Universidad de las Ciencias Informáticas

Facultad 6

Título: Modelador Gráfico de Sistemas Biológicos para alasBioSyS

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Silvio Sadoth López Henquén

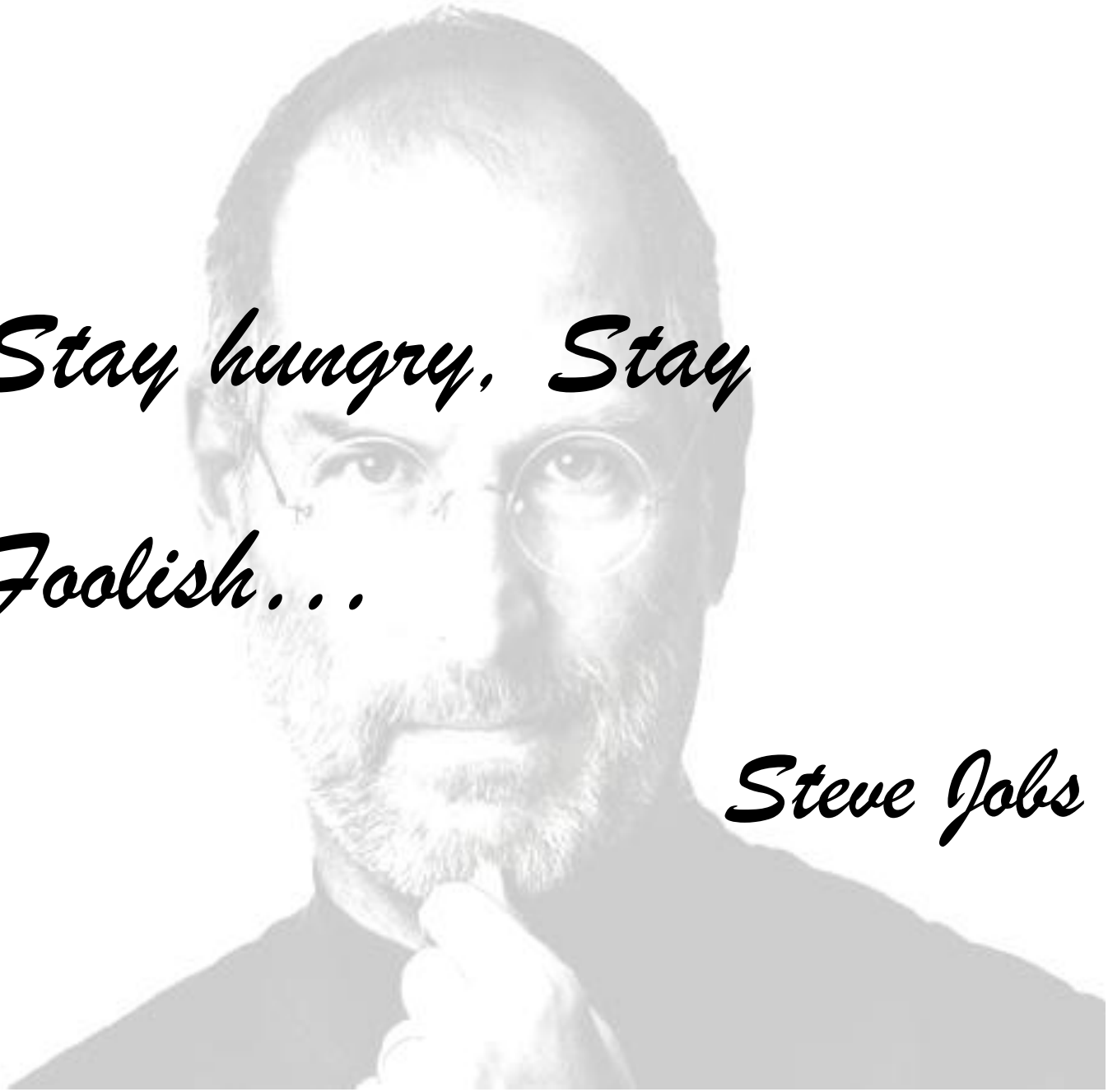
Héctor Alejandro Rodríguez Arias

Tutores: MSc. Yunet González Mulet

Ing. Carlos González Iglesias

La Habana, Junio de 2012

"Año 54 de la Revolución"



Stay hungry, Stay

Foolish...

Steve Jobs

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 _____**.**

Silvio Sadoth López Henquén

_____

Firma del Autor

Héctor Alejandro Rodríguez Arias

_____

Firma del Autor

MSc. Yunet González Mulet

_____

Firma del Tutor

Ing. Carlos González Iglesias

_____

Firma del Tutor

Datos de Contacto

Tutores:

MSc. Yunet González Mulet.
Universidad de las Ciencias Informáticas, Habana, Cuba.
Correo electrónico: ygonzalezmu@uci.cu

Ing. Carlos González Iglesias
Universidad de las Ciencias Informáticas, Habana, Cuba.
Correo electrónico: cgonzalez@uci.cu

Agradecimientos

A nuestros amigos de todos los momentos, por siempre estar con nosotros. Por su apoyo incondicional y su amistad sin límites. Por regalarnos todos esos momentos que nunca olvidaremos. A ustedes gracias.

A ese refugio que desde el día en que nacimos nos acogió. Compuesto por personas especialmente importantes en cada uno de los momentos de nuestra vida. Gracias por educarnos, hacernos fuertes, impulsarnos ante cada etapa de nuestra vida y nunca bajo ninguna circunstancia permitirnos dejar de creer en nosotros. A ustedes nuestra gran familia, gracias.

A dos personas increíbles que nos brindaron su conocimiento, su paciencia y su dedicación en todo momento. Que fueron ejemplo de modestia, profesionalidad y compromiso. Les agradecemos el habernos impulsado ante cada tarea, hacernos saber que podíamos hacerlo sin importar los obstáculos que pudieran existir, ayudarnos en todo momento cuando incluso su tiempo era limitado. Por ser independientemente de todas las cosas, nuestros amigos. A ustedes los mejores tutores que pudimos haber tenido, gracias.

A dos mujeres especiales que se colaron en nuestros corazones y hacen que nuestros días sean mas felices. A ustedes por acompañarnos, por soportarnos, por amarnos y por siempre tener la paciencia de escucharnos y también de aconsejarnos. Ustedes con su cálido amor y su infinita tolerancia, han hecho de nosotros mejores hombres. A ustedes Yuly y Mary, gracias.

Agradecimientos

Para ustedes no hallamos lenguaje universal ni conjunto de palabras que puedan reflejar este profundo agradecimiento, pero aun así trataremos. Primeramente por amarnos sin importar nada, por encaminarnos, por hacernos hombres de bien, por siempre creer en nosotros, por educarnos, por conducirnos ante las difíciles situaciones que hemos tenido en nuestras vidas, por confiar en nosotros y nunca permitirnos rendirnos aún cuando en algún momento pensamos que no podíamos. A ustedes que nos lo han dado todo y mucho más desde el día que hicieron posible que conociéramos este mundo; que han estado para nosotros en todo momento sea éste de paz o de tormenta, ustedes que aún cuando los vientos no han sido favorables nunca nos han mostrado el rostro de la derrota y nos han dado ejemplo de coraje y sacrificio, a ustedes que han sabido ser todo lo que hemos necesitado: amiga, amigo, padre y madre. Por ser la razón de nuestra existencia, a ustedes Argelia y María Rosa, gracias. Siempre trataremos de hacerlas sentir orgullosas, las amamos.

Dedicatoria

A nuestras familias, que siempre han esperado lo mejor de nosotros, a nuestras madres Argelia y María Rosa, a nuestros padres Héctor y Silvio.

A la Revolución Cubana, sin la cual no hubiera sido posible este acontecimiento, en especial a su máximo líder Fidel Castro Ruz.

A nuestros abuelos, que aunque hoy, nos es imposible su compañía y están más presentes que nunca.

Héctor y Silvio

Resumen

El proceso de representación es de vital importancia durante las investigaciones de los sistemas biológicos. Sin embargo ha existido la necesidad de compartir los resultados de dicho proceso sin depender de los idiomas o zonas geográficas. Por esta razón, con el paso de los años se han creado comunidades con el fin de desarrollar estándares universales para realizar dicha representación. Hoy día existen algunos software que incluyen esta cómoda funcionalidad y permiten la estandarización de los sistemas diseñados en ellos, así como su durabilidad en el tiempo.

El presente trabajo brinda a los usuarios del software alasBioSyS, un módulo para la modelación gráfica, el cual permite representar gráficamente los sistemas biológicos que son objeto de su investigación y lograr su estandarización bajo uno de los lenguajes de marcado más conocidos por la comunidad científica vinculada a la Biología de Sistemas, denominado Lenguaje de Marcado para Sistemas Biológicos (SBML por sus siglas en inglés). Este módulo posibilita además, la importación de modelos bajo el mismo lenguaje y la versión en que fueron desarrollados, así como el almacenamiento del modelo en formato de imagen y en el propio lenguaje de marcado.

Palabras clave: Biblioteca, estándares, modelación gráfica, sistemas biológicos, software.

Tabla de Contenidos

Índice de Figuras

Índice de Tablas

Introducción

Durante los últimos años del siglo XX, los adelantos científicos de la Ingeniería Genética y las nuevas tecnologías de la Informática, condicionaron el surgimiento de una disciplina que creó vínculos indisolubles entre la Informática y las ciencias biológicas: la Bioinformática.

El término Bioinformática es relativamente reciente, y apareció en la literatura a principios de 1990, cuando comenzaba a estructurarse el llamado "Proyecto Genoma Humano" y el Centro Nacional para la Información Biotecnológica (National Center for Biotechnology Information) de los Estados Unidos daba sus primeros pasos.[1]

La Bioinformática es un campo de la ciencia en el cual confluyen varias disciplinas tales como: Biología, Computación y las tecnologías de la información. El fin de este campo es facilitar el descubrimiento de nuevas ideas biológicas, así como crear perspectivas globales a partir de las cuales se puedan discernir principios unificadores en Biología. En sus inicios, el concepto de Bioinformática se refería sólo a la creación y mantenimiento de base de datos donde se almacenaba información biológica. El desarrollo de este tipo de base de datos no solamente significaba el diseño de la misma, sino también el desarrollo de interfaces complejas donde los investigadores pudieran acceder a los datos existentes y suministrar o revisar datos. Luego toda esa información debía ser combinada para formar una idea lógica de las actividades celulares normales, de tal manera que los investigadores pudieran estudiar cómo estas actividades se veían alteradas en estados de una enfermedad. De allí viene el surgimiento de la Bioinformática. En la actualidad el campo más recurrido por los especialistas es el análisis e interpretación de varios tipos de datos, incluyendo secuencias de nucleótidos y aminoácidos, dominios de proteínas y estructura de proteínas. [2]

El desarrollo de nuevos algoritmos estadísticos con los cuales se pueden relacionar partes de un conjunto enorme de datos, como por ejemplo métodos para localizar un gen dentro de una secuencia, predecir la estructura, función de proteínas y poder agrupar secuencias de proteínas en familias relacionadas; ha traído consigo la obtención de amplios conocimientos que han sido divididos en diferentes disciplinas para su mejor estudio. Una de las ramas surgidas producto del desarrollo de tecnologías de gran escala y alto rendimiento a nivel experimental y computacional es la Biología de Sistemas. [2]

La Biología de Sistemas es el campo de investigación interdisciplinario de los procesos biológicos que estudia los organismos como un todo o una entidad, y no como una colección de sus componentes. [3] Es una disciplina que precisa del enfoque pluridisciplinar de biólogos, físicos, químicos, matemáticos e informáticos y la interacción de grupos de investigación diversos en proyectos de gran ambición. Representa el paso hacia el conocimiento dinámico y cuantitativo de un proceso o sistema biológico con el objetivo de desarrollar modelos matemáticos que posibiliten la interacción experimental. Convencionalmente, cuando se realizan los estudios a los sistemas biológicos se utiliza el método científico clásico, basado en la aceptación o refutación de una hipótesis al realizarle la confrontación con los resultados experimentales. La Biología de Sistemas utiliza, sin embargo, un enfoque distinto, basado en la modelación matemática de los procesos en estudio.

El proceso de modelado de los sistemas biológicos no es sencillo, su complejidad está dada en el conocimiento previo del sistema y el nivel de capacitación que tenga el experto sobre el mismo. La modelación puede ser realizada de dos formas: gráfica y matemáticamente. Mediante el proceso de modelación gráfica se minimizan deficiencias concernientes al estudio de los sistemas biológicos a diferentes niveles de abstracción, gracias al mayor entendimiento y mejor comprensión de los gráficos, aunque mediante este método no se permite detallar de modo preciso los sistemas biológicos y es necesario recurrir a los modelos matemáticos para permitir la descripción cuantitativa de dichos sistemas.

El método gráfico permite fácilmente la visualización de las interacciones entre los elementos y brindan una mejor base para la simulación de su comportamiento y una mayor capacidad de predicción de las propiedades que no son evidentes para el investigador. En los últimos años la modelación de sistemas biológicos se ha convertido en esencial para el estudio de la Biología de Sistemas, existen muchas organizaciones alrededor del mundo que se dedican al estudio de la misma, entre las primeras empresas que emprendieron el estudio de la modelación de sistemas biológicos se encuentran Entelos y Gene Network Sciences, pero ha cedido su espacio a otras tales como: Bioalma, Biotechvana y Celeromics.

Entre los institutos más destacados en la investigación de esta rama, conformados en su mayoría por grupos de investigación y universidades se encuentran: Magdeburg Centre for Systems Biology, Okinawa Institute of Science and Technology Graduate School y GeneGo Inc. En la actualidad, las

empresas y los grupos de investigación han creado software para el análisis, estudio y comprensión de los sistemas biológicos, tales como: COPASI, CELLWARE, CellDesigner y JSim.

En su mayoría estos software son propietarios por lo que no están exentos de pago. Existen otros, que son libres de pago, pero no poseen código abierto, por lo que su reutilización no es viable. Se encuentran también otros que centran su objetivo en la resolución de un problema específico de la Biología de Sistemas ya sea Análisis, Modelación o Simulación, sin llegar a abordarlos todos.

Los científicos e investigadores han encontrado otro reto que ha sido la creación de una notación que permita la representación de los sistemas biológicos y que a su vez elimine las incongruencias y ambigüedades existentes en su estudio. Esto brindaría la posibilidad de una mayor eficiencia en la comprensión de los mismos. Otro significativo desafío se encontró en la creación de un formato estándar de intercambio que unifique a todos los software que trabajan con la modelación de sistemas biológicos.

En tiempos en que la Revolución Cubana le brinda una gran importancia al sistema de salud, así como a las investigaciones médicas, biológicas y farmacéuticas, los investigadores no cuentan con una herramienta de trabajo propia. Tanto así que no poseen un software capaz de modelar gráficamente los sistemas biológicos, para su posterior simulación y análisis. Además, les resulta imposible el poder intercambiar sin necesidad de presencia física, cualquier tipo de estudios realizados en Cuba, con estudios realizados por especialistas foráneos.

Como parte de las investigaciones realizadas por el Centro de Inmunología Molecular (CIM) surge una posible solución a la problemática anteriormente planteada: un proyecto de software realizado en la Universidad de la Ciencias Informáticas (UCI) para la Simulación y Análisis de Sistemas Biológicos (alasBioSyS). Dicho proyecto estaría compuesto por cuatro módulos fundamentales: Modelación, encargado de la creación de un modelo gráfico; Editor de Ecuaciones, previsto para gestionar todo lo referente a las ecuaciones diferenciales que describen a los sistemas biológicos; Simulación, encargado de la simulación de los sistemas biológicos y el módulo de Análisis, que luego de generadas las simulaciones se encargaría del análisis de las mismas.

El módulo de Modelación ha sido objeto de desarrollo en dos intentos anteriores, pero los resultados obtenidos no fueron los esperados. En el primero de los casos la representación gráfica se realizaba

sin el uso de una biblioteca especializada en la misma y pese a tener el software un buen diseño arquitectónico, el acabado gráfico de los modelos no poseía la calidad requerida, además no se logró una estandarización de la modelación, por lo cual el software no sería competitivo a nivel internacional. En el caso más reciente entre otras dificultades el sistema no contaba con una de las funcionalidades requeridas para el mismo, la importación de ficheros SBML para el trabajo con estos, además de que su diseño estaba basado en el trabajo con una biblioteca (libSBML) la cual era imposible de incluir en el paquete de instalación de la aplicación y debía que ser previamente instalada en las estaciones donde se fuera a utilizar la plataforma.

En la actualidad los software de punta que implementan un módulo de Modelación, tienen en cuenta la utilización de un estándar de modelación gráfica, para facilitar la compresión de los modelos creados por parte de los científicos entendidos en este tema, además de que puedan ser visualizados por cualquier otra herramienta encargada de brindar prestaciones similares o que simplemente se encargue del trabajo o la gestión de sistemas biológicos, bajo el estándar seleccionado.

Ante esta situación, se plantea el siguiente **Problema a resolver:** La necesidad de modelar gráficamente sistemas biológicos, utilizando un estándar de modelación en alasBioSyS. El problema planteado se enmarca en el **Objeto de estudio**: Modelación de Sistemas Biológicos, centrado en el **Campo de acción**: Modelación gráfica de sistemas biológicos. En aras de solucionar el problema planteado, se define como **Objetivo general**: Desarrollar un modelador gráfico que permita la representación de los sistemas biológicos en alasBioSyS utilizando un estándar de modelación.

