

Universidad de las Ciencias Informáticas

Facultad 6



Título: "Extensión de CellDesigner para el acceso a información biológica almacenada en la Web Semántica"

Trabajo de Diploma para Optar por el Título de Ingeniero en Ciencias Informáticas

Autora:

Yanet Pérez Padilla

Tutores:

MSc. Elvismary Molina de Armas

Ing. César Raúl García Jacas

Ing. Edel Moreno Lemus

La Habana, Junio 2012

"Año 54 de la Revolución"

“Nunca consideres el estudio como una obligación sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.”

Albert Einstein

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los _____ días del mes _____ del año _____.

Yanet Pérez Padilla

Firma de la Autora

MSc. Elvismary Molina de Armas

Firma de la Tutora

Ing. César Raúl García Jacas

Firma del Tutor

Ing. Edel Moreno Lemus

Firma del Tutor

Datos del Contacto

Autora:

Yanet Pérez Padilla.

Universidad de las Ciencias Informáticas, Habana, Cuba.

e-mail: yppadilla@estudiantes.uci.cu

Tutores:

MSc. Elvismary Molina de Armas.

Universidad de las Ciencias Informáticas, Habana, Cuba.

e-mail: emolina@uci.cu

Ing. César Raúl García Jacas.

Universidad de las Ciencias Informáticas, Habana, Cuba.

e-mail: crjacas@uci.cu

Ing. Edel Moreno Lemus.

Universidad de las Ciencias Informáticas, Habana, Cuba.

e-mail: emoreno@uci.cu

Agradecimientos

Quizás esta sea la parte más difícil de toda la tesis. Ante todo agradecer a mi familia por los consejos brindados en esta primera etapa de mi vida. A mi madre que me dio la posibilidad de vivir, que siempre espera lo mejor de mí, que ha sabido ser también un excelente padre y que gracias a ella estoy hoy aquí. A mi abuelo Joaquin por sus enseñanzas desde mis primeros días. A mis hermanos y a mi sobrinita por ser mi inspiración. A todos los profesores que han aportado su grano de arena en mi formación profesional. En general a todas las personas que me han ayudado, que han compartido estos años, que supieron ver en mi valores que otros nunca quisieron, que alguna vez se acercaron y preguntaron ¿Qué tal, cómo va la tesis? A todos ustedes les estoy eternamente agradecido y de corazón les digo: muchas gracias.

Dedicatoria

A mi madre, mis hermanos y mi sobrina por ser mi fuente de inspiración.

A mi abuelo Joaquín por todo el amor brindado desde mis primeros días.

A toda mi familia por su apoyo incondicional.

Yanet

Resumen

Las herramientas de modelado, simulación y análisis para redes bioquímicas y reguladoras de genes, son ampliamente utilizadas por la Biología de Sistemas. Estas permiten apoyar las investigaciones sobre el comportamiento de los procesos biológicos y obtener la información necesaria para los especialistas mediante la conexión a las principales bases de datos biológicas.

Con el desarrollo de la Web Semántica se ha logrado vincular información biológica muy importante para los especialistas, tanto la información de las principales bases de datos biológicas, como aquellas que no son tan conocidas por algunos investigadores. Esta integración de la información se logra con el desarrollo de aplicaciones basadas en la Web 3.0.

En el presente trabajo se logra tener acceso a las aplicaciones desarrolladas por la Web Semántica, desde una de las principales herramientas de modelado. La extensión desarrollada a la herramienta CellDesigner permite consultar estas aplicaciones semánticas, posibilitando que los investigadores tengan acceso a la información biológica que está almacenada en múltiples bases de datos existentes. Todas las funcionalidades de la extensión implementada son ejecutadas por el módulo desarrollado en la plataforma de integración SBW (*System Biology Workbench*), que le brinda mayor escalabilidad al resultado de la presente investigación, debido a que este módulo no solo puede ser utilizado por la extensión del CellDesigner sino por todas aquellas herramientas que se encuentren integradas al SBW.

Palabras Claves: Web Semántica o Web 3.0

Índice general

AGRADECIMIENTOS	I
DEDICATORIA	II
RESUMEN	III
INTRODUCCIÓN	1
1. FUNDAMENTO TEÓRICO	5
1.1. HERRAMIENTAS DE MODELADO, SIMULACIÓN Y ANÁLISIS PARA REDES BIOQUÍMICAS Y REGULADORAS DE GENES.	5
1.1.1. INFORMACIÓN GENERAL SOBRE LOS PAQUETES DE SOFTWARE	5
1.1.2. LENGUAJES ESTÁNDAR DE MODELADO PARA ESTAS HERRAMIENTAS.	7
1.1.3. PLATAFORMA PARA LA INTEGRACIÓN DE HERRAMIENTAS DE MODELADO	7
1.1.4. CARACTERÍSTICAS DE LOS PROGRAMAS	8
1.1.5. RESUMEN COMPARATIVO	11
1.2. WEB SEMÁNTICA.	12
1.2.1. LENGUAJES PARA LA REPRESENTACIÓN DE ONTOLOGÍAS EN LA WEB	13
1.2.2. DATOS ENLAZADOS EN LA WEB	13
1.2.3. MOTORES DE BÚSQUEDA ACTUALES Y APLICACIONES DE LA WEB SEMÁNTICA	14
1.2.4. FRAMEWORK PARA CONSTRUIR APLICACIONES DE WEB SEMÁNTICA	16
1.3. OTRAS HERRAMIENTAS Y TECNOLOGÍAS.	17
1.3.1. METODOLOGÍA DE DESARROLLO DE SOFTWARE	18
1.3.2. LENGUAJE DE MODELADO	19

1.3.3.	HERRAMIENTA CASE	19
1.3.4.	IDE Y LENGUAJE DE PROGRAMACIÓN	20
1.4.	CONCLUSIONES	21
2.	CARACTERÍSTICAS Y DISEÑO DEL SISTEMA	22
2.1.	MODELO DE DOMINIO	22
2.2.	BREVE DESCRIPCIÓN DEL SISTEMA	23
2.3.	ESPECIFICACIÓN DE LOS REQUISITOS DEL SISTEMA	24
2.3.1.	REQUISITOS FUNCIONALES	24
2.3.2.	REQUISITOS NO FUNCIONALES	24
2.4.	DEFINICIÓN DE LOS CASOS DE USOS DEL SISTEMA	25
2.4.1.	ACTORES DEL SISTEMA	25
2.4.2.	LISTADO DE CASOS DE USOS DEL SISTEMA	26
2.4.3.	DIAGRAMA DE CASOS DE USOS DEL SISTEMA	27
2.5.	VISTA LÓGICA	28
2.5.1.	SUBSISTEMAS	28
2.5.1.1.	SUBSISTEMA <i>extension</i>	28
2.5.1.2.	SUBSISTEMA <i>module</i>	30
2.6.	REPRESENTACIÓN ARQUITECTÓNICA	32
2.6.1.	ESTILO DE ARQUITECTURA	33
2.6.2.	PATRÓN DE ARQUITECTURA	33
2.6.3.	PATRONES DE DISEÑO UTILIZADOS	34
2.7.	DIAGRAMAS DE SECUENCIA	35
2.7.1.	CU EJECUTAR CONSULTA	36
2.7.2.	CU MEZCLAR RDF	36
2.8.	VISTA DE DESPLIEGUE	37
2.8.1.	DIAGRAMA DE DESPLIEGUE	37
2.9.	CONCLUSIONES.	38
3.	IMPLEMENTACIÓN DEL SISTEMA	39
3.1.	DIAGRAMA DE COMPONENTES	39
3.2.	DIAGRAMA DE DESPLIEGUE DE LOS COMPONENTES	40

3.3. DESCRIPCIÓN DE LOS ALGORITMOS MÁS IMPORTANTES	41
3.3.1. MÓDULO DEL SBW	41
3.4. PRUEBAS AL SISTEMA	43
3.4.1. DISEÑO DE LAS PRUEBAS DE CAJA NEGRA	43
3.4.2. NO CONFORMIDADES DETECTADAS	50
3.5. CONCLUSIONES	51
CONCLUSIONES	52
RECOMENDACIONES	53
REFERENCIAS BIBLIOGRÁFICAS	54
BIBLIOGRAFÍA	57
A. DESCRIPCIÓN DE CASOS DE USOS CRÍTICOS	59

Índice de figuras

1.1. Nube de Datos Enlazados	14
1.2. Arquitectura de Bio2RDF	16
2.1. Diagrama de clases del modelo de dominio	23
2.2. Diagrama de casos de usos del sistema.	27
2.3. Diagrama de casos de usos arquitectónicamente significativos.	27
2.4. Vista general de los subsistemas de la aplicación.	28
2.5. Paquetes relevantes del subsistema <i>extension</i>	29
2.6. Diagrama de clases del diseño del subsistema <i>extension</i>	30
2.7. Paquetes relevantes del subsistema <i>module</i>	31
2.8. Diagrama de clases del diseño del subsistema <i>module</i>	32
2.9. Estilo de arquitectura <i>llamada y retorno</i> aplicado al sistema.	33
2.10. Patrón de arquitectura <i>cliente - servidor</i> aplicado al sistema.	33
2.11. Ejemplo de la aplicación del patrón Chain of responsibility en el sistema.	34
2.12. Ejemplo de la aplicación del patrón Composite en el sistema.	35
2.13. Ejemplo de la aplicación del patrón Singleton en el sistema.	35
2.14. Diagrama de secuencia del CU <i>Ejecutar Consulta</i>	36
2.15. Diagrama de secuencia del CU <i>Transformar RDF a SBML</i>	37
2.16. Diagrama de despliegue del sistema.	37
3.1. Diagrama de componentes del sistema.	40
3.2. Diagrama de despliegue de los componentes.	41

Introducción

La Biología de Sistemas [1] integra toda la información biológica, ofreciendo un mayor entendimiento de las interacciones entre los sistemas vivos y por consiguiente de sus procesos biológicos. Para garantizar esto se han desarrollado modelos matemáticos, simulaciones y técnicas de procesamiento de datos que complementan la estrategia empírica existente en las ciencias biológicas.

Los especialistas de esta rama de la ciencia utilizan una gran variedad de aplicaciones para el modelado, simulación y análisis de redes bioquímicas o reguladoras de genes. La mayoría de estas herramientas se encuentran libremente disponibles como paquetes de software o herramientas en línea. Los ejemplos incluyen WebCell [2] [3], Virtual Cell [4] [5], JDesigner [6], Narrator [7], CellWare [8] y CellDesigner [9] [10], herramientas profesionales que son ampliamente utilizadas en la Biología de Sistemas.

Estas herramientas tienen como objetivo el apoyo a las investigaciones del comportamiento de los procesos biológicos y para ello se han creado lenguajes de modelado propios y estándares que permiten diseñar modelos biológicos. Algunos de los principales lenguajes de modelado son: SBML (System Biology Markup Language) [11] y CellML [12], siendo el primero de ellos el más utilizado por estas herramientas en la actualidad.

La representación biológica de las redes bioquímicas o reguladoras de genes es construida a partir de notaciones gráficas definidas por la herramienta y relacionadas sintácticamente al lenguaje de modelado SBML, permitiendo ser exportadas como fichero para su posterior uso. Estos modelos son almacenados en bases de datos biológicas, permitiendo su análisis, simulación y posible remodelación en caso de ser necesario.

Las principales bases de datos biológicas existentes poseen su propio motor de búsqueda, garantizando el acceso y/o actualización de la información contenida en ellas desde algunas herramientas de modelado. Esta información puede estar relacionada con las características propias de los genes, la localización de la región promotora, y las regiones de unión de factores de transcripción así como la interacción de estos con

otras proteínas. Todos estos elementos ayudan a los especialistas a crear modelos biológicos utilizando las herramientas mencionadas anteriormente, permitiéndoles realizar estudios de correulación e interacción de genes.

El creciente y necesario interés en las investigaciones de Biología de Sistemas ha permitido la evolución y creación de nuevo conocimiento científico, cuya información usualmente es almacenada en bases de datos biológicas. Sin embargo, no todas las instituciones científicas de esta rama disponen de acceso a las principales bases de datos biológicas del mundo, por lo cual utilizan bases de datos propias, que a pesar de contener información que puede ser representativa en el campo y estar disponibles en la web, no están referenciadas o no son reconocidas por la comunidad científica. De esta forma este conocimiento queda fuera del alcance de muchos especialistas en el tema debido a la descentralización de la información biológica.

Con el advenimiento y desarrollo de la Web Semántica [13], los elementos disponibles en la red de redes han sido dotados de “significado”, garantizando así mayor precisión en los resultados obtenidos durante las búsquedas. Sus Datos Enlazados [14] permiten construir la Web de los datos, una gran base de datos interconectados y distribuidos en la Web. Esta información se vincula y se explora a nivel de datos expresados mediante ontologías como lenguajes de representación del conocimiento, lo cual le permite al usuario obtener la información exacta que ha estado buscando. Para definir este tipo de lenguaje se utiliza esencialmente RDF (*Resource Description Framework*) [15] y OWL (*Ontology Web Language*) [16] que ayudan a convertir la Web en una infraestructura capaz de compartir, acceder, integrar y reutilizar información de una forma más sencilla y útil para los usuarios.

Los principios de funcionamiento de la Web Semántica permiten la integración del gran cúmulo de información biológica existente. De esta forma, este valioso conocimiento particionado entre las principales y las no tan conocidas bases de datos biológicas del mundo, puede ser integrado para ganar en exactitud y riqueza durante la búsqueda realizada por los especialistas en el desarrollo de sus investigaciones.

Aunque con el avance de la Web Semántica se han desarrollado aplicaciones que permiten acceder a la información biológica integrada por estas bases de datos, actualmente las herramientas para el modelado, simulación y análisis de redes bioquímicas o reguladoras de genes, no cuentan con la capacidad de consultar las aplicaciones de la Web 3.0 que existen. Esta limitación provoca que los especialistas en la rama no puedan consultar información científica actualizada que se integra con la Web Semántica y así, no obtener resultados de investigación más completos y precisos.

Por todo lo anteriormente descrito se plantea como:

Problema Científico

¿Cómo mejorar el acceso a la información almacenada en múltiples bases de datos desde las herramientas para el modelado de redes bioquímicas o reguladoras de genes?

Objeto de Estudio y Campo de Acción

Para dar respuesta al problema planteado se definió lo siguiente:

Como objeto de estudio: Acceso a la información contenida en bases de datos.

Como campo de acción: Acceso a la información contenida en bases de datos a través de la Web Semántica.

Objetivo General

Desarrollar una extensión que permita acceder a aplicaciones de la Web Semántica desde herramientas para el modelado de redes bioquímicas.

Objetivos Específicos

1. Analizar las principales herramientas para el modelado de redes bioquímicas.
2. Análisis y Diseño de la extensión de CellDesigner.
3. Implementar la extensión de CellDesigner diseñada.
4. Validar la extensión implementada.

Tareas

1. Revisión bibliográfica sobre las aplicaciones para el modelado de redes bioquímicas.
2. Análisis del mecanismo de extensión de CellDesigner.
3. Especificación de los requisitos de la extensión de CellDesigner para proporcionar información biológica obtenida por las aplicaciones desarrolladas por la Web Semántica.

4. Desarrollo de los Casos de Uso del Sistema de la extensión de CellDesigner.
5. Construcción del Diagrama de clases del diseño de la extensión de CellDesigner.
6. Implementación de los Casos de Uso del Sistema.
7. Realización de pruebas de funcionamiento.

