LA MODELOTECA. Biblioteca de Meta – Modelos para Simulación Basados en GALATEA.

Autor: Almathely Vivas R.
Profesor Tutor: Jacinto Dávila

Trabajo presentado como requisito parcial para optar al título de
MAGISTER SCIENTIAE

UNIVERSIDAD DE LOS ANDES
MERIDA, VENEZUELA
Mérida, 10 de Marzo 2008

Tabla de Contenido

Índice de Figuras

Introducción

El título “modelado y simulación” agrupa un conjunto de actividades que tienen como finalidad la construcción de modelos de sistemas reales o teóricos (que pueden no existir, no ser reales) para simularlos usando herramientas computacionales [1], con el fin de experimentar o comprender el comportamiento del sistema.

GALATEA [2] es una plataforma de simulación que incluye una biblioteca de clases, desarrollada en el lenguaje de programación Java, y que proporciona a modelistas y simulistas una herramienta computacional para la simulación de sistemas por eventos discretos, continuos y Multi – Agentes. GALATEA está basada en el formalismo general del modelado y simulación de eventos discretos (DEVS) y en una teoría de Simulación de Sistemas Multi – Agentes desarrollada al efecto [3].

GALATEA es heredera de la semántica y buena parte de la sintaxis de la plataforma de simulación GLIDER [4], que contiene un simulador de eventos discretos desarrollado en el lenguaje de programación Pascal. La diferencia fundamental entre estas dos plataformas de simulación es la incorporación de los agentes a la semántica del lenguaje de simulación.

Tanto GALATEA como GLIDER han contado desde el comienzo con una colección de ejemplos de modelos de simulación, con propósitos instruccionales. En GALATEA se le ha llamado MODELOTECA [2].

Este proyecto derivó en la creación de la META - MODELOTECA a partir del análisis de los objetos fundamentales en el dominio del modelado y la simulación, por lo que ésta es, ya no una colección de modelos acabados solamente, sino una biblioteca o colección de meta-modelos que permitirán a los modelistas y simulistas sin mucha experiencia computacional, la generación de modelos de sistemas y situaciones comunes. En este contexto, un meta-modelo puede ser visto como una clase de plantilla que puede ser instanciada en diversos modelos y situaciones particulares [5].

El objetivo general planteado en este proyecto es diseñar sistemáticamente la META - MODELOTECA de GALATEA a partir del análisis de los objetos fundamentales en el dominio del modelado y la simulación.

En base a este objetivo general se plantearon tres objetivos específicos:

- Definir un conjunto de posibles sistemas y seleccionar cuatro de ellos, los cuales formarán parte de la META - MODELOTECA.
- Determinar los meta – modelos a partir de los modelos seleccionados

haciendo uso de los conceptos proporcionados por la ingeniería de dominio.

- Diseñar e implementar un prototipo de la interfaz gráfica de usuario, que permita la configuración de los meta – modelos.

La metodología usada para el desarrollo de este proyecto es una combinación de los conceptos proporcionados por la ingeniería de dominio, ya que proporcionaba los mecanismos de análisis para los diferentes dominios de conocimiento a los cuales pertenecen los diferentes modelos analizados; así como también, el uso de un modelo de proceso de software evolutivo. ya que el software, como todos los sistemas complejos evolucionan en el tiempo. Los modelos evolutivos son iterativos y se caracterizan por la forma en que permiten desarrollar versiones cada vez más completas del software.

En cuanto a la Ingeniería de Dominio podemos decir que es un proceso que, a través de una serie de fases definidas y los productos de las mismas, permiten que el conocimiento estructural de un dominio sea capturado y vaya evolucionando [6]. En este contexto, el dominio puede definirse como un área de conocimiento o familia de sistemas en el cual se observan los conceptos relacionados, se determinan los aspectos comunes y las variaciones con el fin de resolver problemas mediante la realización de artefactos de software [6].

El modelo de proceso de software evolutivo permite desarrollar una versión inicial de la META - MODELOTECA y de cada meta – modelo e ir refinando la aplicación en *N* versiones hasta llegar al producto final. Cada una de estas versiones del software puede ser vista como un prototipo, lo que permite desarrollar la aplicación de manera incremental [7]. Las fases o etapas del proceso de software evolutivo se adaptan a las etapas contempladas en la ingeniería de dominio.

Por lo tanto las fases del modelo evolutivo quedan definidas de la siguiente manera:

- Análisis del dominio: El Análisis de dominio se refiere a los procesos que corresponden a la selección y definición del dominio, con la ayuda de analistas y expertos del dominio para lograr el entendimiento de los aspectos comunes y las variaciones respecto a otros dominios, esto con el propósito de representar el conocimiento obtenido del dominio y de esta manera lograr integrar o construir una arquitectura del dominio. En este caso, la arquitectura del dominio no es más que la representación de los aspectos comunes y las variabilidades de los sistemas y los requerimientos que existen dentro de un dominio dado [6].

- Diseño de dominio o arquitectura de dominio: El Diseño del dominio es el proceso que corresponde al diseño de una arquitectura genérica utilizando el Modelo del Dominio, la evaluación de dicha arquitectura, y la documentación con el fin de que esta soporte la reutilización de software [6].

- Arquitectura de software: Una Arquitectura de Software es una descripción de los componentes que se utilizaran para construir un sistema. Esta

descripción abarcará lo concerniente a las interfaces, servicios, interconexiones, con el fin de guiar o referenciar el proceso de implementación para que éste pueda cumplir con el comportamiento deseado[6].

- Implementación del dominio: La implementación del dominio consiste en aplicar tecnologías de manera apropiada con el fin de implementar componentes, generadores para ensamblaje automático de componentes, infraestructuras de reutilización y aplicación de procesos de producción. Otra parte importante a considerar es la creación, el manejo y el mantenimiento de un repositorio de elementos reutilizables [6].

En la siguiente figura puede observarse una imagen con la estructura del modelo de proceso evolutivo y su adaptación con los conceptos de la ingeniería de dominio.

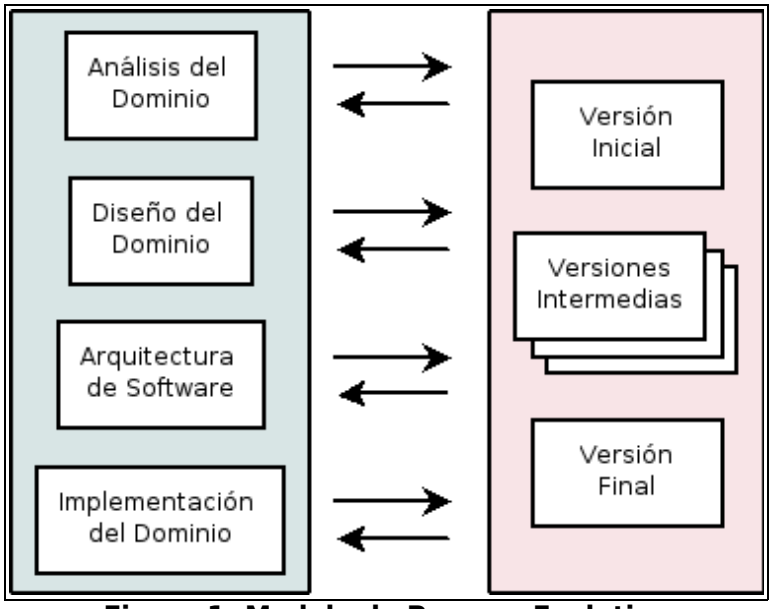


Figura 1: Modelo de Proceso Evolutivo

Esta metodología fue seleccionada debido a la naturaleza evolutiva de este proyecto (es decir, se espera que la META - MODELOTECA siga creciendo indefinidamente). Cada modelo que se ha analizado para formar parte de la META - MODELOTECA es tratado como un ciclo del proyecto. Esto ha permitido ir desarrollando de manera incremental y sistemática la META - MODELOTECA incluyendo paulatinamente cada meta – modelo completamente desarrollado. En los capítulos siguientes se explica como ha ocurrido este proceso para cada modelo, describiendo cada una de las etapas de la ingeniería de dominio descritas anteriormente.

La META - MODELOTECA se presenta en este documento y se realiza la descripción en seis capítulos. El primer capítulo explica que es la MODELOTECA y

como fue concebida. El segundo capítulo describe el proceso que se siguió para generalizar el modelo del sistema ecológico perturbado y llegar a convertirlo en el meta – modelo de sistemas dinámicos. El tercer capítulo explica el procedimiento llevado a cabo para generalizar el modelo de cohortes escolares. El cuarto capítulo precisa los pasos realizados para generalizar el modelo de la ruta de autobús en un meta – modelo de transporte. El quinto capitulo describe el proceso para generalizar el modelo del puerto de buques petroleros en su respectivo meta – modelo. El sexto capítulo presenta un conjunto de ejemplos que pueden servir al lector para aprender a utilizar la META - MODELOTECA. Al final del documento, luego de presentar las conclusiones y recomendaciones, proporcionamos algunos apéndices de apoyo, proporcionando los códigos implementados en GLIDER y en GALATEA/Java de los sistemas escogidos para formar parte de la META - MODELOTECA.

Capítulo I

La META - MODELOTECA

En la actualidad el modelado y la simulación de sistemas es usado en numerosas y diversas áreas de aplicación como una herramienta fundamental tanto en el ámbito de la ciencia de vanguardia e investigación, como a nivel empresarial, ya que permite la exploración y comprensión de sistemas complejos. Así mismo, es de gran valor en los procesos de enseñanza y aprendizaje porque puede usarse en todos los niveles educativos para ayudar a explicar, demostrar o practicar conceptos.

Debido a la importancia del modelado y la simulación de sistemas, tanto en la docencia como en la investigación, han surgido gran cantidad de plataformas, herramientas y lenguajes de simulación.

GALATEA [2] es una plataforma de simulación, desarrollada en la Universidad de los Andes, que incluye una biblioteca de clases, desarrollada en el lenguaje de programación Java, y que proporciona a modelistas y simulistas una herramienta computacional para la simulación de sistemas por eventos discretos, continuos y Multi – Agentes. GALATEA está basada en el formalismo general del modelado y simulación de eventos discretos y en una teoría de Simulación de Sistemas Multi – Agentes desarrollada al efecto [3].

GALATEA es heredera de la semántica y buena parte de la sintaxis de la plataforma de simulación GLIDER [4], que contiene un simulador de eventos discretos (DEVS) desarrollado en el lenguaje de programación Pascal. La diferencia fundamental entre estas dos plataformas de simulación es la incorporación de los agentes a la semántica del lenguaje de simulación.

Tanto GALATEA como GLIDER han contado desde el comienzo con una colección de ejemplos demostrativos de modelos de simulación. En GALATEA se le ha llamado MODELOTECA [2]. Dicha colección de ejemplos, tanto en GLIDER como en GALATEA necesitan de ciertos conocimientos en computación previos para poder entenderlos y usarlos.

Adicionalmente, son muy específicos, es decir, modelos acabados, lo que limita el uso que puede dárseles a solamente un uso instruccional, tanto de la herramienta y lenguaje de simulación, como del sistema específico que modela.

Este proyecto se ha planteado la tarea de vencer dichas limitaciones mediante

el diseño mejorado de la MODELOTECA que no sea solo una colección de modelos acabados solamente como se planteó en [2], sino una biblioteca o colección de meta – modelos dentro de un dominio específico, lo que permite que sean reusables para diferentes sistemas dentro de ese dominio. En este contexto, un meta - modelo puede ser visto como una clase de plantilla que puede ser instanciada en diversos modelos y situaciones particulares [5]. El concepto se comenzó a plantear en la tesis **gSpaces. Meta – Modelo para Simular Desalojos en Espacios Urbanos y Arquitectónicos Basados en GALATEA** [5] con mucho éxito.

La META - MODELOTECA que nos proponemos ahora, reduce la exigencia de que los usuarios posean conocimientos computacionales previos, ya que proporciona una herramienta con una interfaz gráfica en la que los modelistas y simulistas pueden configurar los meta – modelos, sin tener que aprender un nuevo lenguaje de programación, y de esta manera generar modelos de sistemas y situaciones comunes con la posibilidad de simularlos y obtener resultados inmediatamente.

Debido a que el dominio del modelado y la simulación se perfila como uno de los dominios de conocimiento más complejos que pueden existir, precisamente porque esas herramientas se ofrecen para abordar la complejidad de cualquier sistema, se limitó cuidadosamente el alcance de este ejercicio de diseño de la META - MODELOTECA. Para ello se realizó una revisión y análisis de algunos modelos entre los cuales se encuentran: Modelos caóticos, Modelos de Movilidad de Material (continuos y discretos), Modelos Ambientales y Modelos de Transporte (continuos y discretos), entre otros, y se definió una lista de posibles modelos.

Para realizar de una manera fácil y sencilla el proceso de selección de los modelos que conforman la META - MODELOTECA se definió un espacio de meta – modelos, el cual puede describirse como un espacio tridimensional en donde los ejes “x”, “y” y “z” representan “el dominio del conocimiento involucrado en el modelo”, “el tipo de aprovechamiento que se hace del modelo” y “la complejidad computacional del modelo” respectivamente. A continuación, en la figura 2, se presenta una representación de este espacio de meta – modelos.

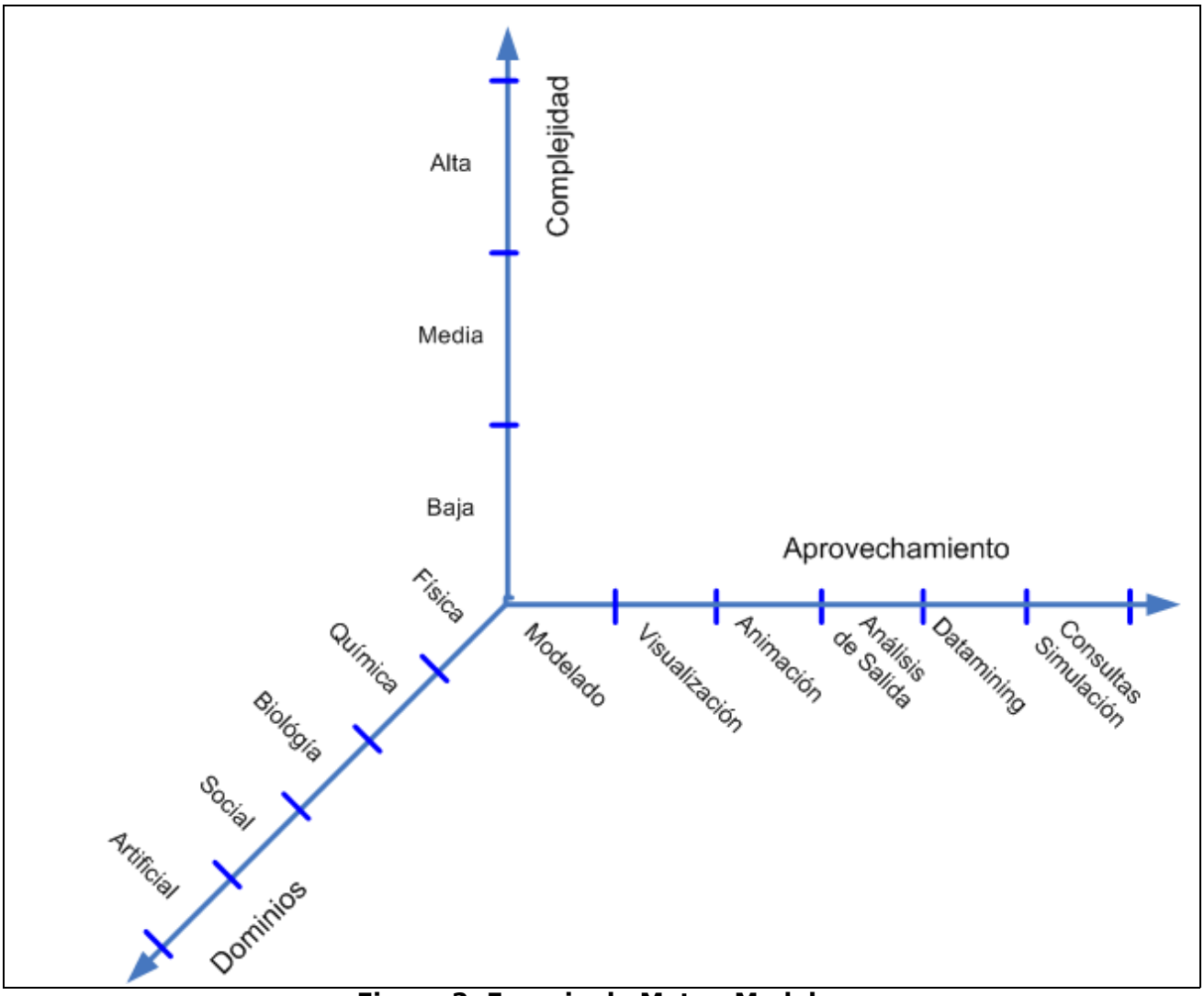


Figura 2: Espacio de Meta - Modelos

El dominio o área de conocimiento se refiere al tema o área específica de conocimiento del sistema que ha sido modelado. En este caso se ha definido una jerarquía de sistemas, la cual se describe a continuación:

1 Sistemas Físicos
  1.1 Sistemas Cinemáticos
  1.2 Sistemas Termodinámicos
  1.3 Sistemas Caóticos

2 Sistemas Químicos
  2.1 Sistemas Bioquímicos
  2.2 Sistemas Químicos – Orgánicos
  2.3 Sistemas Químicos – Inorgánicos.

3 Sistemas Biológicos
  3.1 Sistemas Ambientales
  3.2 Sistemas Ecológicos

4 Sistemas Sociales
  4.1 Sistemas Multi - Agentes
  4.2 Sistemas Culturales

5 Sistemas Artificiales
  5.1 Sistemas Industriales
  5.2 Sistemas Mecánicos
  5.3 Sistemas Lingüísticos
  5.4 Sistemas Financieros o de Micro – Economía
  5.5 Sistemas de Macro - Economía

Aunque ya existen diversos autores que han planteado clasificaciones y taxonomías de sistemas, como por ejemplo Boulding [9], Jordán [10], Checkland [11] y la taxonomía implícita propuesta por Casti [12][13], par la clasificación usada en este trabajo no se tomó ninguna de las propuestas por estos autores, sólo se tomaron las ideas básicas de cada autor en cuanto a sus respectivas taxonomías y se definió una clasificación aproximada que nos permitió ubicar los modelos en un área de conocimiento de una manera sencilla.

El tipo de aprovechamiento tiene que ver con lo que puede hacerse con el modelo (y el computador). Por ejemplo, la simulación, visualización y predicción. En base a este concepto definimos los niveles de aprovechamiento, de menos a más complejo, como sigue:

- Modelado
- Consultas o simulación
- Visualización
- Animación
- Análisis de salida
- Datamining o predicción

La complejidad computacional del modelo la cual se usa para medir el grado de elaboración de cada producto de software, inicialmente es una medida elemental basada en la cantidad de líneas de código que posee el modelo y/o en el número de nodos que forman la red de nodos GALATEA. De acuerdo a este concepto se han definido tres niveles básicos de complejidad:

- Bajo (menos de 500 lineas de código, menos de 5 nodos).
- Medio (entre 500 líneas y 1000 líneas de código, entre 5 y 10 nodos).
- Alto (más de 1000 líneas de código, más de 10 nodos).

El objetivo de definir estas tres áreas de clasificación para los modelos, fue poder elegir de la lista de modelos elegibles los mejores candidatos para meta – modelos, mediante una visualización gráfica de la clasificación de los modelos. Dicha clasificación nos permitió evaluar y seleccionar los meta – modelos que integran la META - MODELOTECA de acuerdo al mayor impacto que puedan tener en el proyecto.

La estructura de la META - MODELOTECA está conformada por un conjunto de

cuatro meta – modelos, los cuales fueron definidos a partir de los sistemas escogidos durante el proceso de selección, en el cual se hizo uso de los conceptos proporcionados por la ingeniería de dominio.

Desde el punto de vista de aprovechamiento del modelo, este proyecto ha dejado de lado media complejidad computacional, así mismo también ha seleccionado una muestra que pareciera ser muy pequeña, sólo cuatro meta – modelos, comparada con el amplio espacio de sistemas estudiados. Todo esto como consecuencia de una fase en la cual se acotó con mucho cuidado el alcance y los objetivos de este trabajo, porque el ejercicio de diseño propuesto en este proyecto conlleva la existencia de riesgos que pueden afectar el proceso de desarrollo. Alguno de los riesgos más importantes que podían afectar el proyecto es una mala estimación del tiempo de desarrollo, así como también que surgiera la necesidad de que el desarrollador tuviera o adquiriera un gran entendimiento sobre algunas de las áreas de conocimiento específicos durante el planteamiento de los meta – modelos. Ambos tipos de riesgos, podemos reportar, no se materializaron.

Entre las áreas de conocimiento correspondientes a los sistemas seleccionados se encuentran los sistemas biológicos, sistemas sociales y sistemas artificiales. Los sistemas biológicos y sociales son intrínsecamente complejos desde el punto de vista del dominio de conocimiento. El sistema artificial específico seleccionado, aunque de baja complejidad en el dominio de conocimiento, ofrece posibilidades para probar un tipo de aprovechamiento más complejo como la animación.

Uno de los aspectos más importantes de este proyecto es la interfaz gráfica de usuario, la cual permite el diseño y la edición de modelos, así como la simulación de éstos, dentro de cualquiera de los dominios proporcionados por las plantillas que posee la herramienta. La interfaz gráfica de la META - MODELOTECA fue pensada e implementada con el propósito de que fuera fácil de usar por el usuario, no programador. Así mismo que contara con una presentación agradable y funcional.

Los requerimientos básicos que fueron especificados para el funcionamiento de la interfaz gráfica se enumeran a continuación:

R1 Cambiar idioma de la interfaz (soporte para internacionalización).
R2 Generar modelos específicos a partir de las plantillas proporcionadas por la aplicación.
  R2.1 Configurar parámetros específicos para cada plantilla.
  R2.2 Cargar lotes de parámetros desde archivos.
R3 Simular los modelos.
R4 Guardar/Abrir modelos.
R5 Mostrar los resultados de la simulación.
  R5.1 Generar tablas de datos.
  R5.2 Generar gráficos de variables.
  R5.3 Generar animación de la simulación.

La herramienta proporciona la posibilidad de cambiar el idioma de la interfaz, el cual puede ser seleccionado de uno de los menús. Esta característica permite

adaptar el software a diferentes lenguajes y regiones sin la necesidad de cambios en el código o recompilación, simplemente agregando un archivo de texto con las traducciones. Se diseño e implementó esta funcionalidad de manera que pudiera ser usada como un componente en la interfaz gráfica. A continuación se muestra el diagrama de clases correspondiente a esta funcionalidad.