Para el alcance del mismo son planteados los siguientes **Objetivos específicos**:

- Seleccionar el estándar de modelación a utilizar en la aplicación.
- Definir las funcionalidades de la aplicación, teniendo en cuenta el estándar seleccionado.
- Diseñar las funcionalidades de la aplicación
- Implementar las funcionalidades de la aplicación.
- Validar la aplicación implementada.

En vista de cumplir los objetivos previstos se desarrollaron las siguientes **tareas**:

- Estudio del estado del arte de la modelación gráfica de los sistemas biológicos.
- Definición de las funcionalidades de la aplicación.
- Realización del diseño de la aplicación.

- Implementación de las funcionalidades de la aplicación.
- Realización de pruebas de caja negra para validar el correcto funcionamiento de la aplicación.

El presente documento está estructurado en 4 capítulos:

Capítulo 1: Fundamento teórico

En este capítulo se brinda una descripción sobre los aspectos fundamentales en relación a la Biología de Sistemas, su origen, evolución, principales características y nuevas iniciativas creadas para dar solución a alguno de sus problemas. Se analiza el estado del arte de la modelación gráfica de sistemas biológicos, características fundamentales y estándares utilizados internacionalmente. Se realizan análisis sobre los software y herramientas a utilizar y se describe la metodología a seguir para dar solución al problema.

Capítulo 2: Características del Sistema.

El capítulo muestra una breve descripción sobre las características del sistema, en las que son identificables las clases del dominio, requerimientos tanto funcionales como no funcionales y se encuentran definidos los actores que conforman el sistema. Se presentan los diagramas de casos de uso del sistema, junto con una detallada descripción de cada uno de ellos.

Capítulo 3: Diseño del Sistema.

En este capítulo se define la arquitectura mediante una descripción basada en el sistema. Se presenta el modelo de diseño definiéndose las clases y los diagramas de interacción que componen cada modelo, así como también los patrones de diseño utilizados.

Capítulo 4: Implementación y Prueba del Sistema

Este capítulo abarca los diagramas de componentes precisados para la implementación del sistema. Presenta también, pruebas de caja negra para validar las funcionalidades del sistema, así como varias pantallas de la aplicación. Se realizó un experimento comparativo para comprobar el funcionamiento del sistema con respecto al de un software profesional.

Capítulo 1.
Fundamento Teórico

En la actualidad, la modelación de sistemas biológicos reviste gran importancia para el estudio de la Biología de Sistemas. Para comprender la necesidad de la modelación gráfica de dichos sistemas es necesario un estudio de las características generales de la ciencia que lo engloba, la Biología de Sistemas, como área interdisciplinaria, sus principales retos, estándares, preferencia de la comunidad internacional de desarrollo de software para darle solución a las problemas que se plantea el estudio de los sistemas biológicos.

1.1 Biología de Sistemas

La Biología de Sistemas se define como una disciplina dedicada al estudio integrado de los componentes de un sistema biológico y sus interacciones a lo largo del tiempo. [4]

La regla para conocer el funcionamiento de una célula ha sido purificar, aislar y analizar los componentes y estructuras en condiciones experimentales simples y bien controladas. Sin embargo, a raíz de los modernos métodos analíticos de alto rendimiento, se ha generado un modo distinto de analizar las funciones celulares que no dependen de aislar los componentes o las funciones, sino por el contrario analizar conjuntos complejos de componentes y funciones.

En la década de los 60 la evolución de la Biología Molecular, el fraccionamiento del conocimiento, los nuevos procedimientos para el acceso a un volumen de datos constantemente en aumento y el desarrollo de las ciencias de la información, trajo consigo la aparición de una nueva disciplina para estudiar la célula como un todo, es decir tomando un punto de vista más holístico que el tradicional, es lo que hoy se denomina Biología de Sistemas, aunque su institucionalización académica no se produjo hasta el año 2000. [5]

La Biología de Sistemas es vista como área interdisciplinaria que engloba conocimientos de varias ciencias como la Matemática, la Informática, la Biología y la Física, sin poderse definir el alcance de cada una de ellas en la interrelación.

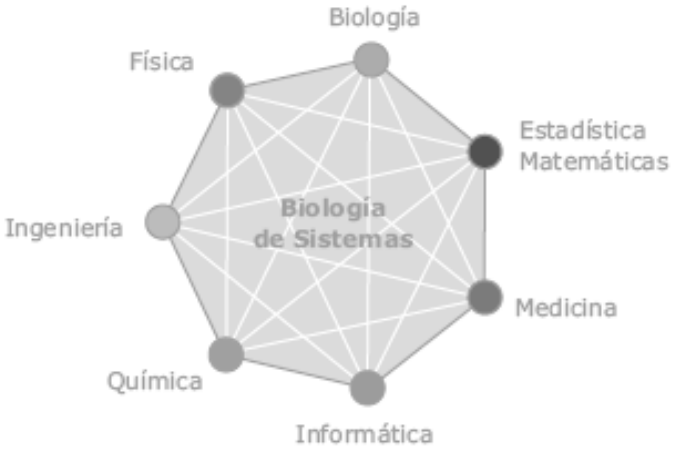


Figura 1. Disciplinas que componen la Biología de Sistemas.
Fuente: Munich Systems Biology Forum (MSBF).

1.1.1 Características de la Biología de Sistemas

A diferencia de los métodos clásicos de estudio usados por los biólogos basados en el método científico, que se refieren a 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 sistema. Se caracteriza por la integración de aproximaciones experimentales y computacionales, en la comprensión de los sistemas biológicos. [6]

Dicha ciencia posee otro grupo de características que describen su adaptabilidad a la integración de los estudios experimentales y los estudios computacionales, para un mayor fortalecimiento de los conocimientos sobre los sistemas biológicos [6]:

- Estudia los sistemas biológicos de una forma global desde nivel molecular hasta niveles de ecosistemas.
- Maneja una gran colección de datos procedentes de estudios experimentales.
- Propone modelos matemáticos que pueden explicar algunos de los fenómenos biológicos estudiados.
- Proporciona soluciones matemáticas que permiten obtener predicciones para los procesos biológicos.
- Realiza estudios de comprobación de la calidad de los modelos descritos por medio de la comparación entre las simulaciones numéricas y los datos experimentales.

1.1.2 Retos, limitaciones y perspectivas del desarrollo de la Biología de Sistemas

La Biología de Sistemas es una disciplina que se encuentra relativamente en sus inicios, por lo que afronta la tarea en los próximos años de la resolución de algunos problemas y retos que serán claves a la hora de conseguir los objetivos que se esperan de ella.

Uno de estos retos es la formulación de modelos matemáticos más robustos, puesto que es esencial en el desarrollo de la Biología de Sistemas, debido a que estos pueden describir el comportamiento de un sistema biológico y permiten desarrollar predicciones sobre el comportamiento del sistema en cuestión. La estandarización de experimentos in vitro e in vivo y sus datos es a menudo inconsistente, inaccesible, incompleta, o desestructurada. Esto conlleva frecuentemente el diseño de modelos independientemente de los datos disponibles, con la esperanza de que aparezcan parámetros adecuados para el modelo, o que los parámetros puedan ajustarse razonablemente al modelo para conseguir que funcione. La integración de los modelos cuantitativos y cualitativos es uno de los principales retos de la Biología de Sistemas, así como también paradójicamente la Biología de Sistemas se enfrenta por igual a un exceso de información y a la carencia de datos suficientes. En el primer caso es importante conocer cuáles datos son relevantes y cuáles son debidos a “ruido” procedente de la propia metodología empleada.

Dentro de las perspectivas de la Biología de Sistemas se encuentra, a corto plazo, la mejora de procesos de la Industria alimentaria mediante estrategias embebidas en esta ciencia, el trabajo futuro también conseguirá optimizar las técnicas de análisis y minería de datos y la evolución de las redes metabólicas de interacción proteína-proteína. Entre las perspectivas previstas a largo plazo se encuentran el avance en la búsqueda de una medicina personalizada y la construcción de modelos predictivos que representen la respuesta inmune en humanos.

1.2 Modelación de Sistemas Biológicos

Los modelos computacionales son requeridos para entender el comportamiento dinámico de determinados objetos de estudio, realizar una adecuada modelación ayuda a los expertos a predecir las consecuencias asociadas a cada una de las soluciones posibles en un problema y dado esto, tomar las mejores decisiones. La modelación computacional permite develar comportamientos e interacciones no programadas, determinar relaciones entre componentes y características en conjunto, desconocidas o incluso imposibles de apreciar. Dentro de las características que los modelos computacionales deben tener se encuentran la capacidad de realismo y el detalle del modelo. [7]

La modelación surge como resultado de la investigación científica en cualquier campo, aunque es notoriamente importante en el mundo biológico. En la actualidad se han estudiado internacionalmente una serie de técnicas, teorías y métodos matemáticos para explicar y comprender el comportamiento fundamentalmente complejo de los sistemas biológicos que, a su vez se ha planteado como una auténtica herramienta o metodología para estudiar sistemas de naturaleza compleja. La modelación de sistemas biológicos puede dividirse en dos grandes grupos, la modelación matemática y la modelación gráfica.

Durante las dos últimas décadas para el estudio de la Biología de Sistemas se ha aplicado la modelización matemática a una gran variedad de problemas. Esta es una metodología sistemática que ha probado con éxito descubrir y comprender los procesos subyacentes y causas en la naturaleza a partir de sus relaciones y partes observables. Un modelo matemático de un sistema biológico es convertido a sistemas de ecuaciones. La solución de las ecuaciones, ya sea por métodos analíticos o numéricos, describe cómo el sistema biológico se comporta en el tiempo.

Los sistemas de ecuaciones diferenciales algebraicas son una forma muy común y útil de modelar un sistema biológico. Estos sistemas se utilizan para modelar una amplia variedad de procesos biológicos, a través de una diversidad de escalas. Por ejemplo, en un extremo están los modelos que describen la acción de los canales iónicos, y en el otro extremo, los modelos de la dinámica depredador-presa.

Pese a todas estas ventajas la modelización matemática posee características que no la hacen del todo favorable, principalmente la dificultad que trae consigo una representación basada en ecuaciones matemáticas en ocasiones muy complejas. Para lograr comprender un sistema biológico descrito matemáticamente es necesario un gran nivel de conocimiento, tanto matemático como de la Biología de Sistemas.

1.2.1 Modelación Gráfica de Sistemas Biológicos

Los conocimientos procedentes de los sistemas biológicos son cada día mayores, lo cual dificulta su comprensión. Modelar ha sido históricamente una de las soluciones dadas por el hombre al problema de comprender sistemas complejos, la Biología de Sistemas no está ausente a esa solución. La modelación gráfica se ha convertido en una metodología internacionalmente explotada por la comunidad de investigadores, mediante la cual se brinda la posibilidad de representación de un sistema biológico mediante un gráfico interpretado que representa sus dependencias.

En la actualidad extensas investigaciones que abarcan las moléculas y sus relaciones, están siendo relevantes para el descubrimiento y comprensión del funcionamiento de importantes sistemas biológicos. 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. 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. [6]

Hay algunos criterios que el sistema de notación debe cumplir, tales como [6]:

- **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 sí 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.

En los inicios de la Biología de Sistemas una de las limitaciones que tuvo y que ha contribuido a que esta disciplina se haya desarrollado de forma muy dispersa en todo el mundo, ha sido la no estandarización de los procedimientos experimentales y la utilización de programas informáticos no estandarizados. En los últimos años han surgido una serie de iniciativas en la creación de normas para

la descripción de los sistemas biológicos que han perfeccionado el estudio de la Biología de Sistemas. Gran parte de estas iniciativas utilizan como lenguaje de programación el XML, por tratarse de un lenguaje abierto que permite la actualización directa de los contenidos de sus aplicaciones y la normalización de la información mediante el desarrollo de jerarquías.

1.3 Estándares de Modelación

Desde la creación de la Biología de Sistemas, los científicos han encontrado algunas dificultades en el estudio de la Biología de Sistemas, uno de ellos es la no estandarización de bases de datos de estudios y de software de modelación. En los últimos años se han dado grandes pasos en la resolución de esta deficiencia, muestra de ello es la creación de estándares para lograr la recopilación y documentación homogénea de los datos procedentes de los estudios realizados. En la actualidad, existen varios de ellos, pero los más usados internacionalmente son: CellML, BioPAX, SBGN y SBML.

1.3.1 CellML

El lenguaje CellML es un estándar abierto basado en el lenguaje de marcado XML. El propósito de este, es almacenar e intercambiar información basada en modelos matemáticos. CellML permite a los científicos compartir modelos, incluso si están utilizando diferentes herramientas de modelado. También les permite reutilizar los componentes de un modelo en otro, lo que acelera el desarrollo del mismo. [8]

CellML está basado en la representación de modelos matemáticos, capaces de representar los sistemas de ecuaciones diferenciales algebraicas (así como otras relaciones matemáticas). Como tal, proporciona un mecanismo ideal para el intercambio y el archivo de modelos. Hay bases de datos públicas que contienen un gran número de modelos CellML, tales como el Depósito de Modelos CellML. La base de datos BioModel también proporciona modelos que han sido traducidos a partir de otros estándares de modelación a CellML.[9]

Sin embargo, por las ventajas científicas de la utilización de formatos como CellML y para el que el intercambio se realice plenamente, es importante que el software utilizado sea capaz de leer y escribir modelos en estos formatos. Para ellos es importante que la comunidad científica posea la capacidad de desarrollar software, basándose en las bases de datos existentes de modelos en estos formatos. [9]

El proyecto CellML ha provisto a su comunidad de una interfaz de programación de aplicaciones (API) la cual simplifica la tarea del procesamiento de un lenguaje XML. La total y correcta comprensión de

este estándar puede ser una tarea difícil, debido a algunas características complejas que posee el lenguaje. Por tanto, es importante que los desarrolladores de software de CellML Project dieran a luz una API para el trabajo con modelos CellML. [9]

1.3.2 BioPAX

BioPAX es un lenguaje para la representación de procesos biológicos a nivel molecular y celular. Apunta a permitir la integración, el intercambio, visualización y análisis de datos biológicos. En concreto, apoya el intercambio estandarizado de datos entre grupos, reduciendo su complejidad, al ofrecer un estándar con una semántica bien definida para la representación de los sistemas biológicos. Además, permite el desarrollo de la visualización de las bases de datos y facilita el análisis de los datos experimentales generados a través de una combinación con el conocimiento previo. [10]

BioPAX es capaz de representar reacciones metabólicas, interacciones moleculares y genéticas y las redes de regulación génica. Su uso, ha permitido que se puedan encontrar millones de interacciones, organizadas en miles de caminos de diferentes organismos en un número creciente de bases de datos. Esta gran cantidad de datos en una forma computable apoya la visualización, análisis y descubrimiento biológico. [11]

BioPAX apoya los esfuerzos para el trabajo hacia una representación completa de los procesos celulares básicos. Pero sólo recientemente han colaborado grupos como parte del proyecto BioPAX, para desarrollar una forma generalmente aceptada de representar estos mapas de ruta para los procesos celulares. Para crear un completo mapa de procesos moleculares debe incluir todas las interacciones, reacciones, las dependencias, la influencia y el flujo de información entre los grupos de moléculas en las células y entre las células. [11]

1.3.3 SBGN

La era de la biología molecular, y más recientemente el auge de la genómica y otras tecnologías de alto rendimiento, han provocado un espectacular aumento en los datos y consigo la problemática de su interpretación. También favoreció el uso rutinario de software para ayudar a formular hipótesis, diseñar experimentos e interpretar resultados. Con este objetivo, se inició el proyecto SBGN (Notación Gráfica para Sistemas Biológicos, por sus siglas en inglés) en 2005, para desarrollar y estandarizar una notación gráfica sistemática y sin ambigüedades para su aplicación en molecular y biología de sistemas. [12]

La misión de SBGN es el desarrollo de lenguajes gráficos estándar de alta calidad, para la representación de procesos biológicos y sus interacciones. El lenguaje SBGN está pensado para fomentar el almacenamiento eficiente, intercambio y reutilización de la información. La simplicidad de la sintaxis y la semántica, hace los mapas SBGN adecuados para su uso en el mundo de la Biología y la Bioquímica. [13]

Las entidades moleculares poseen muchas propiedades que afectan a sus interacciones con otras entidades. El intento de representar todas las posibles reacciones e interacciones en el mismo diagrama es a menudo inútil. El proyecto SBGN se dio a la tarea de crear diferentes estilos de notaciones para controlar esta complejidad, presentando únicamente lo que se necesitaba en un contexto específico. Cada vista se centra en sólo una parte de la semántica del sistema en su conjunto. [12]

1.3.3.1 Descripción de Procesos

La descripción del proceso SBGN, muestra los cursos temporales de las interacciones bioquímicas en una red. Se puede utilizar para mostrar todas las interacciones moleculares que tienen lugar en una red de entidades bioquímicas. [12]

1.3.3.2 Entidad Relación

Entidad-Relación es el lenguaje le permite ver todas las relaciones en las que participa una entidad determinada, independientemente de los aspectos temporales. Las relaciones pueden ser vistas como reglas que describen las influencias de los nodos entidades en otras relaciones. [12]

1.3.3.3 Flujo de Actividades

El flujo de actividades es el lenguaje que representa el flujo de información entre las entidades bioquímicas en una red. Se omite información acerca de las transiciones de estado de las entidades y es particularmente conveniente para representar los efectos de las perturbaciones, ya sea genética o ambiental en la naturaleza. [12]

1.3.4 SBML

SBML (Lenguaje de Marcado para Sistemas Biológicos, por sus siglas en inglés) es un formato legible por la máquina para la representación de modelos, considerado un estándar internacional para la representación de sistemas biológicos. Es neutral con respecto a los lenguajes de programación y codificación de software, sin embargo, está orientado a permitir que los modelos puedan ser codificados utilizando XML. Mediante el apoyo a SBML como un formato para leer y escribir modelos,

diferentes herramientas de software pueden comunicarse directamente y almacenar la misma representación computable de esos modelos. Esto elimina un impedimento para compartir los resultados y permite a otros investigadores comenzar con una representación inequívoca del modelo, realizar una inspección minuciosa, proponer correcciones precisas y extensiones, y aplicar nuevas técnicas y enfoques, en fin, a hacer mejor ciencia. Es orientado a describir los sistemas en que las entidades biológicas están involucradas, y modificadas por los procesos que ocurren en el tiempo. [14]

Una reacción química puede ser dividida en sus elementos conceptuales: las especies, las especies reactantes, las especies productos, reacciones, las leyes de tasas, y los parámetros en las leyes de velocidad. Para analizar o simular una red de reacciones y componentes adicionales, hay que incluir, los compartimentos para las especies, y las unidades en sus diversas cantidades. Una definición de un modelo en SBML consiste simplemente en listas de una o más de estos diversos componentes [15]:

Compartimiento: Un recipiente de volumen finito tanto para especies como para sus reacciones.

