EOSimulator: biblioteca de simulación para un curso de simulación a eventos discretos

S. Alaggia, A. Mauttone, M. Urquhart

Depto. Investigación Operativa, Instituto de Computación,
Facultad de Ingeniería, Universidad de la República
J. Herrera y Reissig 565, Montevideo, Uruguay
{salaggia, mauttone, urquhart}@fing.edu.uy

Marzo 2007

Palabras claves: Simulación, C++, Modelado

Resumen

En este trabajo se presenta la primera versión del paquete EOSimulator, lenguaje de simulación que se utiliza en el curso Simulación a Eventos Discretos que dicta el Departamento de Investigación Operativa del InCo. Se comenta la experiencia obtenida en sus dos primeros años de utilización, se detallan los principios de diseño, las mejoras incorporadas en las distintas versiones y futuros cambios previstos para próximas versiones del paquete.

El curso Simulación a Eventos Discretos utiliza el libro Simulation Modelling with Pascal de Ruth Davies, Robert O'Keefe (1989), como libro de cabecera, que incluye el paquete Pascal_Sim desarrollado por los mismos autores como lenguaje de simulación.

EOSimulator se desarrolla según los siguientes objetivos: a) mantener en lo posible la metodología de enseñanza de Davies y O´Keefe y la simplicidad de PascalSim, b) incorporar a los laboratorios la posibilidad de un lenguaje de simulación orientado a objetos simple con tecnología actualizada de ingeniería de software.

1 Introducción

Simulación a Eventos Discretos es una materia electiva de la carrera Ingeniería en Computación del InCo, se sitúa a partir del semestre 6 y pertenece al área de Investigación Operativa. La asignatura tiene como objetivo introducir al estudiante en el modelado de sistemas, técnicas de experimentación con el modelo y obtener y analizar resultados. La evaluación consiste en un trabajo obligatorio que se aprueba (o no) sin nota, y una única prueba escrita que define la nota final del estudiante. El obligatorio es eliminatorio y consiste en dos entregas.

El curso está fuertemente basado en el libro *Simulation Modelling with Pascal* de Ruth Davies, Robert O'Keefe de 1989 en general y en *Simulation modelling and analysis* de Averill M. Law y W. David Kelton de 1991 para la parte de estadística. La herramienta que se brinda a los estudiantes para que implementen los obligatorios es la biblioteca Pascal_Sim que viene con el libro de cabecera.

El libro da una visión integradora de los conocimientos teóricos y su implementación práctica mediante Pascal_Sim, biblioteca que está implementada de forma sencilla y robusta, lo que favorece su uso y comprensión. Además provee un "marco de desarrollo" intuitivo. Sin embargo el equipo del curso se planteó el desarrollo de un nuevo software, llamado EOSimulator, y lo ha puesto a prueba en las dos últimas ediciones del curso (años 2005 y 2006). Las limitaciones que se trataron de resolver fueron: incorporar herramientas de desarrollo actualizadas, sorteo de números aleatorios más flexible y potente; creación de nuevos tipos de entidades, facilidades en la extensión de propiedades de las entidades sin necesidad de recompilar el modelo. El estudiante tiene la opción de elegir el lenguaje a utilizar durante el curso.

En este trabajo se presentan el software EOSimulator y su aplicación en las ediciones 2005 y 2006 del curso "Simulación a eventos Discretos". En la sección 2 se describen los principios de diseño sobre los cuales se desarrolló a EOSimulator. En la sección 3 se muestra y explica brevemente el diseño de la biblioteca. Luego, en la sección 4, se presentan los resultados obtenidos con la aplicación de la biblioteca en el curso y las mejoras que se realizaron a partir de dicha aplicación. Finalmente se plantean líneas para el trabajo futuro en el desarrollo de la biblioteca, su implantación en curso y posibles mejoras al mismo.

2 Principios de Diseño

A la hora de crear una herramienta para el curso de Simulación a Eventos Discretos se buscó un conjunto de características que permitieran que ésta se adaptara al enfoque del curso y que fuera de fácil manejo por parte de los estudiantes. Debido a que el curso está fuertemente basado en el libro de Davies y O'Keefe y por ende en Pascal_Sim, se trató de que la forma de uso de ambas bibliotecas fuera similar. Se trató de diseñar una herramienta que adquiriera todas las buenas cualidades de Pascal_Sim y mejore sus debilidades. Nos basamos en las siguientes características.

- Enfoque a eventos 2 y 3 fases: La herramienta debe soportar el enfoque a eventos en 2 y 3 fases, que es el tratado en el curso. Si bien actualmente la mayor parte de las herramientas libres disponibles se basan en la orientación a procesos, no es la idea cambiar la metodología del curso.
- Interfases similares: Las interfases que brinda Pascal_Sim para las diferentes funcionalidades están claramente definidas. Es importante mantener interfases similares en la nueva herramienta de modo que se adapte al curso.

- Separar nociones de Modelo y Experimento: La noción de Modelo y Experimento viene heredada de dos fuentes: la biblioteca DesmoJ [DMJ05] y el Formalismo Devs [Wai03]. La idea es separar el modelo que es construido para cada sistema de las estructuras de datos y algoritmos necesarios para simular. Esto nos permite crear motores de simulación más eficientes sin cambiar las interfases de modelado que son utilizadas por los estudiantes.
- Número de torrentes ilimitado: El hecho de tener un número ilimitado de torrentes es necesario para implementar correctamente técnicas de reducción de varianza y un gran número de replicaciones.
- Extensión de entidades: Las entidades varían de un sistema a otro. Tener la capacidad de definir entidades diferentes para cada sistema es necesario.