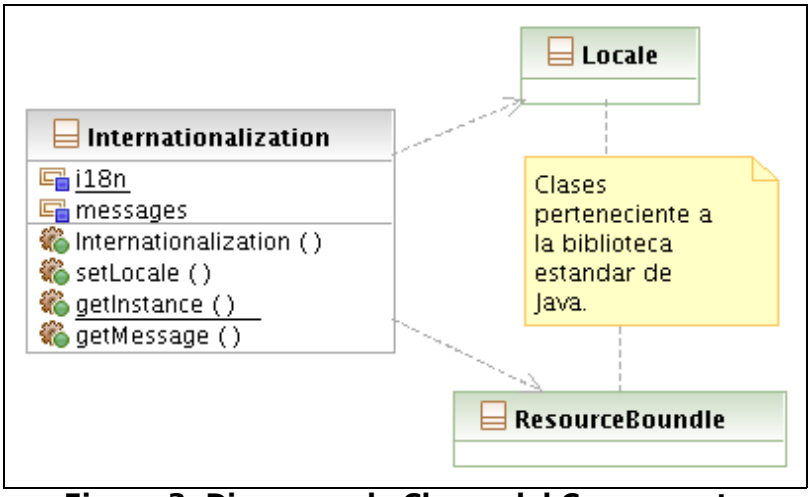


Figura 3: Diagrama de Clases del Componente Internacionalización.

El procedimiento para generar los modelos está basado en el uso de una estructura de árbol en la cual se pueden ir agregando los modelos, los elementos de cada modelo y sus configuraciones correspondientes. Cada elemento es mostrado como una rama en el árbol, lo que permite acceder a ellos fácilmente, así como también editarlos de forma sencilla.

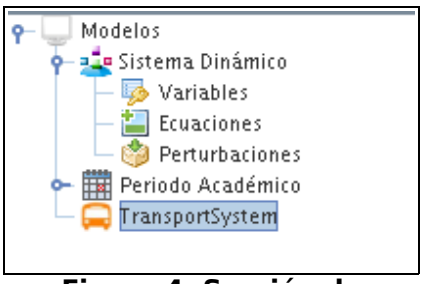


Figura 4: Sección de Configuración de Modelos de la Interfaz Gráfica

Luego de realizada la configuración del modelo particular, el menú de simulación permite el acceso a las opciones en las cuales puedes iniciar la simulación o pararla si ya ha sido iniciada. Luego de completado el proceso de simulación, la plantilla te entrega los resultados de la simulación con los cuales la interfaz gráfica genera las tablas, gráficos y/o animaciones correspondientes.

En cuanto al requerimiento de generar gráficos de variables, se realizó una

búsqueda de bibliotecas o aplicaciones de software libre que permitieran implementar dicha funcionalidad. La biblioteca escogida fue jFreeChart [14], la cual es una biblioteca de software libre que permite generar diversos tipos de gráficas, entre los cuales se pueden nombrar: gráficos de torta en dos y tres dimensiones, gráficos de barras, gráficos de líneas, gráficos de áreas, gráficos de dispersión, gráficos de burbujas, gráficos de series de tiempo, gráficos combinados, gráficos financieros, entre muchos otros.

La biblioteca jFreeChart es una de las bibliotecas con licencia de software libre más extendidas que hay disponible, ya que además de que permite crear gran cantidad de tipos de gráficas, posee entre otras características, la posibilidad de ser integrada no sólo en aplicaciones de escritorio, sino también en Applets, Servlets y JSP. Así mismo, proporciona diversos formatos de salida tales como, componentes swing, archivos de imágenes (PNG y JPG) y formatos de archivos de gráficos vectoriales (incluyendo PDF, EPS y SVG).

Adicionalmente, posee muchas características interactiva entre la cuales se encuentran las opciones de ampliación o acercamiento de los gráficos y . Todas estas características aunadas a un diseño extensible y una buena documentación de la interfaz de programación de aplicaciones hacen de esta biblioteca una de las mejores opciones para integrar la creación de gráficos en aplicaciones Java.

La funcionalidad de generación de gráficos de variables, por ser requerida por varios de los meta – modelos, también se diseño e implementó como un subsistema de la interfaz gráfica, el cual recibe los resultados de la simulación y los usa para crear los gráficos dinámicamente. A continuación se muestra el diagrama de clases para este subsistema.

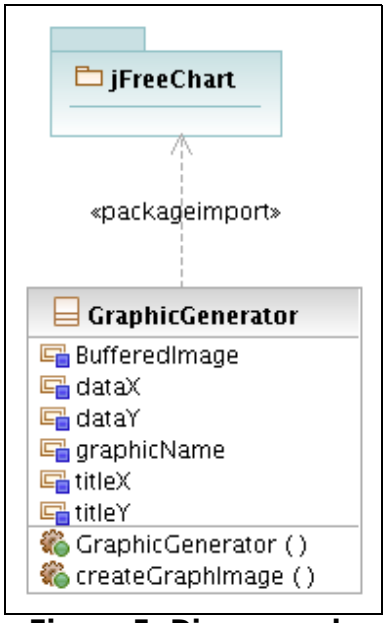


Figura 5: Diagrama de Clases del Componente Generador de Gráficos

La ultima de las funcionalidades básicas descritas en esta sección es la referente a los requerimientos de abrir y guardar los modelos. Para ello se decidió usar el formato de archivo XML. Esta funcionalidad permite guardar los modelos configurados en archivos XML. Así mismo, y siguiendo el mismo esquema de formato XML, es posible que el usuario configure un modelo directamente en un archivo XML (siguiendo las directivas establecidas para dichos archivos), el cual puede ser abierto desde la aplicación, la cual realiza la interpretación del archivo XML y a partir de él, generar el modelo y sus respectivos elementos.

En función de esto se realizó una búsqueda de una biblioteca de software libre que implementara esta funcionalidad y aunque la oferta es muy diversa y variada se decidió usar las clases incluidas en la biblioteca estándar de Java que proporciona dicha funcionalidad [15]. A continuación se muestra el diagrama de clases para implementar dicha funcionalidad:

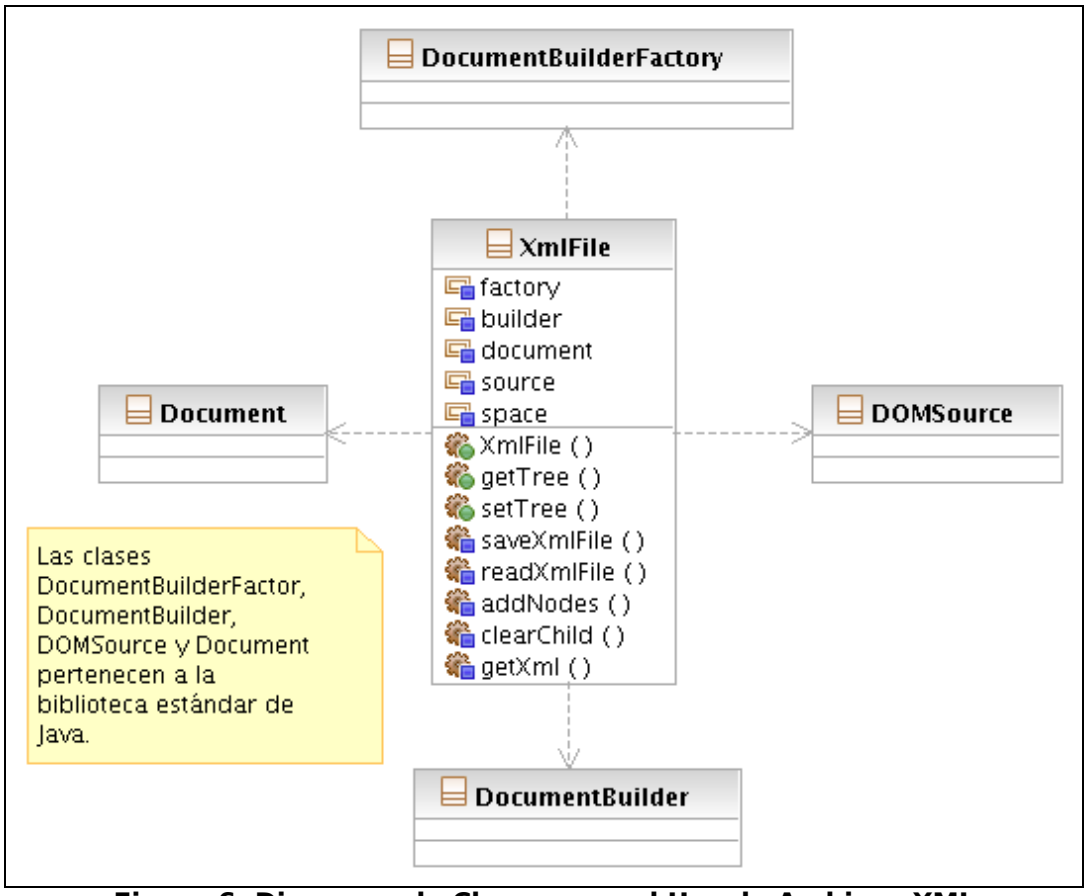


Figura 6: Diagrama de Clases para el Uso de Archivos XML

El diseño de la aplicación se planteó en tres niveles o capas las cuales encapsulan el funcionamiento de la aplicación, permitiendo modificar alguna de ellas sin afectar la arquitectura ni la interacción de unas capas con otras, lo que flexibiliza y facilita el mantenimiento de la aplicación. Los niveles definidos son los siguientes:

- Nivel de simulación: en este nivel se tienen las clases básicas proporcionadas por la biblioteca de GALATEA, las cuales proporcionan el motor de simulación sobre el cual se describen y especifican los meta – modelos.

- Nivel de meta – modelos o plantillas: en este nivel se definen las clases necesarias para plantear cada meta – modelo, así como también instanciarlo y simularlo.

- Nivel de interfaces: en este nivel se encuentran todas las clases relacionadas con la interfaz de usuario, es decir, la capa de más alto nivel de la aplicación, ya que es el único nivel con el que interactúa el usuario directamente.

En el diagrama siguiente se muestra una representación de componentes describiendo los niveles de la aplicación.

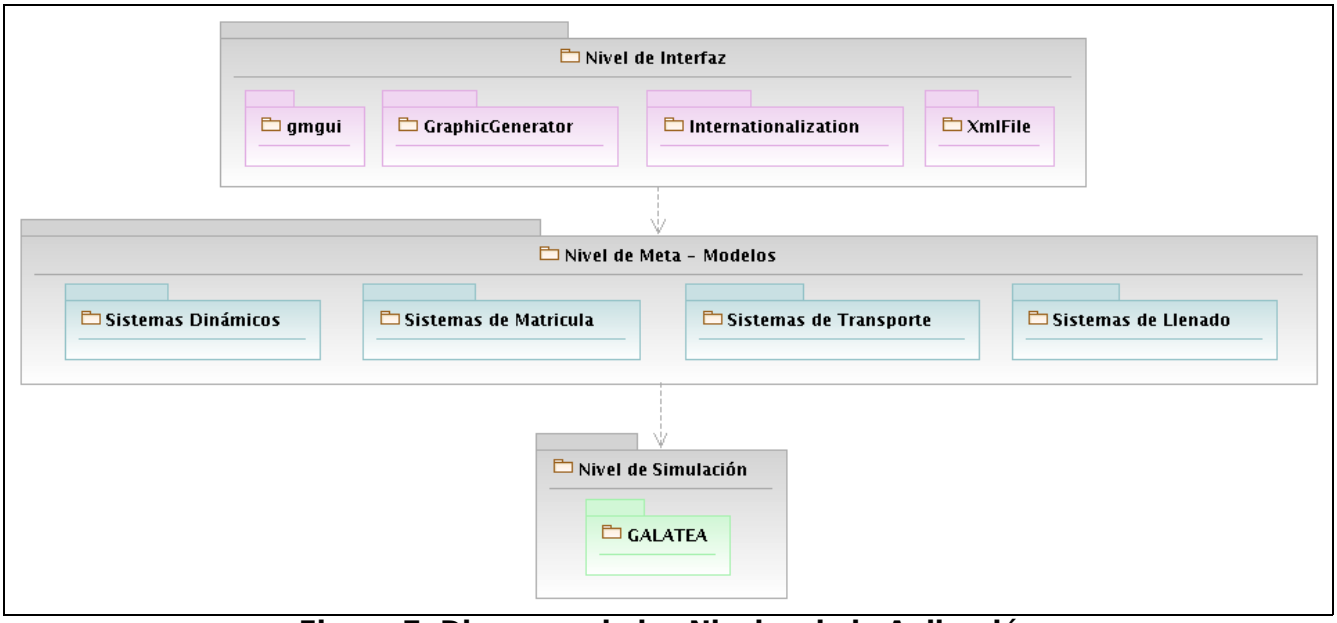


Figura 7: Diagrama de los Niveles de la Aplicación

En resumen, se diseño una aplicación muy versátil y modular, la cual cuenta con una interfaz de usuario muy amigable desde el punto de vista de usuario final.

Capítulo II

Meta – Modelo: Sistemas Dinámicos

El primer modelo seleccionado para formar parte de la META - MODELOTECA fue el Sistema Ecológico Perturbado [16], clasificado en el área de conocimiento de Sistemas Biológicos, el tipo de aprovechamiento es Visualización y complejidad Media. Este modelo describe la manera en que interactúan cinco especies en un ecosistema, dichas especies representan un nivel diferente en la cadena alimenticia. Adicionalmente existen dos perturbaciones que inciden en el ecosistema en determinado momento y altera su estado. A continuación se presenta el planteamiento formal de dicho modelo.

Este modelo está fundamentado en el modelo ecológico de la transferencia de energía basado en el modelo de Odum para resortes de plata [17].

Los componentes del modelo (nivel en la cadena alimenticia):

- P productores primarios (algas marinas, hierba, árboles, etc)
- H herbívoros (peces, tortugas, invertebrados)
- C carnívoros primarios (peces carnívoros e invertebrados)
- S carnívoros secundarios
- D descomponedores (insectos, bacterias)

El índice del cambio de cada nivel en la cadena alimenticia depende de la producción (reproducción y crecimiento) que es proporcional a la biomasa del nivel y a la biomasa del nivel del cual se alimenta. Por otro lado ese índice decrece con:

1. El índice de consumo del nivel de alimentación
2. Pérdida de biomasa por la respiración
3. Pérdida de biomasa por migración
4. Mortalidad

El valor de la biomasa es dado en Kcl/m2/año. El tiempo es dado en años.

Perturbaciones:

1. A los de 2.1 años comienza una extracción de plantas de 400 a 1000 (modo 800, distribución triángular).
2. A los de 8 años ocurre una destrucción de 7 en la biomasa de los descomponedores que luego se repite cada 2 años por la contaminación del

suelo [16].

Este modelo fue propuesto para GLIDER como un ejemplo de sistemas continuos, ya que está modelado usando un sistema de ecuaciones diferenciales ordinarias. Los sistemas dinámicos se han convertido en un método ampliamente usado en el análisis de gran cantidad de clases de sistemas, tales como crecimiento poblacional, sistemas económicos, dinámica urbana, entre otros.

Un sistema dinámico representa el cambio o evolución del estado del sistema en el tiempo, el comportamiento en dicho estado se puede caracterizar determinando los límites del sistema, los elementos y sus relaciones; de esta forma se puede elaborar modelos que buscan representar la estructura del mismo sistema [18]. En este contexto, podemos definir una perturbación como un agente externo al sistema que es capaz de ejercer una influencia en él pudiendo llegar a alterar el estado del sistema en un momento dado.

A continuación se describen las fases y procedimientos realizados para definir el meta – modelo a partir de este modelo de acuerdo a la metodología propuesta, en el marco del dominio del modelado y simulación correspondiente.

2.1 Análisis del Dominio

El dominio de modelado y simulación en el que se encuentra este sistema abarca mucho más que sólo los sistemas biológicos, ya que pertenece al dominio de los sistemas continuos. En este contexto, se realizó un análisis a la implementación de dicho modelo realizada en GLIDER (Anexo A) con el propósito de traducir dicha implementación a Java usando la biblioteca GALATEA, de tal manera que se identificara la estructura orientada a objetos del modelo.

A partir de la implementación del modelo en GLIDER se identificó la red de nodos usados para modelar el sistema. Dicha red está formada por un nodo continuo, en el cual se definen las ecuaciones diferenciales ordinarias y dos nodos autónomos en los que se definen las perturbaciones, las cuales pueden también ser expresadas mediante ecuaciones. En el ámbito de este proyecto una ecuación puede contener funciones matemáticas básicas, así como también funciones de probabilidad. En la figura 5 se muestra un diagrama con la nomenclatura GLIDER de la red de nodos del modelo.

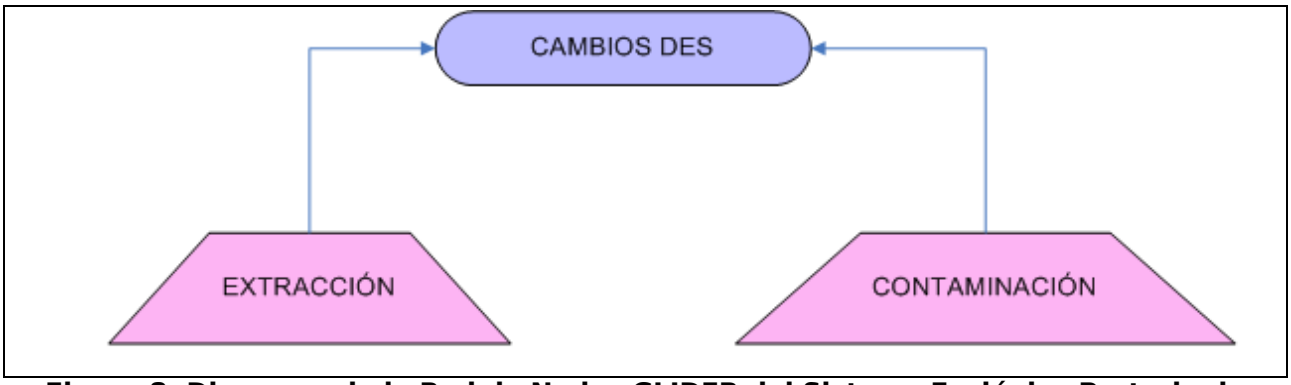


Figura 8: Diagrama de la Red de Nodos GLIDER del Sistema Ecológico Perturbado

Luego de realizado este análisis se procedió a elaborar la implementación orientada a objetos, la cual condujo a la diferenciación de cada ecuación como un objeto, aunque añadido a un nodo continuo en el proceso de definir el sistema de ecuaciones. De esta manera se determinó que en este sistema los aspectos primordiales que se tomaron en cuenta para la definición del meta – modelo son el sistema de ecuaciones diferenciales ordinarias, en este caso cinco, y las perturbaciones que intervienen, en este caso dos. A continuación se muestra una figura con el diagrama que representa el dominio del sistema ecológico perturbado.

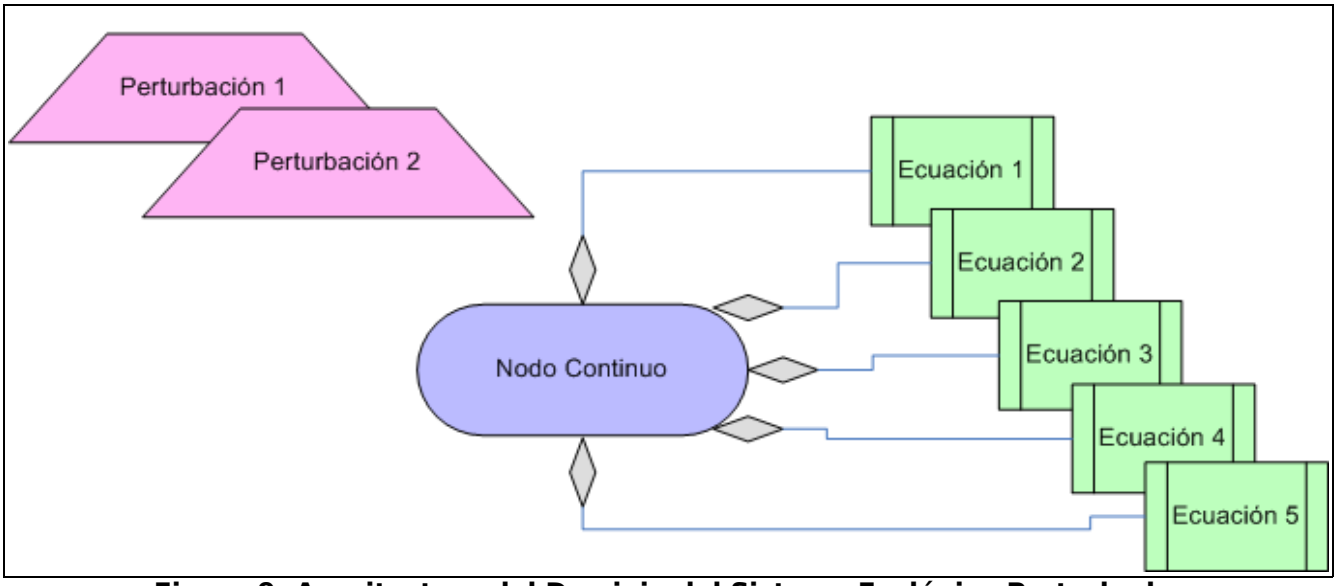


Figura 9: Arquitectura del Dominio del Sistema Ecológico Perturbado.

Por último se realizó un estudio orientado a definir la forma en que el usuario introduciría las ecuaciones diferenciales ordinarias de los modelos particulares al meta – modelo. En este contexto las ecuaciones pueden ser vistas como una igualdad entre dos expresiones matemáticas, sin importar el valor que tomen las variables implicadas en cada expresión [19]. En este modelo las ecuaciones son un aspecto de gran importancia ya que son las que determinan el comportamiento del

sistema dinámico, lo que conduce a que sean un aspecto fundamental en la arquitectura del dominio.

2.2 Diseño del Dominio

La representación del dominio definida en la "análisis del dominio", proporciona el punto de partida en esta etapa, ya que a partir de él se determinaron los aspectos del modelo que pueden derivar en estructuras genéricas y de esta manera lograr definir el meta – modelo. La arquitectura generalizada definida para este modelo se muestra en la siguiente figura.

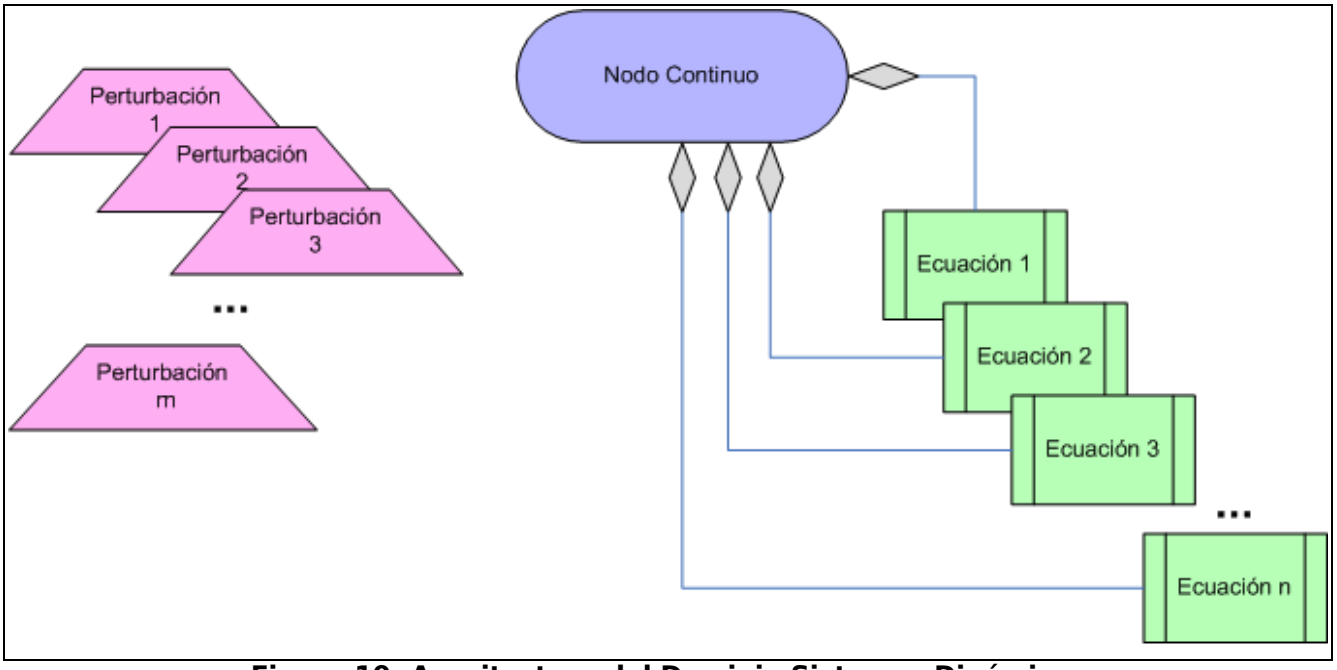


Figura 10: Arquitectura del Dominio Sistemas Dinámicos

Del diagrama puede observarse que uno de los aspectos configurables de este meta – modelo es la cantidad de ecuaciones del sistema, así cómo también la cantidad de perturbaciones que afectan al mismo. Así mismo, la estructura de las ecuaciones pueden generalizarse de manera tal que no sólo pueda estar formada por expresiones matemáticas simples, sino también puedan construirse ecuaciones compuestas de funciones matemáticas simples, como por ejemplo la función seno, coseno, tangente, etc., o funciones de probabilidad, como por ejemplo función binomial, normal, exponencial entre otras.