Especie: una sustancia química o entidad que participa en una reacción. Algunas especies son ejemplos de iones tales como los iones de calcio, incluye también macromoléculas, ácidos nucleicos, entre otras.

Reacción: Es una descripción de algún tipo de transformación, el transporte o el proceso de unión que puede cambiar una o más especies. A las reacciones se han asociado leyes de velocidad que describen la forma en que tienen lugar.

Parámetro: Una cantidad que tiene un nombre simbólico. SBML proporciona la posibilidad de definir los parámetros que son globales a un modelo, así como que son locales para una sola reacción.

Regla: Una expresión matemática que se añade a la ecuaciones del modelo construido a partir de la serie de reacciones. Las reglas pueden ser utilizadas para establecer los valores de los parámetros y establecer restricciones entre las cantidades.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level1"
      level="1" version="2">
  <model name="gene_network_model">
    <listOfUnitDefinitions>
        ...
    </listOfUnitDefinitions>
    <listOfCompartments>
        ...
    </listOfCompartments>
    <listOfSpecies>
        ...
    </listOfSpecies>
    <listOfParameters>
        ...
    </listOfParameters>
    <listOfRules>
        ...
    </listOfRules>
    <listOfReactions>
        ...
    </listOfReactions>
  </model>
</sbml>
```

Figura 2. Estructura en formato XML de un modelo SBML.

La adopción de SBML ofrece muchos beneficios por encima de los otros estándares por lo que es el escogido para el desarrollo del Módulo de Modelación de alasBioSyS, dentro de los que se encuentran: permitir el uso de múltiples herramientas sin tener que reescribir los modelos para cada una, publicar y compartir modelos en una forma que otros investigadores pueden usar, incluso en un entorno de software diferente y por último asegurar la supervivencia de los modelos y el esfuerzo intelectual puesto en ellos, más allá de la vida útil del software utilizado para crearlos.

1.4 Software que modelan sistemas biológicos utilizando SBML

1.4.1 CellDesigner

CellDesigner es un editor de diagramas estructurados para la elaboración de redes bioquímicas. Las redes son graficadas mediante diagramas de procesos, con el sistema de notación gráfica propuesta por Hiroaki Kitano, y se guardan bajo las normas del formato SBML. Como características se le puede señalar a este software que es capaz de representar semánticas bioquímicas, brinda una detallada descripción de las transiciones de los estados de las proteínas, además de ser un software en extremo portable, siendo una aplicación desarrollada en Java. [16]

1.4.2 Cellware

Cellware además de haber sido diseñado para llevar a cabo la modelización y simulación de las vías de regulación de genes y metabólicas, también es capaz de ofrecer un entorno integrado para diversas representaciones matemáticas, la estimación de parámetros y optimización de los mismo. Como característica principal presenta una pantalla gráfica de fácil uso y la capacidad para ejecutar los

modelos grandes y complejos se proporcionan de forma predeterminada. Una característica muy especial de Cellware es que es el primero en crear redes basadas en modelos y simulación en el campo de la Biología de Sistemas. [17]

1.4.3 COPASI

COPASI es una aplicación de software para simulación y análisis de redes bioquímicas y su dinámica. Es un programa independiente que apoya los modelos de la norma SBML y puede simular su comportamiento mediante ecuaciones diferenciales ordinarias o algoritmo de simulación estocástica de Gillespie. Dentro de sus fundamentales características se encuentran que lleva a cabo varios análisis de la red y su dinámica. Tiene un buen soporte para la estimación de parámetros y optimización, proporciona medios para visualizar datos en las secciones personalizables, brinda además histogramas y las animaciones de diagramas de red. [18]

1.5 Necesidad de la creación de un Modelador Gráfico para alasBioSyS

Debido a que la gran mayoría de las herramientas existentes y disponibles en la actualidad poseen los estándares definidos para la modelación de sistemas biológicos, se hace necesario el desarrollo de un Modelador Gráfico de sistemas biológicos para su integración al software alasBioSyS, para que este incorpore dichos estándares, con el fin de ofrecer un software competitivo y compatible con otras herramientas de modelación a la comunidad de investigadores de la Biología de Sistemas. Los módulos de modelación gráfica de los software anteriormente referenciados no se utilizan, puesto que estos no son de código abierto (open source) y poseen licencias que imposibilitan su reutilización.

A continuación se describen las características de la metodología de desarrollo, herramientas y lenguajes de programación utilizados para la descripción de requisitos, el diseño y la implementación del módulo de modelación gráfica para su integración al software alasBioSyS.

1.6 Metodología de desarrollo

Las metodologías de desarrollo de software describen los pasos para la creación de un producto de software. Definen detalladamente qué es lo que se debe hacer, cómo y quién debe hacerlo. Precisan cuáles son las tareas a desarrollar durante el ciclo de vida del proyecto, los artefactos que han de ser construidos, cómo deben hacerse y el orden en que serán realizados, además de designar responsables por cada tarea. Además detallan la información necesaria para comenzar una tarea, y a su vez la información que se debe producir como resultado de esta. Al cumplirse estas normas se obtiene como producto final, una solución eficiente, factible y de gran calidad al problema que le dio

origen. La selección de las metodologías a utilizar se hace sobre la base de las necesidades específicas de cada situación y a las características del equipo de desarrollo.

1.6.1 OpenUP

OpenUP es una metodología de desarrollo de software mínimamente suficiente, lo que significa que sólo el contenido fundamental se incluye. Se considera dentro del grupo de las metodologías ágiles, sus reconocidas prácticas están destinadas a conseguir un sistema de comunicación de equipo para promover una comprensión compartida del proyecto. La mayoría de sus elementos fomentan el intercambio entre los equipos de desarrollo, sincronizando intereses y compartiendo conocimientos, principio fundamental que promueve las buenas prácticas para propiciar un ambiente saludable y de colaboración. Maximiza los beneficios al equilibrar las prioridades, permitiendo a los desarrolladores llevar a cabo una solución que cumpla con los requerimientos del proyecto, minimice el riesgo al centrarse en la arquitectura y proporcione la retroalimentación y mejoramiento continuo al realizar prácticas que permitan incrementos progresivos a las funcionalidades. [20]

En consecuencia con lo anteriormente expuesto la metodología que se utilizó para guiar el desarrollo de las funcionalidades es OpenUp debido a sus características y a ser este modelo el que más se ajusta y cumple con las necesidades del desarrollo de la aplicación, además de ser la metodología escogida y utilizada en el desarrollo de versiones anteriores del software alasBioSyS como una decisión del departamento de Bioinformática del centro DATEC.

1.7 Lenguaje de Modelado

Los lenguajes de modelado permiten la modelación de objetos mediante un conjunto estandarizados de símbolos y técnicas de modelado. Son capaces de modelar una simplificación de la realidad donde se toma de los objetos solo los aspectos significativos, brindando así una mejor comprensión del sistema.

1.7.1 UML

UML (Lenguaje Unificado de Modelado por sus siglas en inglés) está considerado como el lenguaje de modelado gráfico y visual por excelencia, utilizado para especificar, visualizar, construir y documentar los componentes de un sistema de software. Una característica sobresaliente del UML es que no es un método, sino un lenguaje de modelado. Un método define su notación (lenguaje) y su proceso a seguir durante el ciclo de vida de desarrollo del software. Sin embargo UML solo se encarga de la notación gráfica y su significado, a partir de la cual se crearán los diseños de sistemas y no depende de un proceso, el cual sería el encargado de orientar los pasos a seguir para elaborar el diseño. De hecho,

UML trabaja totalmente ajeno a las metodologías de desarrollo, en la actualidad es utilizado por muchos procesos de desarrollo diferentes.

Una característica atractiva es su flexibilidad. UML es extensible o sea, capaz de mejorarse con nuevas características e independiente de cualquier proceso de análisis y diseño orientado a objetos específicos. UML tiene la libertad de diseñar sistemas utilizando varios procesos, pero todos los desarrolladores pueden expresar esos diseños con un conjunto de notaciones gráficas estándar. Este lenguaje permite captar información sobre la estructura estática y dinámica de un sistema, en la cual la estructura estática proporciona información sobre los objetos que intervienen en determinado proceso y las relaciones que existen entre ellos. [21]

Dentro de las características específicas y que definen a UML se encuentra que además de usar tecnología orientada a objetos, posee gran viabilidad en la corrección de errores, cuenta con valiosa capacidad de creación de modelos precisos, no ambiguos e incompletos. También se encuentra su capacidad para la realización de ingeniería inversa al poder conectarse con diferentes lenguajes de programación. Permite además la documentación de todos los artefactos del proceso de desarrollo y es un lenguaje muy expresivo que cubre todas las vistas necesarias para el desarrollo y posterior despliegue del sistema. [21]

Teniendo en cuenta las características anteriormente expuestas y su alto grado de compresión, el lenguaje de modelado escogido para la modelación del sistema es UML. Además de ser el lenguaje de modelado utilizado actualmente por el equipo de desarrollo del software alasBioSyS.

1.8 Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo del software. Las herramientas CASE de modelado con UML brindan la posibilidad de aplicar una metodología de análisis y diseño con abstracción del código fuente, de manera que la arquitectura y el diseño son más obvios y fáciles de comprender y modificar. En la medida que el proyecto es más grande y complejo, resulta más necesario utilizar una herramienta de este tipo, debido a que aporta grandes beneficios para todos los involucrados en un proyecto, por ejemplo, jefe del proyecto, analistas, arquitectos y desarrolladores.

1.8.1 Visual Paradigm

Visual Paradigm es una herramienta CASE diseñada para guiar el proceso de desarrollo de software. Ofrece un completo conjunto de herramientas, necesitadas por los equipos de desarrollo de software para la captura de requisitos, software de planificación, la planificación de controles, el modelado de clases, modelado de datos, entre otros. Facilita la interoperabilidad con otras herramientas CASE y se integra con varios entornos de desarrollo como el NetBeans.

