Universidad de las Ciencias Informáticas
Facultad de Bioinformática

Título: “Software para la Simulación de Sistemas Biológicos: Módulo de modelación gráfica de Sistemas Biológicos”

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es):

Yusdenis Sánchez Perodín.

Yuniesky Armentero Moreno.

Tutor(es):

Dr. Kalet León Monzón.

Lic. Gilberto Arias Naranjo.

Ing. Anthony R. Sotolongo León.

Consultante:

Lic. Eduardo Hermenegildo Caballero.

Ciudad de la Habana, Julio de 2007.
“Año 49 de la Revolución”.

"Si buscas resultados distintos, no hagas siempre lo mismo."
Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los ____ días del mes de _____ del año _____.

Yusdenis Sánchez Perodín	Gilberto Arias Naranjo
_____	_____
Firma del autor	Firma del tutor
Yuniesky Armentero Moreno	Anthony R. Sotolongo León
_____	_____
Firma del autor	Firma del tutor

DATOS DE CONTACTO

Tutores:

Dr. Kalet León Monzón.
Centro de Inmunología Molecular, Habana, Cuba.
Email: kalet@ict.cim.sld.cu

Lic. Gilberto Arias Naranjo.
Universidad de las Ciencias Informáticas, Habana, Cuba.
Email: gilbertoa@uci.cu

Ing. Anthony R. Sotolongo León.
Universidad de las Ciencias Informáticas, Habana, Cuba.
Email: asotolongo@uci.cu

Consultante:

Lic. Eduardo Hermenegildo Caballero.
Universidad de las Ciencias Informáticas, Habana, Cuba.
Email: ecaba@uci.cu

Agradecimientos

A las personas que de una forma u otra han colaborado con nuestra formación como profesionales, al grupo de Bioinformática de la Universidad de las Ciencias Informáticas, a nuestros compañeros de estudio, a nuestros profesores, tutores y familiares.

…gracias.

Dedicatoria

A mis padres y mi hermano que siempre confiaron en mí…
A los que me ofrecieron su amistad y me brindaron su apoyo para que viera realizados mis sueños…
A aquellos que aunque no están presenten les hubiese gustado disfrutar como yo de este inolvidable momento…

Yuniesky Armentero Moreno

A mi hermana…
A mi abuelo Severo, donde quiera que esté…
A mis padres, que apoyaron cada uno de mis pasos y me guiaron siempre por el sendero correcto…
A mi novia y compañeros de estudios, por los momentos vividos a lo largo de la carrera y los que quedan por vivir.

Yusdenis Sánchez Perodín

RESUMEN

La Biología de Sistemas es una de las áreas de las ciencias interdisciplinarias más prometedoras en la actualidad. Una de las tareas más difíciles a la que se enfrentan los investigadores de la Biología de Sistemas es la de comprender el funcionamientos de los Sistemas Biológicos para posteriormente realizar las investigaciones necesarias, pero en la mayoría de las ocasiones el nivel de complejidad de los mismos es elevado por lo que recurren a modelación gráfica para facilitar su comprensión. El uso de software que permitan modelar gráficamente Sistemas Biológicos se ha convertido en una necesidad.

A través del presente trabajo se desarrolló una herramienta computacional que permite modelar gráficamente Sistemas Biológicos que puedan ser descritos mediante dinámica de población.

PALABRAS CLAVE

Sistemas Biológicos

Biología de Sistemas

Modelación

TABLA DE CONTENIDOS

INTRODUCCIÓN

Desde la aparición del ordenador en la década de los 50 su aplicación al estudio de los seres vivos ha revolucionado la Biología. La fusión entre Biología e Informática ha dado como resultado una nueva disciplina - la Bioinformática - en la que la aplicación del ordenador al análisis, modelado y simulación de las estructuras y fenómenos observados en los seres vivos en sus distintos niveles de organización ha conducido a resultados que por su impacto científico, médico y social determinarán en gran medida los avances tecnológicos que sin duda estarán entre los más espectaculares del siglo XXI (1).

En la actualidad la Biología de Sistemas (BS) es una de las áreas de las ciencias interdisciplinarias más prometedoras. Entrelazando campos como la Genómica, Proteómica, Bioinformática y la Biología celular con la Computación y las Matemáticas, se han llegado a descubrir principios fundamentales del funcionamiento de la vida celular antes desconocidos (1).

El evolutivo campo de los Sistemas Biológicos (SB) representa la integración de conceptos e ideas de las ciencias vivas, disciplinas de ingeniería y las ciencias computacionales. Recientes avances de la Biología, incluyendo la secuenciación del genoma humano y masivos acercamientos a probar ejemplos biológicos han creado nuevas oportunidades para entender los problemas biológicos desde la perspectiva de un sistema como un todo. Este nuevo acercamiento hace más énfasis en el estudio del comportamiento de colecciones de componentes funcionando juntos que al estudio de componentes aislados (2).

La modelación y diseño de sistemas son admitidos y fuertemente utilizados en las disciplinas de ingeniería, pero hasta hace poco era relativamente raro aplicarlo en la biología. Para explorar la aplicación de sistemas complejos y analizar problemas biológicos, equipos multidisciplinarios de biólogos, ingenieros y profesionales de la informática están trabajando juntos aplicando principios y técnicas de ingeniería, algoritmos y conceptos computacionales para resolver problemas de la Biología y medicina. De igual forma, a través del trabajo de solución a problemas biológicos, ingenieros y profesionales de la computación están creando nuevos conocimientos en sus propias disciplinas.

La comunidad mundial de científicos se ha volcado con grandes fuerzas hacia la modelación y simulación de SB, campo donde se han obtenido grandes resultados. Con el paso del tiempo han surgido

empresas dedicadas por completo al estudio de SB a través de la BioSimulación, entre ellas se encuentran (3):

- **Entelos**, dedicada a las ciencias de la vida con el objetivo de mejorar la salud humana, utilizando como herramienta fundamental la BioSimulación (www.entelos.com).
- **Genomática**, en la rama de la informática líder en el estudio de SB (www.genomatica.com).
- **Gene Network Sciences**, a través de modelos computacionales de células y organismos a un alto nivel, permite simular el rendimiento clínico de drogas y drogas candidatas (www.gnsbiotech.com).

Una de las tareas más difíciles a la que se enfrentan los investigadores de la BS es a la de comprender el funcionamiento de los SB para a partir de ahí realizar las investigaciones necesarias, pero en la mayoría de las ocasiones el nivel de complejidad de los mismos es elevado por lo que recurren a la modelación gráfica para facilitar su comprensión. El uso de software que permitan modelar gráficamente SB se ha convertido en una necesidad.

Existe un gran número de software en la actualidad que pueden utilizarse como herramientas computacionales en los estudios de los SB. BioUML, Cellware, Dynetica, Stella, son ejemplos de ellos, en el sitio Web www.sbml.com se pueden encontrar algunos más.

Luego de haber realizado un estudio sobre los mismos se detectaron que aún presentan algunas limitaciones en cuanto a las facilidades que pudiesen brindar desde el punto de vista de ser herramientas computacionales que permiten modelar gráficamente SB, razón por la cual surge el presente trabajo de diploma, centrándose en el **problema**: ¿Cómo modelar gráficamente Sistemas Biológicos?.

Tomando como **objeto de estudio** “La modelación gráfica de Sistemas Biológicos”.

Delimitando el **campo de acción** a: La modelación gráfica de Sistemas Biológicos que puedan ser descritos mediante dinámica de población.

El **objetivo general** del presente trabajo es:

Desarrollar una herramienta computacional para la modelación gráfica de Sistemas Biológicos que puedan ser descritos mediante dinámica de población.

Para lograr el mismo se han trazado los siguientes **objetivos específicos**:

- Realizar estudio sobre las características de los software utilizados para la modelación gráfica de Sistemas Biológicos.
- Definir los requisitos para el desarrollo de una herramienta que permita modelar gráficamente Sistemas Biológicos.
- Diseñar una herramienta capaz de modelar gráficamente Sistemas Biológicos.
- Implementar la herramienta diseñada.

Para lograr las metas planteadas se le dará cumplimiento a las siguientes **tareas**:

- Estudio de los principales software de modelación gráfica de Sistemas Biológicos existentes.
- Elaboración del modelo de dominio correspondiente a un Sistema Biológico.
- Definición de los requisitos funcionales y no funcionales de la herramienta.
- Desarrollo del diagrama de casos de uso del sistema.
- Descripción de los casos de uso del sistema.
- Descripción de la arquitectura de la herramienta.
- Descripción de los patrones de diseño utilizados.
- Desarrollo del diagrama de clases del diseño.
- Implementación de la herramienta.
- Realización de pruebas de aceptación a la herramienta implementada.

El presente trabajo de diploma está estructurado en cuatro capítulos y varios anexos. A continuación se da un breve resumen de los capítulos.

Capítulo 1 "Fundamentación teórica": Donde se brinda una descripción general de la Biología de Sistemas y la modelación gráfica de Sistemas Biológicos, así como el análisis de algunos software

existentes que permiten modelar gráficamente Sistemas Biológicos. Finalmente se da una descripción de las herramientas y metodologías a utilizar para desarrollar la solución propuesta.

Capítulo 2 "Características del sistema": Donde se da una breve descripción del sistema ha desarrollar, se describe el modelo de dominio perteneciente a un Sistema Biológico así como la presentación del diagrama de objetos, se describen los requisitos funcionales y no funcionales del sistema, se definen los actores del sistema, el diagrama de caso de uso del sistema y se hace una descripción detallada de los casos de usos del sistema.

Capítulo 3 "Diseño del sistema": Donde se da una descripción del estilo arquitectónico utilizado para el desarrollo de la herramienta, se describirán los patrones de diseño empleados así como los diagramas de clases del diseño.

Capítulo 4 "Implementación y pruebas": Donde se describe como fue implementada la herramienta en términos de componentes. Se desarrollaran varias pruebas de aceptación usando el método de caja negra y la técnica de particiones de equivalencia para garantizar que se han cumplido los requisitos funcionales.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 INTRODUCCIÓN

En el presente capítulo se brinda una descripción general de la Biología de Sistemas y la modelación gráfica de Sistemas Biológicos, así como el análisis de algunos software existentes que permiten modelar gráficamente Sistemas Biológicos.

Finalmente se da una descripción de las herramientas y metodologías a utilizar para desarrollar la solución propuesta a través de presente trabajo de diploma.

1.2 BIOLOGÍA DE SISTEMAS

La BS representa la integración de conceptos e ideas de las ciencias de la vida, disciplinas de ingeniería y ciencias computacionales (Figura 1.1). Recientes avances de la Biología, incluyendo la secuencia del genoma humano y masivos experimentos para probar ejemplos biológicos, han creado nuevas oportunidades para comprender los problemas biológicos desde la perspectiva de un sistema. Este nuevo acercamiento hace mucho más énfasis en el comportamiento de colecciones de componentes funcionando como un todo que a los enfoques tradicionales del estudio de componentes de forma individual (2).

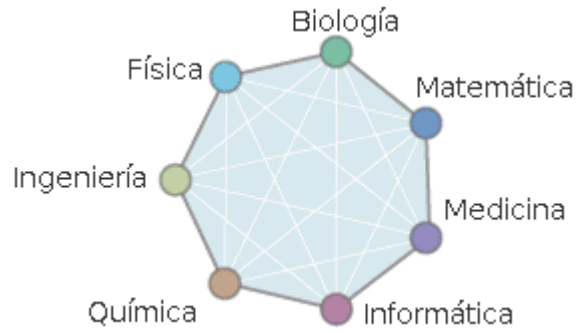


Figura 1.1

La BS es un área de investigación científica que se preocupa del estudio de procesos biológicos usando un enfoque sistémico. A diferencia de los métodos clásicos de estudio usados por los biólogos, que se basan en la confirmación o refutación de hipótesis vía resultados experimentales, la Biología de Sistemas emplea técnicas de predicción rigurosas. Estas técnicas surgen fundamentalmente del uso de

modelos matemáticos que describen el comportamiento del ente en estudio. Los modelos permiten predecir el comportamiento del proceso como un sistema dinámico, generalmente tratado como una red compleja.

Existen quienes erróneamente confunden la Biología de Sistemas con Sistemas Biológicos, por lo cual es preciso aclarar que los Sistemas Biológicos son por definición, sistemas abiertos que operan en condiciones alejadas del equilibrio termodinámico, con muchas y fuertes interacciones no lineales entre sus muchos elementos (4).

La comunidad de investigadores de la BS ha propuesto un modelo al que llaman "las 4 Ms" (Figura 1.2) para el estudio de los SB, dividiéndolo esencialmente en dos áreas, un área experimental y otra computacional. Dentro del área experimental se encuentran la manipulación de datos y las mediciones, dentro del área computacional la minería y la modelación (2).

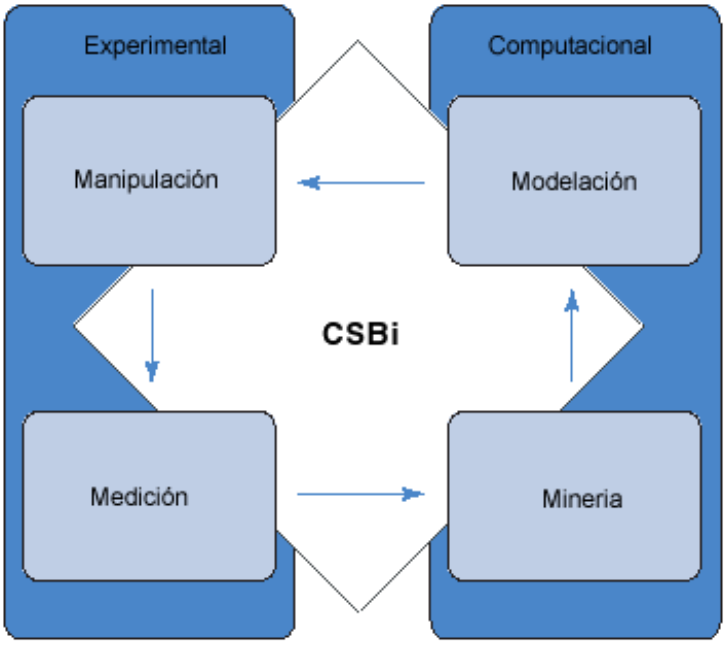


Figura 1.2

Un paso importante para que los científicos comiencen a utilizar el enfoque sistémico en sus investigaciones ha sido el desarrollo de herramientas computacionales para el manejo de la información

en estas 4 áreas. A pesar del gran esfuerzo de la comunidad internacional, aun existen grandes limitaciones en cuanto a software se refiere. Por ejemplo, en el área de modelación, los sistemas desarrollados han sido o muy rígidos en cuanto a los niveles de abstracción a los que permite realizar las modelaciones o muy difíciles de utilizar.

Es por ello que se ha desarrollado la presente investigación con el objetivo de desarrollar un nuevo software que resuelva algunas de estas limitaciones, siendo más flexible en cuanto a la creación de los modelos gráficos de un Sistema Biológico dado.

1.3 MODELACIÓN GRÁFICA DE SISTEMAS BIOLÓGICOS

La información que se tiene de los SB cada día es mayor, lo cual dificulta su comprensión. Modelar ha sido históricamente una de las soluciones dadas por el hombre al problema de comprender sistemas complejos, la BS no está ausente a esa solución. La modelación gráfica se ha convertido en una vía ampliamente adoptada por la comunidad de investigadores de la BS a través de la cual se describe un SB en forma de un gráfico interpretado que representa sus dependencias.

Extensas investigaciones que abarcan las moléculas y sus relaciones en la actualidad están siendo relevantes para el descubrimiento y comprensión del funcionamiento de importantes SB. Sin embargo, los conocimientos sobre las interacciones moleculares están mayormente descritos por texto plano o diagramas tradicionales. El texto plano es ambiguo y sus resultados pueden ser interpretados por varios lectores de forma distinta; los diagramas tradicionales son informales y a menudo confusos (1). Algunos investigadores han interpretado la situación anterior como la necesidad del surgimiento de una potente y estandarizada notación gráfica para describir redes biológicas y sus procesos.

Hay algunos criterios que el sistema de notación debe cumplir, tales como:

- **Expresividad**: El sistema de notación debe ser capaz de describir todas las posibles relaciones entre genes y proteínas, así como procesos biológicos en su conjunto.
- **Sin ambigüedades semánticas**: La notación debe ser sin ambigüedad, diferentes semánticas deben ser asignadas a distintos símbolos que sean claramente distinguibles.
- **Sin ambigüedades visuales**: Cada símbolo debe ser claramente identificado y no puede ser

confundido con otros símbolos.

- **Habilidad de extensión**: El sistema de notación debe ser bastante flexible para adicionar nuevos símbolos y relaciones de manera consistente.
- **Traducción matemática**: La notación debe ser capaz de convertirse por si sola en forma matemática, tales como ecuaciones diferenciales, que pueden ser directamente aplicada a análisis numéricos.
- **Soporte de software**: La notación debe ser soportada por software para su dibujo, edición y traducción a la forma matemática.