Estructura

El presente documento se encuentra estructurado en los siguientes capítulos:

CAPITULO 1. Fundamento teórico: En este capítulo se realiza y se muestra un estudio acerca de las herramientas de modelado y el acceso a la información de los modelos biológicos mediante motores de búsqueda, datos vinculados en la Web Semántica, estándares de representación de la información en la Web y herramientas para su desarrollo. Además se hace una breve valoración de la metodología de investigación para dar cumplimiento al desarrollo de la extensión de la herramienta CellDesigner. Finalmente se hace una descripción de las herramientas y tecnologías a utilizar para desarrollar la solución propuesta.

CAPITULO 2. Características y Diseño del Sistema: En este capítulo se describen las principales características del sistema a desarrollar, sus requisitos funcionales y no funcionales, y los actores que intervienen en el mismo así como el modelo de dominio correspondiente al mismo. Se presenta también el diagrama de casos de usos del sistema, así como una breve descripción de los casos de usos identificados. Además, se dará una descripción del estilo arquitectónico utilizado para el desarrollo del sistema, se describirán los patrones de diseño empleados y los diagramas de clases de diseño e interacción de los casos de usos principales. Se mostrará el diagrama de despliegue del sistema.

CAPITULO 3. Implementación y Prueba del sistema: En este capítulo se describe la implementación del sistema en términos de componentes y la manera en que estos componentes serán desplegados. Además, se explica y se muestra el pseudocódigo de los principales algoritmos implementados en el sistema, así como los resultados de las pruebas realizadas.

Capítulo 1

Fundamento Teórico

En este capítulo se realiza y se muestra un estudio acerca de las herramientas de modelado y el acceso a la información de los modelos biológicos mediante motores de búsqueda, datos vinculados en la Web Semántica, estándares de representación de la información en la Web y herramientas para su desarrollo. Además se hace una breve valoración de la metodología de investigación para dar cumplimiento al desarrollo de la extensión de la herramienta CellDesigner. Finalmente se hace una descripción de las herramientas y tecnologías a utilizar para desarrollar la solución propuesta.

1.1. Herramientas de modelado, simulación y análisis para redes bioquímicas y reguladoras de genes.

1.1.1. Información general sobre los paquetes de software

Actualmente existen un conjunto de herramientas para el modelado de redes bioquímicas y/o de regulación genética. La mayoría de estos programas comparten características pero presentan funcionalidades específicas respecto a la fiabilidad, eficiencia, interfaces gráficas de usuarios y compatibilidad. En la literatura estos software son clasificados en simuladores independientes (*standalone simulators*) y servidores de simulaciones en Internet (*Internet simulation servers*).

Los simuladores independientes (por ejemplo, CellDesigner, CellWare, COPASI) almacenan los modelos y realizan el cálculo numérico en las propias computadoras de los usuarios. Todas las tareas son realizadas localmente y para acceder a las mismas se hace necesario entrar a dicha PC, razón por la cual compartir los resultados o modelos obtenidos debe realizarse por otros medios. La variante más empleada para compartir

los resultados, es a través de un sistema de fichero compartido sobre la Web, lo cual requiere experiencia sobre este tema por parte de los usuarios.

Los sistemas bajo la clasificación anterior van desde los simuladores con interfaces de usuarios amigables hasta relativamente complejos ambientes de modelación. Los primeros son ideales para usuarios no expertos y son muy eficaces para tareas de modelado típico. Además presentan interfaces gráficas intuitivas, y en el caso que carezcan, permite modelos que pueden ser configurados mediante interfaces de texto y de sintáxis simples. Por otro lado los ambientes de modelación (por ejemplo, JSIM) son más apropiados para grandes proyectos, ya que ofrecen funcionalidades para la configuración modular de los modelos y automatización de tareas repetitivas. Su principal inconveniente es que requieren de un profundo conocimiento del lenguaje de modelación específico para cada software.

Los servidores de simulaciones en Internet (por ejemplo Virtual Cell) realizan las tareas en servidores remotos, este enfoque posibilita a varios investigadores en distintas partes del mundo colaborar en el desarrollo y análisis de los modelos. Esto trae como consecuencia que los resultados dependan de la carga de trabajo de los servidores y de la congestión de Internet. Los sistemas bajo esta clasificación son menos indicados para tareas con baja demanda. El compartimiento de información es realizado mediante ambientes de trabajo colaborativos en Internet, lo cual requiere de conocimiento básico en la Informática.

Todos los sistemas revisados pueden ser ejecutados en el sistema operativo Windows, aunque existen versiones para GNU/Linux, MAC, entre otros. Muchos de ellos requieren la máquina virtual de Java (JVM) para su ejecución.

Para la definición de los modelos existen tres tipos de interfaces de usuarios: las basadas en esquemas (por ejemplo, CellDesigner, CellWare, Narrator, JDesigner), las basadas en cajas de diálogo (por ejemplo, COPASI, GEPASI) y las basadas en formato textual (por ejemplo, Jarnac, BASIS). Las interfaces de formato textual utilizan información escrita directamente por los usuarios, quienes deben entrar los datos a través del teclado, lo cual conlleva a errores tipográficos. Es por esto que se recomienda más el uso de las interfaces basadas en cajas de diálogos e interfaces basadas en esquemas, las que brindan una forma más fácil de escribir la información de las ecuaciones, utilizando gráficos y reduciendo considerablemente el riesgo de error.

Dado la gran variedad de programas de simulación se hace necesario confiar en una misma estructura para los modelos compartidos por estas herramientas utilizadas. Con este fin existen lenguajes estándares tales como SBML (*Systems Biology Markup Language*) [11] y CellML [12] como formatos para poder exportar/importar modelos creados en distintos sistemas y que son soportados por una amplia variedad de

programas. Como un elemento importante a resaltar es que existen aplicaciones tales como CellDesigner y JDesigner que logran un alto nivel de integración mediante SBW (*Systems Biology Workbench*), que es un marco de trabajo que permite a los programas comunicarse y utilizar funcionalidades de otros en tiempo de ejecución.

1.1.2. Lenguajes estándar de modelado para estas herramientas.

SBML (*System Biology Markup Language*)

System Biology Markup Language [11] es un lenguaje basado en XML (*eXtensible Markup Language*), para representar modelos de una reacción de redes bioquímicas y reguladora de genes. SBML puede representar redes metabólicas, caminos celulares, redes reguladoras, y otras clases de sistemas estudiados dentro de la Biología de Sistema. SBML tiene tres propósitos principales, permitir el uso de múltiples herramientas de software sin reescribir modelos para cada una de ellas; permitiendo a modelos ser compartidos y publicados de forma que otros investigadores puedan utilizarlo incluso en un diverso ambiente de software y asegurar la supervivencia del modelo. El propósito de SBML no es definir una lengua universal para los modelos cuantitativos. El propósito de SBML es servir como un lenguaje estándar de intercambio usado por diversas herramientas actuales de software para comunicar los aspectos esenciales de un modelo de cómputo.

CellML

CellML [12] es un lenguaje estándar sobre XML, fue desarrollado por la Universidad de Auckland y afiliado por un grupo de investigadores. El propósito de CellML es almacenar e intercambiar modelos matemáticos basados en computadoras.

Permite a los científicos compartir modelos, incluso si están utilizando software de diferentes modelos de construcción. También les permite reutilizar los componentes de un modelo a otro, lo que acelera la construcción de modelos.

1.1.3. Plataforma para la integración de herramientas de modelado

SBW (*System Biology Workbench*)

System Biology Workbench 2.8.1 [17] es un marco de trabajo que permite escribir aplicaciones heterogéneas en diversos lenguajes de programación y ejecutarlas sobre diferentes plataformas para comunicar y usar otras capacidades a través de sistemas binarios rápidos. SBW es una infraestructura de código abierto, simple de implementar y entender. Las aplicaciones disponibles en SBW se comunican mediante un protocolo de

red simple. Las interfaces del sistema son encapsuladas en librerías del lado del cliente para los diferentes lenguajes de programación. SBW consiste en dos componentes, un *Broker* para rutear los mensajes y módulos que envían y reciben mensajes. Todas las conexiones entre los módulos y el *Broker* son a través de *sockets* TCP/IP. Todos los mensajes son enviados en formato binario para lograr un alto rendimiento. Los mensajes necesitan ser enviados entre dos computadoras diferentes, entonces los mensajes son enviados primeramente al *Broker* sobre la máquina remoto, y este entonces envía el mensaje al módulo remoto correcto. Los módulos pueden ser escritos en una variedad de lenguajes, incluyendo, Java, C/C++, Delphi, Perl, Python y Matlab.

1.1.4. Características de los programas

CellDesigner

CellDesigner 4.1 [9] [10] es un editor estructurado de diagramas para dibujar redes bioquímicas y de regulación genética. Las redes son dibujadas con base en el diagrama de proceso utilizando la notación gráfica [18] y son guardadas usando SBML. Es posible ligar estas redes a simulaciones y con otros paquetes de análisis por medio de SBW.

CellDesigner también es compatible con la simulación y la búsqueda de parámetros, con el apoyo de la integración con SBML ODE Solver (SOSlib) [19] , permitiendo a los usuarios simular a través de la sofisticada interfaz gráfica de usuario. CellDesigner fue implementado en Java y, por tanto compatible con varias plataformas (es decir, Windows, Linux y MacOS X). CellDesigner está disponible gratuitamente a través de su Sitio Web. Las principales características de esta herramienta son:

1. Presenta una amplia representación de semánticas bioquímicas.
2. Descripción detallada del estado de transición de las proteínas.
3. Compatibilidad con el estándar SBML (en su última versión permite SBML Nivel 2 Versión 4 y validación de los archivos cuando son abiertos haciendo uso de libSBML [20] 3.4.1)
4. Integración con el SBW permitiendo los módulos de análisis y simulación.
5. Integración con la librería de simulación nativa (SBML ODE Solver).
6. Capacidad de conexión con Base de Datos (en su última versión permite conectarse a base de datos tales como PANTHER Pathway [21] y MetaCyc [22]).
7. Brinda un mecanismo para el desarrollo de plug-in que posibilita adicionarle nuevas funcionalidades a la herramienta.

CellWare

CellWare 3.0.1 [8] es una herramienta de modelaje y simulación desarrollada por el Instituto de Bioinformática de Singapur. No solo ha sido diseñada para conducir el modelaje y la simulación de reguladores genéticos y caminos metabólicos, sino además ofrece un ambiente integrador para diversas representaciones matemáticas, parámetros de estimación y optimización. Además tiene una pantalla gráfica muy amistosa para los usuarios y la capacidad de ejecutar largos y complejos modelos.

Una característica muy especial de Cellware es que es la primera herramienta de modelación y simulación basada en tecnología Grid en el campo de sistemas biológicos. Utiliza un formato propietario (Cellware model o CWN) para almacenar la información perteneciente al modelo y ambiente de simulación correspondiente. También puede importar modelos desde SBML de nivel 1 y 2.

Posibilita conexión a base de datos, en su última versión a la base de datos KEGG [23]. Esta herramienta es desarrollada en Java y ha sido exitosamente probado en Windows, Linux y Mac. Incluye una extensa librería de algoritmos estocásticos (Gillespie, Gibson y Stochsim) y determinísticos (Forward Euler y Runge Kutta).

Narrator

Narrator [7] es una herramienta de software que facilita el desarrollo y simulación de sistemas biológicos como Modelos de Co-dependencia. La Metodología de Co-dependencia complementa la representación de transporte y transformación de especies junto con un mecanismo explícito para expresar el procesamiento de la información biológica. Por lo tanto, los modelos de Co-dependencia explícitamente capturan, por ejemplo, procesamiento de señal de estructuras y la influencia de los factores exógenos o eventos que afectan alguna parte de un sistema biológico o proceso.

Este conjunto de características combinadas proporcionan al biólogo una poderosa herramienta para describir y explorar la dinámica de los fenómenos de la vida. Detrás de la amigable interfaz gráfica de usuario, Narrator oculta una característica flexible que hace que sea relativamente fácil para modelos de mapa definidos a través de la notación gráfica de lenguajes matemáticos, tales como Ecuaciones Diferenciales Ordinarias (ODE por sus siglas en inglés), SBML o el método directo de Gillespie. Esta aplicación fue desarrollada sobre el lenguaje Java y su código fuente es de acceso libre.

Sentero

Sentero [24] es una herramienta de modelación que soporta SBML para el análisis dinámico de redes

bioquímicas. Es un sistema de modelación compatible con SBML para el análisis de rutas metabólicas. La herramienta proporciona una representación de red interactiva para la construcción y visualización de largos modelos cinéticos.

Una característica clave del sistema es la construcción de análisis de sensibilidad. Esto permite que los efectos de incertidumbre de los parámetros sobre el modelo de salida sean investigados rigurosamente usando técnicas de sensibilidad locales o globales. Debido a que estos métodos requieren de simulaciones repetitivas, emplea el ODE Solver de Matlab proporcionando un motor de simulación eficiente computacionalmente. Corre bajo Windows NT, XP y Vista.

WebCell

WebCell [2] [3] es un entorno Web para gestionar la información cualitativa y cuantitativa sobre las redes celulares y explorar interactivamente su comportamiento dinámico y estático en respuesta a sistemáticas perturbaciones. Esta herramienta presenta una interfaz Web amigable, permitiendo que los usuarios puedan construir, visualizar, analizar y almacenar los modelos de cadenas de reacción. Una librería de modelos es también disponible para proveer todas las implicaciones para las dinámicas celulares de los modelos publicados.

Este sistema está diseñado con las tecnologías JSP (*Java Server Page*) y *Java Applet-Servlet*. Soporta SBML (importa y exporta) y MATLAB (exporta). Una vez que el modelo cinético es construido, el comportamiento dinámico del sistema celular puede ser explorado mediante la solución de ODE o ecuaciones diferenciales algebraicas (DAEs, por sus siglas en inglés).

JDesigner/Jarnac

JDesigner 2.0.39 [6] es una aplicación que permite dibujar una red bioquímica y exportar la red a formato SBML. Tiene una interfaz SBW que le permite ser llamado desde otros módulos compatibles con SBW, por ejemplo Python. Además, tiene la capacidad de utilizar Jarnac 2.0 como un servidor de simulación (a través de SBW) y por lo tanto permite que los modelos sean ejecutados desde JDesigner. De esta manera es tanto una herramienta de diseño de redes como un simulador. Los binarios y código fuente completo están disponibles para descarga gratuita.

Jarnac [25] es un entorno interactivo basado en Windows (95/98/NT) para estudiar modelos redes químicas y bioquímicas. Puede ser usado por ejemplo para estudiar los sistemas metabólicos, las redes reguladoras de genes o rutas de transducción de señales. Soporta un lenguaje basado en texto que puede

ser usado para describir e interrogar modelos metabólicos.

Virtual Cell

Virtual Cell 4.7 [5] [4] es un entorno computacional para el modelado y simulación de biología celular. Ha sido específicamente diseñado para ser una herramienta para una amplia gama de científicos, desde biólogos celulares experimentales hasta biofísicos teórico. La creación de modelos biológicos o matemáticos puede ir de lo simple, evaluar hipótesis o interpretar datos experimentales, hasta complejos modelos de varias capas para sondear el comportamiento previsto de los sistemas complejos, altamente no lineales. Estos modelos se pueden basar en datos puramente experimentales y supuestos teóricos. Esta herramienta es desplegada como una aplicación distribuida que se utiliza a través de Internet. Los usuarios pueden construir complejos modelos con un interfaz Java basada en la Web para especificar la topología y la geometría, las características moleculares y los parámetros relevantes en la interacción.

