UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

Software para el Estudio de Sistemas Biológicos versión 2.0

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores: Carlos González Iglesias

Edel Moreno Lemus

Tutores: MSc. Yunet González Mulet

Ing. Julio Antonio Villaverde Martínez

Co-Tutor: MSc. Jorge Luis Vázquez González

CIUDAD DE LA HABANA

JUNIO 2010

"Lo que caracteriza al hombre de ciencia no es la posesión del conocimiento o de verdades irrefutables,

sino la investigación desinteresada e incesante de la verdad."

KARL POPPER

Declaración de Autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos al grupo de Bioinformática del centro DATEC, así como a la Universidad de las Ciencias Informáticas, para que hagan el uso que estimen pertinente con este trabajo. Para que así conste firmamos la presente a los días del mes de del año .

Carlos González Iglesias

Firma del Autor

Edel Moreno Lemus

Firma del Autor

MSc. Yunet González Mulet

Firma del Tutor

Ing. Julio Antonio Villaverde Martínez

Firma del Tutor

MSc. Jorge Luis Vázquez González

Firma del Co-Tutor

DATOS DE CONTACTOS

Tutores:

MSc.Yunet González Mulet.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Email: ygonzalezmu@uci.cu

Ing. Julio Antonio Villaverde Martínez.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Email: jvillaverde@uci.cu

Co-Tutor:

MSc. Jorge Luis Vázquez González.

Universidad de las Ciencias Informáticas, Habana, Cuba.

Email: jlvazquez@uci.cu

AGRADECIMIENTOS

Queremos agradecer primeramente a nuestros tutores, Julio, Yunet y Jorge, por brindar de forma desinteresada sus conocimientos y ayudarnos a estar hoy aquí. Agradecer a nuestros padres que representan los más valioso en nuestras vidas y a toda nuestra familia por su comprensión y ayuda en todo momento. A todos aquellos que tuvieron que ver de una forma u otra con nuestra formación profesional, los profesores de todas las enseñanzas, los de la vocacional, los de la universidad, en fin, a todos, muchas gracias. A todas nuestras amistades por los buenos momentos que pasamos y los recuerdos que hoy llevamos con nosotros, en especial a los que más de cerca han compartido con nosotros y nos han llegado a conocer un poquito mejor. A Noel, por ser el precursor de esta idea, además de ser hermano y amigo en todo momento. Al grupo de Bioinformática de nuestra facultad, por todo su apoyo y ayuda incondicional. A nuestro Comandante en Jefe Fidel por darnos la oportunidad de estudiar en esta universidad de excelencia y hacer realidad todos nuestros sueños.

DEDICATORIA

A mi madre, por ser símbolo de amor, dedicación, comprensión, preocupación, y ser lo más importante que tengo en la vida.

A mi padre, por ser no sólo un padre incondicional sino la guía intelectual, el apoyo en los estudios, el ejemplo a seguir y por ser el máximo contribuyente a que hoy sea la persona que soy.

A mi hermana que la quiero con la vida.

A mi tía Aya, por mimarme tanto y ser como mi segunda madre.

A mis abuelos por entregarme su amor más sincero e incondicional.

A mi tía Mary y a mi tío Migue, por darme todo su amor y su apoyo en todo momento.

A mi chuchichuchi, por entregarme tu amor y ser mi apoyo en estos dos años. Te amo...

A mis hermanos Fidel, Rafael, Jesús, Hamlet por estar junto a mí en los momentos buenos y en los malos.

A Silvia Rita y al grupo Infodanz por dejarme formar parte de ese excelente grupo y enseñarme valores que nunca olvidaré.

A mis amigos de la Universidad, Yoel, Elienny, Angel, Yuli, Juan Carlos, Rosnel, Pascal, Yoan, Reinier, Cuan, Manzano, por todos los buenos momentos vividos en estos últimos años.

A todas mis amistades de la UCI, a los que nunca olvidaré y siempre llevaré en mi corazón.

A mi amigo y compañero Edel, por aguantarme y entenderme estos cinco años, en especial este 5to año. Gracias mi hermano...

Carlos

A mi familia por apoyarme siempre.

A mi mamá y a mi papá por apoyarme en todo siempre, por estar pendientes de cada momento de mi vida, gracias a ustedes estoy aqui hoy.

A mi hermano por ser mi guía y mi ejemplo a seguir en la vida, y por ser el responsable de que yo este aquí hoy defendiendo esta tesis.

A mis abuelos por estar siempre pendientes de mí y estar orgullosos de sus nietos. En especial a mi abuelo Alberto que ya no está entre nosotros pero estaria muy orgulloso de verme aquí hoy, por ser mi primer maestro y ayudarme a dar mis primeros paso en la escuela.

A mis amigos de la Universidad, por todos los buenos momentos vividos en estos 5 años.

A los que tuvieron la suerte de compartir conmigo en Venezuela, fue una experiencia maravillosa.

A mi hermano Carlos, "casi se me olvida el pinareño...", por soportarme estos 5 años de Universidad.

Edel

RESUMEN

La Biología de Sistemas es un área del conocimiento compleja que pretende entender el funcionamiento de los sistemas como un todo y para ello, debe modelarlos usando diferentes herramientas y niveles de abstracción. Esto implica que difícilmente una sola persona, grupo o institución sean capaces de desarrollar por sí solos todo el software que hace falta. Es necesario crear sistemas que puedan extenderse con el desarrollo de nuevos módulos y que haciendo analogías con el software libre, permitan crear comunidades de desarrollo que enriquezcan los sistemas, convirtiéndolos en grandes plataformas computacionales. En este trabajo se presenta un software cuya arquitectura orientada a plug-ins, permite a cualquier persona o institución interesada hacer uso de los plug-ins ya elaborados o desarrollar nuevos para este software. El software presenta integrado varios plug-ins, uno de simulación, el cual permite al usuario realizar simulaciones locales o distribuidas, otro de graficación que permite realizar gráficas en 2D y 3D, así como uno de Base de Datos, brindando la posibilidad de gestionar la información generada en una base de datos.

Palabras Clave: Biología de Sistemas, Software, Plug-ins, Simulación, Graficación, Base de Datos

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La maduración de la Biología Molecular, las nuevas técnicas que dan lugar al acceso a miles de datos, el mayor poder computacional y los nuevos algoritmos, han cambiado la actitud de muchos científicos sobre cómo solucionar algunos de sus problemas, y ha propiciado el nacimiento de una nueva disciplina denominada Biología de Sistemas, capaz de integrar en su seno, a expertos procedentes de diferentes áreas como la Biología, la Informática, la Física, las Matemáticas y otras.

Desde hace años, el método utilizado por los científicos para estudiar un sistema biológico es el método reduccionista, es decir, la descomposición del sistema en sus partes constituyentes, las cuales por separado, son mucho más sencillas de entender y modelar.

Esta estrategia se muestra como una poderosa herramienta capaz de describir los sistemas físicos, porque proporciona un mapa completo de las piezas fundamentales, y de las funciones que éstas van realizando, pero no permite la comprensión de las interacciones entre las partes, o sea, conocer el funcionamiento del sistema como un todo. Este es el marco propicio para el surgimiento de la Biología de Sistemas, que hoy en día, resulta ser uno de los campos más activos, aprovechando no sólo los avances en el conocimiento de las partes fundamentales, sino también la posibilidad de realizar cálculos mucho más complejos y elaborar modelos informáticos gracias a los avances en las Ciencias de la Computación. [1]

La Biología de Sistemas trata de entender los sistemas biológicos a diferentes niveles de abstracción, desde el nivel molecular hasta los ecosistemas y haciendo uso de diferentes tipos de modelos matemáticos y técnicas computacionales, que van desde los Sistemas de Ecuaciones Diferenciales (SED) hasta la Minería de Datos, pasando por Cadenas de Markov, Redes Booleanas y Redes Bayesianas. [1]

Esta nueva ciencia se preocupa del estudio de procesos biológicos usando un enfoque sistémico, que tiene dos componentes fundamentales: uno experimental y uno computacional. El componente experimental proporciona los datos

necesarios para la creación y validación de los modelos matemáticos. En cambio, el componente computacional se divide en dos grandes grupos, la modelación y la minería o análisis de los datos ya existentes o de los generados durante las simulaciones. [2]

Aunque en ambas áreas se aprecia un desarrollo notable, ha sido el componente computacional el que más ha evolucionado. Se puede apreciar cómo la necesidad de desarrollar herramientas informáticas en este campo alcanza los sectores comerciales de la biotecnología.

Así, en los últimos años, en los Estados Unidos se han creado las primeras empresas relacionadas con el área de la Biología de Sistemas cuyos productos están intrínsecamente relacionados con la simulación y análisis de problemas biológicos. Ejemplo de tales empresas son: Gene Network Sciences (www.gnsbiotech.com, fundada 2000), Entelos (www.entelos.com, fundada 1996), Physiome (www.physiome.com, fundada 2001) y Genomatica(www.genomatica.com, fundada 2001). Todas estas empresas parten de la propiedad intelectual sobre una plataforma de software para integrar información, representar matemáticamente y simular sistemas biológicos. No obstante, siguen como estrategia de negocios la formación de alianzas comerciales con empresas biotecnológicas y no la comercialización directa del software.

En la actualidad, el desarrollo de software de forma independiente, también va en ascenso. Así, en el sitio web del SBML(www.sbml.org), se pueden encontrar referencias a muchos software basados en el estandar SBML, los cuales permiten modelar, simular o realizar análisis sobre modelos de sistemas biológicos.

Esta proliferación de herramientas resulta en gran variedad de funcionalidades e interfaces. Sin embargo, a pesar de lo bueno de este auge en el desarrollo de software, existen dos desventajas fundamentales [3]:

1. Cada herramienta usa su propio formato para salvar la información. El resultado es que los modelos salvados por un software determinado no pueden cargarse con otros software.

2. Muchas herramientas implementan las mismas funcionalidades. Escribir algoritmos para simulación es algo complejo y que toma bastante tiempo, es por ello que muchos software tienen tiempos de vida cortos, lo que significa que los autores no son capaces de implementar más que las funcionalidades básicas.

Otro resultado de este segundo problema es que la mayoría de las herramientas no implementan funcionalidades de análisis ni almacenamiento de información en bases de datos.

Por este motivo se decidió desarrollar, en conjunto con el Centro de Inmunología Molecular (CIM) un software para la simulación y análisis de sistemas biológicos que reutilizara algoritmos ya existentes para la simulación e implementara nuevas funcionalidades de análisis no disponibles en los software desarrollados hasta el momento. Adicionalmente, este

software incluía el almacenamiento en una Base de Datos de toda la información generada en los estudios realizados sobre los sistemas biológicos.

De esta forma se desarrolló la versión 1.0 de BioSyS con los siguientes módulos:

1. Editor de Ecuaciones
2. Base de Datos
3. Módulo de Simulación (haciendo uso del Matlab y de un algoritmo de ODEtoJava)
4. Módulo de Análisis (Análisis de los estados finales mediante el uso de técnicas de minería de datos)
5. Conexión a la Plataforma de Tareas Distribuidas (t-arenal) para realizar los cálculos computacionalmente intensos (múltiples simulaciones, bifurcaciones)

Sin embargo, tanto BioSyS como los demás software desarrollados dentro del área de la Biología de Sistemas, siguen teniendo serias deficiencias que se deben resolver para lograr un verdadero avance. Estas son:

1. La Biología de Sistemas es un área del conocimiento compleja que pretende entender el funcionamiento de los sistemas como un todo y para ello debe modelarlos usando diferentes herramientas y a diferentes niveles de abstracción. Por esta causa resulta muy complejo que una sola persona, grupo o institución sean capaces de desarrollar por sí solo todo el software que hace falta. Por lo tanto, es necesario crear sistemas que puedan extenderse con el desarrollo de nuevos módulos y haciendo analogías con el software libre, permitan crear comunidades de desarrollo que enriquezcan los sistemas, convirtiéndolos en grandes plataformas computacionales.
2. A diferencia de otras comunidades, existe poca cultura de reutilización de código dentro de la comunidad de biología de sistemas.

Para tratar de resolver estos dos problemas remanentes se desarrollaron varias iniciativas, dentro de las que destacan el Systems Biology Workbench (SBW) [3] y BioSPICE [4].

Sin embargo, estos dos frameworks se utilizan parcialmente porque desarrollar aplicaciones para cualquiera de los dos es un problema complejo. Es necesario conocer bien la arquitectura de estos sistemas, sólo se pueden desarrollar aplicaciones en los lenguajes para los que sus desarrolladores implementaron interfaces de programación de aplicaciones (APIs).

Teniendo en cuenta esta nueva problemática, dentro del proyecto BioSyS, se decidió diseñar una nueva arquitectura orientada a plug-ins, que basara el desarrollo de los módulos en la interacción de plug-ins, de forma tal que cualquier persona o institución interesada pudiera hacer uso de los plug-ins previamente desarrollados, o implementar otros nuevos para BioSyS, independientemente del sistema operativo y del lenguaje de programación en el que trabaje.

En el pasado año, en el proyecto BioSyS, se trabajó además en el desarrollo de nuevos módulos y se le adicionaron nuevas funcionalidades a otros de los módulos ya existentes, las cuales deben incorporarse en versiones futuras.

Por lo antes expuesto surge el presente trabajo de diploma, centrándose en el siguiente **problema científico:** La necesidad de un software flexible y extensible, que facilite el desarrollo de nuevas herramientas para la Biología de Sistemas.

Para resolver el problema planteado se define como **objeto de estudio** la Biología de Sistemas, enmarcando como **campo de acción** el desarrollo de software para Biología de Sistemas.

Se define como **Objetivo General**:

> **Desarrollar una nueva versión del software BioSyS que implemente las especificaciones de una nueva arquitectura del sistema que facilite el desarrollo de herramientas en el campo de la Biología de Sistemas.**

Desglosado en los siguientes **Objetivos Específicos**:

- **Desarrollar el plug-in de Base de Datos versión 2.0**
- **Desarrollar el plug-in de simulación versión 2.0**
- **Desarrollar el plug-in de graficación 2.0**
- **Integrar los nuevos plug-ins al sistema**
- **Realizar las pruebas unitarias de los plug-ins implementados**

Para alcanzar dichos objetivos se planteó desarrollar las siguientes **tareas**:

- Revisión bibliográfica de sistemas manejadores de plug-ins.
- Selección de la herramienta manejadora de plug-ins a utilizar.
- Definición de los nuevos plug-ins y funcionalidades a incorporar en la versión 2.0.
- Realización del análisis y diseño de la aplicación.
- Realización del diseño de los prototipos de interfaz de usuario para BioSyS versión 2.0.
- Análisis del módulo de base de datos versión 2.0.

- Diseño del módulo de base de datos versión 2.0.
- Implementación del módulo de base de datos versión 2.0.
- Análisis del módulo de simulación versión 2.0.
- Diseño del módulo de simulación versión 2.0.
- Implementación del módulo de simulación versión 2.0.
- Análisis, diseño e implementación del módulo de graficación versión 2.0.
- Diseño del módulo de graficación versión 2.0.
- Implementación del módulo de graficación versión 2.0.
- Realización de pruebas de caja negra a la aplicación.

Este trabajo de diploma se ha estructurado como sigue: Resumen, Introducción, cuatro Capítulos, Conclusiones generales, Recomendaciones, Referencias bibliográficas, Bibliografía y Anexos.

Capítulo 1. Fundamentación Teórica. Se presenta el marco teórico estudiado y seleccionado para desarrollar el presente trabajo, orientado al desarrollo de software en el campo de la Biología de Sistemas. Se presenta una panorámica sobre los conceptos y herramientas necesarios para alcanzar los objetivos planteados .

Capítulo 2. Características del Sistema. Se realiza una breve descripción de la solución propuesta, sus requisitos funcionales y no funcionales. Se presenta el modelo de dominio del sistema, el diagrama de casos de uso del sistema y la descripción de los mismos.

Capítulo 3. Diseño del Sistema. Se describe la representación arquitectónica del sistema. Se hace énfasis en el patrón de arquitectura utilizado, así como los patrones de diseño que fueron empleados. Se lleva a cabo el diseño del sistema.