3 Diseño de EOSimulator

A partir de los principios definidos en el punto anterior se diseñó EOSimulator. Esta biblioteca implementada en C++ soporta el enfoque de eventos en dos y tres fases. Está basada en Pascal_Sim [DO89] así como en la biblioteca de simulación DesmoJ [DMJ05]. Se trató que ofreciera funcionalidades similares a las de Pascal_Sim con algunas mejoras, incluyendo aquellas presentadas en el punto anterior.

3.1 Macro Arquitectura

EOSimulator se divide en 5 módulos cada uno conteniendo un conjunto de funcionalidades bien diferenciadas. Estos módulos son: `Core`, `Dist`, `Graphic`, `Statics` y `Utils`. A continuación se muestra un diagrama de las dependencias de estos módulos.

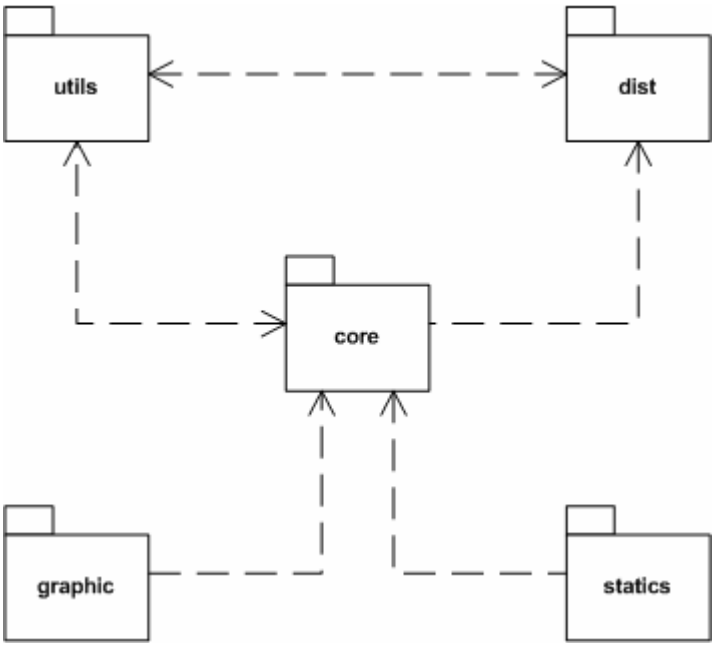

Figura 1 - Macro Arquitectura de EOSimulator

El módulo `Core` encapsula el motor de simulación y las clases utilizadas para el modelado de sistemas. El motor se encuentra básicamente representado por la clase `Experiment` que es donde tiene el

calendario y el ejecutivo. Este módulo también contiene las clases abstractas de modelado como entidades y eventos, siendo *Model* la clase principal. De este modo se separa efectivamente las estructuras de datos y algoritmos necesarios para correr una simulación de las otras utilizadas para modelar sistemas.

El módulo *Dist* contiene todo el manejo de números aleatorios. Se implementan las distribuciones comúnmente utilizadas en simulación y un generador de números aleatorios. El diseño de la biblioteca permite incorporar nuevos generadores y distribuciones fácilmente.

En el módulo *Graphic* se maneja la salida gráfica. EOSimulator brinda visualización gráfica icónica en 2D, utilizando varias bibliotecas auxiliares, todas ellas de código abierto.

El módulo *Statics* contiene los acumuladores de datos para las variables de salida. Estos acumuladores son histogramas; se implementan tres tipos: series de tiempo, observaciones y tiempo ponderados.

Finalmente el módulo *Utils* contiene las colecciones que se utilizan en la biblioteca. Se proveen diversos tipos de contenedores.

