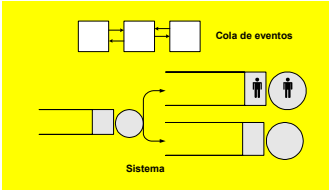

Figura 2. El cliente sale de la ES

Cuando no hay eventos en la cola de espera la simulación termina (aunque ha podido terminar con anterioridad por cualquier otra causa: ha finalizado el tiempo de simulación definido por el usuario o explícitamente el usuario ha decidido finalizar, dependiendo del estado del sistema o de algún evento en concreto).

El núcleo de simulación recorre la cola de eventos, extrayendo el evento en cabeza (es una cola ordenada por tiempo) e invocando a la ES correspondiente (figura 3). El control de la simulación pasa alternativamente de las ES a la cola de eventos.

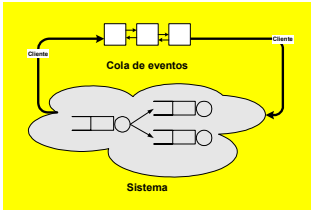

Figura 3. Dinámica del sistema

5. Modelado de Sistemas con JDESK

El modelo de un sistema en JDESK se crea, básicamente, definiendo las ES que componen el modelo.

5.1. Implementación de ES

JDESK define tres tipos de ES:

- Fuentes: tienen como objetivo principal generar clientes dinámicamente. La cadencia con la que la fuente genera clientes y las propiedades de estos, los marca el programador al definir la fuente. Una vez los clientes se crean, dejan la fuente para encaminarse a otra ES.
- Servidores: está compuesto de una cola de espera y una serie de servidores. Cuando un cliente entra en una ES, si no hay un servidor libre, espera en la cola de la ES. Si el cliente entra en un servidor, permanecerá en él recibiendo servicio durante un tiempo determinado.
- Recursos: un recurso en JDESK aúna recursos y semáforos. Un recurso está compuesto por una cola de espera y una serie de recursos que puede reservar el cliente. Si a la llegada del cliente, el número de recursos demandados no está disponible, el cliente espera en la cola a que se liberen recursos suficientes. Si los recursos demandados por el cliente se le pueden asignar, el cliente continúa fluyendo libremente por el sistema. El cliente puede liberar los recursos en cualquier momento.

El comportamiento de la ES se define mediante una serie de parámetros. Estos parámetros varían dependiendo del tipo de ES. En la tabla 1 se muestran los parámetros de la ES, su tipo y cuales de ellos están presentes en la declaración de cada ES. Entre estos parámetros hay:

- Constantes: el valor del parámetro debe obligatoriamente seleccionarse entre una serie de valores constantes.
- Variables: permite cualquier valor del tipo correcto.
- Funciones: se debe pasar como parámetro una función previamente definida. Existen funciones de biblioteca en JDESK para algunas de estas funciones.

PARÁMETROS DE LA ES	SERVIDOR	RECURSO	FUENTE	TIPO DE PARÁMETRO
Nombre	✓	✓	✓	String (variable)
Número de servidores o recursos	✓	✓		Valor (variable)
Política de servicio preemtiva	✓			Constante
Tiempo de servicio demandado por los clientes	✓	✓	✓	Función
Algoritmo de planificación (inserción en cola de espera de la ES)	✓	✓		Función
Algoritmo de planificación (reinserción en cola de espera de la ES)	✓			Función
Número de recursos demandados por el cliente		✓		Función
Algoritmo de encaminamiento de clientes a la salida de la ES	✓	✓	✓	Función
Función de usuario	✓	✓		Función

Tabla 1. Parámetros de definición de ES

El comportamiento de las ES se define en su mayor parte por funciones. Cada una de estas funciones tiene un cometido y se pone en funcionamiento en un instante determinado de la simulación. El usuario puede implementar las funciones o bien utilizar funciones de biblioteca suministradas por JDESK.

Para usuarios inexpertos o modelos con comportamientos típicos, es recomendable usar funciones de biblioteca. Cuando el sistema tiene un comportamiento inusual se debe implementar este comportamiento mediante estas funciones. Las funciones de comportamiento dan potencia y flexibilidad al simulador.

Pueden utilizarse, además de para su cometido específico, para cualquier otro que el usuario considere apropiado.

Permiten variar dinámicamente el sistema: su topología, su estructura o sus características. También son estas funciones las que permiten hacer simulaciones en tiempo real, incluir multimedia, alarmas,...

La función de tiempo de servicio obtiene el tiempo que un cliente determinado debe recibir servicio en la ES actual. El tiempo de servicio asignado podría depender de su estado, de valores aleatorios, de distribuciones temporales (las principales funciones de distribución temporal están implementadas en el simulador como funciones de biblioteca: uniforme, exponencial, erlang, hiperexponencial y normal).