Capítulo 4. Se describe la implementación del sistema, mostrando los diagramas de componentes. Se describe todo lo relacionado con la implementación de los plug-ins, así como lo relacionado con las pruebas realizadas al sistema.

Capítulo 1

FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En el presente capítulo se brinda una breve descripción de la Biología de Sistemas y se analizan algunos software que permiten el estudio de los sistemas biológicos. Se analizan algunas herramientas manejadoras de plug-ins y finalmente se da una descripción de las herramientas y metodologías a utilizar para desarrollar este trabajo de diploma.

1.2. Biología de Sistemas

La Biología de Sistemas(BS) representa la integración de conceptos e ideas de las ciencias de la vida, disciplinas de ingeniería y ciencias computacionales (Figura 1.1). Recientes avances de la Biología, incluyendo la secuencia del genoma humano y masivos experimentos para probar ejemplos biológicos, crearon nuevas oportunidades para comprender los problemas biológicos desde la perspectiva de un sistema. Este nuevo acercamiento hace más énfasis en el comportamiento de colecciones de componentes funcionando como un todo, que los enfoques tradicionales del estudio de componentes de forma individual. [5]

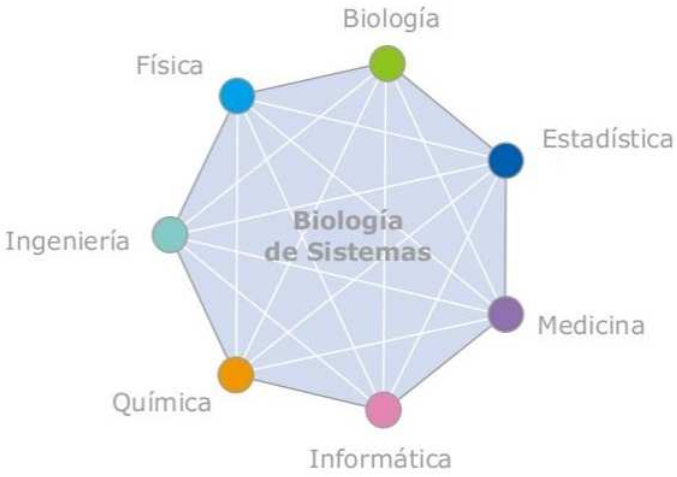


Figura 1.1: Interrelación de las diferentes áreas que conforman la Biología de Sistemas.

A diferencia de los métodos clásicos de estudio usados por los biólogos, que se basan en la confirmación o refutación de hipótesis por medio de resultados experimentales, la Biología de Sistemas emplea técnicas de predicción rigurosas. Estas técnicas surgen fundamentalmente del uso de modelos matemáticos que describen el comportamiento del ente en estudio. Los modelos permiten predecir el comportamiento del proceso como un sistema dinámico, generalmente tratado como una red compleja.

En ocasiones se confunde la Biología de Sistemas con Sistemas Biológicos, por lo cual es preciso aclarar que los Sistemas Biológicos son por definición, sistemas abiertos que operan en condiciones alejadas del equilibrio termodinámico, con muchas y fuertes interacciones no lineales entre sus muchos elementos. [6]

1.2.1. Modelos Matemáticos

El estudio de los Sistemas Biológicos se hace más factible mediante la creación de un modelo matemático que lo describa. Para que un modelo sea útil todos los elementos que lo componen deben ser planteados coherentemente, es decir, tiene que poder explicar lo que sucede de forma lógica. Los modelos no son más que esquemas o ecuaciones que explican aquello que se quiere estudiar, lo cual puede excluir características del ente en estudio que se consideren no sean importantes. [7]

En la actualidad se utilizan los modelos matemáticos en la mayoría de los campos de la ingeniería. Su objetivo es entender ampliamente el fenómeno y tal vez predecir su comportamiento en el futuro. Como por ejemplo: el crecimiento de las poblaciones, la concentración de un producto en una reacción química, el funcionamiento de las neuronas y la dinámica intracelular. [8]

Un modelo matemático nunca es una representación exacta de la realidad, es sólo una idealización que permite tratarla como un problema matemático. Un modelo de un sistema biológico es convertido a sistemas de ecuaciones, aunque la palabra modelo es a menudo usada como el sistema de las ecuaciones correspondientes. La solución de las ecuaciones, ya sea por medios analíticos o numéricos, describe cómo el sistema biológico se comporta ya sea en el tiempo o en equilibrio. [8]

1.3. Software para la Biología de Sistemas

Existe una gran gama de software desarrollados por centros de investigación o universidades que han sido de mucha utilidad en el desarrollo de las investigaciones en la Biología de Sistemas. Baste destacar, que en el sitio web del SBML(www.sbml.org), se pueden encontrar referencias a 181 software[1], basados en este estándar y relacionados con alguna de las áreas de la Biología de Sistemas.

De estos software se seleccionaron aquellos que, se considera, más han aportado al desarrollo de las investigaciones en esta área, los cuales se comentan a continuación.

1.3.1. De propósito general

1.3.1.1. Virtual Cell

Virtual Cell es el resultado del trabajo del Centro Nacional de Recursos para la Modelación y el Análisis Celular de los Estados Unidos. En estos momentos ya está disponible la versión 4.2. Este sistema desarrollado en Java, permite asociar información bioquímica y electrofisiológica con datos experimentales, permitiendo de esta manera la fácil comparación entre los resultados obtenidos por simulación con los resultados experimentales. [9]

Virtual Cell posee dos módulos de diseño, el biológico y el matemático, permitiendo de esta manera que los investigadores puedan escoger de qué forma quieren crear sus modelos. En el caso de aquellos que utilicen el módulo

[1]Revisado el 21 de enero de 2010

biológico, el sistema les permite crear de forma automática el modelo matemático asociado a su modelo biológico. [9]

Además de las facilidades de modelación que hacen de Virtuall Cell una opción muy atractiva para los investigadores, este sistema permite realizar exploraciones de fuerza bruta. Estas exploraciones consisten en ejecutar simulaciones complejas en las que uno o más parámetros dentro del modelo matemático pueden variar acorde a determinadas reglas impuestas por el usuario. Cuando se ejecutan este tipo de simulaciones se generan una serie de combinaciones diferentes que requieren de grandes capacidades de cómputo, para que las mismas pueden simularse en paralelo en diferentes nodos. Los resultados de estas simulaciones pueden visualizarse por separado, seleccionando una combinación de parámetros determinada. Este proceso de simulación elimina el tedioso trabajo de crear, editar, simular y comparar el resultado de muchas simulaciones individuales.

En resumen, Virtuall Cell constituye un software muy bien pensado y diseñado. Permite hacer simulaciones que otros sistemas no conciben, como es el caso de las exploraciones por fuerza bruta. Pero, estas ventajas que presenta, son a la vez las mayores desventajas del mismo, pues, una vez realizadas las múltiples simulaciones, no se pueden hacer más análisis que ver las diferentes dinámicas y compararlas visualmente, además de que se requiere de una conexión estable a Internet porque las simulaciones solo corren sobre sus servidores.

1.3.1.2. Cellware

Desarrollado por el Grupo de Biología de Sistemas del Instituto de Bioinformática de Singapur, Cellware constituye un sistema muy completo por las funcionalidades que pone en manos de los usuarios. Diseñado para modelar reacciones bioquímicas celulares, permite además simular sendas metabólicas y redes de genes. También incluye algoritmos para estimación de parámetros y autoorganización de los grafos que resulten de los diseños realizados. [2]

Las funcionalidades fundamentales son las siguientes:

- Acceso a infraestructura Grid para aquellos problemas que requieran de gran capacidad de cómputo.
- Amplia librería de algoritmos para simulaciones cuantitativas. Esta librería va desde algoritmos para simulaciones en sistemas determinísticos hasta los estocásticos.
- Análisis de redes: Permite realizar análisis estadísticos sobre las redes de genes o metabólicas.

Otra funcionalidad importante dentro de Cellware es que permite acceder a diferentes bases de datos disponibles en Internet, de las cuales se pueden descargar modelos de sistemas biológicos para su posterior simulación. [2]

Sin embargo, al igual que muchos de los software anteriores presenta varias deficiencias. La primera es que cuando se necesitan de grandes capacidades de cómputo para resolver determinados problemas, se tiene que acceder a sistemas de cálculo a través de Internet, lo que obliga, para el buen funcionamiento del software, así como poder explotar todas las prestaciones del mismo, a disponer de una buena conexión. La segunda es que no incluye una base de datos con la cual realizar consultas de los resultados obtenidos de las simulaciones. La tercera es que sólo implementa dos herramientas para el análisis: estimación de parámetros y diseño de gráficas.

1.3.1.3. SimBiology

SimBiology ha sido una extensión hecha al Matlab, con herramientas para la modelación, simulación y análisis de redes bioquímicas. Este sistema permite crear nuevos modelos o importar modelos ya existentes en formato SBML.

Al estar implementado sobre Matlab provee al usuario de múltiples algoritmos para la simulación, ya sean estocásticos o determinísticos. Permite además realizar la estimación de parámetros y análisis de sensibilidad.

SimBiology es una herramienta de reciente inclusión dentro del Matlab y su principal limitante es que posee pocas funcionalidades en comparación con otros software ya analizados. Además, sólo realiza análisis de sensibilidad y no incluye una base de datos donde almacenar los resultados de las simulaciones. [1]

1.3.1.4. BioSyS 1.0

La UCI, de conjunto con el Centro de Inmunología Molecular (CIM), desarrolló una primera versión del software BioSyS, el cual pretende resolver muchas de las deficiencias de los software que anteriormente se explicaron. Sin embargo, la versión 1.0 de BioSyS todavía presenta algunas limitaciones que detallamos a continuación:

1. Aunque se concibió de forma modular, cuando se realizó la integración de los módulos, se crearon determinadas dependencias entre los mismos, por lo que, si se quieren extender las funcionalidades del software se tiene que conocer detalladamente la implementación. Esto dificulta que desarrolladores ajenos al proyecto puedan incorporarle nuevas funcionalidades al mismo.

2. A pesar de usar Hibernate como framework, solo se implementaron funcionalidades para usar los gestores de Base de Datos MySQL y PostgreSQL. Esto hace que BioSyS 1.0 sea dependiente de gestores de base de datos que hay que instalar y configurar, por lo que se requiere de conocimientos informáticos avanzados para instalar el mismo.

3. El Editor de Ecuaciones en la versión 1.0 tiene deficiencias a la hora de insertar en la Base de Datos los modelos matemáticos. Se requiere de la intervención del usuario para seleccionar variables y parámetros que ya fueron introducidos en el sistema. El editor reconoce los subíndices como variables independientes, lo cual es incorrecto.

4. El módulo de simulación de la versión 1.0 solo contiene al Matlab con todos sus métodos numéricos y un método de la librería ODEtoJava como motores de cálculo. En el caso del Matlab, sólo funciona en sistemas operativos GNU/Linux. Esto limita el uso del software porque obliga a usar un sistema operativo determinado o limita la cantidad de métodos numéricos. Además, en esta primera versión sólo se podían configurar las propiedades básicas de los métodos numéricos.

5. El módulo de graficación sólo permite hacer gráficas en 2D, lo que limita la visualización de los resultados.

1.3.2. Basados en plug-ins

1.3.2.1. BioSPICE

BioSPICE surge a partir de la idea de BioComp de poner a disposición de la comunidad de la Biología de Sistemas una herramienta computacional que fuera capaz de integrar el manejo de la información biológica disponible, con la creación de modelos y la simulación computacional.

Basados en el concepto de reutilizar lo que ya está hecho, el mayor aporte del proyecto BioSPICE fue la creación de una interfaz entre este sistema y otros ya existentes. Así, BioSPICE puede integrarse con diferentes software y poner en manos del usuario módulos de modelación que van desde los que usan lenguajes scripts, editores tabulares o las más amigables interfaces de modelación gráfica. De la misma forma integra diferentes herramientas de simulación, simuladores estocásticos o determinísticos y algoritmos de estimación de parámetros. [4]

Algunos sistemas que se integraron a BioSPICE son JDesigner, BioSpreadsheet, Jarnac y JigCell.

La filosofía de BioSPICE es muy buena porque le da la posibilidad al usuario de elegir las herramientas que quiere utilizar para modelar y cuáles para simular, haciendo que el mismo se sienta cómodo con el sistema de cómputo que utiliza en sus investigaciones, sin embargo, presenta dos desventajas fundamentales. La primera es que se requiere de conocimientos de computación para integrar los diferentes sistemas, conocimientos que no muchos investigadores de las ciencias poseen. La segunda es que los análisis que permite realizar sobre las simulaciones son muy básicos.

1.3.2.2. Systems Biology Workbench

El Systems Biology Workbench (SBW) es un marco para compartir los recursos computacionales. Le permite a las aplicaciones comunicarse entre sí de manera eficiente y sin perder su identidad. Las aplicaciones pueden ser escritas en una variedad de idiomas diferentes y se puede ejecutar en diferentes sistemas operativos a través de Internet. El banco de trabajo completo es de código abierto y de proveedores independientes. SBW fue diseñado para ofrecer un rendimiento excelente y que se adapta específicamente a las aplicaciones científicas.

A continuación se resumen algunas de las capacidades de SBW:

- Servicio dinámico y seguimiento de los módulos: El Corredor de SBW mantiene un seguimiento de los módulos, servicios y categorías de servicios, y proporciona un servicio para aprender sobre estos módulos.
- Notificación de eventos: Anuncia la ocurrencia, a todos los módulos que lo conforman, de ciertos eventos de SBW, como la puesta en marcha o parada de una instancia de un módulo, entre otros.
- Método de registro: Los módulos que no se están ejecutando, pero que desean, sin embargo, hacer publicidad de sus servicios, pueden hacerlo mediante el registro con el Corredor de SBW. [3]

A pesar de todas los beneficios que reporta, todavía presenta algunas deficiencias como por ejemplo, no presenta ninguna base de datos para almacenar los resultados obtenidos de las simulaciones realizadas o de los análisis efectuados. Existen varios módulos implementados que realizan algunos análisis pero se puede decir que son los más básicos.

1.4. Necesidad de la Creación de BioSyS 2.0

Por todo lo anteriomente expuesto se hace evidente que se requiere de una herramienta más flexible, adaptable a los conocimientos del usuario, fácil de utilizar y con la posibilidad de extenderse en desarrollos comunitarios. Esta nueva herramienta debe resolver las deficiencias de la versión 1.0 de BioSyS mencionadas en la sección 1.3.1.4 y de los demás software explicados anteriormente.

Teniendo en cuenta que lo que se desea obtener, según la arquitectura definida para el proyecto, es un sistema orientado a plug-ins, lo primero que se debe realizar es un estudio de las herramientas manejadoras de plug-ins existentes y seleccionar una de ellas, para que sirva de base para la implementación de la versión 2.0 de BioSyS.

1.5. Herramientas manejadoras de Plug-ins

Existen varias herramientas que permiten desarrollar plug-ins. Tienen su base fundamental, pero las funcionalidades que el usuario desee se las puede incorporar mediante plug-ins ya creados con anterioridad. Además, permiten a la amplia comunidad de desarrollo crear plug-ins e integrarlos a ellas o utilizarlos como aplicaciones separadas. Entre las que se estudiaron, se encuentran las que se relacionan a continuación.

1.5.1. Eclipse

Eclipse es un Entorno de Desarrollo Integrado(IDE, por sus siglas en inglés), de código abierto multiplataforma y extensible para integrar herramientas. Para los clientes que lo utilizan para desarrollar plug-ins o plataformas de herramientas enteras, Eclipse representa una tecnología probada, fiable y escalable en la que diseñar, desarrollar e implementar productos comerciales.

El IDE de Eclipse emplea módulos (plug-ins) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. También da soporte a otros lenguajes de programación, como son C/C++, Cobol, Fortran, PHP o Python. Permite trabajar con lenguajes para procesado de texto, aplicaciones en red como Telnet y Sistema de Gestión de Base de Datos.