Como características fundamentales y que inclinaron la balanza a favor de la utilización de esta herramienta CASE se encuentran la utilización del lenguaje de modelado UML, su capacidad para la generación de código en varios lenguajes, dentro de los que se encuentra Java. Permite incorporar los dibujos del Visio en cualquier diagrama de UML. Además de permitir la realización de diagramas de UML normales, posibilita también modelar hardware, dominio específico, el software, conectando una red de computadoras los componentes usando sus iconos, más allá de las anotaciones de UML normales. Es multiplataforma, está disponible en los sistemas operativos Windows y Linux y está orientada a la creación de diseños usando el paradigma de programación orientado a objetos.

1.9 Lenguaje de programación

1.9.1 Java

Java ofrece todas las ventajas de un lenguaje potente y robusto, puesto que fue diseñado para crear software altamente fiable. Es un lenguaje de programación basado en clases y orientado a objetos. Hereda su sintaxis de los lenguajes C y C++, pero cuenta con un modelo de objetos mucho más sencillo y elimina elementos de trabajo a bajo nivel como los punteros.

Java es un lenguaje que puede ser ejecutado en múltiples plataformas. Es uno de los escasos lenguajes cuyos programas pueden ser transportados de sistema operativo, computadora o entorno, sin necesidad de cambiar el código. Esta característica fue la que alimentó su auge en Internet: una red que pueda ser accedida desde cualquier máquina, debe apelar a tecnologías multiplataforma. Java ha sido concebido, desde sus orígenes, como un lenguaje capaz de producir código totalmente transportable. [22]

Una de las características más significativas de Java es que posee una arquitectura neutral, es decir, el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. La independencia de la arquitectura representa solo una parte de

su portabilidad. Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas.

Por las características anteriormente expuestas y por ser el lenguaje escogido y utilizado actualmente en el desarrollo del software alasBioSyS, el lenguaje de programación que se utilizó para la implementación del sistema es Java.

1.10 IDE

Un Entorno de Desarrollo Integrado (Integrated Development Environment, IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos.

Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones. Existen diferentes IDE como Zend Studio, PHP Editor, Eclipse y NetBeans.

1.10.1 NetBeans

Para el desarrollo de la aplicación fue necesario escoger la herramienta de desarrollo a utilizar según el lenguaje de programación que se seleccionó, por lo tanto se hizo uso del IDE Netbeans que es una herramienta para el desarrollo de aplicaciones de escritorio usando Java. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de Java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios. Además es un proyecto GNU (libre), tiene un excelente diseñador de interfaces integrado, es muy rápido y fácil de usar.

1.11 Conclusiones

En el presente capítulo se sentaron las bases que fueron necesarias para el comienzo del proceso de desarrollo de una herramienta computacional para la modelación gráfica de sistemas biológicos, la cual se desarrolló bajo la metodología OpenUP, puesto que facilita un proceso rápido y ligero en cuanto a contenido generado. Para lograr un estandarizado de la modelación en la herramienta se decidió el trabajo con SBML, puesto que permite que los modelos creados en el sistema posean perdurabilidad en el tiempo, así como que puedan ser legibles por otros software sin necesidad de reescribirlos. El lenguaje de modelado utilizado es UML, debido a su alto grado de comprensión y seguir el paradigma de programación orientado a objetos. La herramienta CASE usada para visualizar y diseñar los elementos del sistema fue Visual Paradigm ya que es multiplataforma, utiliza como lenguaje de modelado UML y se integra fácilmente con el entorno de desarrollo utilizado, el cual será NetBeans, debido a que soporta Java como lenguaje de programación y es un proyecto GNU (libre), tiene un excelente diseñador de interfaces integrado, es muy rápido y fácil de usar.

Capítulo 2.
Características del Sistema

En el presente capítulo se muestra la descripción del modelo de dominio perteneciente a la modelación de un modelo biológico, los requisitos funcionales y no funcionales del sistema. Se presenta el actor del sistema, así como el diagrama de casos de uso del sistema y las descripciones textuales detalladas de cada uno de los casos de uso.

2.1 Breve descripción del sistema

A través de la realización del presente trabajo de diploma se desarrolló una herramienta computacional para la modelación gráfica de sistemas biológicos, la misma puede funcionar sin la necesidad de usar sistemas externos, pero a la vez fue diseñado de tal forma que en un futuro pueda ser integrado con otras herramientas como el editor de ecuaciones y la herramienta para la simulación y análisis de sistemas biológicos.

Estas tres últimas herramientas, junto con el modelador gráfico desarrollado, en su conjunto se convierten en una potente herramienta computacional para el estudio de los sistemas biológicos, objetivo principal que se busca con el software alasBioSyS.

2.2 Modelado del dominio

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema y las relaciones que poseen entre ellos. Los objetos del dominio representan los elementos que existen o eventos que ocurren en el entorno en el que trabaja el sistema. El modelo de dominio se describe mediante diagramas UML (específicamente mediante diagramas de clases).

2.2.1 Descripción de clases del modelo de dominio

La representación de un sistema biológico, es expresada mediante un modelo biológico que puede contener especies, compartimientos, funciones matemáticas, reacciones químicas, y operadores lógicos.

Las especies pueden representar: químicos simples, macromoléculas, ácidos nucleicos o entidades no especificadas, de estas últimas se desconoce el tipo o simplemente su propósito carece de relevancia

para el modelo. Los compartimientos pueden definirse como abstracciones que son utilizadas para agrupar componentes de los sistemas biológicos. Como ejemplos se pueden señalar dentro del modelado de células: la membrana celular y el interior celular. Las reacciones químicas entre las diferentes especies pueden ser: de consumo, producción, catálisis, inhibición, modulación, disparadoras y de estimulación. En el caso de las funciones matemáticas se encuentran las reglas, que describen las restricciones que existen entre variables y las leyes cinéticas que regulan la velocidad a las que se producen las reacciones químicas. Los parámetros son variables utilizadas en las funciones matemáticas que pueden ser expresados en diferentes unidades. Los operadores lógicos como el AND, OR y NOT son utilizados en ocasiones en las que las especies se relacionan, dando como resultado final un nuevo grupo de especies. El operador AND es utilizado cuando las especies relacionadas dan como resultado un tercero, el OR se utiliza cuando una de las especies relacionadas puede dar como resultado un tercero y el NOT para denotar que la especie no puede producir ninguna salida.

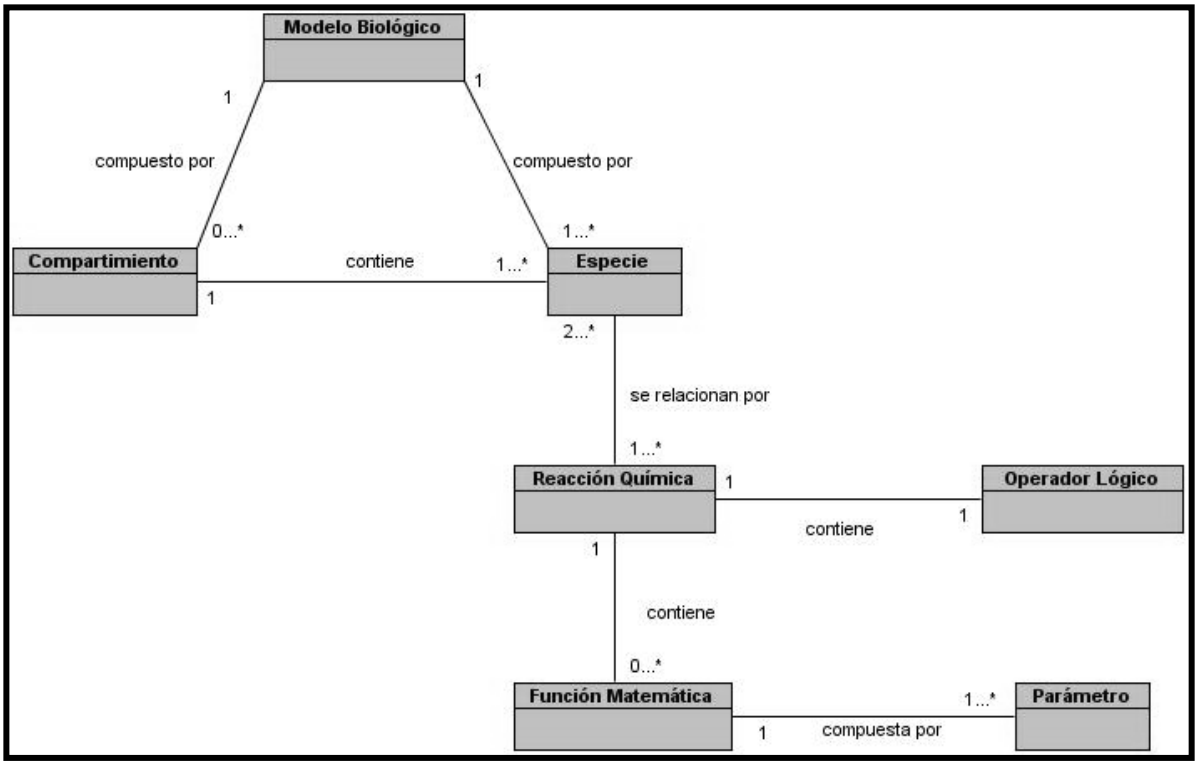


Figura 3. Modelo de Dominio del Modelador Gráfico para alasBioSyS.

2.3 Modelado del Sistema

2.3.1 Requisitos Funcionales

La herramienta a desarrollar a través de la realización del presente trabajo debe cumplir las siguientes capacidades o condiciones:

RF 1 Modelar gráficamente sistemas biológicos.
- RF 1.1 Administrar compartimientos.
  - RF 1.1.1 Insertar compartimientos.
  - RF 1.1.2 Eliminar compartimientos.
- RF 1.2 Administrar especies.
  - RF 1.2.1 Insertar especies.
  - RF 1.2.2 Eliminar especies.
- RF 1.3 Administrar reacciones químicas.
  - RF 1.3.1 Insertar reacciones.
  - RF 1.3.2 Eliminar reacciones.
- RF 1.4 Administrar operadores lógicos.
  - RF 1.4.1 Insertar operadores.
  - RF 1.4.2 Eliminar operadores.
- RF 1.5 Editar propiedades

RF 2 Exportar la información del modelo a formato SBML.

RF 3 Importar modelos creados en formato SBML.

RF 4 Guardar la información del modelo como imagen.

2.3.1 Requisitos No Funcionales

La herramienta desarrollada debe poseer cualidades y características para hacerla atractiva, confiable, usable y segura, dentro de estas se encuentran:

Software.

RNF 1 Para la instalación de la aplicación se debe disponer del sistema operativo Windows o GNU Linux.

RNF 2 Para el uso del sistema es necesario tener instalada la máquina virtual de Java 1.6 o una versión superior, debido a que las bibliotecas jsbml.jar y jgraph.jar la requieren.

Hardware.

RNF 3 La computadora donde se prevé instalar la aplicación debe contar con una memoria RAM de 512 MB como mínimo, aunque lo ideal sería 1 GB.

RNF 4 El sistema debe instalarse en máquinas con procesadores Pentium IV o superiores.

Usabilidad.

RNF 5 El sistema podrá ser usado por aquellos usuarios que posean conocimientos básicos en el campo de la modelación de sistemas biológicos.

RNF 6 Se debe lograr un diseño adaptable, con la capacidad de poder soportar funcionalidades adicionales o modificar las funcionalidades existentes, sin impactar el resto de los requerimientos contemplados en el sistema.

Soporte.

RNF 7 Mantenimiento: El sistema debe estar bien documentado, de forma tal que el tiempo de mantenimiento sea mínimo en caso de necesitarse.

RNF 8 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).

Interfaz de usuario

RNF 9 Se necesita una interfaz amigable, legible, interactiva, fácil de usar, profesional, clara y sencilla.

Requisitos Legales, de Derecho de Autor y otros.

RNF 10 Los derechos de autor serán registrados por la política que sigue la Universidad de las Ciencias Informáticas.

Requerimientos del diseño y la implementación.

RNF 11 El sistema debe cumplir con el estándar del Lenguaje de Marcado de Sistemas Biológicos y utilizar las funcionalidades de la biblioteca jsbml.jar. El lenguaje de programación a utilizar en la implementación debe ser Java para facilitar su integración al software alasBioSyS.

2.3.3 Definición de los Actores

Los actores de un sistema son las personas, sistemas o hardware externo que se relacionan o interactúan con dicho sistema. Cada actor juega un rol determinado en el mismo y diferentes usuarios pueden asumir el mismo rol de un actor. Como actor del sistema en cuestión se definió:

Tabla 1: Definición de los Actores del Sistema

Actor(es)	Descripción
Investigador	Es la persona que interactúa con el sistema. Es la encargada de la creación del modelo gráfico del sistema biológico y de realizar los estudios correspondientes.

2.3.4 Diagrama de Casos de Uso del Sistema

El Modelo de casos de uso describe la funcionalidad propuesta del sistema, para ello muestra la relación entre actor–caso de uso. Un actor es un usuario del sistema (Investigador) y un caso de uso representa una unidad discreta de interacción entre un usuario y el sistema. El siguiente diagrama muestra la interacción entre el investigador y cada caso de uso.

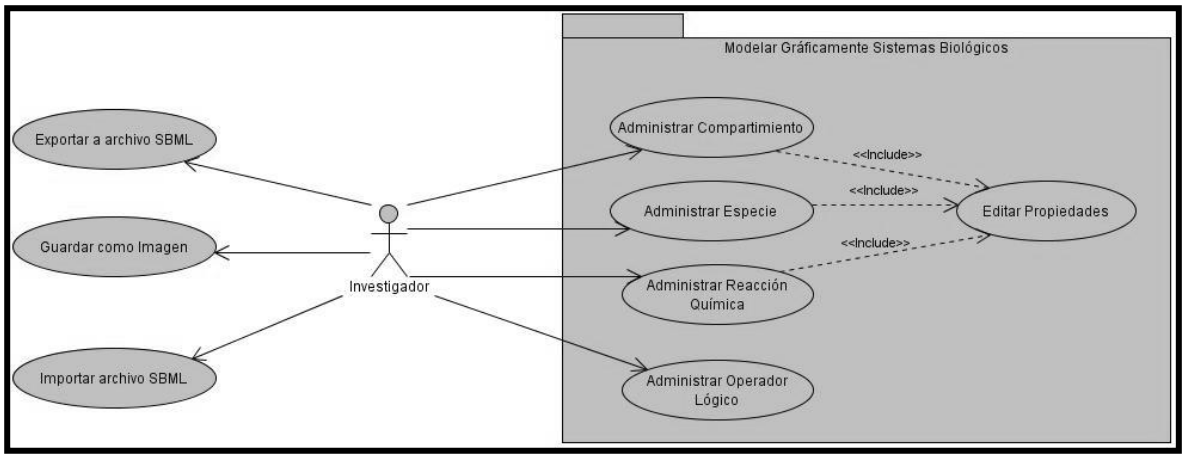


Figura 4. Diagrama de Casos de Uso del Sistema.

2.3.5 Descripción textual de Casos de Uso del Sistema

Para la comprensión de la funcionalidad asociada a cada caso de uso no es suficiente con la representación gráfica del Diagrama de Casos de Uso del Sistema, también es necesario llevar a cabo una detallada descripción textual de los mismos donde se especifican todas las interacciones ocurrentes para su realización. A continuación se muestran las descripciones textuales de 5 de los casos de uso del sistema, la totalidad de estas descripciones se encuentran en el artefacto Especificación de Casos de Uso dentro del Expediente de Proyecto.