3.2Módulo Core

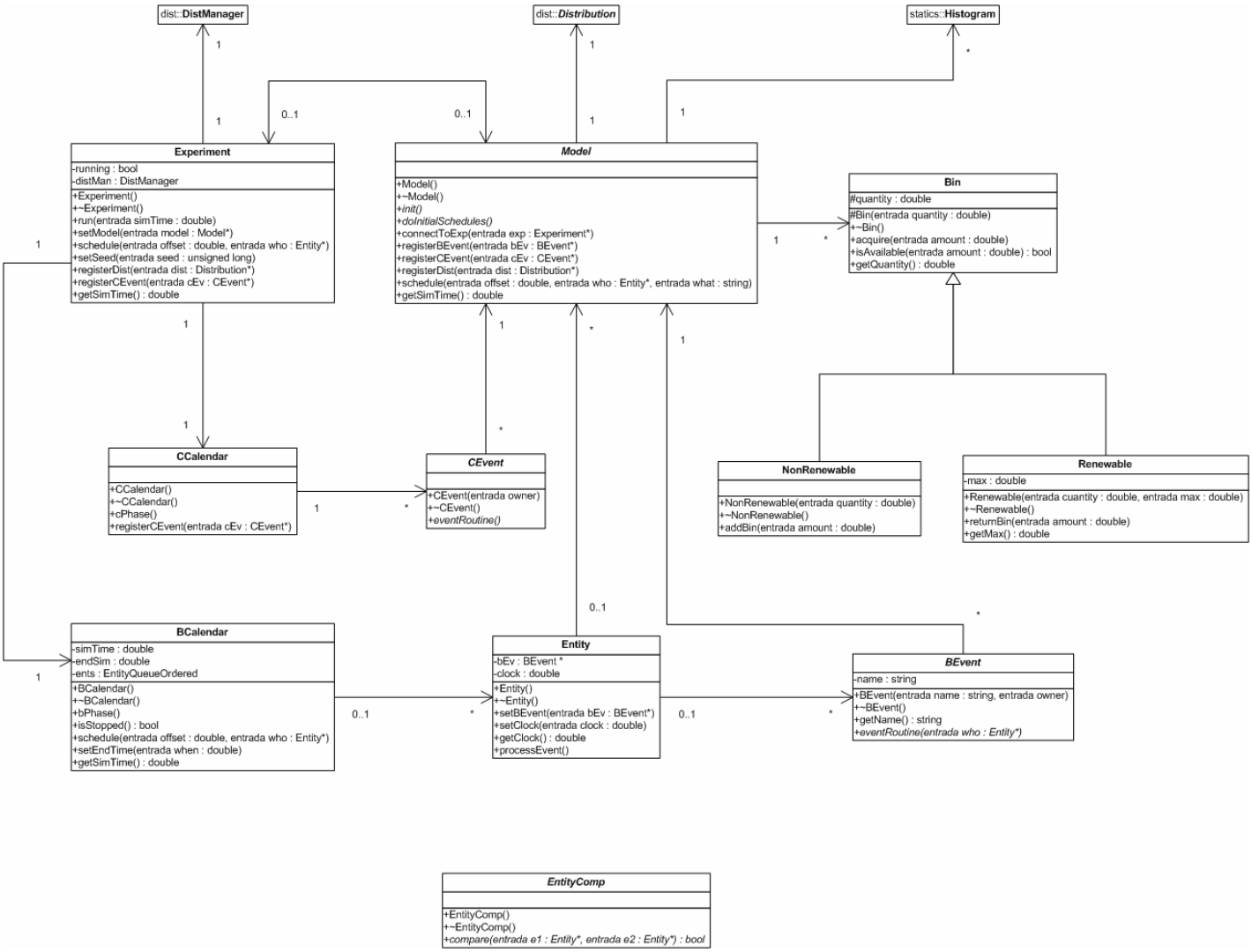

Figura 2 - El Módulo Core

La principal responsabilidad de este módulo es definir el motor de simulación y las clases para el modelado de sistemas.

La clase principal del motor es `Experiment` y se encarga de correr simulaciones. Para ello se manejan dos calendarios: para los eventos fijos `BCalendar` y para los eventos condicionales `CCalendar`. Para manejar los torrentes de números pseudoaletorios se utiliza a `dist::DistManager`. `BCalendar` maneja el tiempo y las entidades que estén asignadas a un `BEvent` (evento fijo), mientras que `CCalendar` contiene a todos los `CEvents` (evento condicional) registrados y los ejecuta en cada salto de tiempo. Finalmente, `DistManager` maneja los torrentes de todas las distribuciones registradas.

Las clases para el modelado de sistemas son abstractas. Estas son *Model* (representa el modelo de simulación), `BEvent`, `CEvent` (eventos fijos y condicionados), `Entity` (tipo primitivo de entidad) and `EntityComp` (comparador de entidad). Ver [EOS06].

3.2.2 Restricciones de Diseño

- Las entidades se encuentran agendadas a un evento (en `BCalendar`), esperando en una cola, y/o pertenecer al modelo (entidades globales). Esta restricción es necesaria para el manejo de memoria y es responsabilidad del programador.
- Las relaciones entre `Model`, `Bins`, `CEvents`, `Distributions`, `Histograms` y `Entity` (globales) deben ser implementadas mediante atributos: `Model` tiene un atributo por cada uno de ellos. Es responsabilidad del programador mantener esta restricción.

3.2.3 Restricciones de Implementación

- Las entidades deben ser creadas en forma dinámica, mientras que `BEvents`, `Bins`, `CEvents`, `Distributions` and `Histograms` pueden ser creados en forma dinámica o estática. Ver [EOS06]. El cumplimiento de esta restricción es responsabilidad del programador.