A pesar de todas estas ventajas, implementar un plug-in para eclipse es un poco engorroso, ya que, la curva de aprendizaje es muy lenta, por lo que desarrollar un plug-ins para Eclipse demoraría un tiempo considerable. Además, desarrollar interfaces visuales resulta un poco complicado.

1.5.2. NetBeans

Se refiere a una plataforma para el desarrollo de aplicaciones 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 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 extenderse agregándole nuevos módulos. Debido a que los módulos pueden desarrollarse independientemente, las aplicaciones basadas en la plataforma NetBeans pueden extenderse por otros desarrolladores de software. Algunas de sus características fundamentales son:

- Es un proyecto de código abierto.
- El IDE es desarrollado para distintas plataformas como Linux, MacOS X, Solaris y también Windows.

- Trae incorporados todos los plug-ins que se necesitan para trabajar, además, es configurable en su instalación, o sea, el usuario escoge los plug-ins con los que quiere instalar el IDE.

A pesar de ser un potente IDE de desarrollo, presenta algunas desventajas como por ejemplo, la curva de aprendizaje es un poco lenta, por lo que también, desarrollar un plug-ins para NetBeans llevaría un tiempo considerable.

1.5.3. Front-End del proyecto alasGRATO

Este Front-End surgió como una necesidad del proyecto alasGRATO en el Polo de Bioinformática de la Facultad 6 de la Universidad de las Ciencias Informáticas para mejorar la interacción de los usuarios con la plataforma alasGRATO. Esta herramienta resuelve la principal dificultad que tenían las otras herramientas manejadoras de plug-ins, o sea, la curva de aprendizaje lenta que presentan estas, por lo que desarrollar algún plug-in en ellas se convertía en un proceso tedioso. El Front-End es dinámico y multiplataforma, y se encarga de la portabilidad de los plug-ins garantizando:

- Unificar la carga de los plug-ins a través de descriptores XML.
- Eliminar impedancias en la interoperabilidad de los plug-ins.
- Ganar en soportes de escalabilidad para el ingreso de futuros plug-ins.
- Ganar en el manejo de memoria en ejecución controlando todos los componentes visuales desde un mismo marco.
- Mantener el principio de conservación, pues en la evolución del Front-End, los plug-ins desarrollados siempre serán compatibles. [10]

Por todo lo antes expuesto se escogió al Front-End desarrollado en el proyecto alasGRATO como la herramienta manejadora de plug-ins a utilizar para el desarrollo de BioSyS 2.0.

1.6. Herramientas y Metodologías utilizadas

1.6.1. Metodologías de desarrollo de software

La dificultad que presentan hoy día los desarrolladores a la hora de realizar una aplicación es la necesidad de saber cómo organizar las actividades para cada desarrollador por separado y para el equipo, definir qué artefactos deben crearse y contar con una serie de criterios que permitan controlar y medir los productos que se obtienen. Por lo tanto, se

necesita de una metodología capaz de dirigir estas actividades y así convertir los requisitos de los usuarios en un producto software. En el mundo existen distintas metodologías para dirigir las actividades vinculadas al proceso de desarrollo de software, entre estas metodologías se estudiaron las siguientes:

1.6.1.1. XP (Extreme Programming)

XP es una metodología ágil, más bien utilizada en proyectos de corto plazo, desventaja fundamental para la aplicación de esta metodología al software en cuestión, debido a que el mismo es un proyecto grande y que requiere un plazo de tiempo más bien largo para su realización. Esta metodología se utiliza para proyectos que tienen una urgencia en la fecha de entrega, por lo que hace uso de una programación rápida o extrema. La metodología XP se basa en:

• Pruebas Unitarias: basadas en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, se pueda hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.

• Re-fabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

• Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. [11]

1.6.1.2. RUP

El Proceso Unificado del Rational (RUP) goza de mucho prestigio debido a que ya cuenta con más de 30 años de experiencia y se desarrolló teniendo en cuenta y tomando lo mejor de otras metodologías orientadas a objetos.

"El Proceso Unificado del Rational es un proceso de desarrollo de Software, sin embargo, es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto." [12]

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

- Dirigido por los casos de uso: Teniendo en cuenta que la razón de ser de un sistema es brindar servicios a los usuarios, RUP define Caso de Uso como el conjunto de acciones que debe realizar un sistema para dar un resultado

de valor a un determinado usuario y los utiliza tanto para especificar los requisitos funcionales del sistema, como para guiar todos los demás pasos de su desarrollo, dígase diseño, implementación y prueba.

- Centrado en la arquitectura: La arquitectura es una vista del diseño completa con las características más importantes, dejando a un lado los detalles. Ésta no sólo incluye las necesidades de los usuarios e inversores, sino también otros aspectos técnicos como el hardware, sistema operativo, sistema de gestión de base de datos, protocolos de red; con los que debe coexistir el sistema. La arquitectura representa la forma del sistema, la cual va madurando en su interacción con los casos de uso hasta llegar a un equilibrio entre funcionalidad y características técnicas.

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

También permite dirigir las tareas de desarrolladores individuales y equipos de trabajo como una sola, incluso ofrece criterios para monitorear y medir los productos y actividades del proyecto. Una característica importante es que permite corregir errores en cada iteración y es flexible a cambios en los requerimientos.

1.6.1.3. OpenUP

OpenUP es un proceso unificado, abierto, mínimo y suficiente, lo que brinda la posibilidad de sólo incluir el contenido fundamental y necesario. Conserva las características principales de la metodología de desarrollo RUP por lo que se aplica de forma iterativa e incremental, provee los componentes básicos que pueden servir de base a procesos específicos a pesar de no tener lineamientos para todos los elementos que se manejan en un proyecto.

Es una forma de desarrollo ágil y ligero donde la mayoría de sus elementos fomentan el intercambio entre los equipos de desarrollo, y de colaboración 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. [13]

Teniendo en cuenta lo explicado anteriormente, la metodología que se utilizará para guiar el desarrollo de las funcionalidades es OpenUp debido a sus características y al ser este modelo el que más se ajusta a las necesidades del desarrollo de la aplicación, además es la metodología escogida y utilizada en el desarrollo de versiones anteriores del software BioSyS.

1.6.2. Lenguaje de Programación

Una de las finalidades que se busca con el desarrollo del presente software es que sea portable, por lo que se decidió implementar el mismo en Java. Este lenguaje utiliza el concepto de Máquina Virtual(VM, por sus siglas en inglés) por lo que el código que se genera no es específico a una plataforma en particular y así de esta manera, un código generado en Java puede correr en cualquier plataforma, o en donde se haya portado la VM. Este lenguaje tiene otras características importantes como son:

- Orientado a Objetos: Trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las características propias del paradigma Orientado a Objetos: Abstracción, Encapsulación, Herencia y Polimorfismo.
- Simple: Ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos.
- Robusto: Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores lo antes posible en el ciclo de desarrollo. Java obliga a la declaración explícita de los tipos de los ítems de información, reduciendo asi las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de la misma.

1.6.3. Herramienta de desarrollo

Para la implementación 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 no es más que una herramienta para el desarrollo de aplicaciones usando Java. La plataforma NetBeans permite que las aplicaciones sean desarrolladas

a partir de un conjunto de componentes de software llamados módulos. Debido a que los módulos pueden desarrollarse independientemente, las aplicaciones basadas en la plataforma NetBeans pueden extenderse fácilmente por otros desarrolladores de software. NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios. [1]

1.6.4. Herramienta CASE

Se decidió utilizar el Visual Paradigm para visualizar y diseñar los elementos de software debido a que es multiplataforma y por las facilidades que brinda para el diseño de los diagramas necesarios y su documentación. A continuación se brindan algunas características de esta herramienta:

- Es una potente herramienta para la Ingeniería de Software Asistida por Ordenador(CASE, por sus siglas en inglés) para visualizar y diseñar elementos de software, para ello utiliza el Lenguaje Unificado de Modelado(UML).
- Proporciona a los desarrolladores una plataforma que les permite diseñar un producto con calidad de una forma rápida.
- Facilita la interoperabilidad con otras herramientas CASE como el Rational Rose y se integra con las siguientes herramientas Java: Eclipse/IBM WebSphere, Jbuilder, NetBeans IDE, Oracle Jdeveloper, BEA Weblogic.
- Está disponible en varias ediciones: Enterprise, Professional, Community, Standard, Modeler y Personal.
- Genera código y realiza ingeniería inversa para diez lenguajes de programación, Java, C++, CORBA IDL, PHP, XML Schema y ADA.
- Genera código para C#, Visual Basic.net, Object Definition Lenguage(ODL), Flasch Action Script, Delphi, Perl y Phyton.
- Se integra con el Visio para importar imágenes del mismo para realizar los diagramas de despliegue.
- Genera documentación para el proyecto en HTML, MS Word y PDF.
- Además exporta e importa los diagramas en el estándar XML y como imágenes (ya sea con extensiones jpg o png).
- Es gratis en su edición Community. [14]

1.6.5. Gestores de Base de Datos

Para el desarrollo del software se utilizaron tres gestores de base de datos.

1.6.5.1. MySQL

Es un gestor de base de datos sencillo de usar y rápido. También es uno de los motores de base de datos más usados en Internet. Las características principales de MySQL son:

- Es un gestor de base de datos: Una base de datos es un conjunto de datos y un gestor de base de datos es una aplicación capaz de manejar este conjunto de datos de manera eficiente.
- Es una base de datos relacional: Una base de datos relacional es un conjunto de datos que están almacenados en tablas entre las cuales se establecen unas relaciones para manejar los datos de una forma eficiente y segura. Para usar y gestionar una base de datos relacional se usa el lenguaje estándar de programación SQL.
- Es Open Source: El código fuente de MySQL se puede descargar y está accesible a quien lo necesite. [15]

1.6.5.2. PostgreSQL

PostgreSQL es un motor de bases de datos relacionales que verifica integridad referencial con gran funcionalidad como base de datos, aunque un poco más lenta que otros motores. Su licencia es tipo BSD(Berkeley Software Distribution). Algunas de sus principales caracteristicas son:

- Corre en casi todos los principales sistemas operativos : Linux, Unix, BSDs, Mac OS, Beos, Windows, etc.
- Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios.
- Comunidades muy activas, varias comunidades en castellano.
- Bajo “Costo de Propiedad Total” (TCO, por sus siglas en inglés) y rápido “Retorno de la Inversión Inicial” (ROI, por sus siglas en inglés)
- Altamente adaptable a las necesidades del cliente.
- Soporte nativo para los lenguajes más populares del medio : PHP, Java, C, C++, Perl, Python, entre otros.

- Soporte de todas las características de una base de datos profesional (triggers, store procedures-funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas, etc.)
- Soporte de protocolo de comunicación encriptado por SSL
- Extensiones para alta disponibilidad, nuevos tipos de indices, datos espaciales, minería de datos, entre otros. [16]

1.6.5.3. HSQL

HSQL es un motor de bases de datos SQL ligero, OpenSource e implementado completamente en Java. Es ideal para realizar pruebas sin tener que conectarse a un gestor de bases de datos, o para aplicaciones donde en lugar de querer administrar la información en ficheros binarios y hacer métodos de consulta, borrado, inserción y demás acciones, se desea tratar esta información al igual que se hace con bases de datos relacionales en Java, es decir, usando JDBC.

Se puede descargar gratuitamente desde la dirección: *http://www.hsqldb.org*

Permite la creación de índices, control de la integridad referencial, sentencias de definición de datos, entre otras funcionalidades.

HSQL es una herramienta muy útil, ahorra bastante tiempo de desarrollo y al trabajar sobre bases de datos es flexible a la hora de realizar cambios en la información a tratar o cuando los requisitos de la información a tratar cambien.

1.6.6. Lenguaje de modelado

El Lenguaje Unificado de Modelado es un lenguaje gráfico para visualizar y documentar los elementos de los sistemas orientados a objetos. Dicho lenguaje es una notación unificada con la que se permite lograr un entendimiento que propicie el intercambio entre los usuarios y los desarrolladores. Se ha convertido en un estándar de la industria del software, debido a que fue impulsado por los autores de los tres métodos más usados de orientación a objetos Grady Booch, Ivar Jacobson y Jim Rumbaugh. El UML estándar está compuesto por tres partes: bloques de construcción (tales como clases, objetos, mensajes), relaciones entre los bloques (tales como asociación, generalización) y diagramas (por ejemplo, diagrama de actividad). [17]

1.6.7. Lenguaje de marcado

El Lenguaje de Marcas Extensible (XML), no es más que un conjunto de reglas para definir etiquetas semánticas que organizan un documento en diferentes partes. XML es un metalenguaje, que define la sintaxis utilizada para definir

otros lenguajes de etiquetas estructurados.

XML es una tecnología sencilla, que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad, ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. Es un estándar de fácil proceso tanto por humanos como por cualquier software que separa la información o contenido de su presentación o formato y está diseñado para procesarse en cualquier lenguaje o alfabeto. No pertenece a ninguna compañía en específico lo cual lo hace libre y constituye un estándar internacionalmente reconocido. [18]

1.6.8. Herramientas acopladas a la aplicación

Siguiendo la idea de reutilización tanto de código como de software ya existentes, para el desarrollo del software se han utilizado una serie de estas herramientas que permiten centrarse en aquellas cosas que son nuevas en la aplicación y utilizar sistemas ya probados para la realización de determinadas tareas.

A continuación se relacionan estas herramientas, así como la función que realiza cada una de ellas dentro del sistema:

- **mysql-connector-java-3.1.13**: Librería desarrollada en Java para interactuar con Gestores de Bases de Datos MySQL. Se utiliza para gestionar todo el proceso de interacción con la base de datos.
- **postgresql-8.3-603.jdbc3**: Librería desarrollada en Java para interactuar con Gestores de Bases de Datos PostgreSQL. Se utiliza para gestionar todo el proceso de interacción con la base de datos.
- **T-arenal**: Plataforma de propósito general para la realización de cálculos distribuidos. Sobre esta plataforma se han implementado una serie de algoritmos que permiten realizar múltiples simulaciones utilizando los recursos de cómputo disponibles dentro de una red corporativa.
- **jmathplot**: Librería diseñada en Java para graficar en 2D y 3D.
- **Matlab 7.0 para GNU/Linux**: Es un asistente con una amplia comunidad de desarrollo, que pone a disposición del usuario múltiples métodos numéricos parra resolver Sistemas de Ecuaciones Diferenciales(SED).
- **Octave 3.0**: Es un programa libre para realizar cálculos numéricos. Ofrece un intérprete permitiendo ejecutar órdenes en modo interactivo. Octave no es un sistema de álgebra computacional como podría ser Maxima, sino que usa un lenguaje que está orientado al análisis numérico.

- **Paquete Librerías ODEtoJava:** Es un paquete de software implementado en Java que permite resolver problemas de valor inicial para Ecuaciones Diferenciales Ordinarias rígidas y no rígidas. Este paquete contiene diferentes variantes del método de Runge-Kutta, que permiten resolver cualquier tipo de problema descrito mediante SED.

1.7. Conclusiones

El desarrollo de software para Biología de Sistemas ha ido en ascenso en los últimos años. En el presente capítulo se hizo una revisión de los software más relevantes, por su amplio uso y sus prestaciones, desarrollados dentro del área de la Biología de Sistemas. Se detallaron sus ventajas y sus principales dificultades, explicándose la necesidad de la creación de BioSyS 2.0.

Para el desarrollo de BioSyS 2.0, se seleccionó como herramienta manejadora de plug-ins, el Front-End desarrollado en el proyecto alasGRATO, como metodología de desarrollo de software OpenUp, como herramienta de desarrollo el NetBeans y como lenguaje de programación Java. Se seleccionó como gestores de base de datos MySQL, PostgreeSQL y HSQL, como herramienta CASE el Visual Paradigm, como lenguaje de modelado UML y como lenguaje de marcado XML.

Capítulo 2

CARACTERÍSTICAS DEL SISTEMA

2.1. Introducción

En el presente capítulo se brinda una breve descripción de la solución propuesta, sus requisitos funcionales y no funcionales, así como los actores que intervienen en ella. Se presenta el diagrama de casos de uso del sistema, así como la descripción de los mismos.