Con el empleo de una notación gráfica para la representación de procesos e interacciones entre los distintos componentes de un Sistema Biológico los investigadores podrían en gran medida comprender procesos complejos. Un importante investigador de la rama, Hiroaki Kitano, propuso un estándar de notación gráfica para la modelación de SB.

Existen varios software que pueden ser usados para modelar SB, en el siguiente epígrafe se describen algunos de ellos. Algunas características que harían de dichos software herramientas computacionales factibles para el estudio de SB además de las mencionadas anteriormente para la notación gráfica serían:

- **Uso de estándares**. Característica que permite a los usuarios que utilicen el software reutilizar modelos elaborados por otros investigadores, reduciendo el tiempo de estudio y los riesgos a cometer errores. Dentro de los estándares más utilizados en la rama de la BS están el SBML, MathML y el CellML.

- **Modelar a más de un nivel de abstracción**. La BS abarca diferentes niveles de abstracción, que van desde el nivel molecular, hasta los ecosistemas, pasando por el nivel celular, tejidos, órganos, sistemas de órganos, organismos y poblaciones. Por lo general, los software desarrollados sólo permiten modelar a uno de estos niveles. Es importante que estos software permitan modelar integrando diferentes niveles (2).

- **Interfaz amigable**. La BS es un campo interdisciplinario, si la herramienta para modelar presenta una interfaz agradable permitiría a los investigadores de dicha área no especialistas en Biología o ciencias afines elaborar modelos con facilidad.

- **Crear componentes reutilizables**. Existen algunos modelos que tienen gran parecido, muchos de los componentes presentes en los mismos podrían extraerse como factor común. Si el usuario pudiera crear a partir de dichos modelos un componente reutilizable, reduciría el período de las investigaciones.

1.4 SOFTWARE PARA LA MODELACIÓN GRÁFICA DE SISTEMAS BIOLÓGICOS

1.4.1 BIOUML

BioUML es un framework diseñado para el trabajo con SB que brinda una notación gráfica formalizada para describir el funcionamiento y la estructura de los Sistemas Biológicos, su visualización y simulación, así como acceder a bases de datos con datos experimentales relevantes (5).

Arquitectónicamente está estructurado por varios módulos, un visor de diagrama, un editor de diagramas, un modulo de bases de datos que incluye un motor de búsquedas con el cual el usuario puede interactuar de manera amigable y herramientas para la simulación.

Centrados en el módulo de modelación gráfica se puede decir que permite modelar Sistemas Biológicos pero presenta algunas deficiencias como:

- Las interacciones entre los distintos componentes del modelo son visualmente ambiguas, cada tipo de interacción no es claramente identificada de forma visual.
- No permite crear componentes reutilizables a partir de un modelo creado.
- No da la posibilidad de modelar amigablemente a más de un nivel de abstracción (Nivel intracelular y extracelular).
- La interfaz para modelar no es lo suficientemente amigable como para permitirle a investigadores no especialistas en sistemas biológicos modelar fácilmente.

1.4.2 DYNETICA

Dynetica es una interfaz para la modelación que posibilita una fácil construcción, visualización y análisis de modelos cinéticos. Ha sido diseñado específicamente para modelar SB. Permite llevar a cabo la simulación de los modelos elaborados a través de distintos algoritmos para la simulación tales como (RungeKutta4, RungeKutaFehlberg, FirstReactionMethod, OptimizedDirectMethod, DirectMethod)

Posibilita elaborar modelos utilizando los siguientes tipos de reacciones:

- MassAction: representa la cinética del modelo basada en una ley de acción de masa.
- Generator: representa una reacción en la cual la producción de una sustancia X no influye sobre su productor.
- Decay: representa la descomposición de una sustancia a través de una regla cinética.
- MichaelisMentenReaction: representa una reacción basada en la ley de cinética Michaelis-Menten.

En la Figura 1.3 se puede apreciar un modelo elaborado utilizando Dynetica.

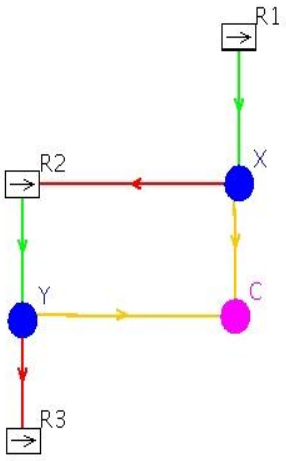


Figura 1.3

Entre las dificultades encontradas desde el punto de vista de ser un editor gráfico se encuentra que las interacciones representadas en los modelos son visualmente ambiguas, en el ejemplo mostrado en la Figura 1.3 hay representada varias interacciones en los componentes X,Y,C que desde el punto de vista biológico son distintas pero si se observa el modelo visualmente no se aprecian diferencias.

Solo se permite elaborar modelos de sistemas utilizando los componentes predefinidos, impidiéndole al usuario crear componentes reutilizables a partir de modelos elaborados.

No utiliza ninguno de los estándares existentes para salvar ni leer los modelos, característica que constituye una debilidad muy fuerte debido a que no podrá servirle a los usuarios para el estudio de sistemas conocidos que se encuentran en este tipo de formato (SBML, MathML).

1.4.3 CELLDESIGNER

CellDesigner es un editor de diagramas estructurados para la modelación de genes y redes bioquímicas. Las redes son modeladas a través de diagramas de procesos con la notación propuesta por Kitano y almacenados siguiendo el formato del estándar SBML (Systems Biology Markup Language).

Características del CellDesigner:

- Representación de semánticas bioquímicas.
- Descripción detallada de las transiciones de estados de las proteínas.
- Utiliza el estándar SBML.
- Portabilidad extrema siendo una aplicación desarrollada en Java.

El objetivo del desarrollo del CellDesigner fue proveer un editor de diagramas de procesos con tecnología estandarizada para cualquier plataforma computacional, posibilitando ofrecer sus beneficios a tantos usuarios como fuese posible (6). El modelo generado con este sistema puede ser usado por otras aplicaciones que estén basadas en los estándares antes mencionados.

A pesar de las características mencionadas el CellDesigner presenta algunas limitantes, tales como no modelar a más de un nivel de abstracción, no permitir al usuario crear componentes reutilizables, su interfaz no posibilita a investigadores no especialistas en Sistemas Biológicos modelar fácilmente.

1.4.4 CYTOSCAPE

CytoScape es un software bioinformático para la visualización de redes de interacciones moleculares, posibilita la integración de las mismas con otros datos del estado tales como perfiles de expresión de genes. Algunas facilidades adicionales pueden agregársele a través de plugins como el acceso a bases de datos y análisis de los modelos elaborados.

La organización central del CytoScape es un diagrama de red donde las especies moleculares son representadas como nodos y las interacciones moleculares como aristas entre los nodos, dicha representación es visualmente ambigua.

Permite elaborar modelos de sistemas utilizando los componentes predefinidos, impidiéndole al usuario crear componentes reutilizables a partir de modelos elaborados.

Los modelos elaborados solo abarcan un nivel de abstracción, no permitiendo a los usuarios modelar procesos internos al fenómeno estudiado que podrían variar el sentido de la investigación.

1.5 NECESIDAD

Como se ha visto en los análisis realizados, aunque el desarrollo de diferentes software ha ido en ascenso en los últimos años, falta mucho por hacer todavía. La mayoría de estos sistemas presentan limitantes que impiden su utilización fuera del marco para el cual fueron diseñados.

Tratando de limar un poco estas deficiencias se desarrolla el presente trabajo de diploma, con el objetivo de crear un nuevo sistema que sea capaz de integrar en una plataforma amigable, las ideas y conceptos que a lo largo de este capítulo se han ido introduciendo.

A continuación se hará un breve análisis de las principales tendencias en cuanto a metodologías de desarrollo, herramientas y lenguajes de programación utilizadas para el análisis, diseño e implementación de este tipo de software y se seleccionarán aquellas que se consideren más apropiadas para el desarrollo de la aplicación.

1.6 HERRAMIENTAS Y TECNOLOGÍAS

1.6.1 METODOLOGÍAS Y PROCESOS PARA EL DESARROLLO DE SOFTWARE

En el desarrollo de un software los desarrolladores se enfrentan a un reto importante, desarrollar con calidad y en el menor tiempo posible. Con el transcurso del tiempo y a medida que se revoluciona la producción de software a nivel mundial, se han ido creando metodologías y procesos que aceleran, fortalecen y mejoran la producción.

Una de las metodologías utilizadas es **Extreme Programming** (XP) o programación extrema, una metodología reciente en el desarrollo de software. La filosofía de XP es satisfacer al completo las necesidades del cliente, por eso lo integra como una parte más del equipo de desarrollo.

XP fue inicialmente creada para el desarrollo de aplicaciones donde el cliente no sabe muy bien lo que quiere, lo que provoca un cambio constante en los requisitos que debe cumplir la aplicación. Por este motivo es necesaria una metodología ágil como XP que se adapta a las necesidades del cliente y dónde la aplicación se va reevaluando en períodos de tiempo cortos (7).

XP está diseñada para el desarrollo de aplicaciones que requieran un grupo de programadores pequeño, dónde la comunicación sea más factible que en grupos de desarrollo grandes. La comunicación es un punto importante y debe realizarse entre los programadores, los jefes de proyecto y los clientes.

Las características esenciales de esta metodología son las siguientes:

- Comunicación: Los programadores están en constante comunicación con los clientes para satisfacer sus requisitos y responder rápidamente a los cambios de los mismos. Muchos problemas que surgen en los proyectos se deben a que después de concretar los requisitos que debe cumplir el programa no hay una revisión de los mismos, pudiendo dejar olvidados puntos importantes.
- Simplicidad: Codificación y diseños simples y claros. Muchos diseños son tan complicados que cuando se quieren ampliar resulta imposible hacerlo y se tienen que desechar y partir de cero.

- Realimentación (Feedback): Mediante la realimentación se ofrece al cliente la posibilidad de conseguir un sistema apto a sus necesidades ya que se le va mostrando el proyecto a tiempo para poder ser cambiado y poder retroceder a una fase anterior para rediseñarlo a su gusto.

XP no es recomendable aplicarla al desarrollo de la herramienta debido a que la misma exige la integración total del cliente al equipo de trabajo; condición con la cual no se cuenta, pues el cliente es un agente externo al equipo de trabajo.

El **Proceso unificado de desarrollo de software** (RUP) es en la actualidad también frecuentemente utilizado en el desarrollo de software.

RUP es un proceso de desarrollo de Software, o sea es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el Proceso Unificado de desarrollo de software es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto (8).

Existen tres características claves presentes en RUP, ellas son: dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental.

- Dirigido por los casos de uso: Teniendo en cuenta que la razón de ser de un sistema es brindar servicios a los usuarios, RUP define caso de uso como el conjunto de acciones que debe realizar un sistema para dar un resultado de valor a un determinado usuario y los utiliza tanto para especificar los requisitos funcionales del sistema, como para guiar todos los demás pasos de su desarrollo, dígase diseño, implementación y prueba.
- Centrado en la arquitectura: La arquitectura es una vista del diseño completo con las características más importantes, dejando a un lado los detalles. Esta no solo incluye las necesidades de los usuarios e inversores, sino también otros aspectos técnicos como el hardware, sistema operativo, sistema de gestión de base de datos, protocolos de red; con los que debe coexistir el sistema. En otras palabras, la arquitectura representa la forma del sistema, la cual va madurando en su interacción con los casos de uso hasta llegar a un equilibrio entre funcionalidad y características técnicas.

- Iterativo e incremental: La alta complejidad de los sistemas actuales hace que sea factible dividir el proceso de desarrollo en varios mini-proyectos. Cada uno de estos mini-proyectos se les denomina iteración y pueden o no representar un incremento en el grado de terminación del producto completo. En cada iteración los desarrolladores seleccionan un grupo de casos de uso, los cuales se diseñan, implementan y prueban. La planificación de iteraciones hace que se reduzcan los riesgos de los costes de un solo incremento, no sacar al mercado un producto en el tiempo previsto, mantener la motivación del equipo pues puede ver avances claros a corto plazo y que el desarrollo pueda adaptarse a los cambios en los requisitos.

RUP es el proceso seleccionado para el desarrollo del presente trabajo por su adaptabilidad al mismo y las condiciones de desarrollo que ofrece que garantizan el cumplimiento de las metas trazadas.

1.6.2 LENGUAJE DE PROGRAMACIÓN

Durante las últimas dos décadas, C y C++ han sido los lenguajes más utilizados para desarrollar software. Estos lenguajes ofrecen una gran flexibilidad pero, en cambio, la productividad no es muy alta ya que requieren mucho tiempo de desarrollo.

Si bien un aspecto importante a tener en cuenta es el tiempo de desarrollo otro aspecto mucho más importante es la portabilidad, es decir usar la misma aplicación en distintas arquitecturas o sistemas operativos sin tener que recompilar.

Java y C Sharp (C#) son lenguajes de programación que cumplen con las características antes mencionadas. Cuando se refiere a tiempo de desarrollo se puede decir que C# es superior a Java, pero en cuanto a la portabilidad, ¿qué sucede?

Existen algunos proyectos como Mono y Portable.NET para hacer de C# un lenguaje portable. En la actualidad aunque se han notado avances dichos proyectos todavía están en desarrollo.

En cuanto a Java, utiliza el concepto de maquina virtual (VM). El código que se genera no es específico a una plataforma en particular. Un programa nativo: la VM se encarga de traducir este código para que cualquier ordenador pueda ejecutarlo. De esta manera un código generado en Java puede correr en cualquier plataforma, en donde se haya portado la VM.

Finalmente se utilizará Java como lenguaje de programación para desarrollar la herramienta por ser un lenguaje cuya portabilidad está verdaderamente probada y no requerir largos períodos de tiempo para el desarrollo de las aplicaciones. Otras características de dicho lenguaje son:

- Orientado a Objetos: Java trabaja con sus datos como objetos y con interfaces a esos objetos, soporta las características propias del paradigma orientado a objetos: abstracción, encapsulamiento, herencia y polimorfismo.
- Simple: Posee una curva de aprendizaje muy rápida. Ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos.
- Robusto: Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores lo antes posible en el ciclo de desarrollo. Java obliga a la declaración explícita de los tipos de los ítems de información, reduciendo así las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de la misma.

1.6.4 SISTEMA DE CONTROL DE VERSIONES

Un sistema de control de versiones es un software que administra el acceso a un conjunto de ficheros y mantiene un historial de cambios realizados. El control de versiones es útil para guardar cualquier documento que cambie con frecuencia, como una novela, o el código fuente de un programa (9).

Normalmente consiste en una copia maestra en un repositorio central, y un programa cliente con el que cada usuario sincroniza su copia local. Esto permite compartir los cambios sobre un mismo conjunto de ficheros. Además, el repositorio guarda registro de los cambios realizados por cada usuario, y permite volver a un estado anterior en caso de necesidad.

¿Por qué son necesarios? Supongan que se encuentran desarrollando un programa de cierto tamaño en colaboración con otra persona. Lo más primitivo es compartir cambios usando ficheros comprimidos. Pero este sistema es propenso a errores: ¿están enviando todo el código?, ¿están sobrescribiendo algún cambio?, ¿que ficheros deben actualizar?, ¿quien tiene la versión maestra del código?

Todos los sistemas de control de versiones tienen ciertas características que acaban con estas preocupaciones. Esto es lo que aporta un sistema de control de versiones a un equipo: actualización de ficheros modificados, copias de seguridad centralizadas, historial de cambios, acceso remoto, seguridad.

Para el desarrollo de esta herramienta se utilizará el sistema de control de versiones Subversion (SVN). ¿Por qué SVN?

En muchos proyectos comerciales, y prácticamente en todos los de código abierto se utiliza el control de versiones Concurrent Versions System (CVS), el cual presenta algunos problemas que son solucionados con SVN, los problemas son los citados a continuación:

- No registra cambios en la estructura de directorios: no es posible mover, renombrar, ni copiar. Estas operaciones se consiguen eliminando y añadiendo, pero con esto se pierde el historial de cambios.
- Es necesario interrumpir el acceso al repositorio para crear copias de seguridad.
- No permite "conjuntos de cambios". Cuando un desarrollador sube un conjunto de cambios, se van subiendo uno a uno, quizás al mismo tiempo que otro desarrollador hace lo mismo. Al no ser una operación atómica, nadie puede asegurar que el estado del repositorio tras su commit, sea el mismo que el estado que probó en local, y por tanto, el proyecto puede estar en un estado que nadie ha probado. Además, deshacer un conjunto de cambios requiere recorrer el repositorio entero comparando las fechas.
- Almacena ficheros binarios enteros (no sus diferencias entre versiones). Esto consume espacio en disco y ancho de banda.
- No usa la red eficientemente. Las diferencias entre versiones solo se envían desde el servidor al cliente, cuando el cliente sube sus cambios envía ficheros enteros.
- El código fuente es difícil de mantener. CVS comenzó como un conjunto de shellscripts que usaban RCS e implementaban algoritmos desarrollados entre los años 60-80. El resultado actual es producto de sucesiones de parches, y no tiene un diseño fácil de entender o mejorar. Esto dificulta su evolución. La idea de crear un nuevo CVS desde cero, surgió en la propia compañía que ofrecía soporte comercial para el CVS.