3.3Módulo Dist

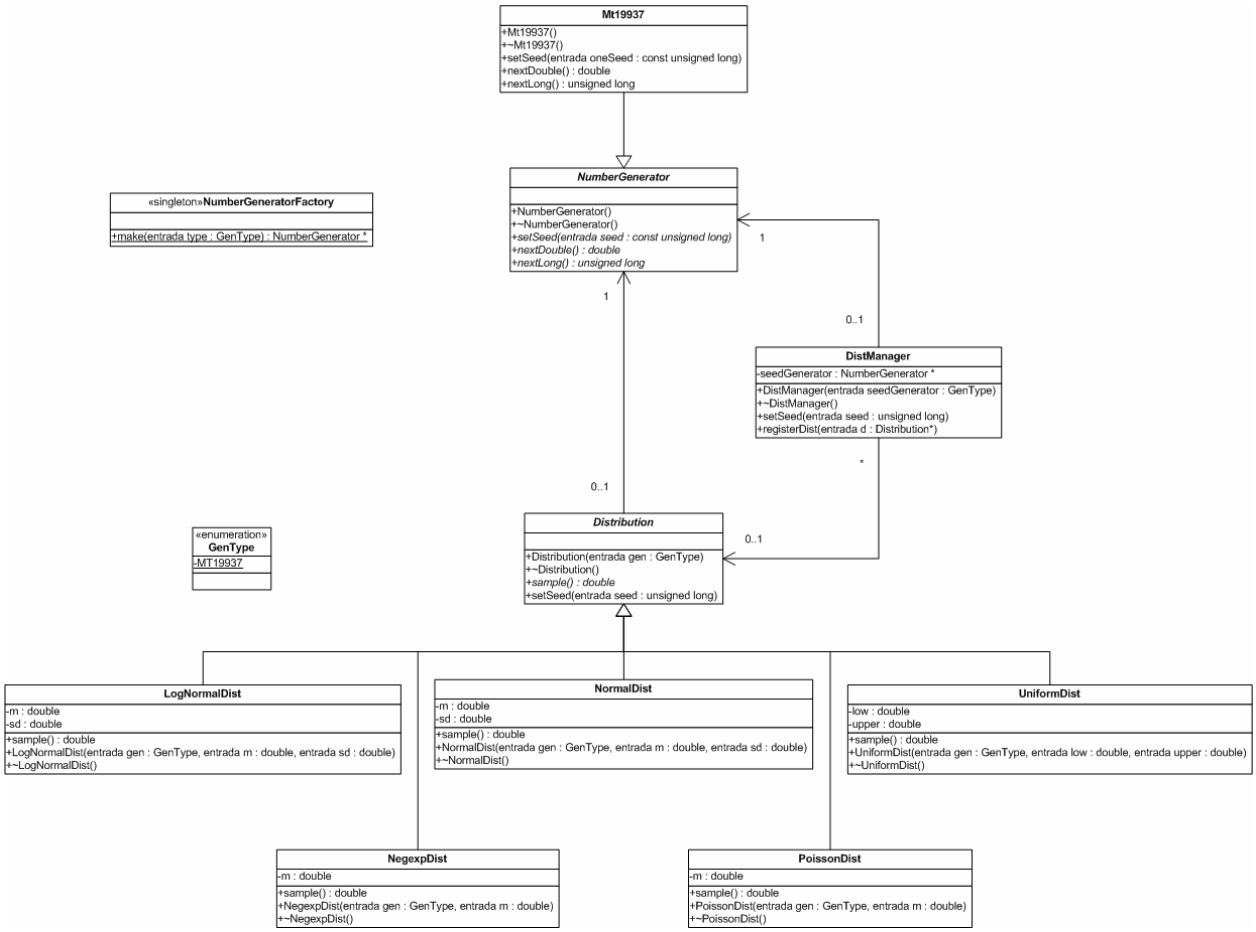

Figura 3 - El Módulo Dist

Este módulo contiene generadores de números pseudoaleatorios y distribuciones para muestreo de variables aleatorias. Todas las distribuciones son subclases de `Distribution`, clase abstracta que

define el comportamiento de todas las distribuciones de EOSimulator. Cada distribución tiene un generador asociado, definido por la interfaz `NumberGenerator`.

`NumberGeneratorFactory` es la clase encargada de crear instancias de `NumberGenerator` de acuerdo con el enumerado `GenType` que se pasa como parámetro. Este diseño permite al usuario agregar nuevos generadores y distribuciones.

3.3.2 Restricciones de Diseño

- Para agregar nuevos generadores `GenType` tiene que ser modificado de modo que se le de una nueva etiqueta al nuevo generador. También se tiene que modificar el método `NumberGeneratorFactory::make` para que cree instancias del nuevo generador.

3.3.3 Restricciones de Implementación

- Las distribuciones deben eliminar su propio generador cuando estas son eliminadas.