2.2. Breve descripción del sistema

Biosys 2.0 estará orientado a plug-ins facilitando que el usuario aumente o disminuya funcionalidades del mismo. En esta versión se realizarán tres plug-ins los cuales son indispensables para el buen funcionamiento del software, los cuales son, el de Simulación, el de Graficación y el de Base de Datos, además de incorporársele el plug-in de Editor de Ecuaciones para facilitar el trabajo con los modelos matemáticos.

2.3. Modelo de Dominio

Debido a la poca estructuración de los procesos del negocio que tienen que ver con el objeto de estudio y para poder entender el contexto en que se emplaza el sistema, se describe el funcionamiento de la aplicación mediante una serie de conceptos, entidades y sus relaciones, agrupándolos en un Modelo de Dominio.

El Modelo de Dominio es una representación de los conceptos u objetos del mundo real, significativos para un

problema. Tiene como objetivo fundamental la descripción de las clases más importantes en el sistema y representa conceptos del mundo real, no de los componentes de software.

2.3.1. Representación del Modelo de Dominio

El diagrama de clases del Modelo de Dominio del presente trabajo se muestra en la figura 2.1, donde se aprecia que el investigador estudia los sistemas biológicos, los cuales están descritos por uno o varios modelos matemáticos, que a su vez contienen una o varias variables y uno o varios parámetros. A estos modelos matemáticos se le puede realizar una o varias simulaciones de la cuales se puede graficar su dinámica de poblaciones.

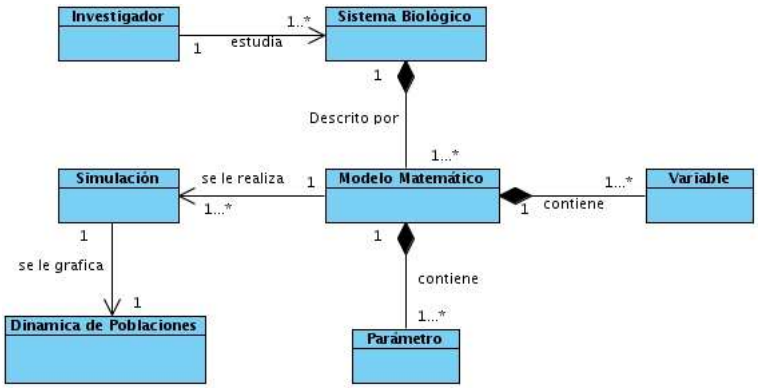


Figura 2.1: Representación del Modelo de Dominio de BioSyS 2.0

A continuación se describen cada una de las clases mostradas en el diagrama anterior.

Investigador: Persona capacitada para interactuar con la aplicación.

Sistema Biológico: Conjunto de entes biológicos que se relacionan entre sí.

Modelo Matemático: Conjunto de ecuaciones diferenciales que describen al Sistema Biológico.

Variable: Representa un ente dentro del sistema biológico.

Parámetro: Condición que influye en el comportamiento del sistema biológico.

Dinámica de poblaciones: Comportamiento de las variables en el tiempo.

2.4. Especificación de los Requisitos del Sistema

Los Requisitos son condiciones o capacidades que necesita el usuario para resolver un problema o conseguir un objetivo determinado. Esta definición se extiende y se aplica a las condiciones que debe cumplir o poseer un sistema -o uno de sus componentes- ,para satisfacer un contrato, una norma o una especificación.

2.4.1. Requisitos Funcionales

Los Requisitos Funcionales de un sistema describen su funcionalidad, los servicios que de él se esperan, o los que proveerá, entre ellos: sus entradas, salidas y excepciones. A continuación se muestran los requisitos funcionales de los tres plug-ins desarrollados.

2.4.1.1. Plug-in Base de Datos

- RF1 – Conectar a una Base de Datos.
- RF2 – Desconectar una Base de Datos.
- RF3 – Gestionar sistema biológico.
  - ✓ RF3.1- Insertar el sistema biológico a estudiar.
  - ✓ RF3.2- Eliminar un sistema biológico existente.
  - ✓ RF3.3- Modificar un sistema biológico existente.
- RF4 – Eliminar modelo matemático.
- RF5 – Mostrar información de un modelo matemático.

2.4.1.2. Plug-in Simulación

- RF6 – Simular modelo matemático.
- RF7 – Editar preferencias para simular.

2.4.1.3. Plug-in Graficación

- RF8 – Graficar simulación en 2D.
- RF9 – Graficar simulación en 3D.
- RF10 – Salvar gráfica.
- RF11 – Mostrar datos de una gráfica.
- RF12 – Editar propiedades de una gráfica.

2.4.2. Requisitos No Funcionales

Los Requisitos no Funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable, por ejemplo, pudiera desearse que el sistema responda dentro de un intervalo de tiempo especificado o que obtenga los resultados de los cálculos con un nivel de precisión dado. Por lo general, están vinculados a requisitos funcionales, es decir, una vez que se conozca lo que el sistema debe hacer, se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. En muchos casos los requisitos no funcionales son fundamentales en el éxito del producto.

2.4.2.1. Apariencia o interfaz externa

- La aplicación deberá tener una interfaz externa amigable, que sea sencilla y fácil de entender por el usuario.

2.4.2.2. Software

- No hay restricciones en cuanto al sistema operativo instalado en la PC puesto que la aplicación es multiplataforma.
- La máquina virtual de Java 1.6.

2.4.2.3. Hardware:

- 512 MB de memoria RAM como mínimo.
- Procesadores Pentium IV o superiores.

2.4.2.4. Requerimientos en el diseño y la implementación.

- Lenguaje de programación Java.
- Netbeans como IDE de desarrollo.
- Visual Paradigm como herramienta CASE.
- MySQL, PostgreeSQL ó HSQL como gestores de bases de datos.

2.4.2.5. Seguridad

- Confidencialidad:
  - ✓ Se requiere de usuario y contraseña para poder acceder a la información de la base de datos.
  - ✓ Se requiere de usuario y contraseña para poder acceder al servidor T-Arenal.
- Disponibilidad:
  - ✓ En caso de tener el usuario y la contraseña se le garantiza poder acceder a la información almacenada en todo momento.

2.4.2.6. Usabilidad

- El sistema podrá usarse por aquellos usuarios que posean conocimientos básicos en el campo de la modelación de sistemas biológicos.
- El sistema le ofrecerá al investigador la posibilidad de realizar simulaciones locales o distribuidas.
- 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.

2.4.2.7. Rendimiento

- Para hacer más rápida la obtención de los resultados de las simulaciones se implementará una aplicación que haga uso del cálculo distribuido de las mismas para agilizar el proceso de obtención de dichos resultados.

2.4.2.8. Soporte

- Se realizarán distintas pruebas al software una vez concluido para comprobar su funcionalidad.
- Terminado el software se prestarán los servicios de instalación y configuración de la aplicación.
- Se prestarán servicios de mantenimiento del software.

2.4.2.9. Requisitos Legales y Derecho de Autor

- Los derechos de autor serán registrados por la UCI.

2.5. Casos de Uso del Sistema

Los Casos de Uso son descripciones de las funcionalidades del sistema y se utilizan para obtener información de cómo debe trabajar éste. Independientemente de la implementación, describen, bajo la forma de acciones y reacciones, el comportamiento de un sistema desde el punto de vista del usuario.

2.5.1. Actores del Sistema

Se le llama actor a toda entidad externa al sistema que guarda una relación con éste y que le demanda una funcionalidad. Esto incluye a los operadores humanos, pero también incluye a todos los sistemas externos, así como a entidades abstractas como el tiempo.

Actor	Descripción
Investigador	Representa al usuario que va a hacer uso del sistema, y quien tiene la posibilidad de interactuar con todas las funcionalidades de éste.

2.5.2. Diagrama de Casos de Uso del Sistema

El Diagrama de Casos de Uso del Sistema se utiliza para especificar la comunicación y el comportamiento de un sistema, mediante su interacción con los usuarios y/u otros sistemas. A continuación se muestra el diagrama de casos de uso de la solución propuesta.

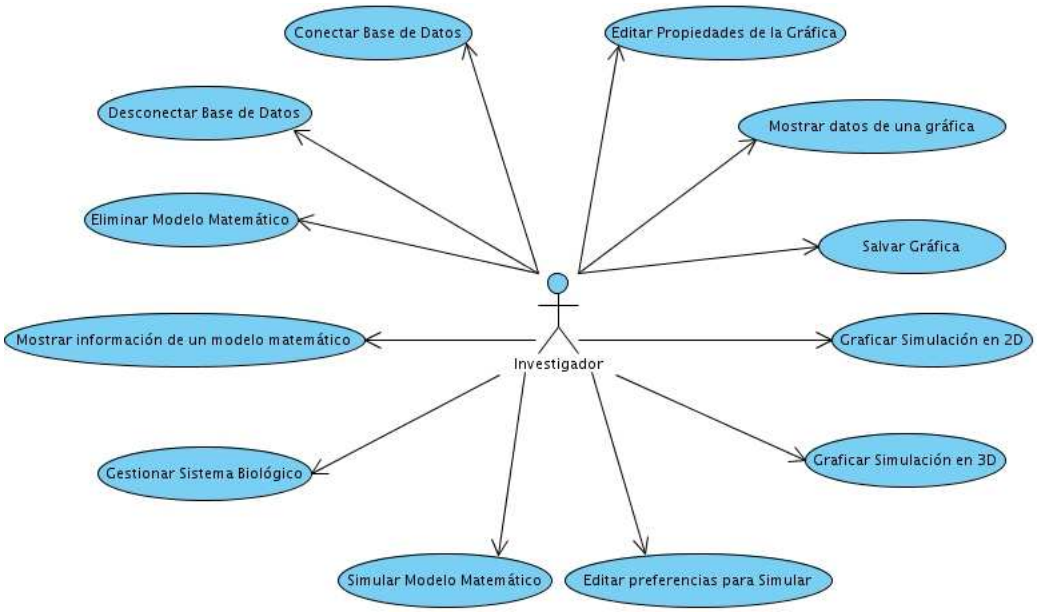


Figura 2.2: Diagrama de Casos de Uso del Sistema de BioSyS 2.0

2.5.3. Descripción de los Casos de Uso del Sistema

Un Caso de Uso es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario, o con otro sistema, para conseguir un objetivo específico. A continuación se muestra la descripción de los casos de uso del sistema de la solución propuesta.

Las descripciones textuales de los restantes casos de uso, se encuentra en el expediente de proyecto de BioSyS 2.0.

2.5.3.1. Caso de Uso Simular modelo matemático

Caso de Uso	**Simular modelo matemático**
Actores	**Investigador**
Propósito	Simular un modelo matemático (MM) de forma local o distribuida.
Resumen	El caso de uso se inicia cuando el investigador selecciona en el árbol de la aplicación un modelo matemático y escoge la opción simular en el menú del clic derecho.
Referencia	RF6.
Precondiciones	– El sistema debe estar conectado a una base de datos. – Debe existir al menos un modelo matemático en la base de datos.
Prioridad	Crítico
Flujo Normal de Eventos	
Acción de Actor	**Respuesta del Sistema**
1.- El investigador selecciona en el árbol de la aplicación un MM y escoge en el menú del clic derecho la opción Simular.	1.1.- El sistema muestra en el área de trabajo los datos necesarios para realizar la simulación.
2.- El investigador introduce los datos y escoge la herramienta matemática con la que desea simular, Octave, OdeToJava o Matlab.	2.1.- El sistema muestra en la barra de herramientas los métodos numéricos de la herramienta seleccionada.

3.- El investigador selecciona el método numérico con el cual desea realizar la simulación y escoge una de las siguientes opciones: • Simulación local • Simulación distribuida El investigador oprime el botón Correr Simulación.	3.1.- El sistema valida los datos introducidos. 3.2.- El sistema ejecuta una de las siguientes secciones: • Si seleccionó Simulación local ir a la sección Simulación local. • Si seleccionó Simulación distribuida ir a la sección Simulación distribuida.
Flujos Alternos	
Acción de Actor	**Respuesta del Sistema**
	3.1.- El sistema detecta errores en los datos introducidos y muestra una ventana de error.
Sección "Simulación Local" Flujo Normal de Eventos	
Acción de Actor	**Respuesta del Sistema**
	1.- Si seleccionó la herramienta matemática Octave o Matlab el sistema ejecuta el programa correspondiente en el directorio especificado.
	2.- El sistema comienza a simular el MM y muestra la barra de progreso que representa el estado de la simulación en ese momento, dando la posibilidad de pausarla o detenerla.
	3.- El sistema almacena los resultados de la simulación en la base de datos.

Sección “Simulación Local” Flujos Alternos	
Acción de Actor	**Respuesta del Sistema**
	1.- El directorio de la herramienta matemática no es correcto. El sistema muestra una ventana de diálogo con la opción de encontrarlo automáticamente.
2.- El investigador selecciona la siguiente opción: • Si • No	2.1- El sistema ejecuta una de las siguientes opciones: • Si seleccionó la opción Si el sistema busca el directorio de la herramienta y retorna al flujo normal de eventos numero 1. Si no encuentra el directorio de la herramienta muestra una ventana de error. • Si seleccionó la opción No el sistema muestra una ventana de diálogo para que el investigador especifique el directorio de la herramienta.
3.- El investigador especifica el directorio de la herramienta matemática.	3.1.- El sistema retorna al flujo normal de eventos numero 1.
Sección “Simulación Distribuida” Flujo Normal de Eventos	
Acción de Actor	**Respuesta del Sistema**
	1.- El sistema muestra una ventana para que el investigador se autentique en el servidor T-Arenal.
2.- El investigador introduce usuario y contraseña y oprime el botón Conectar.	2.1.- El sistema Graficar simulación en 2D. valida los datos introducidos y se conecta al servidor T-Arenal.

	2.2.- El sistema envía los datos para la ejecución de las simulaciones al servidor T-Arenal y se comienza a simular el modelo matemático seleccionado.
	2.3.- El sistema muestra una ventana de diálogo con las ejecuciones que se estén realizando, brindando la posibilidad de ver su estado o detenerla.
Sección "Simulación Distribuida" Flujos Alternos	
Acción de Actor	**Respuesta del Sistema**
2.- El investigador oprime el botón Cancelar.	2.1.- El sistema cierra la ventana correspondiente y cancela la simulación.
	2.1.1.- El sistema detectó algún error en los datos o no pudo establecer la conexión con el servidor y muestra una ventana de error.
Poscondiciones	Se realizó una simulación de forma local o de forma distribuida.

2.6. Conclusiones

- Se brinda una clara definición de los requisitos que debe cumplir el sistema, seleccionando el actor, así como los casos de uso del sistema.
- Se ganó claridad en cuanto al sistema a construir. También se sentaron las bases para las restantes fases del proceso de diseño e implementación, a través de la descripción de los casos de uso.

Capítulo 3

DISEÑO DEL SISTEMA

3.1. Introducción

En el presente capítulo se dará una descripción del estilo arquitectónico utilizado para el desarrollo de la herramienta, se describirán los patrones de diseño empleados, así como los diagramas de clases del diseño. Además se mostrará el diagrama de despliegue.

3.2. Patrón Arquitectónico utilizado

Los patrones arquitectónicos especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones, para organizar los distintos componentes.

Para el desarrollo de la herramienta se utilizó el patrón Modelo Vista Controlador (Model-View-Controller - MVC), clásico patrón de diseño utilizado para diseñar aplicaciones con sofisticadas interfaces, donde la lógica de interfaz de usuario cambia con más frecuencia que los almacenes de datos y la lógica de negocio.

Este patrón está catalogado como un patrón de arquitectura de software donde:

- **Modelo:** Representa específicamente el dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. La lógica de dominio añade significado a los datos. El modelo encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.

- **Vista:** Presenta el modelo en un formato adecuado para interactuar, usualmente con un elemento de interfaz de usuario. Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.