1.6.4 ENTORNO DE DESARROLLO

Dentro de los entornos de desarrollo integrado (IDE del inglés Integrated Development Environment) para el desarrollo de aplicaciones usando como lenguaje de programación Java se pueden encontrar NetBeans, JBuilder y Eclipse.

El **NetBeans** es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans, es un producto libre y gratuito sin restricciones de uso.

La versión actual todavía no permite la integración del entorno de desarrollo con un control de versiones Subversion, por esa razón no fue seleccionado este IDE para el desarrollo de la herramienta.

JBuilder es un entorno de desarrollo integrado para el lenguaje de programación Java desarrollado por Borland. Posee varias ediciones, la Enterprise, para aplicaciones J2EE, Web Services y struts. La Developer, para el desarrollo completo de aplicaciones Java, y la Foundation, con capacidades básicas para iniciarse en Java.

No fue seleccionado para el desarrollo de la herramienta JBuilder por no ser multiplataforma, solo puede ser ejecutado sobre el sistema operativo Windows.

El **Eclipse** es un IDE para todo tipo de aplicaciones, inicialmente desarrollado por IBM, y actualmente gestionado por la Fundación Eclipse.

La característica clave de Eclipse es la extensibilidad. Eclipse es una gran estructura formada por un núcleo y muchos plugins que van conformando la funcionalidad final. La forma en que los plugins interactúan es mediante interfaces o puntos de extensión; así, los nuevos aportes se integran sin dificultad ni conflictos.

Se seleccionó este último IDE para el desarrollo de la herramienta principalmente porque permite el desarrollo de aplicaciones utilizando Subversion como sistema de control de versiones y por el potente editor de código que presenta.

Entre otras características de dicho IDE se encuentran que es multiplataforma, soporte para distintas arquitecturas, resaltado de sintaxis, auto completado, tabulador de un bloque de código seleccionado, asistentes (wizards): para la creación, exportación e importación de proyectos y para generar plantillas de códigos (templates).

1.6.5 HERRAMIENTA CASE

Las **Herramientas CASE** (**C**omputer **A**ided **S**oftware **E**ngineering, Ingeniería de Software Asistida por Computadoras) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software.

Algunas herramientas CASE conocidas son el ArgoUML, Rational Rose, Visual Paradigm, Easy CASE, Xcase, CASE Studio 2, CASEWise entre otras. Dentro de las más conocidas se encuentra el Rational Rose y el Visual Paradigm.

Rational Rose es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables.

Visual Paradigm es una potente herramienta CASE que utiliza como lenguaje de modelación el UML. La herramienta fue desarrollada para una amplia gama de usuarios incluyendo Ingenieros de Software, analista de sistemas, analistas del negocio y arquitectos de sistemas. Proporciona a los desarrolladores una plataforma con interfaz amigable que les permite diseñar un producto con calidad de forma rápida.

Facilita la interoperabilidad con otras herramientas CASE y se integra con las siguientes herramientas Java: Eclipse/IBM WebSphere, JBuilder, NetBeans IDE, Oracle JDeveloper, BEA Weblogic. Está disponible en varias ediciones: Enterprise, Professional, Community, Standard, Modeler y Personal.

Por el equipo de desarrollo trabajar sobre la plataforma GNU\Linux y no contar con la versión de Rational Rose que corre sobre dicha plataforma se usará para el desarrollo de la aplicación la herramienta CASE Visual Paradigm.

1.6 CONCLUSIONES

Se desarrollará una herramienta computacional para la modelación gráfica de sistemas biológicos utilizando la metodología de desarrollo de software RUP, el lenguaje de programación Java, el sistema de control de versiones Subversion, el IDE Eclipse y la herramienta CASE Visual Paradigm.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 INTRODUCCIÓN

En el presente capítulo se da una breve descripción del sistema ha desarrollar, se describe el modelo de dominio perteneciente a un Sistema Biológico así como la presentación del diagrama de objetos, se describen los requisitos funcionales y no funcionales del sistema, se definen los actores del sistema, el diagrama de caso de uso del sistema y se hace una descripción detallada de los casos de usos del sistema.

2.2 BREVE DESCRIPCIÓN DEL SISTEMA

El sistema ha desarrollar a través de la realización del presente trabajo de diploma constituirá una herramienta computacional para la modelación gráfica de SB que puedan ser descritos mediante dinámica de población, podrá funcionar sin la necesidad de usar sistemas externos pero a la vez será diseñado de tal forma que pueda ser integrado con otras herramientas como un editor de ecuaciones y una herramienta para la simulación y análisis de SB.

Las tres herramientas anteriormente mencionadas en su conjunto constituirán una potente herramienta computacional para el estudio de SB, pues permitirá modelar gráficamente, modelar matemáticamente, realizar simulaciones y analizar los resultados de las mismas. Será uno de los resultados alcanzados por el grupo de Bioinformática de la Universidad de las Ciencias Informáticas (UCI).

2.3 MODELO DE DOMINIO

La modelación del negocio es una de las fases que RUP incluye en el desarrollo de software, uno de sus objetivos es comprender el objeto a automatizar. Cuando no existen procesos definidos RUP propone realizar un modelo del dominio, que es un subconjunto del modelo de negocio.

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las cosas que existen o eventos que ocurren en el entorno en el que trabaja el sistema (9). El modelo de dominio se describe mediante diagramas UML (específicamente mediante diagramas de clases). Ver anexo 1 para observar el modelo de dominio correspondiente al presente trabajo.

Descripción textual del modelo de dominio.

Un SB puede ser representado por varios compartimientos, procesos, señales, operadores y químicos.

Los compartimientos pueden definirse como abstracciones que los investigadores utilizan para delimitar y agrupar componentes de los SB. Dentro de los compartimientos que agrupan componentes se encuentran la membrana celular y el interior celular, la unión de los dos es utilizada para modelar células.

Los procesos se utilizan para modelar interacciones entre los distintos componentes de un SB, existen procesos químicos y procesos celulares. Dentro de los procesos químicos se encuentran la degradación, la catálisis, la inhibición y la reacción. Dentro de los celulares la proliferación, la diferenciación, la muerte y la unión.

En el interior de las células existen interacciones; para modelar las mismas serán utilizadas las señales, dentro de las cuales se encuentran la inhibición, la catálisis y la auto-activación.

Tanto los procesos como las señales en ocasiones pueden relacionarse dando como resultado un nuevo proceso o señal, esta acción es modelada a través de los operadores, el operador **and** y el **or**. El **and** es utilizado cuando los dos procesos relacionados dan como resultado el tercero. El **or** es utilizado cuando alguno de los dos procesos relacionados da como resultado el tercero.

Los químicos son utilizados para modelar sustancias o componentes que influyen en el comportamiento de los SB. La Citoquina es uno de los químicos utilizados para modelar compuestos o sustancias que alteran los procesos celulares.

Los receptores, canales de salida y complejos son químicos especiales utilizados para modelar acciones que ocurren en la membrana celular al interactuar procesos o señales con la misma.

2.4 DEFINICIÓN DE LOS REQUISITOS FUNCIONALES

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. La herramienta a desarrollar a través de la realización del presente trabajo debe cumplir los siguientes requisitos:

- **RF 1-)** Modelar gráficamente sistemas biológicos que puedan ser descritos mediante dinámica de población.
  - **RF 1.1-)** Modelar gráficamente compartimientos.
    - **RF 1.1.1-)** Insertar compartimientos.
    - **RF 1.1.2-)** Modificar las propiedades de los compartimientos.
    - **RF 1.1.3-)** Eliminar compartimientos.
  - **RF 1.2-)** Modelar gráficamente células.
    - **RF 1.2.1-)** Insertar células.
    - **RF 1.2.2-)** Modificar las propiedades de las células.
    - **RF 1.2.3-)** Eliminar células.
  - **RF 1.3-)** Modelar gráficamente compuestos químicos.
    - **RF 1.3.1-)** Insertar compuestos químicos.
    - **RF 1.3.2-)** Modificar las propiedades de los compuestos químicos.
    - **RF 1.3.3-)** Eliminar compuestos químicos.
  - **RF 1.4-)** Modelar gráficamente interacciones presentes entre los distintos componentes del sistema biológico a nivel extracelular.
    - **RF 1.4.1-)** Insertar procesos.
    - **RF 1.4.2-)** Modificar las propiedades de los procesos.
    - **RF 1.4.3-)** Eliminar procesos.
  - **RF 1.5-)** Modelar gráficamente interacciones presentes entre los distintos componentes del sistema biológico a nivel intracelular.
    - **RF 1.5.1-)** Insertar señales.
    - **RF 1.5.2-)** Modificar las propiedades de las señales.
    - **RF 1.5.3-)** Eliminar señales.
  - **RF 1.6-)** Modelar gráficamente operadores que modifican las interacciones presentes en el sistema biológico.
    - **RF 1.6.1-)** Insertar operadores.
    - **RF 1.6.2-)** Eliminar operadores.

- **RF 2-)** Modelar gráficamente a más de un nivel de abstracción (nivel intracelular y extracelular).
- **RF 3-)** Mostrar propiedades de los componentes del sistema biológico.
- **RF 4-)** Permitir el trabajo con nuevos componentes reutilizables a partir de un modelo gráfico creado.
  - **RF 4.1-)** Crear nuevos componentes reutilizables.
  - **RF 4.2-)** Insertar nuevos componentes reutilizables.
- **RF 5-)** Guardar la información del sistema biológico en formato BioSyS.
- **RF 6-)** Cargar la información del sistema biológico desde el formato BioSyS.

2.5 DEFINICIÓN DE LOS REQUISITOS NO FUNCIONALES

Los requisitos no funcionales son otros requisitos que no forman parte de la funcionalidad principal de la aplicación, como requisitos del entorno de desarrollo o ejecución (sistema operativo, servidores en los que correrá, lenguajes, etc.), restricciones que se aplicarán, prestaciones (tiempo de respuesta mínimo, alta disponibilidad, etc.), fiabilidad, portabilidad, interfaces externas, seguridad y otros. Se puede decir que responden a aquellas cualidades y características que el producto debe tener para que sea atractivo, confiable, usable y seguro.

Dentro de los requisitos no funcionales para el desarrollo de la herramienta se encuentran:

Funcionalidad: A pesar de que el sistema podrá ser usado en un tiempo breve por aquellos usuarios que se encuentren capacitados para modelar Sistemas Biológicos, se debe realizar un adiestramiento previo con el objetivo de que puedan detectarse errores (en caso de que existan) y posibilitar la familiarización con la herramienta.

Apariencia o Interfaz externa: Se necesita una interfaz amigable, legible, interactiva, fácil de usar, profesional, clara y sencilla.

Usabilidad: El sistema podrá ser usado por aquellos usuarios que posean conocimientos básicos en el campo de la modelación de sistemas biológicos.

Soporte

- **Mantenimiento**: El sistema debe estar bien documentado de forma tal que el tiempo de mantenimiento sea mínimo en caso de necesitarse.
- **Instalación**: La instalación del sistema debe caracterizarse por su facilidad, claridad y sencillez.

Portabilidad: El sistema debe ser multiplataforma (ser capaz o caracterizarse por poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas).

Confiabilidad

- **Tiempo medio de reparación**: La reparación del sistema en caso de surgir fallas en el mismo debe realizarse en el menor tiempo posible, poniendo todos los esfuerzos en función de que no supere las 72 horas.
- **Ayuda y Documentación**: El sistema constará con una ayuda en línea donde esté presente la documentación básica que posibilite comprender su funcionamiento y las funcionalidades generales a tener en cuenta para garantizar la utilización del mismo de manera eficiente.

Software: Para el uso del sistema es necesario tener instalado la máquina virtual de java 1,5 o una versión superior.

2.6 DIAGRAMA DE CASOS DE USO DEL SISTEMA

Los actores representan terceros fuera del sistema que interactúan con este. Acorde a la situación en la que se desarrollará la herramienta fueron encontrados los siguientes actores:

Actor(s)	Justificación
Investigador	Es la persona que interactúa con el sistema. Elabora el modelo gráfico del sistema biológico, procesa sus datos y realiza los estudios correspondientes.

Cada forma en que los actores usan un sistema se representa con un caso de uso, los casos de uso son "fragmentos" de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. De manera más precisa un caso de uso especifica una secuencia de acciones que el sistema debe llevar a cabo, interactuando con sus actores donde se obtiene un resultado de valor para los actores (8).

Un diagrama de casos de uso del sistema contiene actores, casos de uso del sistema y las relaciones existente entre los mismo. Ver anexo 2.2 correspondiente al diagrama de casos de uso del sistema del presente trabajo.

2.7 DESCRIPCIÓN TEXTUAL DE LOS CASOS DE USO DEL SISTEMA

Para entender la funcionalidad asociada a cada caso de uso no es suficiente con la representación gráfica del Diagrama de casos de uso, también se lleva a cabo la descripción textual de cada caso de uso donde se especifican todas las acciones necesarias para la realización del mismo.

A continuación se muestran las descripciones textuales de los casos de uso del sistema.

Caso de uso		
CU- 01	Insertar compartimiento	
Propósito	Permitir al investigador agregar un nuevo compartimiento al modelo.	
Actores: Investigador (Inicia)		
Resumen: El caso de uso se inicia cuando el investigador selecciona de la paleta de componentes el compartimiento, la célula o un componente reutilizable, posteriormente indica cuál será su localización en el área de modelación y el sistema agrega el nuevo componente al modelo.		
Referencias	**R1.1.1 , R1.2.1 , R4.2**	
Acción del actor		**Respuesta del sistema**
1. Selecciona de la paleta de componentes el componente compartimiento.		
2. Selecciona la ubicación en el modelo del sistema biológico donde será creado.		3. Actualiza el modelo del SB.
		4. Muestra la información del compartimiento creado.

Caso de uso		
CU- 02	Insertar químico	
Propósito	Permitir al investigador agregar un químico al modelo.	
Actores: Investigador (Inicia)		
Resumen: El caso de uso se inicia cuando el investigador selecciona de la paleta de componentes un químico, posteriormente indica cuál será su localización en el área de modelación y el sistema agrega el nuevo componente al modelo.		
Referencias	**R1.3.1**	
Acción del actor		**Respuesta del sistema**
1. Selecciona de la paleta de componentes el componente químico.		
2. Selecciona la ubicación en el modelo del sistema biológico donde será creado.		3. Chequea que la ubicación del químico es válida.
		4. Actualiza el modelo del SB..
		5. Muestra la información del químico creado.
Flujo alternativo		
Acción del actor		**Respuesta del sistema**
Acción 3		
		4. Si la ubicación del químico no es válida, muestra un mensaje al investigador indicando que la ubicación escogida para el químico no es válida.

Caso de uso		
CU- 03	Insertar Proceso	
Propósito	Permitir al investigador agregar un proceso al modelo.	
Actores Investigador (Inicia)		
Resumen: El caso de uso se inicia cuando el investigador selecciona de la paleta de componentes un proceso. Luego selecciona en el área de modelación cuál será el componente que le dará origen y en caso que lo requiera, selecciona también el componente destino. Posteriormente el sistema agrega el nuevo proceso al modelo.		
Referencias	**R1.4.1 , R.1.6.1**	
Acción del actor		**Respuesta del sistema**
1. Selecciona de la paleta de componentes el componente proceso.		
2. Selecciona del modelo el componente que dará origen al proceso.		3. Chequea si el origen del proceso es válido.
		4. Determina el tipo de proceso. ● En caso que el proceso sea unión, diferenciación, reacción, catálisis o inhibición ir a la sección 1
		5. Actualiza el modelo del SB.
		6. Muestra la información del proceso creado.
Flujo alternativo		
Acción del actor		**Respuesta del sistema**
Acción 3		
		4. Si el origen no es válido, muestra un mensaje al investigador indicando que el origen escogido para el proceso no es válido.
Sección 1		
Acción del actor		**Respuesta del sistema**
5. Selecciona del modelo el componente destino del proceso.		6. Chequear si el destino del proceso es válido.
Flujo alternativo		
Acción del actor		**Respuesta del sistema**
Acción 6		
		7. Si el destino no es válido, muestra un mensaje al investigador indicando que el destino escogido para el proceso no es válido.