3.4Módulo Graphic

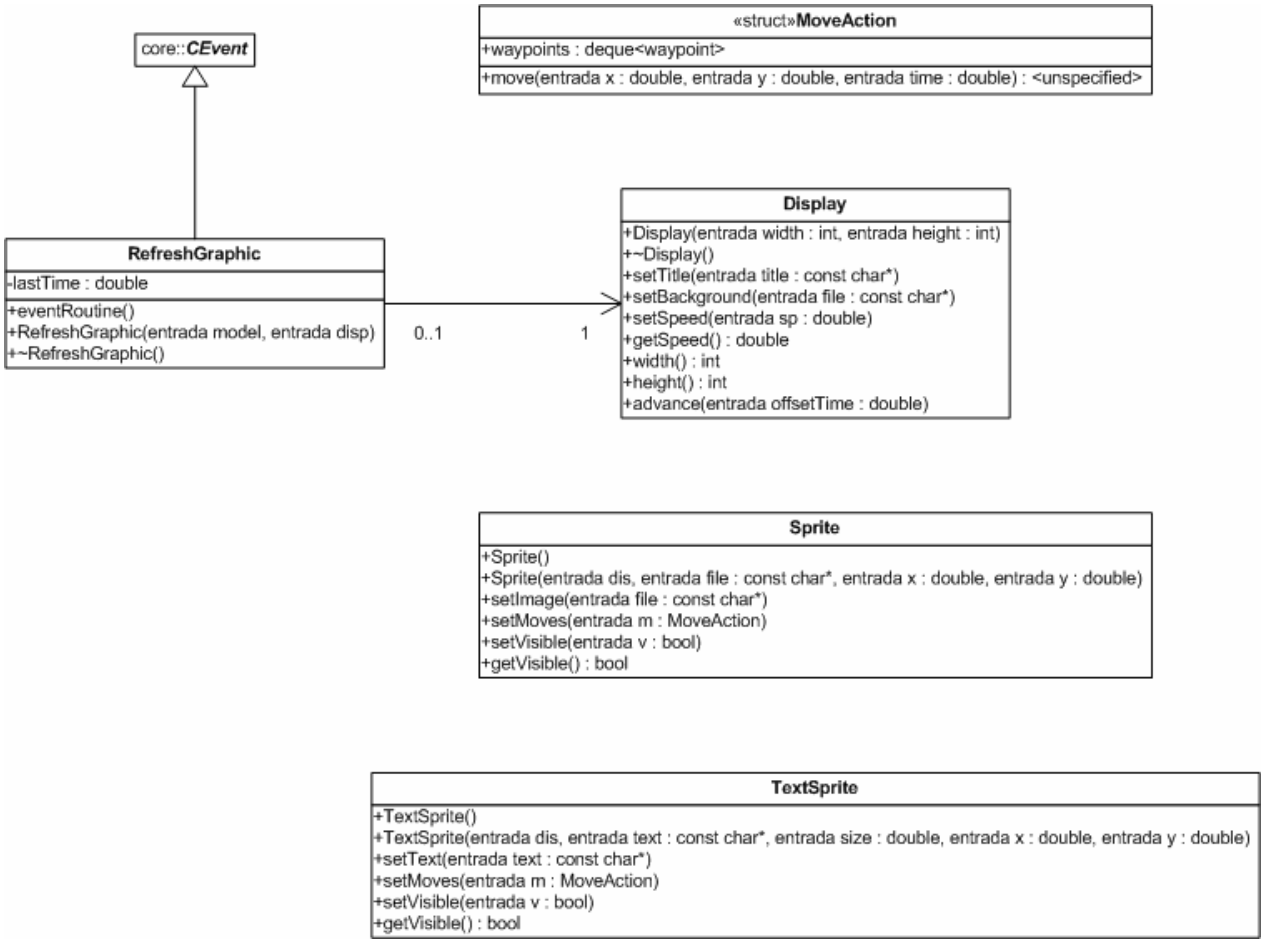

Figura 4 - El Módulo Graphic

El módulo `Graphic` provee la visualización gráfica de la simulación icónica 2D. Su interfaz fue diseñada para satisfacer de forma simple los requerimientos más comunes de la visualización de

simulaciones. Los elementos, tanto gráficos (*Sprite*) como texto (*TextSprite*), pueden ser creados y desplazados por la ventana, de modo que se muevan de acuerdo al tiempo de simulación. Los movimientos de dichos elementos se asignan utilizando la operación *Sprite::setMoves*. Ver [EOS06].

3.4.2 Restricciones de Diseño

- Para que la simulación se visualice de forma más realista, los objetos se mueven todos a la vez en la operación *Display::advance* que se invoca con el intervalo de tiempo transcurrido entre fases B.
- El avance de tiempo de los gráficos es realizado en la fase condicional por medio del evento condicional *RefreshGraphic* que debe ser registrado último.

3.4.3 Restricciones de Implementación

- Cuando ya no quedan referencias a una instancia de *Sprite* o *TextSprite* salvo por la del *Display*, la instancia se destruye y se elimina del *Display*.
- Este módulo fue implementado utilizando Boost C++ Libraries [BST06], SDL [SDL06], SDL_image [SDI06], Anti-Grain Geometry [AGG06] y FreeType [FT06]. Todas estas son bibliotecas de código libre y están disponibles para varias plataformas.