- **Controlador:** Responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Los eventos son traducidos a solicitudes de servicio ("services request") para el modelo o la vista.

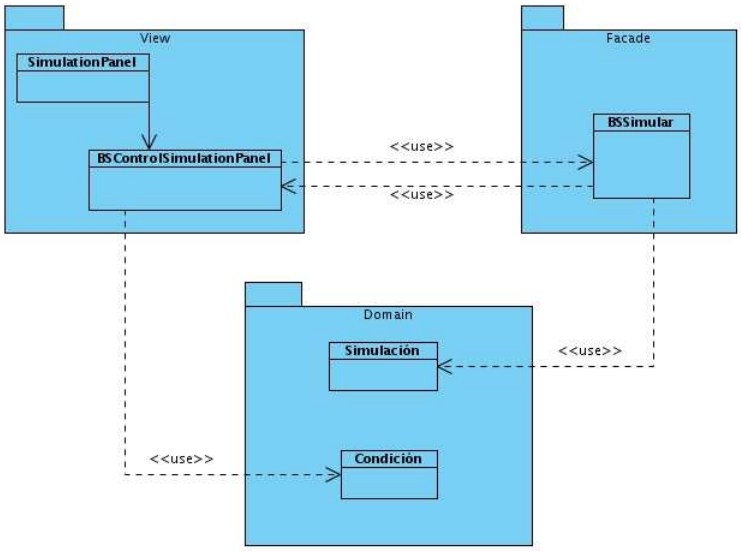


Figura 3.1: Ejemplo del patrón MVC en la solución propuesta

3.3. Patrones de Diseño utilizados

3.3.1. Patrón Creador

Este patrón es el encargado de que una clase B cree una instancia de una clase A, siempre que:

- La clase B contenga a la clase A

- B sea una agregación (o composición) de A

- B almacene a A

- B tenga los datos de inicialización de A(datos que requiere su constructor)
- B use a A

Este patrón se utilizó en la implementación de las clases de la aplicación, ejemplo de algunas son: SimulationPanel, BSControlSimulationPanel y BSControlSimulaciones.

3.3.2. Patrón Bajo Acoplamiento

Este patrón asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades. El controlador no las realiza, las delega en otras clases con las que mantiene un modelo de alta cohesión. En la Figura 3.3, se brinda un ejemplo donde se evidencia este patrón.

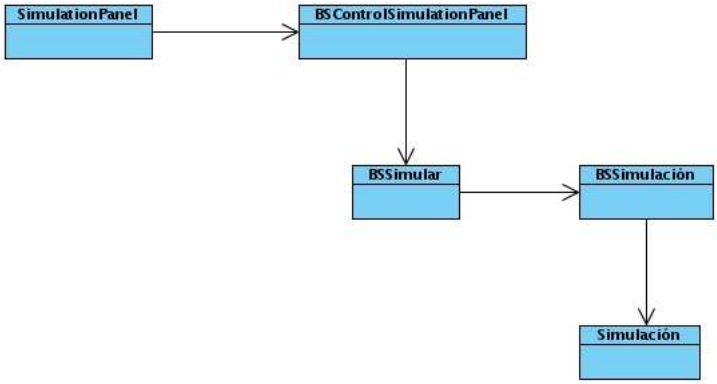


Figura 3.2: Aplicación del patrón Bajo Acoplamiento en BioSyS 2.0.

3.3.3. Patrón Alta Cohesión

La cohesión es una medida de fuerza con las que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada elemento de nuestro diseño debe realizar una labor única y autoidentificable dentro del sistema; no desempeñada por otro de ellos. Una clase con baja cohesión hace muchas operaciones no relacionadas, o trabaja en demasía. Ejemplo de la aplicación de este patrón en el sistema desarrollado se puede ver en las clases BSSimular y BSInsertarSimulacion.

3.3.4. Patrón Facade

Provee una interfaz unificada para un conjunto de interfaces en un subsistema. Define una interfaz de alto nivel que permite usar fácilmente un subsistema. [19] En la Figura 3.4,se muestra un ejemplo donde se evidencia la aplicación de este patrón.

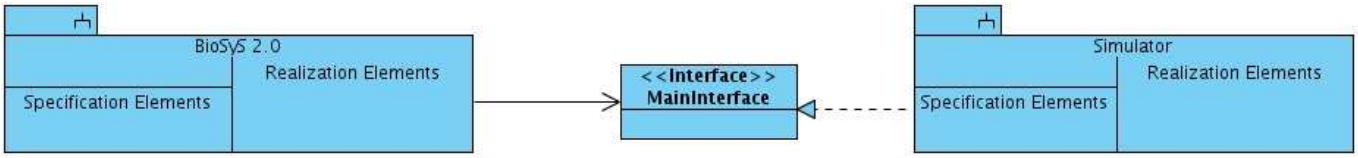


Figura 3.3: Aplicación del patrón Facade en BioSyS 2.0

3.4. Modelo de Diseño

Es una abstracción del Modelo de Implementación y su código fuente, el cual se emplea fundamentalmente para representar y documentar su diseño. Se utiliza como entrada esencial en las actividades relacionadas con la implementación. Representa a los casos de uso en el dominio de la solución. A continuación se abordarán los temas relacionados con el diseño de la aplicación.

3.4.1. Diagramas de Clases del Diseño

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema; también muestra sus clases, atributos y las relaciones entre ellos. Los diagramas de clases se utilizan durante el proceso de análisis y diseño de los sistemas, cuando se crea el diseño conceptual de la información que se manejará en el sistema, conjuntamente con los componentes que se encargarán del funcionamiento y la relaciones entre unos y otros.

Los diagramas de clases del diseño de los restantes casos de uso, se encuentran en el expediente de proyecto de BioSyS 2.0.

3.4.1.1. Diagrama de Clases del Caso de Uso Simular MM

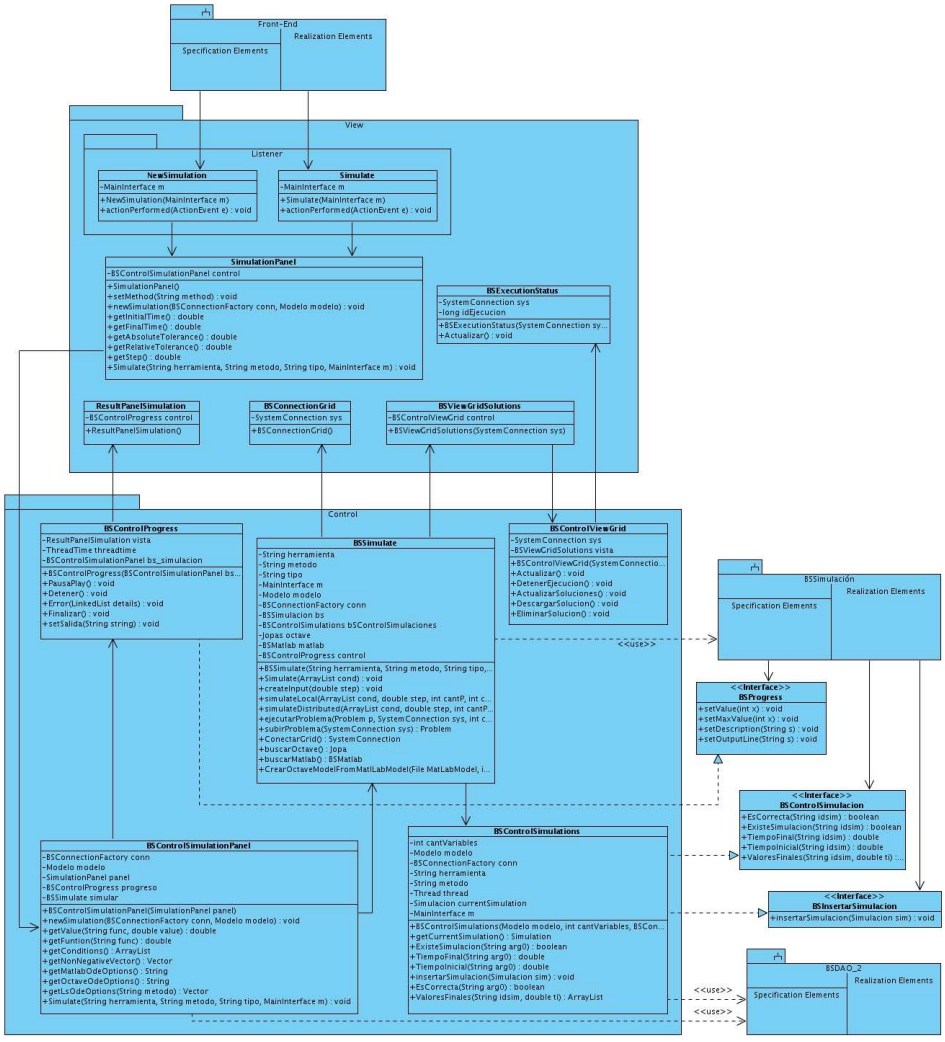


Figura 3.4: Diagrama de Clases del Caso de Uso Simular MM

3.4.2. Diagramas de Secuencia

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema. La mayoría de las veces, esto implica modelar instancias concretas o prototípicas de clases, interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento. Los diagramas de secuencia de los restantes casos de uso, se encuentran en el expediente de proyecto de BioSyS 2.0.

- Diagrama de secuencia: Caso de Uso Simular modelo matemático(sección Simular Local).**[Ver Anexo 1]**
- Diagrama de secuencia: Caso de Uso Simular modelo matemático(sección Simular Distribuido).**[Ver Anexo 2]**

3.4.3. Modelo de Despliegue

El modelo de despliegue es utilizado para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos. En la Figura 3.17, se observa que el cliente se conecta mediante un protocolo JDBC al servidor de Base de Datos y también se conecta al servidor T-arenal por el protocolo RMI para realizar las simulaciones distribuidas. A su vez este servidor tiene conectado varias computadoras para distribuir las tareas que se le asignen, las cuales están conectadas al servidor de Base de Datos por el protocolo JDBC.

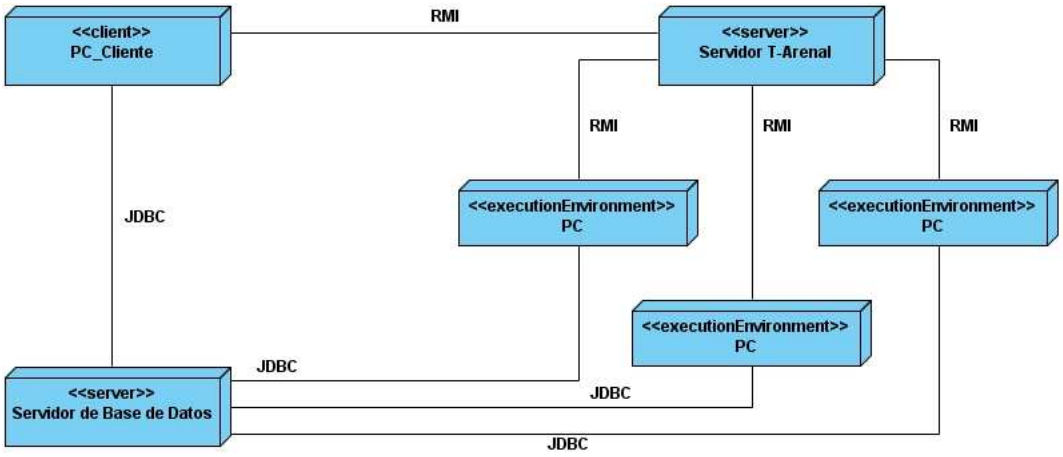


Figura 3.5: Diagrama de despliegue de BioSyS 2.0

3.5. Conclusiones

- Para la implementación de la aplicación se escogió el patrón arquitectónico Modelo-Vista-Controlador.
- En el diseño de la aplicación se utilizaron diferentes patrones de diseño como: Creador, Alta Cohesión, Bajo Acoplamiento y Facade.

Capítulo 4

Implementación y Prueba del Sistema

4.1. Introducción

En este capítulo se describe la implementación del sistema, mostrando los diagramas de componentes de los plug-ins desarrollados. Además se presenta todo lo relacionado con las pruebas realizadas al sistema.

4.2. Implementación

4.2.1. Diagrama de Componentes

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

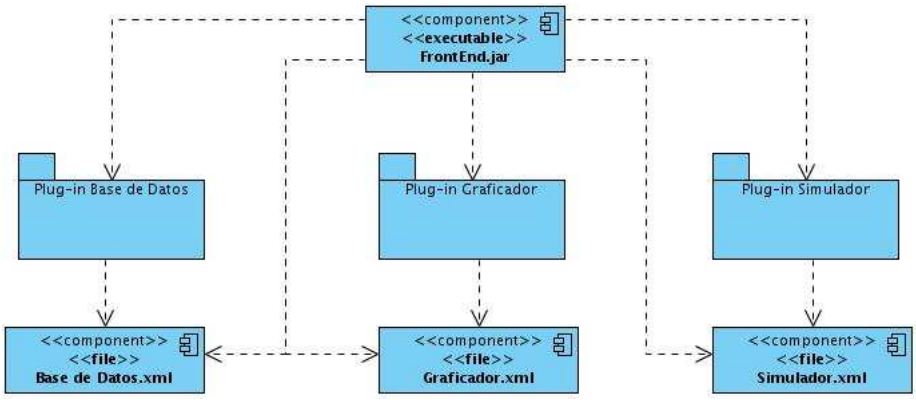


Figura 4.1: Diagrama de componentes de BioSyS 2.0

Debido a lo extenso del diagrama se agruparon los componentes en paquetes, los cuales corresponden a los plug-ins desarrollados. A continuación se muestran los diagramas de componentes de los tres plug-ins:

- Diagrama de componentes del plug-in Base de Datos. **[Ver Anexo 3]**
- Diagrama de componentes del plug-in Graficador. **[Ver Anexo 4]**
- Diagrama de componentes del plug-in Simulador. **[Ver Anexo 5)]**

4.2.2. Pasos para implementar un Plug-in para BioSyS 2.0

Para desarrollar un plug-in para BioSyS 2.0 se deberá tener en cuenta un grupo de pasos. (Ver [10])

4.2.3. Prototipos funcionales de BioSyS 2.0

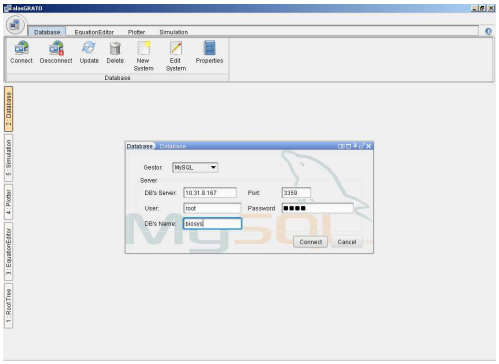


Figura 4.2: Plug-in Base de Datos de BioSyS 2.0

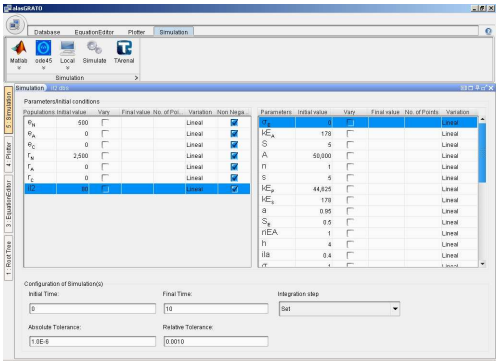


Figura 4.3: Plug-in Simulador de BioSyS 2.0

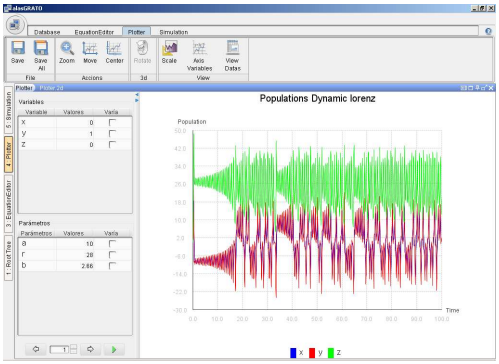


Figura 4.4: Plug-in Graficador de BioSyS 2.0

4.3. Pruebas del Sistema

Las pruebas de software son una actividad en la cual un sistema o componente se ejecuta bajo unas condiciones o requerimientos especificados, los resultados se observan y registran, y se hace una evaluación 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.