Caso de uso		
CU- 04	Insertar señal	
Propósito	Permitir al investigador agregar una señal al modelo.	
Actores Investigador (Inicia)		
Resumen: El caso de uso se inicia cuando el investigador selecciona de la paleta de componentes una señal. Posteriormente selecciona en el área de modelación el componente que dará origen y el destino de la señal. El sistema agrega la señal al modelo.		
Referencias	**R1.5.1 , R1.6.1 , R2**	
Acción del actor		**Respuesta del sistema**
1. Mediante un doble clic selecciona la célula donde insertará la señal.		2. Redimensiona la célula permitiéndole al investigador modelar a nivel intracelular.
3. Selecciona de la paleta de componentes el componente señal.		
4. Selecciona en la célula el componente que dará origen a la señal.		5. Chequea si el origen de la señal es válido.
6. Selecciona en el modelo del sistema biológico el componente destino de la señal. Si el componente destino seleccionado es: ● Canal de salida o Señal ver **sección 1** ● Membrana celular ver **sección 2** ● Operador ver **sección 3**		
		7. Actualiza el modelo del SB.
		8. Muestra la información de la señal creada.
Flujo alternativo		
Acción del actor		**Respuesta del sistema**
Acción 5		
		6. Si el origen no es válido, muestra un mensaje al investigador indicando que el origen escogido para el proceso no es válido.
Sección 1 Tipo de operador		
Acción del actor		**Respuesta del sistema**
		7. Muestra una ventana al investigador donde le da la posibilidad de seleccionar la relación que existirá entre la nueva señal y el destino seleccionado.
8. Selecciona la relación deseada.		9. Crea la relación seleccionada.
Sección 2 Nuevo Canal de Salida		
Acción del actor		**Respuesta del sistema**
		7. Crea un nuevo canal de salida al cual estará asociada la señal creada.

Sección 3 Insertar solo la señal	
Acción del actor	**Respuesta del sistema**
	7. Asocia la señal creada al operador seleccionado.

Caso de uso		
CU- 05	Modificar Componente	
Propósito	Permitir al investigador modificar las propiedades de los componentes del sistema biológico.	
Actores Investigador (Inicia)		
Resumen: El caso de uso se inicia cuando el investigador selecciona un componente en el área de modelación, el sistema muestra las propiedades del componente a través de un inspector de objetos. El investigador selecciona la propiedad que desea modificar e incorpora su nuevo valor, si es posible realizar la modificación el sistema actualiza el modelo.		
Referencias	**R1.1.2 , R1.2.2 , R1.3.2 , R1.4.2 , R1.5.2**	
Acción del actor		**Respuesta del sistema**
1. Ver caso de uso mostrar información.		
2. El usuario selecciona del inspector de propiedades la propiedad a modificar y modifica el valor existente.		3. Chequea si la modificación hecha es válida.
		4. Actualiza el modelo del SB.
		5. Muestra la información de la señal creada.
Flujo alternativo		
Acción del actor		**Respuesta del sistema**
Acción 3		
		4. Si la modificación hecha no es válida, muestra un mensaje al investigador indicando que no pudo modificarse la propiedad del componente.

Caso de uso		
CU- 06	Eliminar componente	
Propósito	Permitir al investigador eliminar un componente del modelo	
Actores Investigador(Inicia)		
Resumen: El caso de uso se inicia cuando el investigador selecciona del área de modelación el componente y posteriormente solicita a través de un menú u oprimiendo la tecla suprimir eliminar el componente. El sistema solicita confirmación de la acción, en caso de ser confirmada elimina el componente del modelo.		
Referencias	**R1.1.3 , R1.2.3 , R1.3.3 , R1.4.3 , R1.5.3**	
Acción del actor		**Respuesta del sistema**
		1. Ver caso de uso mostrar información.
2. Solicita al sistema borrar el componente seleccionado.		3. Muestra una ventana donde solicita al investigador confirmar o abortar la operación de eliminar el componente.
4. Confirma la operación.		5. Elimina el componente.
		6. Actualiza el modelo del SB.
Flujo alternativo		
Acción del actor		**Respuesta del sistema**
Acción 3		
4. Aborta la operación.		

Caso de uso		
CU- 07	Eliminar Compartimiento	
Propósito	Permitir al investigador eliminar un compartimiento del modelo	
Actores Investigador(Inicia)		
Resumen: El caso de uso se inicia cuando el investigador selecciona del área de modelación un compartimiento y posteriormente solicita a través de un menú u oprimiendo la tecla suprimir, eliminar el componente seleccionado. El sistema elimina el compartimiento, todos sus componentes internos y los procesos asociados a este.		
Referencias	**R1.1.3 , R1.2.3**	
Acción del actor		**Respuesta del sistema**
		1. Ver caso de uso eliminar componente
		2. Elimina los componentes internos del compartimiento y los procesos asociados a los mismos
		3. Actualiza el modelo del SB.

Caso de uso		
CU- 08	Eliminar químico	
Propósito	Permitir al investigador eliminar un químico del modelo.	
Actores Investigador(Inicia)		
Resumen: El caso de uso se inicia cuando el investigador selecciona del área de modelación un químico y posteriormente solicita a través de un menú u oprimiendo la tecla suprimir eliminar el componente seleccionado. El sistema elimina el químico, los procesos y señales asociados.		
Referencias	**R1.3.3**	
Acción del actor		**Respuesta del sistema**
		1. Ver caso de uso eliminar componente.
		2. Elimina los procesos y/o señales asociados al químico.
		3. Actualiza el modelo del SB.

Caso de uso		
CU- 09	Eliminar proceso	
Propósito	Permitir al investigador eliminar un proceso del modelo.	
Actores Investigador(Inicia)		
Resumen: El caso de uso se inicia cuando el investigador selecciona del área de modelación un proceso y posteriormente solicita a través de un menú u oprimiendo la tecla suprimir eliminar el componente seleccionado. El sistema elimina del modelo el proceso y los componentes dependientes del mismo.		
Referencias	**R1.4.3 , , R1.6.2**	
Acción del actor		**Respuesta del sistema**
		1. Ver caso de uso eliminar componente.
		2. Chequea si existe algún operador lógico al cual esté asociado el proceso
		3. Si existe chequea si el origen del proceso es el operador.
		4. Si el origen no es el operador chequea si existen más de dos procesos asociados al operador.
		5. Si no existen más de dos procesos asociados al operador se elimina el operador.
		6. Actualiza el modelo del SB.
Flujo alternativo		
Acción del actor		**Respuesta del sistema**
Acción 2		
		3. Actualiza el modelo del SB.
Acción 3		
		4. Elimina el operador y todos los procesos asociados al mismo.
		5. Actualiza el modelo del SB.

Caso de uso		
CU- 10	Eliminar señal	
Propósito	Permitir al investigador eliminar una señal del modelo.	
Actores Investigador(Inicia)		
Resumen: El caso de uso se inicia cuando el investigador selecciona del área de modelación una señal y posteriormente solicita a través de un menú u oprimiendo la tecla suprimir eliminar el componente seleccionado. El sistema elimina del modelo la señal y los componentes dependientes de la misma.		
Referencias	**R1.5.3 , R1.6.2**	
Acción del actor		**Respuesta del sistema**
		1. Ver caso de uso eliminar componente.
		2. Eliminar los componentes (canal de salida, auto-activación, catálisis, inhibición) asociados a la señal.
		3. Chequea si existe algún operador lógico al cual esté asociada la señal.
		4. Si existe un operador asociado chequea si el origen de la señal es el operador.
		5. Si el origen no es el operador chequea si existen más de dos señales asociadas al operador.
		6. Si no existen más de dos señales asociadas al operador se elimina el operador.
		7. Actualiza el modelo del SB.
Flujo alternativo		
Acción del actor		**Respuesta del sistema**
Acción 3		
		4. Actualiza el modelo del sistema biológico.
Acción 4		
		5, Elimina el operador y todas las señales asociadas al mismo.
		6, Actualiza el modelo del sistema biológico.
Acción 5		
		6. Actualiza el modelo del sistema biológico.

Caso de uso		
CU- 11	Mostrar información	
Propósito	Mostrar la información de un componente del modelo.	
Actores Investigador (Inicia)		
Resumen: El caso de uso se inicia cuando el investigador selecciona un componente del modelo del cual el sistema muestra su información.		
Referencias	**R3**	
Acción del actor		**Respuesta del sistema**
1. El investigador selecciona un componente.		2. Muestra la información correspondiente al componente seleccionado.

Caso de uso		
CU- 12	Guardar modelo del sistema biológico	
Propósito	Permitirle al investigador guardar el modelo del sistema biológico en formato BioSyS.	
Actores Investigador(inicia)		
Resumen: El caso de uso se inicia cuando el investigador solicita guardar el modelo del sistema biológico, después de haber seleccionado la ruta donde será guardado, el sistema lleva a cabo dicho proceso.		
Referencias	**R5**	
Acción del actor		**Respuesta del sistema**
1. Solicita guardar el modelo.		2. Muestra una ventana solicitando los datos necesarios para llevar a cabo el procedimiento.
3. Llena la solicitud.		4. Guarda el modelo.

Caso de uso		
CU- 13	Cargar modelo del sistema biológico	
Propósito	Permitirle al investigador cargar un modelo de sistema biológico existente.	
Actores Investigador(inicia)		
Resumen: El caso de uso se inicia cuando el investigador solicita cargar un modelo de sistema biológico desde un fichero. La información almacenada en el fichero debe ser en nuestro formato (BioSyS). El sistema, después de haber chequeado la validez del archivo carga el modelo en el visualizador.		
Referencias	**R6**	
Acción del actor		**Respuesta del sistema**
1. Solicita cargar el modelo.		2. Muestra una ventana solicitando los datos necesarios para llevar a cabo el procedimiento.
3. Llena la solicitud.		4. Chequea la validez.
		5. Carga el modelo del sistema biológico.

Flujo alternativo	
Acción del actor	**Respuesta del sistema**
Acción 4	
	5. El sistema muestra un mensaje indicando el problema ocurrido (formato incorrecto o no fue encontrado en la dirección indicada).

Caso de uso		
CU- 14	Crear componente reutilizable	
Propósito	Permitir la creación de componentes a partir de un sistema biológico modelado que puedan ser reutilizados en otros modelos.	
Actores: Investigador(inicia)		
Resumen: El caso de uso se inicia cuando el investigador solicita crear un componente a partir de un sistema biológico modelado, el sistema crea el componente y agrega una referencia del mismo a la paleta de componentes.		
Referencias	**R4.1**	
Acción del actor		**Respuesta del sistema**
1. El investigador solicita crear un componente a partir del sistema biológico modelado.		2. Crea el componente a partir del sistema biológico modelado.
		3. Salva el componente.
		4. Agrega una referencia del mismo a la paleta de componentes.

2.7 CONCLUSIONES

En el presente capítulo fueron elaborados algunos de los artefactos propuestos por RUP para el desarrollo de software tales como el modelo de dominio, definición de los requisitos funcionales y no funcionales, diagrama de casos de uso del sistema y la descripción textual de los casos de uso del sistema.

CAPÍTULO 3: DISEÑO DEL SISTEMA

3.1 INTRODUCCIÓN

Durante el diseño se modela el sistema y se encuentra su forma (incluida la arquitectura) para que soporte todos los requisitos – incluyendo los requisitos funcionales y no funcionales – que se le suponen(8). A través del presente capítulo se dará una descripción del estilo arquitectónico utilizado para el desarrollo de la herramienta, se describirán los patrones de diseño empleados así como los diagramas de clases del diseño.

3.2 ESTILO ARQUITECTÓNICO UTILIZADO

Para el desarrollo de la herramienta se utilizó el estilo arquitectónico Modelo Vista Controlador (Model-View-Controller - MVC), clásico patrón de diseño utilizado para diseñar aplicaciones con sofisticadas interfaces donde la lógica de interfaz de usuario cambia con más frecuencia que los almacenes de datos y la lógica de negocio (10), dicho en otras palabras para el diseño de herramientas que necesitan la habilidad de mantener múltiples vistas para los mismo datos, tales como editores gráficos.

Este estilo se basa principalmente en separar en tres capas el diseño de las aplicaciones, el modelo de datos, la presentación de los mismos y las acciones de los usuarios (Figura 3.1), utilizando la capa que gestiona las acciones (controlador) como administradora de los posibles eventos.

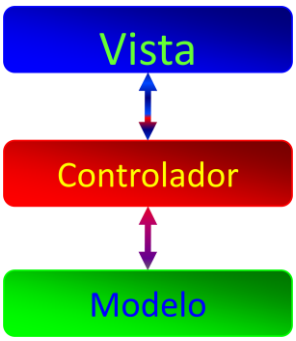


Figura 3.1

Su objetivo es realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

Entre las principales ventajas que presenta este estilo arquitectónico se encuentran:

Soporte para múltiples vistas: puesto que la vista se separa del modelo y no hay ninguna dependencia directa entre vista y modelo, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos al mismo tiempo.

Mayor soporte a los cambios: los requisitos de interfaz tienden a cambiar más rápidamente que las reglas de negocio. Puesto que el modelo no depende de las vistas, la adición de nuevos tipos de vista al sistema generalmente no afectan al modelo. Por tanto, el ámbito del cambio se limita a la vista.

3.3 PRINCIPALES PATRONES DE DISEÑO UTILIZADOS

Los Patrones de Diseño (Design Patterns) son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias).

En el presente epígrafe se describirán algunos de los patrones de diseño utilizados para el diseño de la herramienta así como los objetivos de su uso.

3.3.1 PATRÓN DELEGATION

Este es un patrón fundamental de tipo estructural. Indica cuándo no usar herencia. La delegación es una forma de extender y reutilizar la funcionalidad de una clase, escribiendo una clase adicional con funcionalidad extra que usa instancias de la clase original para proveer su propia funcionalidad. La delegación es una forma de extender el comportamiento de una clase mediante llamadas a métodos de otra clase, en lugar de heredando de ella. La delegación es más apropiada que la herencia en muchas situaciones. Por ejemplo, la herencia es útil para modelar relaciones de tipo *es-un* o *es-una*, ya que estos tipos de relaciones son de naturaleza estática. Sin embargo, relaciones de tipo es-*un-rol-ejecutado-por*

son mal modeladas con herencia. La solución general propuesta en este patrón es: incorporar la funcionalidad de la clase original usando una instancia de la clase original y llamando sus métodos.

Figura 3.2

En la figura 3.2, se muestra una clase con rol Delegador que usa una clase con el rol Delegado. Aquí se usa la delegación para reutilizar y extender el comportamiento de la clase.

Para describir como fue utilizado dicho patrón en el diseño de la herramienta primero se describirá otro patrón utilizado (patrón interfaz) debido a la fuerte relación que existe entre dichos patrones en el diseño del sistema.

3.3.2 PATRÓN INTERFAZ

Es un patrón fundamental de tipo estructural. Mantiene una clase (la interfaz) que usa datos y servicios provistos por otras clases independientes, para proveer un acceso uniforme. Esta clase interfaz provee a sus clases herederas acceso uniforme a métodos y atributos específicos, sin que deban saber a qué clase específica pertenecen. El diagrama general de este puede apreciarse en la figura 3.3:

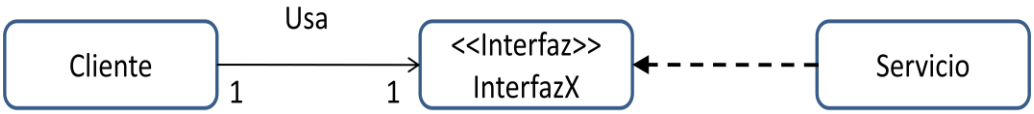


Figura 3.3

En la figura anterior, la clase Cliente usa otras clases que implementan la interfaz InterfazX la cual provee la indirección que mantiene a la clase Cliente independiente de las clases que proveen los servicios (clase Servicio). Por lo general, los patrones Interfaz y Delegación son usados juntos.

Una vez descrito los patrones anteriores se expondrá como fueron usados. Se mostrará el caso de personalización de pintado de conectores, pero el caso de pintado de componentes es análogo.

Como se puede observar en la figura 3.4 la clase ViewConnector cumple el rol de delegador y delega el pintado del conector en la interfaz DrawProcess que hace las veces de delegado. Por tanto la clase ViewConnector extenderá sus funcionalidades de pintado de conectores a través de la interfaz DrawProcess. De esta forma se pueden crear varias instancias de la clase ViewConnector que usen una interfaz DrawProcess distinta y podrán ser representados varios procesos biológicos a la vez como se puede observar en la figura 3.5.