2.3.5.1 Caso de Uso "Administrar Compartimientos"

Tabla 2: Descripción del caso de uso Administrar Compartimientos.

Nombre del Caso de Uso	**Administrar Compartimientos**
Actores	Investigador
Propósito	Permitir al investigador insertar o eliminar algún compartimiento.
Resumen	El caso de uso se inicia cuando el investigador desea realizar una de las siguientes opciones: • Insertar compartimiento: Se inicia cuando el investigador inserta nuevo compartimiento al modelo y finaliza cuando el sistema visualiza el mismo y actualiza los paneles de propiedades y de distribución jerárquica. • Eliminar Compartimiento: Se inicia cuando el investigador elimina el compartimiento del modelo, finalizando así el caso de uso.
Referencias	RF 1, RF 1.1, RF 1.1.1, RF 1.1.2
Precondiciones	Ninguna
Postcondiciones	En dependencia de la acción del investigador: • Se inserta un compartimiento al modelo. • Se elimina un compartimiento del modelo.
Descripción	
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El investigador selecciona la acción que desea realizar: • Insertar compartimiento • Eliminar compartimiento	2. En dependencia de la acción que el investigador realice: • Insertar compartimiento: ir al escenario "Insertar compartimiento" • Eliminar compartimiento: ir al escenario "Eliminar compartimiento"
Escenario "Insertar compartimiento"	
1. El investigador selecciona el compartimiento nuevo en la barra de herramientas y lo arrastra hacia el área de trabajo.	2. El sistema agrega el nuevo compartimiento al modelo biológico, lo muestra en el área de trabajo y actualiza los paneles de propiedades y de distribución jerárquica. Finalizando así el CU.

Escenario "Eliminar compartimiento"	
1. El investigador selecciona el compartimiento que desea eliminar del área de trabajo y presiona el botón del teclado "Delete".	2. El sistema elimina el compartimiento, todo lo que contiene, visualiza el cambio, finaliza cuando actualiza los paneles de propiedades y de distribución jerárquica. Finalizando así el CU.
Prioridad:	Crítico

2.3.5.2 Caso de Uso "Editar Propiedades"

Tabla 3: Descripción del caso de uso Editar Propiedades.

Nombre del Caso de Uso	**Administrar Editar Propiedades**
Actores	Investigador
Propósito	Permitir al investigador editar las propiedades de los componentes del modelo
Resumen	El caso de uso se inicia cuando el investigador va a realizar una de las siguientes opciones: • Editar Propiedades de Compartimiento: Se inicia cuando el investigador selecciona un compartimiento y se activa la paleta de propiedades de los mismos. • Editar Propiedades de Especie: Se inicia cuando el investigador selecciona una especie y se activa la paleta de propiedades de las mismas. • Editar Propiedades de Reacciones Químicas: Se inicia cuando el investigador selecciona una reacción química y se activa la paleta de propiedades de las mismas.
Referencias	RF 1, RF 1.5
Precondiciones	Debe existir algún compartimiento, especie o reacción química.
Postcondiciones	En dependencia de la acción del investigador: • Se modifican las propiedades de los compartimientos. • Se modifican las propiedades de las especies. • Se modifican las propiedades de las reacciones químicas.
Descripción	
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**

1. El investigador selecciona la acción que desea realizar: • Editar las propiedades de los compartimientos. • Editar las propiedades de las especies. • Editar las propiedades de las reacciones químicas.	2. En dependencia de la acción que el investigador realice: • Editar las propiedades de los compartimientos: ir al escenario “Editar compartimiento” • Editar las propiedades de las especies: ir al escenario “Editar especie” • Editar las propiedades de las reacciones químicas: ir al escenario “Editar reacción”
Escenario “Editar Compartimiento”	
1. El investigador presiona doble clic sobre el compartimiento a editar, se muestra un formulario con campos editables. El investigador procede a realizar el cambio y presiona el botón “Update”.	2. El sistema modifica la propiedad del compartimiento, visualiza el cambio de ser necesario. Finalizando así el caso de uso.
Escenario “Editar Especie”	
1. El investigador presiona doble clic sobre la especie a editar, se muestra un formulario con campos editables. El investigador procede a realizar el cambio y presiona el botón “Update”.	2. El sistema modifica la propiedad de la especie, visualiza el cambio de ser necesario. Finalizando así el caso de uso.
Escenario “Editar Reacción”	
1. El investigador presiona doble clic sobre la reacción química a editar, se muestra un formulario con campos editables. El investigador procede a realizar el cambio y presiona el botón “Update”.	2. El sistema modifica la propiedad de la reacción química, visualiza el cambio de ser necesario. Finalizando así el caso de uso.
Prioridad:	Crítico

2.3.5.3 Caso de Uso “Exportar a archivo SBML”

Tabla 4: Descripción del caso de uso Exportar a archivo SBML.

Nombre del Caso de Uso	**Exportar a archivo SBML**
Actores	Investigador
Propósito	Permitir al investigador exportar el modelo creado al formato estándar SBML.
Resumen	El caso de uso se inicia cuando el investigador va a exportar el modelo creado a un archivo con formato SBML.
Referencias	RF 2.

Precondiciones	Debe existir algún modelo.
Postcondiciones	
Descripción	
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El investigador se dirige hasta la barra de menú, dando clic en el menú Archivo.	2. El sistema muestra un menú con las opciones correspondientes.
3. El investigador procede a dar clic en la opción de menú "Exportar a SBML"	4. El sistema muestra una ventana para las opciones de nombrar el fichero, su tipo y la ubicación del mismo.
5. El investigador llena el campo del nombre y procede a dar una dirección para la ubicación del fichero y da clic en el botón "Aceptar".	6. El sistema muestra un mensaje refiriendo que la exportación ha terminado con éxito. Finalizando así el caso de uso.
Flujos Alternos	
5.1. Partiendo del punto 5 del Flujo Normal de Eventos. En caso de que el nombre insertado posea caracteres especiales o se encuentre en blanco.	5.2. El sistema muestra un mensaje de error refiriendo que el nombre propuesto no es válido, y se remite a la acción 4 de este caso de uso.
Prioridad:	Crítico

2.3.5.4 Caso de Uso "Importar archivo SBML"

Tabla 5: Descripción del caso de uso Importar archivo SBML.

Nombre del Caso de Uso	**Importar archivo SBML**
Actores	Investigador
Propósito	Permitir al investigador importar un modelo creado en formato SBML.
Resumen	El caso de uso se inicia cuando el investigador va a importar un modelo creado con formato SBML.
Referencias	RF 3.
Precondiciones	
Postcondiciones	En el área de trabajo se debe mostrar el modelo importado
Descripción	
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El investigador se dirige hasta la barra de menú, dando clic en el menú Archivo.	2. El sistema muestra un menú con las opciones correspondientes.

3. El investigador procede a dar clic en la opción de menú “Importar archivo SBML”	4. El sistema muestra una ventana para la opción de seleccionar la ubicación del fichero a importar y su tipo.
5. El investigador busca la ubicación del fichero, lo selecciona y da clic en el botón “Aceptar”.	6. El sistema muestra el fichero SBML importado de forma gráfica en el área de trabajo. Actualiza los paneles de componentes. Finalizando así el caso de uso.
Flujos Alternos	
5.1. Partiendo del punto 5 del Flujo Normal de Eventos. En caso de que el archivo seleccionado no sea formato SMBL (XML).	5.2. El sistema muestra un mensaje de error refiriendo que el formato del fichero es ilegible, y se remite a la acción 4 de este caso de uso.
Prioridad:	Crítico

2.4 Conclusiones

Al finalizar este capítulo se obtuvo una concreta claridad de la concepción del sistema, gracias al modelado del mismo a través del Modelo de Dominio y la definición de 11 requisitos no funcionales y 12 funcionales que el sistema debe ser capaz de alcanzar. Estos últimos fueron modelados en el diagrama de casos de uso del sistema usando los patrones: empaquetamiento, CRUD parcial e inclusión; gracias a los cuales se reorganizaron las funcionalidades en casos de uso, brindando así una mayor percepción del sistema a desarrollar. Para mostrar en mayor nivel de detalle los casos de uso, así como para conocer la secuencia de su funcionamiento, se especificó la descripción textual de los mismos; sentando así las bases para las restantes fases del proceso de desarrollo de la aplicación como diseño e implementación.

Capítulo 3.
Diseño del Sistema

Como resultado del capítulo anterior se obtuvo una vista externa del sistema. En este capítulo, a través del diseño del sistema, se profundiza en los casos de usos, detallándolos de manera que permitan reflejar una vista interna del sistema descrita con el lenguaje de los desarrolladores. En esta vista interna se determinan las clases que contendrán las funcionalidades definidas para la aplicación.

El diseño del sistema tiene como objetivo fundamental crear los modelos centrados en los requisitos y en vistas de la solución, preparando así el sistema para su implementación. En el presente capítulo se desarrolla el diseño, a través del cual se modela la aplicación, la que debe ser capaz de soportar todos los requerimientos mencionados en el capítulo anterior. Para ello se utilizarán diferentes artefactos como son: los diagramas de interacción, los diagramas de clases del diseño y las descripciones de algunas de las principales clases del diseño.

3.1 Estilo Arquitectónico

El desarrollo del sistema se basó en una arquitectura orientada a objetos y el estilo arquitectónico utilizado es de Llamada y Retorno que encapsula la arquitectura en capas y tiene como objetivo fundamental la separación de la lógica de negocios de la lógica de diseño. Dicho estilo posee como ventaja fundamental, que el desarrollo se puede llevar a cabo en varios niveles.

Para el desarrollo de la herramienta se utilizó 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 las bases de datos y la lógica de negocio. [23] Dicho patrón sigue con la anteriormente mencionada filosofía del estilo arquitectónico de llamada y retorno, gracias a sus características brinda la posibilidad de la obtención de varios niveles de abstracción al soportar múltiples vistas. La vista se separa del modelo permitiendo que no exista dependencia directa entre ellos, la interfaz de usuario es capaz de mostrar múltiples vistas simultáneas de los mismos datos. Permite además un mayor soporte a los cambios, producto a que los requisitos de interfaz tienden a cambiar más rápidamente que las reglas de negocio.

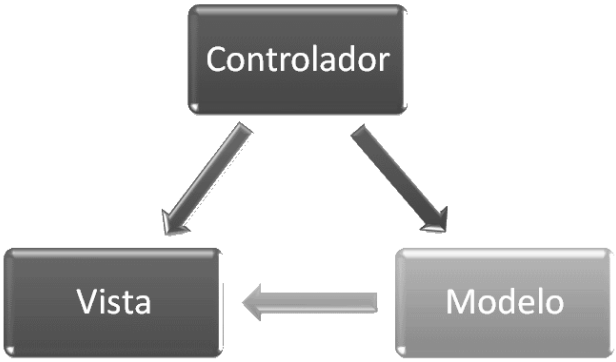


Figura 5. Representación del patrón Modelo Vista Controlador.

3.2 Patrones de Diseño

3.2.1 Alta Cohesión

El patrón Alta Cohesión se basa en una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Como solución el patrón permite asignar una responsabilidad de modo que la cohesión siga siendo alta. [24]

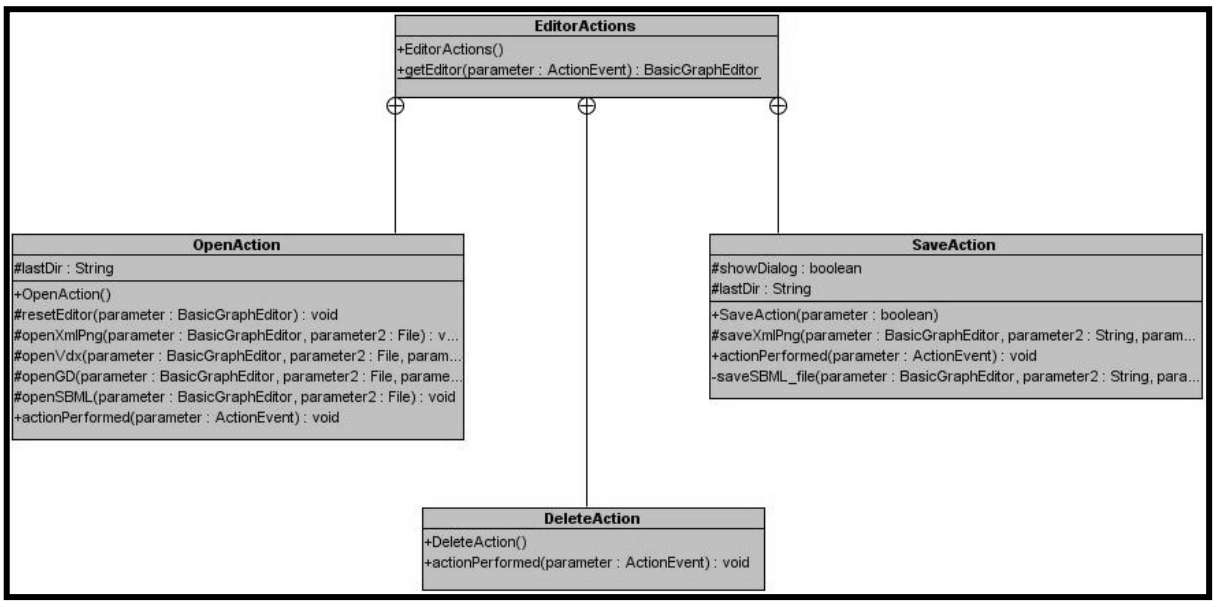


Figura 6. Uso del patrón Alta Cohesión en el sistema.

En la Figura 6 se evidencia el patrón de Alta Cohesión en la cual las responsabilidades rigurosamente relacionadas son esparcidas por clases como OpenAction, SaveAction y DeleteAction. Liberando así a la clase EditorAction de la realización de un trabajo mayor.

3.2.2 Controlador