En este contexto, y en base al concepto de ecuación presentado en el apartado anterior, se define una expresión matemática como una palabra o cadena de caracteres pertenecientes al lenguaje matemático.

Los símbolos del alfabeto que pueden componer una expresión matemática son los siguientes:

- Números.
- Operadores.

- El alfabeto inglés.
- Signos de puntuación [20].

Con este grupo de símbolos pueden construirse cualquiera de los elementos de alto nivel que forman las ecuaciones. Estos elementos pueden ser:

i. Constantes: formadas por cualquier cadena de caracteres numéricos y/o signos de puntuación.

ii. Variables: pueden ser cualquier cadena de caracteres alfanuméricos.

iii. Operadores matemáticos.

iv. Funciones matemáticas simples, como por ejemplo función seno, función coseno, etc.

v. Funciones de probabilidad, como por ejemplo exponencial, binomial, normal, etc.

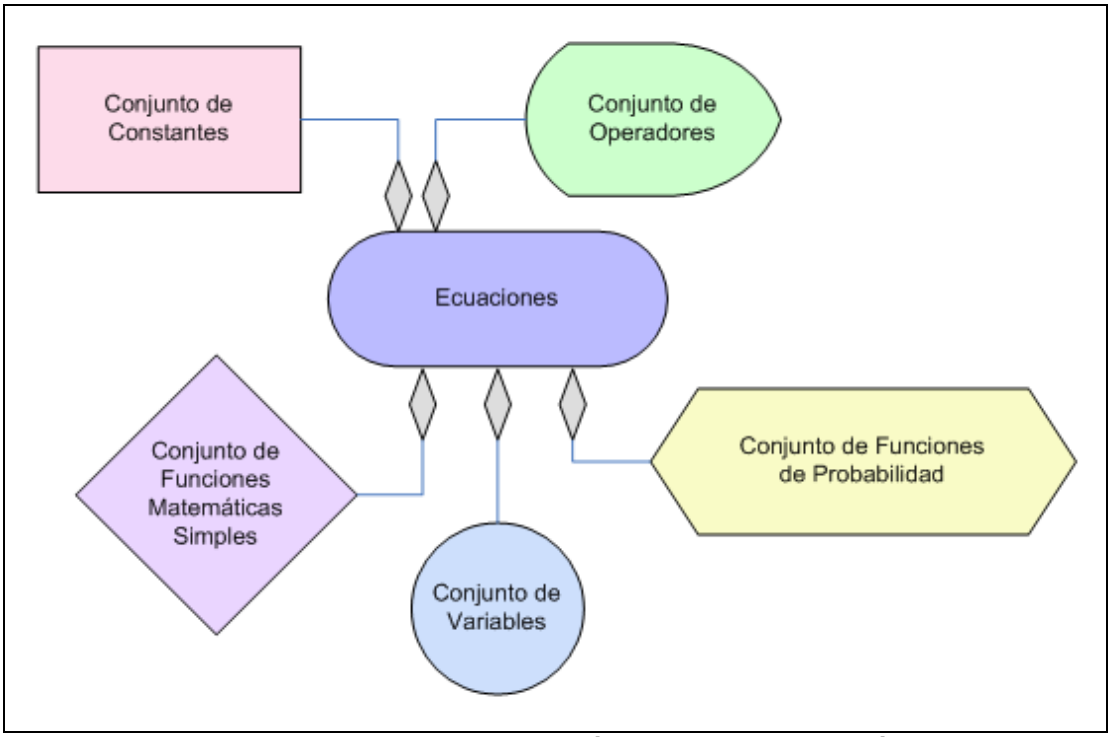


Figura 11: Estructura Genérica de una Ecuación

La arquitectura generalizada del dominio proporciona el marco de trabajo para determinar los aspectos configurables del modelo de manera que se pueda llegar a definir el meta – modelo, en este caso de un Sistema Ecológico Perturbado. En la sección “análisis del dominio” se definió un sistema ecológico perturbado genérico, en esta fase, se parametrizó dicha definición de manera tal, que el meta – modelo permita configurar todas las variantes del modelado que se puedan encontrar en un sistema ecológico perturbado.

De esta manera los aspectos configurables que se especificaron para este meta – modelo son la cantidad de ecuaciones que forman el sistema de ecuaciones diferenciales ordinarias y la cantidad de perturbaciones que afectan al sistema dinámico. Cabe destacar que el número de perturbaciones que afectan el sistema puede ser cero, lo que significa que el meta – modelo no está limitado a modelos de sistemas con alteraciones externas, sino que también puede ser usado para sistemas sin alteraciones.

Adicionalmente, la estructura fijada para las ecuaciones es el aspecto que proporciona la mayor generalización, ya que no limita el uso de una ecuación específica, sino que proporciona el mecanismo para construir las ecuaciones que sean necesarias, del tamaño que sea necesario y formada por una combinación de cualquier número de elementos pertenecientes a los conjuntos previamente definidos.

2.3 Arquitectura de Software

El componente de software principal involucrado en el meta – modelo es la biblioteca de simulación GALATEA, la cual proporciona el motor para la simulación de sistemas continuos.

La generalización especificada para las ecuaciones implica que éstas son introducidas a la aplicación en forma de cadenas de caracteres, desde el punto de vista del tipo de dato **String** en el lenguaje de programación Java. Esto trae como consecuencia la necesidad de que la aplicación evalúe la expresión matemática correspondiente a su representación en cadena de caracteres en tiempo de ejecución.

Las funciones de evaluación permiten tomar cualquier cadena de caracteres del lenguaje en cuestión (contenida en una variable) y ejecutarla como parte de la aplicación en tiempo de ejecución. Ya que Java es un lenguaje compilado no posee ningún tipo de función de evaluación, la cual si poseen los lenguajes interpretados tales como php o JavaScript.

La necesidad de evaluar expresiones matemáticas contenidas en una variable en Java en tiempo de ejecución, derivó en la inclusión de un componente de software que proporcionara dichas capacidades.

El tipo de aprovechamiento de este meta – modelo es la visualización, lo que implica la necesidad de generar gráficos de las variables, para lo cual será usado el componente generador de gráficos implementado en la interfaz gráfica de usuario.

Adicionalmente, es necesario definir estructuras de datos específicas que almacenen las configuraciones realizadas por los usuarios a la hora de instanciar el meta – modelo en un modelo específico. Así mismo, se deben definir estructuras de datos para mantener y retornar los resultados del modelo luego de ser ejecutado.

Otro componente necesario corresponde a la interfaz que debe proporcionar el meta – modelo para ser usado desde la interfaz gráfica de la aplicación. Esta

interfaz permite que el meta – modelo sea tratado como un componente de software reutilizable, ya que desde la aplicación gráfica puede ser creado, configurado y luego realizado el proceso de simulación del mismo. Igualmente, luego de ser simulado puede retornar los resultados para que desde la interfaz gráfica sean manipulados y mostrados.

A continuación se muestra un diagrama con los componentes identificados en la arquitectura de software.

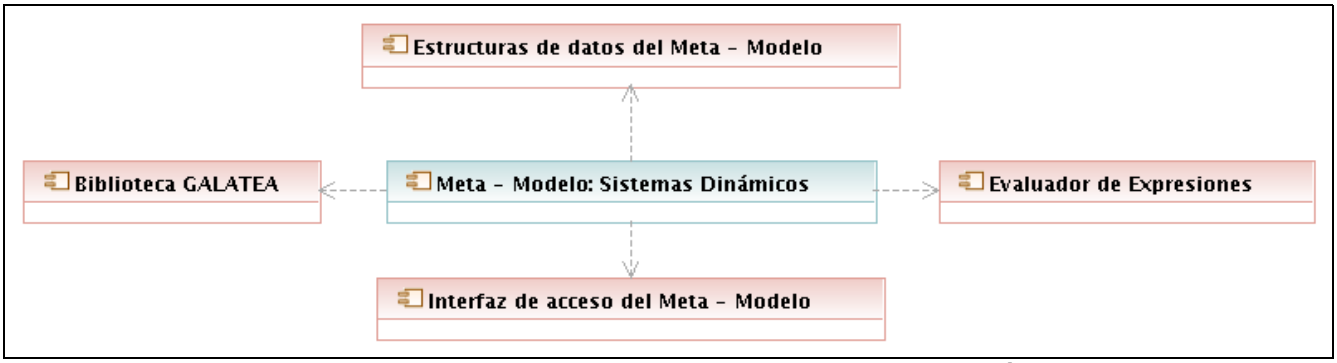


Figura 12: Componentes de Software: Sistemas Dinámicos

2.4 Implementación del Dominio

El primer paso fue definir e implementar las estructuras de datos necesarias para interactuar con la interfaz gráfica, las cuales permiten pasar los parámetros y configuraciones desde el nivel más alto de la aplicación hasta el nivel de meta – modelo. A continuación se muestran los diagramas de dichas estructuras.

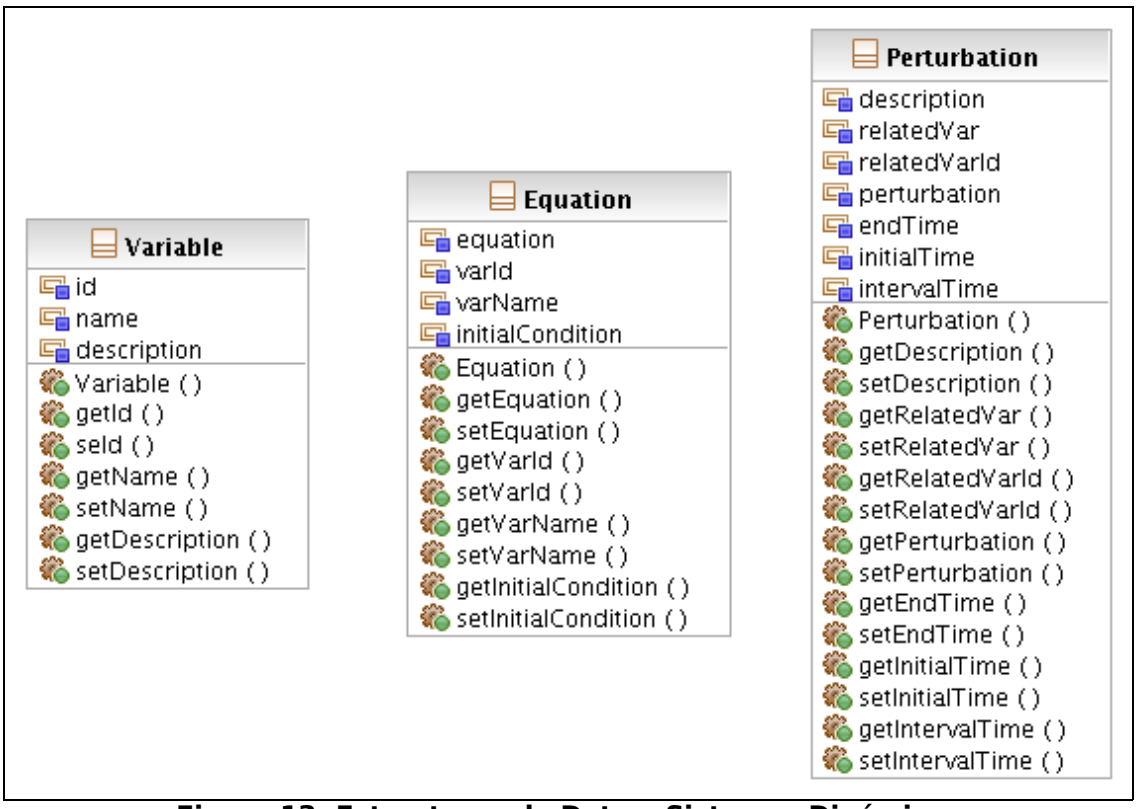


Figura 13: Estructuras de Datos: Sistemas Dinámicos

Para la evaluación de las expresiones matemáticas en tiempo de ejecución, se realizó una búsqueda de aplicaciones de software libre que implementaran dicha funcionalidad. Entre varias bibliotecas evaluadas se escogió la biblioteca JEPLite [21], pues proporciona la posibilidad de agregar funciones matemáticas adicionales a las que ya trae definidas de una manera sencilla.

Una vez seleccionada la biblioteca se procedió a la modificación del código fuente para añadir las características que proporcionan el soporte de todas las funciones de probabilidad implementadas en GALATEA, de manera que también estén accesibles al momento de realizar la traducción y evaluación de las expresiones matemáticas durante la simulación.

Luego de solucionado este aspecto, se procedió a realizar la implementación de las clases que describen el meta – modelo. En este sentido, se definió la clase PerturbationNode, en la cual se modelan las perturbaciones. Dicha clase hereda de Node, en la cual se define el nodo autónomo. En esta clase se programa la activación del nodo de acuerdo a lo definido en la perturbación, tiempo de inicio, tiempo final y frecuencia de ocurrencia de la perturbación. Así mismo, cuando ocurre la activación, se altera el estado de la variable a la que afecta la perturbación. En este proceso se usa la evaluación de la expresión matemática que represente la perturbación.

Adicionalmente, se define la clase EquationCont, la cual hereda de Cont, donde se definen las ecuaciones diferenciales ordinarias. Aquí también se hace uso de la

biblioteca de evaluación de expresiones para poder interpretar la expresión matemática definida por el usuario a la hora de configurar el modelo. Junto con esta clase se implementó la clase EquationsSystemNode, la cual hereda de Node e implementa un nodo tipo continuo. A esta clase se le agregan instancias de la clase EquationCont, definiendo de esta manera el sistema de ecuaciones diferenciales ordinarias.

Por último, se define la clase DynamicSystem en la cual se programa la creación de la red de nodos de simulación específica para el modelo, se establece el tiempo de simulación, las salidas de traza y las estadísticas de GALATEA, se inician las semillas de los generadores de números aleatorios y se procesa la simulación.

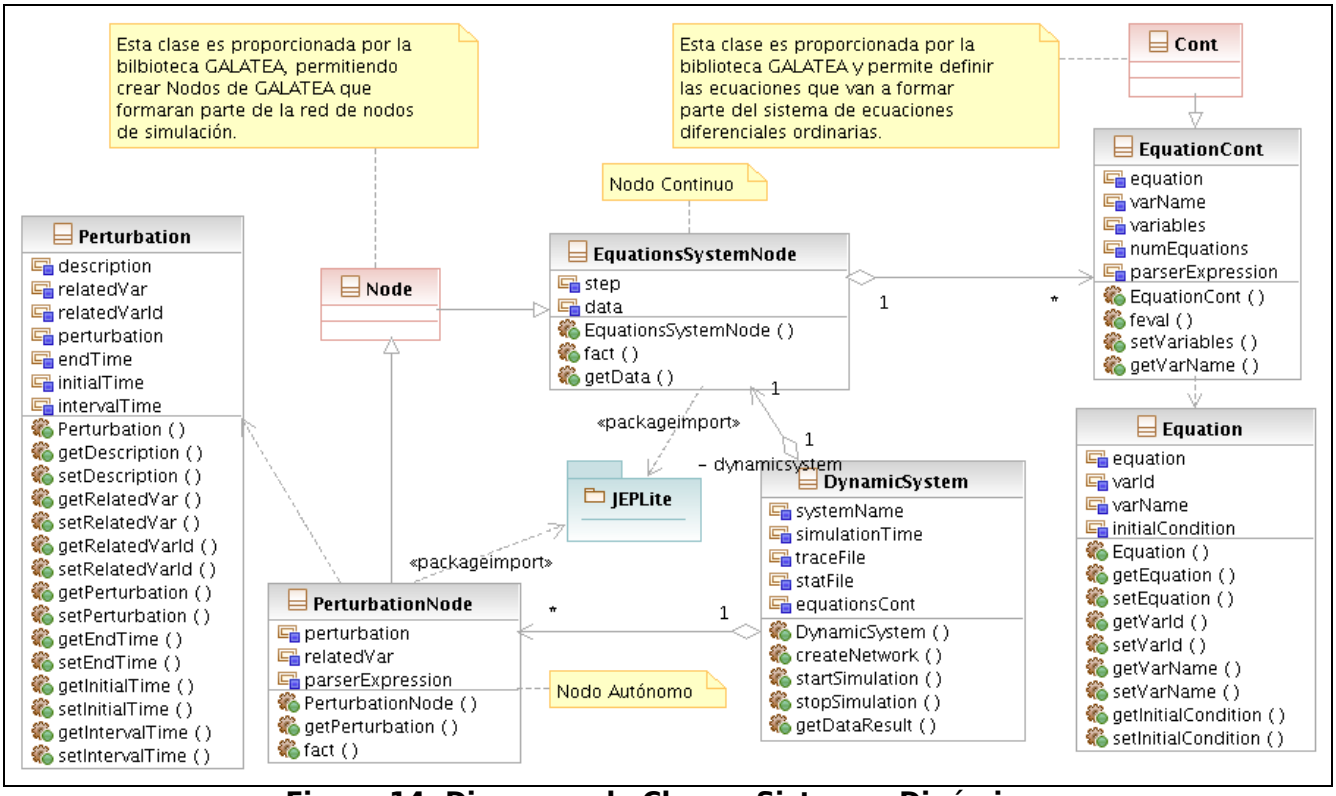


Figura 14: Diagrama de Clases: Sistemas Dinámicos

Este meta – modelo no se limita a sistemas basados en ecuaciones diferenciales ordinarias en los cuales hayan factores externos que afecten al sistema, ya que la generalización que se hizo hace posible que esta capacidad sea también configurable, por lo que puede ser usado para instanciar una variedad de sistemas dentro de este dominio de modelado. Es decir, es posible generar modelos específicos de cualquier sistema con una estructura isomorfa a un sistema dinámico descrito con ecuaciones diferenciales ordinarias.

Capítulo III

Meta – Modelo: Sistema Matricular

El segundo modelo seleccionado para formar parte de la META - MODELOTECA fue el Modelo Matemático de Simulación Matricular [20], clasificado en el área de conocimiento de Sistemas Sociales, tipo de aprovechamiento Visualización y complejidad Media.

Este modelo describe el flujo matricular de los niveles de educación preescolar, básica, media y diversificada, los cuales son sistemas educativo de permanencia cerrada. Es decir, este sistema matricular se define con una cantidad de periodos académicos y una cantidad de materias por cada periodo. Hay prelación por nivel o período. No hay un sistema de prelaciones entre cursos o temas, sino que se debe aprobar el periodo académico completo para poder avanzar al siguiente. A continuación se presenta el planteamiento formal de este modelo:

El modelo matemático de simulación matricular permite analizar el aspecto cuantitativo de los subsistemas de Educación Preescolar, Básica, Media y Diversificada. Este sistema presenta características ideales para su análisis como son: la cantidad de profesores por asignatura necesarios para atender la matricula, el número de secciones que se necesitan para cada grado, número de alumnos al final de cada grado y algunos indicadores importantes tales como el índice de promoción, el índice de permanencia y el índice de variación de matrícula.

Los resultados del modelo servirán para estimar posteriormente los recursos que se deben invertir para atender incrementos en la matricula. Con lo cual podemos obtener de forma inmediata las estimaciones de matricula.

Para analizar sistemas educativos el modelo de flujo matricular es uno de los más utilizados. a través de este se puede estimar el comportamiento de la matricula estudiantil en los diferentes grados que conforman el sistema educativo. Algo muy útil desde el punto de vista cualitativo, ya que nos permite destinar recursos económicos para cubrir con eficiencia la ejecución de la planificación educativa.

En el análisis del flujo matricular se suelen establecer algunas relaciones básicas, las cuales permiten observar el movimiento de efectivos a través de todo el sistema educativo, o parte de él [22].

Los parámetros que usamos en este modelo son el resultado de un trabajo de análisis estadístico de datos históricos de matrícula escolar en el estado Mérida. A continuación se describen los procedimientos realizados para definir el meta – modelo a partir de este modelo de acuerdo a la metodología propuesta, en el marco del dominio del modelado y simulación correspondiente.

3.1 Análisis del dominio

El dominio de modelado y simulación de este sistema no está basado en el formalismo de eventos discretos, sino que se fundamenta en cálculos matemáticos ejecutados por cada unidad de tiempo durante el período de simulación. En este contexto, se revisó cuidadosamente la implementación del modelo para el nivel básico y medio de educación, realizadas en GLIDER [22] (Anexo B), para determinar la estructura de la red de simulación.

A partir de la implementación del modelo en GLIDER se identificó que la red de simulación de GLIDER estaba formada por un solo nodo autónomo en el cual se establece el modelo matemático analizado. Este modelo matemático está formado por un conjunto de ecuaciones en las cuales se toman en cuenta parámetros como horas semanales de trabajo de profesores o número máximo de estudiantes por aula.

Al inicio del modelo se encuentra la ecuación de crecimiento poblacional exponencial del sistema, la cual es presentada a continuación:

l1 = TRUNC(15395*EXP(0,012754*TIME));

Siendo 15395 la población inicial del sistema, 0,012754 la constante de crecimiento, TIME representa el tiempo y l1 representa la cantidad de alumnos que inicialmente se incorporan al primer periodo.

Las materias o cursos son representadas en una lista formada por dieciséis nombres, los cuales se usan para definir la cantidad de profesores por materia y asignar las materias que componen cada periodo académico y sus correspondientes horas de clase.

Los periodos académicos poseen varios parámetros entre los que están los coeficientes de permanencia y deserción, así como también el porcentaje de estudiantes externos incorporados al periodo. En este caso el sistema matricular está formado por nueve periodos académicos.

Mediante todos estos componentes, se realizan los cálculos de los cuales se obtienen los siguientes datos:

- Índice de promoción para cada periodo, el cual viene dado por el total de aprobados entre el total de alumnos para ese periodo. Este índice calcula la cantidad relativa de alumnos que son promovidos de un periodo al siguiente.
- Índice de permanencia, que es la relación entre el total de aprobados en un periodo entre el total de aprobados del periodo anterior. Este índice es equivalente a la probabilidad

- Índice de variación de matricula, que es la relación entre la cantidad de alumnos que inician un periodo dividido entre la cantidad de alumnos que iniciaron el periodo anterior. Este índice calcula el incremento relativo de la matrícula del periodo inmediato anterior.
- Así mismo, se obtienen estadísticas como la cantidad de profesores necesarios por materia, entre otras cosas.