La función de planificación para inserción en cola de espera permite insertar en cola de espera de la ES un cliente que no puede servirse porque todos los servidores están ocupados. Las principales políticas de servicio están implementadas en el simulador como funciones de biblioteca: lifo, fifo, según prioridades y las versiones preemptivas de estas políticas.

La función de planificación para reinserción en cola de espera (sólo para ES preemtivas), con la llegada de un cliente más prioritario, el cliente que está recibiendo servicio actualmente deja de recibir servicio y se reinserta en la cola de espera. El hecho de diferenciar entre inserción y reinserción permite dotar de una mayor flexibilidad al simulador.

La función de encaminamiento define la ES donde transitará el cliente cuando abandone la ES actual. El conjunto de las funciones de encaminamiento define la topología del sistema. Usar funciones para definir la topología del sistema permite variarla dinámicamente.

La función de obtención del número de recursos permite obtener el número de recursos que solicita un cliente cuando llega a una ES tipo recurso. El número de recursos demandados podría modificarse dinámicamente dependiendo de algún criterio.

La función de usuario no es necesaria para el funcionamiento del simulador, pero, se ha incluido en JDESK, pues dota al simulador de una gran flexibilidad. Esta función se ejecuta cuando un cliente comienza a recibir servicio en una ES. Puede servir para hacer trazas, para simulaciones en tiempo real,... es decir, para cualquier comportamiento atípico.

Estas funciones son métodos de determinadas clases definidas a tal efecto.

5.2. Pasos en la creación del modelo

Para definir un modelo en JDESK se deben seguir los siguientes pasos:

- Inicializar el simulador
- Definir las ES del modelo:
  - Implementar las funciones de comportamiento de cada ES o usar funciones predefinidas o implementadas para otras ES.
  - Determinar las características de la ES.
- Introducir clientes en el sistema. Para sistemas cerrados o para sistemas abiertos con comportamientos determinados, es necesario comenzar la simulación con clientes en el sistema. La creación de un cliente supone insertarlo en una ES.
- Comenzar la simulación. Se puede indicar la forma de finalización de la simulación. La forma más usual de finalizar la simulación es indicar el tiempo máximo de la simulación. También, permite comenzar a obtener medidas de la simulación en un instante determinado para eliminar estados transitorios.
- Obtener resultados de la simulación.

5.3. Clases de JDESK para la Simulación

JDESK proporciona al usuario biblioteca de clases para la simulación. Los métodos de estas clases

permiten: control de la simulación (simulación, inicialización, finalización, inicio de contabilidad y gestión de errores), creación de ES de cada uno de los tipos, gestión de hijos, gestión de recursos y contabilidad. Estos métodos pueden usarse dentro de las funciones de comportamiento de la ES con el objeto de dotar al sistema de comportamiento específicos.

5.4. Utilización de JDESK

JDESK es un simulador basado en texto, por lo que el modelo simulado es código escrito en Java con funciones de biblioteca específicas del simulador.

JDESK contiene un editor de texto que permite escribir el código del modelo de simulación. Permite editar modelos creados anteriormente o ejemplos de modelos completos del repositorio de JDESK. El código debe tener el formato de un programa principal de Java.

Para usuarios inexpertos es aconsejable usar el asistente (figura 4). El asistente es un Applet de Java que permite crear modelos de forma rápida y sencilla mediante controles, escribiendo únicamente algunas funciones de comportamiento. El asistente construye el código Java del modelo de simulación a partir de las especificaciones del usuario.

Figura 4. Asistente

El asistente permite:

- Definir los parámetros generales de la simulación, como tiempos de inicio y final de la simulación.
- Añadir funciones al modelo. El asistente permite ver ejemplos de funciones de la biblioteca de JDESK. Si el usuario elige escribir directamente la función, aparece una ventana de edición con un esqueleto de la función (en forma de método de la clase correspondiente), de forma que el usuario sólo debe escribir el código estrictamente necesario. Esta función se podrá asignar posteriormente a una ES.
- La inserción de una ES en el modelo simulado utilizando el asistente se realiza utilizando los controles de la ventana de inserción de ES (figura 5). El asistente permite seleccionar el tipo de ES y según el tipo seleccionado habilita únicamente los controles correspondientes. Permite definir las funciones de comportamiento de la ES: seleccionándolas entre las previamente definidas, para ésta o para otras ES, o entre las funciones de biblioteca o bien escribir la función directamente. Si se escribe la función muestra un esqueleto de la clase correspondiente.
- Insertar clientes: permite crear clientes con unas determinadas características e insertarlos en una ES previamente definida.
- Obtener resultados: permite indicar que resultados de la simulación se desea obtener y en que formato.