4.3.1. Plan de Prueba

Un plan de pruebas está constituido por un conjunto de pruebas. Cada prueba debe dejar claro: qué tipo de propiedades se quieren probar (corrección, robustez, fiabilidad); cómo se mide el resultado; especificar en qué consiste la prueba (hasta el último detalle de cómo se ejecuta) y definir cuál es el resultado que se espera.

4.3.1.1. Configuración del entorno de Prueba

La configuración del entorno donde se vayan a ejecutar las diferentes pruebas que se realizan a un software es un aspecto muy importante dentro del proceso de pruebas, pues si no se analizan bien los recursos de software y hardware que necesita el producto que se está construyendo, a la hora de probarlo se prescindirá de los elementos necesarios para la ejecución de un proceso de pruebas exitoso.

Por ello se tuvo en cuenta algunos requerimientos de hardware y de software que hicieron posible un mejor desarrollo de las pruebas, en aras de que se lograran minimizar los errores de la aplicación en desarrollo.

Requerimientos de software:

- PC con Windows XP o superior.
- PC con Linux (Ubuntu).
- Máquina Virtual de Java 1.6.

Requerimientos de hardware:

- PC con Microprocesador Pentium IV o superior.
- 160 GB de Disco Duro o superior.

- 512 MB de memoria RAM o superior.

Fecha de Inicio	Fecha de Fin	Actividades	Personal Implicado
21/4/2010	21/4/2010	Aceptación y firma del Plan de Prueba.	Carlos González Iglesias Edel Moreno Lemus Yunet González Mulet
23/4/2010	24/4/2010	Verificación de las condiciones previas para el inicio de las pruebas.	Carlos González Iglesias Edel Moreno Lemus Yunet González Mulet
Primera Iteración del plug-in Base de Datos			
26/4/2010	26/4/2010	Ejecución de las pruebas de caja negra al caso de uso Conectar una Base de Datos.	Carlos González Iglesias Edel Moreno Lemus
27/4/2010	27/4/2010	Ejecución de las pruebas de caja negra al caso de uso Desconectar una Base de Datos.	Carlos González Iglesias Edel Moreno Lemus
28/4/2010	29/4/2010	Ejecución de las pruebas de caja negra al caso de uso Gestionar sistema biológico.	Carlos González Iglesias Edel Moreno Lemus
30/4/2010	30/4/2010	Ejecución de las pruebas de caja negra al caso de uso Eliminar modelo matemático.	Carlos González Iglesias Edel Moreno Lemus
1/5/2010	1/5/2010	Ejecución de las pruebas de caja negra al caso de uso Mostrar información de un modelo matemático.	Carlos González Iglesias Edel Moreno Lemus
3/5/2010	3/5/2010	Análisis de las no conformidades y las solicitudes de cambio.	Carlos González Iglesias Edel Moreno Lemus Yunet González Mulet
Primera Iteración del plug-in Simulador			
4/5/2010	6/5/2010	Ejecución de las pruebas de caja negra al caso de uso Simular modelo matemático.	Carlos González Iglesias Edel Moreno Lemus

7/5/2010	7/5/2010	Ejecución de las pruebas de caja negra al caso de uso Editar preferencias para simular.	Carlos González Iglesias Edel Moreno Lemus
8/5/2010	8/5/2010	Análisis de las no conformidades y las solicitudes de cambio.	Carlos González Iglesias Edel Moreno Lemus Yunet González Mulet
Primera Iteración del plug-in Graficador			
10/5/2010	10/5/2010	Ejecución de las pruebas de caja negra al caso de uso Graficar simulación en 2D.	Carlos González Iglesias Edel Moreno Lemus
11/5/2010	11/5/2010	Ejecución de las pruebas de caja negra al caso de uso Graficar simulación en 3D.	Carlos González Iglesias Edel Moreno Lemus
12/5/2010	12/5/2010	Ejecución de las pruebas de caja negra al caso de uso Salvar gráfica.	Carlos González Iglesias Edel Moreno Lemus
12/5/2010	12/5/2010	Ejecución de las pruebas de caja negra al caso de uso Mostrar datos de una gráfica.	Carlos González Iglesias Edel Moreno Lemus
13/5/2010	15/5/2010	Ejecución de las pruebas de caja negra al caso de uso Editar propiedades de una gráfica.	Carlos González Iglesias Edel Moreno Lemus
17/5/2010	17/5/2010	Análisis de las no conformidades y las solicitudes de cambio.	Carlos González Iglesias Edel Moreno Lemus Yunet González Mulet

4.3.2. Diseño de las Pruebas de Caja Negra

Las Pruebas de Caja Negra son aquellas que se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene.

El diseño de las pruebas de caja negra de los restantes casos de uso, se encuentra en el expediente de proyecto de BioSyS 2.0.

Caso de Uso: Simular modelo matemático

Secciones a probar en el caso de uso.

Nombre de la Sección	Escenarios de la Sección	Descripción de la funcionalidad	Flujo Central
SC 1: Simular modelo matemático de forma local.	EC 1.1: Simular un modelo matemático con Matlab de forma local.	El usuario selecciona un modelo matemático en el árbol de la aplicación y selecciona la opción Simular en el menú del click derecho. El usuario introduce los valores de las variables y parámetros, selecciona la herramienta matemática Matlab y escoge el método a utilizar, escoge la opción simular local y oprime el botón Comenzar. El sistema valida los datos, realiza la simulación, muestra una barra de progreso y guarda los resultados en la base de datos.	Selecciona la opción Simular en el menú del click derecho.
SC 1: Simular modelo matemático de forma local.	EC 1.2: Simular un modelo matemático con OdeToJava de forma local.	El usuario selecciona un modelo matemático en el árbol de la aplicación y selecciona la opción Simular en el menú del click derecho. El usuario introduce los valores de las variables y parámetros, selecciona la herramienta matemática OdeToJava y escoge el método a utilizar, escoge la opción simular local y oprime el botón Comenzar. El sistema valida los datos, realiza la simulación, muestra una barra de progreso y guarda los resultados en la base de datos.	Selecciona la opción Simular en el menú del click derecho.

SC 1: Simular modelo matemático de forma local.	EC 1.3: Simular un modelo matemático con Octave de forma local.	El usuario selecciona un modelo matemático en el árbol de la aplicación y selecciona la opción Simular en el menú del click derecho. El usuario introduce los valores de las variables y parámetros, selecciona la herramienta matemática Octave y escoge el método a utilizar, escoge la opción simular local y oprime el botón Comenzar. El sistema valida los datos, realiza la simulación, muestra una barra de progreso y guarda los resultados en la base de datos.	Selecciona la opción Simular en el menú del click derecho.
SC 1: Simular modelo matemático de forma local.	EC 1.4: Simular un modelo matemático con Matlab de forma local y el directorio de Matlab no ha sido configurado.	El usuario selecciona un modelo matemático en el árbol de la aplicación y selecciona la opción Simular en el menú del click derecho. El usuario introduce los valores de las variables y parámetros, selecciona la herramienta matemática Matlab y escoge el método a utilizar, escoge la opción simular local y oprime el botón Comenzar. Se muestra el mensaje ”El directorio de Matlab no existe. ¿Desea que el sistema trate de encontrarlo?”.	Selecciona la opción Simular en el menú del click derecho.
SC 2: Simular modelo matemático de forma distribuida.	EC 2.1: Simular un modelo matemático con Matlab de forma distribuida.	El usuario selecciona un modelo matemático en el árbol de la aplicación y selecciona la opción Simular en el menú del click derecho. El usuario introduce los valores de las variables y parámetros, selecciona la herramienta matemática Matlab y escoge el método a utilizar, escoge la opción simular distribuido y oprime el botón Comenzar. Se muestra una ventana para loguearse en el servidor T-Arenal. El usuario introduce usuario y contraseña y oprime el botón Aceptar. El sistema se conecta al servidor y envía los datos de la simulación, se muestra una ventana con el estado de la simulación.	Selecciona la opción Simular en el menú del click derecho.

SC 2: Simular modelo matemático de forma distribuida.	EC 2.2: Simular un modelo matemático con Matlab de forma distribuida y el servidor T-Arenal esté apagado.	El usuario selecciona un modelo matemático en el árbol de la aplicación y selecciona la opción Simular en el menú del click derecho. El usuario introduce los valores de las variables y parámetros, selecciona la herramienta matemática Matlab y escoge el método a utilizar, escoge la opción simular distribuido y oprime el botón Comenzar. Se muestra una ventana de error con el mensaje ”Servidor fuera de servicio”.	Selecciona la opción Simular en el menú del click derecho.

El diseño de las pruebas por cada sección definida anteriormente se pueden apreciar en el Anexo 6.

4.3.2.1. No Conformidades detectadas

Elemento	No	No Conformidad	Aspecto correspondiente	Etapa de detección	Signif	No Signif
Interfaz	1	No se actualizaba el árbol de la aplicación.	Cuando se insertaba un nuevo sistema biológico.	Etapa de diseño y aplicación de Pruebas al plug-in de Base de Datos.	X	
Interfaz	2	No se actualizaba el árbol de la aplicación.	Cuando se eliminaba un modelo matemático.	Etapa de diseño y aplicación de Pruebas al plug-in de Base de Datos.	X	
Interfaz	3	Se mostraba un mensaje de error y no se actualizaba el árbol de la aplicación.	Cuando se presionaba dos veces consecutivas el botón Actualizar de la barra de herramientas.	Etapa de diseño y aplicación de Pruebas al plug-in de Base de Datos.	X	

Interfaz	4	Se tenía que esperar a que terminara de insertarse en la base de datos toda la información para realizar cualquier otra actividad.	Cuando se realizaban una o varias simulaciones.	Etapa de diseño y aplicación de Pruebas al plug-in de Simulación.	X	
Interfaz	5	Era obligatorio configurar las preferencias del servidor T-Arenal.	Cuando se editaban las preferencias para simular.	Etapa de diseño y aplicación de Pruebas al plug-in de Simulación.	X	
Interfaz	6	Se mostraba una gráfica en 2D de la relación entre las variables del modelo matemático.	Cuando se graficaba en 3D un modelo matemático con menos de 3 variables.	Etapa de diseño y aplicación de Pruebas al plug-in Graficador.	X	

4.4. Conclusiones

- Se logró implementar los plug-ins de Base de Datos, de Simulación y de Graficación con éxito.
- Se integraron los tres plug-ins a la aplicación.
- Se le realizaron pruebas de caja negra al sistema y se detectaron 6 no conformidades las cuales fueron registradas y solucionadas para un mejor funcionamiento del software .

CONCLUSIONES

- Se desarrolló la versión 2.0 del software BioSyS con una arquitectura orientada a plug-ins, utilizando para ello el Front-End desarrollado en el proyecto alasGRATO como herramienta manejadora de plug-ins.
- Se desarrolló el plug-in de Base de Datos para BioSyS 2.0, el cual brinda la posibilidad de gestionar la información a través de una base de datos.
- Se desarrolló el plug-in de Simulación para BioSyS 2.0, el cual brinda la posibilidad de realizar simulaciones locales o distribuidas con diferentes herramientas matemáticas como Matlab, Octave y OdeTojava.
- Se desarrolló el plug-in de Graficación para BioSyS 2.0, el cual brinda la posibilidad de realizar gráficas en 2D y en 3D de la dinámica de poblaciones y de la relación entre las variables de las simulaciones respectivamente.
- Se validó la aplicación a través de pruebas de caja negra, detectándose solamente 6 no conformidades, lo cual demuestra el buen funcionamiento del software.

RECOMENDACIONES

1. Incorporar el software Mathematica como una nueva herramienta para simular modelos matemáticos.
2. Desarrollar los plug-ins de los restantes módulos ya implementados en BioSyS 1.0.
3. Continuar con el desarrollo de nuevos plug-ins para BioSyS 2.0, con el objetivo de ampliar la potencialidad del software.

REFERENCIAS BIBLIOGRÁFICAS

[1] MULET, Yunet G. ; DELGADO, Yanet A.: Software para la Simulación de Sistemas Biológicos: Módulo de Simulación y Análisis. (2007), S. 6–30. – Tesis de grado, Universidad de las Ciencias Informáticas. [Consultado el 10 de noviembre de 2009] (document), 1.3.1.3, 1.6.3

[2] LEMUS, Noel M.: BioSyS: Software para la simulación y análisis de sistemas biológicos. (2007), S. 5–24. – Tesis de maestría, Universidad de las Ciencias Informáticas. [Consultado el 13 de noviembre de 2009] (document), 1.3.1.2

[3] 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. – [Consultado el 13 de noviembre de 2009] (document), 1.3.2.2

[4] GARVEY, T.D. ; LINCOLN, P. ; PEDERSEN, C.J. ; MARTIN, D. ; JOHNSON, M.: BioSPICE: access to the most current computational tools for biologists. In: *OMICS* 7 (2003), S. 411–420. – [Consultado el 19 de noviembre de 2009] (document), 1.3.2.1

[5] KITANO, H.: A graphical notation for biochemical networks. In: *BIOSILICO* Vol. 1 (2003), Nr. 5. – [Consultado el 10 de noviembre de 2009] 1.2

[6] OSORIO, Karel: Plataforma computacional para el desarrollo de la Biología de Sistema. In: *Facultad de Matemática Computación, Universidad de La Habana. Ciudad de la Habana* (2004). – [Consultado el 29 de noviembre de 2009] 1.2

[7] EDELSTEIN, L.: *Mathematical Models in Biology*. The Random House, 1988. – [Consultado el 4 de diciembre de 2009] 1.2.1

[8] CASTILLO, Yiliana R. ; GÁLVEZ, Yuandry N.: BioSyS: Implementación del Módulo de Análisis. 1 (2008), S. 5–6. – Tesis de grado, Universidad de las Ciencias Informáticas. [Consultado el 10 de diciembre de 2009] 1.2.1

[9] LOEW, L.M. ; SCHAFF, J.C.: The Virtual Cell: a software environment for computational cell biology. In: *Trends Biotechnol* 19 (2001), S. 401–406. – [Consultado el 10 de enero de 2010] 1.3.1.1

[10] MARTÍNEZ, Julio V.: Un nuevo Front-End para la plataforma alasGRATO. 1 (2009), S. 4–21. – Tesis de grado, Universidad de las Ciencias Informáticas. [Consultado el 14 de enero de 2010] 1.5.3, 4.2.2

[11] ROBLES, Gregorio ; FERRER, Jorge: Programación Extrema y Software Libre. In: *Universidad Politécnica de Madrid.* (2002). – [Consultado el 20 de enero de 2010] 1.6.1.1

[12] BOOCH, IVAR JACOBSON G.: El Proceso Unificado de Desarrollo de Software. 1 (1999). – [Consultado el 10 de diciembre de 2009] 1.6.1.2

[13] *Ayuda Extendida OpenUp (2008).* : *Ayuda Extendida OpenUp (2008).*. – [Consultado el 20 de enero de 2010] 1.6.1.3

[14] *Visual Paradigm for UML.*. – [En línea] Disponible en: http://www.versioncero.com/noticia/210/visual-paradigm-for-uml. [Consultado el 20 de diciembre de 2009] 1.6.4

[15] GRACIA, J.: *Introdución a MySQL*. – [En línea] Disponible en: http://www.webestilo.com/mysql/intro.phtml. [Consultado el 15 de enero de 2010] 1.6.5.1

[16] *Sistema Gestor de Base de Datos PostgreSQL.*. – [En línea] Disponible en: http://www.http-peru.com/postgresql.php. [Consultado el 15 de enero de 2010] 1.6.5.2

[17] CARO, Patricio S. ; HISTCHFELD, Nancy: *Manual de UML*. – [En línea] Disponible en: http://www.dcc.uchile.cl/ psalinas/uml [Consultado el 20 de diciembre de 2009] 1.6.6

[18] ARENAS, María I. G.: *Curso XML 1era Edición.* – [En línea] Disponible en: http://geneura.ugr.es/ maribel/xml/introduccion/index.html [Consultado el 10 de diciembre de 2009] 1.6.7

[19] GAMMA, E. ; HELM, R. ; JOHNSON, R. ; VLISSIDES, J.: Patrones de diseño. (2000). – Disponible en: http//www.vico.org/pages/PatronsDisseny.html. [Consultado el 20 de marzo 2010]. 3.3.4

BIBLIOGRAFÍA

[1] *Ayuda Extendida OpenUp (2008).* [Consultado el 20 de enero de 2010].

