Instituto Superior Politécnico "José Antonio Echeverría"

Facultad de Ingeniería Industrial

Universidad de las Ciencias Informáticas

HERRAMIENTA PARA LA MODELACIÓN DE SISTEMAS BIOLÓGICOS

Trabajo de diploma para optar por el título de Ingeniería en Informática

Autor: *Anthony Rafael Sotolongo León*
Tutor: *Lic. Noel Moreno Lemus*

Ciudad de La Habana, Cuba
Junio, 2006

Agradecimientos

Quisiera agradecer a todas las personas que han contribuido con su ayuda directa e indirecta con la realización de este trabajo. En especial a: toda mi familia que me ayudó en toda mi vida de estudiante, principalmente a mis padres y mi hermano.

A todos los profesores que de una forma u otra participaron en mi formación como profesional, además de los que pusieron su granito de arena para la realización de este documento, Ana Lupe Delgado y mi tutor Noel

Agradezco a todos mis amigos con los que compartí buenos momentos, fiestas, trabajo, estudio, juegos y que alguna vez me brindaron su apoyo en momentos difíciles.

También agradezco a mi jevita que e ayudo en la redacción de este documento.

Dedicatoria

Este trabajo esta dedicado a mi mamá, mi papá y a mi hermano que han seguido mi formación como profesional, desde los inicios me han dado todo su apoyo y orientación para llegar a esta meta.

Resumen

En el presente trabajo se realizó un estudio de los software para el diseño de sistemas biológicos disponibles en Internet, llegando a la conclusión de que es necesaria la implementación de una nueva aplicación que se ajuste más a las necesidades del Centro de Inmunología Molecular (CIM) y a los estándar internacionales existentes.

Se realizó además el análisis y diseño de dicha aplicación llevando a cabo las etapas del proceso unificado de desarrollo de software, demostrando la factibilidad de la implementación del mismo.

Índice

Índice de tablas

Índice de figuras

Introducción

Las enfermedades y los virus han sido unos de los males que a través de la historia han azotado a nuestra civilización. Su inicio data desde el surgimiento del hombre o mucho antes. Muchas enfermedades han cobrado la vida de millones de personas a lo largo de los siglos tales como: [3]

- La peste: en 1649, causó una epidemia dramática en Sevilla que acabó en tan sólo unas semanas con la vida de la mitad de la población.
- El cólera: durante el pasado siglo, esta enfermedad afectó a Rusia y al resto de Europa. Sólo en Francia se produjeron 600.000 víctimas entre 1826 y 1837.
- La lepra: causó verdadero pánico en todo occidente durante siglos. Esta enfermedad pasó por periodos de recrudescencia conforme aumentaban los peregrinajes a Tierra Santa y los intercambios comerciales. En la actualidad, todavía se registran casos en África y Asia.

El hombre no ha cesado en su empeño de poder vencerlas. Es así que la primera vacuna descubierta data de 1796, por el médico inglés Edward Jenner, y fue la vacuna para la viruela, uno de los peores males que sufría la humanidad. Una de cada diez personas moría por su causa. Jenner, un boticario y cirujano de Berkeley, notó que los que habían sufrido previamente viruela vacuna (enfermedad de las vacas que causa sólo síntomas de poca importancia en el hombre) demostraban resistencia cuando se exponían a la viruela. El 14 de mayo de 1796 extrajo pus de una pústula de la mano de Sarah Nelmes, una ordeñadora que había contraído viruela vacuna de su vaca lechera e inoculó a James Phipps, un niño saludable de 8 años, mediante dos incisiones superficiales. El niño desarrolló una leve enfermedad que desapareció sin la menor complicación. El 1º de julio, se inoculó al niño con la temida viruela mediante varios pinchazos e incisiones leves, pero no se enfermó. [1]

A partir de aquí comenzaron a surgir un gran número de vacunas para acabar con las enfermedades que atacan a la humanidad. He aquí una cronología de las más importantes. [2]

Siglo XIX

1. 1879 Primera vacuna para Cólera
2. 1882 Primera vacuna para Rabia
3. 1897 Primera vacuna para Peste

Siglo XX

1. 1926 Primera vacuna para Tos ferina
2. 1927 Primera vacuna para Tuberculosis
3. 1935 Primera vacuna para Fiebre amarilla
4. 1937 Primera vacuna para Tifus
5. 1945 Primera vacuna para Gripe

Pero casi a la misma vez que el hombre lucha contra las enfermedades, con los adelantos de la ciencia, creando nuevos medicamentos también aparecen nuevas. Tal es el caso de SIDA que tiene más de 20 años de aparición y que sigue cobrando millones de víctimas en todo el mundo.

En la actualidad existen grandes compañías dedicadas a la búsqueda de nuevas soluciones farmacéuticas para las enfermedades peligrosas para el hombre. Las mismas están equipadas por la mejor tecnología. Aun así las investigaciones duran más de 12 años y sus costos ascienden a 500 millones de dólares.

En la búsqueda incesante de nuevos métodos de investigación, más rápidos y menos costosos, ha surgido la modelación *in silico*.

La modelación *in silico,* según el criterio de los científicos será la próxima ola de descubrimientos biológicos. También tiene el potencial de disminuir los tiempos de obtención de medicamentos de 12 años a tan solo 12-24 meses obteniéndose nueva y mejores drogas sin la necesidad de ser blanco de cuestionamientos en materia de ética profesional y humana [4].

En los últimos años, el desarrollo impetuoso de la Biología y el abandono cada vez más creciente de los paradigmas reduccionistas de la Biología Molecular, se abre paso a una nueva biología que hace creciente uso de las modelaciones matemáticas de sus problemas. Esta biología es la que se ha dado a llamar "biología de sistemas", en alusión a que tendrá por objeto de estudio la comprensión integrada de un sistema biológico y no de sus partes independientes.

Ya en Estados Unidos se han creado varias empresas encargadas de desarrollar plataformas para el desarrollo de la biología de sistemas. Ejemplo de ellas son: [4]

1. Gene Network Sciences (www.gnsbiotech.com, fundada 2000).
2. Entelos (www.entelos.com, fundada 1996).
3. Physiome (www.physiome.com, fundada 2001)
4. Genomática (www.genomatica.com, fundada 2001)

Nuestro país ha alcanzado un gran nivel en las ramas biotecnológicas y en el desarrollo de aplicaciones médicas en las que se han obtenido muy buenos resultados. Es por ello que el desarrollo de una plataforma de modelación de sistemas biológicos que permita acelerar el proceso de obtención de medicamentos sería una herramienta, más que necesaria, imprescindible para poner a la biotecnología cubana en igualdad de condiciones con el resto de las compañías del mundo en este campo. [4]

En el Centro de Inmunología Molecular (CIM) existe un desarrollo creciente en la aplicación de modelos matemáticos en la solución de los problemas biológicos que se estudian.

Por lo que el ***problema científico*** de nuestro trabajo es la necesidad de contar con un diseño informático para una herramienta computacional que les facilite las investigaciones en el campo de los sistemas biológicos.

La ***Hipótesis*** es que con el diseño a desarrollar se facilitará la implementación de la aplicación informática para acelerar las investigaciones en el campo de los sistemas biológicos.

Este trabajo persigue como ***objetivo general*** desarrollar un diseño para una herramienta capaz de modelar sistemas biológicos y que sea de gran uso para los especialistas del CIM.

De este objetivo general se derivan los siguientes ***objetivos específicos***

1. Realizar una investigación que abarque lo referente a los sistemas biológicos y las interacciones dentro de los mismos.
2. Analizar y diseñar una herramienta capaz de modelar y almacenar la información de un Sistema Biológico
3. Analizar la factibilidad del proyecto

Para darle solución a la problemática planteada y lograr el cumplimiento de los objetivos se proponen las siguientes tareas:

1. Realizar una investigación que permita recopilar la información necesaria para lograr un estudio de los sistemas biológico y su comportamiento.
2. Realizar un estudio de las tendencias y metodologías actuales usadas a nivel mundial en la producción de software.
3. Seleccionar la metodología de análisis y diseño informático que facilite la creación y garantice la calidad del sistema, así como su mantenimiento.
4. Realizar el estudio de factibilidad de la aplicación

Para lograr una mejor comprensión el presente documento se estructura en cinco capítulos de contenidos donde se incluye todo lo relacionado con el trabajo investigativo realizado.

En el ***Capítulo 1: Fundamentación Teórica***, se realiza una descripción de cual es la misión de la entidad (CIM) relacionada con el proyecto, se incluye el análisis de la biología de sistemas así como los métodos matemáticos a utilizar.

Se describen algunos software dedicados al desarrollo de la biología de sistemas. También se realiza un análisis a las tendencias y tecnologías actuales con el objetivo de seleccionar las herramientas para desarrollar un producto de calidad.

El ***Capítulo 2: Modelo del dominio***, describe la solución propuesta utilizando los componentes del modelo del dominio de la metodología RUP. De este modelo se tendrán en cuenta la definición de las entidades y los conceptos principales, también se conocerán las reglas del negocio.

El ***Capítulo 3: Requisitos***, describe los Requisitos Funcionales y No Funcionales del sistema, así como los Actores que intervienen en el sistema. También se encuentra el Diagrama de Casos de Uso y la descripción de cada uno.

El ***Capítulo 4: Descripción de la solución propuesta,*** realiza el diagrama de clases del diseño, así como el diagrama de clases persistentes y se describen los tipos de ficheros donde se van a almacenar la información de los Sistemas Biológicos. Se analiza el diseño de interfaz a utilizar y el tratamiento de errores.

El ***Capítulo 5: Estudio de factibilidad,*** describe y se aplica la técnica basada en casos de uso para la estimación de esfuerzo.