Virtual Cell convierte automáticamente la descripción biológica en un sistema matemático de ecuaciones diferenciales ordinarias y/o parciales. Distintos marcos biológicos y matemáticos se engloban dentro de una interfaz gráfica única. Los resultados obtenidos pueden ser visualizados y analizados en línea o descargados en el ordenador del usuario en una variedad de formatos.

1.1.5. Resumen Comparativo

En las siguiente tabla se muestra el resumen comparativo de las herramientas de modelado descritas anteriormente.

Tipo de Software	Programa	Sistema operativo	Formato de importación	Formato de exportación	Integrada al SBW	Código abierto
Simuladores individuales	CellDesigner	Windows, Linux y OS X	SBML	SBML	Si	No
	CellWare	Windows, Linux y OS X	SBML	SBML	No	No
	Narrator	Windows, Linux y OS X	SBML	SBML	No	Si
	Sentero	Windows NT, XP y Vista	SBML	SBML	No	Por petición al autor
Ambientes de modelación	JDesigner/Jarnac	Windows y Linux	SBML y MATLAB	SBML, GML y MATLAB	Si	Si

Servidores de simulación en Internet	Web Cell	Multiplataforma	SBML	SBML y MATLAB	No	No
	Virtual Cell	Multiplataforma	SBML, CellML y MATLAB	SBML, CellML y MATLAB	No	No

Tabla 1.1: Resumen comparativo.

Todos los sistemas comparados pueden ser ejecutados en el sistema operativo *Windows*, aunque existen versiones para GNU/Linux, MAC, entre otros. Además, muchos de ellos requieren la máquina virtual de Java (JVM) para su ejecución. También, cada una de las herramientas trabajan con el lenguaje de modelado SBML, aunque no todas están vinculadas a la plataforma de integración SBW. Su código abierto no está disponible en la mayoría de estas aplicaciones, aunque en su gran totalidad se encuentran libremente disponibles como paquetes de *software* o herramientas en línea.

La herramienta CellDesigner no posee un código abierto disponible, pero brinda un mecanismo para el desarrollo de extensiones que posibilita adicionarle nuevas funcionalidades a la misma. Además se encuentra integrada a la plataforma SBW que es un marco de trabajo que permite que los programas puedan comunicarse y utilizar funcionalidades de otros en tiempo de ejecución.

1.2. Web Semántica.

La Web Semántica [13] permite superar las limitaciones de la Web actual mediante la introducción de descripciones explícitas del conocimiento existente en Internet. Con el crecimiento de los recursos y la ausencia de una organización estructural de la Web actual, la Web Semántica aboga por clasificar y estructurar los recursos con semántica explícita procesable por las máquinas. Su estructura está formada por una red de nodos tipificados e interconectados mediante clases y relaciones definidas por una ontología.

Una ontología es una jerarquía de conceptos con atributos y relaciones que permiten definir redes semánticas de unidades de informaciones interrelacionadas. Finalmente, cualquier aplicación que navegue por esta Web Semántica podría reconocer las distintas unidades de información descritas por estas ontologías, obteniendo datos específicos o razonamientos sobre relaciones complejas.

1.2.1. Lenguajes para la representación de ontologías en la Web

Para obtener una adecuada definición de los datos, la Web Semántica utiliza algunos lenguajes como el RDF [15] y OWL [16], mecanismos que ayudan a convertir la Web en una infraestructura global en la que es posible compartir, y reutilizar datos y documentos entre diferentes tipos de usuarios.

RDF (*Resource Description Framework*)

RDF [15] es un lenguaje para la definición de ontologías y metadatos en la Web. RDF es hoy el estándar más popular y extendido en la comunidad de la Web semántica. El elemento de construcción básica en RDF es el “triple” o sentencia, que consiste en dos nodos (sujeto y objeto) unidos por un arco (predicado), donde los nodos representan recursos, y los arcos propiedades. Proporciona información descriptiva simple sobre los recursos que se encuentran en la Web y que se utiliza, por ejemplo, en catálogos de libros, directorios, colecciones personales de música, fotos, eventos, etc.

OWL (*Ontology Web Language*)

OWL [16] es un mecanismo para desarrollar temas o vocabularios específicos en los que asociar esos recursos. Lo que hace OWL es proporcionar un lenguaje para definir ontologías estructuradas que pueden ser utilizadas a través de diferentes sistemas. Las ontologías, que se encargan de definir los términos utilizados para describir y representar un área de conocimiento, son utilizadas por los usuarios, las bases de datos y las aplicaciones que necesitan compartir información específica, es decir, en un campo determinado como puede ser el de las finanzas, medicina, deporte, etc. Las ontologías incluyen definiciones de conceptos básicos en un campo determinado y la relación entre ellos.

1.2.2. Datos enlazados en la Web

Los Datos Enlazados [26] es la forma que tiene la Web Semántica de vincular los distintos datos que están distribuidos en la Web, de forma que se referencian de la misma forma que lo hacen los enlaces de las páginas Web.

La Web Semántica no se trata únicamente de la publicación de datos en la Web, sino que éstos se pueden vincular a otros, de forma que las personas y las máquinas puedan explorar la Web de los datos, pudiendo llegar a información relacionada que se hace referencia desde otros datos iniciales.

De la misma forma que la Web del hipertexto son enlaces de relaciones entre puntos de los documentos escritos en HTML (*Hypertext Markup Language*), la Web de los datos se construye mediante documentos

en la Web. Sin embargo, y a diferencia de la Web del hipertexto, la Web Semántica enlaza los datos de manera arbitraria mediante recursos RDF.

Los Datos Enlazados permiten construir la Web de los datos, una gran base de datos interconectada y distribuida en la Web, como se muestra en la Figura 1.1 [14]. Los datos se vinculan y se exploran de una forma similar a la utilizada para vincular los documentos HTML.

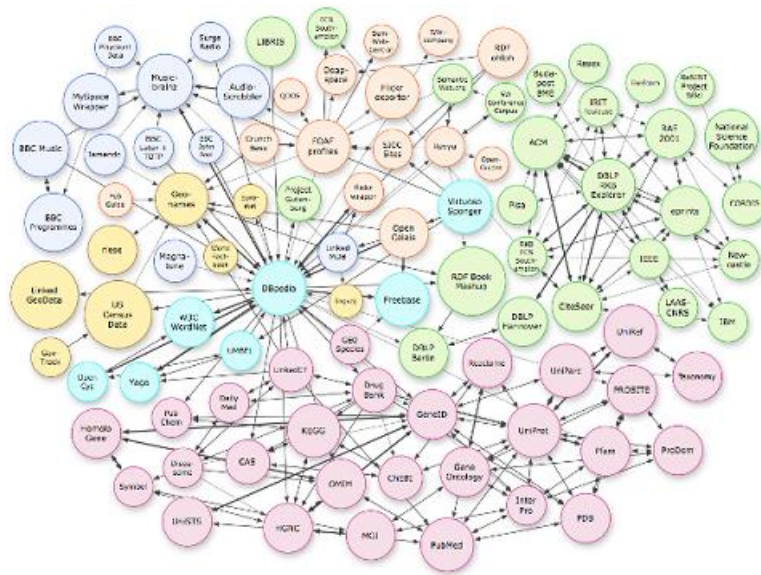


Figura 1.1: Nube de Datos Enlazados

La interconexión descrita permite reutilizar la información de cualquier manera esperada o inesperada, lo que ofrece un valor añadido a la Web.

1.2.3. Motores de búsqueda actuales y aplicaciones de la Web Semántica

Una vía rápida de buscar información en la Web es a través de los motores de búsqueda, el resultado generalmente es una extensa lista de hipervínculos sugeridos de acuerdo al criterio de búsqueda del usuario y no a partir del contexto semántico o la interpretación humana. Cada año son publicadas nuevas bases de datos biológicas; dado este amplio crecimiento se hace necesaria la implementación de un sistema capaz de consultar estas bases de datos sin necesidad de centralizarlas en un repositorio.

La Web Semántica tiene como objetivo principal aprovechar la información biológica disponible sin necesidad de centralizarla. Para resolver el problema de la integración de los datos, la comunidad de la Web Semántica propone una solución basada en los estándares RDF para documentos y OWL como especificación de ontologías. RDF y OWL generan una serie de entidades llamadas “triple” en forma de sujeto,

predicado y objeto. Los sistemas de bases de datos capaces de manipular estas entidades son conocidas como “Triplestore”. Se han desarrollado nuevos *frameworks* que permiten utilizar estos sistemas, algunas se encuentran aún en fase de desarrollo, otras se encuentran en condiciones de ser utilizadas en sistemas de producción, como son Sesame [27] y Jena [28]. Los *frameworks* acceden a los sistemas de Web Semántica mediante SPARQL Endpoint [29], que constituyen la interfaz de consulta de estos sistemas.

NCBI's Entrez

NCBI's Entrez [30] utiliza todas las bases de datos organizadas por el NCBI (*National Center for Biotechnology Information*), su enfoque de integración de datos está basado en hipervínculos y se ilustra con su esquema de base de datos. Cada una de las base de datos contienen registros con campos predefinidos, índices en cada campo, y viene con una interfaz que permite consultas booleanas específicas sobre el terreno de la búsqueda. Los enlaces pueden ser especificados por los registros de la misma base de datos (enlaces dentro de la base de datos), o entre registros en diferentes bases de datos (enlaces entre bases de datos) del NCBI.

Kegg's DBGET

Kegg's DBGET [31] es un motor de búsqueda dedicado a los genes y rutas metabólicas, utiliza un sistema de recuperación de base de datos integrada para las principales bases de datos biológicas, que se clasifican en cinco categorías:

1. Base de datos KEGG en DBGET.
2. Otras bases de datos DBGET.
3. Bases de datos de búsquedas en la Web
4. Bases de datos sólo de enlace en la Web
5. Bases de dato PubMed.

Bases de datos en la tercera categoría se integran para la búsqueda de palabras clave, pero los datos reales deben ser obtenidos de los sitios originales. Bases de datos de la cuarta categoría sólo están disponibles en el sistema LinkDB¹. PubMed es una base de datos solamente de enlaces, pero usa el servicio del NCBI con el fin de integrar mejor Kegg y otras bases de datos de DBGET.

¹Es un sistema centralizado para administrar los enlaces, registro de quién, cómo y cuándo se accede a un enlace.

Bio2RDF

Bio2RDF [32] [33] es una aplicación Web Semántica desarrollada para ofrecer solución al problema de la integración del conocimiento en Bioinformática. Este sistema emplea documentos RDF y una lista de reglas para crear URI's (*Uniform Resource Identifier*) que permitirán la creación de datos vinculados. Puede ser vista como una aplicación de tipo mashup porque combina datos procedentes de más de una fuente de información externas. Bio2RDF integra información pública disponible de algunas de las más reconocidas bases de datos biológicas. Bio2RDF puede ser descrito como una aplicación mashup de datos, utilizando un enfoque de Web semántica para la integración de los datos o conocimiento.

La arquitectura de Bio2RDF fue diseñada con posibilidad de ampliación, Figura 1.2 [32]. Además los servicios Web de Bio2RDF, permiten a los usuarios integrar datos locales y privados; y vincularlo con el espacio de conocimiento del Bio2RDF.

Nuevas fuentes de base de datos pueden ser agregadas al sistema fácilmente. Una vez que se ha escrito un nuevo programa para una base de datos pública, podría ser presentado al equipo del proyecto Bio2RDF para su inclusión en el servicio público del mismo.

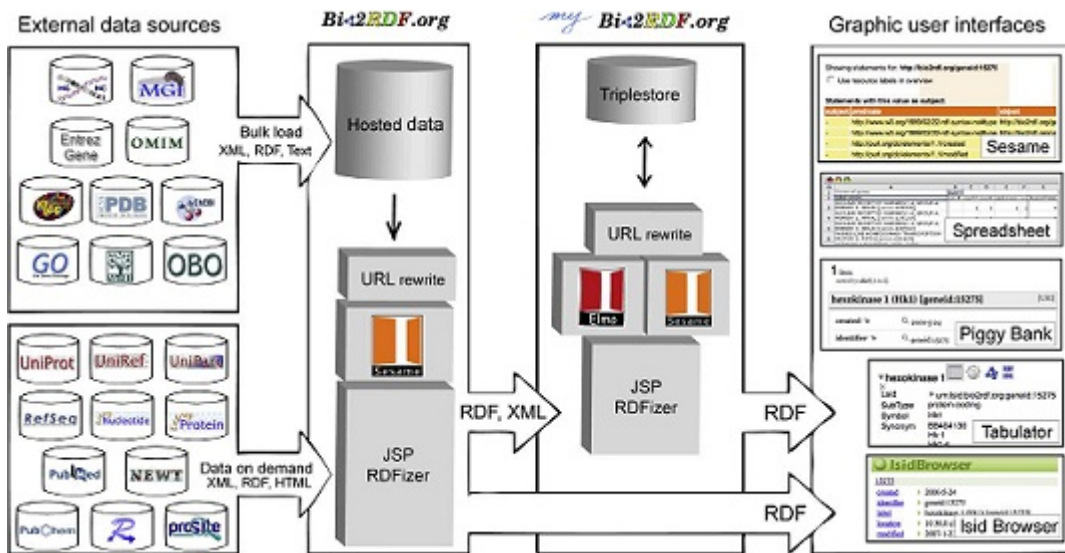


Figura 1.2: Arquitectura de Bio2RDF

1.2.4. Framework para construir aplicaciones de Web Semántica

Para el desarrollo de aplicaciones que permiten gestionar toda la información almacenada en la Web Semántica existe un número considerable de framework. En esta sección se hará énfasis de los que utilizan el lenguaje de programación Java.

Jena

Jena 2.6.3 [28] es un *framework* de código abierto en Java para construir aplicaciones de la Web Semántica. Provee un entorno de programación para RDF, RDFS (RDF *Schema*)², OWL, SARQL (*Protocol and RDF Query Language*)³ e incluye un motor de inferencia basado en reglas.

El framework incluye:

- Un API (*Application Programming Interface*) RDF
- Lectura y escritura de RDF en RDF / XML, N3 y N-Triples
- Un API OWL
- Almacenamiento persistente y en memoria.
- Motor de consultas SPARQL

Sesame

Sesame 2.0 [27] es un *framework* para Java, es decir, un entorno para el desarrollo de aplicaciones en el lenguaje de programación Java para la Web semántica. Es un marco de desarrollo para almacenamiento, consulta y razonamiento con RDF y RDF Schema.

SeRQL (Sesame RDF *Query Language*) es un lenguaje de consulta en RDF o RDF Schema que en la actualidad está siendo desarrollado como parte de Sesame y combina los mejores aspectos de otros lenguajes de consulta.

Sesame puede ser usado como base de datos para RDF y RDF *Schema*, o como una librería de Java para aplicaciones que necesitan trabajar internamente con RDF. De manera más general, Sesame proporciona a los desarrolladores de aplicaciones un conjunto de herramientas muy útil para hacer cualquier cosa por uno mismo con RDF.

1.3. Otras herramientas y tecnologías.

Existen creencias de que un grupo de desarrollo debería organizarse en torno a las habilidades de los individuos altamente calificados, que saben como hacer el trabajo y lo hacen bien y que raramente necesitan

²Esquemas RDF.

³Lenguaje de consultas para RDF.

dirección. Esto constituye una equivocación en la mayoría de los casos, y un grave error en el desarrollo de *software*. Por lo tanto, es necesario un proceso que esté ampliamente disponible de forma que todos los interesados puedan comprender su papel en el desarrollo en el que se encuentran implicados. Todo esto unido a una correcta selección de las herramientas, métodos, técnicas y procedimientos que ayuden a obtener un producto de elevada calidad.