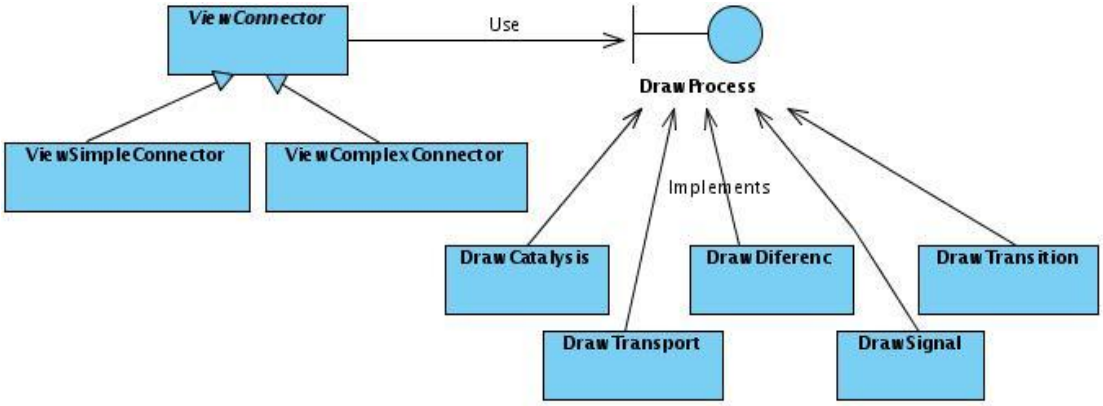


Figura 3.4

El uso de dicho patrón permite crear nuevos procesos biológicos (nuevos conectores hablando en un ámbito general) con facilidad, el propósito anterior se cumple solamente creando la interfaz DrawProcess encargada de pintarlo e implementándole el siguiente método:

```
public void Draw(Graphics g, Color color, Point[] p);
```

En este método se define cómo pintar el proceso (conector) al cual se le pasan como parámetros el Graphics (g) encargado de pintarlo, el Color (c) del que se pintará y Point [] listado de puntos (p) a través de los cuales se pintará.

Un ejemplo es el DrawProcess encargado de pintar al proceso Transport que se verá a continuación.

```
public void Draw(Graphics g, Color color, Point[] p) {

        DrawConnectorCenter.DrawLine(g, color, p);
```

```
DrawArrowCenter.DrawTransport(g, color, p[p.length-2], p[p.length-1]);
}
```

Lo primero que se hace es dibujar una línea a través de los puntos, el siguiente paso es pintarle la saeta al conector. De esa forma sencilla es incluido un nuevo proceso a la herramienta de modelación gráfica, ver figura 7, conector del extremo superior izquierdo.

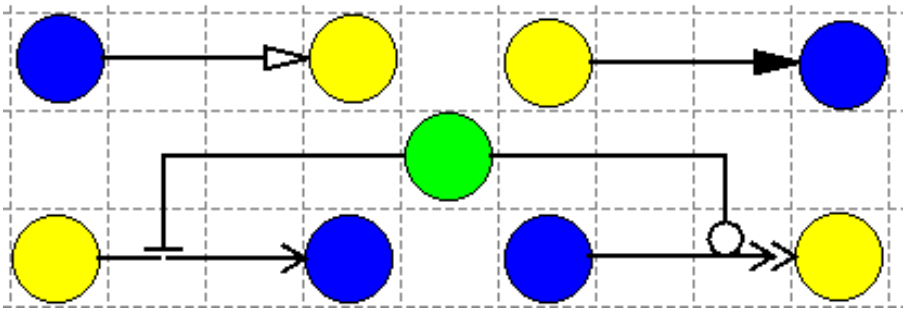


Figura 3.5

Es válido aclarar que DrawArrowCenter es una clase auxiliar que tiene implementado varios métodos para pintar saetas, si se desea crear algún proceso que utilice una saeta que no se encuentre registrada en dicha clase debe programarse la forma de dibujarla.

El DrawConnectorCenter es una clase análoga a DrawArrowCenter con la diferencia que en ella se encuentran varios métodos para pintar la línea de conexión.

3.3.3 PATRÓN COMPOSITE

La idea básica del patrón de diseño es crear un modelo en el cual existen clases de tipo componente y clases que agrupan componentes, llamadas compositoras (Composite). Estas clases pueden compartir una interfaz común, representada por una superclase. Así, no es necesario diferenciar en general entre clases componentes y clases compositoras.

Si se sigue la idea intuitiva de crear tipos de clases particularizadas para las funcionalidades de contenedor y componente, el código que las maneja ha de tratarlas también de manera diferente, con las complejidades que eso entraña.

Este patrón se puede utilizar cuando:

- Se quiere representar una jerarquía de objetos todo-parte, es decir, de agregación.
- Cuando se pretende que los clientes sean capaces de ignorar la diferencia entre objetos compuestos y objetos individuales de cara a determinadas características y comportamientos.

Como se puede ver este patrón encaja perfectamente en la idea del desarrollo de la herramienta, pues en el modelo existirán componentes y contenedores de componentes. En la figura 3.6 se muestra parte del diagrama de clases del diseño donde se muestra el uso del patrón Composite.

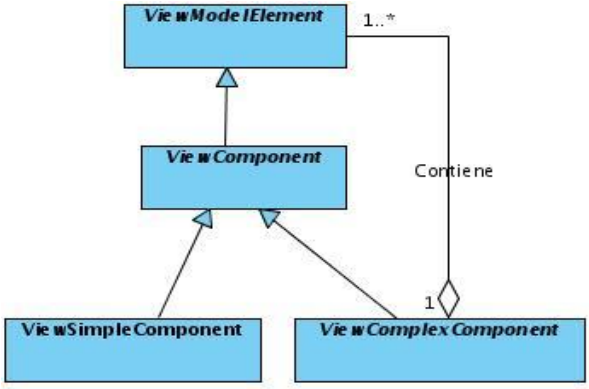


Figura 3.6

Todos los elementos del modelo van a ser ViewModelElement, pero existen los componentes simples (ViewSimpleComponent) como es el caso de los químicos, receptores y canales de salida, los componentes compuestos (ViewComplexComponent) como es el caso de las células y los modelos de componentes que se utilizarán para agrupar componentes, tanto simples como compuestos. En la figura 3.7 parte de un modelo gráfico de un SB donde se aprecia la utilidad del patrón descrito en esta sección.

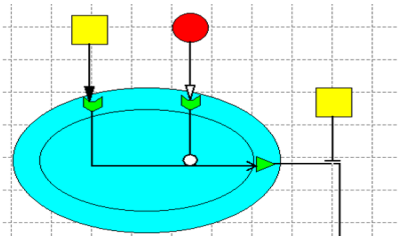


Figura 3.7

3.3.4 PATRÓN ABSTRACT FACTORY

El propósito de este patrón es proporcionar una interfaz para crear familias de objetos relacionados sin especificar sus clases concretas.

Consiste en definir una clase abstracta (AbstractFactory) que declare una interfaz para crear toda una familia de productos definidos también en clases abstractas y programar toda la lógica de la aplicación interactuando con estas clases abstractas (ver figura 3.8).

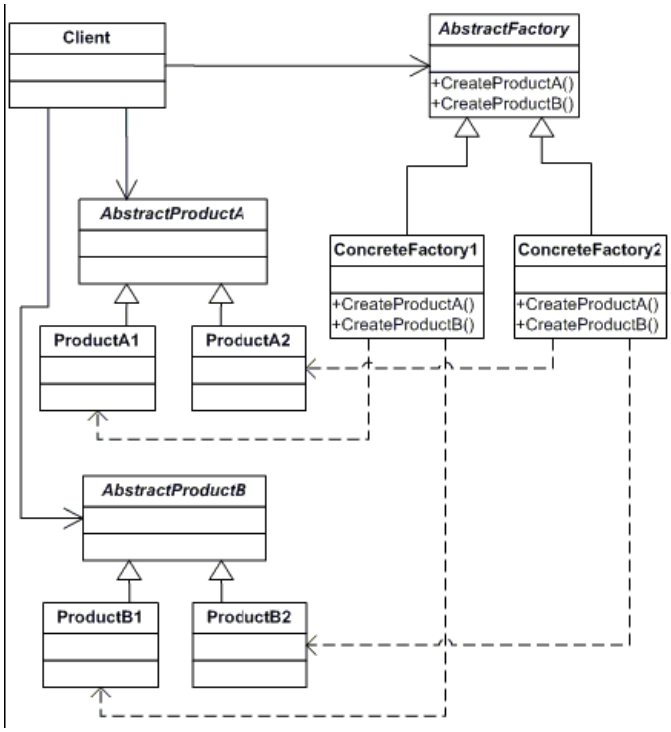


Figura 3.8

Cuando la lógica necesite nuevos productos no tiene más que utilizar esta interfaz para su creación, sin necesidad de saber exactamente qué ConcreteFactory está utilizando ni qué ConcreteProducts está recibiendo.

La ventaja fundamental que aporta este patrón es que facilita la inclusión de nuevos tipos de componentes otorgándole un elevado grado de extensibilidad a la herramienta.

Como ejemplo se mostrarán las factorías dedicadas a la creación de nuevos componentes biológicos, (ver figura 3.9) en el caso de la creación de nuevos procesos biológicos el funcionamiento es análogo.

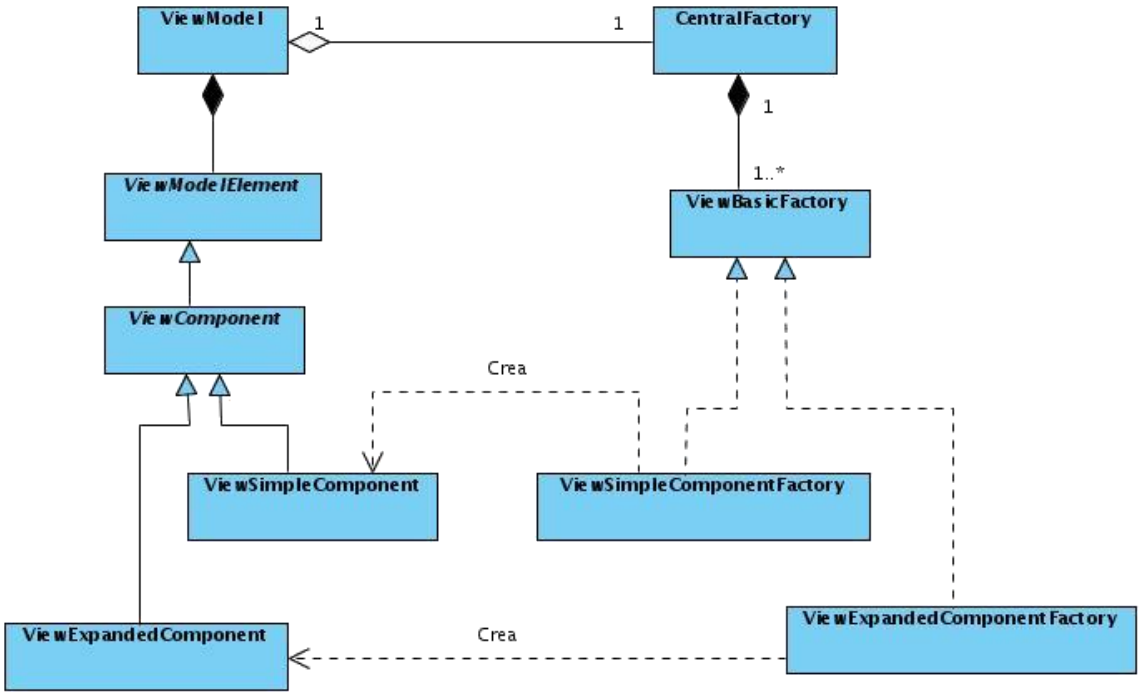


Figura 3.9

La clase ViewModel representa el modelo gráfico del sistema biológico, dicha clase contiene una referencia a la factoría central de componentes (CentralFactory), a la cual le hacen la solicitud de nuevos componentes y a través de ViewBasicFactory ella les proporciona los componentes solicitados.

Para agregar un nuevo tipo de componente a la herramienta es muy sencillo, solo crear una clase encargada de pintarlo y ponerla a implementar la interfaz DrawComponent y luego crear una factoría para dicho componente e implementar la interfaz ViewBasicFactory, de esta forma ya la herramienta sería capaz de modelar utilizando un nuevo tipo de componentes.

La interfaz ViewBasicFactory presenta los siguientes métodos que deben ser implementados:

Devuelve el nombre del tipo de elemento capaz de crear la factoría

public String getName();

Devuelve un elemento pasado el elemento origen, las coordenadas del origen, el elemento destino y las coordenadas destino.

public ViewModelElement getModelElement(ViewModelElement source, Point po, ViewModelElement target, Point pd);

Devuelve un elemento pasado su contenedor y el punto del eje de coordenadas donde será pintado.

public ViewModelElement getModelElement(ViewModelElement source, Point po);

Crea el componente a partir de un XML, se utiliza para el caso de leer la factoría después de almacenada en un XML.

public ViewModelElement getModelElement(String id, String type, Element element, ArrayList<ViewModelElement> source);

Devuelve si el tipo de elemento pasado por parámetro (type) puede ser origen del elementoa crear

public boolean isSource(String type);

Salva la factoría en un XML

public Element saveToXML();

3.3.5 PATRÓN FACADE

El patrón de diseño fachada (Facade) sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas.

El objetivo principal de uso de este patrón es ocultar todo lo posible la complejidad de un sistema o el conjunto de clases o componentes que lo forman, de forma que sólo se ofrezca un (o unos pocos) punto de entrada al sistema tapado por la fachada.

Se utilizó dicho patrón en el diseño debido a que, al utilizar el estilo arquitectónico modelo vista controlador para cada entidad de la vista existe una entidad en el modelo y mientras el usuario se encuentra modelando gráficamente, a través de un inspector de propiedades (figura 3.10) puede modificar tantos las propiedades de la vista como las del modelo. Debido a esto se creo una fachada que permite al usuario modificar las propiedades tanto del objeto vista como del correspondiente objeto del modelo. En la figura 3.11 se muestra el diagrama de clases correspondiente.

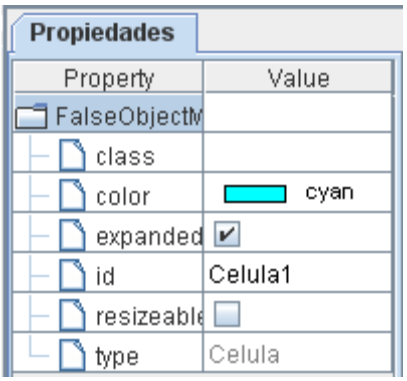


Figura 3.10

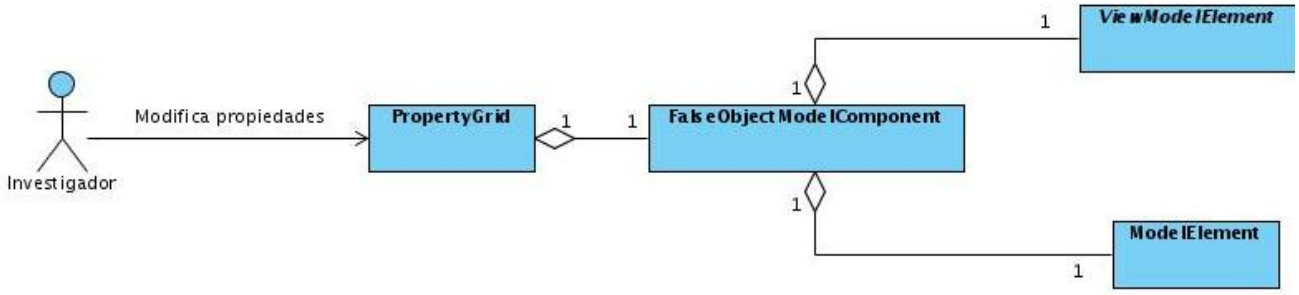


Figura 3.11

3.4 DIAGRAMA DE CLASES DEL DISEÑO

Un diagrama de clases de diseño muestra las especificaciones para las clases software de una aplicación. Incluye la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Navegabilidad.
- Dependencias.

Este tipo de diagramas muestra definiciones de entidades software más que conceptos del mundo real y se elabora para tener en cuenta los detalles concretos de la implementación del sistema.

El los anexos 3,4 y 5 se encuentran los diagramas de clases del diseño de las tres capas, modelo, vista y controlador respectivamente.

3.5 ASPECTO DEL DISEÑO A DESTACAR

Existen varios aspectos que debe caracterizar el diseño de un software:

- Correcto: Conduce a una puesta en práctica satisfactoria.
- Adaptable: Facilita el cambio (el cambio es inevitable).
- Eficiente: Gastar solo los recursos necesarios.
- Simple: Ser tan comprensible como sea posible.

El diseño realizado de la herramienta cumple con las características anteriormente citadas, pero se hará referencia a continuación a su facilidad de cambio por la importancia que para el presente trabajo tiene.

La modelación gráfica ha sido una de las principales vías empleada por el hombre a lo largo de los años para comprender el funcionamiento de sistemas. En la actualidad el hombre modela un conjunto de procesos que son de su interés, procesos químicos, biológicos, industriales, sociales, etc. Por tal razón se

desarrolló el diseño de la herramienta enfocándolo en convertirla en un framework para el desarrollo de aplicaciones capaces de modelar gráficamente procesos.

El objetivo anterior fue cumplido gracias al uso adecuado de los patrones de diseño descritos en secciones anteriores, lo que aporta un elevado grado de extensibilidad a la herramienta.