Un evento de entrada externa consiste en cualquier evento generado por un actor externo. Un Controlador es un objeto de interfaz no destinado al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. El patrón Controlador sugiere asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema, a una clase que represente un manejador de los eventos del sistema. [24] Este patrón se muestra en los Diagramas de Clases de cada uno de los casos de uso (ver Figura 10), específicamente en la clase BasicGraphEditor, la cual es la encargada de capturar, manejar y dar solución a los eventos del sistema.

3.2.3 Creador

El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización. El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento, ya sea agregar, registrar o utilizar. Al escogerlo como creador, se da soporte al bajo acoplamiento. [24]

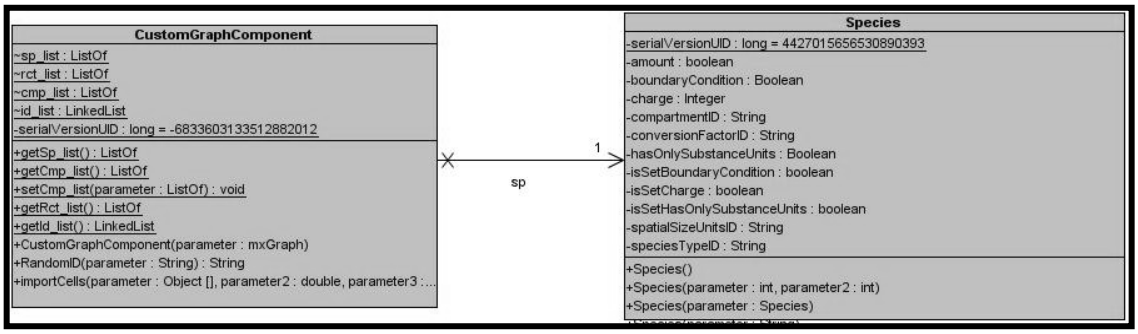


Figura 7. Uso del patrón Creador en el sistema.

En la Figura 7 se evidencia este patrón, puesto que es la clase CustomGraphComponent la que posee la lista de Especies y es la encargada del trabajo con las mismas, tanto en la creación, como en la edición de sus propiedades y finalmente en la eliminación de estas.

3.2.4 Experto

Si las responsabilidades a las clases se asignan en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, lo que permite reutilizar los componentes en futuras aplicaciones. Por lo que este patrón brinda la posibilidad de asignar una responsabilidad al experto en información, es decir, asignar dicha responsabilidad a la clase que cuenta con la información necesaria para cumplirla. [24]

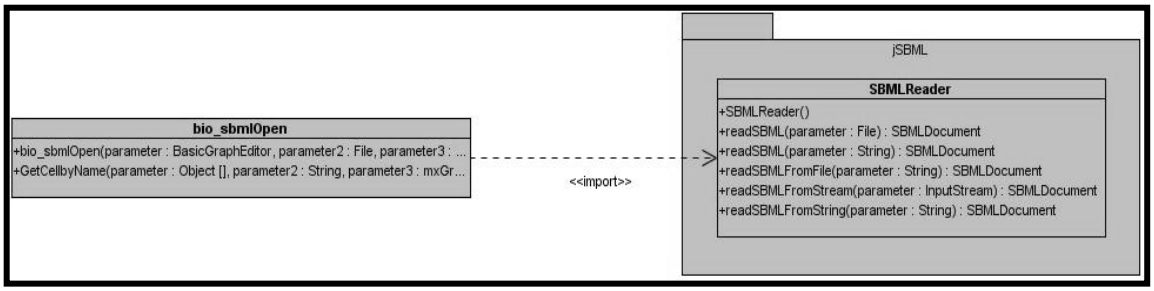


Figura 8. Uso del patrón Experto en el sistema.

En la Figura 8 se referencia el uso de dicho patrón cuando, al necesitar la clase bio_sbmlOpen la lectura de un fichero en formato SBML, se auxilia de la clase SBMLReader, contenida en la biblioteca jSBML para la lectura de dicho fichero, por ser esta clase la verdadera experta en la funcionalidad requerida.

3.2.5 Composite

La idea básica de este 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. [24]

En la Figura 9 las clases Species y Compartment heredan de una clase Symbol. Aunque ambas clases contienen funcionalidades completamente diferentes, existe una clase que puede ser utilizada como interfaz común para el trabajo con otras funcionalidades, en las cuales no es necesario la diferenciación entre ellas.

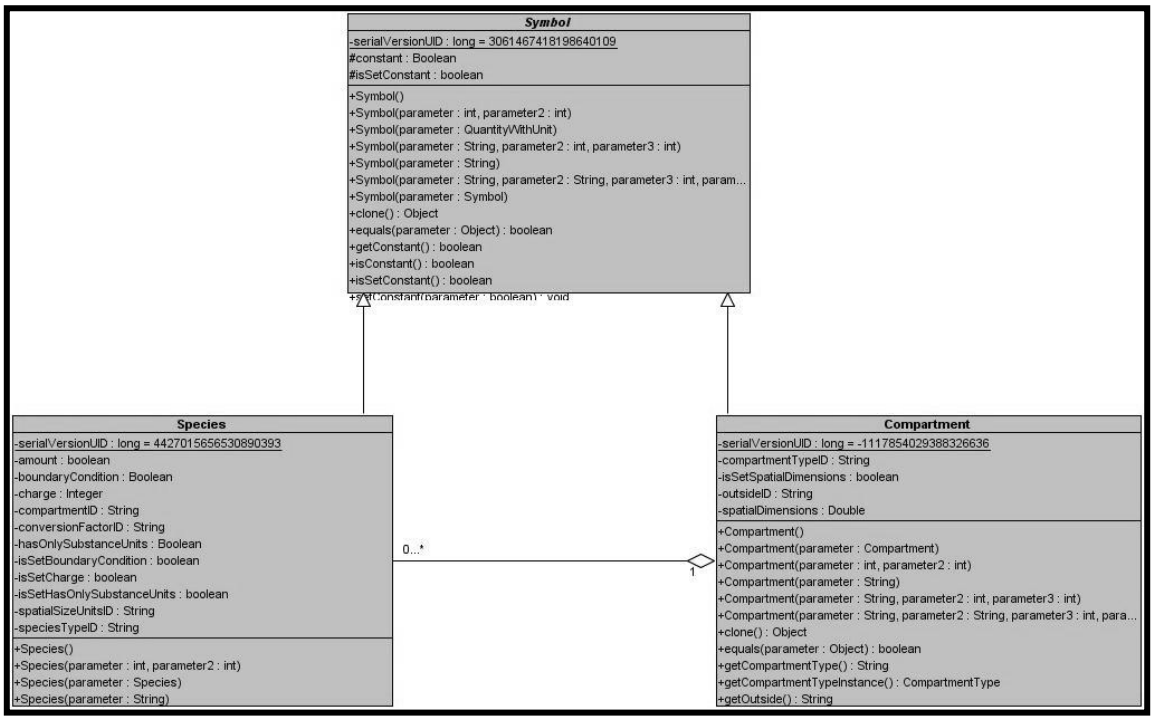


Figura 9. Uso del patrón Composite en el sistema.

3.3 Modelo de Diseño

3.3.1 Diagrama de Clases del Diseño

Los diagramas de clases son realizados para cada caso de uso y en ellos se muestran las diferentes clases que componen un sistema y sus respectivas relaciones. A continuación se muestran los diagramas de clases del diseño, confeccionados para dos casos de uso de la aplicación. Se han agrupado las clases de acuerdo a su funcionalidad en tres grandes paquetes, haciendo además referencia al patrón arquitectónico expuesto anteriormente: Modelo Vista Controlador. A continuación se muestran los Diagramas de Clases del Diseño de 2 de los casos de uso, la totalidad de estos diagramas se encuentran en el artefacto Modelo de Diseño dentro del Expediente de Proyecto.

3.3.1.1 Diagrama de Clases del Diseño del Caso de Uso Administrar Especie

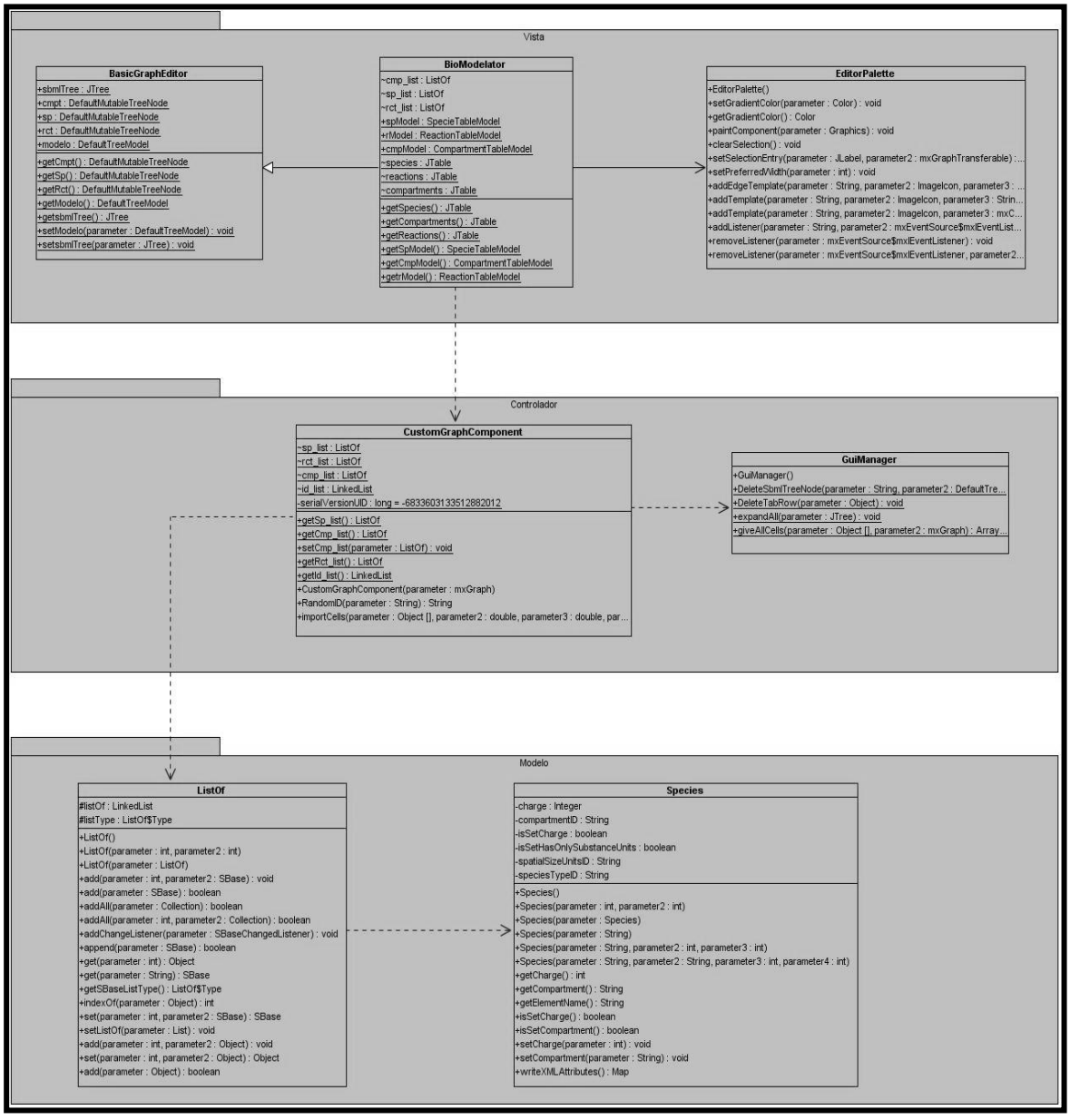


Figura 10. Diagrama de Clases del Diseño del Caso de Uso: Administrar Especie.

3.3.1.1.2 Descripción de Clases del Caso de Uso Administrar Especie

El caso de uso Administrar Especie se precisó utilizando el anteriormente mencionado patrón de diseño Modelo Vista Controlador. En el cual se definieron las siguientes clases.

BasicGraphEditor: Esta clase es la encargada de manejar los modelos de representación jerárquica, los objetos especie, compartimiento y reacción química, los eventos de teclado y el componente gráfico.

BioModelator: Está diseñada como clase principal, es la encargada de representar visualmente todos los cambios que tienen lugar en el sistema, contiene también los modelos de las tablas de propiedades de cada tipo de objeto insertado en el modelo. Al extender de la clase Basic GraphEditor obtiene y utiliza algunas de sus funcionalidades.

EditorPalette: Es la encargada de crear las pestañas con las formas a seleccionar por el usuario, además de manejar los eventos *pressed* y *released* del mouse para que luego sea creado el gráfico seleccionado. En dicha barra se encontrarán todos los componentes necesarios para la modelación gráfica de un sistema biológico.

customGraphComponent: Contiene las listas fundamentales del modelo SBML del sistema por medio de las cuales se crea el mismo, es la encargada de manejar además, la representación en BioModelator de los gráficos y la actualización de la interfaz visual.

guiManager: Contiene métodos auxiliares para la actualización de los diferentes modelos y la GUI. Las clases Species y listOf son modelos propios de jSBML, por medio de las cuales se crean las instancias requeridas para lograr el modelo de SBML.

3.3.1.2 Diagrama de Clases del Diseño del Caso de Uso Importar Archivo SBML

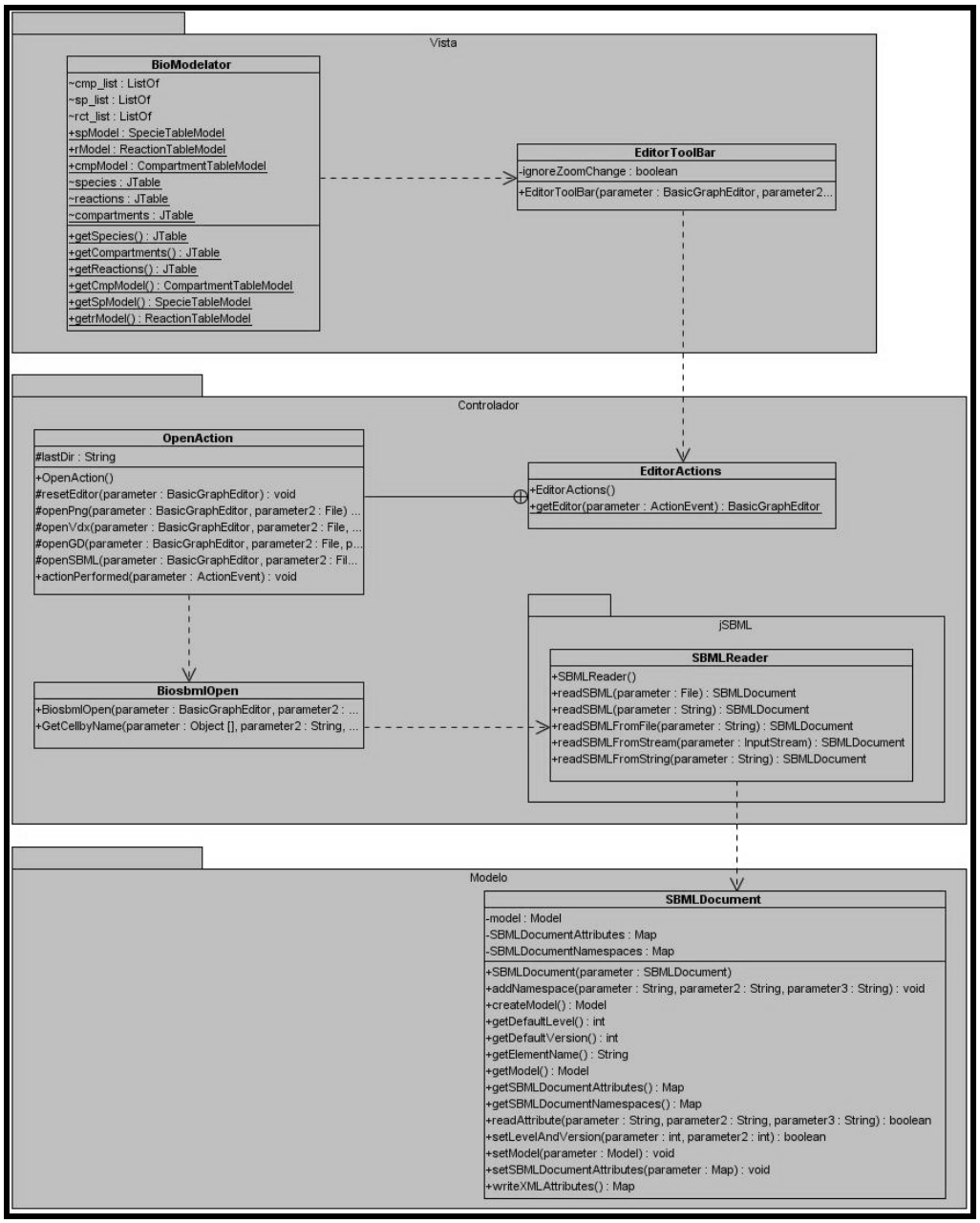


Figura 11. Diagrama de Clases del Diseño del Caso de Uso: Importar Archivo SBML

3.3.1.2.1 Descripción de Clases del Caso de Uso Importar Archivo SBML

En el caso de uso Importar Archivo SBML también se rige por el patrón MVC y se incluyen nuevas clases para el trabajo con el documento SBML.

EditorToolBar: Es la clase encargada de crear una barra de herramientas de acceso rápido para facilitarle al usuario el acceso a las funcionalidades fundamentales de la aplicación.

EditorAction: Contiene todas las acciones que se realizarán por el usuario en el Editor, como openAction() dentro de la cual se encuentra abrir el fichero SBML, método que se encarga de leerlo y a partir de la información capturada, representar el mismo en la interfaz de usuario.

bio_sbmlOpen: Es la clase llamada por openAction() y es la encargada de hacer posible la lectura del documento SBML y de guardar todos los datos leídos en el modelo de la aplicación, esta clase hace uso de la biblioteca jSBML, específicamente de la clase ***SBMLReader*** y de la clase modelo ***SBMLDocument***, siendo la primera la encargada leer el fichero SBML y crear una instancia de la segunda.

3.3.2 Diagramas de Interacción (Secuencia)

Los diagramas de secuencia son altamente efectivos para modelar interacciones entre objetos dentro de un sistema. Estos muestran la interacción de un conjunto de objetos en un sistema a través del tiempo y contienen detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes pasados entre los objetos. Se desarrollan tras examinar la descripción de un caso de uso para determinar cuáles objetos son necesarios para la implementación del escenario. La totalidad de los Diagramas de Secuencia se encuentran en el artefacto Modelo de Diseño, dentro del Expediente de Proyecto.

3.3.2.1 Diagrama de Secuencia del Caso de Uso Administrar Especie (escenario Insertar Especie)

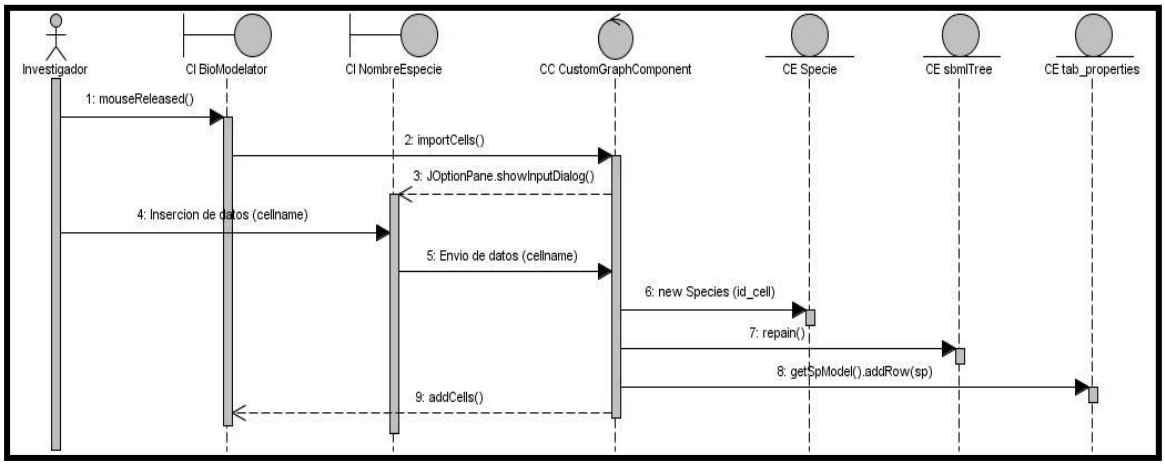


Figura 12. Diagrama de Secuencia del Caso de Uso Administrar Especie (escenario Insertar Especie)

3.3.2.2 Diagrama de Secuencia del Caso de Uso Importar Archivo SBML

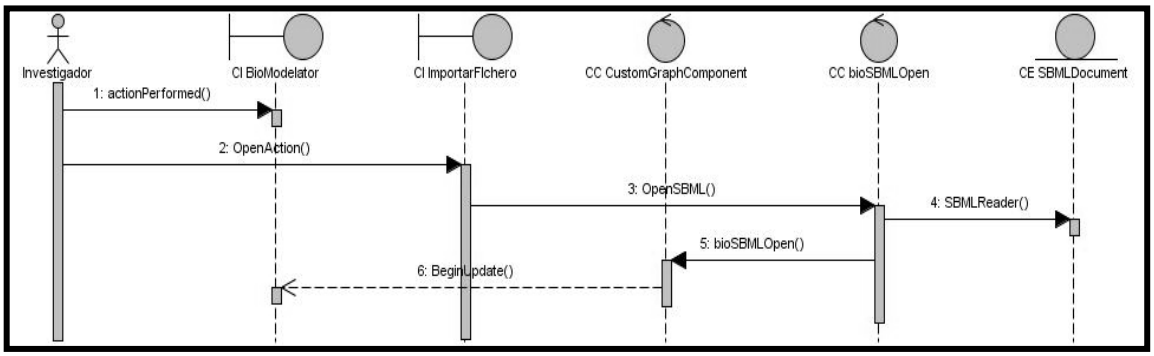


Figura 13. Diagrama de Secuencia del Caso de Uso Importar Archivo SBML.

3.4 Conclusiones

En el presente capítulo se definió como estilo arquitectónico a utilizar Llamada-Retorno, el cual permite que el desarrollo se pueda llevar a cabo en varios niveles. Se precisó el uso del patrón arquitectónico Modelo–Vista–Controlador en el diseño de los diagramas de clases, organizando las clases en 3 paquetes fundamentales coincidiendo con los nombres de dicho patrón. Esto facilitó aislar la vista del modelo y creó un paquete que contiene un conjunto de clases encargadas de controlar el funcionamiento del sistema. Para la creación de dichos diagramas se utilizaron los patrones de diseño Alta Cohesión, Controlador, Creador, Experto y Composite; los cuales permitieron mejor diseño de la arquitectura y mucho más adaptable a posibles cambios. Se obtuvo con la realización de los diagramas de secuencia una sucesión de pasos lógicos a realizar para cada caso de uso, con el objetivo de hacer más directa y precisa la implementación del sistema.

Capítulo 4.
Implementación y Pruebas del Sistema

El proceso de implementación comienza con el resultado producido por el diseño y su implementación es basándose en componentes. En el presente capítulo se muestra el total desarrollo del sistema, buscándose como objetivo principal la consecución de un prototipo funcional de la aplicación. Se representa cómo quedó implementada la aplicación y se muestran las principales pantallas de la misma con sus descripciones.

A la herramienta desarrollada se le aplicaron pruebas de aceptación con el objetivo de verificar las funcionalidades del sistema. Para esto se utilizó el método de caja negra usando la técnica de partición de equivalencia. Se realizó un experimento comparativo con el objetivo de verificar el correcto funcionamiento del sistema comprobándolo con un software profesional.

4.1 Modelo de Implementación

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. Este artefacto describe cómo se implementan los componentes, congregándolos en subsistemas organizados en capas y jerarquías, y señala las dependencias entre estos.

4.1.1 Diagrama de Componentes

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. Representa un fragmento de un sistema software que puede ser ensamblado con otros fragmentos para formar piezas más grandes o aplicaciones completas. Los diagramas de componentes muestran los componentes del software por los que está compuesto, como los archivos de código fuente y las bibliotecas. A continuación se muestran los diagramas de componentes realizados para dar paso a la implementación del sistema. Los componentes se pueden agrupar en 3 paquetes principales, siendo estos “Vista”, “Controlador” y “Modelo”, en correspondencia con el patrón de arquitectura utilizado Modelo-Vista-Controlador.

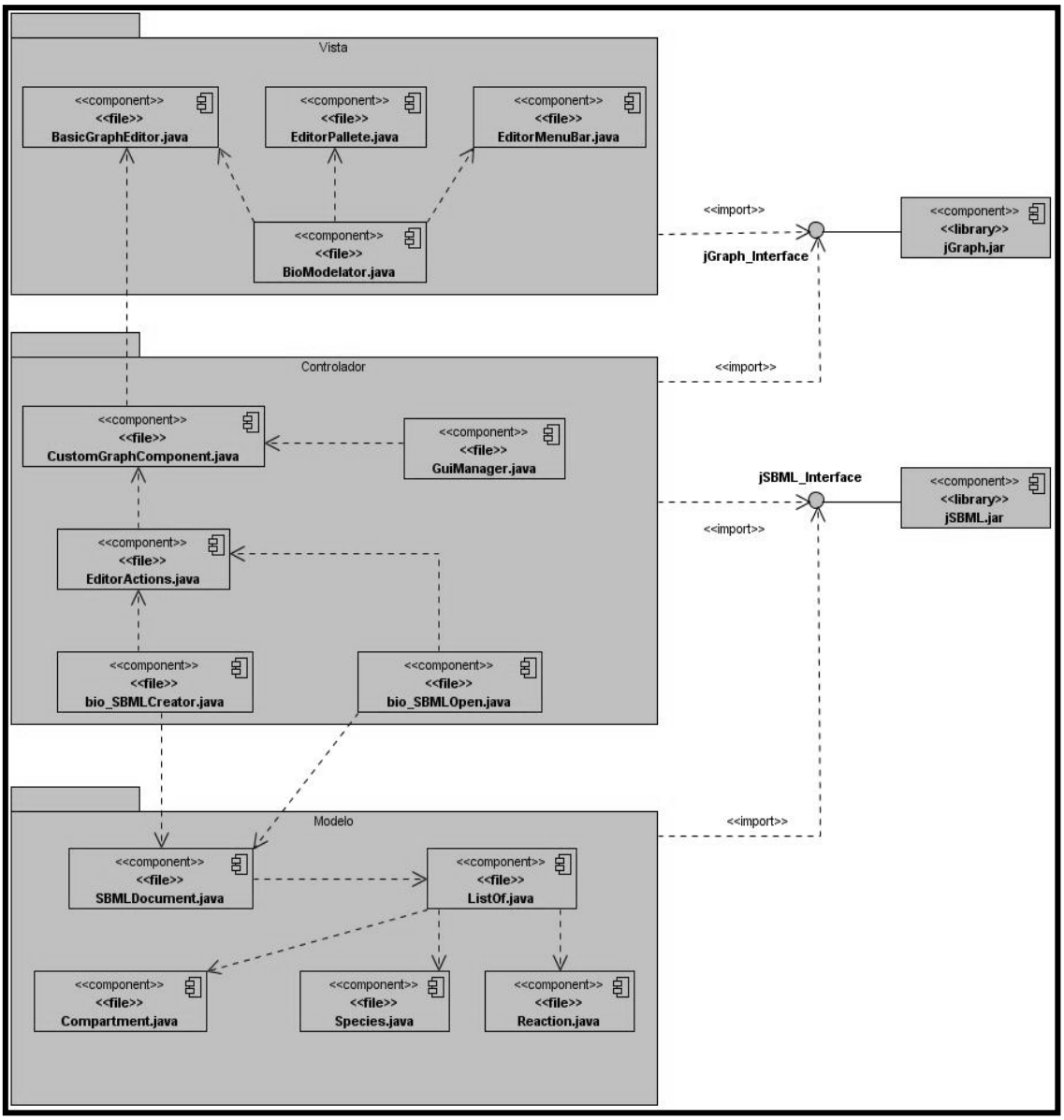


Figura 14. Diagrama de Componentes del Caso de Uso Modelar Gráficamente Sistemas Biológicos

El diagrama de componentes resultó estructurado como se muestra en la Figura 14. El mismo quedó fraccionado en subsistemas de implementación, coincidiendo algunos con las capas definidas por el patrón MVC. El diagrama de componentes está compuesto también por las bibliotecas jGraph y jSBML, de las cuales se importaron funcionalidades para facilitar el trabajo con el sistema.

Subsistema Vista: Contiene los archivos de código fuente encargados de mostrar y manejar todo lo referente a la interfaz de usuario, así como la captura de los eventos realizados por el usuario y el tratamiento que se le debe dar a cada uno.

Subsistema Controlador: En este subsistema se evidencian los códigos fuente de las clases controladoras que se encargan del trabajo con cada una de las funcionalidades requeridas por el cliente, sirviendo de enlace entre los subsistemas Vista y Modelo

Subsistema Modelo: Cuenta con un grupo de clases pertenecientes a la biblioteca jSBML. Las mismas fueron objeto de redefinición por la necesidad de agregarles datos necesarios para el trabajo con la aplicación.

jGraph: Es una biblioteca de código abierto en Java para la visualización de gráficos. Es utilizada en el sistema por sus prestaciones y funcionalidades que permiten con alguna facilidad el trabajo con los gráficos.