1.3.1. Metodología de desarrollo de software

Uno de los principales retos que enfrentan los desarrolladores de *software* es obtener un producto con calidad y de manera eficiente, lo cual supone un paso importante hacer una selección correcta de las herramientas y procesos necesarios. Con este propósito se crearon diferentes metodologías de desarrollo, las cuales son un conjunto de pasos y procedimientos que deben seguirse. Con los años y las experiencias que han ido teniendo las diferentes instituciones productoras de *software*, se han ido mejorando algunas metodologías y se han creado otras con el objetivo de aumentar la productividad.

Entre la familia de metodologías existen las denominadas ágiles, las cuales intentan evitar los intrincados y burocráticos caminos de las metodologías tradicionales, enfocándose en las personas y los resultados. Estas metodologías recogen ventajas de las metodologías tradicionales e incorporan características nuevas que hacen que el proceso de desarrollo sea más simple, basándose en que lo más importante en un proyecto es valorar más a los individuos que a los procesos y herramientas, al *software* que funciona más que a la documentación exhaustiva, a la colaboración del cliente más que a la negociación contractual, a la respuesta al cambio más que al seguimiento de un plan.

Elegir la metodología adecuada es vital para lograr un sistema de alta calidad en un tiempo razonablemente corto. Existen varias metodologías de desarrollo de *software*, dentro de las cuales se encuentran RUP (Proceso Unificado del Rational), XP (*Extreme Programming*) y OpenUP ().

Esta última es una de las metodologías ágiles de desarrollo de *software*. Ofrece las mejores prácticas de una variedad de líderes en ideas sobre producción de *software* y comunidades de desarrollo que cubren un diverso conjunto de perspectivas y necesidades de desarrollo. Preserva las características esenciales de RUP que incluye el desarrollo interactivo, casos de uso y escenarios de conducción de desarrollo, la gestión de riesgo y el enfoque centrado en la arquitectura.

OpenUP/Basic es la forma más ágil y ligera de OpenUP, y se basa en una donación de código abierto al proceso de contenido, conocido como el Proceso Unificado Básico (BUP). La mayoría de las partes de RUP han sido excluidas de esta metodología y muchos elementos se han fusionado, siendo el resultado un

proceso mucho más sencillo que coincide con los principios de RUP. OpenUP/Basic es aplicable a proyectos pequeños con grupos de 3 a 6 personas interesadas en el desarrollo rápido e interactivo.

Además de lo anterior, OpenUP/Basic define un proceso de desarrollo de *software* mínimo y completo. Mínimo porque solamente lo fundamental es incluido dentro del proceso, y completo porque define un conjunto de componentes que guían y definen dicho proceso de desarrollo hasta la obtención del producto. Es extensible, de manera que se pueden añadir artefactos, actividades u otro componente a la metodología, según lo requiera el sistema que se desarrolla. Por las razones antes expuestas y por decisión del Grupo de Bioinformática de DATEC de la UCI, OpenUP/Basic constituye la metodología seleccionada para el desarrollo de la aplicación que se propone en el presente trabajo.

1.3.2. Lenguaje de modelado

Para dar solución al problema planteado usaremos como lenguaje de modelado UML (Lenguaje Unificado de Modelado) [34] [35]. Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios.

El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Es utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos.

1.3.3. Herramienta CASE

Las herramientas CASE (*Computer Aided Software Engineering*) son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo del sistema de información, completamente o en algunas fases. Como ejemplo de herramientas CASE tenemos MagicDraw, Rational Rose, Umbrello, Visual Paradigm for UML, ArgoUML y otras. Por decisión del Grupo de Bioinformática la Herramienta CASE a utilizar es Visual Paradigm.

Visual Paradigm 8.0 es una herramienta CASE que soporta la última versión del Lenguaje de Modelado

Unificado (UML) y la Notación del Proceso de Modelado de Negocio (BPMN), y genera código para un gran número de lenguajes de programación. Además brinda una versión libre para uso no comercial. La herramienta fue desarrollada para una amplia gama de usuarios incluyendo ingenieros de software, analistas de sistemas, analistas del negocio y arquitectos de sistemas. Permite la integración con varias herramientas de Java, y brinda además una gran interoperabilidad con otras herramientas CASE como Rational Rose.

1.3.4. IDE y lenguaje de programación

El IDE (Entorno de Desarrollo Integrado) Eclipse Europa 3.3 es un entorno de desarrollo de Java que emplea módulos (en inglés plugin) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están generalmente prefijadas, las necesite el usuario o no. El mecanismo de módulos permite que el entorno de desarrollo soporte otros lenguajes de programación además de Java, así como introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo.

Este IDE fue seleccionado principalmente por su potente editor de código, la interfaz amigable que presenta y la existencia en el ciberespacio de una gran cantidad de plug-ins que hacen del IDE una herramienta potente. Entre otras de sus características tenemos que es un *software* libre y multiplataforma.

El lenguaje de programación seleccionado es Java. La razón principal de esta selección es que es independiente de plataforma y arquitectura de red. Por eso una aplicación escrita en Java, puede ejecutarse en cualquier sistema.

En los primeros días de Java, gran parte de sus críticas se originaban de su pobre rendimiento respecto a lenguajes nativos como C y Fortran. Mucho ha cambiado desde entonces. Actualmente se han producido enormes mejoras en el rendimiento de la Java Virtual Machine (JVM), principalmente atribuida a la introducción del compilador *just-in-time* y la tecnología *hotspot* [36]. Estas mejoras han dado lugar a que la ejecución de la JVM, sea comparable a la de otros lenguajes nativos [37]. Otro de los aspectos más importante de Java es el modelo de seguridad [38] [39]. Este simplifica enormemente la implementación de una estricta política de seguridad para una aplicación Java. Esto posibilita que las aplicaciones puedan cargar dinámicamente código, sin tener que preocuparse por las posibles implicaciones. Además de las características mencionadas con anterioridad, Java constituye un lenguaje “*simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico*” [40].

1.4. Conclusiones

Después del análisis comparativo entre algunas herramientas de modelado, se llegó a la conclusión que la herramienta CellDesigner 4.1 se encuentra entre las principales aplicaciones profesionales que son ampliamente utilizadas en la Biología de Sistemas. Está vinculada al marco de trabajo SBW permitiéndole el acceso a los módulos desarrollados sobre este marco. El CellDesigner permite además implementar nuevas extensiones a la herramienta.

Las bases de datos biológicas donde se encuentra la información necesaria para los investigadores, están integradas a las aplicaciones desarrolladas por la Web Semántica. Por lo tanto, el acceso a la información biológica se realizará mediante aplicaciones como el Bio2RDF. Para lograr este acceso a las aplicaciones de la Web 3.0 se utilizará el framework Jena, de código abierto en Java y que puede trabajar en varios lenguajes de representación de la información, tales como, RDF y OWL. Presenta además un motor para consultas SPARQL, que son las consultas más utilizadas en estas aplicaciones de Web Semántica.

La aplicación para acceder a la información biológica necesaria se integrará como un nuevo módulo del SBW y así podrá ser utilizado genéricamente por otras herramientas de modelado que se encuentre integrada a este marco de trabajo. Mientras que la extensión que se implementará para CellDesigner, será la interfaz del módulo implementado en el SBW.

Capítulo 2

Características y Diseño del Sistema

En este capítulo se describen las principales características del sistema a desarrollar, sus requisitos funcionales y no funcionales, y los actores que intervienen en el mismo. Además, se presenta el diagrama de casos de usos del sistema, así como una breve descripción de los casos de usos identificados.

Se describe también la representación arquitectónica del sistema, haciendo énfasis en el estilo y el patrón de arquitectura utilizado, así como en los patrones de diseño más importantes. Se muestran los diagramas de clases de los paquetes relevantes de cada subsistema, así como los diagramas de secuencias necesarios para comprender el funcionamiento del presente trabajo.

2.1. Modelo de Dominio

El modelo de dominio es una representación visual de los conceptos del mundo real significativos para un problema. El diagrama de clases del modelo de dominio de la presente investigación se muestra en la figura 2.1.

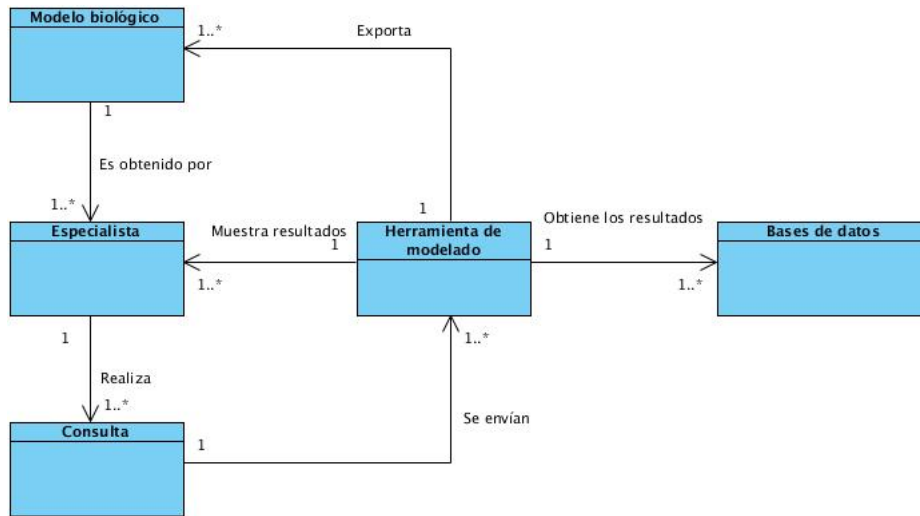


Figura 2.1: Diagrama de clases del modelo de dominio

Para una mejor comprensión, a continuación se describen cada una de las clases que intervienen en el diagrama mostrado:

- **Especialista:** representa la entidad que realiza las consultas, obtiene los resultados de la misma y además los nuevos modelos biológicos construidos.
- **Consulta:** representa la entidad que es realizada por un determinado *especialista* y se envía a la *herramienta de modelado*.
- **Herramienta de modelado:** respresenta la entidad encargada de procesar la *consulta*, mostrar los resultados de la misma y construir los *modelos biológicos*.
- **Bases de datos:** representa la entidad encargada de brindar la información necesaria para que la *herramienta de modelado* pueda procesar la *consulta*.
- **Modelo biológico:** es la entidad que representa el archivo SBML obtenido por el especialista con información biológica nueva o actualizada.

2.2. Breve descripción del sistema

El sistema a desarrollar permitirá el acceso a la información biológica desde múltiples bases de datos existentes. En esta aplicación se desarrollará un módulo para el SBW (System Biology Worbench) y extensión para el CellDesigner.

El módulo para el SBW será la aplicación que llevará a cabo las principales operaciones del sistema. Además, será el encargado de realizar y validar todas las consultas que se le realicen a las aplicaciones desarrolladas por la Web Semántica, las cuales integran un gran cúmulo de información biológica procedente de múltiples bases de datos. Con el resultado de las consultas ejecutadas por el módulo se obtienen algunos recursos RDF que son transformados al lenguaje SBML, que es utilizado por el CellDesigner y otras herramientas de modelado. El resultado de este proceso es integrado a un fichero SBML existente con el objetivo de adicionarle nueva información biológica obtenida por estas consultas.

La extensión para el CellDesigner será la interfaz con el módulo del SBW que permitirá el acceso a todas las funcionalidades que brinda el mismo. Esta relación entre los diferentes componentes del sistema ofrece gran escalabilidad al mismo, debido a que a puede ser utilizado por la herramienta CellDesigner y por todas aquellas que se encuentren integradas al SBW.

2.3. Especificación de los requisitos del sistema

2.3.1. Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, y se mantienen invariables sin importar con que propiedades o cualidades se relacionen. En el presente trabajo se identificaron los siguientes:

RF1. Ejecutar la consulta seleccionada.

RF1.1. Obtener los resultados de la consulta ejecutada.

RF2. Adicionar una nueva consulta al sistema.

RF2.1. Validar la nueva consulta.

RF3. Obtener los recursos RDF.

RF4. Salvar los recursos RDF.

RF5. Tranformar los recursos RDF que se adicionan al archivo SBML.

RF5.1. Abrir el archivo SBML.

2.3.2. Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener, y que harán del mismo un sistema confiable y seguro.

Transparencia

- El sistema ocultará la naturaleza de la conexión a la Base de Datos, permitiendo que los usuarios interactúen a través de una misma aplicación, la propia interfaz de la extensión.
- Los usuarios no tienen que encargarse de obtener las instancias del recurso en la consulta, solo debe especificar si lo desea obtener y el sistema se encargará de obtener estas instancias a las que se está consultando.

Portabilidad

- El sistema será multiplataforma, razón por la cual podrá ser utilizado en los sistemas operativos Windows, Linux y OS X.

Software

- Para el uso del sistema se deberá disponer como mínimo de la máquina virtual de Java 1.6 , debe estar instalado el SBW (*System Biology Workbench*) y la herramienta de modelado CellDesigner.

Hardware

- La PC debe estar conectada a Internet para lograr que el sistema que se va desarrollar funcione. Debe tener al menos 350 Mb disponibles en el disco para la instalación del CellDesigner y el SBW.

2.4. Definición de los casos de usos del sistema

2.4.1. Actores del sistema

Los actores representan a terceros fuera del sistema que interactúan con él. En el sistema que se describe se identificó el siguiente actor:

Actor	Descripción
Usuario	Representa al actor encargado de entrar y ejecutar las consultas del sistemas.

2.4.2. Listado de casos de usos del sistema

La forma en que los actores usan el sistema es representada a través de los casos de usos. Estos últimos son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del actor.

Los casos de usos identificados en la presente investigación son enunciados a continuación:

Orden	Nombre	Prioridad	Breve descripción
1	Ejecutar Consulta (ver Anexo A pág 59)	Crítico	El usuario inicia el caso de uso cuando decide ejecutar la consulta seleccionada. El sistema la ejecuta la consulta, mostrando los resultados de la misma.
2	Adicionar Consulta	Secundario	El usuario inicia el caso de uso cuando decide adicionar una nueva consulta al sistema. El sistema valida la misma y finalmente es adicionada a la lista de consultas.
3	Obtener RDF (ver Anexo A pág 60)	Crítico	El caso de uso se inicia cuando el usuario decide visualizar un recurso RDF de una consulta ejecutada.
4	Salvar RDF	Secundario	El usuario inicia el caso de uso cuando decide salvar un recurso RDF visualizado especificando la dirección del mismo. El sistema salva el recurso RDF en el fichero seleccionado.
5	Transformar RDF a SBML (ver Anexo A pág 61)	Crítico	El usuario inicia el caso de uso con la opción de convertir el RDF a SBML, luego debe seleccionar la opción de abrir el archivo SBML. Adiciona los recursos RDF a su lista de recursos y selecciona los campos del SBML que desea modificar. Finalmente, selecciona la opción de transformar el RDF. El sistema salva el SBML seleccionado con los nuevos recursos añadidos.

2.4.3. Diagrama de casos de usos del sistema

Los diagramas de casos de uso del sistema representan gráficamente a los procesos y su interacción con los actores.