3.2 Diseño del Dominio

A partir del análisis previo, se establecieron los aspectos del modelo que pueden derivar en estructuras genéricas y de esta manera lograr definir el meta – modelo. La arquitectura generalizada definida para este modelo se muestra en la figura 13. En el modelo particular en estudio n=9 y m=16.

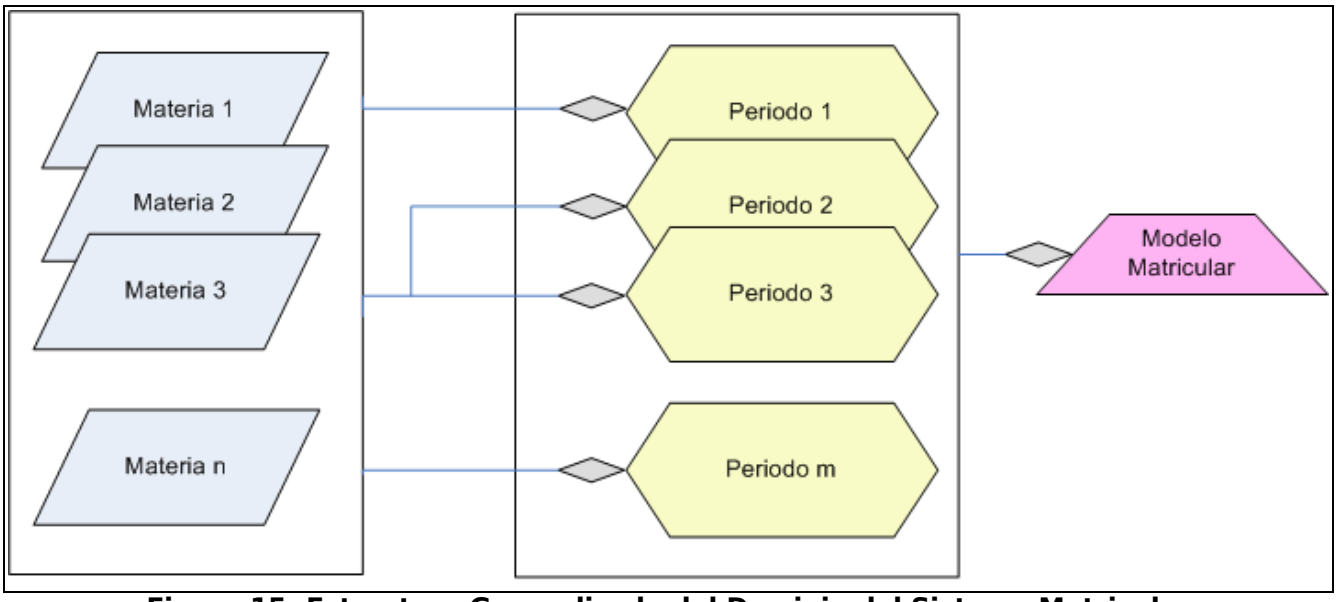


Figura 15: Estructura Generalizada del Dominio del Sistema Matricular

La estructura generalizada del dominio proporciona la base para determinar los aspectos configurables para el modelo, los cuales son la cantidad de materias total que intervienen en el modelo, la cantidad de periodos que forman el sistema escolar, las materias pertenecientes a cada periodo y sus horas de clase correspondientes. Adicionalmente en cada periodo puede ser configurado los coeficientes de deserción, de permanencia y de estudiantes externos incorporados al periodo. Aunado a esto también se permite configurar los valores para la población inicial del sistema y la constante de crecimiento poblacional.

3.3 Arquitectura de Software

La biblioteca de simulación GALATEA es un componente fundamental en el meta – modelo, la cual proporciona el motor para la simulación del modelo matemático.

El tipo de aprovechamiento de este meta – modelo es la visualización, lo que

implica la necesidad de generar gráficos de las variables, para lo cual será usado el componente generador de gráficos implementado en la interfaz gráfica de usuario.

Es necesario definir estructuras de datos específicas que almacenen las configuraciones realizadas por los usuarios a la hora de instanciar el meta – modelo, así como también las estructuras de datos para mantener y retornar los resultados del modelo luego de ser ejecutado.

Otro componente necesario corresponde a la interfaz que debe proporcionar el meta – modelo para ser usado desde la interfaz gráfica de la aplicación. Esta interfaz permite que el meta – modelo sea tratado como un componente de software reutilizable, ya que desde la aplicación gráfica puede ser creado, configurado y ejecutado. Igualmente, luego de ser ejecutado puede retornar los resultados para que desde la interfaz gráfica sean manipulados y mostrados.

A continuación se muestra un diagrama con los componentes identificados en la arquitectura de software.

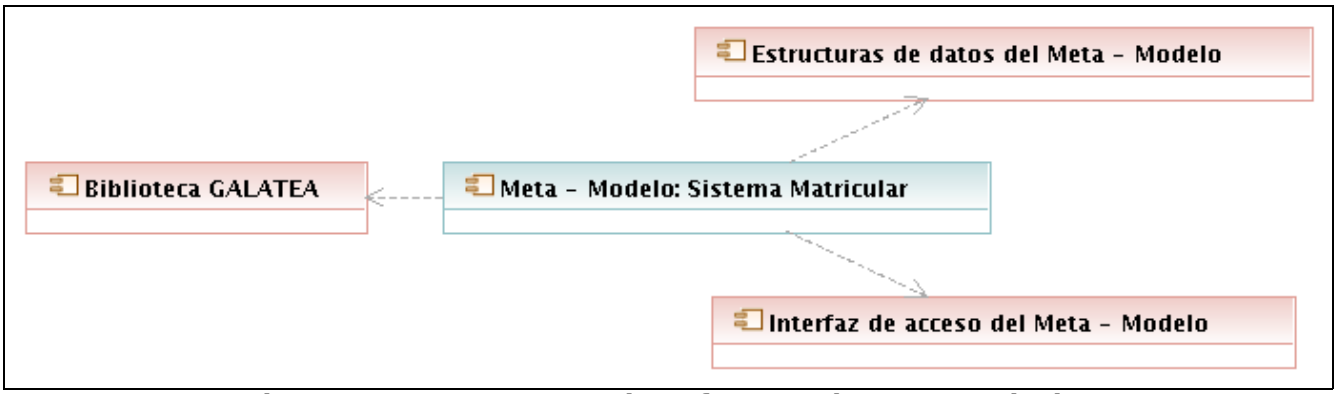


Figura 16: Componentes de Software: Sistema Matricular

3.4 Implementación del Dominio

El primer paso fue definir e implementar las estructuras de datos necesarias para interactuar con la interfaz gráfica, las cuales permiten pasar los parámetros y configuraciones desde el nivel más alto de la aplicación hasta el nivel de meta – modelo. A continuación se muestran los diagramas de clases de dichas estructuras.

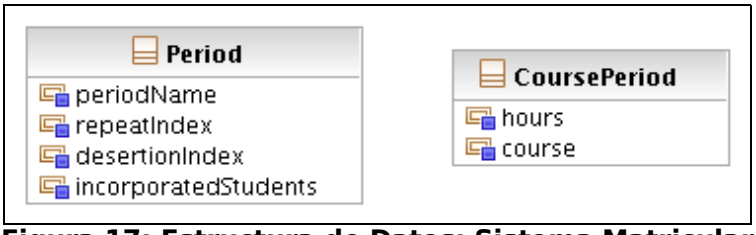


Figura 17: Estructura de Datos: Sistema Matricular

Una vez definidas las estructuras de datos necesarias para capturar la configuración del usuario se procedió a realizar la implementación de las clases que

describen el meta – modelo. En este sentido, se implementó la clase MatematicalModelNode, la cual hereda de la clase Node de GALATEA e implementa un nodo tipo autónomo. En dicha clase se realizan todos los cálculos necesarios para obtener los resultados del modelo matemático basados en la cantidad de periodos y materias definidos por el usuario para el modelo instanciado.

Adicionalmente se creó la clase EnrollSystem en la cual se programa la creación de la red de simulación específica para el modelo, se establece el tiempo de simulación, las salidas de traza y las estadísticas de GALATEA, se inician las semillas de los generadores de números aleatorios y se procesa la simulación.

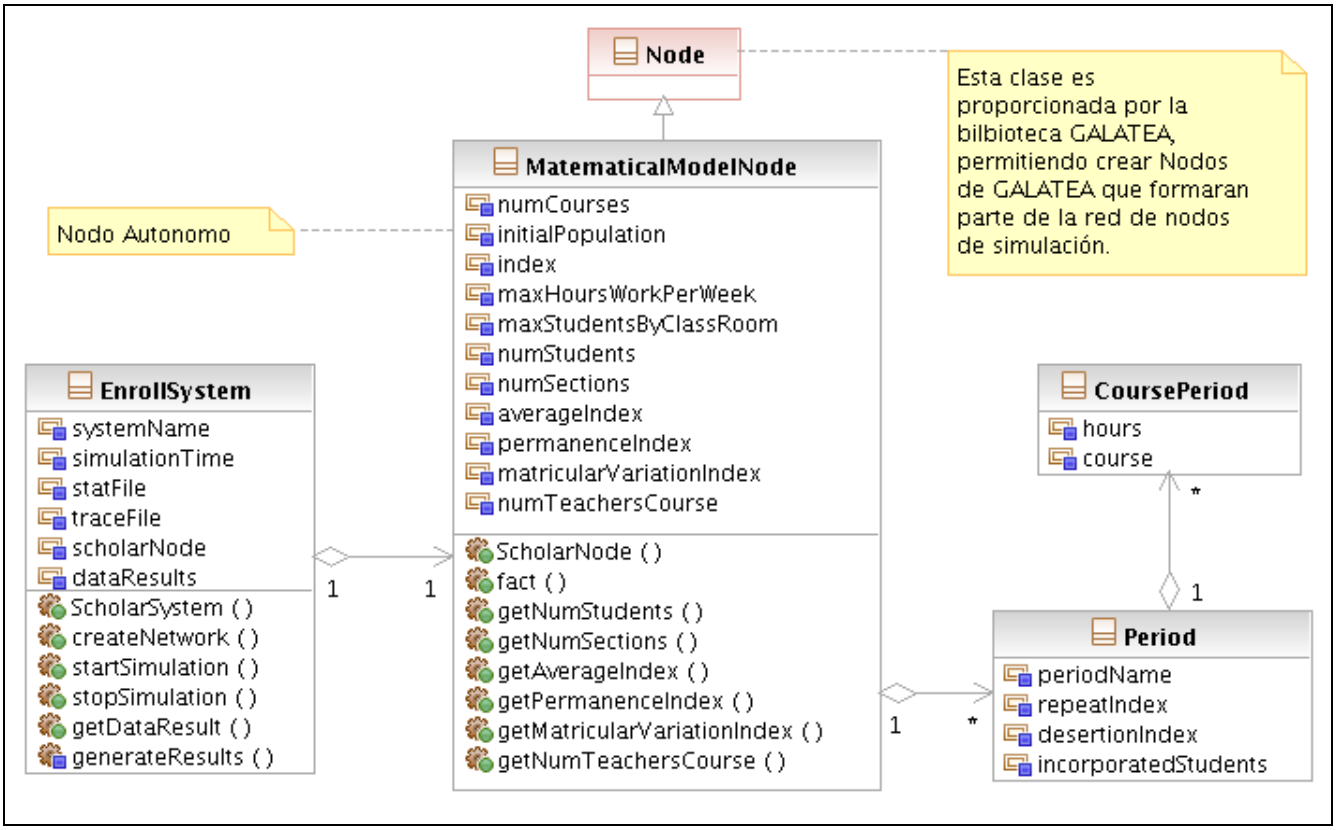


Figura 18: Diagrama de Clases Sistema Matricular

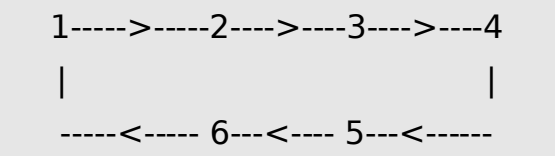
Capítulo IV

Meta – Modelo: Sistemas de Transporte

El tercer modelo seleccionado para formar parte de la META - MODELOTECA fue el Modelo de Transporte o Ruta de Autobús [16], clasificado en el área de conocimiento de Sistemas Sociales, tipo de aprovechamiento Animación y complejidad Media. Este modelo describe el funcionamiento de una ruta de autobuses cerrada, lo que implica que cuando un autobús llega a la última parada, la siguiente parada es la primera. Se toman en cuenta parámetros tales como la probabilidad de pasajeros que viajen de la parada ***X*** hasta la parada ***Y*** y tiempo promedio que le toma a un autobús llegar de una parada a otra.

A continuación se presenta el planteamiento formal de dicho modelo.

Cuatro autobuses funcionan en una ruta cerrada con seis paradas. La parada número uno es el origen, el terminal:



Los pasajeros llegan a cada parada a intervalos que tiene una distribución exponencial. La media es diferente para cada parada. Un destino es asignado a cada pasajero de una manera aleatoria basada en una distribución de frecuencia dada por una matriz de frecuencias origen – destino. La fila **I** y columna **J** es proporcional a la probabilidad de que un pasajero que toma el autobús en la parada **I** deje el autobús en la parada **J.**

Cuando un autobús alcanza una parada los pasajeros con ese destino dejan el autobús. El número de pasajeros que toman el autobús es el mínimo entre la longitud de la cola y la capacidad libre en el autobús.

El autobús viaja a la próxima parada. La trayectoria es cíclica, es decir, cuando el autobús está en la parada seis su destino es la parada uno [16].

Este modelo fue propuesto como un ejemplo de sistemas de eventos discretos en GLIDER [16]. El formalismo de modelado y simulación basado en eventos

discretos permite representar un sistema cuyo comportamiento entrada salida pueda ser descrito por una secuencia de eventos sujeto a que el estado tenga un número finito de cambios en cualquier intervalo finito de tiempo. En este contexto, un evento es la representación de un cambio instantáneo en alguna parte de un sistema.

4.1 Análisis del Dominio

El dominio de modelado en el que se encuentra este sistema, el formalismo de simulación por eventos discretos, permite modelar varios tipos de sistemas en los cuales la ocurrencia de ciertos eventos afecten el estado del sistema. En concreto, este sistema abarca muchos sistemas de transporte de pasajeros. En este contexto, se realizó un análisis a la implementación de dicho modelo realizada en GLIDER (Anexo C) con el propósito de identificar la red de nodos de simulación de este modelo.

La red de nodos usados para modelar el sistema está formada como se describe a continuación: en primer lugar posee un nodo tipo entrada que se encarga de generar los buses que entran al sistema y otros seis nodos tipo entrada encargados de generar los pasajeros para cada parada. Luego tiene otros seis nodos tipo recurso que representan las paradas a las cuales llegan los pasajeros y esperan el autobús. Adicionalmente existen seis nodos tipo recurso los cuales modelan el viaje de los autobuses de una parada a otra. Por último existe un nodo tipo salida en el cual los pasajeros salen del sistema. En la siguiente figura se muestra un diagrama con la nomenclatura GLIDER de la red de nodos del modelo.

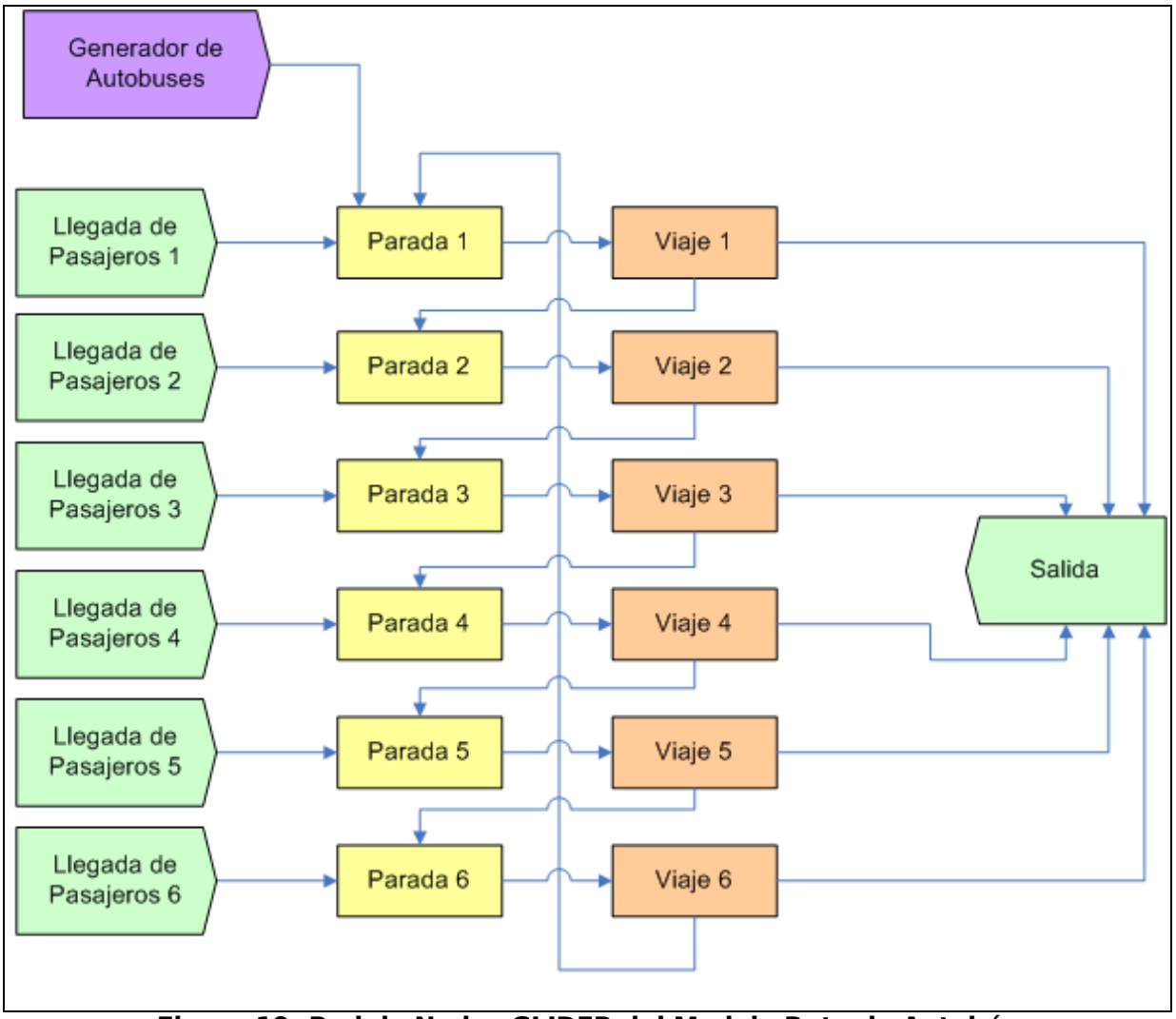


Figura 19: Red de Nodos GLIDER del Modelo Ruta de Autobús

4.2 Diseño del Dominio

La estructura de nodos GLIDER del modelo proporciona un punto de partida para esta fase, ya que a partir de él se determinaron los aspectos del modelo que pueden derivar en estructuras genéricas y de esta manera lograr definir el meta – modelo. La arquitectura generalizada definida para este modelo se muestra en la siguiente figura.

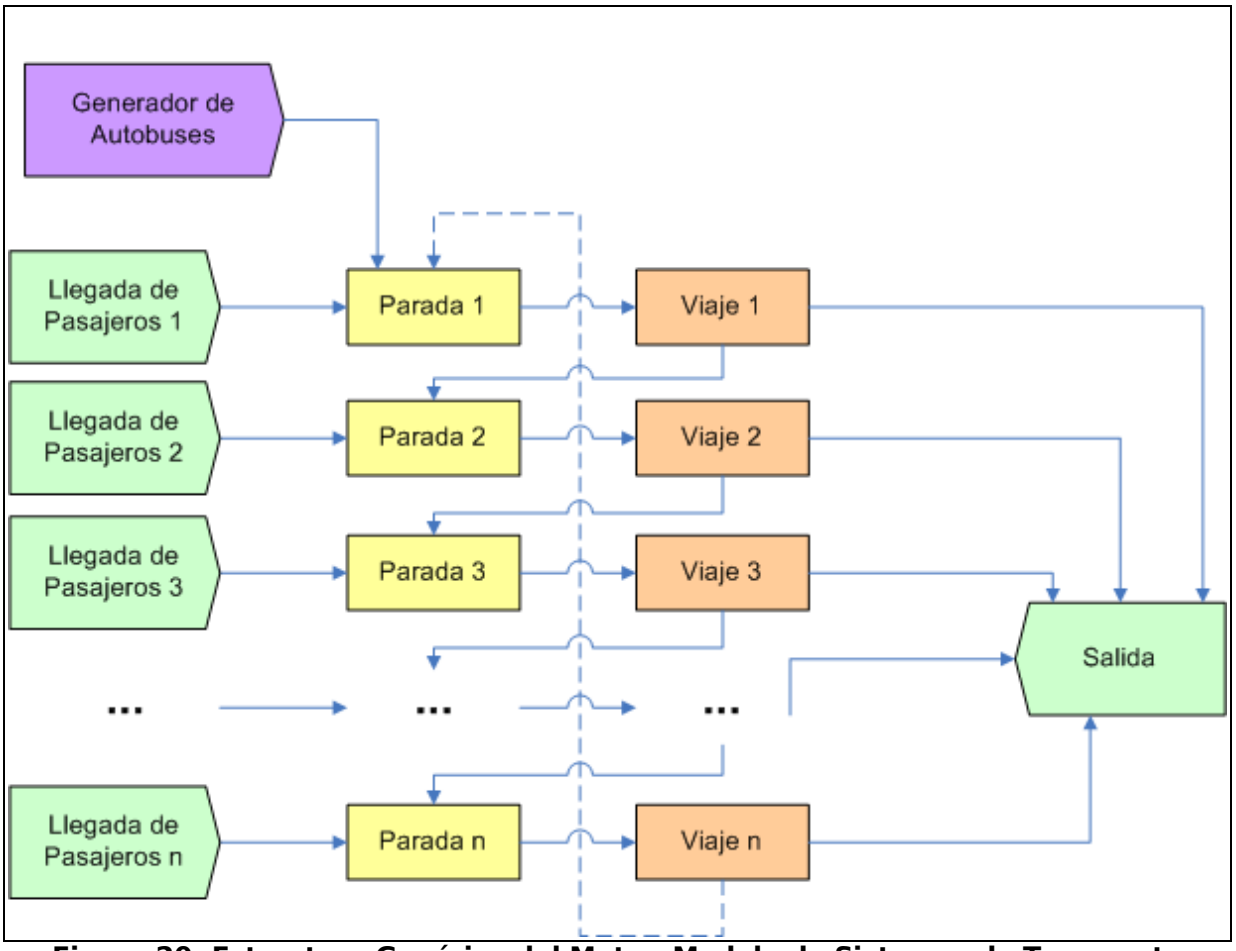


Figura 20: Estructura Genérica del Meta - Modelo de Sistemas de Transporte

Del diagrama puede observarse que uno de los aspectos configurables de este meta – modelo es la cantidad de paradas que posee el sistema, así cómo también puede ser configurable si la ruta es cerrada o abierta, es decir, si la ruta es cíclica o una vez que un autobús ha llegado a la última parada sale del sistema. Adicionalmente, todos los parámetros del modelo, como por ejemplo, la media entre llegada de pasajero a una parada, la media del tiempo de los viajes, o la probabilidad de que cierto pasajero en la parada **X** vaya a la parada ***Y*** son parámetros configurables.