jSBML: Es una biblioteca destinada al trabajo con los archivos SBML puesto que encapsula un grupo de funcionalidades y clases, de las cuales se redefinieron algunas y otras simplemente se instanciaron o se utilizaron funciones pertenecientes a la misma.

4.2 Modelo de Despliegue

Los Diagramas de Despliegue muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación.

Para aplicaciones como la que se propone, que se ejecuta en una sola máquina y todos los dispositivos con que se relaciona son los estándares (teclado, mouse), el diagrama de despliegue va a estar constituido por un nodo.

4.3 Pantallas de la aplicación

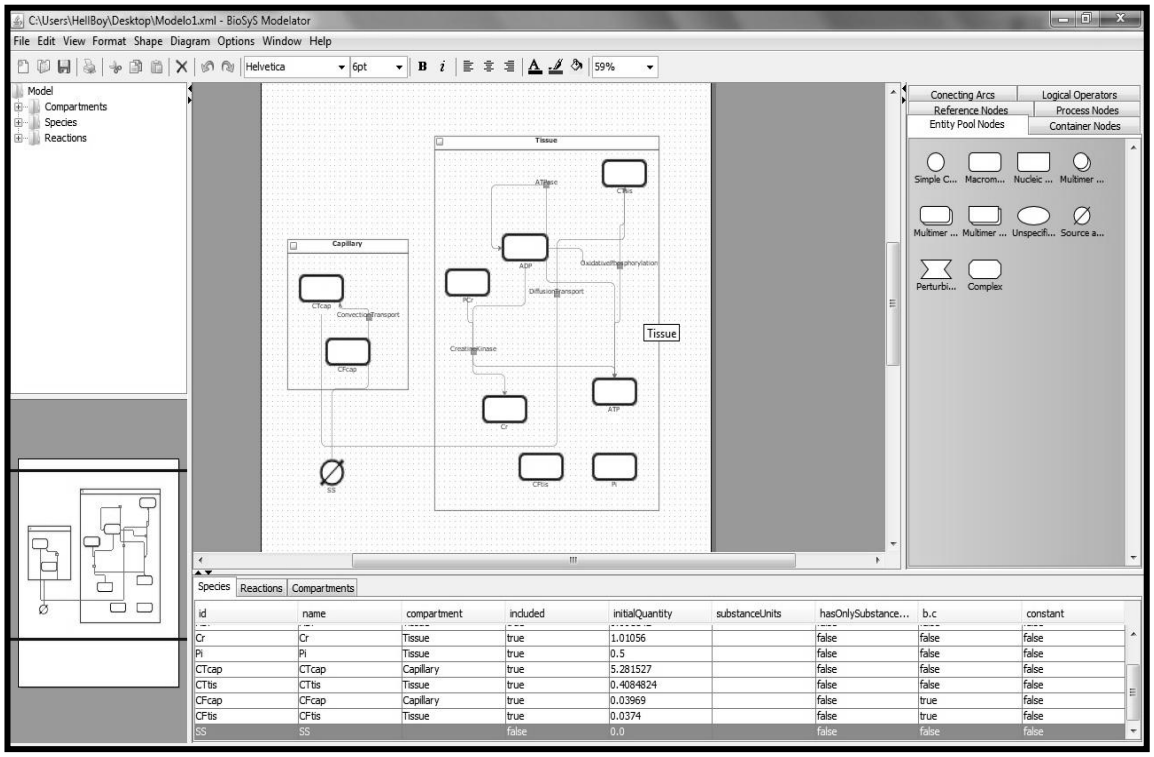


Figura 15. Prototipo Funcional del Modelador Gráfico de Sistemas para alasBioSyS

Figura 16. Principales áreas de la interfaz del Modelador Gráfico para alasBioSyS.

4.4 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 determinadas situaciones.

Las pruebas de software son una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación.

A la herramienta desarrollada se le aplicaron pruebas de aceptación con el objetivo de verificar las funcionalidades del sistema. Para esto se utilizó el método de caja negra usando la técnica de partición de equivalencia.

4.4.1 Pruebas de Caja Negra

La prueba de Caja Negra se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten que a partir de un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un sistema. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema. [25]

A partir de la aplicación de estas pruebas se pueden obtener los siguientes errores:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación. [25]

Para preparar los casos de pruebas hacen falta datos que ayuden a la ejecución de los estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa. Según si lo que se desea es, hallar un error, o probar una funcionalidad. Los datos

se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del sistema, a fin de verificar su completo funcionamiento.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:
***Técnica de Partición de Equivalencia*:** esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

Técnica del Análisis de Valores Límites: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Dentro del método de Caja Negra la técnica de la Partición de Equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software.

4.4.1.1 Técnica de Partición de Equivalencia

Una partición equivalente es una técnica de prueba de Caja Negra que divide el dominio de entrada de un programa, en clases de datos, de los que se pueden derivar casos de prueba. El diseño de estos casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia.

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. De este modo es capaz de descubrir de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar. [25]

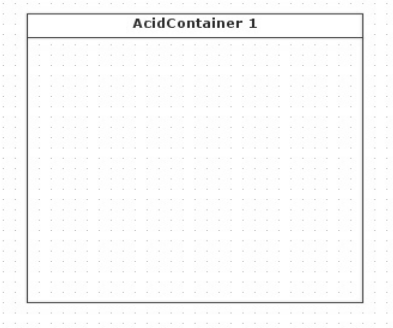
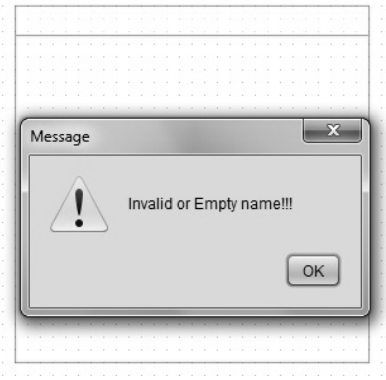
A continuación se presentan los casos de prueba realizados a 2 casos de uso. La totalidad de estos casos de prueba se encuentran en el artefacto Casos de Prueba contenido dentro del expediente de proyecto.

4.4.2 Casos de Prueba

4.4.2.1 Caso de Uso Administrar Compartimiento

4.4.2.1.1 Sección Insertar Compartimiento

Tabla 6: Caso de Prueba para el CU Administrar Compartimiento, escenario Insertar Compartimiento.

Escenario	Descripción	Variable Nombre	Respuesta del sistema
EC Insertar Compartimiento con nombre válido	Permite al investigador insertar un compartimiento.	AcidContainer	
		Acid Container	
		1- Acid Container	
		Acid Container 1	
EC Insertar Compartimiento con nombre inválido	Error al intentar insertar nuevo compartimiento.	AcidComp@rt	
		AcidCompart#1	
		Acid^Cells	
		AcidCell$	
EC Insertar Compartimiento con nombre vacío	Error al intentar insertar nuevo compartimiento.	“ ”	

4.4.2.1.2 Sección Eliminar Compartimiento

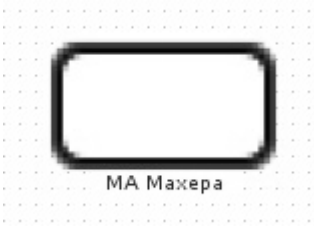
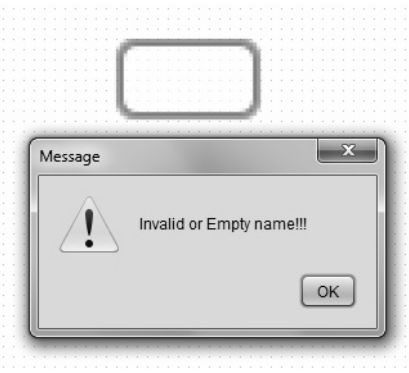
Tabla 7: Caso de Prueba para el CU Administrar Compartimiento, escenario Eliminar Compartimiento**.**

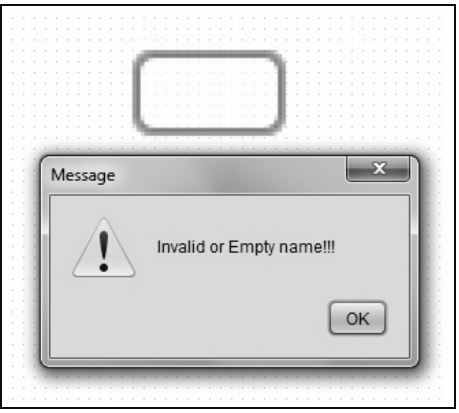
Escenario	Descripción	Variable	Respuesta del sistema
EC Eliminar Compartimiento vacío	Permite al investigador eliminar un compartimiento.		El compartimiento antes mostrado en el área de trabajo se elimina completamente de ella. Además se actualiza la barra de propiedades eliminando dicho compartimiento.
EC Eliminar Compartimiento con especies y reacciones incluidas	Permite al investigador eliminar un compartimiento con todo lo que contiene.		El compartimiento antes mostrado en el área de trabajo se elimina completamente de ella, en conjunto con las especies y reacciones que contiene. Además se actualiza la barra de propiedades eliminando dicho compartimiento, dichas especies y dichas reacciones.

4.4.2.2 Caso de Uso Administrar Especie

4.4.2.2.1 Sección Insertar Especie

Tabla 8: Caso de Prueba para el CU Administrar Especie, escenario Insertar Especie.

Escenario	Descripción	Variable Nombre	Respuesta del sistema
EC Insertar Especie con nombre válido	Permite al investigador insertar una especie.	MAMaxepa	
		MA Maxepa	
		1- MA Maxepa	
		MA Maxepa 1	
EC Insertar Especie con nombre inválido	Error al intentar insertar nueva especie	M@Maxepa	
		MAMaxepa #1	
		MA^Maxepa	
		MAMaxepa$	

EC Insertar Especie con nombre vacío	Error al intentar insertar nueva especie	“ ”	

4.4.2.2.2 Sección Eliminar Especie

Tabla 9: Caso de Prueba para el CU Administrar Especie, escenario Eliminar Especie.

Escenario	Descripción	Variable	Respuesta del sistema
EC Eliminar Especie	Permite al investigador eliminar una especie.		La especie antes mostrada en el área de trabajo se elimina completamente de ella. Además se actualiza la barra de propiedades eliminando dicha especie.
EC Eliminar Especie relacionada con otras especies	Permite al investigador eliminar una especie y las reacciones en las que está presente.		La especie antes mostrada en el área de trabajo se elimina completamente de ella, en conjunto con las reacciones en las que forma parte. Además se actualiza la barra de propiedades, eliminando dichas especie y reacciones.

4.4.3 Descripción de Variables

Tabla 10: Descripción de variables para los casos de prueba.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de Texto	No	El campo puede estar compuesto tanto por números, como por letras en cualquier orden definible. Solo posee las restricciones de valor nulo y de la inserción de caracteres especiales.

4.4.4 No Conformidades Detectadas

Tabla 11: No conformidades detectadas en los casos de prueba.

No	Elemento	No Conformidad	Aspecto Correspondiente	Etapa de Detección	Significativo
1	Interfaz	No actualización del árbol de distribución jerárquica.	Inserción de una nueva reacción química.	Etapa de aplicación de pruebas a la aplicación.	X
2	Interfaz	Falta el mensaje de error ante nombres inválidos con caracteres especiales.	Inserción de un nuevo compartimiento.	Etapa de aplicación de pruebas a la aplicación.	X
3	Interfaz	Falta el mensaje de error ante nombres inválidos con caracteres especiales.	Inserción de una nueva especie.	Etapa de aplicación de pruebas a la aplicación.	X
4	Interfaz	Falta el mensaje de error ante nombres inválidos con caracteres especiales.	Inserción de una nueva reacción química.	Etapa de aplicación de pruebas a la aplicación.	X
5	Interfaz	Falta el mensaje de error ante nombres inválidos con caracteres especiales.	Inserción de un nuevo operador lógico.	Etapa de aplicación de pruebas a la aplicación.	X

4.5 Experimento Comparativo

En vista de comparar el resultado del funcionamiento del sistema, específicamente en la construcción de un modelo SBML correcto, cumpliendo con el formato y estructuración XML se realizó un experimento comparativo en el cual se modela un sistema biológico utilizando el Modelador Gráfico, se exporta a formato SBML y es cargado por un software profesional para el trabajo con estos modelos.

4.5.1 Software y modelo utilizado

Para el desarrollo de este experimento se hizo necesario el uso de elementos externos al sistema, los cuales son descritos a continuación:

- **CellDesigner:** El software utilizado para el desarrollo de este experimento es un editor de diagramas estructurados para la elaboración de modelos de sistemas biológicos y es uno de los software para la modelación de estos sistemas con más número de usuarios a nivel mundial, dicho software es descrito con mayor detalle en el Capítulo 1.
- **BioModels Database**: Es un repositorio de estudios internacionales descritos mediantes modelos computacionales. Dichos modelos son principalmente del campo de la Biología de Sistemas. De esta base de datos fue descargado el modelo que se utilizó.
- **Descripción del modelo"Lai2007_O2_Transport_Metabolism":** El modelo descargado describe la versión SBML de un modelo de sistema biológico con el nombre: Vinculación de consumo de oxígeno pulmonar, la utilización del oxígeno muscular y el metabolismo celular durante el ejercicio, publicado por la revista Ann Biomed Eng el 35 de Junio de 2007, con ID de Publicación Médica 17380394. Este modelo simula el transporte de oxígeno y el metabolismo en el músculo esquelético, en respuesta al cambio de un estado estacionario de calentamiento a un ritmo de trabajo más alto, correspondiente al ejercicio en los diferentes niveles de intensidad: moderada, pesado y muy pesado. [26]

4.5.2 Desarrollo del experimento

El modelo Lai2007_O2_Transport_Metabolism fue representado en el Modelador Gráfico, utilizando como guía la figura presentada en el artículo publicado en la revista Ann Biomed Eng, referenciado anteriormente, quedando de la siguiente forma:

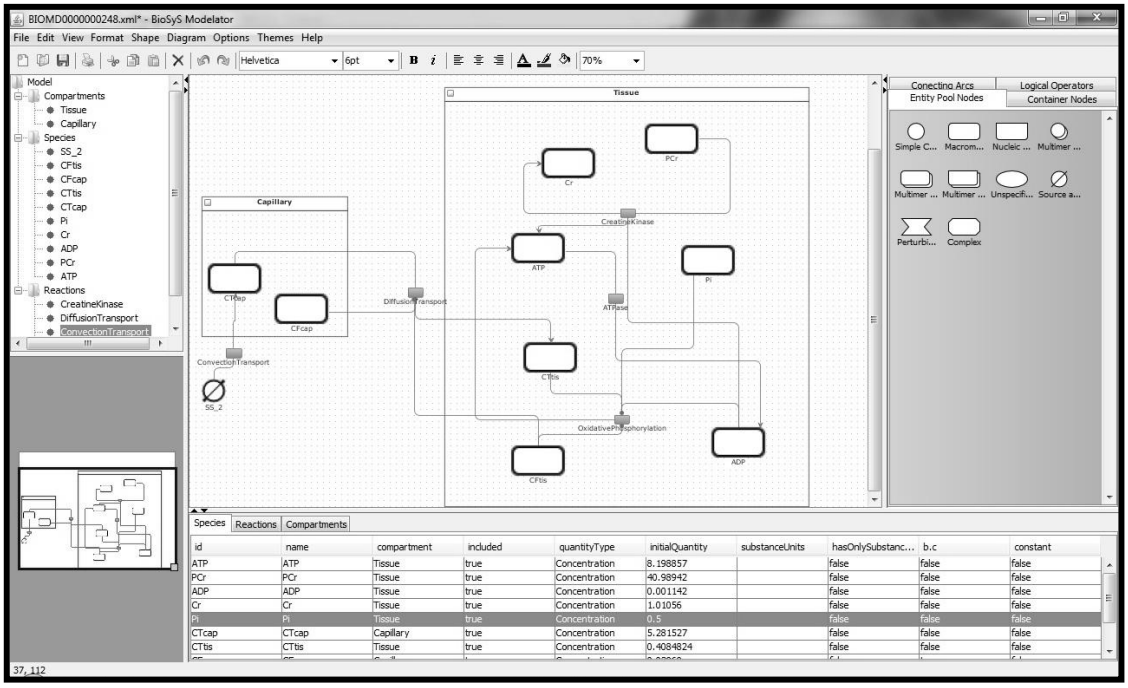


Figura 17. Modelo Lai2007_O2_Transport_Metabolism representado en el Modelador Gráfico.

Se procedió a exportar este modelo a formato SBML y a importarlo desde el software CellDesigner con el objetivo de comprobar la correcta creación del fichero SBML por parte del Modelador Gráfico. Dicho software modeló el fichero exportado de la siguiente forma:

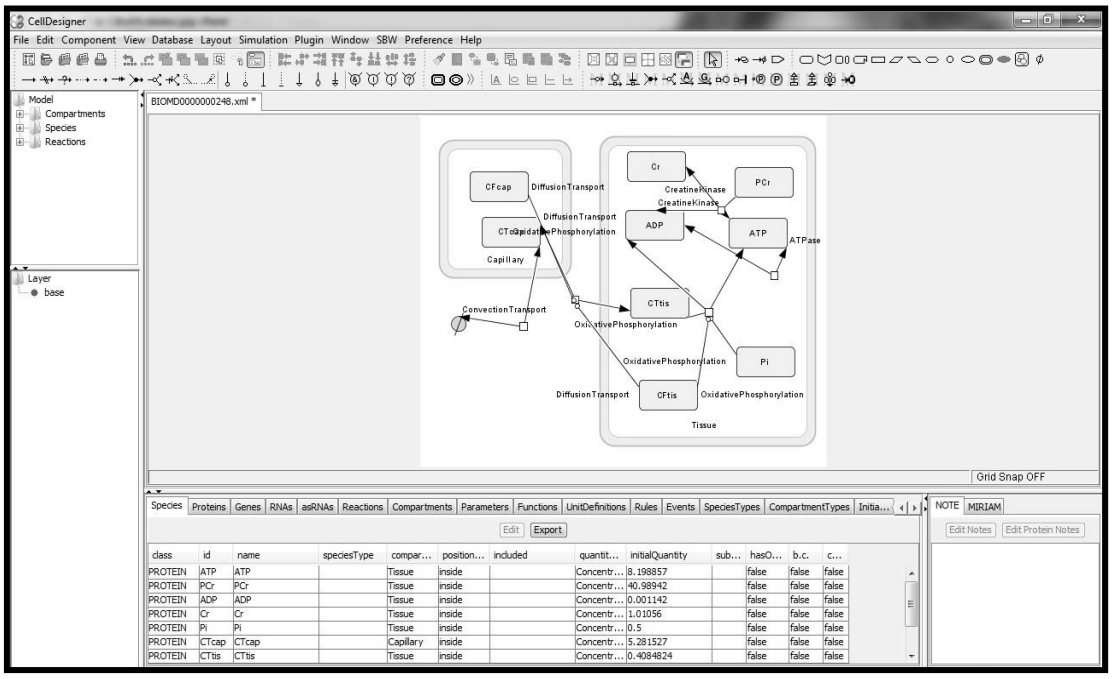


Figura 18. Modelo Lai2007_O2_Transport_Metabolism representado por CellDesigner.

4.5.3 Conclusiones del experimento

Luego de la realización de este experimento se constató que CellDesigner grafica satisfactoriamente el modelo. Representa la totalidad de especies, compartimientos y reacciones que contiene el fichero generado por el Modelador Gráfico, así como las propiedades de cada uno de ellos. Sin embargo difieren en cuanto al modo de situar los elementos en el área de trabajo, puesto que CellDesigner posee varios algoritmos avanzados de ordenamiento gráfico, que facilitan la ubicación automática de los elementos de los modelos.

4.6 Conclusiones

Mediante la realización del presente capítulo se describió cómo fue implementada la herramienta en términos de componentes, los cuales fueron separados por paquetes como fue definido en el modelo de diseño. Se precisaron las áreas de la interfaz gráfica con las cuales contaría el prototipo funcional de la aplicación. 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. De este modo se obtuvieron 5 no conformidades las cuales fueron registradas en el artefacto Casos de Prueba y solucionadas para un mejor funcionamiento del software. Se desarrolló una prueba comparativa, para comprobar los resultados de la aplicación con respecto al CellDesigner, en la cual se logró crear un fichero SBML acertado en su formato y legible por este software profesional. Se evidenció que además de importar el modelo eficientemente, CellDesigner posee algoritmos avanzados de ordenamiento gráfico que permiten una mayor comprensión del modelo sin necesidad de organizarlo manualmente, considerándose útil para cualquier modelador gráfico de sistemas biológicos.