3.5Módulo Statics

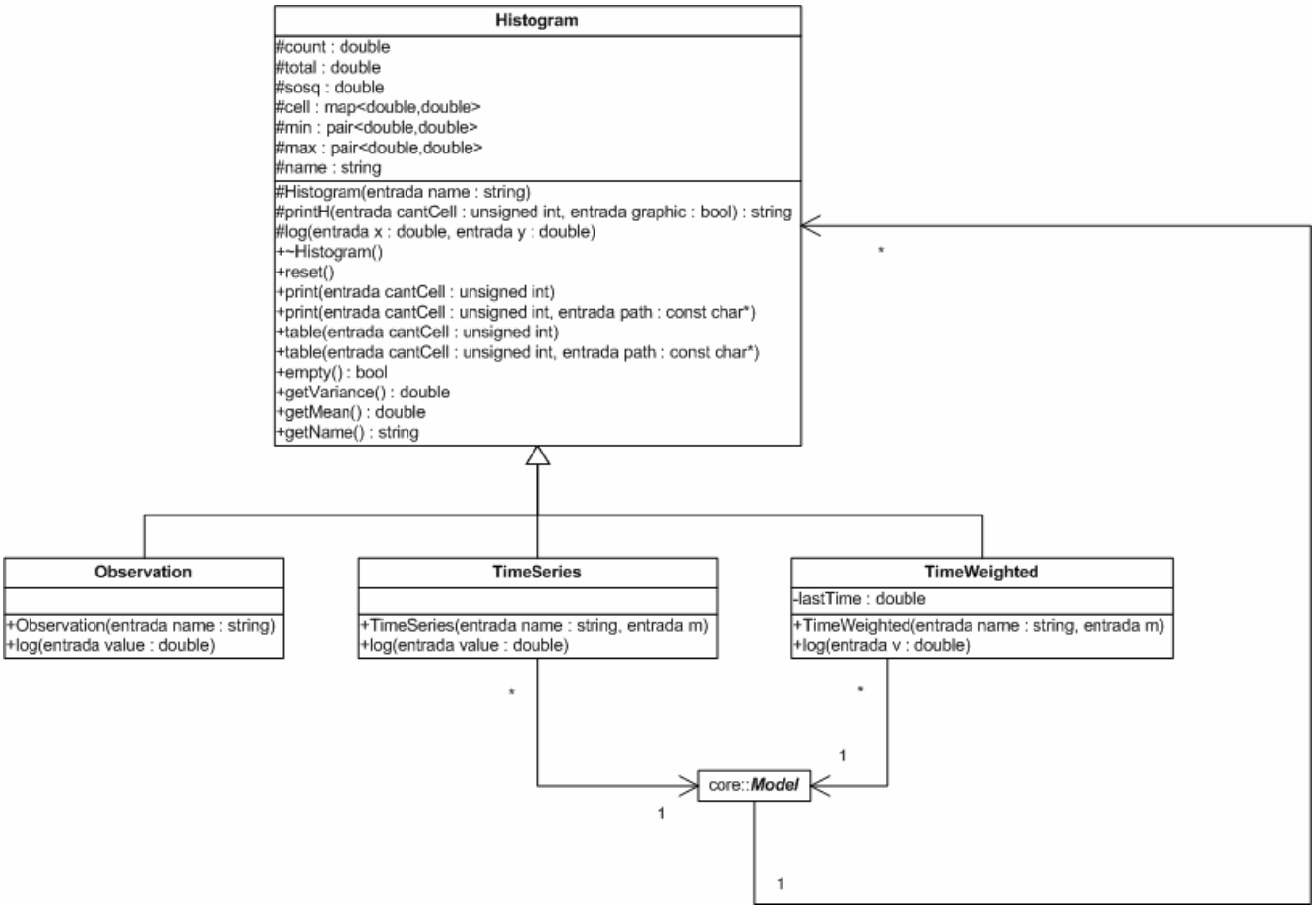

Figura 5 - El Módulo Statics

En este módulo se implementan los acumuladores de datos, en nuestro caso histogramas. Los histogramas que se brindan son de tres tipos: series de tiempo (`TimeSeries`), tiempos ponderados (`TimeWeighted`) y de observaciones (`Observation`). El comportamiento general de éstos está definido en `Histogram`. Los datos se guardan como pares (x, y) y se acumulan para x iguales. Para imprimir el histograma se ingresan la cantidad de categorías que se desean.

3.5.2 Restricciones de Diseño

- Como se ingresa la cantidad de categorías en las que se desea imprimir el histograma, el tamaño de estas es calculado automáticamente.