Capítulo 1 Fundamentación Teórica

1.1 Introducción

En este capítulo se realiza una descripción de cual es la misión de la entidad (CIM) relacionada con el proyecto, así como un análisis de como se realizan los procesos relacionados con la materia de estudio en la misma. Se realiza también un análisis de la biología de sistemas así como los métodos matemáticos a utilizar, se describen algunos software dedicados al desarrollo de la biología de sistemas y se analizan las tendencias y tecnologías actuales con el objetivo de seleccionar las herramientas para desarrollar un producto de calidad.

1.2 Objeto de estudio

1.2.1 Objetivos estratégicos de la organización

El 5 de Diciembre de 1994 se inaugura el Centro de Inmunología Molecular (CIM) al oeste de la capital. El Centro de Inmunología Molecular tiene como principal misión obtener y producir nuevos biofármacos destinados al tratamiento del cáncer y otras enfermedades crónicas no transmisibles e introducirlos en la salud pública cubana. Hacer la actividad científica y productiva económicamente sostenible y realizar aportes importantes a la economía del país.

En el Centro de Inmunología Molecular laboran cerca de 400 trabajadores, en su mayoría científicos e ingenieros de forma multidisciplinaría. Actualmente el CIM fabrica productos Biofarmacéuticos, tales como: un anticuerpo monoclonal anti CD3 para el tratamiento de pacientes con rechazo al trasplante de órganos, Eritropoyetina humana recombinante para el tratamiento de la anemia. [5]

1.2.2 Flujo actual de los procesos

En la entidad (CIM) se viene modelando sistemas biológicos desde hace varios años con el objetivo de acelerar el proceso de obtención de nuevos fármacos. Esta tarea se realiza de forma manual, es decir, se realiza un trabajo de mesa donde se estudia el sistema a modelar, se crea el modelo matemático correspondiente al mismo y dicha información se almacena en papel.

1.2.3 Análisis crítico de la ejecución de los procesos

El proceso que se realiza tiene dificultades debido a que toda información relacionada a los sistemas biológicos se diseña y almacena en papel. Esto trae como consecuencia que el posterior análisis de dicha información sea muy tedioso y lento, por lo que se hace necesario el uso de computadoras que además cuenten con las herramientas matemáticas para agilizar la obtención de resultados.

1.3 Procesos objeto de automatización

Para el análisis de los sistemas biológicos, basado totalmente en la dinámica de poblaciones, se tienen en cuenta las formas más comunes de interacción entre poblaciones para las que se tienen las ecuaciones matemáticas que lo describen. Se hace necesario mencionar que después de un análisis con los especialistas se tuvieron en cuenta los casos que se describen a continuación. [4]

Relaciones externas

Proliferación o nacimiento de una población.

Representa la proliferación o nacimiento de una población. La ecuación diferencial asociada a esta interacción es:

$$dP/dt = k*P$$

Muerte de una población.

Representa la muerte de una población. Así la ecuación que representa la muerte de una población está dada por:

$$dP/dt = -k*P$$

Diferenciación.

Diferenciación o tránsito directo de una población a otra. Así las ecuaciones que representa la diferenciación está dada por:

$$dP_1/dt = -c_1*k_1*P_1^{C_1}$$

$$dP_2/dt = c_2*k_2*P_1^{C_1}$$

Donde:

c_1 y c_2 son los coeficientes estequiométricos para esta relación de las poblaciones P_1 y P_2 respectivamente.

k_1 y k_2 son coeficientes o expresiones que determinan la velocidad de pérdida y ganancia respectivamente.

También puede darse el caso que las poblaciones P_1 y P_2 se diferencian en la población P_3. Para este ejemplo las ecuaciones serian:

$$dP_1/dt = -c_1*k_1*P_1^{C_1}*P_2^{C_2}$$

$$dP_2/dt = -c_2*k_1*P_1^{C_1}*P_2^{C_2}$$

$$dP_3/dt = c_3*k_2*P_1^{C_1}*P_2^{C_2}$$

Donde en cada ecuación debe incluirse la expresión de los otros nodos que interactúan.

Producción de compuestos químicos.

Como resultado de las transformaciones de una población se pueden formar determinados compuestos químicos. La particularidad de este proceso es que

no afecta la cantidad de individuos de la población o las poblaciones que originan la sustancia.

La ecuación correspondiente es:

$$dQ / dt = c_q * k * P^{C}_{p}$$

donde:

c_q y c_p son los coeficientes estequiométricos del compuesto químico Q y la población P respectivamente.

Catalizadores e inhibidores.

Las poblaciones y sustancias químicas pueden actuar como reguladores de la velocidad de un proceso, bien sea como catalizador o inhibidor de la misma. Los catalizadores o inhibidores tienen como característica que ellos aceleran o retardan la velocidad del proceso sin variar su cantidad de sustancia o de individuos del mismo, según sea el caso.

A continuación se muestran algunas de las funciones más usadas:

- $k \frac{P^C}{1+P^C}$
- $k \frac{P^C}{1-P^C}$
- C*ln(P)
- $k*a^{-P}$

Suponiendo que en la diferenciación de P1 a P2 interactúa P3 como catalizador.

Utilizando la primera de las funciones que aquí exponemos las ecuaciones del modelo serían:

$$dP_1 / dt = -C_1 * k_1 * \frac{P_3^{C_3}}{1+P_3^{C_3}} P_1^{C_1}$$

$$dP_2 / dt = C_2 * k_2 * \frac{P_3^{C_3}}{1 + P_3^{C_3}} P_1^{C_1}$$

Operadores internos

Autoactivación es una señal de inicio interna que depende de la población y su ecuación es:

$$\frac{(s1)^n}{Ks1 + (s1)^n}$$

Activación es una señal que proviene de un receptor que fue activado por un factor externo y su ecuación es:

$$\frac{(s1)^n}{Ks1 + (s1)^n}$$

Inhibición es una señal que actúa sobre otra señal que significa que inhibe la señal y su ecuación es:

$$\frac{1}{Ks1 + (s1)^n}$$

Catálisis es una señal que actúa sobre otra señal acelerando el proceso en cuestión y su ecuación es:

$$\frac{(s1)^n}{Ks1 + (s1)^n}$$

En el caso de la catálisis y la Inhibición se multiplica su expresión por la de la señal sobre la cual inciden.

Ejemplo:

Sea la Señal $-\alpha * \frac{(s1)^n}{Ks1 + (s1)^n}$ y sobre ella ocurre una inhibición entonces como resultado se obtiene $^{-\alpha} * \frac{(s1)^n}{Ks1 + (s1)^n} * \frac{1}{Ks1 + (s1)^n}$

AND es un operador sobre varias señales y lo que se obtiene de el es el producto de dichas señales de manera que si algunas de las señales es cero el producto también lo es.

Ejemplo:

Sea señal 1 $\frac{(s1)^n}{Ks1+(s1)^n}$ y señal 2 $\frac{(s2)^n}{Ks2+(s2)^n}$ se obtendría

$$\frac{(s1)^n}{Ks1+(s1)^n} * \frac{(s2)^n}{Ks1+(s2)^n}$$

OR es un operador sobre varias señales y lo que se obtiene de el es la suma de dichas señales de manera que si algunas de las señales es cero la suma no lo sea.

1.4 Sistemas automatizados existentes vinculados al campo de acción

BIOUML

BioUML es una aplicación que permite diseñar sistemas biológicos, proporciona la estructura de la notación y el funcionamiento gráficos formalizados de sistemas biológicos, de su visualización y de la simulación, así como el acceso a las bases de datos de datos experimentales relevantes. Tiene módulo para los modelos en formato SBML; módulo para la base de datos de GeneNet; módulo para la base de datos de KEGG/Pathways; módulo para la base de datos de TRANSPATH.

Requisitos

- Máquina virtual de Java 1.4 o superior.
- Pentium II o superior.
- una memoria más alta: Mb 128 o superior.

- Espacio de disco: 20 Mb o superior.

En los diagramas de clases que definen uno está dedicado a la parte visual y el otro a la parte de datos interna. Explotan poco la parte interna, no contemplan datos necesarios para otro tipo de análisis más general y no tiene un modelamiento para las interacciones internas de las poblaciones biológicas.

Esta aplicación tiene una versión para Linux y otro para Windows, pero no posee salva de ficheros MathML lo que representa una dificultad a la hora del análisis del las ecuaciones del sistema biológico. [6]

STELLA

STELLA es un simulador que analiza la dinámica de sistemas. Sus modelos proporcionan oportunidades para educadores y estudiantes; usarlo para estudiar todo lo relacionada con la economía, la física, cálculo, química entre otros temas. Permite el análisis de la sensibilidad, revela los puntos dominantes del sistema y las condiciones óptimas. Los resultados son presentados como gráficos, tablas, animaciones, exporta e importa datos para Microsoft Excel, permite bloquear modelos con contraseña con el objetivo de lograr seguridad en los mismos. [7]

Requerimientos del sistema

Windows:

233 MHz Pentium

Microsoft Windows™ 2000/XP

(versión en ingles)

64 MB RAM

70 MB espacio en disco duro

16-BIT color

QuickTime

Macintosh:

120 MHz PowerPC

Mac OS 10.2.8 o superior

(versión en ingles)

128 MB RAM

70 MB espacio en disco duro
QuickTime

Esta aplicación se aleja un poco de la biología de sistema porque no trata ficheros relacionado con los mismos y presenta dificultad para modelarlos.

SimBiology

SimBiology es una potente herramienta que tiene un ambiente integrado para modelar procesos biológicos, simular el comportamiento dinámico de estos procesos, y analizar el modelo mediante simulaciones y su comparación con datos experimentales. Incluye modelos de procesos metabólicos y genéticos.