Conclusiones

Luego de todo lo expuesto en el presente trabajo de diploma, se concluye que:

- Se seleccionó del estándar de modelación SBML para ser usado por la herramienta, según el cual se definieron las funcionalidades para el desarrollo de la misma.

- Se diseñaron las funcionalidades y se implementó una herramienta que permite modelar gráficamente sistemas biológicos, cumpliendo con el estándar SBML.

- Se validaron las funcionalidades de la herramienta y se refinaron cada una de las no conformidades detectadas.

- Se realizó una prueba comparativa del funcionamiento de la herramienta con respecto al software profesional CellDesigner, lográndose la creación de un fichero SBML legible por el mismo.

Recomendaciones

- Integrar el Modelador Gráfico de Sistemas Biológicos obtenido como resultado de la investigación, a la plataforma alasBioSyS, a través de la herramienta System Biology Workbench (SBW) utilizando el SBW Core.

- Incorporar el trabajo con el estándar SBGN a la aplicación, para lograr una mayor polivalencia de la misma.

- Implementar un algoritmo de ordenamiento gráfico para brindar una organización automática y óptima a los modelos representados.

Referencias Bibliográficas

[1] Hagen, JB. The origin of bioinformatics. s.l.: Nat Rev Genet, 2000. 231-6.

[2] Nurse, Paul. Systems biology: understanding cells. New York: Nature, 2003. 6951.

[3] Mulet, Yunet González. Metodología para el análisis de estabilidad de Sistemas de Ecuaciones Diferenciales n-dimensionales. Ciudad de la Habana: s.n., 2010.

[4] Trelles, Oswaldo. Bioinformática Básica. Málaga: Universidad de Málaga.

[5] Mayorga, Luis S. Biología de Sistemas y su aplicación en Biología Celular. Cuyo. Argentina: Revista Médica Universitaria. Universidad Nacional de Cuyo, 2008. 1069 8881.

[6] Y. Sanchez, Y. Armentero. Software para la Simulación de Sistemas Biológicos: Módulo de modelación gráfica de Sistemas Biológicos. Ciudad de la Habana: Universidad de las Ciencias Informáticas, 2007.

[7] Francisco Montero, Federico Morán. Biofísica. Biofísica: Procesos de Autoorganización en Biología. Madrid: EUDEMA, SA , 1992.

[8] About CellML. CellML.org. [En línea] University of Auckland, 25 de Noviembre de 2011. [Citado el: 29 de Noviembre de 2011.] http://www.cellml.org/about.

[9] An overview of the CellML API and its implementation. Andrew K Miller, Justin Marsh, Adam Reeve, Alan Garny, Randall Britten, Matt Halstead, Jonathan Cooper, David P Nickerson and Poul F Nielsen. Oxford : BMC Bioinformatics, 2010.

[10] BioPAX - Biological Pathway Exchange. BioPAX.org. [En línea] 20 de Noviembre de 2011. [Citado el: 29 de Noviembre de 2011.] http://www.biopax.org.

[11] The BioPAX community standard for pathway data sharing. Emek Demir, Michael P Cary, Suzanne Paley, Ken Fukuda, Christian Lemer, Imre Vastrik, Guanming Wu, Peter D'Eustachio, Carl Schaefer, Joanne Luciano, Frank Schacherer, Irma Martinez-Flores, Zhenjun Hu, Veronica Jimenez-Jacinto, Geeta Joshi-Tope, Kumaran. s.l. : Nature Biotechnology, 2012.

[12] The Systems Biology Graphical Notation. Fedor Kolpakov, Augustin Luna, Sven Sahle, Esther Schmidt, Steven Watterson, Guanming Wu, Igor Goryani, Douglas B Kell, Chris Sander, Herbert Sauro, Jacky L Snoep, Kurt Kohn & Hiroaki Kitano. s.l. : Nature Biotechnology, 2009.

[13] SBGN About. System Biology Graphical Notation. [En línea] SBGN.org, 28 de Noviembre de 2011. [Citado el: 29 de Noviembre de 2011.] http://www.sbgn.org/About.

[14] SBML About. The System Biology Markup Languaje. [En línea] SBML.org, 29 de Noviembre de 2011. [Citado el: 20 de Noviembre de 2011.] http://sbml.org/SBML.org:About

[15] The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita. s.l. : BioInformatics, 2003.

[16] CellDesigner.org. CellDesigner.org. [En línea] CellDesigner, 28 de Noviembre de 2011. [Citado el: 29 de Noviembre de 2011.] http://www.celldesigner.org.

[17] CellWARE. CellWARE. [En línea] Singapore Bioinformatics Institute, 27 de Noviembre de 2011. [Citado el: 29 de Noviembre de 2011.] http://www.bii.a-star.edu.sg/achievements/applications/cellware.

[18] COPASI: biochemical network simulator. COPASI.org. [En línea] COPASI.org, 29 de Noviembre de 2011. [Citado el: 29 de Noviembre de 2011.] http://www.copasi.org.

[19] Sauro, H. M. ; Hucka, M. ; Finney, A. ; Wellock, C. ; Bolouri, H. ; Doyle, J. ; Kitano, H.: Next Generation Simulation Tools: The Systems Biology Workbench and BioSPICE Integration. In: OMICS A Journal of Integrative Biology 7 (2003), S. 353–370.

[20] Balduino, Ricardo. Introduction to OpenUP (Open Unified Process). s.l. : IBM Rational Software, 2007.

[21] Larman, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objetos. Mexico DF: Prentice Hall Inc., 1999. 970-17-0261.

[22] Flanagan, David. Java en pocas palabras. s.l.: McGraw-Hill Interamericana Editores, 1999. 970-10-2070-7.

[23] Ramírez, Alejandro. Subversion. 2004.

[24] Erich Gamma, Richard Helm, Ralph Johnson, John Vlisside. Design Patterns. Elements of Reusable Object-Oriented Software. s.l.: Addison-Wesley, 1995.

[25] Pressman, Roger S. Ingeniería de Software. Un enfoque práctico. s.l. : McGraw-Hill Companies, 2002. 8448132149.

[26] Vinculación de consumo de oxígeno pulmonar, la utilización del oxígeno muscular y el metabolismo celular durante el ejercicio. Lai N, Camesasca M, Saidel GM, Dash RK, Cabrera ME. Cleveland,OH : Ann Biomed Eng, 2007.

Bibliografía

[1] Francisco Montero, Federico Morán. Biofísica. Biofísica: Procesos de Autoorganización en Biología. Madrid : EUDEMA, SA , 1992.

[2] About CellML. CellML.org. [En línea] University of Auckland, 25 de Noviembre de 2011. [Citado el: 29 de Noviembre de 2011.] http://www.cellml.org/about.

[3] BioPAX - Biological Pathway Exchange. BioPAX.org. [En línea]

[4] BioPAX - Biological Pathway Exchange. BioPAX.org. [En línea] 20 de Noviembre de 2011. [Citado el: 29 de Noviembre de 2011.] http://www.biopax.org/.

[5] SBGN About. System Biology Graphical Notation. [En línea] SBGN.org, 28 de Noviembre de 2011. [Citado el: 29 de Noviembre de 2011.] http://www.sbgn.org/About.

[6] CellDesigner.org. CellDesigner.org. [En línea] CellDesigner, 28 de Noviembre de 2011. [Citado el: 29 de Noviembre de 2011.] http://www.celldesigner.org.

[7] CellWARE. CellWARE. [En línea] Singapore Bioinformatics Institute, 27 de Noviembre de 2011. [Citado el: 29 de Noviembre de 2011.] http://www.bii.a-star.edu.sg/achievements/applications/cellware/.

[8] COPASI: biochemical network simulator. COPASI.org. [En línea] COPASI.org, 29 de Noviembre de 2011. [Citado el: 29 de Noviembre de 2011.] http://www.copasi.org/.

[9] Balduino, Ricardo. Introduction to OpenUP (Open Unified Process). s.l. : IBM Rational Software, 2007.

[10] Larman, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objetos. Mexico DF : Prentice Hall Inc., 1999. 970-17-0261.

[11] Aderem, Alan. Systems Biology: Its Practice and Challenges . Washington : The Institute for Systems Biology, 2005.

[12] Flanagan, David. Java en pocas palabras. s.l. : McGraw-Hill Interamericana Editores, 1999. 970-10-2070-7.

[13] Hagen, JB. The origin of bioinformatics. s.l. : Nat Rev Genet, 2000. 231-6.

[14] Marta López, Gema Ruiz Romero, Miguel Vega. Biología de Sistemas. Informe de Vigilancia Tecnológica. Madrid. España : Genoma España, 2007. 84-609-9762-6.

[15] Mayorga, Luis S. Biología de Sistemas y su aplicación en Biología Celular. Cuyo. Argentina : Revista Médica Universitaria. Universidad Nacional de Cuyo, 2008. 1069 8881.

[16] Mulet, Yunet González. Metodología para el análisis de estabilidad de Sistemas de Ecuaciones Diferenciales n-dimensionales . Ciudad de la Habana : s.n., 2010.

[17] Nurse, Paul. Systems biology: understanding cells. New York : Nature, 2003. 6951.

[18] Trelles, Oswaldo. Bioinformática Básica. Málaga : Universidad de Málaga.

[19] Y. Sanchez, Y. Armentero. Software para la Simulación de Sistemas Biológicos: Módulo de modelación gráfica de Sistemas Biológicos. Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2007.

[20] SBML About. The System Biology Markup Languaje. [En línea] SBML.org, 29 de Noviembre de 2011. [Citado el: 20 de Noviembre de 2011.] http://sbml.org/SBML.org:About.

[21] Erich Gamma, Richard Helm, Ralph Johnson, John Vlisside. Design Patterns. Elements of Reusable Object-Oriented Software. s.l. : Addison-Wesley, 1995.

[22] Ramírez, Alejandro. Subversion. 2004.

[23] An overview of the CellML API and its implementation. Andrew K Miller, Justin Marsh, Adam Reeve, Alan Garny, Randall Britten, Matt Halstead, Jonathan Cooper, David P Nickerson and Poul F Nielsen. Oxford : BMC Bioinformatics, 2010.

[24] The BioPAX community standard for pathway data sharing. Emek Demir, Michael P Cary, Suzanne Paley, Ken Fukuda, Christian Lemer, Imre Vastrik, Guanming Wu, Peter D'Eustachio,

Carl Schaefer, Joanne Luciano, Frank Schacherer, Irma Martinez-Flores, Zhenjun Hu, Veronica Jimenez-Jacinto, Geeta Joshi-Tope, Kumaran. s.l. : Nature Biotechnology, 2012.

[25] The Systems Biology Graphical Notation. Fedor Kolpakov, Augustin Luna, Sven Sahle, Esther Schmidt, Steven Watterson, Guanming Wu, Igor Goryani, Douglas B Kell, Chris Sander, Herbert Sauro, Jacky L Snoep, Kurt Kohn & Hiroaki Kitano. s.l. : Nature Biotechnology, 2009.

[26] The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita. s.l. : BioInformatics, 2003.

[27] Pressman, Roger S. Ingeniería de Software. Un enfoque práctico. s.l. : McGraw-Hill Companies, 2002. 8448132149.

[28] Climent, Miguel LLoret. Sistemas vivos y sus modelos matemáticos. Modelización de un ecosistema. Alicante : Universidad de Alicante, 1996.

[29] Vázquez, Antonio Cruz. Modelado y Simulación de sistemas biológicos. Barcelona : Universidad Autónoma de Barcelona, 2009.

[30] Octavio Miramontes, Bartolomé Luque. Biología de Sistemas, Física y Fenómenos Colectivos. Mexico D.F. : Medigraphic Artemisa, 2007.

[31] Vinculación de consumo de oxígeno pulmonar, la utilización del oxígeno muscular y el metabolismo celular durante el ejercicio. Lai N, Camesasca M, Saidel GM, Dash RK, Cabrera ME. Cleveland,OH : Ann Biomed Eng, 2007.

Glosario de Términos

alasBioSyS: Biological System Simulator. (Simulador de Sistemas Biológicos, en inglés) Software para la simulación de sistemas biológicos.

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.

CRUD: Acrónimo de Crear, Obtener, Actualizar y Borrar (del original en inglés: Create, Read, Update and Delete). Es usado para referirse a las funciones básicas que se realizan sobre los datos persistentes.

SBGN: Notación Gráfica para Sistemas Biológicos, por sus siglas en inglés. Estándar de notación gráfica para la modelación de sistemas biológicos

SBML: Lenguaje de Marcado para Sistemas Biológicos, por sus siglas en inglés. Estándar de modelación de sistemas biológicos basado en formato XML

Sistema Biológico: Sistema abierto que opera en condiciones alejadas del equilibrio termodinámico, con muchas y fuertes interacciones no lineales entre sus muchos elementos.

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.