[2] Bartocci, E. ; Cacciagrano, D. ; Cannata, N. ; Corradini, F. ; Merelli, E. ; Milanesi, L. ; Romano, P. An agent-based multilayer architecture for bioinformatics grids. *IEEE TRANSACTIONS ON NANOBIOSCIENCE*, 6(2), June 2007. [Consultado el 13 de noviembre de 2009].

[3] Cannataro, M. ; Congiusta, A. ; Pugliese, A. ; Talia, D. ; Trunfio, P. Distributed data mining on grids: Services, tools, and applications. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART B: CYBERNETICS*, 34(6):2451–2465, 2004. [Consultado el 10 de diciembre de 2009].

[4] Castillo, Yiliana R. ; Gálvez, Yuandry N. Biosys: Implementación del módulo de análisis. 1:5–6, 2008. Tesis de grado, Universidad de las Ciencias Informáticas. [Consultado el 10 de diciembre de 2009].

[5] Chee Meng, T. ; Somani, S. ; Dhar, P. Modeling and simulation of biological systems with stochasticity. *In Silico Biology*, 4:293–309, 2004. [Consultado el 20 de enero de 2010].

[6] Cheng, J. ; Scharenbroich, L. ; Baldi, P. ; Mjolsness, E. Sigmoid: A software infrastructure for pathway bioinformatics and systems biology. *IEEE INTELLIGENT SYSTEMS*, pages 1541–1672, 2005. [Consultado el 20 de noviembre de 2009].

[7] Cornish-Bowden, A. ; Cárdenas, M. L. Systems biology may work when we learn to understand the parts in terms of the whole. *Biochemical Society Transactions*, 33:516–519, 2005. [Consultado 10 de noviembre de 2009].

[8] Crampin, E. J. ; Halstead, M. ; Hunter, P. ; Nielsen, P. ; Noble, D. ; Smith, N. ; Tawhai, M. Computational physiology and the physiome project. *Exp. Physiol*, pages 1–26, 2003. [Consultado el 14 de didiembre de 2009].

[9] Dhar, P. K. ; Meng, T. C. ; Somani, S. ; Ye, L. ; Sakharkar, K. ; Krishnan, A. ; Ridwan, A. B. M. ; Ho Kok Wah, S. ; Chitre, M. ; Hao, Z. Grid cellware: the first grid-enabled tool for modelling and simulating cellular processes. *Bioinformatics*, 21(7):1284–1287, 2005. [Consultado el 15 de enero de 2010].

[10] Edelstein, L. *Mathematical Models in Biology*. The Random House, 1988. [Consultado el 4 de diciembre de 2009].

[11] Ehrenberg, M. ; Elf, J. ; Aurell, E. ; Sandberg, R. ; Tegner, J. Systems biology is taking off. *Genome Research*, Vol. 13:2377–2380, 2003. [Consultado el 13 de noviembre de 2009].

[12] Gamma, E. ; Helm, R. ; Johnson, R. ; Vlissides, J. Patrones de diseño. 2000. Disponible en: http//www.vico.org/pages/PatronsDisseny.html. [Consultado el 20 de marzo 2010].

[13] Garvey, T.D. ; Lincoln, P. ; Pedersen, C.J. ; Martin, D. ; Johnson, M. Biospice: access to the most current computational tools for biologists. *OMICS*, 7:411–420, 2003. [Consultado el 19 de noviembre de 2009].

[14] Gershon, D. Bioinformatics in a post-genomics age. *Nature*, 389:417–418, 1997. [Consultado el 12 de octubre de 2009].

[15] Hoops, S. ; Sahle, S. ; Gauges, R. ; Lee, C. ; Pahle, J. ; Simus, N. ; Singhal, M. ; Xu, L. ; Mendes, P. ; Kummer, U. Copasi-a complex pathway simulator. *Bioinformatics*, 22(24):3067–3074, 2006. [Consultado el 11 de noviembre de 2009].

[16] http://www.forofilos.com/index.php?topic=2444.msg30120. [consultado el 10 de enero de 2010].

[17] Hucka, M. ; Hoops, S. ; Keating, S. M. ; Le Novere, N. ; Sahle, S. ; Wilkinson, D. J. Systems biology markup language (sbml) level 2: Structures and facilities for model definitions. *Nature Precedings*, 2008. [Consultado el 20 de noviembre de 2009].

[18] Kitano, H. A graphical notation for biochemical networks. *BIOSILICO*, Vol. 1(5), 2003. [Consultado el 10 de noviembre de 2009].

[19] Lemus, Noel M. Biosys: Software para la simulación y análisis de sistemas biológicos. pages 5–24, 2007. Tesis de maestría, Universidad de las Ciencias Informáticas. [Consultado el 13 de noviembre de 2009].

[20] Loew, L.M. ; Schaff, J.C. The virtual cell: a software environment for computational cell biology. *Trends Biotechnol*, 19:401–406, 2001. [Consultado el 10 de enero de 2010].

[21] Luscombe, N. M. ; Greenbaum, D. ; Gerstein, M. What is bioinformatics? an introduction and overview. *Yearbook of Medical Informatics*, 2001. [Consultado el 15 de octubre de 2009].

[22] Martínez, Julio V. Un nuevo front-end para la plataforma alasgrato. 1:4–21, 2009. Tesis de grado, Universidad de las Ciencias Informáticas. [Consultado el 14 de enero de 2010].

[23] Mulet, Yunet G. ; Delgado, Yanet A. Software para la simulación de sistemas biológicos: Módulo de simulación y análisis. pages 6–30, 2007. Tesis de grado, Universidad de las Ciencias Informáticas. [Consultado el 10 de noviembre de 2009].

[24] Osorio, Karel. Plataforma computacional para el desarrollo de la biología de sistema. *Facultad de Matemática Computación, Universidad de La Habana. Ciudad de la Habana*, 2004. [Consultado el 29 de noviembre de 2009].

[25] Paxson, R. ; Zannella, K. Systems biology: Studying the world's most complex dynamic systems. Technical report, The MathWorks, 2007. [Consultado el 20 de enero de 2010].

[26] Robles, Gregorio ; Ferrer, Jorge. Programación extrema y software libre. *Universidad Politécnica de Madrid.*, 2002. [Consultado el 20 de enero de 2010].

[27] Sarai, N. ; Matsuoka, S. ; Noma, A. simbio: A java package for the development of detailed cell models. *Progress in Biophysics and Molecular Biology*, Vol. 90:360–377, 2006. [Consultado el 5 de febrero de 2010].

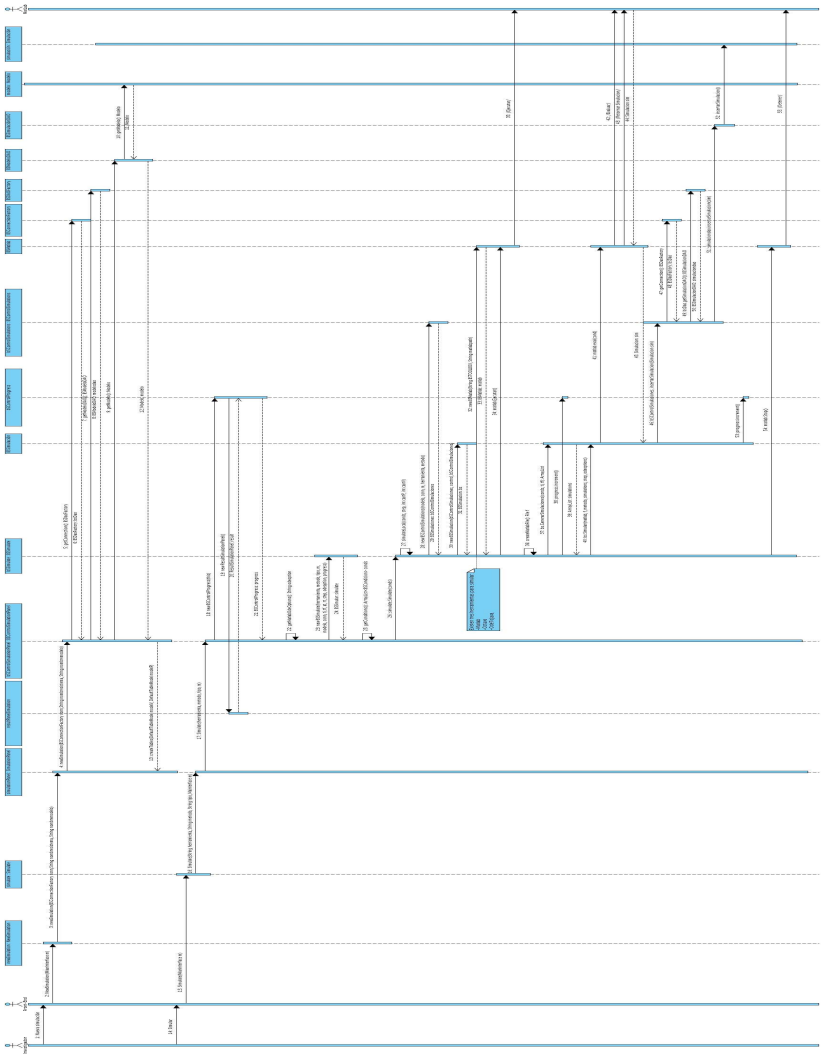
[28] Sauro, H. M. ; Hucka, M. ; Finney, A. ; Wellock, C. ; Bolouri, H. ; Doyle, J. ; Kitano, H. Next generation simulation tools: The systems biology workbench and biospice integration. *OMICS A Journal of Integrative Biology*, 7:353–370, 2003. [Consultado el 13 de noviembre de 2009].

[29] Schmidt, Henning. Sbaddon: high performance simulation for the systems biology toolbox for matlab. *BIOINFORMATICS*, 23(5):646–647, 2007. [Consultado el 20 de febrero de 2010].

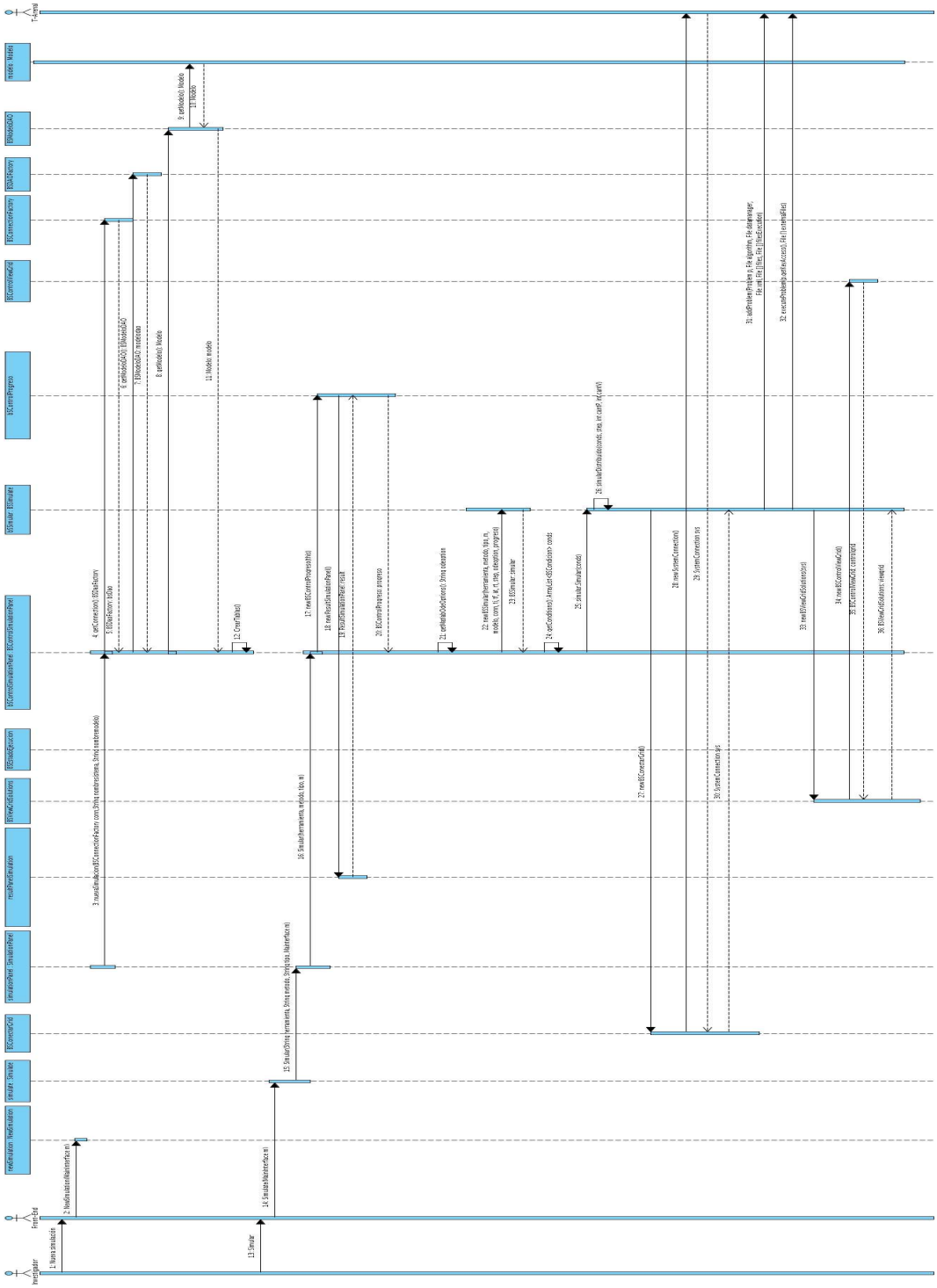
[30] Tadmor, B. ; Tidor, B. Interdisciplinary research and education at the biology engineering computer science interface: a perspective (reprinted article). *Biosilico*, 10(17), September 2005. [Consultado el 10 de noviembre de 2009].