Esta fase proporciona el marco de trabajo para lograr la implementación del meta - modelo de transporte.

4.3 Arquitectura de Software

El primer y más importante componente de software que intervienen en este meta – modelo es la biblioteca de simulación GALATEA, la cual proporciona la plataforma de simulación basada en eventos discretos.

Otro de los componentes de software identificados en este meta – modelo es el relacionado con las estructuras de datos necesarias para almacenar las

configuraciones realizadas por los usuarios a la hora de instanciar el meta – modelo y las estructuras de datos para guardar los resultados de la simulación del modelo.

Adicionalmente, el tipo de aprovechamiento definido para este modelo es la Animación, lo que conlleva a la necesidad de integrar una biblioteca que permita implementar esta funcionalidad o crear un componente a partir de las clases para el manejo de imágenes que proporciona la biblioteca estándar de Java.

Por último, es necesario definir la manera de cómo la interfaz gráfica va a interactuar con el meta – modelo, como se va a realizar el proceso de instanciación y de retorno de resultados, para luego ser mostrados. Para esto se definen un nivel de abstracción del meta – modelo que proporciona la dinámica de acceso para la interfaz gráfica.

A continuación se muestra un diagrama con los componentes identificados en la arquitectura de software.

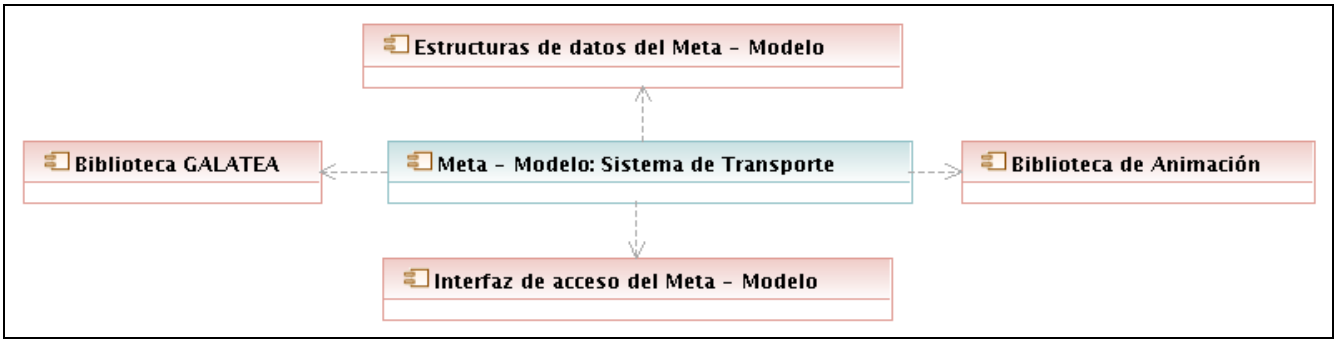


Figura 21: Componentes de Software Sistema de Transporte

4.4 Implementación del Dominio

El primer paso fue definir e implementar las estructuras de datos necesarias para interactuar con la interfaz gráfica, las cuales permiten pasar los parámetros y configuraciones desde el nivel más alto de la aplicación hasta el nivel de meta – modelo. A continuación se muestran los diagramas de dichas estructuras.

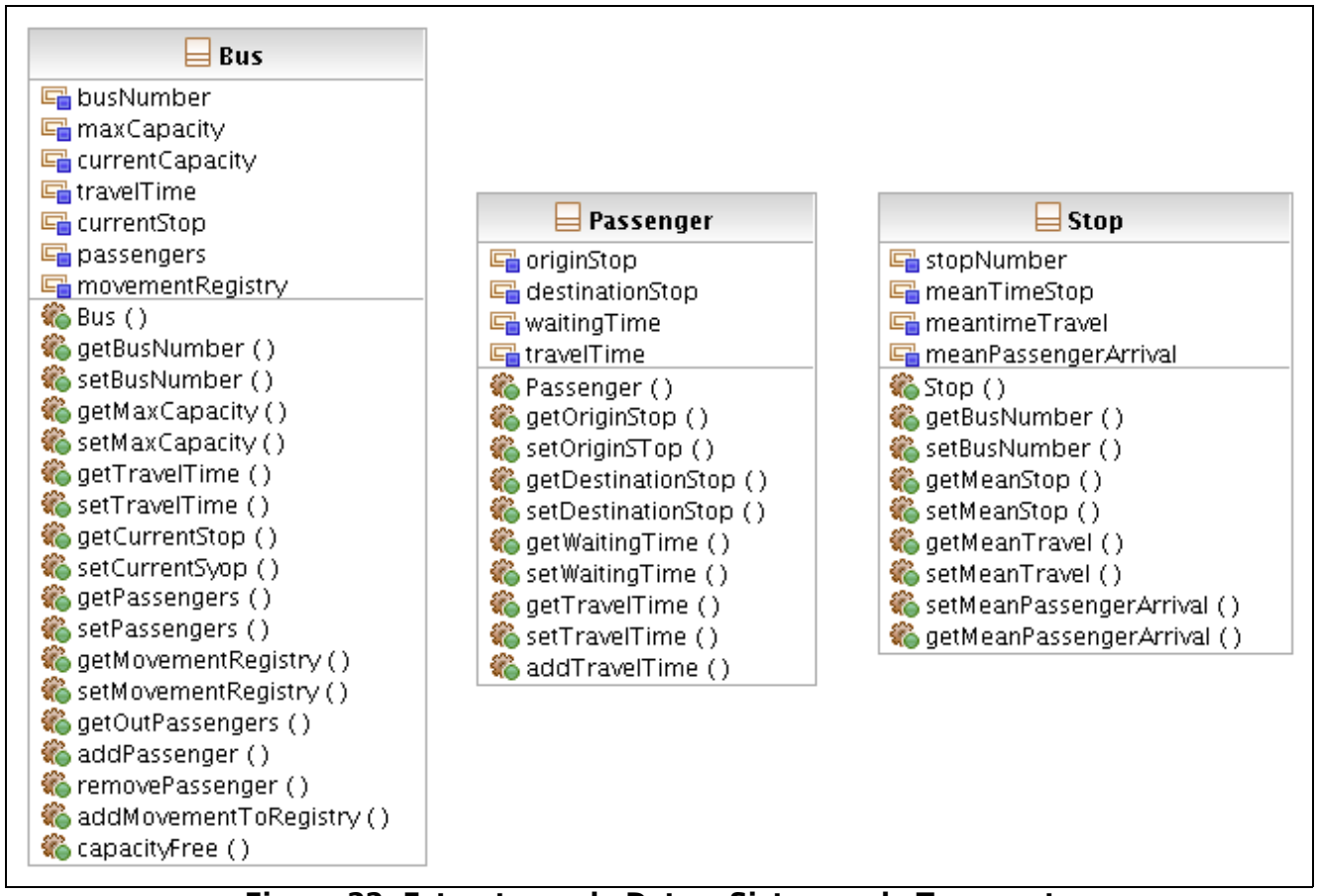


Figura 22: Estructuras de Datos: Sistemas de Transporte

Una vez definidas las estructuras de datos necesarias para capturar la configuración del usuario se procedió a realizar la implementación de las clases que describen el meta – modelo. En este sentido, se implementó la clase BusArrivalNode, la cual hereda de la clase Node de GALATEA e implementa un nodo tipo entrada. Dicha clase representa el generador de buses del sistema.

Luego de definidas las estructuras de datos se procedió a definir la clase PassengerArrivalNode, la cual hereda de la clase Node de GALATEA e implementa, al igual que el BusArrivalNode, un nodo tipo entrada. Dicha clase representa el generador de llegadas de pasajeros. El número de instancias de esta clase en el sistema dependerá del número de paradas definidas por el usuario.

Así mismo, se crearon las clases StopNode y TravelNode, la cuales heredan de Node e implementan un nodo tipo recurso. En la clase StopNode se modela la parada con la cola de pasajeros y la clase TravelNode representa el viaje de un autobús desde una parada a otra. El número de instancias de esta clase en el sistema, al igual que con la clase PassengerArrivedNode, dependerá del número de paradas definidas por el usuario.

Adicionalmente, se definió la clase ExitNode, la cual hereda de la clase Node e implementa un nodo de salida. Esta clase permite modelar las salidas de los

pasajeros del sistema.

Por último, se definió la clase TransportationSystem en la cual se programa la creación de la red de nodos de simulación específica para el modelo, se establece el tiempo de simulación, las salidas de traza y estadísticas de GALATEA, se inician las semillas de los generadores de números aleatorios y se procesa la simulación.

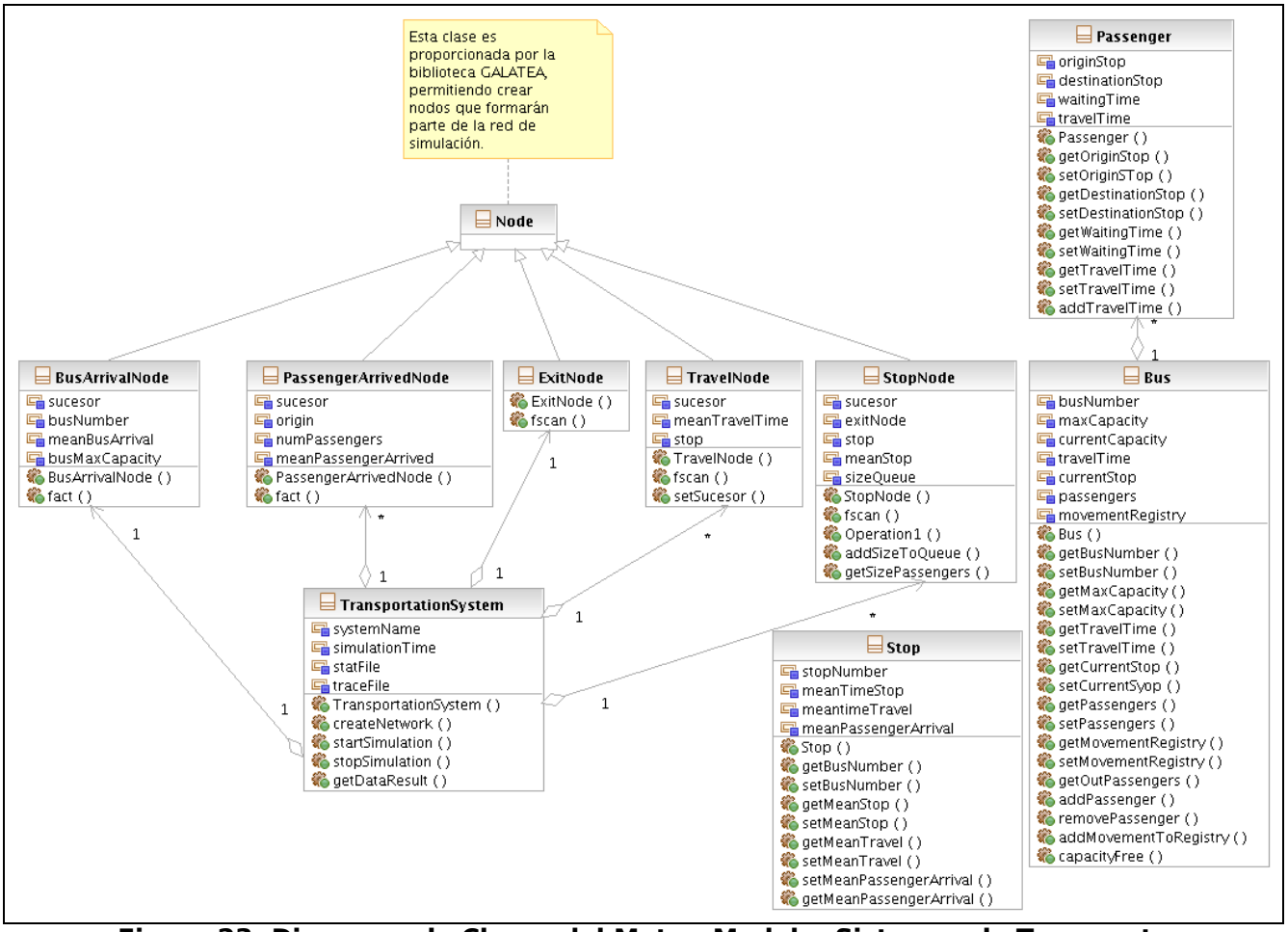


Figura 23: Diagrama de Clases del Meta - Modelo: Sistemas de Transporte

Capítulo V

Meta – Modelo: Sistema de Llenado

El cuarto modelo seleccionado para formar parte de la META - MODELOTECA fue el Modelo de Puerto de Buques Petroleros [16], clasificado en el área de conocimiento de Sistemas Artificial, tipo de aprovechamiento Animación y complejidad Media. Este modelo describe el funcionamiento de un centro de llenado de buques petroleros, en el cual sirven cinco tipos de petróleo en cinco diferentes atracaderos. A este centro de llenado pueden llegar diferentes tipos de buques de acuerdo a la capacidad de los buques, que pueden ser cinco volúmenes diferentes y del tipo de petróleo que sirven, lo que proporciona veinticinco tipos de buques posibles.

A continuación se presenta el planteamiento formal de dicho modelo.

En un puerto para los buques de petróleo se envían 5 tipos de petróleo que van por tubos a 5 diferentes tanques en el puerto.

Hay un flujo constante de petróleo desde los depósitos cerca de los pozos de petróleo a los tanques. El flujo a un tanque se detiene solamente cuando el petróleo alcanza un volumen máximo en el tanque. El flujo se reinicia cuando, dado a la extracción para llenar los buques, el volumen es reducido a una cierta cantidad por debajo de un mínimo nivel permitido.

Cada tanque es asociado con una litera en el puerto en la cual viene el buque a ser llenado. De cada tanque el petróleo se bombea al buque petrolero a una taza constante. Este flujo sólo para cuando el buque petrolero se llena con la cantidad requerida y el buque puede dejar el puerto, o cuando se agota el tanque (volumen mínimo) y es necesario esperar hasta que el tanque alcance un volumen (algo mayor que el mínimo) que permita continuar con la operación del llenado del buque petrolero.

Hay buques con 5 capacidades: 200, 300, 500, 700 y 900 miles de barriles. Un buque toma petróleo de sólo una clase. por lo tanto, hay 25 tipos diferentes de buques de acuerdo a su capacidad y clase de petróleo. Algunos de estos tipos pueden ser vacíos. El tiempo entre llegadas sucesivas es diferente para cada tipo y puede ser aproximado por una distribución triangular cuyos parámetros (mínimo, modo y máximo) depende del tipo de buque.

Si el buque tiene que esperar mas de 42 horas desde que llega al puerto hasta que sale, debe pagar una multa al propietario del buque de acuerdo al tiempo excedido. En el modelo se puede experimentar cambiando la capacidad de los tanques y las tazas de bombeo (ambos costosos), tratando de evitar, largos periodos en donde el flujo de llenado está detenido (que pueden resultar en el cierre costoso de los pozos de petróleo) y tratando de evitar largas esperas de los buques (que pueden resultar en multas) [16].

Este modelo fue propuesto como un ejemplo de modelos mixtos en GLIDER [16], es decir, que usan eventos discretos y simulación continua al mismo tiempo.

5.1 Análisis del Dominio

El modelo de puertos de buques petroleros es un modelo bastante particular, ya que combina dentro del mismo modelo dos técnicas de modelado, el formalismo de simulación por eventos discretos y la simulación de sistemas continuos, representando de esta manera un modelo mixto desde el punto de vista del lenguaje de simulación GLIDER. En este contexto, se realizó un análisis a la implementación de dicho modelo realizada en GLIDER (Anexo D) con el propósito de identificar la red de nodos de simulación de este modelo.

La red de nodos usados para modelar el sistema está formada como se describe a continuación: en primer lugar posee veinticinco nodos tipo entrada los cuales se encargan de generar los diferentes tipos de buques (todas las posibles combinaciones entre capacidades y tipos de petróleo) que llegan al sistema. Luego tiene cinco nodos tipo puerta, los cuales evalúan si un buque puede o no pasar al atracadero correspondiente. Seguidamente posee cinco nodos tipo recurso que modelan los atracaderos, en los cuales es servido el petróleo. La red de simulación posee un nodo tipo salida, el cual es el encargado de modelar las salidas de los buques del sistema. Adicionalmente posee dos nodos tipo continuo en los cuales se realizan los cálculos correspondientes al llenado o vaciado de los tanques de almacenamiento, así cómo también, los cálculos referidos al llenado de los buques.

En la siguiente figura se muestra un diagrama con la nomenclatura GLIDER de la red de nodos del modelo.

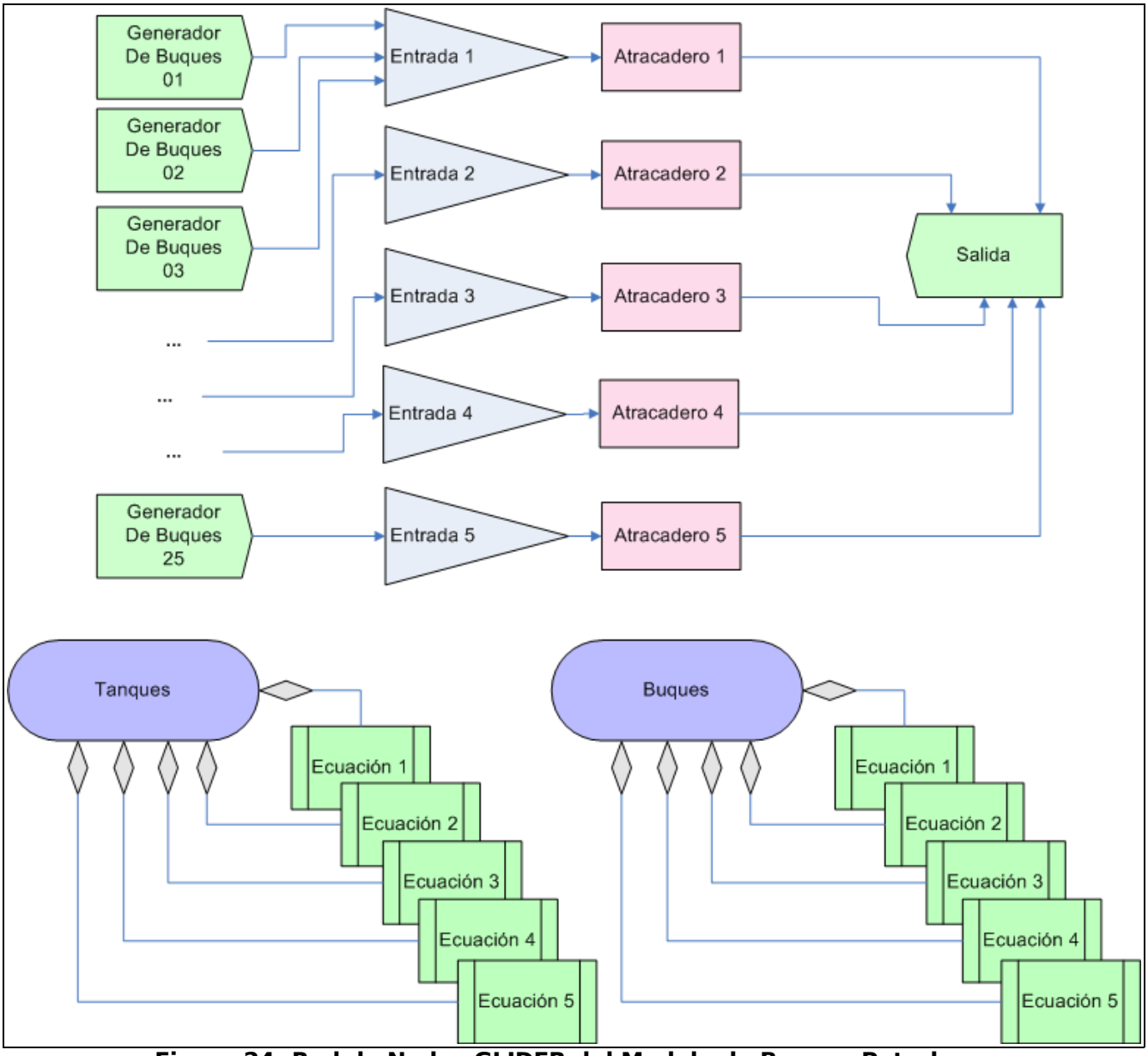


Figura 24: Red de Nodos GLIDER del Modelo de Buques Petroleros

5.2 Diseño del Dominio

La estructura de nodos GLIDER del modelo proporciona un punto de inicio para determinar el diseño del dominio, ya que a partir de él se definieron los aspectos del modelo que pueden derivar en estructuras genéricas y de esta manera lograr definir el meta – modelo. La arquitectura generalizada definida para este modelo se muestra en la siguiente figura.