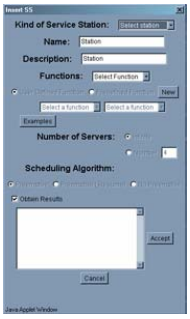

Figura 5. Creación de una ES utilizando el asistente

El código creado mediante el asistente aparece en el editor de texto y puede ser modificado para ajustar o variar su comportamiento.

Una vez se ha definido el modelo, se compila utilizando para ello un control de JDESK. Si la compilación produce errores se mostrarán los

mensajes correspondientes en la ventana de mensajes. Si la compilación tiene éxito, el fichero conteniendo el modelo de simulación queda disponible para que el usuario lo ejecute de forma local. Este modelo es un fichero Java.

6. Conclusión

JDESK es un simulador generalista de eventos discreto escrito en Java. El modelo de simulación se implementa describiendo la topología del sistema: las entidades del sistema, su comportamiento y su interconexión. JDESK facilita la definición, depuración y modificación de modelos, lo que lo hace especialmente apropiado para su utilización en docencia. JDESK no impone restricciones en el modelo a simular y permite definir cualquier comportamiento del sistema. Permite incluir elementos externos dentro de la simulación, como multimedia o alarmas y simulación en tiempo real. Las características del sistema pueden variar dinámicamente, incluso su topología. Permite crear o destruir dinámicamente elementos del modelo de simulación.

JDESK incluye un asistente para ayudar a los estudiantes a definir modelos con comportamientos típicos, de forma fácil y rápida. El asistente guía al alumno durante todo el proceso de creación del modelo, permitiéndole definirlo controles.

El modelo de simulación está implementado en Java, por lo que, una vez compilado por JDESK, puede ser ejecutado en diferentes plataformas. Además, el alumno no debe aprender un lenguaje específico para utilizar el simulador.

JDESK es un simulador basado en web, lo que permite su acceso desde cualquier navegador. Tanto alumnos como profesores pueden acceder a JDESK desde laboratorios o desde su propia casa.

Referencias

[1] Coos, R. *Simulación: un enfoque práctico*. Limusa, Mexico.1992

[2] Fishman, G.S. *Conceptos y Métodos en la Simulación Digital de Eventos Discretos*. Limusa, 1978.

[3] Fishwick, P.A. *Web-Based Simulation: Some Personal Observations*. 28th Winter simulation conference, California, 1996.

[4] Healy, K.J. Kilgore, R.A. *Introduction to Silk and Java-Based Simulation*. 30th Winter simulation conference.1998.

[5] Howell, F. McNab, R. *Simjava: a Discrete Event Simulation Package for Java with Applications in Computer Systems Modelling*. First International Conference on Web-based Modelling and Simulation. 1998.

[6] http://chief.cs.uga.edu/~jam/jsim/

[7] http://chief.cs.uga.edu/~jam/home/theses/huang_thesis/userguide/userguide.html

[8] http://www.dcs.ed.ac.uk/home/hase/simjava/

[9] http://www.sei.cmu.edu/publications/articles/arch-dist-int-sim.html

[10] http://www.sig.upv.es/proyectos/simulacion/JDESK.htm

[11] Kindley, R. *The Power of Simulation-Based e-Learning*. The e-Learning Developer's Journal. 2002. www.eLearningGuild.com

[12] Kuljis, J. Paul, R.J. *A Review of Web Based Simulations: Whither we wander?* 2000 Winter Simulation Conference.

[13] Law, A.M. Kelton, W.D. *Simulation Modeling and Analysis*. McGraw-Hill Series in Industrial Engineering and Management Science. 1982.

[14] MacDougal, M.H. *SMPL - A Simple Portable Simulation Language*. Amdahl. 1980.

[15] McNab, R. Howell, F.W. *Using Java for Discrete Event Simulation*. 12th UK Computer and Telecommunications Performance Engineering Workshop (UKPEW), 1996.

[16] Miller, J.A. Nair, R.S. Zhang, Z. Zhao, H. *JSIM: A Java-Based Simulation and Animation Environment*. 30th Annual Simulation Symposium. 1997.

[17] Miller, J.A. Seila, A.F. Xiang, X. *The JSIM Web-Based Simulation Environment. Future Generation Computer Systems*. 17-2. 2000.

[18] *QNAP: queuing network analysis package*. 1st Int. Conf. on the numerical Solutions of Markov chains. 1990.