Diseña y construye los modelos utlizando reacciones, especie, parámetros, leyes cinéticas y reglas. Presenta una interfaz gráfica y está integrado a MATLAB. Tiene la posibilidad de importar ficheros en formato SBML.
Requerimientos [25]

Windows
Windows 2000 o superior
Procesador Pentium III o superior
Espacio disco 420 mb
Ram 512mb
Linux
Kernel: 2.4.x o 2.6.x
Procesador Pentium III o superior
Espacio disco 420 mb
Ram 512mb
Esta herramienta tampoco tiene la posibilidad de exportar ficheros MathML lo que representa un freno a la hora de analizar el comportamiento del sistema biológico.

Analizando los software relacionados con la materia de estudio de este trabajo y teniendo en cuenta los siguientes criterios: mecánismo de elaboración de un sistema biológico, salva de ficheros con información biológica en diferentes formatos (MathML y SBML) y que sea multiplataforma, se decide hacer una aplicación propia que cumpla con los criterios antes mencionados. Si nombre sería **BSModel**, acrónimo del inglés **Bio**logical **S**ystem **Model**ation.

1.5 Fundamentación de los objetivos

1.5.1 Objetivo general

Desarrollar un diseño para una herramienta capaz de modelar sistemas biológicos y que sea de gran uso para los especialistas del CIM.

1.5.2 Objetivos específicos

- Realizar una investigación que abarque lo referente a los sistemas biológicos y la interacción dentro de los mismos.
- Analizar y diseñar una herramienta capaz de modelar y almacenar la información de un Sistema Biológico
- Analizar la factibilidad del proyecto

1.6 Tendencias y tecnologías actuales

Con el objetivo de desarrollar un producto de calidad se hace necesario realizar un análisis de las tecnologías actuales, conocidas, con el fin de seleccionar y utilizar la más conveniente, así como elegir una metodología de desarrollo óptima para su terminación.

Lenguaje de Modelamiento Unificado (UML)

Es un lenguaje que permite modelar, construir y documentar una aplicación informática orientada a objeto. Los autores Grady Booch, Ivar Jacobson y Jim Rumbaugh han dado un impulso a este lenguaje de modelado por la importancia que tiene, la cual ha sido aceptada en el mundo entero por el prestigio de sus autores. Uno de los objetivos principales de la creación de UML era posibilitar el

intercambio de modelos entre las distintas herramientas CASE orientadas a objetos del mercado. Para ello era necesario definir una notación y semántica común. UML también intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos los desarrollos se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo. [8]

Procesos de desarrollo

Los procesos de desarrollo de software están dirigidos a elevar la calidad del software en producción según las fases por la que pasa el mismo. El establecimiento de un proceso de desarrollo es una tarea más a medio-largo plazo que de resultados inmediatos, a los trabajadores les cuesta un poco de tiempo la adaptación pero una vez que esto sucede se recupera en tiempo con creces.

Proceso Unificado de Desarrollo (RUP)

El Proceso Unificado (RUP - Rational Unified Process) es un marco genérico de trabajo que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto.

Utiliza UML para modelar todas las especificaciones de un sistema de software, de hecho UML es una parte esencial del proceso unificado.

RUP tiene como objetivo: Asegurar la producción de software de calidad dentro de plazos y presupuestos predecibles.

El Proceso Unificado está *dirigido por casos de uso, es centrado en la arquitectura y es iterativo e incremental*: [9]

- *Guiado por los casos de uso*: Los casos de uso son el instrumento para validar la arquitectura del software y extraer los casos de prueba.

- *Centrado en la arquitectura*: Los modelos son proyecciones del análisis y el diseño, constituyen la arquitectura del producto a desarrollar.
- *Iterativo e incremental*: Durante todo el proceso de desarrollo se producen versiones incrementales (que se acercan al producto terminado) del producto en desarrollo.

RUP divide el proceso de desarrollo en ciclos, teniendo un producto al final de cada ciclo.

Cada ciclo se divide en cuatro Fases (Figura. 1):

- Inicio
- Elaboración
- Construcción
- Transición

Cada fase concluye con un hito bien definido donde deben tomarse ciertas decisiones.

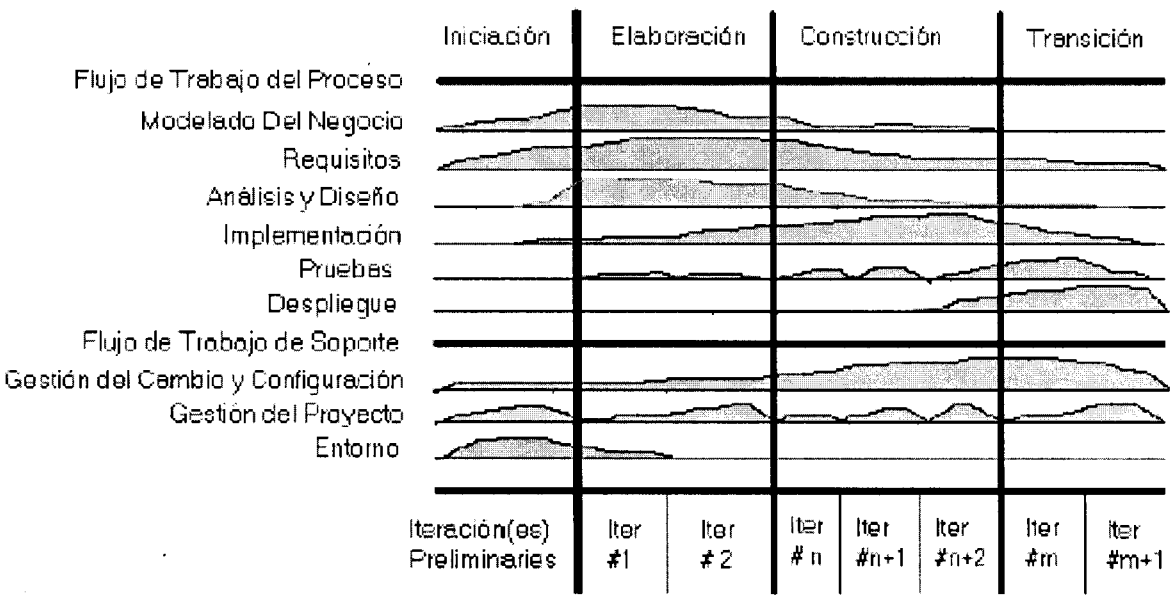


Figura 1 El Proceso de Modelado Unificado

Fase de inicio

- Se establece la oportunidad y alcance el proyecto.

- Se identifican todas las entidades externas con las que se trata (actores) y se define la interacción a un alto nivel de abstracción:
  - Identificar todos los casos de uso
  - Describir algunos en detalle
- La oportunidad del negocio incluye:
  - Criterios de éxito
  - Identificación de riesgos
  - Estimación de recursos necesarios
  - Plan de las fases incluyendo hitos
- Un documento de visión general:
  - Requisitos generales del proyecto
  - Características principales
  - Restricciones
- Modelo inicial de casos de uso (10% a 20 % listos).
- Glosario.
- Caso de negocio:
  - Contexto
  - Criterios de éxito
  - Pronóstico financiero
- Identificación inicial de riesgos.
- Plan de proyecto.
- Uno o más prototipos.

Hitos:

- Las partes interesadas deben acordar el alcance y la estimación de tiempo y costo.
- Comprensión de los requisitos plasmados en casos de uso.

Fase de elaboración

- Objetivos:
  - Analizar el dominio del problema
  - Establecer una arquitectura base sólida

  - Desarrollar un plan de proyecto
  - Eliminar los elementos de mayor riesgo para el desarrollo exitoso del proyecto
- Modelo de casos de uso (80% completo) con descripciones detalladas.
- Otros requisitos no funcionales o no asociados a casos de uso.
- Descripción de la Arquitectura del Software.
- Un prototipo ejecutable de la arquitectura.
- Lista revisada de riesgos y del caso de negocio.
- Plan de desarrollo para el resto del proyecto.
- Un manual de usuario preliminar.

Hitos:

- Condiciones de éxito de la elaboración:
  - ¿Es estable la visión del producto?
  - ¿Es estable la arquitectura?
  - ¿Los riesgos han sido abordados y resueltos?
  - ¿Es el plan del proyecto algo realista?
  - ¿Están de acuerdo con el plan todas las personas involucradas?

Fase de construcción

- En esta fase todas las componentes restantes se desarrollan e incorporan al producto.
- Todo es probado en profundidad.
- El énfasis está en la producción eficiente y no ya en la creación intelectual.
- Puede hacerse construcción en paralelo, pero esto exige una planificación detallada y una arquitectura muy estable.
- El producto de software integrado y corriendo en la plataforma adecuada.
- Manuales de usuario.

Hitos:

- Se obtiene un producto Beta que debe decidirse si puede ponerse en ejecución sin mayores riesgos.

- Condiciones de éxito:
  - ¿El producto está maduro y estable para instalarlo en el ambiente del cliente?
  - ¿Están los interesados listos para recibirlo?

Fase de Transición

- El objetivo es traspasar el software desarrollado a la comunidad de usuarios.
- Una vez instalado surgirán nuevos elementos que implicarán nuevos desarrollos (ciclos).
- Incluye:
  - Pruebas Beta para validar el producto con las expectativas del cliente
  - Ejecución paralela con sistemas antiguos
  - Conversión de datos
  - Entrenamiento de usuarios
  - Distribuir el producto
- Obtener autosuficiencia de parte de los usuarios.
- Concordancia en los logros del producto de parte de las personas involucradas.
- Lograr el consenso cuanto antes para liberar el producto al mercado.