3.6Módulo Utils

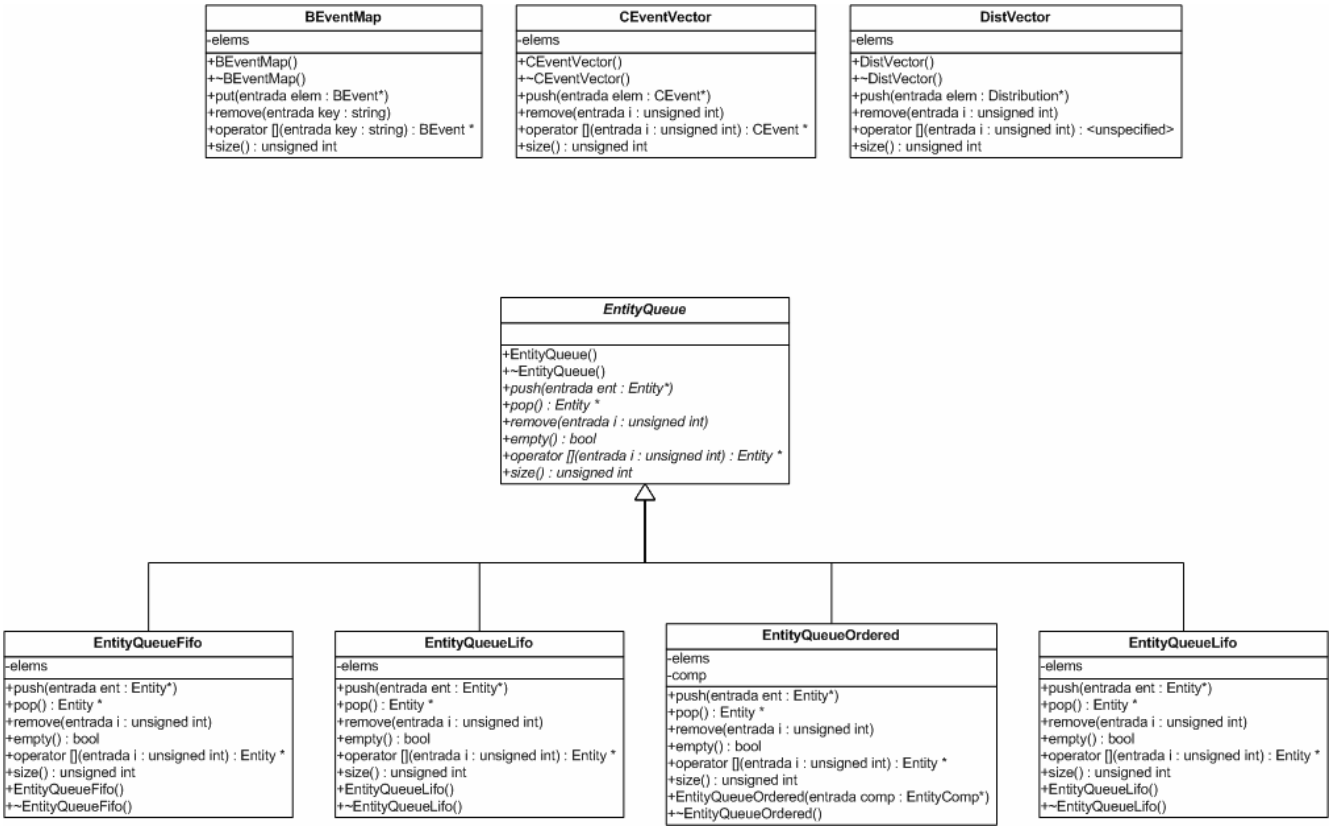

Figura 6 - El Módulo Utils

Este módulo contiene todas las colecciones que se utilizan en EOSimulator. Están basadas en los contenedores estándar de C++ (`vector`, `deque` y `map`). Las colecciones de la biblioteca están definidas básicamente como adaptaciones; de este modo se oculta la implementación al usuario. Como los contenedores que se utilizan soportan el acceso aleatorio, las colecciones de EOSimulator soportan el acceso mediante `operator[]` en forma eficiente.

3.6.2 Restricciones de Implementación

- Si bien `EntityQueueOrdered::pop` es de O(1) debido a la implementación de `std::deque`, el algoritmo de inserción es de O(n).
- Las operaciones que se brindan en `EntityQueue` se basan en la eficiencia de los contenedores estándar de C++. Si se cambian los contenedores utilizados, algunas funciones se aceleraran y otras no. La implementación dada es eficiente para iterar e insertar en la cabeza y la cola de `EntityQueue`, pero no en el medio de ésta.

4 Aplicación a Simulación a Eventos Discretos

La biblioteca EOSimulator se construyó en el primer semestre de 2005. La documentación incluye un manual de usuario, un manual de desarrollador y un conjunto de páginas web que documenta cada clase. Luego del desarrollo de algunas pruebas la biblioteca quedó lista para su aplicación en el curso

de ese año. Esta primera versión de la biblioteca poseía todos los módulos anteriormente mencionados salvo *Graphic*, o sea que no soportaba visualización gráfica.

En esta primera aplicación no se obligó a los estudiantes a utilizar la biblioteca, simplemente se la presentó como alternativa a Pascal_Sim. Varios grupos, un 24% del total de grupos que cursaron la materia, (en particular los estudiantes de postgrado) utilizaron la biblioteca sin mayores problemas. En este proceso se registraron algunos errores en la biblioteca que fueron corregidos, quedando en evidencia que la documentación generada era difícil de mantener pues estaba muy dispersa: una vez reparado el error se debía buscar en toda la documentación y corregirla, proceso que insumía mucho tiempo.

Al final del curso se registraron las opiniones de los estudiantes que derivaron en las siguientes observaciones:

- Histogramas: En la primera implementación se proveían dos tipos: *TimeSeries* y *TimeWeighted*. Si se quería imprimir histogramas vacíos estos abortaban el programa. Además el histograma *TimeSeries* podía ser usado como un histograma general. También se sugirió que dicho histograma promediara valores si estos se registraban en un mismo intervalo. Finalmente se planteó simplificar los constructores de los histogramas.
- Manejo de Errores: El manejo de errores se realiza mediante la función *assert* que aborta el programa cuando se viola un invariante de la aplicación. Esta función simplemente aborta sin dar información alguna sobre el error cometido.
- Manejo de distribuciones: El mecanismo de creación de generadores por medio del manejo de etiquetas no es muy adecuado.
- Motor de simulación: El acceso al calendario de eventos fijos (*BCalendar*) está restringido, esto no permite modelar algunas conductas especiales de las entidades. Además la clase *Experiment* solo maneja una corrida, el usuario debe crearse estructuras secundarias para obtener los resultados de múltiples corridas.
- Implementación de la biblioteca: No hay constructores por copia o por defecto de ninguna clase, lo que dificulta el uso de los contenedores estándar de C++ por parte del usuario.
- Distribución de la biblioteca: Se planteó repensar la forma de distribución de la biblioteca.

En base a estos aportes y para el curso del año 2006 se realizaron las siguientes mejoras:

- Documentación: Se hizo un esfuerzo por hacer una única documentación que se extrajera del código. De este modo cuando se cambia el código, se modifica la documentación que está en el mismo, y luego se genera ésta nuevamente. Esto hace que la corrección de errores y modificación de documentación insuma menos tiempo. Para ello se utilizó Doxygen[DOX06], una herramienta libre que genera documentación de una biblioteca por medio de los comentarios que se encuentran en el código fuente.
- Histogramas: Se rediseñó el sistema de histogramas, brindando tres tipos de histogramas. Se simplificaron los constructores de los mismos y se los especializó tratando de seguir las observaciones del año anterior.
- Distribución de la biblioteca: Se decidió soportar dos ambientes de programación: Visual Studio .Net de Microsoft y DevC++ que es un entorno libre. Estas herramientas utilizan compiladores diferentes, pero se logró que la biblioteca funcionara en ambos. La biblioteca se distribuye como un archivo comprimido que contiene el código fuente de la biblioteca y

varios ejemplos de sistemas. Para compilar tanto la biblioteca como los ejemplos se proveen proyectos que son soportados por los entornos antes mencionados. También se distribuye la documentación de usuario.

- Visualización Gráfica: Se incorporó la herramienta gráfica `simpp::display` desarrollada por el proyecto de grado "SimPP biblioteca de Simulación en C++". Ver [AMP05].