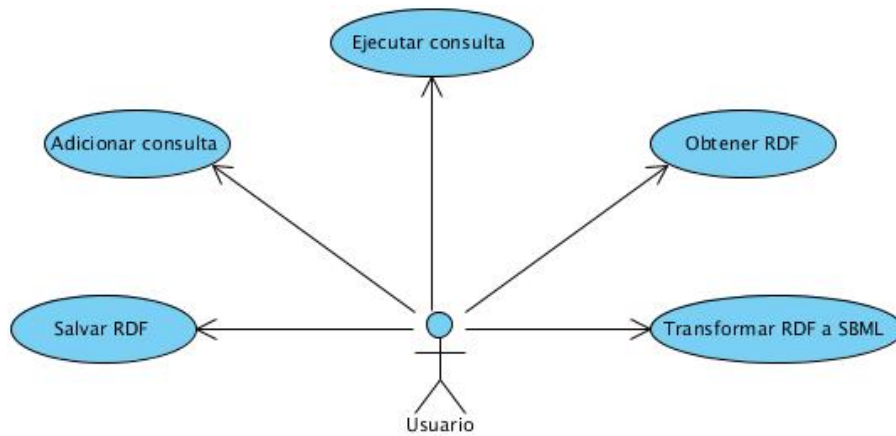


Figura 2.2: Diagrama de casos de usos del sistema.

Vista de casos de usos arquitectónicamente significativos

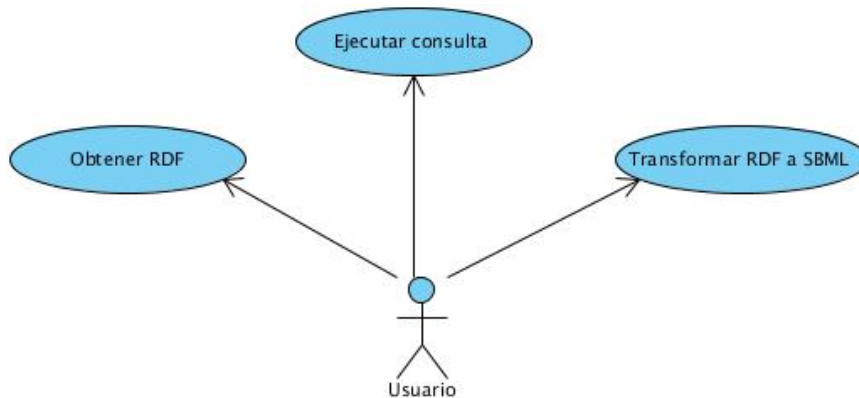


Figura 2.3: Diagrama de casos de usos arquitectónicamente significativos.

2.5. Vista lógica

En esta sección se describe las partes arquitectónicamente significativas del modelo de diseño, como son la descomposición en capas, subsistemas y paquetes.

La descripción de la vista lógica se realizará por módulos para un mejor entendimiento de las características de cada una de las partes. Se entiende por módulo cada uno de los subsistemas que componen el sistema. Se mostrará por cada módulo, los diagramas de paquetes de aquellos arquitectónicamente significativos, teniendo en cuenta su relación y composición. Se ilustrará además, los diagramas de clases del diseño de los paquetes que se consideren necesarios, acompañados con una breve explicación de algunas de las clases o interfaces que lo componen.

2.5.1. Subsistemas

El subsistema *extension* representa la extensión del CellDesigner que accede al subsistema *module* que es el módulo del SBW que ejecuta las principales funcionalidades del sistema.

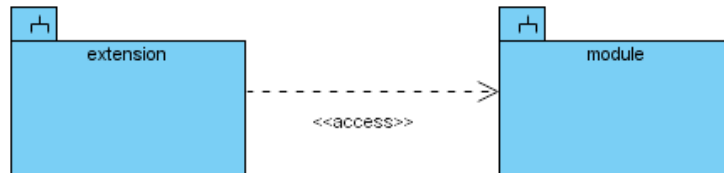
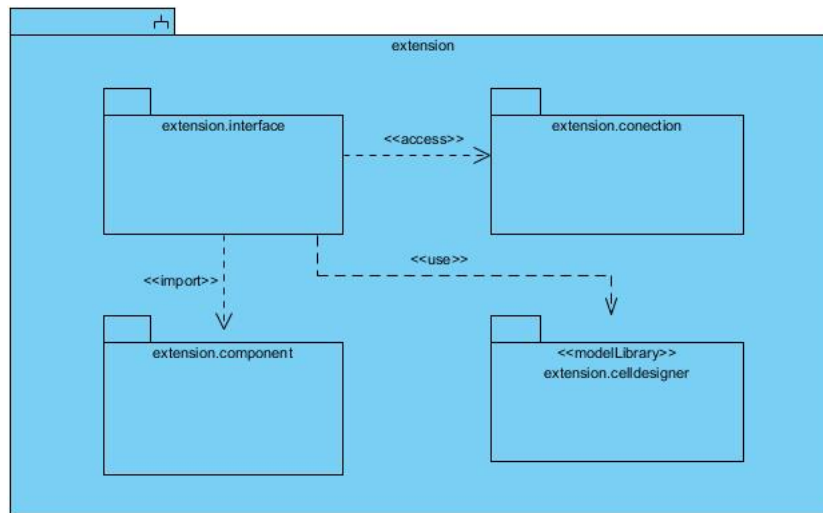


Figura 2.4: Vista general de los subsistemas de la aplicación.

2.5.1.1. Subsistema *extension*

El subsistema *extension* contiene todos los paquetes y relaciones necesarias para el diseño de la extensión del CellDesigner.

Figura 2.5: Paquetes relevantes del subsistema *extension*.

Descripción de los paquetes del subsistema *extension*

- **extension.interface:** este paquete contiene la interfaz gráfica de usuario principal de la extensión de CellDesigner .
- **extension.component:** contiene las clases que gestionan toda la información enviada por el módulo del SBW y que finalmente se le muestra a la interfaz gráfica de usuario principal.
- **extension.conection:** tiene la clase encargada de recibir y enviar las peticiones del usuario al módulo del SBW.
- **extension.celldesigner:** contiene las clases que añaden la extensión como una nueva funcionalidad de la herramienta CellDesigner.

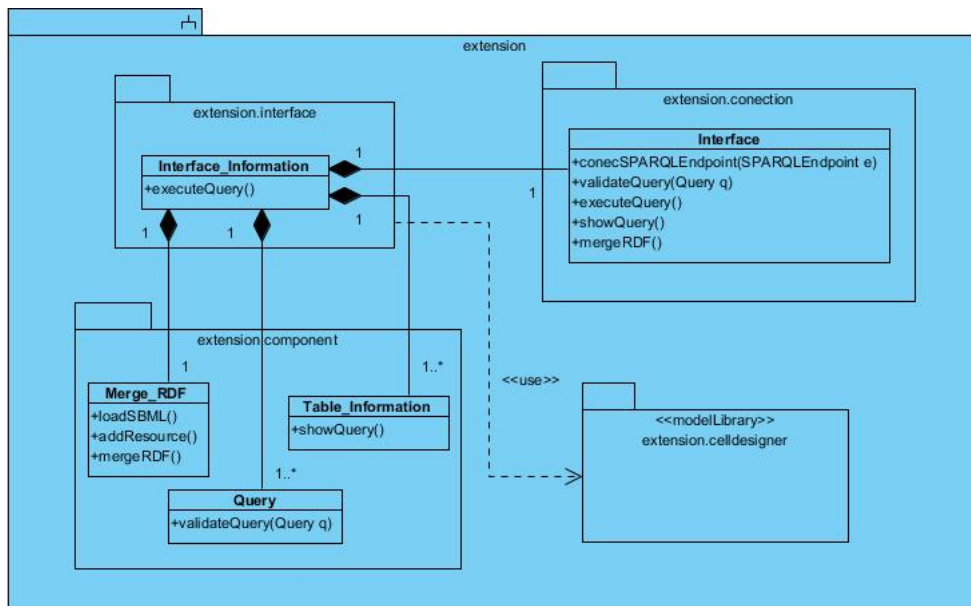


Figura 2.6: Diagrama de clases del diseño del subsistema *extension*.

Diagrama de clases del diseño del subsistema *extension*

- **Interface_Information:** es la clase principal de la extensión de CellDesigner que se encargada de recibir y mostrar todas las peticiones de los usuarios.
- **Merge_RDF:** se encarga de insertar los recursos RDF al archivo SBML. Permite realizar otras funciones, como cargar el archivo SBML que se desea transformar y adicionar nuevos recursos RDF al mismo.
- **Table_Information:** es la clase encargada de recibir y mostrar adecuadamente los datos de la consulta realizada por el usuario.
- **Query:** permite adicionar y validar consultas del sistema.
- **Interface:** es la encargada de comunicar la extensión de CellDesigner con el módulo de SBW. Esta recibe las peticiones de la Interface_Information y se las envía al módulo. Luego, el módulo envía la respuesta mediante esta clase al paquete de extension.component.

2.5.1.2. Subsistema *module*

El subsistema *module* contiene todos los paquetes y relaciones necesarias para el diseño del módulo del SBW que permite ejecutar las principales funcionalidades del sistema.

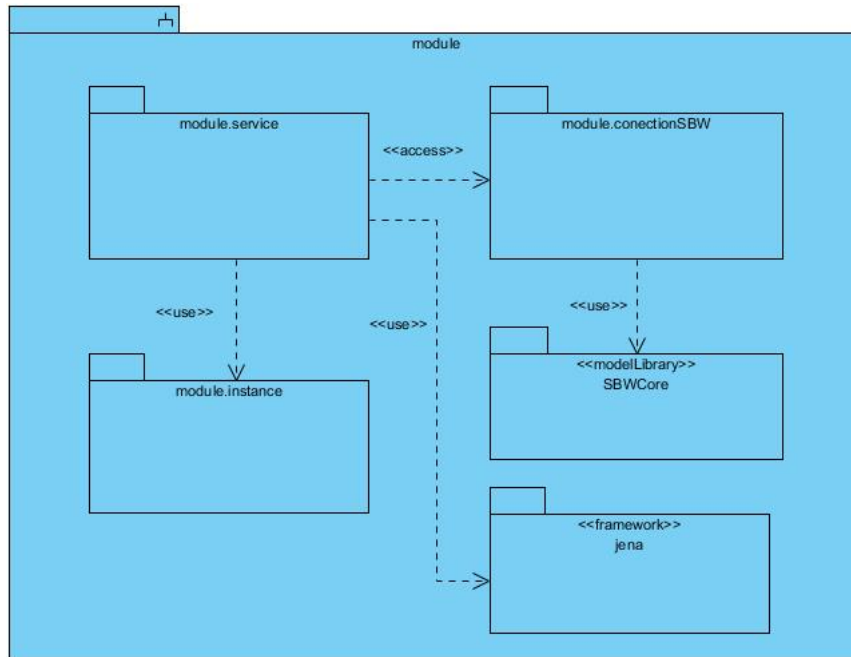


Figura 2.7: Paquetes relevantes del subsistema *module*.

Descripción de los paquetes del subsistema *module*

- **module.service:** es el paquete que contiene las clases que ejecutan las funcionalidades del sistema.
- **module.instance:** contiene las clases que constituyen tipos de datos importantes para el sistema, como las Query y los SPARQL Endpoint.
- **module.conectionSBW:** este paquete contiene la clase encargada de crear la instancia del módulo en el SBW.
- **jena:** es el framework encargado de conectar el módulo diseñado en el SBW a las aplicaciones desarrolladas por la Web Semántica.
- **SBWCore:** es la librería que permite adicionar el módulo diseñado a la plataforma del SBW.

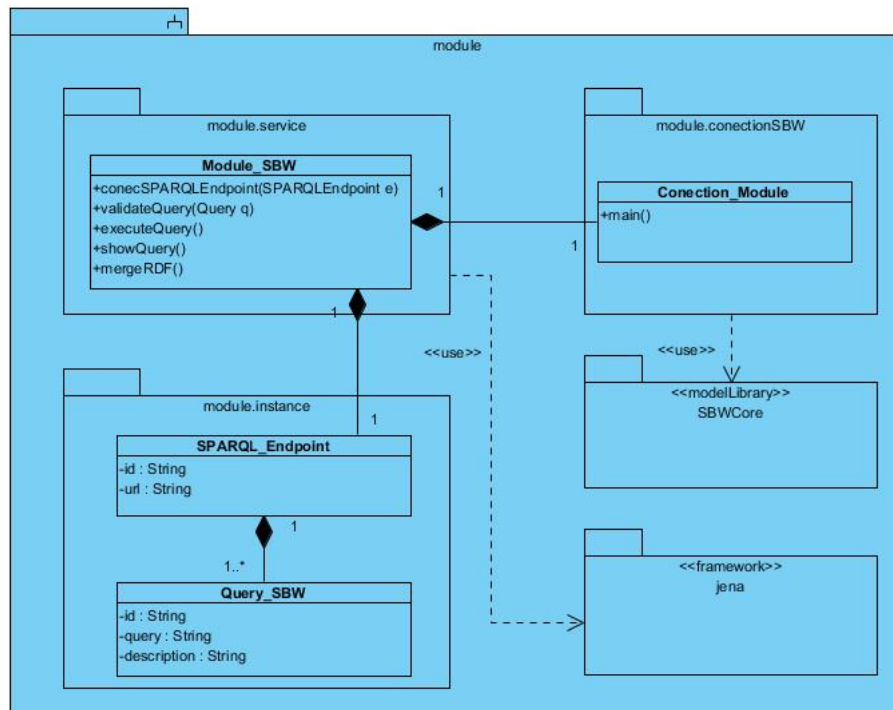


Figura 2.8: Diagrama de clases del diseño del subsistema *module*.

Diagrama de clases del diseño del subsistema *module*

- **Query_SBW:** es la clase encargada de crear el tipo de dato Query.
- **SPARQL_Endpoint:** es la clase encargada de crear el tipo de dato SPARQLEndpoint.
- **Module_SBW:** es la clase encargada de realizar todas las funcionalidades del sistema.
- **Conection_Module:** crea una instancia del módulo en el SBW.

2.6. Representación arquitectónica

La misión principal de la arquitectura del sistema es mostrar una panorámica general de los entes y subsistemas que lo integran, explicando cada uno de estos en diferentes vistas. El modelo de representación arquitectónica está basado en el modelo de las 4 + 1 vistas. La vista de procesos no se describirá porque no se considera relevante la información que esta brinda. La vista de casos de uso se describió en el capítulo anterior y la vista de implementación se encuentra descrita en el siguiente capítulo.

2.6.1. Estilo de Arquitectura

El estilo de arquitectura adoptado fue el de llamada y retorno [39] debido a que refleja la estructura del lenguaje de programación presente en el sistema. Permite que los diseñadores de la aplicación puedan construir una estructura relativamente fácil de modificar y ajustar a escala. Se basa en la bien conocida abstracción de procedimientos, funciones y métodos. Persigue la escalabilidad y modificabilidad del sistema.

En la arquitectura de llamada y retorno la comunicación y la coordinación entre los componentes se consiguen a través del paso de mensaje. La interfaz envía un mensaje al módulo y este envía uno o varios mensajes con la respuesta.

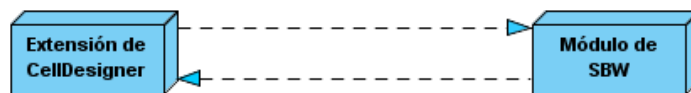


Figura 2.9: Estilo de arquitectura *llamada y retorno* aplicado al sistema.

Como se puede apreciar en la figura 3.1, la extensión de la herramienta CellDesigner envía solicitudes al módulo de SBW y este a su vez le envía mensajes con la respuesta.

2.6.2. Patrón de arquitectura

El patrón de arquitectura que sigue el sistema propuesto es *cliente - servidor*, debido a que este permite distribuir una aplicación entre dos o más componentes especializados cuya ejecución se distribuye entre uno o más equipos. Define dos tipos de entidades diferenciadas que se responsabilizan de acciones diferentes: cliente y servidor. Su modelo de interacción está basado en el concepto de servicio o implementado sobre un diálogo de petición - respuesta:

- Cliente: inicia el diálogo mediante el envío de peticiones.
- Servidor: presta el servicio y responde las peticiones recibidas.