XP

La metodología XP intenta minimizar el riesgo de fallo del proceso por medio de la disposición permanente de un representante competente del cliente a disposición del equipo de desarrollo. Dicho representante debe estar a disposición de los desarrolladores para contestar cualquier pregunta de inmediato y disminuir la demora. [10]

Define UserStories como base del software a desarrollar. Estas historias las escribe el cliente y describen escenarios sobre el funcionamiento del software, el modelo, dominio, etc. A partir de las UserStories y de la arquitectura perseguida se crea un plan de releases entre el equipo de desarrollo y el cliente.

Para cada una releases se debatirán los objetivos con el representante del cliente y se definirán las iteraciones (de poca duración) necesarias para cumplir con los objetivos de la misma. El resultado de cada iteración es un programa que se transmite al cliente para que lo critique. En base a su opinión se definen las siguientes iteraciones del proyecto y si el cliente no esta contento se adaptará a su decisión. A continuación se muestra una vista general del ciclo de vida de XP.

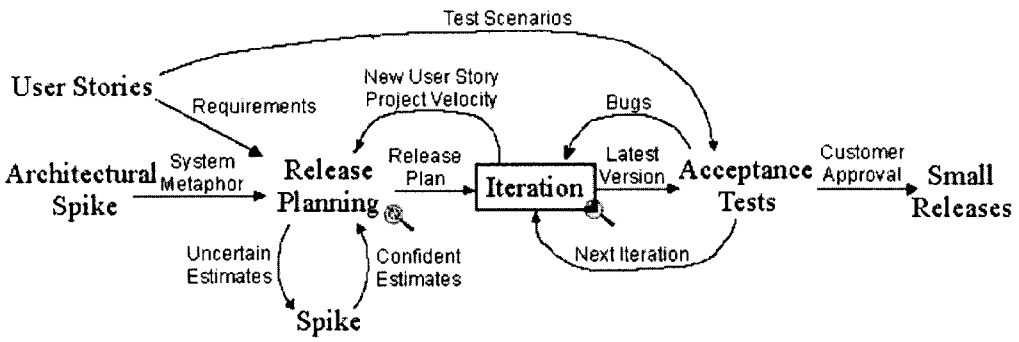


Figura 2 Ciclo de vida de XP

Plataforma de modelación y lenguajes de programación

Rational Rose

Rational Rose es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables. [11]

Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue).

El Racional Rose esta dirigido a Profesionales relacionados con algunas de las siguientes actividades:

- Dirección de proyectos de desarrollo de software.
- Análisis y programación de componentes de software.
- Diseño de la arquitectura de sistemas.
- Implementación de patrones de funcionalidad, análisis y diseño.
- Estudios de métrica y estimación de costes en el desarrollo de software.
- Certificaciones de calidad, acreditación y auditoria de sistemas.
- Re-ingeniería de procesos y mejora continúa de flujos de trabajo.

Borland Delphi

Borland Delphi es un entorno rápido de desarrollo (RAD) diseñado para la programación de propósito general, con un énfasis especial en la programación visual y utiliza el Object Pascal como lenguaje de programación. Obtiene su nombre (pronunciado delfi) del oráculo de Delfos. Compilando su código se generan programas ejecutables en plataformas Windows de 32 Bits (win32.exe), tanto en entorno gráfico (GUI) como en modo de consola. También pueden programarse aplicaciones de múltiples capas (clientes y servidores), aplicaciones cgi, utiliza la programación orientada a objeto y permite la reutilización de código, posee componentes inteligentes de datos para construir poderosas aplicaciones de base de datos, sus aplicaciones solo funcionan sobre sistema Operativo Windows. [12]

Borland C++ Builder

Es una herramienta de propósito general, utiliza C++ como lenguaje de programación, soporta la programación orientada a objeto, es un ambiente de desarrollo rápido de aplicaciones (RAD), posee componentes inteligentes de

datos para construir poderosas aplicaciones de base de datos, presenta asistente para recurso DLL, depurador de las DLL para un más fácil y completo control de depuración, visor de CPU para la depuración de bajo nivel. Permite la importación de código C++ existente. Solo funcionan sobre la plataforma de trabajo Windows. Como Borland Delphi, [13]

Eclipse

Eclipse es una poderosa herramienta que permite integrar diferentes aplicaciones para construir un entorno integrado de desarrollo (IDE). Es un potente entorno de desarrollo de Java para que las tareas de programación, prueba, usa java como lenguaje de programación aunque permite plugins para varios lenguajes más, soporta la programación orientada a objetos (POO), la depuración e implementación de aplicaciones resultan mucho más sencillas. La plataforma esta construida en base a plugins. Este mecanismo permite desarrollar, integrar y correr nuevos plugins. Existen por lo general, dos formas de instalar los nuevos plugins en Eclipse. En la mayoría de los casos sólo hay que descompactar el zip del plugin en el directorio en el que se encuentra instalado Eclipse. La misma tiene varios beneficios como son [14,15]:

- Es una herramienta open-source.
- Soporta herramientas que manipulan diferentes tipos de lenguajes, como por ejemplo Java , C, C++,
- Corre en una gran cantidad de sistemas operativos incluyendo Windows y Linux.
- Provee a los desarrolladores, herramientas (ej.- PDE) que facilitan la creación de plugins.

1.7 Conclusiones

Después de este análisis hecho a las herramientas y teniendo en cuenta los siguientes aspectos:

- Se necesita una aplicación multiplataforma.
- Se requiere que se almacene información biológica.
- Se necesita documentación para posibles versiones futuras

Los desarrolladores tienen más dominio sobre el lenguaje lenguaje C++ y el Java que sobre Object Pascal y la metodología de desarrollo que conocen es RUP.

Se concluye que:

Ha sido seleccionada la metodología RUP, el lenguaje Java utilizando como plataforma de programación el eclipse para llevar a cabo la implementación del software propuesto, así como el Rational Rose para el diseño de los diagramas UML.

Capítulo 2 Modelo del dominio

2.1 Introducción

En el presente capítulo se describe la solución propuesta utilizando los componentes del modelo del dominio de la metodología RUP. De este modelo se tendrán en cuenta la definición de las entidades y los conceptos principales, así como se representará gráficamente dicho modelo, también se conocerán las reglas del negocio.

2.2 ¿Por qué modelo del dominio?

Teniendo en cuenta que la definición de procesos y roles del proceso del negocio que tienen que ver con el objeto de estudio se hace difícil encontrarlos, por lo que se ve a simple vista la necesidad de describir el funcionamiento de la aplicación mediante una serie de conceptos, entidades y sus relaciones, agrupándolos en un modelo del dominio con el fin del fácil entendimiento de la aplicación.

2.3 Definición de las entidades y los conceptos principales

Un Modelo del dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan los principales conceptos. Muchos de los objetos o clases pueden obtenerse de una especificación de requisitos [16]. La modelación del dominio tiene como objetivo fundamental la comprensión y descripción de las clases más importantes en el sistema.

Con la aplicación del Modelo del dominio se detectaron las siguientes entidades y conceptos (objetos):

- Sistema Biológico: conjuntos de entes biológicos que interactúan entre si y funcionan como un todo integro

- Población: es la unidad estructural y funcional que representa entidades biológicas
- Compuesto químico: es una sustancia formada por la unión de dos o más elementos de la tabla periódica
- Interacción: relaciones que existen entre las poblaciones y compuesto químicos
- Componentes internos: son los operadores internos que tienen las Poblaciones
- Sistema de ecuaciones: es lo que tiene todas las ecuaciones del sistema biológico
- Archivo MathML: este archivo almacena la información relacionada a un sistema de ecuaciones.
- Archivo SBML: este archivo almacena la información relacionada a un sistema biológico de manera general.
- Ecuación: tiene la información matemática de una población expresada en términos de una ecuación diferencial
- Expresión: tiene la información matemática de una interacción

2.4 Reglas del negocio a considerar

Un sistema biológico esta compuesto por una cantidad determinada de poblaciones y compuestos químicos así como las interacciones que los relacionan. Las interacciones que pueden relacionar a las poblaciones son: el nacimiento, la muerte y la diferenciación. La interacción que relaciona poblaciones y CQ es la de formación de CQ que solo pueden ser formados por una población. También se encuentran los procesos de catalización e inhibición, los que pueden ser originados por poblaciones y CQ y que solo actúan sobre otras interacciones. Las poblaciones y CQ cuentan con un coeficiente estequiométrico (C) y una ecuación matemática que esta compuesta por los expresiones de los proceso vinculados con ellas. Los procesos tienen una

expresión característica con un coeficiente que determina la velocidad de pérdida o ganancia del mismo. Dentro de las poblaciones también ocurren reacciones internas que justifican el comportamiento de las mismas, es decir el porque de los procesos que salen o entran de ellas y su combinaciones dentro. Los componentes internos que tienen son los siguientes: autoactivación, que es un operador que actúa solo dentro de la misma población y no depende de factores externos, provocando una respuesta; activación que es el operador que depende de un factor externo, la inhibición y catálisis que actúan sobre otra señal; y por ultimo los operadores *and* y *or* que actúan sobre varias señales dando como resultado una salida, resultado de una combinación de señales.