3.6 CONCLUSIONES

En el presente capítulo fue descrito el estilo arquitectónico que se utilizó en el diseño de la herramienta así como los patrones de diseño utilizados y la fundamentación de su uso. También fue desarrollado el diagrama de clases del diseño.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1 INTRODUCCIÓN

A través de la realización del presente capítulo se describirá como fue implementada la herramienta en términos de componentes. Se desarrollaran varias pruebas de aceptación usando el método de caja negra y la técnica de particiones de equivalencia para garantizar que se han cumplido los requisitos funcionales.

4.2 IMPLEMENTACIÓN

En la implementación se comienza con el resultado del diseño y se implementa el sistema en término de componentes, es decir ficheros de código fuente, librerías, ejecutables, scripts, fichero de código binario y similares (8).

En el flujo de trabajo de diseño se propone crear un plano del modelo de implementación. El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue.

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes software, sean estos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. En la figura 4.1 se muestra el diagrama de componentes de la herramienta desarrollada.

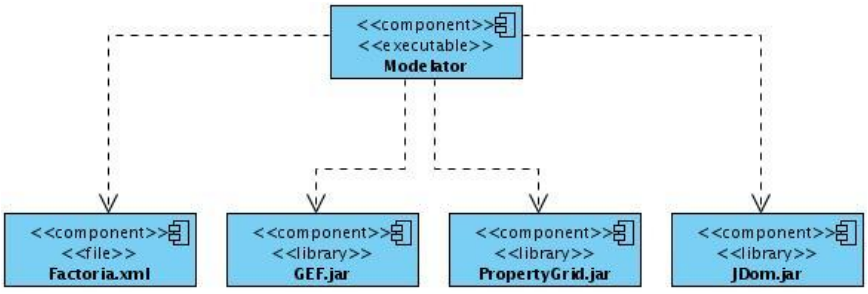


Figura 4.1

4.3 MODELO DE PRUEBA

En el desarrollo de software se requiere gran esfuerzo mental por parte de los desarrolladores, por lo que la posibilidad de cometer errores es muy alta. No se puede asegurar que un producto es ciento por ciento fiable ni que cumplirán al máximo con las expectativas de los clientes. El método con el que se cuenta para garantizar la calidad y el buen funcionamiento de un producto es la realización de pruebas. Las pruebas no confirman la ausencia de errores en nuestro software, solo brindan una medida de cómo responderá el mismo ante algunas situaciones determinadas (11).

A la herramienta desarrollada se le aplicaron pruebas de aceptación con el objetivo de verificar las funcionalidades del sistema. Para esto se utilizo el método de caja negra usando la técnica de partición de equivalencia. A continuación se muestran los casos de prueba desarrollados.

Caso de Uso	Insertar compartimiento
Caso de Prueba	Insertar Célula
Entrada	Componente a crear: Célula1 Componente destino: Modelo_BioSyS Tipo del componente destino: Modelo
Resultado Esperado	Se agrega una instancia del componente Célula a Modelo_BioSyS.
Resultado de la prueba	Al componente Modelo_BioSyS se le agrega una nueva instancia del componente Célula con el nombre Célula1. (Figura 4.1)
Condiciones	Debe existir en el modelo gráfico del Sistema Biológico modelado el componente Modelo_BioSyS y ser un componente del tipo Modelo.

Caso de Uso	Insertar compartimiento
Caso de Prueba	Insertar Célula
Entrada	Componente a crear: Célula1 Componente destino: Citoquina_B Tipo del componente destino: Citoquina
Resultado Esperado	Mostrar mensaje de error al usuario.
Resultado de la prueba	Se muestra un mensaje de error explicando al usuario que Citoquina_B no puede ser un componente fuente de un componente del tipo Célula. (Figura 4.2)
Condiciones	Debe existir en el modelo gráfico del Sistema Biológico modelado el componente Citoquina_B y ser un componente del tipo Citoquina.

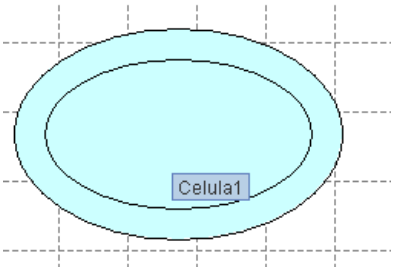


Figura 4.1

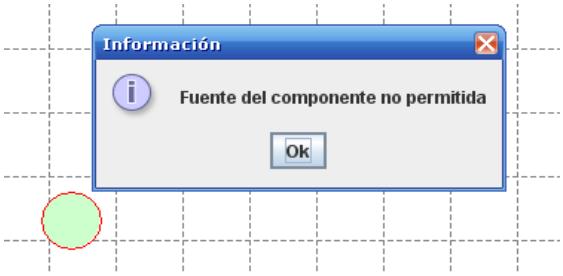


Figura 4.2

Caso de Uso	Insertar compartimiento
Caso de Prueba	Agrupar componentes
Entrada	Componente a crear: Grupo_de_componentes1 Componentes a agrupar: Citoquina1, Citoquina2, Citoquina3, And1, Transport1, Transport2 y Transport3.
Resultado Esperado	Agrupar los componentes Citoquina1, Citoquina2, Citoquina3, And1, Transport1, Transport2 y Transport3.
Resultado de la prueba	Se crea un grupo de componentes llamado Grupo_de_componentes1 agrupando los componentes Citoquina1, Citoquina2, Citoquina3, And1, Transport1, Transport2 y Transport3. (Figura 4.3)
Condiciones	Debe existir en el modelo gráfico del Sistema Biológico modelado los componentes Citoquina1, Citoquina2, Citoquina3, And1, Transport1, Transport2 y Transport3.

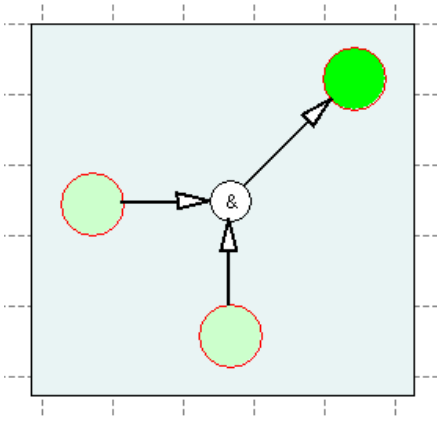


Figura 4.3

Caso de Uso	Insertar Químico
Caso de Prueba	Insertar Citoquina
Entrada	Componente a crear: Citoquina1 Componente destino: Modelo_BioSyS Tipo del componente destino: Modelo
Resultado Esperado	Agregar un nuevo químico con el nombre de Citoquina1 al componente Modelo_BioSyS.
Resultado de la prueba	Al componente Modelo_BioSyS se le agrega una nueva instancia del componente Citoquina con el nombre Citoquina1. (Figura 4.4)
Condiciones	Debe existir en el modelo gráfico del Sistema Biológico modelado el componente Modelo_BioSyS y ser un componente del tipo Modelo.

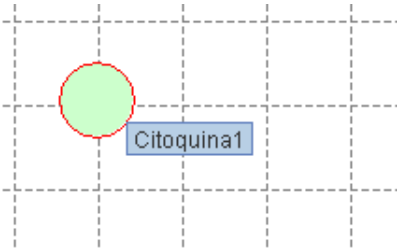


Figura 4.4

Caso de Uso	Insertar Químico
Caso de Prueba	Insertar Receptor
Entrada	Componente a crear: Receptor Componente destino: Célula_B Tipo del componente destino: Célula
Resultado Esperado	Agregar un nuevo químico con el nombre de Receptor al componente Célula_B.
Resultado de la prueba	Al componente Célula_B se le agrega una nueva instancia del componente Receptor con el nombre Receptor. (Figura 4.5)
Condiciones	Debe existir en el modelo gráfico del Sistema Biológico modelado el componente Célula_B y ser un componente del tipo Célula.

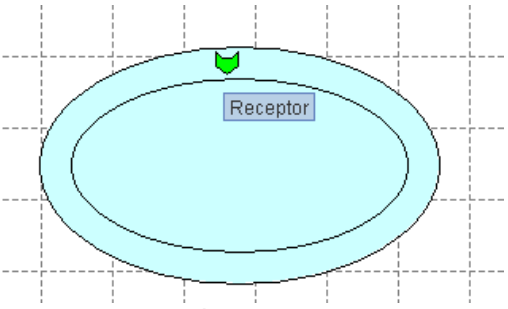


Figura 4.5

Caso de Uso	Insertar Químico
Caso de Prueba	Insertar Receptor
Entrada	Componente a crear: Receptor Componente destino: Modelo_BioSyS Tipo del componente destino: Modelo
Resultado Esperado	Mostrar mensaje de error al usuario.
Resultado de la prueba	Se muestra un mensaje de error explicando al usuario que Modelo_BioSyS no puede ser un componente fuente de un componente del tipo Receptor. (Figura 4.6)
Condiciones	Debe existir en el modelo gráfico del Sistema Biológico modelado el componente Modelo_BioSyS y ser un componente del tipo Modelo.

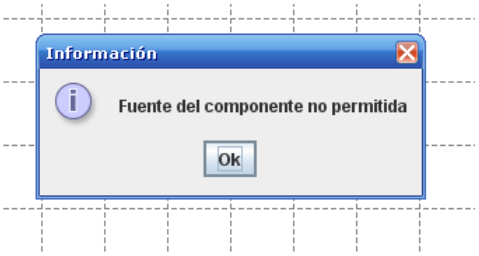


Figura 4.6

Caso de Uso	Insertar Proceso
Caso de Prueba	Insertar Diferenciación
Entrada	Componente a crear: Diferenciación1 Componente fuente: Célula_A Componente destino: Célula_B Tipo del componente fuente: Célula Tipo del componente destino: Célula
Resultado Esperado	Agregar proceso de diferenciación con fuente Célula_A y destino Célula_B.
Resultado de la prueba	En el modelo gráfico del Sistema Biológico modelado se agrega un nuevo proceso de diferenciación llamado Diferenciacion1 que tiene como fuente a Célula_A y destino Célula_B. (Figura 4.7)
Condiciones	Deben existir en el modelo gráfico del Sistema Biológico modelado los componentes Célula_A y Célula_B, ambos deben ser componentes de tipo Célula.

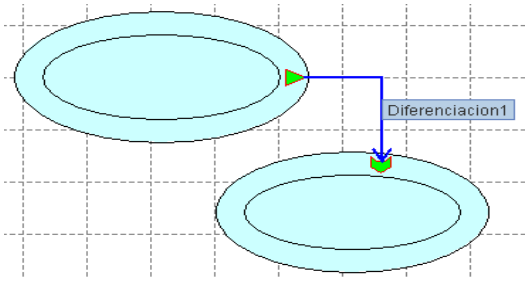


Figura 4.7

Caso de Uso	Insertar Proceso
Caso de Prueba	Insertar Diferenciación
Entrada	Componente a crear: Diferenciación1 Componente fuente: Célula_A Componente destino: Transport_A Tipo del componente fuente: Célula Tipo del componente destino: Transport
Resultado Esperado	Mostrar mensaje de error al usuario.
Resultado de la prueba	Se muestra un mensaje de error explicando al usuario que Transport_A no puede ser un componente destino del proceso. (Figura 4.8)
Condiciones	Deben existir en el modelo gráfico del Sistema Biológico modelado los componentes Célula_A del tipo célula y Transport_A del tipo Transport.

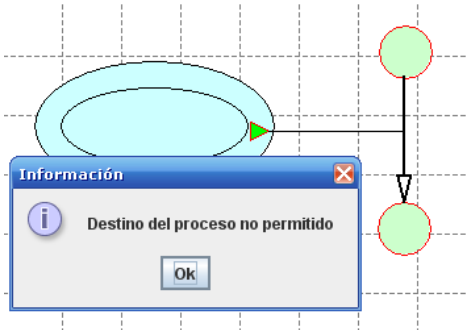


Figura 4.8

Caso de Uso	Insertar Proceso
Caso de Prueba	Insertar Diferenciación
Entrada	Componente a crear: Diferenciación1 Componente fuente: Transport_A Componente destino: Célula_A Tipo del componente fuente: Transport Tipo del componente destino: Célula
Resultado Esperado	Mostrar mensaje de error al usuario.
Resultado de la prueba	Se muestra un mensaje de error explicando al usuario que Transport_A no puede ser un componente origen del proceso. (Figura 4.9)
Condiciones	Deben existir en el modelo gráfico del Sistema Biológico modelado los componentes Célula_A del tipo célula y Transport_A del tipo Transport.

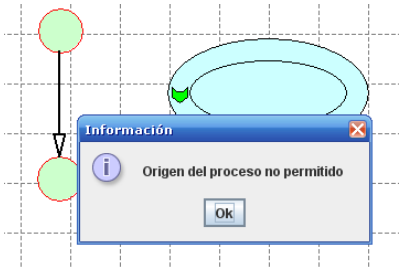


Figura 4.9

Caso de Uso	Insertar Señal
Caso de Prueba	Insertar Señal
Entrada	Componente a crear: Señal1 Componente fuente: Receptor1 Componente destino: Canal_de_salida1 Tipo del componente fuente: Receptor Tipo del componente destino: CanalDeSalida
Resultado Esperado	Agregar una señal con fuente Receptor1 y destino Canal_de_salida1.
Resultado de la prueba	En el modelo gráfico del Sistema Biológico modelado se agrega una nueva señal llamada Señal1 que tiene como fuente a Receptor1 y destino Canal_de_salida1. (Figura 4.10)
Condiciones	Deben existir en el modelo gráfico del Sistema Biológico modelado los componentes Receptor1 del tipo Receptor y Canal_de_salida1 del tipo CanalDeSalida pertenecientes a la misma célula.

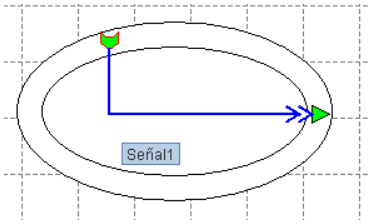


Figura 4.10

Caso de Uso	Insertar Señal
Caso de Prueba	Insertar Señal
Entrada	Componente a crear: Señal1 Componente fuente: Receptor1 Componente destino: Canal_de_salida1 Tipo del componente fuente: Receptor Tipo del componente destino: CanalDeSalida
Resultado Esperado	Mostrar mensaje de error al usuario.
Resultado de la prueba	Se muestra un mensaje de error explicando al usuario que las señales son procesos de una misma célula. (Figura 4.11)
Condiciones	Deben existir en el modelo gráfico del Sistema Biológico modelado los componentes Receptor1 del tipo Receptor y Canal_de_salida1 del tipo CanalDeSalida pertenecientes dos células distintas.

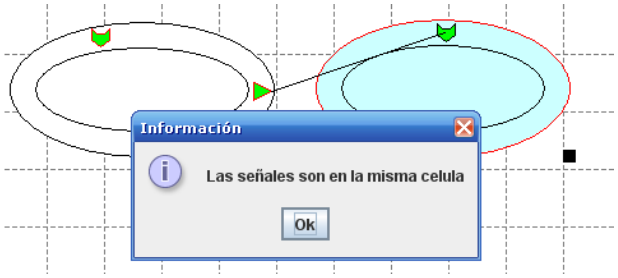


Figura 4.11

Caso de Uso	Modificar componente
Caso de Prueba	Modificar Célula
Entrada	Componente a modificar: Célula1 Propiedades a modificar: Identificador, Color Identificador: Antiguo = Célula1, Nuevo = CélulaB Color: Antiguo = Azul, Nuevo = Amarillo
Resultado Esperado	Cambiar las propiedades color e identificador de Célula1.
Resultado de la prueba	Al componente Célula1 se le cambia el identificador por CélulaB y el color por Rojo. (Figura 4.12).
Condiciones	No puede existir en el modelo gráfico del Sistema Biológico modelado un componente con el identificador CélulaB.

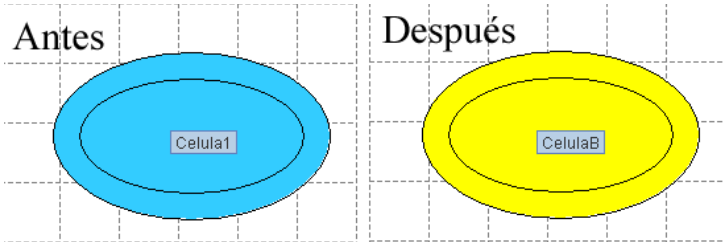


Figura 4.12

Caso de Uso	Modificar componente
Caso de Prueba	Modificar Célula
Entrada	Componente a modificar: Célula1 Propiedades a modificar: Identificador Identificador: Antiguo = Célula1, Nuevo = CélulaB
Resultado Esperado	Mostrar mensaje de error al usuario.
Resultado de la prueba	Se muestra un mensaje de error explicando al usuario que hay otro componente con el identificador CélulaB. (Figura 4.14)
Condiciones	Debe existir en el modelo gráfico del Sistema Biológico modelado un componente con el identificador CélulaB.