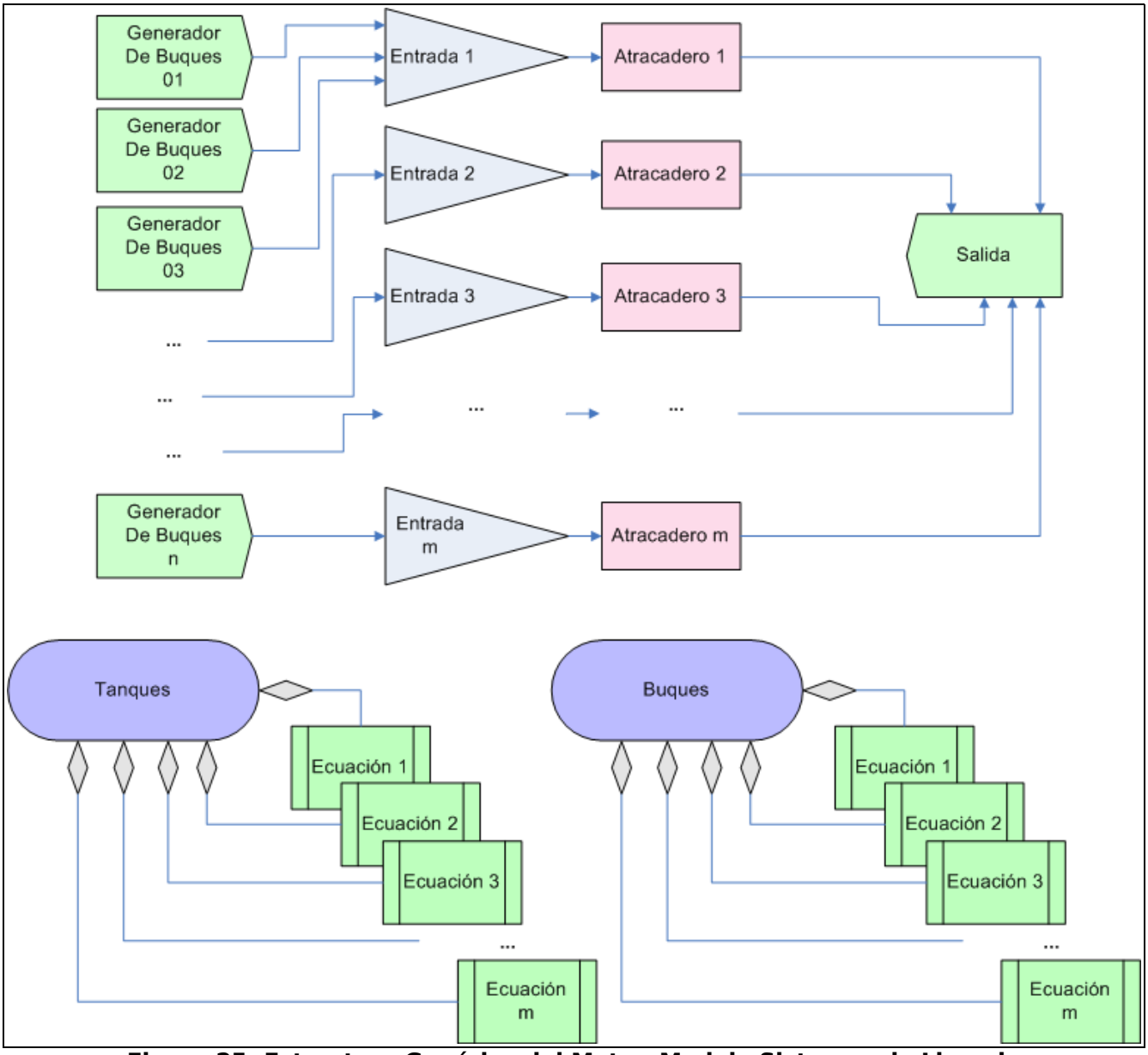


Figura 25: Estructura Genérica del Meta - Modelo Sistemas de Llenado

Del diagrama puede observarse que uno de los aspectos configurables de este meta – modelo es la cantidad de tipos de petróleo que sirven lo que incide directamente en la cantidad de atracaderos que posee el centro de llenado. Así

mismo, la cantidad de diferentes capacidades de los buques es algo que puede ser configurado, lo que implica la variación en la cantidad de tipos de buques (todas las posibles combinaciones entre capacidades y tipos de petróleo) que intervienen en el sistema. Adicionalmente, todos los parámetros del modelo, como por ejemplo, la media entre llegadas de buques, las ratas de llenado y de vaciado de los tanques y los buques son parámetros configurables.

Esta fase proporciona el entorno de trabajo para lograr la implementación del meta - modelo de transporte.

5.3 Arquitectura de Software

Al igual que en los meta – modelos descritos en los capítulos anteriores, el componente principal es la biblioteca de simulación GALATEA, la cual proporciona la plataforma de simulación para modelos mixtos.

Otro de los componentes de software identificados en este meta – modelo es el relacionado con las estructuras de datos necesarias para almacenar las configuraciones realizadas por los usuarios a la hora de instanciar el meta – modelo y las estructuras de datos para guardar los resultados de la simulación del modelo.

Adicionalmente, el tipo de aprovechamiento definido para este modelo es la Animación, lo que conlleva a la necesidad de integrar una biblioteca que permita implementar esta funcionalidad o crear un componente a partir de las clases para el manejo de imágenes que proporciona la biblioteca estándar de Java.

Por último, es necesario definir la manera de cómo la interfaz gráfica va a interactuar con el meta – modelo, como se va a realizar el proceso de instanciación y de retorno de resultados, para luego ser mostrados. Para esto se definen un nivel de abstracción del meta – modelo que proporciona la dinámica de acceso para la interfaz gráfica.

A continuación se muestra un diagrama con los componentes identificados en la arquitectura de software.

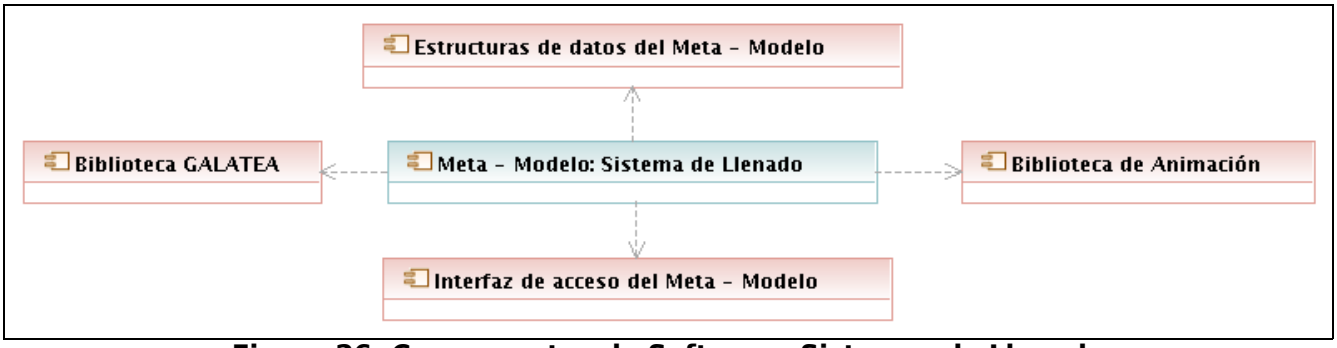


Figura 26: Componentes de Software: Sistemas de Llenado

5.4 Implementación del Dominio

El primer paso fue diseñar e implementar las estructuras de datos necesarias para interactuar con la interfaz gráfica, las cuales permiten pasar los parámetros y configuraciones desde el nivel de interfaz de la aplicación al nivel de meta – modelo. A continuación se muestran los diagramas de dichas estructuras.

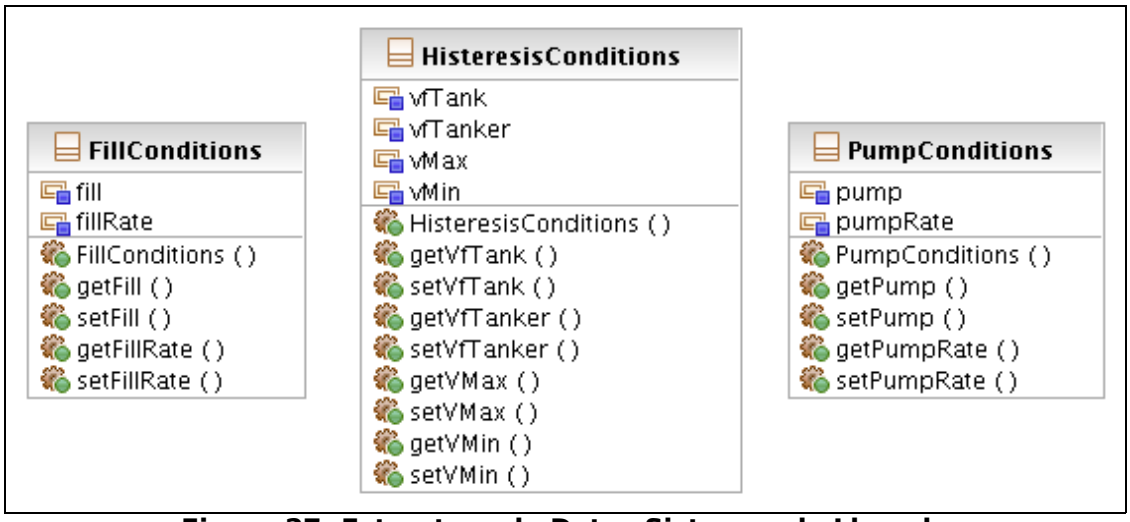


Figura 27: Estructura de Datos Sistemas de Llenado

Una vez definidas las estructuras de datos necesarias para capturar la configuración del usuario se procedió a realizar la implementación de las clases que describen el meta – modelo. En primer lugar se definieron las clases correspondientes a la parte discreta del modelo, de manera de describir los eventos de llegadas y salidas de los contenedores del sistema. En este sentido, se implementaron varias clases las cuales son descritas a continuación.

Se definió la clase ContainerArrivalNode, la cual hereda de la clase Node de GALATEA e implementa un nodo tipo entrada. Dicha clase representa el generador de buques del sistema. El número de instancias creadas de esta clase en el modelo instanciado dependerá de la configuración de tipos de fluidos y capacidades de los contenedores.

Luego se procedió a crear la clase EntranceNode, la cual hereda de la clase Node de GALATEA e implementa un nodo tipo puerta. En esta clase está implementado el funcionamiento que verifica la disponibilidad de la estación de llenado cuando llega un contenedor, según el tipo de líquido correspondiente.

Seguidamente se implementó la clase FillingStationNode, la cual hereda de la clase Node de GALATEA y describe un nodo tipo recurso. En este nodo permanecen los contenedores durante el proceso de llenado.

La última clase definida para la parte de eventos discretos del sistema fue la clase ExitNode, la cual hereda de la clase Node de GALATEA y representa un nodo tipo salida. Esta clase permite modelar las salidas de los contenedores del sistema.

Una vez definidas las clases correspondientes a la fracción discreta del modelo se procedió a determinar las clases necesarias para la sección continua del modelo. En función de esto se implementaron las clases ContainerCont y TankCont las cuales heredan de la clase Cont de GALATEA. Estas clases representan las ecuaciones diferenciales ordinarias que modelan el proceso de bombeado de los contenedores y el proceso de llenado de los tanques respectivamente. La cantidad de objetos de cada una de estas ecuaciones en el modelo instanciado viene dado por la cantidad de tipos de fluido que pueden ser servidos por el sistema.

A continuación se definieron las clases ContainerNode y TankNode, las cuales heredan de la clase Node de GALATEA y representan nodos tipo continuo. A estas clases se agregan la cantidad de objetos de tipo ContainerCont y TankCont respectivamente, dependiendo, como ya se dijo, de la cantidad de tipos de fluidos presentes en la definición del sistema.

Adicionalmente se implementaron las clases VolumenTotalServedCont y VolumenTotalServedNode, las cuales modelan el cálculo del volumen total servido por cada estación de llenado. La clase VolumenTotalServedCount la cual hereda de la clase Count de GALATEA y representa la ecuación diferencial ordinaria para el cálculo del volumen. La clase VolumenTotalServerNode hereda de la clase Node de GALATEA y representa un nodo tipo continuo, a la cual se agregan tantos objetos tipo VolumenTotalServedCont como estaciones de llenado se hayan definido.

Por último, se define la clase FillingSystem en la cual se programa la creación de la red de nodos de simulación específica para el modelo, se establece el tiempo de simulación, las salidas de traza y estadísticas de GALATEA y se realiza el proceso de simulación.

A continuación se muestra el diagrama de clases de la implementación del meta – modelo de Sistema de Llenado.

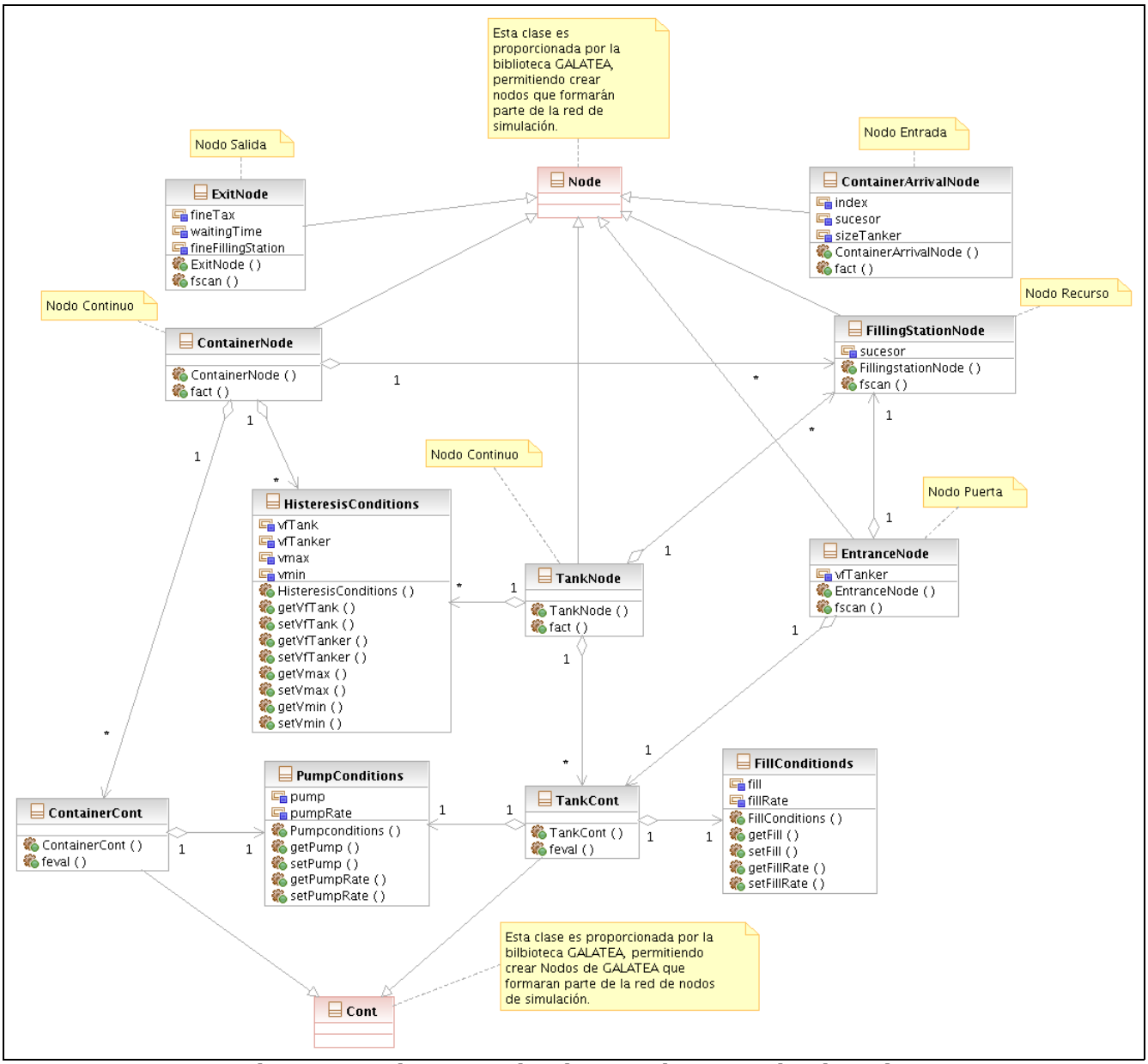


Figura 28: Diagrama de Clases: Sistemas de Llenado

Capítulo VI

Ejemplos

En este capítulo se pretende explicar con algunos ejemplos cómo se realiza el proceso de instanciar un meta – modelo en un modelo particular con la intención de mostrar, no sólo como se representan los modelos particulares dentro de la definición del meta – modelo, sino también de dar una pequeña guía de uso de la aplicación.

6.1 Ejemplo 1: Sistemas Dinámicos

A continuación se describirá el proceso a seguir para usar la plantilla del meta – modelo de sistemas dinámicos para instanciarlo en un modelo particular. En este caso vamos a usar las ecuaciones del atractor de Lorenz.

- Se hace click derecho en “Modelos” y se selecciona “Nuevo Sistema Dinámico”

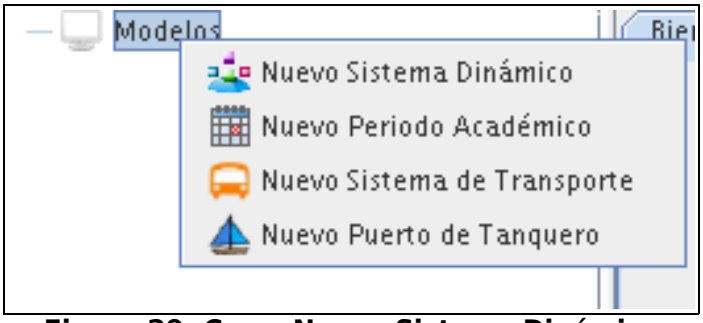


Figura 29: Crear Nuevo Sistema Dinámico

- Se coloca el nombre del modelo y una descripción del mismo en la sección de propiedades.

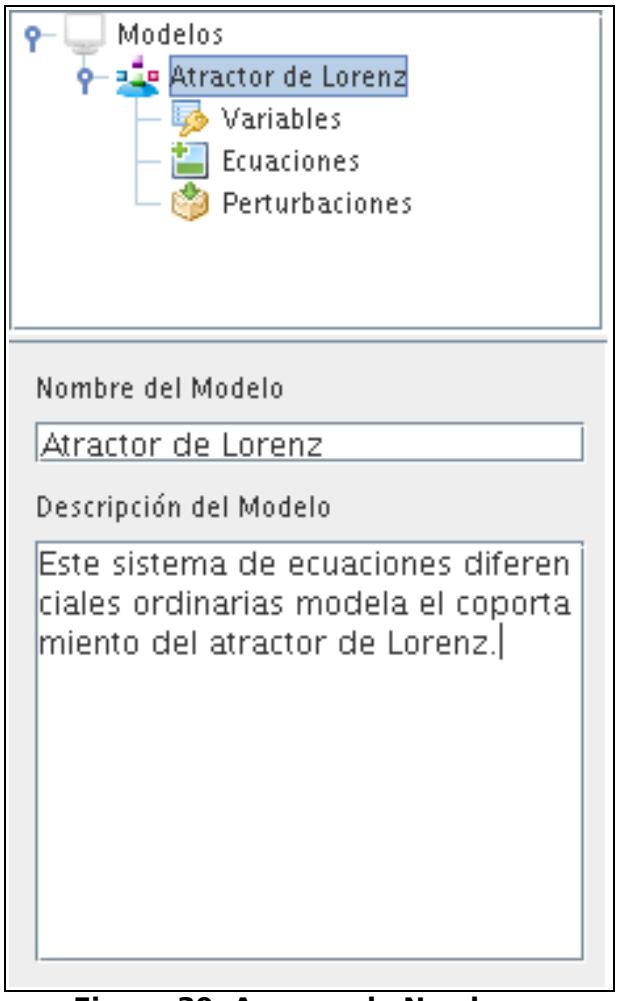


Figura 30: Agregando Nombre y Descripción del Modelo

- Luego se agregan las variables que intervienen en el sistema. Para esto se hace click derecho sobre el item “Variables” y se selecciona “Nueva”.

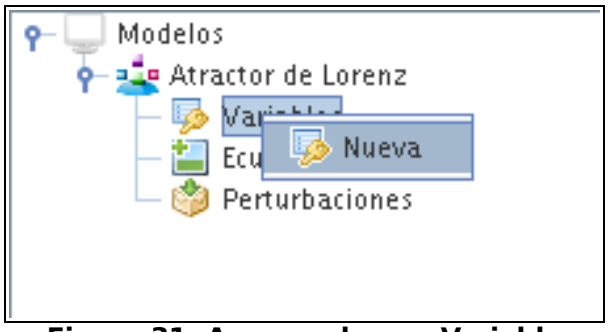


Figura 31: Agregando una Variable

- En la sección de propiedades se define el nombre de la variable y una descripción.

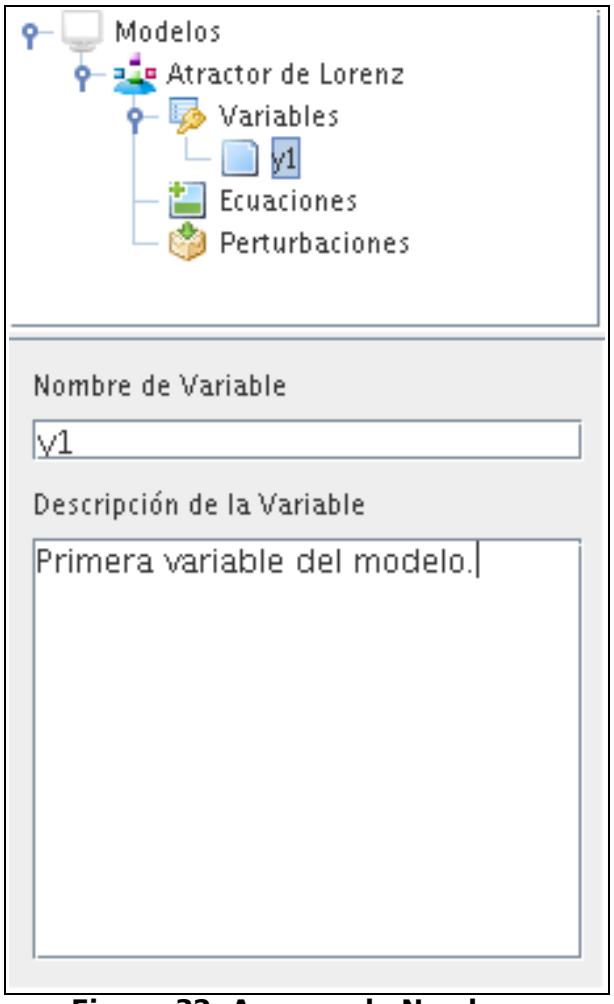


Figura 32: Agregando Nombre y Descripción de Variable

- De esta misma manera se agregan todas las variables del sistema de ecuaciones diferenciales ordinarias.
- Una vez definidas las variables, se pasa a definir las ecuaciones de dichas variables. Para ellos se hace click derecho en el item “Ecuaciones” y se selecciona “Nueva”.
- En la sección de propiedades se define a qué variable pertenece la ecuación y se proporciona la ecuación y las condiciones iniciales.

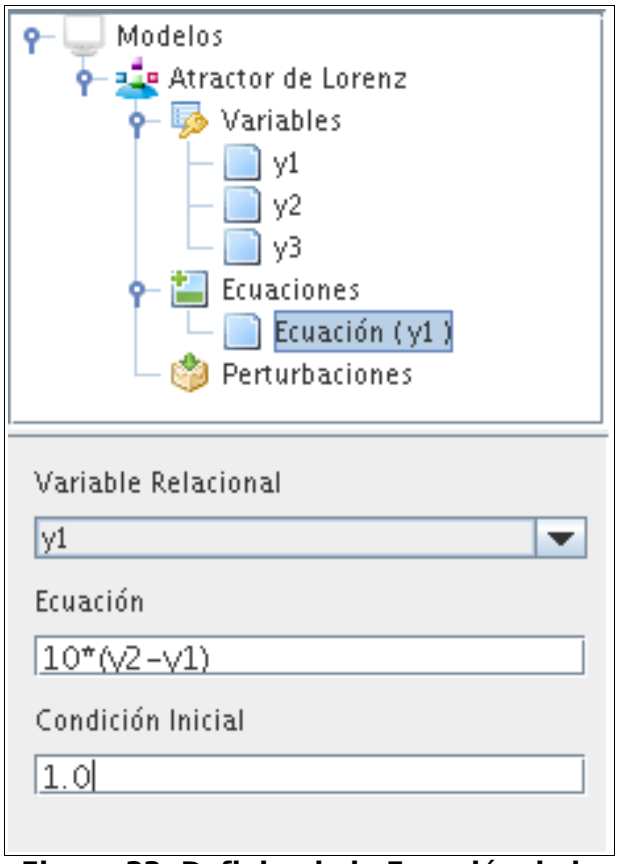


Figura 33: Definiendo la Ecuación de la Variable y1

- De esta misma manera se agregan todas las ecuaciones del sistema.
- En este caso no existen perturbaciones, pero si existieran el proceso es análogo, hacer click derecho en el item “Perturbaciones” y seleccionar “Nueva”. Luego en la sección de propiedades se define a que variable del sistema afecta dicha perturbación, cual es la ecuación que define la perturbación. También se proporcionan los tiempos de inicio y final de la perturbación y un intervalo de ocurrencia.