En la edición 2006 del curso se distribuyó el trabajo de modo que un tercio de los estudiantes utilizaran la biblioteca. De las mejoras antes mencionadas, las tres últimas se realizaron durante el curso, sin presentarse problemas. Los estudiantes manifestaron que la documentación era completa y útil, y que la forma de distribución de la biblioteca (con proyectos y ejemplos) facilitó la utilización del paquete. Sobre los histogramas queda pendiente el desarrollo de una interfaz para imprimirlos, puesto que el definir cantidad de "categorías" tiene limitaciones; como alternativa se sugirió ingresar el ancho de cada categoría. La visualización gráfica no presentó problemas, sin embargo se volvieron a plantear observaciones con respecto al manejo de errores, manejo de distribuciones y motor de simulación.

Otro aspecto a destacar es que los estudiantes consideraron que EOSimulator es de fácil aprendizaje, prácticamente similar al de Pascal_Sim. Esto se debe a que ambas bibliotecas tienen un uso parecido y que cuando se describen en clase entonces se destacan los aspectos similares para luego describir como sería la implementación particular en cada uno.

Como conclusión se puede afirmar que el desarrollo del paquete es satisfactorio, aunque se detectan varias funcionalidades que habría que mejorar. Las modificaciones realizadas se han probado y han funcionado correctamente. La biblioteca no es difícil de usar e incorpora mejoras a algunas de las funcionalidades de Pascal_Sim (listar cuáles). La experiencia ha sido positiva por tanto se continuará con la misma.

5 Trabajo Futuro

El trabajo futuro se basa fundamentalmente en el desarrollo de la biblioteca y su implantación en el curso.

En cuanto al desarrollo de la biblioteca se trata de seguir trabajando sobre las mejoras ya realizadas y las sugeridas en los años anteriores. En particular los histogramas y el módulo gráfico son aspectos a mejorar. Sobre los histogramas varios estudiantes manifestaron que no es muy intuitiva la forma en que estos se muestran. Es probable que cambiando la interfaz de modo que en lugar de pasar como parámetro la cantidad de columnas a imprimir se pase el intervalo de valores que comprende cada columna. El módulo gráfico en cambio resultó intuitivo y de fácil uso, pero se encuentra implementado con versiones no actualizadas de las bibliotecas Anti-Grain Geometry y FreeType. Sería deseable actualizar el módulo para que funcione con las nuevas versiones de dichas bibliotecas. Otra de las mejoras podría ser eliminar las dependencias bidireccionales entre los módulos.

Sobre las mejoras sugeridas por los estudiantes se encuentra mejorar el reporte de errores mediante manejo de excepciones, cambiar la forma de identificación de eventos mediante `strings`, cambiar la forma de creación de los generadores de números aleatorios. En cuanto al manejo de errores al momento se está haciendo mediante el uso de la función `assert` lo que aborta el programa si se produce un error. Esto hace dificultoso encontrar errores ya que no se sabe exactamente como se provocó el error. Mediante el manejo de excepciones es posible evitar este problema. La identificación de eventos mediante `strings` es un mecanismo en parte heredado de Pascal_Sim donde los eventos se identifican mediante un entero. Una idea es pasar el objeto mismo a la hora de insertar una entidad

en el calendario, pero esto trae problemas sobre la visibilidad y acceso a los eventos. Finalmente está modificar la forma de creación de los generadores. Actualmente esto se hace pasando un enumerado que identifica el tipo de generador, este una vez creado es asignado a una distribución. La biblioteca está diseñada pensando en la adición de varios generadores, pero es necesario recompilar la biblioteca cada vez que se agrega un nuevo generador.

Por otro lado se tiene la implantación de la biblioteca en el curso. La idea es progresivamente aumentar el uso de EOSimulator, sin embargo es importante mantener el uso de Pascal_Sim para estudiantes de otras carreras con poca formación en lenguajes de programación.

6 Referencias

[AMP05] ALAGGIA, Sebastián; MARTÍNEZ, Bruno; PARDO, Fernando. SimPP, Librería de Simulación en C++. URQUHART, Maria; MAUTTONE, Antonio (tutores) Trabajo de Grado. Facultad de Ingeniería. Instituto de Computación. Montevideo, 2005.

[AGG06] Anti Grain Geometry Home Page

http://www.antigrain.com/ Diciembre 2006

[BST06] Boost C++ Libraries Home Page

http://www.boost.org Diciembre 2006

[DMJ05] DesmoJ Home Page

http://www.desmoj.de/ Abril 2005

[DO89] DAVIES, Ruth M.; O'KEEFE, Robert M. Simulation modelling with Pascal. Hertfordshire: Prentice Hall, 1989. p.302. ISBN: 0-13-811571-0

[DOX06] Doxygen Home Page

http://www.stack.nl/~dimitri/doxygen/ Diciembre 2006

[EOS06] EOSimulator Home Page

http://www.fing.edu.uy/inco/cursos/simulacion/eosim_html/index.html Diciembre 2006

[FT06] Free Type Home Page

http://www.freetype.org/ Diciembre 2006

[SDL06] SDL Home Page

http://www.libsdl.org Diciembre 2006

[SDI06] SDL_image Home Page

http://www.libsdl.org/projects/SDL_image/ Diciembre 2006

[Wai03] WAINER, Gabriel A. Metodologías de modelización de y simulación de eventos discretos. Buenos Aires: Nueva Librería, 2003. p.380. ISBN: 987-1104-01-4