Figura 2.10: Patrón de arquitectura *cliente - servidor* aplicado al sistema.

El cliente del sistema es la extensión de la herramienta CellDeisgner que realiza las peticiones al servidor que es el módulo de SBW mediante el paso de mensajes, a través de *sockets* TCP/IP.

2.6.3. Patrones de diseño utilizados

Un patrón es una solución de un problema en determinado contexto [40], donde un contexto es una situación en la que el patrón es aplicable y generalmente es recurrente. A continuación se describen los patrones de diseños más importantes aplicados en el desarrollo del sistema:

Chain of responsibility

- **Problema:** la clase *Interface_Information* de la extensión de CellDesigner no puede atender todas las funcionalidades del sistema.
- **Solución:** proporcionar a más de una clase la capacidad de atender una petición, para así evitar el acoplamiento con la clase que hace la petición. Se forma con estos objetos una cadena, en la cual cada objeto satisface la petición o la pasa al siguiente. La clase *Interface_Information* debe desplegar esas peticiones hacia la clase *Interface* que se comunica con el módulo del SBW mediante su clase principal *Module_SBW* para darle respuesta a las solicitudes enviadas.

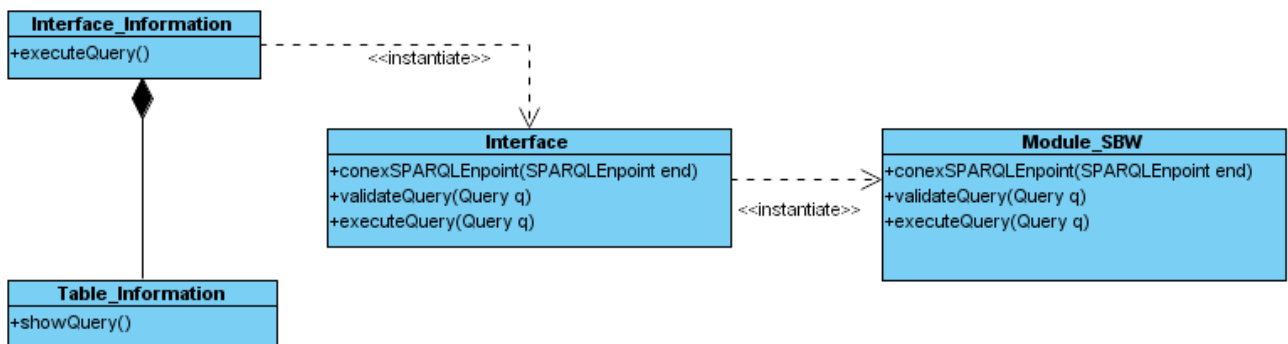


Figura 2.11: Ejemplo de la aplicación del patrón Chain of responsibility en el sistema.

Composite

- **Problema:** los usuarios acceden a todas las funcionalidades del sistema, desde la clase principal *Interface_Information* de la extensión de CellDesigner.
- **Solución:** permitir usar adecuadamente la composición para tratar uniformemente las peticiones al sistema. Ocultando al usuario los diferentes objetos que ejecutan las peticiones. *Interface_Information* realiza las solicitudes mediante los objetos *Merge_RDF*, *Query* y *Table_Information*.

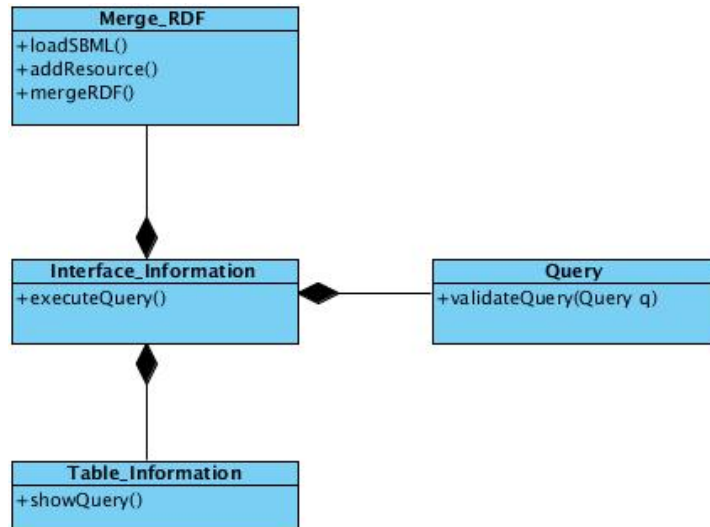


Figura 2.12: Ejemplo de la aplicación del patrón Composite en el sistema.

Singleton

- Problema:** en el sistema se necesitaba que existiese una sola instancia de la clase *Interface_Information* y que esta tuviese un único punto de acceso, de manera que sólo exista una interfaz para la extensión del CellDesigner.
- Solución:** se diseña la clase *Interface_Information* con una referencia a un objeto de ella misma y un método estático que devuelve dicho objeto.

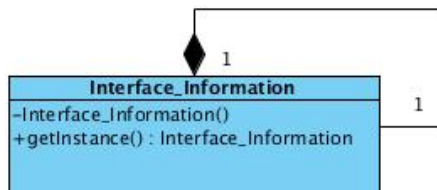


Figura 2.13: Ejemplo de la aplicación del patrón Singleton en el sistema.

2.7. Diagramas de secuencia

Un diagrama de secuencia muestra las interacciones entre objetos ordenadas temporalmente. Ilustra los objetos que se encuentran en un escenario, y la secuencia de mensajes intercambiados entre ellos para llevar a cabo la funcionalidad descrita.

2.7.1. CU Ejecutar Consulta

Es relevante destacar que la clase Interface es el mediador de peticiones entre la interfaz y el módulo. La extensión cuando envía una solicitud al módulo crea una instancia de ese módulo que está corriendo en el SBW. Esa instancia permite una comunicación directa entre la extensión y el módulo, cuando se termina la ejecución de alguna de las funcionalidades del sistema el módulo finaliza la ejecución.

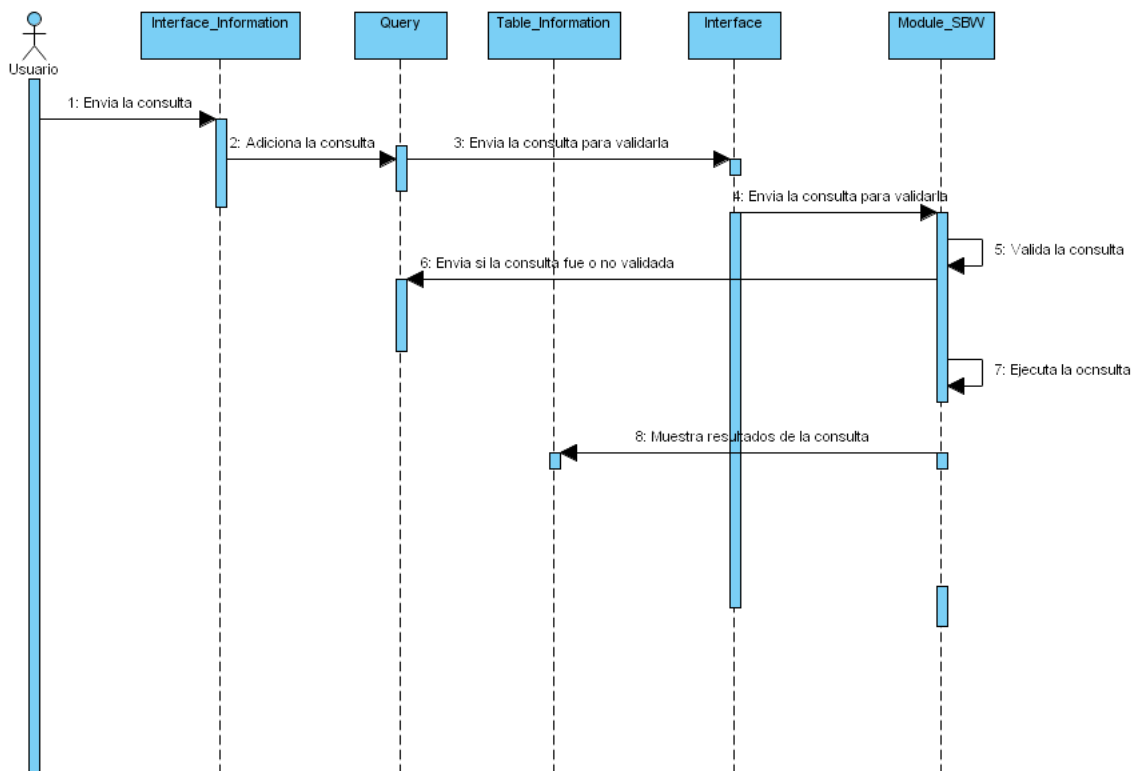


Figura 2.14: Diagrama de secuencia del CU *Ejecutar Consulta*.

2.7.2. CU Mezclar RDF

El diagrama muestra la mezcla de nuevos recursos RDF a un archivo SBML. Esto permite que el usuario a partir de los recursos RDF obtenidos en las consultas que ha ejecutado pueda añadirlos a un SBML cualquiera. Este nuevo archivo SBML que genera el sistema, puede ser cargado en cualquier herramienta de modelado. Estas herramientas podrán tener acceso a esa nueva información que ha sido añadida por los nuevos recursos RDF.

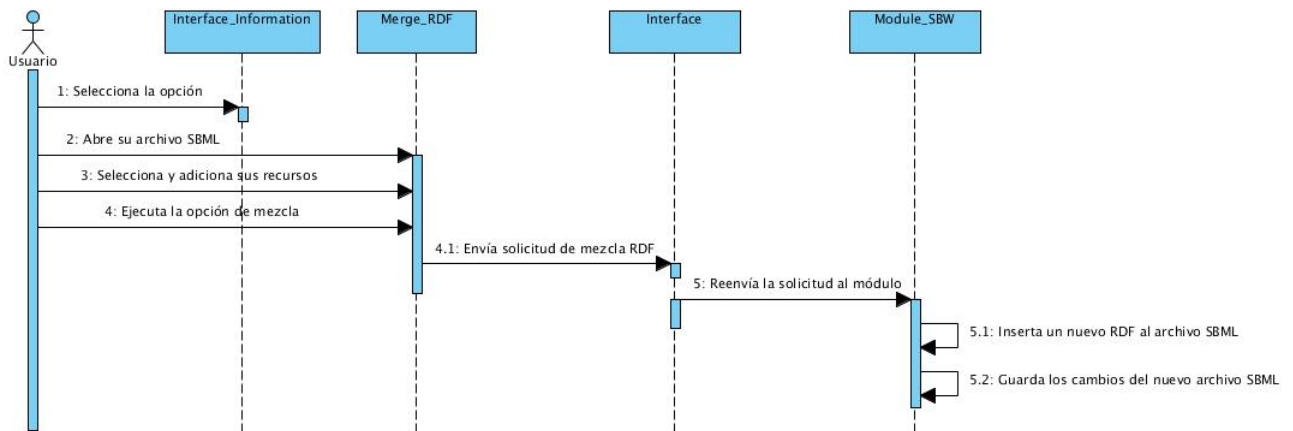


Figura 2.15: Diagrama de secuencia del CU *Transformar RDF a SBML*.

2.8. Vista de despliegue

Esta vista constituye la visión física del sistema, representa un grafo de nodos unidos por conexiones de comunicación.

2.8.1. Diagrama de despliegue

En el siguiente diagrama se muestra el despliegue de los nodos del sistema y la comunicación entre cada uno de ellos.

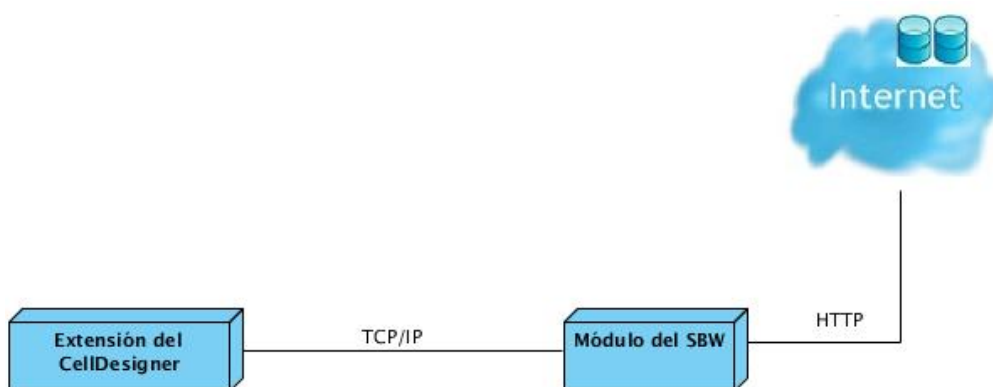


Figura 2.16: Diagrama de despliegue del sistema.

Descripción de los nodos

- **Nodo Extensión de CellDesigner:** es el nodo donde se despliega la extensión de CellDesigner.

- **Nodo Módulo de SBW:** es el nodo donde se despliega el módulo de SBW.

Descripción de la comunicación

- **TCP/IP:** este protocolo será utilizado para el intercambio de mensajes entre la extensión y el módulo.
- **HTTP:** este protocolo será utilizado por el módulo para acceder a las aplicaciones de la Web Semántica en Internet.

2.9. Conclusiones.

Como resultado de este capítulo se identificaron los requerimientos funcionales, los actores y casos de usos del sistema que sirvieron de base para realizar el diseño del producto que se presenta en este trabajo.

Se obtuvo además la arquitectura del sistema sustentada en el estilo *llamada y retorno*. Se elaboró el diseño del sistema desde la estructura de paquetes hasta los diagramas de clases por cada paquete, mostrando solamente los más significativos. También se muestra la vista de despliegue que permite mostrar la distribución física del sistema.

Capítulo 3

Implementación del Sistema

En este capítulo se describe la implementación del sistema en términos de componentes y la manera en que estos componentes serán desplegados. Además, se explica y se muestra el pseudocódigo de los principales métodos implementados en el sistema, así como los resultados de las pruebas realizadas.

3.1. Diagrama de componentes

El diagrama de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean estos de código fuente, binarios, archivos, bibliotecas cargadas dinámicamente o ejecutables.

En la figura 3.1 se muestran los componentes relevantes del sistema y la interfaz que este implementa. El componente fundamental de este sistema es el *Module_SBW*, el mismo ejecuta las principales funcionalidades del sistema e incluye otro componente secundario como es el caso de las librerías *jena.jar*.

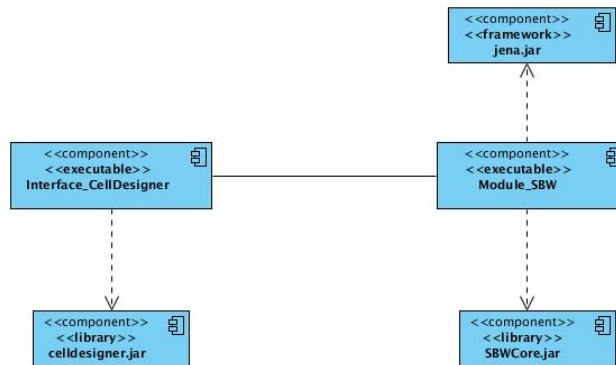


Figura 3.1: Diagrama de componentes del sistema.

El *Module_SBW* además se encuentra conectado a Internet permitiéndole tener acceso a la información biológica necesaria para el funcionamiento del sistema a implementar.