Figura 4.14

Caso de Uso	Eliminar proceso
Caso de Prueba	Eliminar Transport
Entrada	Componente a eliminar: Transport1 Componentes relacionados: Catalisis1, Citoquina1,Citoquina2
Resultado Esperado	Eliminar Catalisis1 y Transport1 del modelo.
Resultado de la prueba	Del modelo gráfico del Sistema Biológico modelado se eliminan los procesos Catalisis1 y Transport1. (Figura 4.15)
Condiciones	Deben existir en el modelo gráfico del Sistema Biológico modelado los componentes Catalisis1 y Transport1, el segundo componente debe ser el proceso catalizado por el primero.

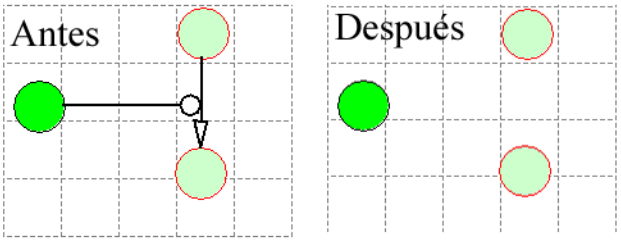


Figura 4.15

Caso de Uso	Eliminar Señal
Caso de Prueba	Eliminar Señal
Entrada	Componente a eliminar: Señal1 Componentes relacionados: Receptor1, CanalDeSalida1.
Resultado Esperado	Eliminar Señal1 del modelo.
Resultado de la prueba	Del modelo gráfico del Sistema Biológico modelado se elimina Señal1. (Figura 4.16)
Condiciones	Deben existir en el modelo gráfico del Sistema Biológico modelado los componentes Receptor1, CanalDeSalida1 y Señal1.

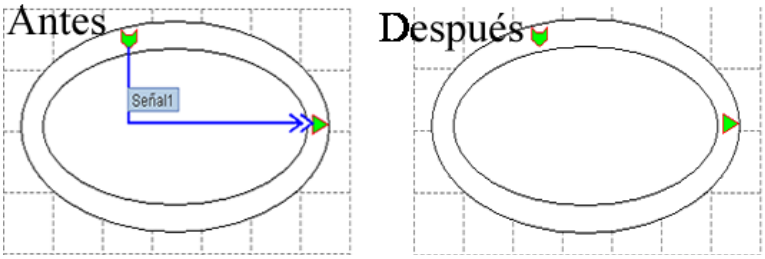


Figura 4.16

Caso de Uso	Eliminar Químico
Caso de Prueba	Eliminar Canal de Salida
Entrada	Componente a eliminar: CanalDeSalida1 Procesos relacionados: Señal1, Catalisis1.
Resultado Esperado	Eliminar del modelo CanalDeSalida1 y los procesos relacionados.
Resultado de la prueba	Del modelo gráfico del Sistema Biológico modelado se eliminan los componentes CanalDeSalida1, Señal1 y Catalisis1. (Figura 4.17)
Condiciones	Deben existir en el modelo gráfico del Sistema Biológico modelado los componentes CanalDeSalida1, Señal1 y Catalisis1, donde CanalDeSalida1 sea el destino de Señal1 y el Catalisis1 se encuentre catalizando a Señal1.

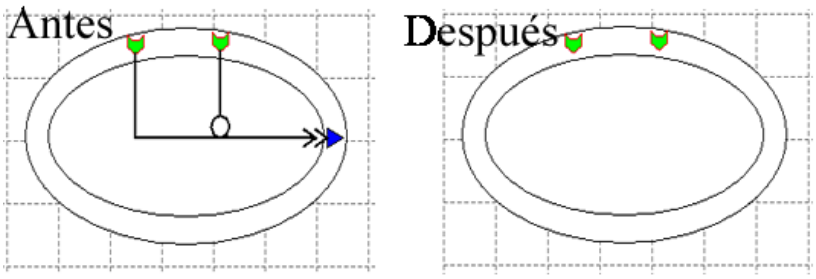


Figura 4.17

Caso de Uso	Eliminar Químico
Caso de Prueba	Eliminar Citoquina
Entrada	Componente a eliminar: Citoquina Componentes relacionados: Citoquina1 y Citoquina2. Procesos relacionados: Transport1, Catalisis1.
Resultado Esperado	Eliminar Citoquina y procesos relacionados.
Resultado de la prueba	Del modelo gráfico del Sistema Biológico modelado se eliminan los componentes Citoquina, Transport1, Catalisis1. (Figura 4.18)
Condiciones	Deben existir en el modelo gráfico del Sistema Biológico modelado los componentes Citoquina, Citoquina1, Citoquina2, Transport1 y Catalisis1, Transport1 como proceso que relaciona a Citoquina y Citoquina2, Catalisis1 como proceso que cataliza a Transport1.

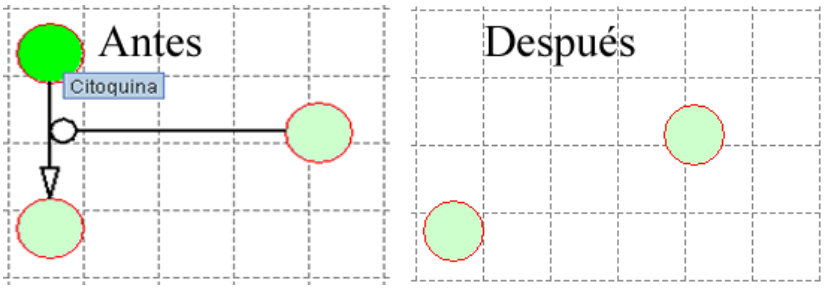


Figura 4.18

Caso de Uso	Eliminar compartimiento
Caso de Prueba	Eliminar célula
Entrada	Componente a eliminar: CélulaA. Componentes internos del componente a eliminar: Receptor1, CanalDeSalida1 y Señal1. Procesos asociados: Diferenciacion1.
Resultado Esperado	Eliminar CélulaA, sus componentes internos y procesos asociados.
Resultado de la prueba	Del modelo gráfico del Sistema Biológico modelado se eliminan los componentes CélulaA, Receptor1, CanalDeSalida1, Señal1 y Diferenciacion1. (Figura 4.19)
Condiciones	Deben existir en el modelo gráfico del Sistema Biológico modelado los componentes CélulaA, Receptor1, CanalDeSalida1, Señal1 y Diferenciacion1. Receptor1, CanalDeSalida1, Señal1 como componentes internos a CélulaA y Diferenciacion1 como un proceso que relaciona a un químico con CélulaA.

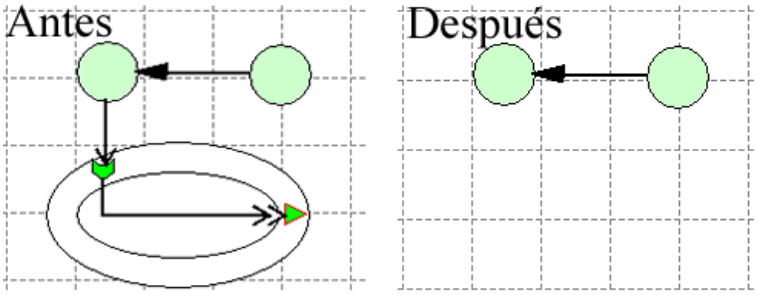


Figura 4.19

Caso de Uso	Eliminar Compartimiento
Caso de Prueba	Eliminar Grupo de componentes
Entrada	Componente a eliminar: GrupoA. Componentes internos del componente a eliminar: Citoquina1, Citoquina2, Citoquina3, Transport1. Procesos asociados: Catalisis1.
Resultado Esperado	Eliminar CélulaA, sus componentes internos y procesos asociados.
Resultado de la prueba	Del modelo gráfico del Sistema Biológico modelado se eliminan los componentes GrupoA, Citoquina1, Citoquina2, Citoquina3, Transport1 y Catalisis1. (Figura 4.20)
Condiciones	Deben existir en el modelo gráfico del Sistema Biológico modelado los componentes GrupoA, Citoquina1, Citoquina2, Citoquina3, Transport1 y Catalisis1. Catalisis1 debe encontrarse catalizando el proceso Transport1. Los componentes Citoquina1, Citoquina2, Citoquina3 y Transport1 deben pertenecer al grupo de componentes GrupoA.

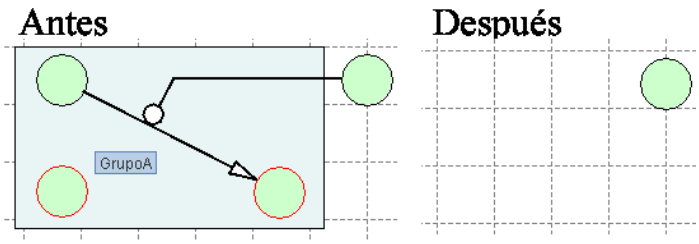


Figura 4.20

Caso de Uso	Crear Componente Reutilizable
Caso de Prueba	Crear Componente Reutilizable
Entrada	Componente a crear: Modelo_BioSyS
Resultado Esperado	Debe agregarse a la paleta de componentes un nuevo componente llamado Modelo_BioSyS que podrá ser utilizado para modelar Sistemas Biológicos.
Resultado de la prueba	Se agrega a la paleta de componentes un nuevo componente llamado Modelo_BioSyS que puede ser utilizado para modelar Sistemas Biológicos. (Figura 4.21)
Condiciones	El usuario debe haber seleccionado del modelo gráfico del Sistema Biológico modelado al menos un componente que formará parte del submodelo que se podrá crear a partir del componente reutilizable Modelo_BioSyS.

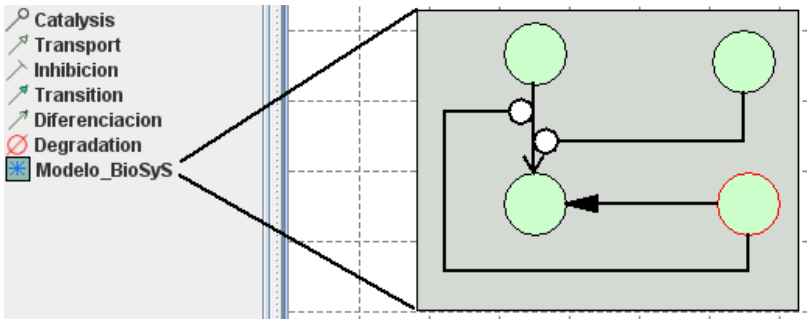


Figura 4.21

Caso de Uso	Guardar modelo del Sistema Biológico
Caso de Prueba	Guardar modelo del Sistema Biológico
Entrada	Modelo a guardar: Modelo_BioSyS. Dirección: /home/modelo Nombre: modelo.xml
Resultado Esperado	Crear un fichero en formato xml en la dirección /home/modelo con el nombre modelo.xml donde es almacenado Modelo_BioSyS.
Resultado de la prueba	Se crea un fichero en formato xml en la dirección /home/modelo con el nombre modelo.xml donde es almacenan los datos Modelo_BioSyS para posteriores estudios.
Condiciones	La dirección /home/modelo debe ser válida. El usuario debe tener permiso de escritura en /home/modelo.

Caso de Uso	Cargar modelo del Sistema Biológico
Caso de Prueba	Cargar modelo del Sistema Biológico
Entrada	Dirección del modelo: /home/modelo/modelo.xml
Resultado Esperado	Cargar el modelo almacenado en modelo.xml en el panel de modelación.
Resultado de la prueba	Se le agrega al panel de modelación una nueva pizarra en la cual se carga el modelo almacenado en modelo.xml.
Condiciones	La dirección /home/modelo/modelo.xml debe ser valida y el usuario debe tener permiso de lectura en la misma.

4.4 CONCLUSIONES

Mediante la realización del presente capítulo se describió como fue implementada la herramienta en términos de componentes. Se desarrollaron varias pruebas de aceptación usando el método de caja negra y la técnica de particiones de equivalencia para garantizar que se han cumplido los requisitos funcionales.

CONCLUSIONES

- Se realizó un estudio sobre la Biología de Sistemas que permitió conocer aspectos importantes para el desarrollo de una herramienta que permita modelar gráficamente Sistemas Biológicos.
- Se definieron los requisitos para el desarrollo de una herramienta que permita modelar gráficamente Sistemas Biológicos.
- Se desarrolló una herramienta para la modelación gráfica de Sistemas Biológicos que puedan ser descritos mediante dinámica de población.

RECOMENDACIONES

- Añadirle a la herramienta desarrollada módulos que permitan simular y analizar los modelos gráficos de los Sistemas Biológicos elaborados.
- Añadirle a la herramienta un módulo que permita editar el sistema de ecuaciones diferenciales asociado a los modelos gráficos de los Sistemas Biológicos elaborados.
- Incorporarle las funcionalidades de exportar los modelos gráficos y matemáticos asociados en los formatos internacionales SBML y MathML respectivamente.

BIBLIOGRAFÍA

1. Bioinformatic@.es Newsletter. [En línea] 06 de 03 de 2002. [Citado el: 06 de 06 de 2007.] http://www.webzinemaker.com/digitalbiology/.

2. *A graphical notation for biochimical networks.* **Kitano, H.** 5, 2003, Biosilico, Vol. 1.

3. *Interdisciplinary research and education at the biology–engineering–computer science interface: a perspective.* **Tadmor, Brigitta.** 17, 2005, Biosilico, Vol. 10.

4. **Osorio, Karel.** *Plataforma computacional para el desarrollo de la Biología de Sistema.* Facultad de Matemática Computación, Universidad de La Habana. Ciudad de la Habana : s.n., 2004.

5. **Torres, Nestor.** *Caos en Sistemas Biológicos.* Facultad de Biología, Universidad de Laguna. Tenerife, Islas Canarias : s.n., 2002.

6. BioUML. [En línea] [Citado el: 19 de mayo de 2007.] http://www.biouml.org.

7. *CellDesigner: a process diagram editor for gene-regulatory and biochemical networks.* **Funahashi, Akira.** 5, 2003, Biosilico, Vol. 1.

8. **Robles, Gregorio y Ferrer, Jorge.** *Programación eXtrema y Software Libre.* Universidad politécnica de Madrid. Madrid : s.n., 2002.

9. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* s.l. : PEARSON EDUCACIÓN S.A, 2000.

10. **Ramírez, Alejandro.** *Subversion.* 2004.

11. **Lago, Ramiro.** Patrones de diseño software. [En línea] Abril de 2007. [Citado el: 20 de Mayo de 2007.] http://www.proactiva-calidad.com/java/patrones/index.html#algunos_patrones.

12. **Pressman, Roger S.** *Ingeniería de Software. Un enfoque práctico.* s.l. : McGraw-Hill/Interamericana, 2002.

13. **Fonseca, Bernardo y Guerra, Yudiel.** *Sistema de servicio comunitario.* ISP José Antonio Echevería. Ciudad Habana : s.n., 2004. Tesis.

14. **Larman, Graig.** *Uml y patrones, introducción al análisis y diseño orientado a objetos.* México : Prentice Hall, 1999. 970-17-0261-1.