ANEXOS

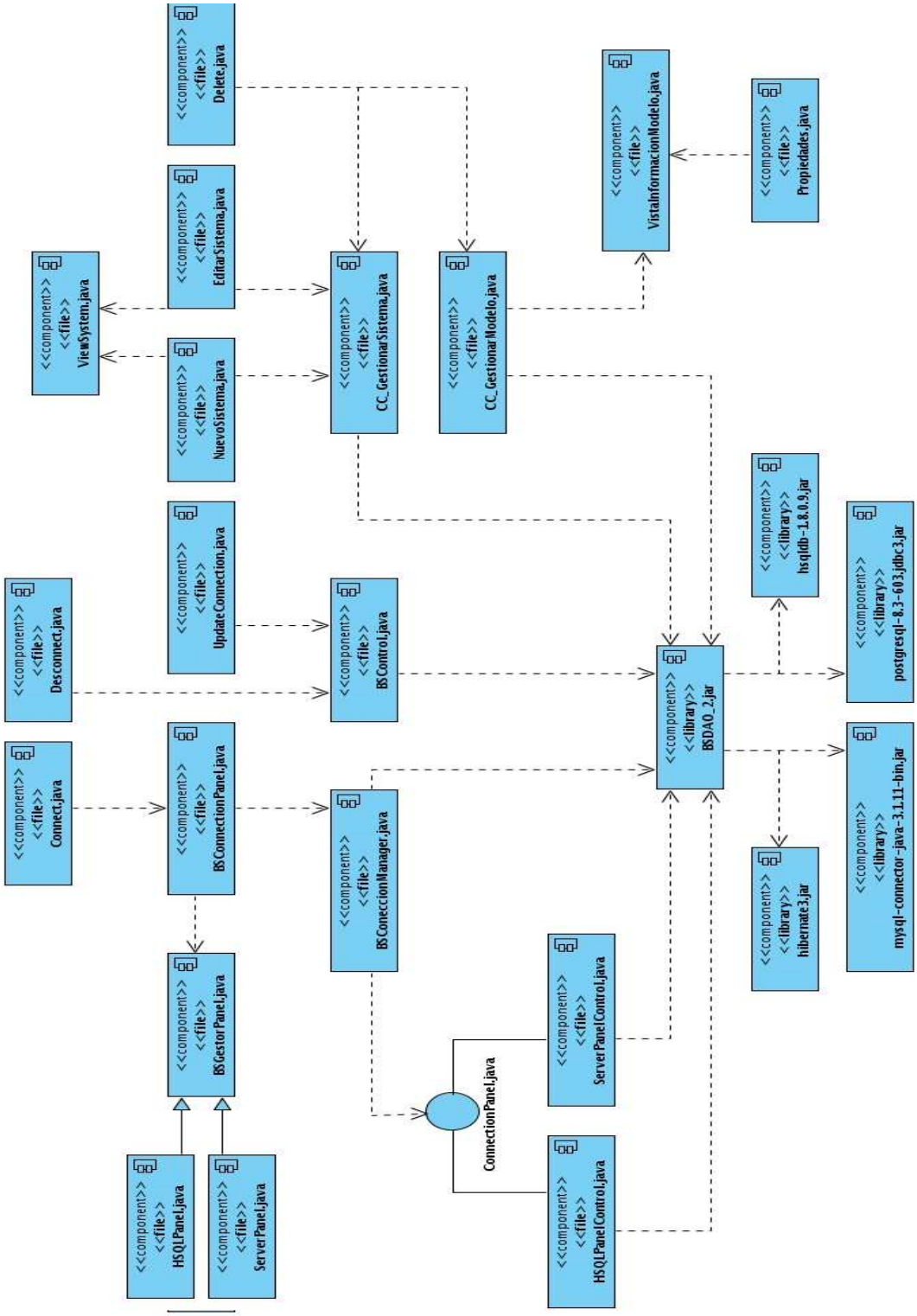
Anexo 1. Diagrama de secuencia del Caso de Uso Simular modelo matemático (sección Simular Local).



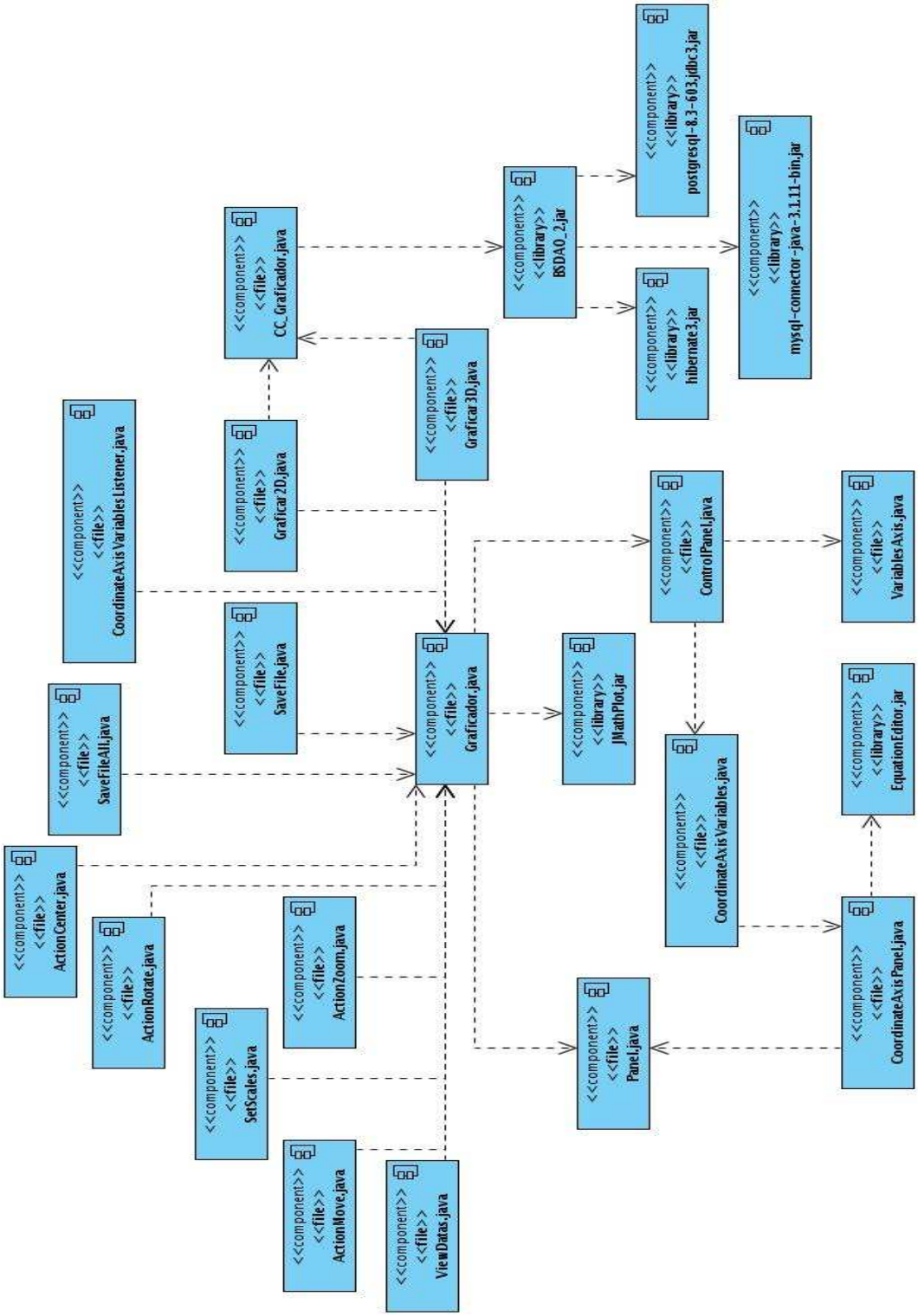
Anexo 2. Diagrama de secuencia del Caso de Uso Simular modelo matemático (sección Simular Distribuido).



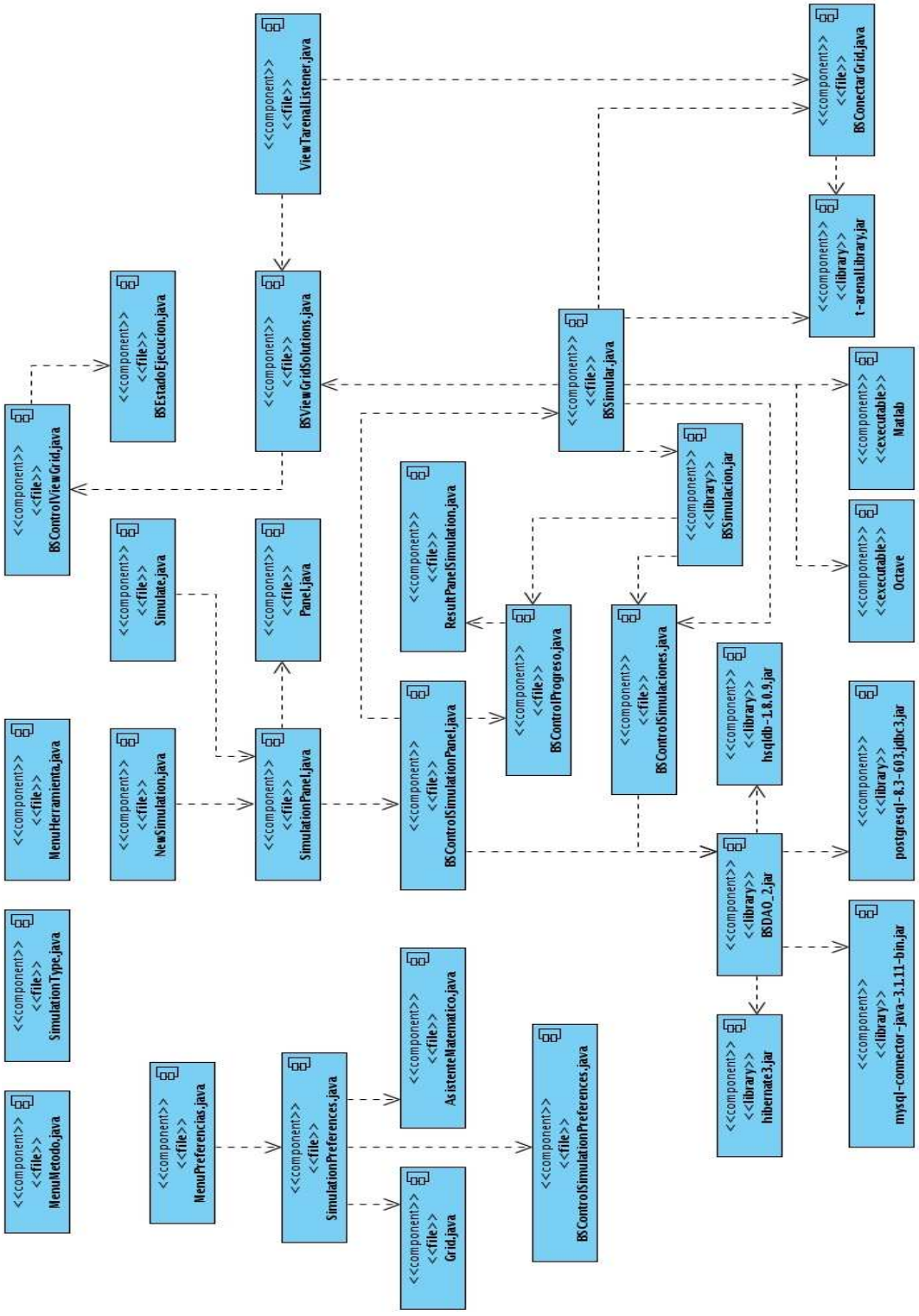
Anexo 3. Diagrama de componentes del plug-in Base de Datos.



Anexo 4. Diagrama de componentes del plug-in Graficador.



Anexo 5. Diagrama de componentes del plug-in Simulador.



Anexo 6. Diseño de las pruebas por cada sección del caso de uso Simular modelo matemático.

SC 1: Simular modelo matemático de forma local.

Variables

1.- Nombre MM: presa-depredador

2.- Valores de variables: x=10, y=10

3.- Valores de parámetros: a=0.04, b=0.0005, c=0.2, d=0.1

4.- Tiempo inicial: t=0

5.- Tiempo final: t=10

6.- Herramienta matemática: Matlab

7.- Método numérico: Ode45

Id del escenario	Escenario	V1	V2	V3	V4	V5	V6	V7	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	Simular un modelo matemático con Matlab de forma local.	V	V	V	V	V	V	V	El sistema valida los datos, realiza la simulación, muestra una barra de progreso y guarda los resultados en la base de datos.	Se realizó la simulación y se guardaron los resultados en la base de datos.

Variables

1.- Nombre MM: presa-depredador

2.- Valores de variables: x=10, y=10

3.- Valores de parámetros: a=0.04, b=0.0005, c=0.2, d=0.1

4.- Tiempo inicial: t=0

5.- Tiempo final: t=10

6.- Herramienta matemática: OdeToJava

7.- Método numérico: ErkTriple

Id del escenario	Escenario	V1	V2	V3	V4	V5	V6	V7	Respuesta del Sistema	Resultado de la Prueba
EC 1.2	Simular un modelo matemático con OdeToJava de forma local.	V	V	V	V	V	V	V	El sistema valida los datos, realiza la simulación, muestra una barra de progreso y guarda los resultados en la base de datos.	Se realizó la simulación y se guardaron los resultados en la base de datos.

Variables

1.- Nombre MM: presa-depredador

2.- Valores de variables: x=10, y=10

3.- Valores de parámetros: a=0.04, b=0.0005, c=0.2, d=0.1

4.- Tiempo inicial: t=0

5.- Tiempo final: t=10

6.- Herramienta matemática: Octave

7.- Método numérico: lsode-stiff

Id del escenario	Escenario	V1	V2	V3	V4	V5	V6	V7	Respuesta del Sistema	Resultado de la Prueba
EC 1.3	Simular un modelo matemático con Octave de forma local.	V	V	V	V	V	V	V	El sistema valida los datos, realiza la simulación, muestra una barra de progreso y guarda los resultados en la base de datos.	Se realizó la simulación y se guardaron los resultados en la base de datos.

Variables

1.- Nombre MM: presa-depredador

2.- Valores de variables: x=10, y=10

3.- Valores de parámetros: a=0.04, b=0.0005, c=0.2, d=0.1

4.- Tiempo inicial: t=0

5.- Tiempo final: t=10

6.- Herramienta matemática: Matlab

7.- Método numérico: Ode45

Id del escenario	**Escenario**	**V1**	**V2**	**V3**	**V4**	**V5**	**V6**	**V7**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1.4	Simular un modelo matemático con Matlab de forma local y el directorio de Matlab no ha sido configurado.	V	V	V	V	V	F	V	El sistema muestra el mensaje ”El directorio de Matlab no existe. ¿Desea que el sistema trate de encontrarlo?”.	Se mostr el mensaje ”El directorio de Matlab no existe. ¿Desea que el sistema trate de encontrarlo?”.

SC 2: Simular modelo matemático de forma distribuida.

Variables

1.- Nombre MM: presa-depredador

2.- Valores de variables: x=10, y=10

3.- Valores de parámetros: a=0.04, b=0.0005, c=0.2, d=0.1

4.- Tiempo inicial: t=0

5.- Tiempo final: t=10

6.- Herramienta matemática: Matlab

7.- Método numérico: Ode45

8.- Usuario de la plataforma T-Arenal: root

9.- Contraseña de la plataforma T-Arenal: administrator

10.- Servidor T-Arenal: 10.7.19.50

Id del escenario	Escenario	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	Respuesta del Sistema	Resultado de la Prueba
EC 2.1	Simular un modelo matemático con Matlab de forma distribuida.	V	V	V	V	V	V	V	V	V	V	El sistema muestra una ventana para loguearse en el servidor T-Arenal. El sistema se conecta al servidor y envía los datos de la simulación. Se muestra una ventana con el estado de la simulación.	Se enviaron los datos correctamente y se mostró una ventana con el estado de la ejecución. Se realizó correctamente la simulación.

Variables

1.- Nombre MM: presa-depredador

2.- Valores de variables: x=10, y=10

3.- Valores de parámetros: a=0.04, b=0.0005, c=0.2, d=0.1

4.- Tiempo inicial: t=0

5.- Tiempo final: t=10

6.- Herramienta matemática: Matlab

7.- Método numérico: Ode45

8.- Usuario de la plataforma T-Arenal: root

9.- Contraseña de la plataforma T-Arenal: administrator

10.- Servidor T-Arenal: 10.7.19.50

Id del escenario	**Escenario**	**V1**	**V2**	**V3**	**V4**	**V5**	**V6**	**V7**	**V8**	**V9**	**V10**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 2.2	Simular un modelo matemático con Matlab de forma distribuida y el servidor T-Arenal esté apagado.	V	V	V	V	V	V	V	V	V	F	El sistema muestra una ventana de error con el mensaje ”Servidor fuera de servicio”.	Se mostró el mensaje de error ”Servidor fuera de servicio”.

GLOSARIO DE TÉRMINOS

Bioinformática: Es la aplicación de los ordenadores y los métodos informáticos en el análisis de datos experimentales y simulación de los sistemas biológicos.

Plug-in: Aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica.

UCI: Universidad de las Ciencias Informáticas.

CIM: Centro de Inmunología Molecular.

SBML: Formato legible por el ordenador para la representación de modelos de procesos biológicos.

VM: Programa ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario.

APIs: Conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

JDBC: Es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede.

RMI: Es un mecanismo ofrecido por Java para invocar un método de manera remota.