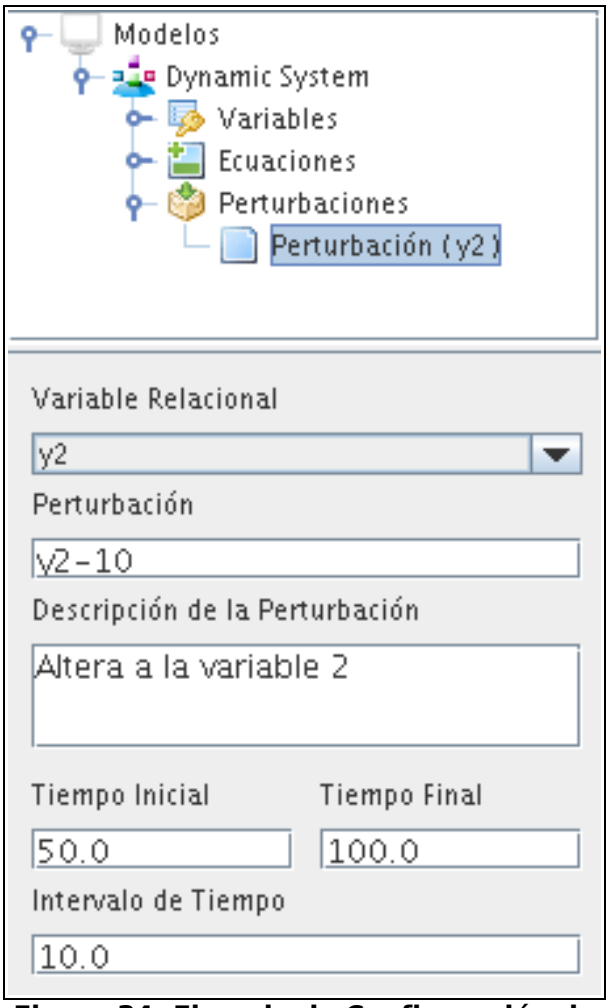
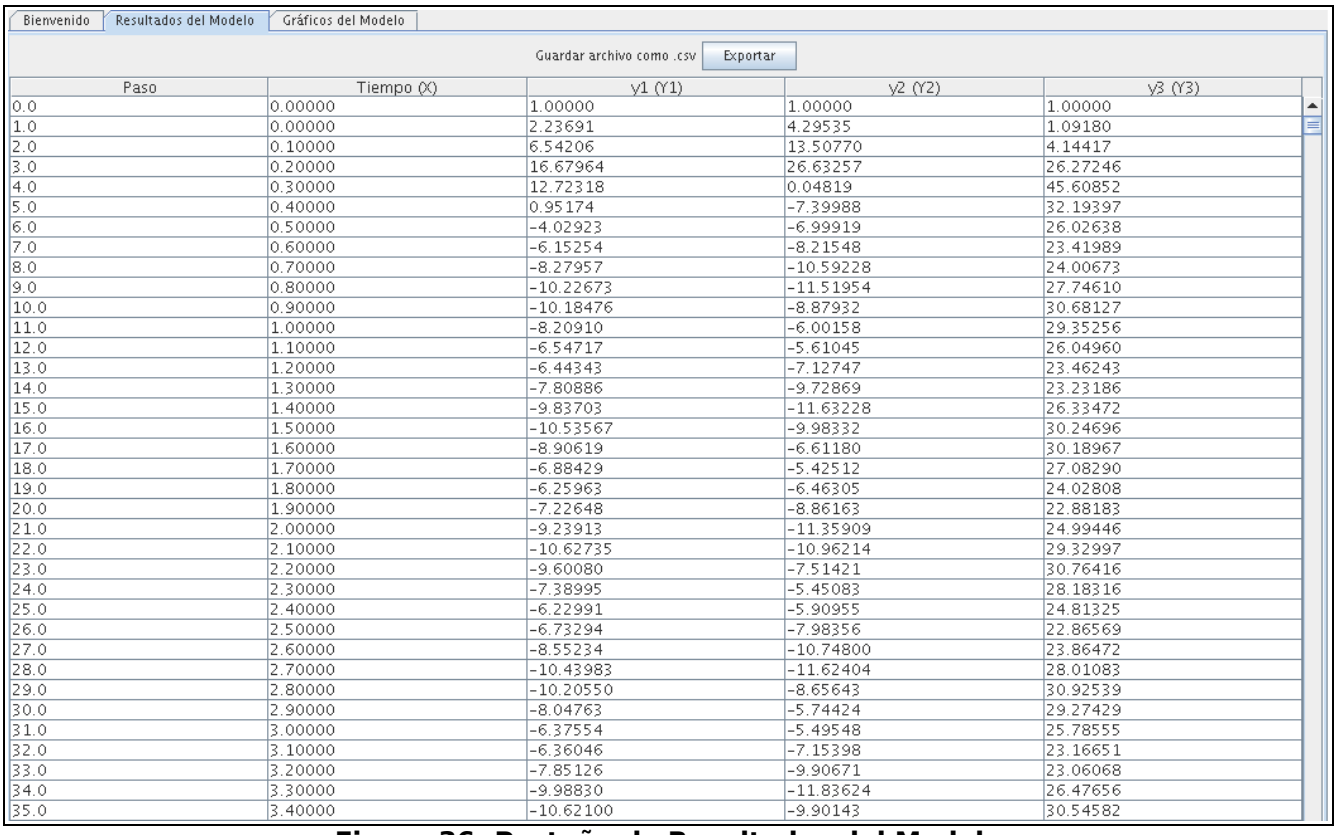


Figura 34: Ejemplo de Configuración de una Perturbación

- Luego de definido el sistema dinámico se procede a realizar la simulación, para ello se hace click derecho en el item con el nombre de nuestro modelo, en este caso “Atractor de Lorenz” y se selecciona “Iniciar Simulación”. Este evento nos muestra una ventana de dialogo donde se solicitan ciertos datos para la simulación, como el tiempo de simulación (por omisión 100), así como también los nombres de los archivos de traza y estadística de GALATEA, los cuales tienen valores por omisión relacionados con el nombre del modelo.

Figura 35: Configurando los Parámetros de Simulación

- Una vez ejecutada la simulación los resultados son mostrados en la sección de resultados (parte derecha de la aplicación). En dicha sección se muestran dos pestañas. En la pestaña de “Resultados del Modelo” se muestra una tabla con los valores de las variables durante la simulación. En la pestaña “Gráficos del Modelo” se muestran los gráficos de las variables.



Paso	Tiempo (X)	y1 (Y1)	y2 (Y2)	y3 (Y3)
0.0	0.00000	1.00000	1.00000	1.00000
1.0	0.00000	2.23691	4.29535	1.09180
2.0	0.10000	6.54206	13.50770	4.14417
3.0	0.20000	16.67964	26.63257	26.27246
4.0	0.30000	12.72318	0.04819	45.60852
5.0	0.40000	0.95174	-7.39988	32.19397
6.0	0.50000	-4.02923	-6.99919	26.02638
7.0	0.60000	-6.15254	-8.21548	23.41989
8.0	0.70000	-8.27957	-10.59228	24.00673
9.0	0.80000	-10.22673	-11.51954	27.74610
10.0	0.90000	-10.18476	-8.87932	30.68127
11.0	1.00000	-8.20910	-6.00158	29.35256
12.0	1.10000	-6.54717	-5.61045	26.04960
13.0	1.20000	-6.44343	-7.12747	23.46243
14.0	1.30000	-7.80886	-9.72869	23.23186
15.0	1.40000	-9.83703	-11.63228	26.33472
16.0	1.50000	-10.53567	-9.98332	30.24696
17.0	1.60000	-8.90619	-6.61180	30.18967
18.0	1.70000	-6.88429	-5.42512	27.08290
19.0	1.80000	-6.25963	-6.46305	24.02808
20.0	1.90000	-7.22648	-8.86163	22.88183
21.0	2.00000	-9.23913	-11.35909	24.99446
22.0	2.10000	-10.62735	-10.96214	29.32997
23.0	2.20000	-9.60080	-7.51421	30.76416
24.0	2.30000	-7.38995	-5.45083	28.18316
25.0	2.40000	-6.22991	-5.90955	24.81325
26.0	2.50000	-6.73294	-7.98356	22.86569
27.0	2.60000	-8.55234	-10.74800	23.86472
28.0	2.70000	-10.43983	-11.62404	28.01083
29.0	2.80000	-10.20550	-8.65643	30.92539
30.0	2.90000	-8.04763	-5.74424	29.27429
31.0	3.00000	-6.37554	-5.49548	25.78555
32.0	3.10000	-6.36046	-7.15398	23.16651
33.0	3.20000	-7.85126	-9.90671	23.06068
34.0	3.30000	-9.98830	-11.83624	26.47656
35.0	3.40000	-10.62100	-9.90143	30.54582

Figura 36: Pestaña de Resultados del Modelo

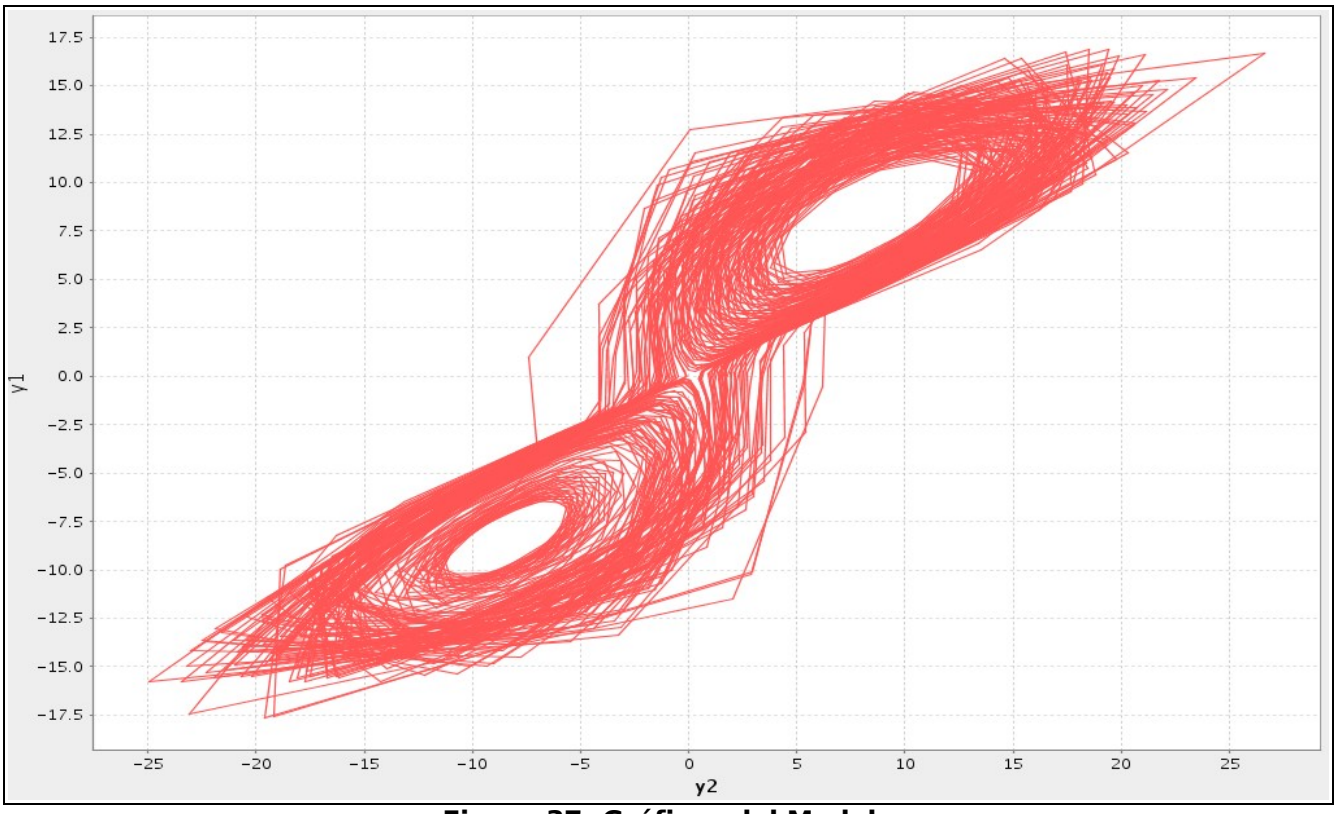


Figura 37: Gráficos del Modelo

Conclusiones y Recomendaciones

En este documento se ha presentado el diseño e implementación del prototipo funcional de la MODELOTECA de GALATEA. Con esta herramienta se pone al alcance de modelistas y simulistas sin mucha experiencia computacional la posibilidad de diseñar y ejecutar modelos de simulación basados en los cuatro meta – modelos o plantillas que conforman este prototipo, los cuales son:

- Sistemas Dinámicos
- Sistemas Matriculares
- Sistemas de Transporte
- Sistemas de Llenado

También se ha presentado, como parte de este trabajo los procedimientos y tareas llevadas a cabo para plantear cada uno de los meta – modelos, siguiendo una metodología basada en los conceptos de Ingeniería de Dominio. Estas plantillas proporcionan los objetos genéricos en cada meta – modelo, permitiendo, en base a la configuración realizada por el usuario en la interfaz gráfica, instanciar los objetos específicos o particulares de un modelo particular y generar la red de simulación en tiempo de ejecución.

Realizar el proceso de generalización, que consiste en tomar un modelo particular y convertirlo en una plantilla o meta – modelo, supone analizar y comprender este modelo en estudio dentro del dominio de modelado y simulación proporcionado por la biblioteca GALATEA. Esto implicó, en primer lugar, realizar la codificación en Java/GALATEA de los modelos específicos usados como base para las plantillas. El modelo de sistemas matriculares, fue una excepción de este requerimiento, debido a que su estructura desde el enfoque de la plataforma de simulación GALATEA no se modificaba, la plantilla de este modelo se encuentra plasmada en la generalización del modelo matemático planteado por Linares [22].

La metodología de desarrollo elegida para realizar este ejercicio de diseño permitió seguir un procedimiento formal y estandarizado para analizar cada uno de los modelos, debido a que proporciona un lineamiento en el proceso de conceptualización de los meta – modelos.

A pesar de las diferencias inherentes a cada meta – modelo, se logró implementar una interfaz gráfica de usuario con un diseño genérico, la cual permite realizar la configuración e instanciación de los meta – modelos en modelos específicos de una forma rápida y sencilla, manteniendo la flexibilidad que requieren las particularidades propias de cada meta – modelo, tanto en su configuración como en su tipo de aprovechamiento.

También se proporciona un mecanismo para guardar los modelos configurados en archivos con formato XML, lo que garantiza gran versatilidad a la hora de usar la herramienta. Así mismo, es posible que el usuario configure un modelo directamente en un archivo XML (siguiendo las directivas establecidas para cada uno de los meta – modelos), el cual puede ser interpretado por la aplicación.

Para instanciar un meta – modelo a un modelo particular, se requiere que el usuario final de la aplicación comprenda en buena medida la estructura y funcionalidades que se han definido para cada una de las plantillas, de tal manera que conociendo esto pueda generar un modelo específico. Al momento de utilizar las plantillas el usuario de la aplicación debe tener en cuenta que existen ciertas limitaciones inherentes a cada una de ellas. El meta – modelo de sistemas dinámicos es el mas amplio ya que es posible generar modelos específicos de cualquier sistema con una estructura isomorfa a un sistema dinámico descrito con ecuaciones diferenciales ordinarias. Los otros tres meta – modelos tienen un uso mas restringido, ya que tanto su dominio de modelado como el área de conocimiento a la que pertenecen limitaron el grado de generalización que fue posible lograr para estos modelos.

Con los modelos usados para plantearse los meta – modelos que conforman parte de esta MODELOTECA, se han evaluado y probado las plantillas correspondientes.

Para completar y extender la MODELOTECA se recomienda principalmente, el desarrollo de más meta – modelos que puedan ser integrados a esta aplicación, así como también es posible mejorar las plantillas ya presentadas. Por ejemplo, en los sistemas dinámicos es posible diseñar e implementar un editor de ecuaciones de tal forma que no sea necesario realizar una traducción de las ecuaciones definidas por el usuario en tiempo de ejecución. Otra mejora posible sería agregarle al modelo de Sistemas de Transporte características y parámetros para realizar análisis económicos.

Referencias

[1]Dávila J., Tucci K. y Uzcategui M. **Simulación Multiagente con GALATEA**. Universidad de los Andes. Mérida. Venezuela (2004).

[2]Uzcátegui M. **Diseño de la Plataforma de Simulación GALATEA**. Tesis de Maestría, Maestría en Computación, Universidad de los Andes. Mérida, Venezuela (2002).

[3]Dávila J. y Tucci K. **Towards a logic-based, multi-agent simulation theory**. Special Issue 2000, Association for the advancement of Modeling & Simulation techniques in Enterprises (2002).

[4]Domingo C., Hernández M., Sananes M., Silva J. y Tonella G. **GLIDER, Lenguaje para Simulación de Sistemas.** Universidad de Los Andes, Mérida, Venezuela (1992).

[5]Lafaille K. **gSpaces. Meta – Modelo para Simular Desalojos en Espacios Urbanos y Arquitectónicos Basados en GALATEA.** Tesis de Maestría, Maestría en Modelado y Simulación de Sistemas, Universidad de los Andes, Mérida, Venezuela (2005).

[6]Hamar V. **Aspectos Metodológicos del Desarrollo y Reutilización de Componentes de Software.** Tesis de Maestría, Maestría en Computación, Universidad de los Andes, Mérida, Venezuela (2003).

[7] Pressman R. **Ingeniería del Software: un enfoque práctico.** McGraw Hill. Quinta edición. (2002).

[8] **Document E09. SourceForge.net: Subversion (Version Control for Source Code).** OSTG **Open Source Technology Group**. Disponible en: http://sourceforge.net/docman/display_doc.php?docid=31070&group_id=1 (2006).

[9] Boulding, k. T**eoría General de Sistemas**. Prentice Hall, México (1970).

[10] Jordán, J. **Los Sistemas y la Organización**. McGraw Hill Series Management (1968).

[11] Checkland, A. **Sistemas Orgánicos**. Prentice Hall, US (1971).

[12] Casti J. **Reality Rules I: Picturing the world in mathematics.** John-

Wiley, New York. (1992).

[13] Casti J. **Reality Rules II: Picturing the world in mathematics.** John-Wiley, New York**.** (1992).

[14] Object Refinery Limited. **jFreeChart.** Disponible en: http://www.jfree.org/jfreechart/

[15] Biblioteca XML parser.

[16] Domingo, C. y Tonella, G. **GLIDER Examples**. Disponible en: http://afrodita.faces.ula.ve/Glider/GLIDERexamples.html (2006).

[17]Odum, H.T. **Trophic Structure and Productivity of Silver** Springs. Ecological Monograph. 27,55-112 (1957).

[18] Wikipedia Foundation Inc. **Sistema Dinámico.** Disponible en: http://es.wikipedia.org/wiki/Sistema_din%C3%A1mico

[19] Wikipedia Foundation Inc. **Ecuación.** Disponible en: http://es.wikipedia.org/wiki/Ecuaci%C3%B3n

[20] Wikipedia Foundation Inc. **Expresión Matemática.** Disponible en: http://es.wikipedia.org/wiki/Expresi%C3%B3n_matem%C3%A1tica

[21] Stephen Kolaroff. **JEPLite – JEP enlited.** Disponible en: http://jeplite.sourceforge.net/

[22] Linares, E. **Modelo Matemático de Simulación Matricular para el Estado Mérida.** Tesis de Maestría, Maestría en Estadística Aplicada y Computación, Universidad de los Andes, Mérida, Venezuela (2003).

Anexo A

```
NETWORK

CHANGES (C)::
      (*Differential equations for biomass*)
      (*Yearly change in radiation*)
      SUN:=MSUN*(1+0.1*SIN(6.2832*TIME));
      (*The efficiency of the photosyntesis depends on the
        biomass of the decomposers with a delay*)
      RETARD(3,PhotEff,0.5,D*0.000582857);
            (*22*0.000582857=0.01282*)
      P':=PhotEff*SUN-RespP*P-ConsPH*P*H-MortP*P-MigrP*P;
      IF P<0 THEN P:=0;
      H':=ConsPH*P*H+EntrH-RespH*H-ConsHC*H*C-MortH*H-MigrH*H*H;
      C':=ConsHC*H*C-RespC*C-ConsCS*C*S-MortC*C+MigrC*C*C;
      S':=ConsCS*C*S-RespS*S-MortS*S-MigrS*S*S;
      D':=MortP*P+MortH*H+MortC*C+MortS*S
          -RespD*(MortP*P+MortH*H+MortC*C+MortS*S)-MortD*D-
           MigrD*D;

EXTRAC  (A):: IT:=1+TRIA(-0.1,0.0,0.1);
              (*Extraction of biomass yearly more or less 10%*)
              P:=P-TRIA(400,800,1000) ;

CONTAM (A)::  D:=D-7.0;   (*Destruction of decomposers*)
              IF TIME <9 THEN IT:=1 ELSE DEACT(CONTAM);
GRAPHIC (A):: IT:=0.046875;   (*graphic period*)
              GRAPH(0,300,BLACK; TIME:6:1,WHITE;
                      P:6:0,PROD,0,4000,GREEN;
                      H:6:0,HERV,0,400,BLUE;
                      C:6:0,CAR1,0,400,MAGENTA;
                      S:6:0,CAR2,0,200,RED;
                      D:6:0,DESC,0,100,BROWN);
INIT TSIM:=300;
     ACT(CHANGES,0);
     ACT(GRAPHIC,0);
     ACT(EXTRAC,2.1);
     ACT(CONTAM,8);
     (*Parameters*)
     PhotEff:=0.01282;(*absortion of solar energy by vegetals *)
     RespP:=3.70117 ; (*Respiration of Producers*)
     ConsPH:=0.00627; (*Consumption of Producers by Herbivores*)
     MortP:=0.976823; (*Mortality of Producers *)
     MigrP:=0.71199 ; (*Migration of Producers *)
     RespH:=11.3855 ; (*Respiration of Herbivores *)
     ConsHC:=0.04614; (*Consumption of herbivores by carnivores*)
     MortH:=8.63253 ; (*Mortalidad Herbivores *)
     MigrH:=0.00537 ; (*Migration of Herbivores *)
```

```
    RespC:=6.3200;   (*Respiration of primary Carnivores *)
    ConsCS:=0.0600;  (*Consumption or primary carnivores by
                       Secondary*)
    MortC:=0.7800 ;  (*Mortality of primary Carnivores*)
    MigrC:=0.00280;  (*Migration of primary Carnivores *)
    RespS:=1.85714;  (*Respiration of Secondary carnivores *)
    MortS:=1.00000;  (*Mortality of Secondary carnivores *)
    MigrS:=0.14286;  (*Migration of Secondary carnivores*)
    RespD:=0.99000;  (*Respiration of Decomposers *)
    MortD:=0.29090;  (*Mortality of  Decomposers *)
    MigrD:=1.81820;  (*Migration of Decomposers*)
    MSUN:=1700000;   (*Mean solar energy Kcal/m2/a  *)
    EntrH:=486   ;   (*Entry of exogenous energy to Herbivores*)

    (*Initial values of variables*)

    P:=3236     ;      (*biomass (energy) of Primary
                         Producers*)
    H:=166      ;      (*biomass (energy) of Hervibores*)
    C:=50       ;      (*biomass (energy) of Carnivores*)
    S:=7        ;      (*biomass (energy) of Secondary*)
    D:=22       ;      (*biomass (energy) of Decomposers*)
    DT_CHANGES:=0.0078125 ;

    DECL VAR  RespP,ConsPH,MortP,MigrP,
              RespH,ConsHC,MortH,MigrH,
              RespC,ConsCS,MortC,MigrC,
              RespS,       MortS,MigrS,
              RespD,       MortD,MigrD,
              SUN,MSUN, EntrH:REAL;
              P,H,C,S,D: CONT; PhotEff: RET:3;



END.
```