Representación del modelo del dominio

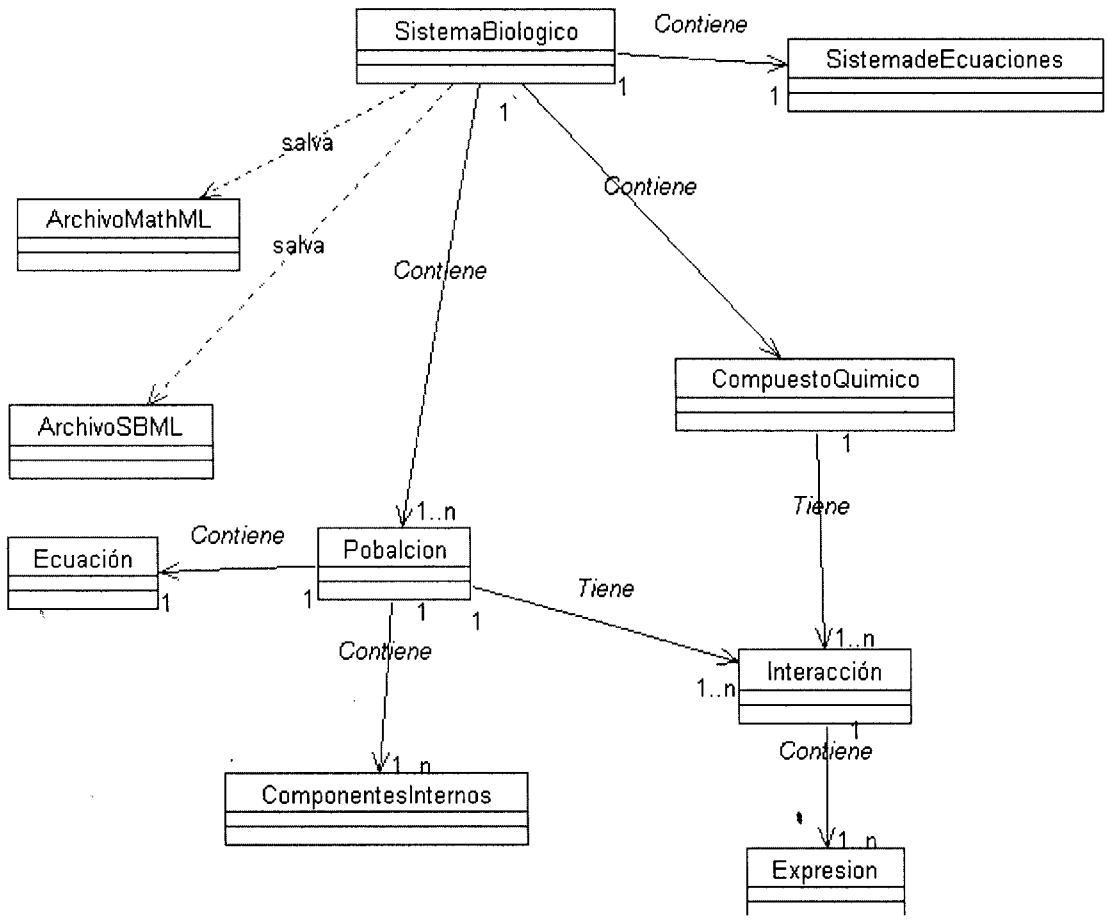


Figura 3 Modelo del dominio

2.5 Conclusiones

En este capítulo se ha descrito la solución propuesta utilizando el Modelo del Dominio, en el que se definieron 10 conceptos y sus relaciones. También se describieron las reglas del negocio.

Capítulo 3 Requisitos

3.1 Introducción

En el presente capítulo se describe los Requisitos Funcionales y No Funcionales del sistema, así como los Actores que intervienen en el sistema, también se encuentra el Diagrama de Casos de Uso y la descripción de cada uno.

3.2 Actores del sistema a automatizar

El término *actor* significa el rol que algo o alguien juega cuando interactúa con el sistema. Un candidato a actor del sistema es cualquier individuo, grupo, organización o máquina que interactúa en los casos de uso. De acuerdo con esta idea un actor del sistema representa un tipo particular de usuario del sistema más que un usuario físico, ya que varios usuarios físicos pueden realizar el mismo papel en relación al negocio, o sea, ser instancias de un mismo actor. Una vez que hemos identificado los actores del sistema, tenemos identificado el entorno externo del sistema. [16]

Nombre del Actor	**Justificación**
Investigador	Cualquier usuario que sea especialista de la Biología de sistema que interactúe con el sistema, que necesite realizar el diseño de un Sistema Biológico. Este usuario tendrá la posibilidad de interactuar con todos la funcionalidades del sistema.

Tabla.1 Actores del sistema

3.3 Definición de requisitos funcionales

Los requerimientos funcionales permiten expresar una especificación más detallada de las responsabilidades del sistema que se propone. Ellos permiten determinar, de una manera clara, lo que debe hacer el sistema. [16]

Los requerimientos funcionales del software propuesto son los siguientes:

R1. Editar un sistema biológico
- R1.1 Crear un sistema biológico

R 2. Editar población
- R2.1 Creación de población
- R2.2 Eliminar población
- R2.3 Modificar población
- R2.4 Mostrar información general de la población

R3. Editar componentes internos
- R3.1 Creación de componentes internos
- R3.2 Eliminar componentes internos
- R3.3 Modificar componentes internos
- R3.4 Mostrar información general de las componentes internos

R4. Modificar ecuaciones

R5. Guardar archivos con formato MathML

R6. Editar los compuestos químicos.
- R6.1 Creación de un compuesto químico.
- R6.2 Eliminar un compuesto químico.
- R6.3 Modificar un compuesto químico.
- R6.4 Mostrar información de un compuesto químico

R7. Editar Interacción

R7.1 Creación de una interacción

R7.2 Eliminación de una interacción

R7.4 Modificar Interacción

R7.3 Mostrar información de una interacción

R8. Guardar archivos con formato SBML

3.4 Definición de los requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener, como restricciones del entorno o de implementación, rendimiento, etc. [16]

1. Apariencia o interfaz externa.

La aplicación estará diseñada de modo tal que el usuario tenga el mayor control posible de la misma, le permitirá navegar dentro de ella con facilidad, la Interfaz estará ajustada al estándar de de ventanas.

2. Usabilidad.

La aplicación estará orientada para que se use por especialista de la biología de sistema. El sistema tendrá un buen nivel de uso una vez instalado en cualquier institución del polo científico del país. Su explotación permitirá agilizar el proceso de obtención de nuevos medicamentos en el país.

3. Soporte

El sistema debe propiciar su mejoramiento y la anexión de otras opciones que se le incorporen en un futuro.

4. Portabilidad.

El sistema se ejecutará en cualquier plataforma, es decir deberá correr en Windows y Linux.

5. Confiabilidad.

   El sistema no debe presentar fallos y en casos de alguno debe garantizar que las pérdidas de información sean mínimas.

6. Ayuda y documentación en línea.

   Tendrá una ayuda en línea con documentación básica que comprenda los componentes y funcionalidades generales a tener en cuenta para explotar la aplicación así como información de la teoría de la biología de sistema.

7. Software.

   Se debe disponer de cualquier versión Windows 95 o superior, y cualquier versión de Linux.

   Máquina Virtual de Java 1.4 o superior.

   Hardware.

   Para el desarrollo y puesta en práctica del proyecto se requieren máquinas con los siguientes requisitos:

   - Procesador PENTIUM
   - 128 Mbyte de RAM
   - 120 Mbyte de HDD
   - UPS o fuente de corriente ininterrumpida.

3.5 Descripción de los casos de uso

Los casos de uso se utilizan para obtener información de cómo debe trabajar el sistema, son descripciones de la funcionalidad del sistema independiente de la implementación, describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista del usuario. [17,18]

Los casos de usos definidos son los siguientes

- Editar un Sistema Biológico.
- Editar población.
- Editar componentes internos.
- Modificar ecuaciones.
- Guardar archivos con formato MathML.
- Guardar archivos con formato SBML.
- Editar compuestos los químicos.
- Editar una interacción.

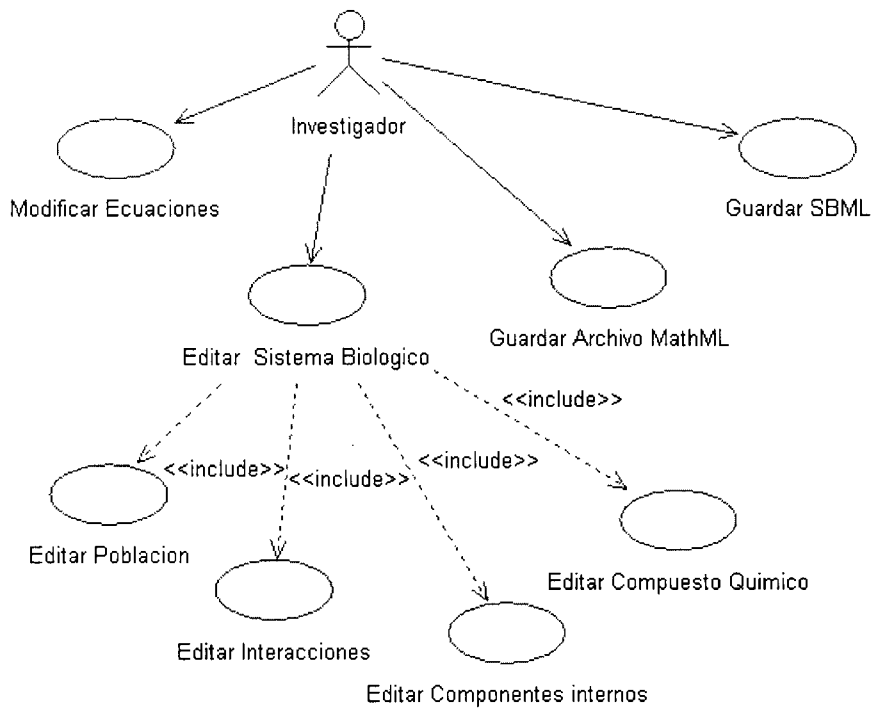


Figura 4 Diagrama de caso de uso del sistema

Caso de uso:	Editar un SB.
Actores:	Investigador (inicia).
Proposito : Editar un SB	
Descripción: Un Investigador quiere elaborar un SB y cuenta con toda la información necesaria que se basa en población, compuestos químicos, relaciones entre ellas y las relaciones entre los operadores dentro de las poblaciones. El software le da la posibilidad de insertar en un SB las poblaciones o compuestos químicos en forma de nodos e insertar los enlaces que existen entre estas, con sus correspondientes nombres. El Investigador tiene la posibilidad de **Editar Población, Editar Compuesto Químico, Editar interacciones y Editar componentes Internos** que está creando.	
Referencias:	R2 ,R2.1, R2.2, R2.3 R2.4, R3, R3.1, R3.2, R3.3, R3.4, R6, R6.1, R6.2, R6.3, R6.4, R7, R6.1, R7.2 y R7.3 **Casos de uso asociado** Editar Población, Editar Compuesto Químico, Editar interacción Editar componentes Internos
Precondiciones:	
Postcondiciones:	Se creó un nuevo SB.
Requerimientos especiales:	
Curso Normal de los eventos	
Acciones del Actor	**Respuesta del sistema**
1. El Investigador decide realizar un nuevo SB.	1.1 El sistema da la posibilidad de crear un nuevo SB y muestra los componentes (población, relaciones, compuestos químicos, etc.).