La interfaz *Interface_CellDesigner* permite que el usuario se relacione con el componente *Module_SBW* y ejecute las funcionalidades del sistema, mediante esta interfaz de usuario.

El *componente jena.jar* es un *framework* permite que el *Module_SBW* se conecte a la Web Semántica, específicamente a todas las bases de datos biológicas que se encuentran integradas mediante aplicaciones semánticas en Internet.

El *celldesigner.jar* sería la librería encargada de insertar a la interfaz *Interface_CellDesigner* como una extensión más de la herramienta CellDesigner.

SBWCore.jar es la librería que adiciona al componente *Module_SBW* como un nuevo módulo de la plataforma del SBW.

3.2. Diagrama de despliegue de los componentes

En este diagrama se muestra la distribución de los componentes por los distintos nodos del despliegue, mostrados en la figura 3.1.

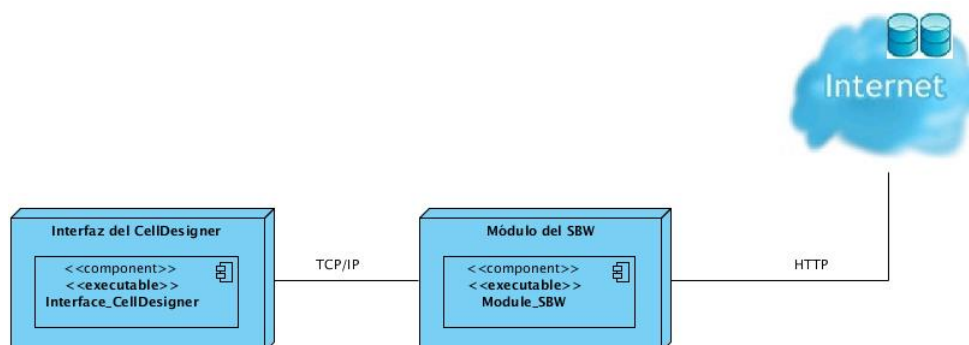


Figura 3.2: Diagrama de despliegue de los componentes.

3.3. Descripción de los algoritmos más importantes

En esta sección se describen los algoritmos más importantes implementados en el sistema, para satisfacer las principales funcionalidades del presente trabajo.

3.3.1. Módulo del SBW

Algoritmo para ejecutar una consulta

El algoritmo es implementado por la función *executeQuery()* de la clase *Engine*, que es invocado cuando los usuarios interactúan con la interfaz del CellDesigner y deciden ejecutar una consulta del sistema. Los parámetros especificados son: la dirección del SPARQL Endpoint y la consulta que se le realiza al mismo. El algoritmo desarrollado extrae las variables de la consulta (línea 4). Establece la conexión con el SPARQL Endpoint especificado (línea 5). Posteriormente se transforma la consulta de entrada con el *framework* Jena (línea 6) y luego se inicia el servicio de consulta de dicho *framework* (línea 7), obteniendo las soluciones de la misma (línea 8). El resultado obtenido se le devuelve al usuario para cada una de las variables (líneas de la 9 a la 20), finalmente se cierra el servicio de consulta del *framework* (línea 22).

Algoritmo 3.1 *executeQuery()*

```

1 Entrada: la URI del SPARQL Endpoint, la consulta
2 Salida: información de la consulta
3
4 V := cargar variables de la consulta
5 URL_q := establece la conexión mediante Jena de la URI especificada como entrada
6 Q_Jena := genera la consulta [Q] mediante el Jena
7 Q_Exec := comienza el servicio de consulta generado por Jena [Q_Jena] al SPARQL Endpoint [URL_q]
8 Result := soluciones obtenido [Q_Exec]
9     mientras Result tenga una próxima solución
10         C := cadena
11         S := [Result] obtiene la próxima solución
12         para i := 1 hasta [V] hacer
13             si S.V[i] <> null entonces
14                 C := concatenar C con la solución S de la variable V[i]
15             si no
16                 C := concatenar C con espacio vacío
17             fin si
18         fin para
19         I adiciona la [C]
20     repetir
21
22 Q_Exec se cierra el servicio de consulta al finalizar

```

Algoritmo para insertar nuevos recursos RDF a un archivo SBML existente

Este algoritmo es implementado por la función *addResource()* de la clase *Engine*, que es invocado cuando los usuarios del CellDesigner insertan los nuevos recursos RDF al archivo SBML existente. Los parámetros especificados son: el archivo SBML existente, la lista de recursos RDF seleccionada por el usuario y el nodo del archivo SBML que se va a modificar con esta nueva información. Permite a partir de los nuevos recursos RDF insertar nueva información biológica en el nodo del archivo SBML seleccionado (líneas de la 6 a la 14) .

Algoritmo 3.2 *addResource()*

```

1 Entrada: archivo SBML, lista de recursos RDF, nodo del archivo SBML
2
3 F_SBML := archivo SBML de entrada
4 Nod_SBML := nodo del archivo SBML de entrada
5
6 para i := 1 hasta n hacer
7
8     Element := se crea un nuevo elemento [F_SBML].("rdf:li")
9     Att := se crea un nuevo atributo [F_SBML].("rdf:resource")
10    se adicionan los recursos RDF al nodo de este atributo.(i)
11    se adiciona el atributo al elemento Element.[Att]
12    se adiciona el elemento al nodo Nod_SBML.[Element]
13
14 fin para

```

3.4. Pruebas al sistema

Las pruebas de software es 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.

3.4.1. 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. A continuación se especifican los escenarios de prueba de cada uno de los casos de uso.

Escenarios de prueba para el caso de uso *Adicionar Consulta*

El escenario de prueba del caso de uso *Adicionar Consulta* tiene como objetivo adicionar una nueva consulta a la extensión de CellDesigner. La consulta adicionada se valida mediante las variables: *SPARQL Endpoint* y *Query*.

La primera de ellas, especifica a la aplicación semántica que se va a consultar y no es vulnerable a errores porque es un campo de selección para el usuario. Mientras que la segunda es validada por el sistema

para el caso que esté vacío el campo de entrada de la variable o la consulta contenga errores. La variable *Description* no es vulnerable a resultados inválidos porque su dato es un texto y además puede tener valor nulo porque el usuario no está obligado a describir la consulta adicionada.

No.	Nombre de la variable	Clasificación	Valor nulo	Descripción
1	SPARQL Enpoint	Campo de selección	No	El dato es una URI
2	Description	Campo de texto	Si	El dato es texto
3	Query	Campo de texto	No	El dato es una consulta SPARQL

(a) Descripción de las variables.

Escenario	Descripción	Variables de entrada			Respuesta del sistema	Flujo Central
		SPARQL Enpoint	Description	Query		
EC 1.1. Adicionar correctamente los datos de la consulta	Adicionar nueva consulta al sistema	Válido	Válido	Válido	Se le adiciona la nueva consulta al sistema	1.Opción “Add Query” 2.Adicionar los nuevos datos de la consulta en la opción “Add”
EC 1.2. Adicionar correctamente los datos de la consulta	Adicionar nueva consulta al sistema	Válido	Válido	Válido	El sistema emite un mensaje de error “ <i>null</i> ” (no esperado)	1.Opción “Add Query” 2.Adicionar los nuevos datos de la consulta en la opción “Add”

EC 1.3. Dejar campos requeridos en blanco	Adicionar nueva consulta, dejando campos requeridos en blanco	Válido	Válido	Inválido	El sistema emite un mensaje de error	1.Opción “Add Query” 2. Algunos campos requeridos quedan en blanco cuando se selecciona la opción “Add”
EC 1.4. Introducir datos incorrectos	Adicionar nueva consulta e introducir datos incorrectos	Válido	Válido	Inválido	El sistema emite un mensaje de error	1.Opción “Add Query” 2. Se introducen algunos datos incorrectos cuando se selecciona la opción “Add”

Tabla 3.1: Caso de uso *Adicionar Consulta*.

Los resultados de la prueba realizada arrojó la no conformidad de que no se verificaba si el resultado de la consulta era nulo (ver la tabla 3.6), siendo detectada en la etapa de prueba y solucionada mediante la incorporación de una nueva condición en el algoritmo implementado en el caso de uso.

Escenarios de prueba para el caso de uso *Transformar RDF a SBML*

El escenario de prueba del caso de uso *Transformar RDF a SBML* tiene como objetivo adicionarle nuevos recursos RDF a un archivo SBML existente. La variable *Resource* no es vulnerable a errores tipográficos porque es añadida por el usuario mediante la selección de texto ya existente en la misma interfaz, pero puede pasar que el usuario no adicione los recursos RDF y el sistema emite un mensaje de error. El *MetaID*

es la variable encargada de actualizar cada uno de los metaid¹ del archivo SBML importado por el usuario y que permite ser seleccionado para determinar a cual se le adiciona los recursos RDF.

No.	Nombre de la variable	Clasificación	Valor nulo	Descripción
1	Resource	Campo de selección	No	El dato es texto
2	MetaID	Campo de selección	No	El dato es texto

(a) Descripción de las variables.

Escenario	Descripción	Variables de entrada		Respuesta del sistema	Flujo Central
		Resource	MetaID		
EC 1.1. Insertar nuevos recursos RDF a un archivo SBML	Inserta recursos RDF al archivo SBML	Válido	Válido	Se inserta el nuevo recurso RDF al archivo SBML	1. Selecciona la opción “RDF to SBML” 2. Luego la opción “Open SBML” actualiza los campos MetaID y se adicionan los recursos RDF 3. Finalmente la opción “Send” y luego el usuario decide sobrescribir el archivo SBML modificado
EC 1.2. Dejar campos requeridos en blanco	Insertar recursos RDF al archivo SBML, dejando campos requeridos en blanco	Inválido	Válido	El sistema emite un mensaje de error	1. Selecciona la opción “RDF to SBML” 2. Luego la opción “Open SBML” actualiza los campos MetaID y “no” se adicionan los recursos RDF 3. La opción “Send”

¹Son los identificadores de cada una de las especies o reacciones presentes en el modelo biológico del archivo SBML.

EC 1.3. Dejar campos requeri- dos en blanco	Insertar recursos RDF al archivo SBML, dejando campos requeridos en blanco	Válido	Inválido	El sistema no emite un mensaje de error	1. Selecciona la opción “RDF to SBML” 2. Luego la opción “Open SBML” no se actualizan los campos MetaID
--	---	--------	----------	--	---

Tabla 3.2: Caso de uso *Transformar RDF a SBML*.

Los resultados de la prueba realizada arrojó la no conformidad de que no se verificaba que el archivo importado fuese de tipo SBML (ver la tabla 3.6), siendo detectada en la etapa de prueba y solucionada mediante la validación de los archivos SBML en uno de los algoritmos implementados en el caso de uso.

Escenario de prueba para el caso de uso *Obtener RDF*

El escenario de prueba del caso de uso *Obtener RDF* tiene como objetivo obtener el recurso RDF seleccionado por el usuario. La variable *Resource URI* es el identificador del recurso RDF que permite visualizar al mismo, que es seleccionado por el usuario mediante un doble clic.

No.	Nombre de la variable	Clasificación	Valor nulo	Descripción
1	Resource URI	Campo de selección	No	El dato es una URI

(a) Descripción de la variable.

Escenario	Descripción	Variable de entrada		
		Resource URI	Respuesta del sistema	Flujo Central
EC 1.1. Obtener el recurso RDF seleccionado	Seleccionar el recurso RDF que se desea mostrar	Válido	Se muestra el recurso RDF	<ol style="list-style-type: none"> 1. Seleccionar el recurso RDF que se muestra en el resultado de la consulta con “doble clic” 2. Se muestra el recurso RDF al usuario
EC 1.1. Obtener el recurso RDF seleccionado	Seleccionar el recurso RDF que se desea mostrar	Inválido	El sistema no emite un mensaje de error	<ol style="list-style-type: none"> 1. Seleccionar el recurso RDF que se muestra en el resultado de la consulta con “doble clic” 2. No se muestra el recurso RDF al usuario

Tabla 3.3: Caso de uso *Obtener RDF*.

Los resultados de la prueba realizada arrojó la no conformidad de que no se verificaba si la selección del usuario fuese un recurso RDF (ver la tabla 3.6), siendo detectada en la etapa de prueba y solucionada mediante la validación de los recursos RDF en el algoritmo implementado en el caso de uso.

Escenario de prueba para el caso de uso *Salvar RDF*

El escenario de prueba del caso de uso *Salvar RDF* tiene como objetivo salvar el recurso RDF seleccionado por el usuario. La variable *RDF file* es el recurso RDF obtenido a partir del caso de uso *Obtener RDF* y que desea ser guardado por el usuario.

No.	Nombre de la variable	Clasificación	Valor nulo	Descripción
1	RDF file	Campo de tipo archivo	No	El dato es un archivo RDF

(a) Descripción de la variable.

Escenario	Descripción	Variable de entrada		
		RDF file	Respuesta del sistema	Flujo Central
EC 1.1. Salvar el recurso RDF	Salvar el archivo RDF seleccionado por el usuario	Válido	Mostrar una ventana para seleccionar la ubicación del archivo RDF que se desea salvar	<ol style="list-style-type: none"> 1. Selecciona la opción "Save RDF" 2. El usuario selecciona la ubicación para guardar el archivo RDF 3. El usuario acepta

Tabla 3.4: Caso de uso *Salvar RDF*.

Los resultados de la prueba realizada no arrojó ninguna no conformidad, todas las respuestas del sistemas fueran las esperadas.

Escenario de prueba para el caso de uso *Ejecutar Consulta*

El escenario de prueba del caso de uso *Ejecutar Consulta* tiene como objetivo ejecutar la consulta seleccionada por el usuario. La variable Query es la consulta que el usuario desea ejecutar, obteniendo finalmente el resultado deseado.

No.	Nombre de la variable	Clasificación	Valor nulo	Descripción
1	Query	Campo de texto	No	El dato es una consulta SPARQL

(a) Descripción de la variable.

Escenario	Descripción	Variable de entrada		Flujo Central
		Query	Respuesta del sistema	
EC 1.1. Ejecutar consulta	Se ejecuta la consulta seleccionada	Válido	Muestra el resultado de la consulta ejecutada	1. Selecciona la opción "Execute"

Tabla 3.5: Caso de uso *Ejecutar consulta*.

Los resultados de la prueba realizada no arrojó ninguna no conformidad, todas las respuestas del sistemas fueran las esperadas.

3.4.2. No Conformidades detectadas

Las no conformidades que fueron detectadas en los distintos escenarios de pruebas quedaron resueltas para lograr el correcto funcionamiento del sistema.

Elemento	No.	No Conformidad	Aspecto correspondiente	Etapa de detección	Signif.	No Signif.
Interfaz	1	No se verificaba que el archivo importado fuese de tipo SBML	Cuando se cargaba un archivo SBML para insertarle los recursos RDF	Etapa de Pruebas a la extensión de CellDesigner	X	
Interfaz	2	No se verificaba si el resultado de la consulta era nulo	Cuando se deseaba adicionar una nueva consulta al sistema	Etapa de Pruebas a la extensión de CellDesigner	X	
Interfaz	3	No se verificaba si la selección del usuario fuese un recurso RDF	Cuando se deseaba visualizar un recurso RDF	Etapa de Pruebas a la extensión de CellDesigner		X

Tabla 3.6: No Conformidades detectadas.

3.5. Conclusiones

Como resultado de este capítulo se obtuvo el diagrama de componentes y el de despliegue de los componentes del sistema, se realizó una breve explicación de los métodos más importantes que se implementaron. Se realizaron las pruebas de caja negra que permitieron detectar, documentar y solucionar algunos errores existentes en la aplicación implementada, arrojando tres no conformidades a las que se le dieron solución, quedando el sistema en correcto funcionamiento.