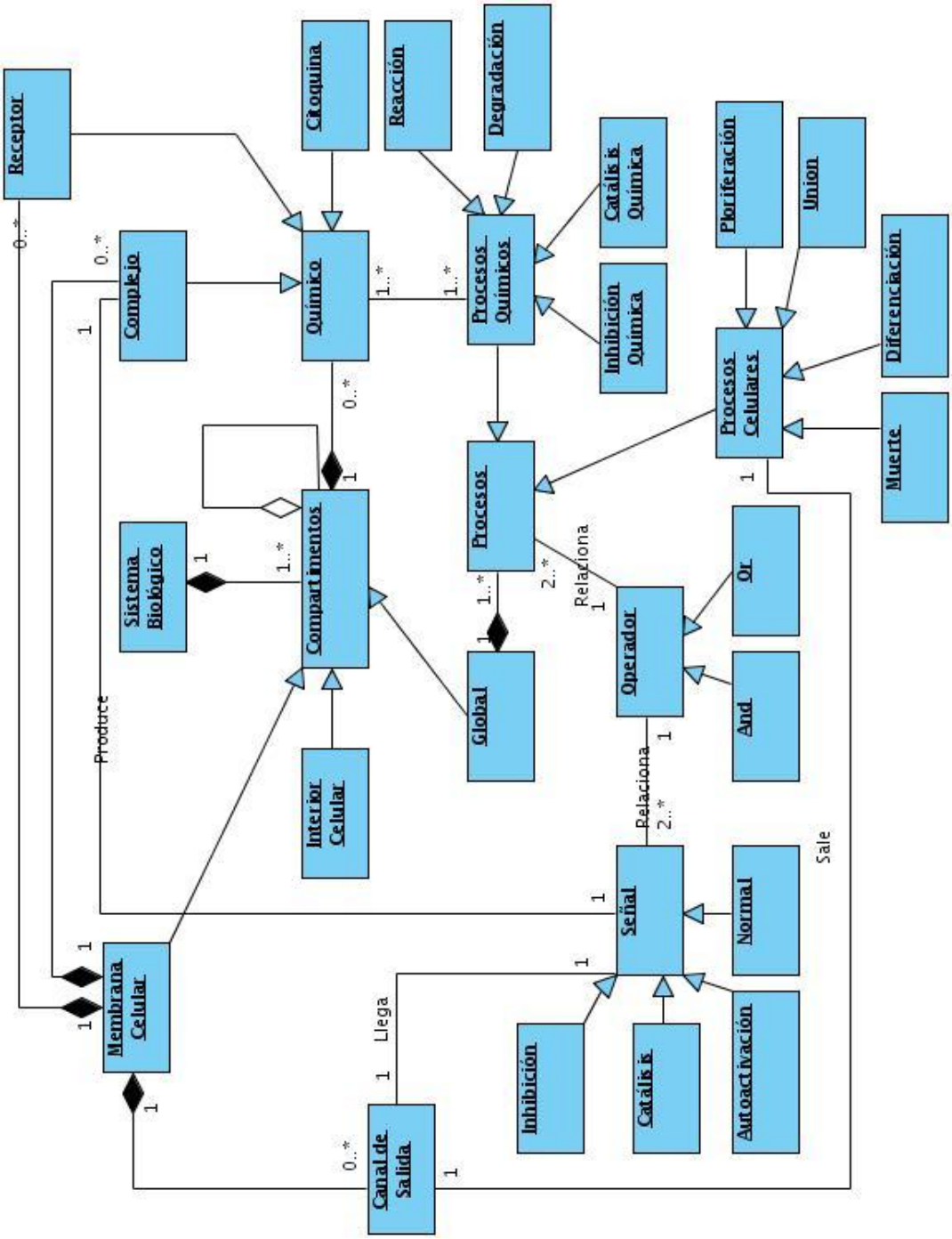
15. **Becerril, Francisco.** *Java a su alcance.* México : McGraw-Hill, 1998. 970-10-1774-9.

16. *Biological System Interactions.* **Adominan, G.** 2007, Vol. 81. 2938-2940.

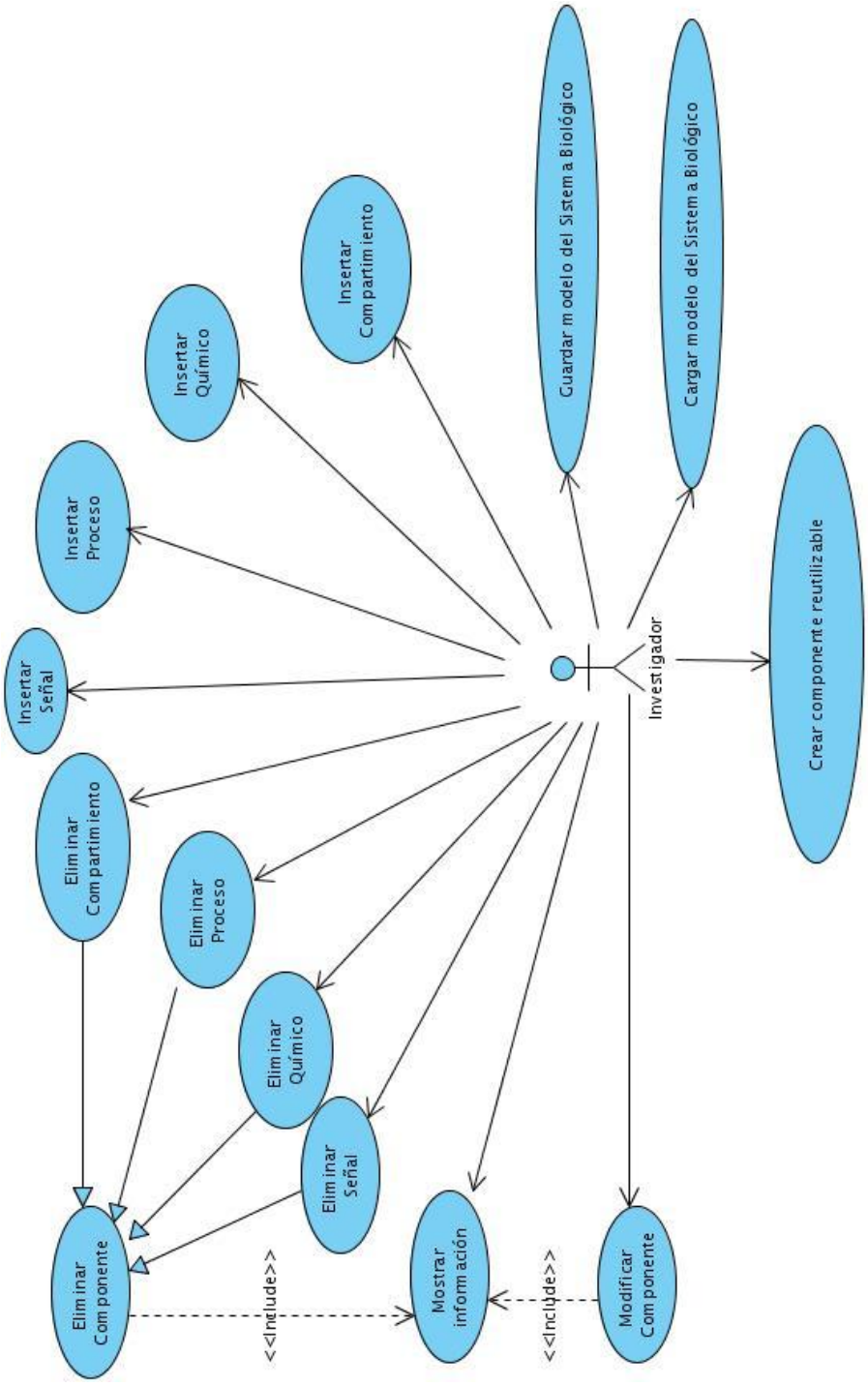
17. *Computational Systems Biology.* **Kitano, H.** 420, s.l. : Nature, 2002. 206-210.

ANEXOS

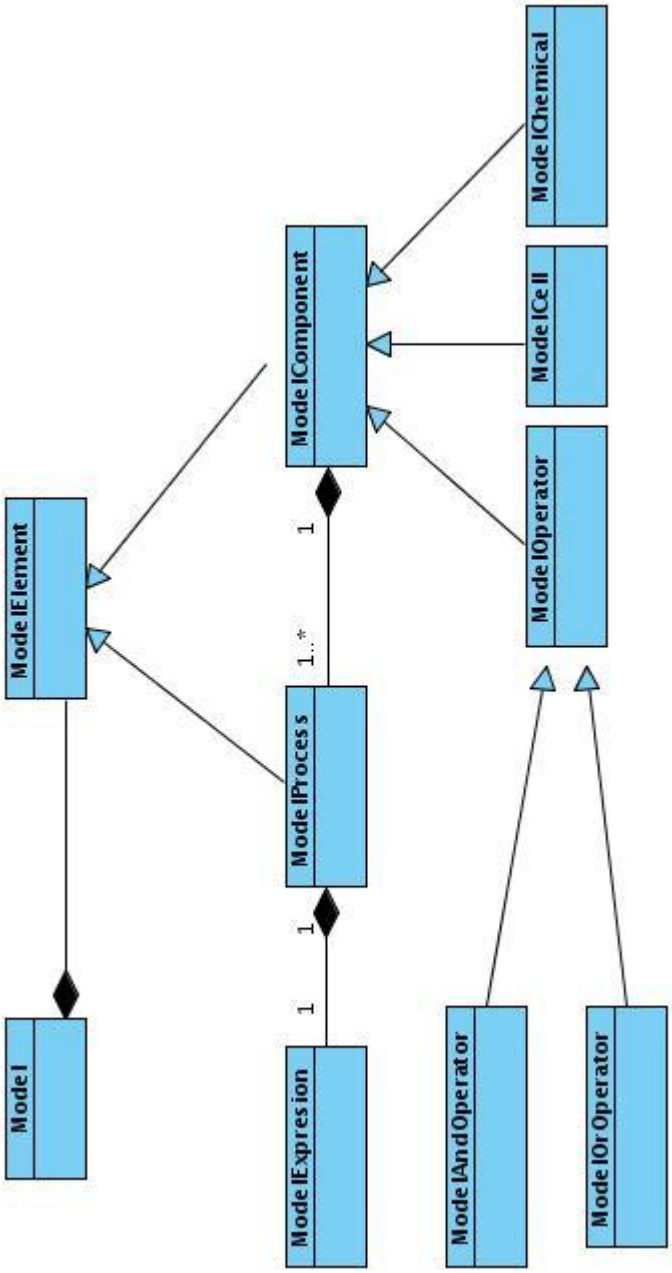
Anexo 1. Modelo de dominio.



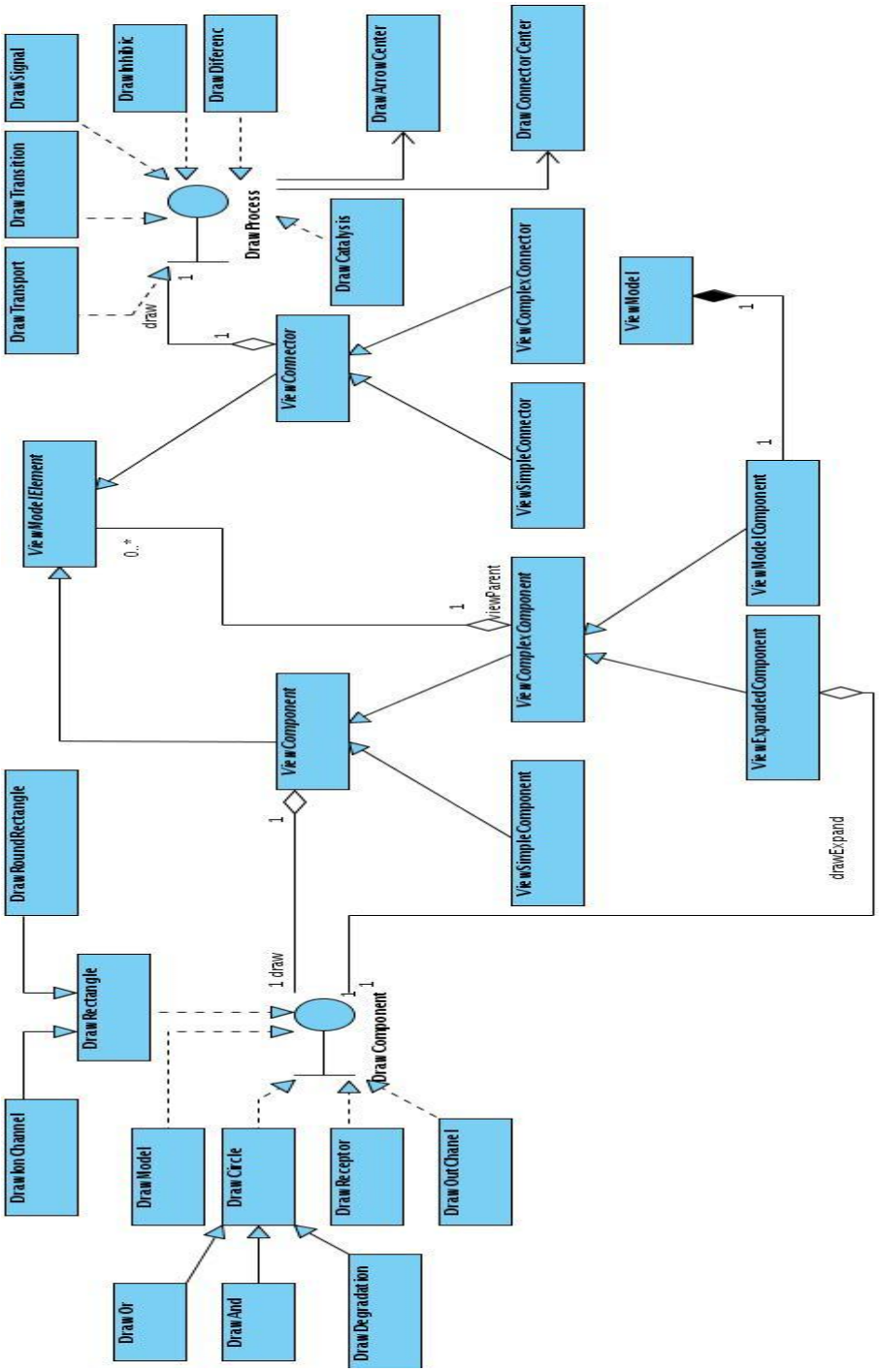
Anexo 2. Diagrama de casos de uso del sistema.



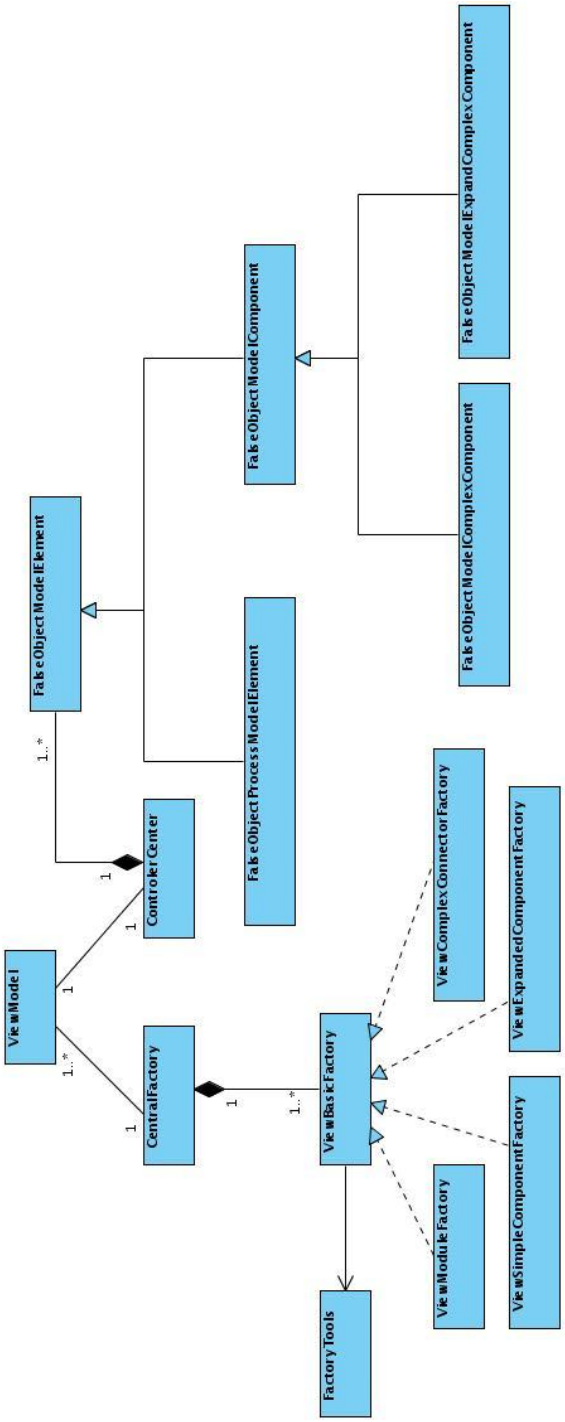
Anexo 3. Diagrama de clases perteneciente a la capa del modelo.



Anexo 4. Diagrama de clases perteneciente a la capa de las vistas.



Anexo 5. Diagrama de clases perteneciente a la capa controlador.



GLOSARIO

Bioinformática: Es la aplicación de los ordenadores y los métodos informáticos en el análisis de datos experimentales y simulación de los sistemas biológicos. Una de las principales aplicaciones de la Bioinformática es la simulación, la minería de datos y el análisis de los datos obtenidos en un estudio.

Biología de Sistemas: Área de investigación científica que se preocupa del estudio de procesos biológicos usando un enfoque sistémico.

Metodología: Define quién hace qué, cómo y cuando.

Modelación: Es un método de obtención del conocimiento, de aplicación en varias ciencias, en el cual se opera con un objeto, no en forma directa sino utilizando cierto sistema intermedio auxiliar conocido como modelo.

Modelo: Representación abstracta de la realidad. Diagramas que representan la estructura de un sistema dado.

Patrones: Unidad de información nombrada, instructiva e intuitiva que captura la esencia de una familia exitosa de soluciones probadas a un problema recurrente dentro de un cierto contexto.

Requisitos: Capacidades o condiciones que se deben cumplir.

Sistemas Biológicos: Sistemas abiertos que operan en condiciones alejadas del equilibrio termodinámico, con muchas y fuertes interacciones no lineales entre sus muchos elementos.

Software: Termino genérico que designa al conjunto de programas que posibilitan realizar una tarea específica en un ordenador.