2. El Investigador escoge los determinados componentes necesarios para su SB así como las relaciones que considere	2.1 El sistema ejecuta una de las siguiente acciones A) Si el Investigador quiere editar una población ir al caso de uso Editar Población. B) Si el quiere editar un compuesto químico ir al caso de uso Editar Compuesto Químico. C) Si el quiere editar una interacción ir al caso de uso Editar Interacción. D) Si el Investigador quiere editar componentes internos ir al caso de uso Editar Componentes Internos.

Tabla. 2 Caso de uso editar Sistema biológicos

Caso de uso:	Editar Población
Actores:	Investigador (inicia).
Propósito: Editar población	
Descripción: Un Investigador quiere editar una población y cuenta con las características generales de esa población El software le da la posibilidad de editar la población deseada. El Investigador tiene la posibilidad de **Crear Población, Modificar población, Eliminar población y Mostrar Información de la población.**	
Referencias:	R2.1, R2.2, R2.3, R2.4
Precondiciones:	Se debe estar trabajando sobre un SB
Postcondiciones:	Se editó una población
Requerimientos especiales:	
Curso Normal de los eventos	
Acciones del Actor	**Respuesta del sistema**
1. El Investigador decide editar una población.	1.1 El sistema da la posibilidad de crear una nueva población, de eliminar, mostrar su información y de modificarla.

	2.1 El sistema ejecuta la opción tomada por el Investigador. Opciones: A) Crear una población: se inserta una nueva población en el sistema biológico y lo muestra en el panel gráfico para dibujar el sistema biológico. B) Modificar población: se modifica la información de la población y se muestra en la barra que tiene la información de los componentes C) Eliminar población: se elimina la población del sistema biológico y se quita del panel donde se encuentra el sistema biológico. D) Mostrar información de la población: se muestra información de la población seleccionada, dicha información la mostrara en la barra de descripción.

Tabla.3 Caso de uso Editar Población

Caso de uso:	Editar Compuesto Químico
Actores:	Investigador (inicia).
Propósito: Editar Compuesto Químico	
Descripción: Un Investigador quiere editar un Compuesto Químico y cuenta con las características generales de ese Compuesto Químico El software le da la posibilidad de editar el Compuesto Químico deseado. El Investigador tiene la posibilidad de **Crear Compuesto Químico, Modificar Compuesto Químico, Eliminar Compuesto Químico y Mostrar Información del Compuesto Químico**.	
Referencias:	R6.1, R6.2, R6.3, R6.4
Precondiciones:	Se debe estar trabajando sobre un SB
Poscondiciones:	Se editó un Compuesto Químico
Requerimientos especiales:	
Curso Normal de los eventos	
Acciones del Actor	**Respuesta del sistema**
1. El Investigador decide editar un compuesto químico.	1.1 El sistema da la posibilidad de crear un nuevo compuesto químico, de eliminar, mostrar su información y de modificarla.

2. El Investigador escoge de las opciones que le brinda el sistema para el trabajo con compuesto químico.	2.1 El sistema ejecuta la opción tomada por el Investigador. Opciones: A) Crear un Compuesto Químico: se inserta un nuevo Compuesto Químico en el sistema biológico lo muestra en el panel gráfico para dibujar el sistema biológico. B) Modificar Compuesto Químico: se modifica la información del Compuesto Químico se muestra en la barra que tiene la información de los componentes C) Eliminar Compuesto Químico: se elimina el Compuesto Químico del sistema biológico se quita del panel donde se encuentra el sistema biológico. D) Mostrar información del Compuesto Químico: se muestra información del Compuesto Químico seleccionado y se mostrara en la barra de descripción.

Tabla.4 Caso de uso editar Compuesto químico

Caso de uso:	Editar Interacción
Actores:	Investigador (inicia).
Propósito: Editar una interacción	

Descripción: Un Investigador quiere editar una interacción y cuenta con las características generales de esa inatracción El software le da la posibilidad de editar la interacción deseada. El Investigador tiene la posibilidad de **Crear interacción, Eliminar interacción y Mostrar Información de la interacción.**	
Referencias:	R7.1, R7.2, R7.3
Precondiciones:	Se debe estar trabajando sobre un SB
Poscondiciones:	Se editó una interacción
Requerimientos especiales:	
Curso Normal de los eventos	
Acciones del Actor	**Respuesta del sistema**
1. El Investigador decide editar una interacción.	1.1 El sistema da la posibilidad de crear una nueva interacción, de eliminar, mostrar su información y de modificarla .

2. El Investigador escoge de las opciones que le brinda el sistema para el trabajo con la interacción.	2.1 El sistema ejecuta la opción tomada por el Investigador: Opciones: A) Crear una Interacción: se inserta una nueva Interacción en el sistema biológico ya se con otra población que con un compuesto químico y lo muestra en el panel gráfico para dibujar el sistema biológico. B) Eliminar una Interacción: se elimina la Interacción del sistema biológico y se quieta del panel gráfico donde esta el sistema biológico. C) Mostrar información de la Interacción: se muestra información de la Interacción seleccionada y se mostrara en la barra de descripción. D) Modificar información de la interacción: se modifica la información de la interacción que se muestra en la barra de información.

Tabla.5 Caso de uso editar interacción

Caso de uso:	Editar Componentes Internos
Actores:	Investigador (inicia).
Propósito: Editar Operadores Internos	

Descripción: Un Investigador quiere editar componentes Internos. El software le da la posibilidad de editar los componentes Internos deseados. El Investigador tiene la posibilidad de **Crear componentes Internos, Modificar Operadores Internos, Eliminar Operadores Internos y Mostrar Información de los Operadores Internos**	
Referencias:	R3.1, R3.2, R3.3 , R3.4
Precondiciones:	Se debe estar trabajando sobre un SB y dentro de la población
Poscondiciones:	Se editó Componentes Internos
Requerimientos especiales:	
Curso Normal de los eventos	
Acciones del Actor	**Respuesta del sistema**
1. El Investigador decide editar Operadores Internos y selecciona la población sobre la cual va a trabajar.	1.1 El sistema da la posibilidad de crear un nuevo Operadores Internos de eliminar, mostrar su información y de modificarla; todo esto sobre la población seleccionada

2. El Investigador escoge de las opciones que le brinda el sistema para el trabajo con los componentes Internos.	2.1 El sistema ejecuta la opción tomada por el Investigador: Opciones: A) Crear un Componente Interno: se inserta un nuevo Componente Interno en la población seleccionada y se dibujara dentro de ella. B) Modificar Componente Interno: se modifica la información del Componente Interno y se muestra en la barra de descripción C) Eliminar Componente Interno: se elimina el Componente Interno de población seleccionada. D) Mostrar información del Componente Interno: se muestra información del Componente Interno seleccionado.

Tabla.6 Caso de uso editar Componentes internos

Caso de uso:	Modificar Ecuaciones
Actores:	Investigador (inicia).
Propósito: Permitir la Modificación de las ecuaciones del Sistema Biológico.	

Descripción: Un Investigador desea modificar algunas ecuaciones del SB y el software le da la posibilidad de que modifique en un SB las ecuaciones necesarias.	
Referencias:	R3.1, R3.2, R3.3 , R3.4
Precondiciones:	Se debe estar trabajando sobre un SB
Poscondiciones:	Se editó Operadores Internos
Requerimientos especiales:	
Curso Normal de los eventos	
Acciones del Actor	**Respuesta del sistema**
1. El Investigador decide editar las ecuaciones necesarias	1.1 El sistema da la posibilidad de modificar las ecuaciones asociadas al sistema.
2. El Investigador escoge las ecuaciones a modificar.	2.1 El sistema ejecuta la modificación del Investigador sobre la ecuación.

Tabla.7 Caso de uso Modificar ecuaciones

Caso de uso:	Guardar archivo MathML
Actores:	Investigador (inicia).
Propósito: Salvar un archivo que almacene la información relacionada a los sistemas de ecuaciones de un SB	
Descripción: Un Investigador desea guardar la información del sistema de ecuaciones de un SB que ha creado previamente para en un futuro pueda analizar el comportamiento del mismo, el software le da la posibilidad de que lo salve con dicho formato.	
Referencias:	R 5

Precondiciones:	Se debe estar trabajando sobre un SB
Postcondiciones:	Se guardo un archivo MathML
Requerimientos especiales:	
Curso Normal de los eventos	
Acciones del Actor	**Respuesta del sistema**
1. El Investigador decide guardar las ecuaciones del sistema en un archivo MathML	1.1 El sistema da la posibilidad de guardar las ecuaciones asociadas al sistema en un archivo MathML.
2. El Investigador escoge donde va a guardar el archivo y el nombre del mismo.	2.1 El sistema guarda la información de las ecuaciones del sistema biológico donde el Investigador definió guardarla con el nombre que deseado.