Anexo B

```
NETWORK

MATRICULAR (A)::IT=1;

    l1 := TRUNC(15395*EXP(0.012754*TIME));
    C1F := TRUNC(l1*(1+(R1+T1+KD1)));
    C2F := TRUNC(C1F*(1+(R2+T2+KD2)));
    C3F := TRUNC(C2F*(1+(R3+T3+KD3)));
    C4F := TRUNC(C3F*(1+(R4+T4+KD4)));
    C5F := TRUNC(C4F*(1+(R5+T5+KD5)));
    C6F := TRUNC(C5F*(1+(R6+T6+KD6)));
    C7F := TRUNC(C6F*(1+(R7+T7+KD7)));
    C8F := TRUNC(C7F*(1+(R8+T8+KD8)));
    C9F := TRUNC(C8F*(1+(R9+T9+KD9)));

    S1 := TRUNC(C1F/AL);
    S2 := TRUNC(C2F/AL);
    S3 := TRUNC(C3F/AL);
    S4 := TRUNC(C4F/AL);
    S5 := TRUNC(C5F/AL);
    S6 := TRUNC(C6F/AL);
    S7 := TRUNC(C7F/AL);
    S8 := TRUNC(C8F/AL);
    S9 := TRUNC(C9F/AL);

    ALS1 := TRUNC(C1F/S1);
    ALS2 := TRUNC(C2F/S2);
    ALS3 := TRUNC(C3F/S3);
    ALS4 := TRUNC(C4F/S4);
    ALS5 := TRUNC(C5F/S5);
    ALS6 := TRUNC(C6F/S6);
    ALS7 := TRUNC(C7F/S7);
    ALS8 := TRUNC(C8F/S8);
    ALS9 := TRUNC(C9F/S9);

    IPRO1 := C1F/(l1*(1+(R1+T1)));
    IPRO2 := C2F/(C1F*(1+(R2+T2)));
    IPRO3 := C3F/(C2F*(1+(R3+T3)));
    IPRO4 := C4F/(C3F*(1+(R4+T4)));
    IPRO5 := C5F/(C4F*(1+(R5+T5)));
    IPRO6 := C6F/(C5F*(1+(R6+T6)));
    IPRO7 := C7F/(C6F*(1+(R7+T7)));
    IPRO8 := C8F/(C7F*(1+(R8+T8)));
    IPRO9 := C9F/(C8F*(1+(R9+T9)));

    IPER1 := C2F/C1F;
    IPER2 := C3F/C2F;
```

```
    IPER3 := C4F/C3F;
    IPER4 := C5F/C4F;
    IPER5 := C6F/C5F;
    IPER6 := C7F/C6F;
    IPER7 := C8F/C7F;
    IPER8 := C9F/C8F;

    IVMA1 := C1F*(1+(R2+T2))/(l1*(1+(R1+T1)));
    IVMA2 := C2F*(1+(R3+T3))/(C1F*(1+(R2+T2)));
    IVMA3 := C3F*(1+(R4+T4))/(C2F*(1+(R3+T3)));
    IVMA4 := C4F*(1+(R5+T5))/(C3F*(1+(R4+T4)));
    IVMA5 := C5F*(1+(R6+T6))/(C4F*(1+(R5+T5)));
    IVMA6 := C6F*(1+(R7+T7))/(C5F*(1+(R6+T6)));
    IVMA7 := C7F*(1+(R8+T8))/(C6F*(1+(R7+T7)));
    IVMA8 := C8F*(1+(R9+T9))/(C7F*(1+(R8+T8)));

INIT
    ACT(MATRICULAR,0);
    TSIM := 3;
    AL := 30;
    T1 := 0.01;
    T2 := 0.01;
    T3 := 0.01;
    T4 := 0.01;
    T5 := 0.01;
    T6 := 0.01;
    T7 := 0.01;
    T8 := 0.01;
    T9 := 0.01;
    R1 := 0.01;
    R2 := 0.04;
    R3 := 0.02;
    R4 := 0.02;
    R5 := 0.01;
    R6 := 0.01;
    R7 := 0.09;
    R8 := 0.05;
    R9 := 0.07;
    KD1 := 0.02;
    KD2 := 0.02;
    KD3 := 0.03;
    KD4 := 0.15;
    KD5 := 0.12;
    KD6 := 0.11;
    KD7 := 0.09;
    KD8 := 0.32;
    KD9 := 0.27;

DECL VAR

     IPRO1, IPRO2, IPRO3, IPRO4, IPRO5, IPRO6, IPRO7, IPRO8, IPRO9, IPER1, IPER2,
IPER3, IPER4, IPER5, IPER6, IPER7, IPER8, IVMA1,    IVMA2, IVMA3, IVMA4, IVMA5,
IVMA6, IVMA7, IVMA8, T1, T2, T3, T4, T5, T6, T7, T8, T9, R1, R2, R3, R4, R5, R6, R7,
R8, R9, KD1, KD2, KD3, KD4, KD5, KD6, KD7, KD8, KD9: REAL;

    AL, l1, S1, S2, S3, S4, S5, S6, S7, S8, S9, C1F, C2F, C3F, C4F, C5F, C6F, C7F,
```

```
C8F, C9F: INTEGER;

END.
```

Anexo C

```
NETWORK

BusGen (I) Stop[INO]::     (*Bus generation*)
           NumPas:=0;      (*Bus starts void*)
           Bus:=TRUE;      (*it is a bus*)
           BN:=BN+1;
           BusNum:=BN;     (*Bus number*)
           SENDTO(FIRST,Stop[1]);

PassArriv (I) [1..6]::
       IT:=EXPO(TiBeArr[INO]);    (*Passengers arrivals*)
       Dest:=OriDesFun(INO);      (*Assign destination*)
       Bus:=FALSE;                (*Passenger is not a bus*)
{IF P=1 THEN WRITELN('New Arrival of Person to Queue ',INO);}
       SENDTO(Stop[INO]);



Stop  (G) [1..6] Travel[INO],Exit[INO] ::
        IF Bus THEN
         BEGIN (*Process a bus (Bus=TRUE) or a passenger
                (Bus=FALSE) *)

         IF P=1 THEN
            WRITELN('BUS N ',BusNum,' AT STOP ',INO,
 ' CARRYING ',NumPas,'      ',LL(EL_Stop[INO])-1,'  IN QUEUE');

          Off:=0;
          DISASSEMBLE
          (*Take off the assembled with this destination*)
           IF O_Dest=INO THEN
            BEGIN

             Off:=Off+1;        (*Count how many go out*)
             NumPas:=NumPas-1;
             (*Discount to those remaining in the bus*)
             SENDTO(Exit[INO])  (*Out of system*)
            END;
          UPDATE(MESS);
          (*To pass new value of NumPas to Bus message *)

          IF P=1 THEN
                 WRITELN(Off,' PASSENGERS OFF');

           (*Number to take the bus*)
           (*Note: In GetIn add 1 to take account of the bus*)
           GetIn:=MINI(BusCap-NumPas+1,LL(EL_Stop[INO]));
```

```
          IF GetIn>1   (*ASSEMBLE passengers to bus*)
          THEN
           BEGIN

             IF P=1 THEN BEGIN
                WRITELN(GetIn-1,' PASSENGERS ENTER THE BUS');
                PAUSE END;

             Enter:=TRUE;
             ASSEMBLE(FIRST,GetIn,Enter)
             BEGIN NumPas:=NumPas+GetIn-1; SENDTO(Travel[INO]);
               IF P=1 THEN BEGIN
                  WRITELN(NumPas,' PASSENGERS NOW IN THE BUS');
                  PAUSE END;
               Enter:=FALSE;
               PasTotNum[BusNum]:=PasTotNum[BusNum]+GetIn-1
             END;
             IF P=1 THEN BEGIN
                WRITELN(LL(EL_Stop[INO]),' REMAINED IN QUEUE');
                PAUSE END;
           END
          ELSE

           BEGIN  (*There are not passengers*)
            SENDTO(Travel[INO]);

            IF P=1 THEN BEGIN
               WRITELN('No passengers loaded'); PAUSE END;

           END
        END
        ELSE STOPSCAN; (*if arrived message is not bus
                         (it is a passenger) remain in queue*)

Travel (R) [1..6] Stop[INO]::
        RELEASE
         BEGIN  (*account travel time*)

TotTravTime[BusNum]:=TotTravTime[BusNum]+TravTime[INO];
          J:=(INO MOD 6)+1;  (*compute next stop number*)
          SENDTO(FIRST,Stop[J]);
         END;
        STAY:=TravTime[INO];  (*time of travel*)

        IF P=1 THEN
           BEGIN WRITELN('BUS N ',BusNum,' starts travel
',INO);
             WRITELN; PAUSE
           END;

Exit (E) [1..6] ::

RESULT (A)::
  OUTG TotTravTime[1..4]:8:2:Total time travel by bus          ;
  OUTG PasTotNum [1..4] :8  :Total number of passengers in bus ;
```

```
  PAUSE;
  ENDSIMUL;

INIT

 ACT(RESULT,600);
 ACT(BusGen,12);
 ACT(BusGen,44);
 ACT(BusGen,70);
 ACT(BusGen,100);
 ACT(PassArriv[1],0);
 ACT(PassArriv[2],10);
 ACT(PassArriv[3],45);
 ACT(PassArriv[4],60);
 ACT(PassArriv[5],67);
 ACT(PassArriv[6],80);

 FOR J:=1 TO 6 DO  Travel[J]:=MAXREAL;

 BusCap:=25;
 BN:=0;
 P:=0;
 ASSI PasTotNum[1..4]:=(0,0,0,0);
 ASSI TotTravTime[1..4]:=(0,0,0,0);
 ASSI TiBeArr[1..6]:=(2.8,3.2,3.0,2.5,3.1,2.2);
 ASSI TravTime [1..6]:=(15,24,17,17,24,15);
 ASSI OriDesMat [1..6,1..6]:=
      (( 0,12,17,15, 0, 0), (*ORIGIN DESTINATION*)
       ( 0, 0,11,15, 0, 0), (*FRECUENCIES*)
       ( 0, 0, 0,15, 0, 0),
       (14, 0, 0, 0, 9,17),
       (15, 0, 0, 0, 0,12),
       (16, 0, 0, 0, 0, 0));

 (*EXPERIMENTAL PARAMETERS*)
 INTI BusCap:5:Maximum capacity of buses;
 INTI TravTime[1..6]:7:2:Time travel since stop number;
 INTI P:Put 1 for passengers detailed movements;

DECL

 VAR GetIn,J,N,BusCap,Off,BN,P:INTEGER;
     OriDesMat:ARRAY[1..6,1..6] OF REAL;
     TiBeArr,TravTime:ARRAY[1..6] OF REAL;
     TotTravTime:ARRAY[1..4] OF REAL;
     PasTotNum:ARRAY[1..4] OF INTEGER;
     Enter:BOOLEAN;

 MESSAGES PassArriv(Bus:BOOLEAN;Dest:INTEGER);
          BusGen(Bus:BOOLEAN; NumPas,BusNum:INTEGER);

 STATISTICS BusGen,PassArriv,Stop,Travel,Exit;

 PROCEDURES

  FUNCTION OriDesFun(I:INTEGER):INTEGER;
```

```
  (*compute probable 8destination according to frequencies in
    OriDesMat for passenger departing from I stop*)
  VAR K:INTEGER; R,P,FTOT:REAL;
  BEGIN FTOT:=0;
    FOR K:=1 TO 6 DO FTOT:=FTOT+OriDesMat[I,K];
    R:=RANDOM; K:=1; P:=OriDesMat[I,K]/FTOT;
    WHILE P<R DO BEGIN K:=K+1; P:=P+OriDesMat[I,K]/FTOT END;
    OriDesFun:=K;
  END;

END.
```

Anexo D

```
NETWORK

 (*The arriving ships may be of one of 25 types, (5 capacities
   and 5 oil types . According to the class of oil the ship is
   sent to the corresponding entrance to the berths *)

ARRIVAL (I) [1..25] ENTRANCE[INO]::
                             TYP:=INO;
                             IT:= TRIA(TBAMIN[INO],TBAMO[INO],
                                  TBAMAX[INO]);
                             BERT:=((TYP-1) MOD 5)+1;
                             SIZE:=(TYP DIV  5) + 1;
                             SENDTO(ENTRANCE[BERT]);
  IF R=1 THEN
         BEGIN WRITELN
              ('T= ',TIME:8:2,
               'Arrival of tanker of type',TYP:3,
               '  Size ',SIZET[SIZE]:7:1,
               '   it goes to berth ',BERT);
       END;

      (*If the berth is free and there are enough oil in its
tank
        the tanker is sent to the bert*)
ENTRANCE (G)[1..5]  BERTH[INO]::
                         IF (F_BERTH[INO]>0) AND
(VTANK[INO]>VFTANKER[INO])
                           THEN
                            BEGIN
                             DOEVENT
                              BEGIN
                               (*size is stored in SIZETANKER*)
                               SIZETANKER[INO]:=SIZET[SIZE];
                               IF R=1 THEN
                                WRITELN('T= ',TIME:8:2,
                                'Ship assigned to berth ',BERT);
                               SENDTO(BERTH[INO]);
                              END;
                            END;

      (*The ship is harbored in the berth*)

BERTH(R) [1..5] EXIT ::
      (*Exit and accumulated fine for the berth *)

EXIT(E) :: TE:=TIME-GT;
           IF TE>1.75 THEN
```

```
              FINE[BERT]:=FINE[BERT]+(TE-1.75)*FINETAX;

TANK (C) var I: integer; ::

(*Computes the total volume served in each berth
  by integration of the volumes in the tankers*)
              VTT'[1]:=VTANKER[1]; VTT'[2]:=VTANKER[2];
              VTT'[3]:=VTANKER[3];
              VTT'[4]:=VTANKER[4]; VTT'[5]:=VTANKER[5];
              (*Shut down conditions:
              FILL[I]=1 if tank I is being filled
              PUMP[I]=1 if there is pumping to the ship in
                        berth I *)
              FOR I:=1 TO 5 DO
               BEGIN
                IF VTANK[I]>VMAX[I] THEN FILL[I]:=0;
                IF (VTANK[I]<VFTANK[I]) AND (FILL[I]=0.0)
                 THEN FILL[I]:=1;
                IF (VTANK[I]<=VMIN[I]) THEN PUMP[I]:=0;
                IF (VTANK[I]>VFTANKER[I]) AND (F_BERTH[I]=0)
                 THEN PUMP[I]:=1;
               END;

  (*Rate of filling of the tanks*)
  VTANK'[1]:=FILL[1]*FILLRATE[1]-PUMP[1]*PUMPRATE[1] ;
  VTANK'[2]:=FILL[2]*FILLRATE[2]-PUMP[2]*PUMPRATE[2];
  VTANK'[3]:=FILL[3]*FILLRATE[3]-PUMP[3]*PUMPRATE[3];
  VTANK'[4]:=FILL[4]*FILLRATE[4]-PUMP[4]*PUMPRATE[4];
  VTANK'[5]:=FILL[5]*FILLRATE[5]-PUMP[5]*PUMPRATE[5];

MODULE (* Division of the NETWORK section *)

PUMPING (C) EXIT var I, SIZE: integer; ::
                  (*Rate of filling of the tankers*)
                    VTANKER'[1]:=PUMPRATE[1]*PUMP[1];
                    VTANKER'[2]:=PUMPRATE[2]*PUMP[2];
                    VTANKER'[3]:=PUMPRATE[3]*PUMP[3];
                    VTANKER'[4]:=PUMPRATE[4]*PUMP[4];
                    VTANKER'[5]:=PUMPRATE[5]*PUMP[5];

                  (*Conditions of filling in each berth*)
  FOR I:=1 TO 5 DO
    IF (VTANKER[I] >= SIZETANKER[I]) AND (F_BERTH[I]=0)
                      THEN
                        BEGIN
                           REL (IL_BERTH[I],TRUE) SENDTO (EXIT);
                           IF R=1 THEN
                            BEGIN
                              WRITELN('T= ',TIME:8:2,
                              'Tanker at berth ',I,' filled');
                              PAUSE
                            END;
                           PUMP[I]:=0; VTANKER[I]:=0;
                        END;

     (*Initial arrivals and assignment of type*)
```

```
PRIMARRIVAL(A) var SIZE, BERT: integer; ::
   FOR SIZE:=1 TO 5 DO
       BEGIN
         FOR BERT:=1 TO 5 DO
             BEGIN
               TYP:=(SIZE-1)* 5 + BERT;
               IF TBAMO[TYP]<>0 THEN
                  BEGIN
                    TPLL:=TRIA(TBAMIN[TYP],TBAMO[TYP],
                               TBAMAX[TYP]);
                    ACT(ARRIVAL[TYP],TPLL);
                    IF R=1 THEN
                    BEGIN
                       WRITELN('T= ',TIME:8:2,
          'Activated arrival Size ',SIZET[SIZE]:7:1,
          ' Berth ',BERT:3,'  Type ',TYP:3,'   Time',TPLL:8:1);
                       PAUSE;
                    END;
                   END;
              END;
         END;
      (*Graph volumes of each tank*)
GRA(A) :: IT:=0.0625;
  IF R=2 THEN
        BEGIN
         GRAPH (0,365,BLACK; TIME:6:1,WHITE;
             VTANK[1]:7:1, TANK1,0,1000, MAGENTA;
             VTANK[2]:7:1, TANK2,0,1000, RED;
             VTANK[3]:7:1, TANK3,0,1000, GREEN;
             VTANK[4]:7:1, TANK4,0,1000, BLUE;
             VTANK[5]:7:1, TANK5,0,1000, YELLOW);
        END;

(*Report of fines and shipments in each berth*)
RESUL (A)::
                  REPORT
                  OUTG FINE[1..5]:10:1:Fine in each berth ;
                  VTT1:=VTT[1];VTT2:=VTT[2];VTT3:=VTT[3];
                  VTT4:=VTT[4];VTT5:=VTT[5];
                  OUTG VTT1:15:1:Oil of class 1 shipped ;
                  OUTG VTT2:15:1:Oil of class 2 shipped ;
                  OUTG VTT3:15:1:Oil of class 3 shipped ;
                  OUTG VTT4:15:1:Oil of class 4 shipped ;
                  OUTG VTT5:15:1:Oil of class 5 shipped ;
                  ENDREPORT;
                  PAUSE;
                  ENDSIMUL;

      INIT  ACT(TANK,0); ACT (GRA,0);
            ACT(PRIMARRIVAL,0); ACT(PUMPING,0);
ACT(RESUL,365.5);
            ASSI FILLRATE[1..5]:=(32,56.5,145,80,150);
            ASSI PUMPRATE[1..5]:=(1200,1000,1000,1200,1000);
            ASSI VMAX[1..5]:= (690, 600, 780, 700, 510);
            ASSI VFTANK[1..5]:=(630, 580, 760, 630, 490);
            ASSI VMIN[1..5]:=(90,60,60,80,60);
```

```
            ASSI VFTANKER[1..5]:=(140,100,100,140,100);
            ASSI FILL[1..5]:=(1.0,1.0,1.0,1.0,1.0);
            ASSI PUMP[1..5]:=(0.0,0.0,0.0,0.0,0.0);

           ASSI TBAMIN[1..25]:=
                           (12.6,  10.17,  7.13, 16.25, 10.17,
                            22.33, 16.25,  5.3,  34.5,  12.6,
                             0.0,  34.5,  16.25, 43.63, 34.5,
                             0.0,   0.0,  28.42,180.5, 43.63,
                             0.0,   0.0,  43.5, 119.67,71.0);

           ASSI TBAMO[1..25]:=
                     (14.6,  12.17,  9.13, 18.25, 12.17,
                      24.33, 18.25,  7.3,  36.5,  14.6,
                       0.0,  36.5,  18.25, 45.63, 36.5,
                       0.0,   0.0  ,30.42,182.5,  45.63,
                       0.0,   0.0,  45.5, 121.67, 73.0);

           ASSI TBAMAX[1..25]:=
                      (16.6,  14.17, 11.13, 20.25, 14.17,
                       26.33, 20.25,  9.3,  38.5,  16.6,
                        0.0,  38.5,  20.25, 47.63, 38.5,
                        0.0,   0.0,  32.42, 184.5, 47.63,
                        0.0,   0.0,  47.5, 123.67, 75.0);

           ASSI SIZET[1..5]:=(200,300,500,700,900);
           ASSI SIZETANKER[1..5]:=(1000,1000,1000,1000,1000);
          (*VALOR GRANDE*)
           EABS:=10000; EREL:=10000;
           DT_TANK:=0.03125;  DT_PUMPING:=0.03125;
           FINETAX:=300;
           ASSI FINE[1..5]:=(0,0,0,0,0);
           VTANK[1]:=250; VTANK[2]:=300; VTANK[3]:=350;
           VTANK[4]:=300;
           VTANK[5]:=466;
           ASSI VTT[1..5]:=(0,0,0,0,0);
           ASSI VTANKER[1..5]:=(0,0,0,0,0);
           R:=0;
           INTI R:2:WRITE 1   GRAF 2  NONE 0 ;

DECL
           VAR
           (*For each berth or class of oil: *)
           FILLRATE,(*filling date of the tank mb/d*)
           PUMPRATE,(*Pumping rate mb/d *)
           VMAX,    (*Maximum volume in the tank mb *)
           VMIN,    (*Minimum volume in the tank mb *)
           VFTANK,  (*Volume to restart the filling of the tank
                      mb *)
           VFTANKER,(*Volume to restart the filling of the
                      tanker mb *)
           FILL,  (*FILL=1 if the tank is filling else FILL=0*)
           PUMP,      (*PUMP=1 if there is pumping to tanker
                             else PUMP=0 *)
           SIZETANKER,(*Capacity of the tanker being filled*)
           SIZET,     (*Capacity to be filled in the tanker mb*)
```

```
           FINE       (*Accumulated fine*)
           :ARRAY[1..5] OF REAL;
           VTANK,     (*Actual volume in the tank mb *)
           VTT,       (*Total volume shipped mb *)
           VTANKER    (*Actual volume in the tanker mb *)
           :ARRAY[1..5] OF CONT;

           (*For each type of tanker: *)
           TBAMIN,    (*Minimum time between arrivals d *)
           TBAMO,     (*Most likely time between arrivals d *)
           TBAMAX     (*Maximum time between arrivals d *)
           :ARRAY[1..25] OF REAL;

           J,I,R:INTEGER; CH:CHAR;
           FINETAX,   (*Fine tax by day $/d *)
           TPLL,      (*Auxiliary variables*)
           TE,
           VG1,VG2,VG3,VG4,VG5,
           VTT1,VTT2,VTT3,VTT4,VTT5:REAL;

           MESSAGES ARRIVAL(TYP,BERT,SIZE:INTEGER);
           STATISTICS ARRIVAL,ENTRANCE,BERTH,EXIT;

END.
```