Conclusiones

1. En la investigación realizada se obtuvieron los conocimientos necesarios para la selección de la herramienta de modelado CellDesigner .
2. Con el análisis y diseño de la extensión de CellDesigner se logró implementar todas las funcionalidades del sistema.
3. El sistema implementado ofrece la posibilidad de acceder a aplicaciones de la Web Semántica desde la herramienta de modelado CellDesigner.
4. Las principales funcionalidades del sistema se implementaron en el SBW con la finalidad de brindarle una mejor escalabilidad al sistema desarrollado. Esto le permite a otras herramientas vinculadas al SBW, reutilizar las funcionalidades del módulo implementando una nueva interfaz.
5. Se realizó la evaluación de las principales funcionalidades del sistema haciendo uso de las pruebas de Caja Negra con la realización de las mismas, se logró comprobar la correcta implementación de las funcionalidades.

Recomendaciones

1. Incorporar un mecanismo de análisis a las consultas SPARQL que realiza la aplicación implementada, que permita generarlas automáticamente. Esto permite que los usuarios finales que utilizan estas herramientas no tengan que utilizar el lenguaje de consulta SPARQL, debido a que algunos especialistas desconocen dicho lenguaje.
2. Implementar una nueva funcionalidad en la extensión del CellDesigner, que permita “identificar” la información biológica añadida mediante recursos RDF para que el usuario final sepa que se han añadido por un proceso automático y que podría contener errores.

Referencias bibliográficas

- [1] Biología de Sistemas, Informe de Vigilancia Tecnológica. 2007;.
- [2] Lee DY, Yun C, Cho A, Hou BK, Park S, Lee SY. WebCell: a web-based environment for kinetic modeling and dynamic simulation of cellular networks. 2006 Sept;22. Bioinformatics Applications.
- [3] MODELING AND SIMULATION OF CELLULAR NETWORK; [cited 2012 Feb 5]. Available from: <http://webcell.kaist.ac.kr/>.
- [4] Virtual Cell Modeling and Analysis Software; [cited 2012 Feb 6]. Available from: <http://www.nrcam.uchc.edu/>.
- [5] SCHAFF J, LOEW LM. The Virtual Cell;4. Pacific Symposium on Biocomputing.
- [6] JDesigner: A Biochemical Network Layout Tool; [cited 2012 Feb 6]. Available from: <http://www.sys-bio.org/software/jdesigner.htm>.
- [7] Mandel JJ, FuB H, Palfreyman NM, Dubitzky W. Modeling biochemical transformation processes and information processing with Narrator;BMC Bioinformatics.
- [8] Dhar P, Chee Meng T, Somani S, Ye L, Sairam A, Chitre M, et al. Cellware: a multi-algorithmic software for computational systems biology. 2004 Aug;20. Bioinformatics Applications.
- [9] CellDesigner: A modeling tool of biochemical networks; [cited 2012 Jan 25]. Available from: <http://www.celldesigner.org/>.
- [10] Funahashi A, Morohashi M, Hiroaki K. CellDesigner: a process diagram editor for gene-regulatory and biochemical networks;.
- [11] Systems Biology Markup Language; [cited 2012 Jan 25]. Available from: <http://www.sbml.org/>.

- [12] The CellML project; [cited 2012 Jan 15]. Available from: <http://www.cellml.org/>.
- [13] Guía Breve de Web Semántica; [cited 2011 Dec 3]. Available from: <http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica>.
- [14] Guía Breve de Linked Data; [cited 2011 Dec 3]. Available from: <http://www.w3c.es/divulgacion/guiasbreves/LinkedData>.
- [15] RDF; [cited 2012 Mar 5]. Available from: <http://www.w3.org/RDF/>.
- [16] OWL Web Ontology Language Overview; [cited 2012 Mar 5]. Available from: <http://www.w3.org/TR/owl-features/>.
- [17] Systems Biology Workbench (SBW); [cited 2012 Feb 10]. Available from: <http://sbw.sourceforge.net/>.
- [18] A Visual Notation for Network Diagrams in Biology; [cited 2012 Apr]. Available from: http://www.sbgm.org/Main_Page/.
- [19] The SBML ODE Solver Library; [cited 2012 Jan 16]. Available from: <http://www.tbi.univie.ac.at/~raim/odeSolver/>.
- [20] libSBML; [cited 2012 May 11]. Available from: <http://sbml.org/Software/libSBML/>.
- [21] PANTHER, Classification System; [cited 2012 Feb 15]. Available from: <http://www.pantherdb.org/pathway/>.
- [22] MetaCyc, A member of the BioCyc database collection; [cited 2012 Feb 15]. Available from: <http://metacyc.org/>.
- [23] KEGG: Kyoto Encyclopedia of Genes and Genomes; [cited 2012 Feb 15]. Available from: <http://www.genome.jp/kegg/>.
- [24] Wilkinson S. Sentero: A Tool for the Analysis of Biochemical Networks;
- [25] Sauro HM. JARNAC: a system for interactive metabolic analysis;
- [26] Linked Data - Connect Distributed Data across the Web; [cited 2012 Jan 15]. Available from: <http://linkeddata.org/>.

- [27] User Guide for Sesame 2.3; [cited 2012 Mar 10]. Available from: <http://www.openrdf.org/doc/sesame2/2.3.2/users/index.html>.
- [28] Jena, A Semantic Web Framework for Java; [cited 2012 Mar 10]. Available from: <http://jena.sourceforge.net/>.
- [29] SparqlEndpoints; [cited 2012 March 2]. Available from: <http://www.w3.org/wiki/SparqlEndpoints>.
- [30] Entrez, The Life Sciences Search Engine; [cited 2012 Feb 16]. Available from: <http://www.ncbi.nlm.nih.gov/sites/gquery>.
- [31] DBGET Search; [cited 2012 Feb 16]. Available from: <http://www.genome.jp/dbget/>.
- [32] Belleau F, Nolin MA, Tourigny N, Rigault P, Morissette J. Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. 2008 March;41. Journal of Biomedical Informatics.
- [33] Bio2RDF.org, Semantic web atlas of postgenomic knowledge; [cited 2012 Feb 17]. Available from: <http://bio2rdf.org/>.
- [34] Unified Modeling Language;. Available from: <http://www.uml.org/> [cited 2009 Feb 20].
- [35] James Rumbaugh IJ, Booch G. El Lenguaje Unificado de Modelado. Manual de Referencia. Segunda Edición. Addison Wesley; 2007.
- [36] Armstrong E. HotSpot: A new breed of virtual machine, Javaworld;. Available from: <http://www.javaworld.com/jw-03-1998/jw-03-hotspot.html> [cited 2009 Feb 15].
- [37] J M Bull LP L A Smith, Freeman R. Benchmarking Java against C and Fortran for scientific applications. Journal of the ACM. 2001;.
- [38] Java SE Security; 2009 Feb 10. Available from: <http://java.sun.com/security/>.
- [39] Oaks S. Java Security (2nd Edition). O'Reilly Media, Inc.; 2001.
- [40] Zukowski J. Programación Java 2 J2SE 1.4. vol. 1. SYBEX, Inc.; 2003.

Bibliografía

- [1] Ayuda Extendida OpenUp (2008).
- [2] CellDesigner: A modeling tool of biochemical networks; [cited 2010 Jan 25]. Available from: <http://www.celldesigner.org/>.
- [3] Funahashi A, Morohashi M, Hiroaki K. CellDesigner: a process diagram editor for gene-regulatory and biochemical networks;.
- [4] James Rumbaugh IJ, Booch G. El Proceso Unificado de Desarrollo de Software. Addison Wesley; 1999.
- [5] Extreme Programming: A gentle introduction;. Available from: <http://www.extremeprogramming.org/> [updated 2009 Sep 28; cited 2012 Feb 15].
- [6] Eclipse Process Framework Project;. Available from: <http://www.eclipse.org/epf/> [cited 2012 Feb 20].
- [7] Unified Modeling Language;. Available from: <http://www.uml.org/> [cited 2012 Feb 20].
- [8] James Rumbaugh IJ, Booch G. El Lenguaje Unificado de Modelado. Manual de Referencia. Segunda Edición. Addison Wesley; 2007.
- [9] Systems Biology Markup Language; [cited 2012 Jan 25]. Available from: <http://www.sbml.org/>.
- [10] Pablo Castells. La web semántica; Universidad Autónoma de Madrid.
- [11] Guía Breve de Web Semántica; [cited 2011 Dec 3]. Available from: <http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica>.
- [12] RDF; [cited 2012 Mar 5]. Available from: <http://www.w3.org/RDF/>.
- [13] Systems Biology Workbench (SBW); [cited 2012 Feb 10]. Available from: <http://sbw.sourceforge.net/>

- [14] Jena, A Semantic Web Framework for Java; [cited 2012 Mar 10]. Available from: <http://jena.sourceforge.net/>.
- [15] SparqlEndpoints; [cited 2012 Mar 2]. Available from: <http://www.w3.org/wiki/SparqlEndpoints>.
- [16] Belleau F, Nolin MA, Tourigny N, Rigault P, Morissette T Jean. Bio2RDF: Towards a mashup to build bioinformatics knowledge systems; *Journal of Biomedical Informatics*.

Apéndice A

Descripción de Casos de Usos Críticos

Caso de uso Ejecutar Consulta

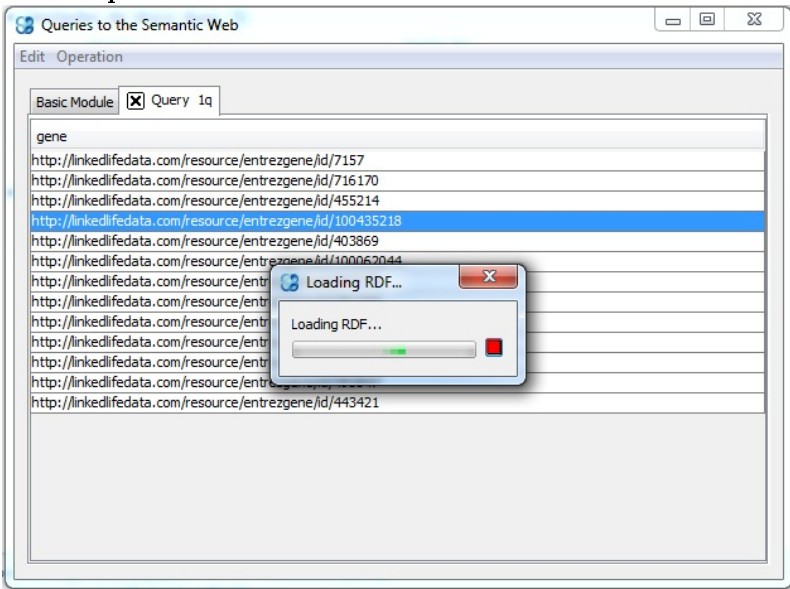
Nombre del CU	Ejecutar Consulta	
Actor	Usuario	
Propósito	Obtener resultado de la consulta.	
Resumen	El usuario inicia el caso de uso cuando decide ejecutar la consulta seleccionada. El sistema la ejecuta la consulta, mostrando los resultados de la misma.	
Referencias	RF 1	
Curso Normal de Eventos		
Acciones del Actor	Respuesta del Sistema	
1. El usuario ejecuta la consulta seleccionada.	1.1. El sistema muestra el resultado de la consulta. Finaliza el caso de uso.	
Cursos Alternativos		
CA1		

Prototipo de Interfaz de Usuario

Prioridad	Crítico
------------------	----------------

Caso de uso Obtener RDF

Nombre del CU	Obtener RDF
Actor	Usuario
Propósito	Obtener recursos RDF.
Resumen	El caso de uso se inicia cuando el usuario decide visualizar un recurso RDF de una consulta ejecutada.
Referencias	RF3
Curso Normal de Eventos	
Acciones del Actor	Respuesta del Sistema
1. El usuario selecciona dentro de los resultados de la consulta el recurso RDF que desea visualizar.	1.1. El sistema verifica si el resultado seleccionado por el usuario es un recurso RDF. 1.2. El sistema obtiene el recurso RDF y lo visualiza. Finaliza el caso de uso.

Cursos Alternativos	
CA1	
	1.1. Si el resultado seleccionado no es un recurso RDF, el sistema mostrará el siguiente mensaje de error “ <i>Not select a resource.</i> ”
Prototipo de Interfaz de Usuario	
	
Prioridad	Crítico

Caso de uso Transformar RDF a SBML

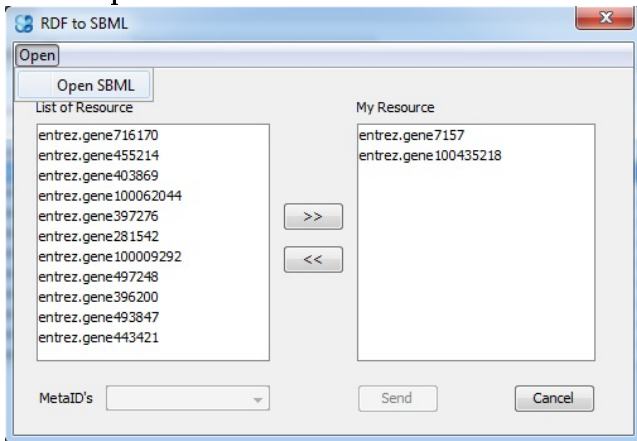
Nombre del CU	Transformar RDF a SBML
Actor	Usuario
Propósito	Obtener nuevos archivos SBML, con nueva información.
Resumen	El usuario inicia el caso de uso con la opción de convertir el RDF a SBML, luego debe seleccionar la opción de abrir el archivo SBML. Adiciona los recursos RDF a su lista de recursos y selecciona los campos del SBML que desea modificar. Finalmente, selecciona la opción de transformar el RDF. El sistema salva el SBML seleccionado con los nuevos recursos añadidos.
Referencias	RF5, RF5.1

APÉNDICE A: DESCRIPCIÓN DE CASOS DE USOS CRÍTICOS

Curso Normal de Eventos	
Acciones del Actor	Respuesta del Sistema
1. El usuario selecciona la opción "RDF to SBML".	1.1 El sistema muestra la interfaz con la lista de recursos obtenidos en las consultas ejecutadas.
2. El usuario abre su archivo SBML seleccionado en la opción "open SBML".	2.1 El sistema valida que el archivo es un SBML válido. 2.2 El sistema muestra los metaID del SBML cargado.
3. El usuario selecciona su metaID correspondiente y adiciona sus recursos a su lista.	
4. El usuario selecciona la opción "Send".	4.1. El sistema le permite al usuario seleccionar si desea sobrescribir el archivo SBML abierto o crear uno nuevo con las transformaciones correspondientes.
5. El usuario selecciona la opción que no desea sobrescribir el archivo SBML seleccionado.	5.1. El sistema le muestra un navegador de archivos.
6. El usuario selecciona la ubicación y el nombre del nuevo archivo que desea crear.	6.1. El sistema realiza la transformación correspondiente de los recursos RDF seleccionados por el usuario al metaID SBML especificado por el mismo. Finaliza el caso de uso.
Cursos Alternativos	
CA1	
	2.1. Si el archivo abierto no es un SBML válido, el sistema muestra el siguiente mensaje de error " <i>Your did not provide a correct SBML file!</i> ". Retorna a la actividad número 2. del flujo normal de eventos.
CA2	

5. El usuario selecciona la opción que desea sobrescribir el archivo SBML seleccionado. Retorna a la actividad número 6.1. del flujo normal de eventos.

Prototipo de Interfaz de Usuario



Prioridad

Crítico