Tabla.8 Caso de uso guardar archivo MathML

Caso de uso:	Guardar archivo SBML
Actores:	Investigador (inicia).
Propósito: Salvar un archivo que almacene la información de un SB en formato SBML	
Descripción: Un Investigador desea guardar la información general del SB que ha creado previamente para en un futuro pueda analizar este archivo con otra aplicación.	
Referencias:	R8
Precondiciones:	Se debe estar trabajando sobre un SB
Poscondiciones:	Se guardo un archivo SBML
Requerimientos especiales:	
Curso Normal de los eventos	
Acciones del Actor	**Respuesta del sistema**

1. El Investigador decide guardar la información del sistema en un archivo SBML	1.1 El sistema da la posibilidad de guardar la información del sistema en un archivo SBML.
2. El Investigador escoge donde va a guardar el archivo y el nombre del mismo.	2.1 El sistema guarda toda la información del SB donde el Investigador definió guardarla con el nombre que deseado.

Tabla.9 Caso de uso guardar archivo SBML

3.6 Conclusiones

En este capítulo se ha determinado el actor que interactúa con el sistema. Se definieron también 20 requerimientos funcionales y los no funcionales. Se definieron 8 Casos de Uso, descritos todos por el diagrama de Casos de Uso y la Descripción de Alto Nivel.

Capitulo 4 Descripción de la solución propuesta

4.1 Introducción

En este capítulo se realiza el diagrama de clases del diseño que es uno de los artefactos principales del flujo de trabajo del RUP, se describen también los principios de diseño a tener en cuenta para el desarrollo de la aplicación y se da una explicación detallada de los ficheros a guardar por la misma.

4.2 Diagrama de clases del diseño

4.2.1 Caso de uso editar sistema biológico anexo 6

4.2.2 Casos de uso editar población anexo 7

4.2.3 Caso de uso editar compuesto químico anexo 8

4.2.4 Caso de uso editar interacción anexo 9

4.2.5 Caso de uso editar Componentes internos anexo 10

4.2.6 Caso de uso modificar ecuaciones anexo 11

4.2.7 Caso de uso guardar archivos MathML anexo 12

4.2.8 Caso de uso guardar archivo SBML 13

4.2.9 Diagrama general de clases del diseño 14

4.3 Principios de diseño

4.3.1 Interfaz al usuario

La interfaz diseñada para el sistema está basada en el estándar de ventanas. El tipo de letra a utilizar será *Arial* de estilo regular y tamaño 8 y el diseño de la aplicación deber ser adecuado para el tipo de usuario (Investigador), logrando la fácil comprensión del mismo en el lenguaje utilizado para las opciones que

brindan. El sistema debe mostrar una barra de menú en la parte superior donde deben estar la mayoría de las opciones de trabajo y debajo una barra de herramienta para el trabajo con los componentes de los sistemas biológicos. Además debe presentar un panel donde se dibuja el sistema biológico en desarrollo y una barra de descripción donde debe mostrar la información del componente de dicho sistema que sea seleccionado. El icono asociado a la aplicación será el logotipo que identifica al sistema. Un Prototipo de interfaz se presenta en el anexo 5

4.3.2 Ayuda

Una parte importante de cualquier sistema lo constituye la ayuda, en el caso nuestro en la barra de menú aparecerá una opción de ayuda con temas relacionado a la utilización de la aplicación y también con teoría sobre los sistemas biológicos con el objetivo de aclarar alguna duda al especialista. En la misma debe aparecer información con imágenes para facilitar la comprensión, también debe presentar la posibilidad de acceder a la misma con solo presionar la tecla *F1*.

4.3.3 Tratamiento de errores

En el diseño de la interfaz se debe tener en cuenta el tratamiento de errores logrando que los mensajes de error que emita el sistema sean de fácil comprensión para el usuario y lo más descriptivos posibles y además debe alertarlos de posibles riesgos de las operaciones que realice. Para los mismos se utilizaran los iconos correspondientes a los estándares, ejemplo:

Figura 5 Icono de información

Figura 6 Icono de error

Figura 7 Icono de confirmación

4.4 Modelos de clases persistentes

Las clases persistentes son almacenadas en ficheros con las estructuras siguientes: Fichero MathML y SBML que se explican a continuación:

Los datos que se almacenarán son los correspondientes al sistemas de ecuaciones diferenciales del Sistema Biológico y será en un archivo xml definidos por un estándar internacional que es el MathML, que esta diseñado para guardar expresiones matemáticas, el cual almacena una lista de cuaciones.

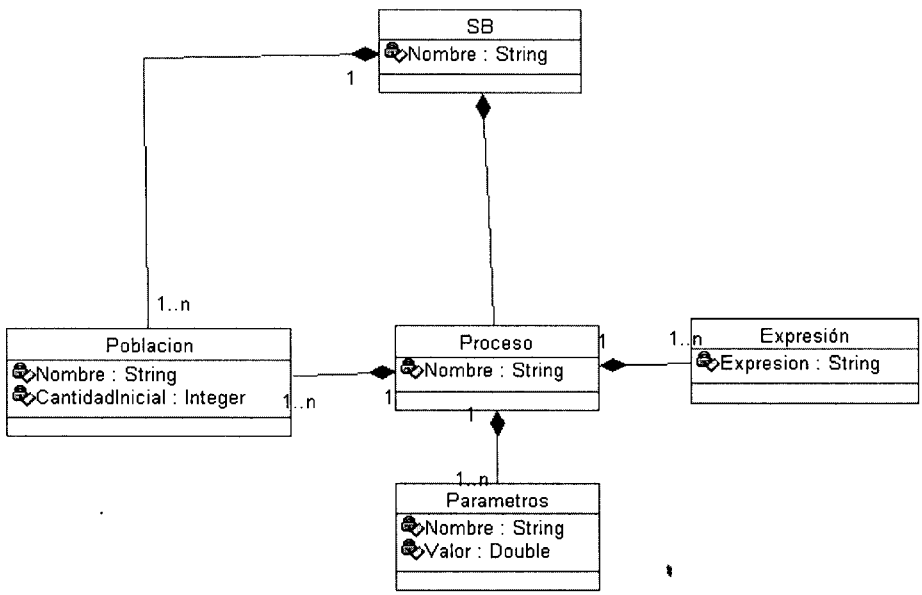


Figura 8 Diagrama de clases persistentes

MathML

El formato MathML (*Mathematical Markup Language*) surge por la necesidad de tener un formato matemático común para aplicaciones con operaciones matemáticas de forma general. Una de las cosas más importantes del lenguaje de marcado es el poder codificar la notación que representa a un objeto matemático y la estructura matemática del mismo. [19]

El mismo esta compuesto por:

1. Elementos de presentación
2. Elementos de contenido
3. Elementos de interfaz

Elementos de presentación:

Corresponden a la notación matemática tradicional, es decir, los tipos básicos de símbolos y estructuras para la construcción de expresiones, a partir de los cuales cualquier parte de la notación tradicional de la matemática puede generarse. Los `símbolos' matemáticos deben ser representados por medio de elementos símbolo de MathML. Los principales son los identificadores (<mi>x</mi>), números (<mn>98</mn> y operadores (<mo>+</mo>). En la notación tradicional matemática las expresiones se construyen a partir de expresiones más pequeñas, y finalmente a partir de simples símbolos, agrupados usando otras estructuras notacionales (operadores, paréntesis, etc). En MathML las expresiones se construyen igual, siendo los esquemas de disposición los que juegan el rol de constructores de expresiones. Dentro de estos tenemos por ejemplo a <mrow>...</mrow>, que dice que se despliegue en una fila la información dentro de los tags, y a <mfrac>...</mfrac>, que sirve para denotar fracciones.

Un ejemplo de esto se puede encontrar en el anexo 2.

Elementos de contenido:

La intención fundamental de la codificación de contenido en MathML es proveer una codificación específica de la estructura matemática subyacente de una expresión, más allá de cualquier representación particular para la misma.

La codificación de contenido de MathML esta basada en el concepto de árbol de expresión. En este árbol las hojas corresponden a objetos matemáticos básicos como son números (<cn>63</cn>), variables (<ci>p</ci>), etc. Los nodos intermedios generalmente representan algún tipo de función (<power/>) u otra construcción matemática que crea un objeto compuesto. Dado que no es la intención de este documento el enseñar a usar MathML, no profundizaremos en las distintas clases de elementos de contenido que existen, pero si aclararemos que el elemento <apply>...</apply> es quizás uno de los más importantes, ya que es el que se usa para aplicar la función a sus argumentos. Nuevamente el orden de los hijos del elemento si importa, pero no esta forzado en la DTD (Data Type Definition) de MathML.

Los elementos de contenido de MathML pueden ser agrupados en las siguientes categorías según su uso:

- contenedores
- operadores y funciones
- calificadores
- relaciones
- condiciones
- mapeos semánticos
- constantes y símbolos

Por ejemplo tenemos el anexo 3.

Elementos de interfaz:

Los elementos de interfaz son aquellos relacionados en la generación y representación de MathML, siendo particularmente importantes los que respectan a insertar MathML en HTML y XHTML.

SBML

El formato SBML (*Systems Biology Markup Language*) es un lenguaje de marcado en beneficio de la biología de sistemas. Está basado en XML y creado para describir sistemas biológicos. Es un lenguaje orientado a la representación de las redes bioquímicas comunes, incluyendo por ejemplo la célula. Señala caminos, caminos metabólicos, reacciones bioquímicas, la regulación de genes, y muchas otras. Las motivaciones para desarrollar SBML provinieron de la carencia actual de intercambiar modelos biológicos entre las herramientas bioquímicas existentes. [20,21]

El documento SBML debe estar formado por la estructura que se muestra en el anexo 4.

En nuestro sistema solo de deben almacenar la lista de ecuaciones, lista de poblaciones, lista de parámetros y la lista de procesos

En el anexo 1 se muestra un ejemplo completo de la utilización de estos lenguajes de marcado.

4.5 Diagrama de Despliegue

El diagrama de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Para aplicaciones como la que se propone, que se ejecuta en una sola máquina y todos los dispositivos con que se relaciona son los estándares (teclado, mouse, impresora) el diagrama de despliegue está constituido por un nodo. Se ha considerado que la representación del nodo en este tópico no reportaría mucho. [16]

4.6 Conclusiones

Como resultado en este capítulo, han sido definidas las clases del diseño y han sido representadas sus relaciones en el Diagrama de Clases del Diseño, así como las clases persistente y una detallada explicación de los formatos en los que se va a almacenar la información necesaria. Se han descritos los principios del diseño seguidos en el sistema propuesto, concepción general de la ayuda y el tratamiento de errores.

5. Estudio de factibilidad

5.1 Introducción

La estimación de esfuerzo es un factor muy importante en el proceso de desarrollo de software, debido a que esta nos da una medida del esfuerzo a realizar en el desarrollo del sistema teniendo en cuenta la complejidad del mismo. Una correcta valoración de este proceso es un aspecto determinante, debido a que puede evitar pérdidas económicas y de los clientes, así como dar prestigio a los desarrolladores por su correcta estimación. [22, 23]

En el este capitulo se describe y se aplica la técnica basada en casos de uso para le estimación de esfuerzo.

5.2 Planificación basada en casos de uso

La estimación basada en caso de uso es un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores. A continuación se describen los pasos a seguir. [22]

El primer paso consiste en el cálculo de los Puntos de Casos de Uso sin ajustar; y esta descrito por la siguiente ecuación

UUCP = UAW + UUCW

Donde,

- **UUCP**: Puntos de Casos de Uso sin ajustar
- **UAW**: Factor de Peso de los Actores sin ajustar
- **UUCW**: Factor de Peso de los Casos de Uso sin ajustar

UAW: Este valor se calcula mediante un análisis de la cantidad de Actores presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Actores se establece teniendo en cuenta en primer lugar si se trata de una persona o de otro sistema, y en segundo lugar, la forma en la que el actor interactúa con el sistema. Los criterios se muestran en la siguiente tabla:

Tipo de actor	Descripción	Factor de peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Application Programming Interface)	1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto	2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica	3

Tabla.10 Tipo de actor

Se tiene a un usuario como actor por lo que se le da peso 3 y el valor de UAW

UAW=CantU * FactorPeso

UAW =1*3 = 3

(UUCW)

Este valor se calcula mediante un análisis de la cantidad de Casos de Uso presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Casos de Uso se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo, donde una transacción se entiende como una secuencia de actividades atómica, es decir, se efectúa la secuencia de Actividades completas, o no se efectúa ninguna de las actividades de la secuencia. Los criterios se muestran en la siguiente tabla:

Tipo de caso de uso	Factor de peso asociado
Simple	5
Medio	10
Complejo	15

Tabla.11 Tipo de caso de uso

Caso de Uso	Factor de peso asociado
Gestionar Sistema Biológico	5
Editar Población	10
Editar Compuesto Químico	10
Editar Interacciones	10
Editar componentes internos	10
Guardar MathML	5
Guardar SBML	5
Modificar Ecuaciones	5

Tabla.12 Peso de los casos de uso

La formula para calcular este factor es:

UUCW = Pesoi * CantCU= 5*4+10*4=60

Finalmente los Puntos de Casos de Uso sin ajustar tiene el siguiente valor

UUCP =60+3 = 63

El segundo paso es determinar los puntos de caso de uso ajustados que están descritos por la siguiente ecuación:

UCP = UUCP * TCF * EF

donde,

UCP: Puntos de Casos de Uso ajustados

UUCP: Puntos de Casos de Uso sin ajustar

TCF: Factor de complejidad técnica

EF: Factor de ambiente

TCF

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un

aporte muy importante. En la siguiente tabla se muestra el significado y el peso de cada uno de éstos factores:

El Factor de complejidad técnica se calcula mediante la siguiente ecuación:

TCF = 0.6 + 0.01 * Σ (Pesoi x Valor asignadoi)

Nombre Factor	Descripción	Peso	Valor	Factor	Comentario
T1	Sistema distribuido	2	0	0	Distribuido
T2	Tiempo de repuesta	1	3	3	Tiempo normal
T3	Eficiencia para usuario final	1	1	1	Eficaz
T4	Procesamiento interno complejo	1	3	3	Normal
T5	Reusabilidad	1	5	5	Reusabilidad alta
T6	Facilidad de instalación	0,5	1	0,5	Fácil a instalar
T7	Facilidad de uso	0,5	3	1.5	Normal
T8	Portabilidad	2	1	2	Fácil a transportar
T9	Facilidad de cambio	1	1	1	Fácil de cambiar
T10	Concurrencia	1	1	1	Concurrencia baja
T11	Objetivos especiales de seguridad	1	2	2	Seguridad baja
T12	Prever accesos directos a terceras partes	1	1	1	Bajo acceso
T13	Facilidades especiales de entrenamiento a usuario	1	1	1	Fácil de usar
Total				**22**	

Tabla.13 Peso de los factores de complejidad

TCf = 0.6+0.01* 22 =0.82

EF

Las habilidades y el entrenamiento del grupo involucrado en el desarrollo tienen un gran impacto en las estimaciones de tiempo. Estos factores son los que se contemplan en el cálculo del Factor de ambiente. El cálculo del mismo es similar al cálculo del Factor de complejidad técnica, anteriormente descrito, es decir, se trata de un conjunto de factores que se cuantifican con valores de 0 a 5. En la siguiente tabla se muestra el significado y el peso de cada uno de éstos factores. Y esta determinado por la siguiente ecuación:

EF =1.4 - 0.03 * Σ (Pesoi x Valor asignadoi)

Factor	Descripción	Peso	Valor	Factor	Comentario
E1	Familiaridad con el modelo del proyecto utilizado	1,5	3	4,5	Familiaridad media
E2	Experiencia en la aplicación	0,5	3	1,5	Experiencia Media
E3	Experiencia en O.O	1	5	5	Amplia experiencia
E4	Capacidad del analista líder	0,5	3	1,5	El analista es un estudiante de 5 año Cujae
E5	Motivación	1	5	5	Alta motivación
E6	Estabilidad de los requerimientos	2	5	10	Estables
E7	Personal media jornada	-1	3	-3	Media
E8	Dificultad del lenguaje	-1	3	-3	Se usara Java
				21,5	

Tabla.14 Peso de los factores de esfuerzo

EF =1.4 - 0.03 * 21,5 =2.405

UCP=63+0,82+2,405 =66.225

El tercer paso es calcular el esfuerzo en horas-hombre que la ecuación que lo describe es la siguiente:

E = UCP * CF

Donde,

E: esfuerzo estimado en horas-hombre

UCP: Puntos de Casos de Uso ajustados

CF: factor de conversión

Considerando CF =20 horas-hombre

E=66.225*20=1324,5 horas-hombre

Este es el esfuerzo total para el proyecto, pero ahora se desglosará teniendo en cuanta las siguientes tareas.

Et: esfuerzo total

Et = (100*E)/40=3311,25 horas-hombre

Tareas	%	Horas-Hombre
Análisis Diseño	30	993,375
Programación	40	1324,5
Pruebas	15	496,6875
Sobrecarga(otras actividades)	15	496,6875
Total	100	3311,25

Tabla.15 Distribución de hora- hombres

TDES (total) = E (total) / CH (hombres) donde:

TDES: Tiempo de Desarrollo

E: Esfuerzo

CH: Cantidad de Hombres

El desarrollo de este proyecto es llevado a cabo por cinco personas.

TDES = 3311,25/5

TDES = 662,25 horas

C (total) = E (total en HH) x CHH

C (total): Cálculo del costo total a partir del esfuerzo en horas -hombres

CHH: Costo por Hombre Horas

CHH = K x THP

K: Coeficiente que tiene en cuenta los costos indirectos (1,5 y 2,0)

THP: Tarifa Horaria Promedio

El salario promedio de las personas que trabajan en el proyecto dividido entre 160 horas

Se asume como salario promedio mensual $250

E (Total en HH) = 3311,25

THP = 250/160 = 1,5625

Se asume que el coeficiente (k) que tiene en cuenta los costos indirectos es de 1.5

C (total) = E (total en HH) x K x THP

C (total) = 3311,25x 1.5 x1, 5625

C (total) = $7760,7421

5.3 Beneficios tangibles e intangibles

Los beneficios intangibles son muy importantes y pueden tener implicaciones de relevancia para el negocio, en su relación con personas tanto ajenas como propias de la organización. [24] Ejemplo de beneficios intangibles serían:

- La mejora del proceso de toma de decisiones
- El incremento de precisión
- El incremento de la satisfacción de los empleados al eliminar tareas de naturaleza tediosa.

Los beneficios intangibles que se obtendrán con el desarrollo del sistema propuesto son:

- Permitirá modelar con más precisión sistemas biológicos con el objetivo de analizar su comportamiento.

- Fomentar este tipo de investigación en los diferentes centros del polo científico del país

- También los científicos se sentirán más cómodos con este sistema a la hora de modelar.

Los beneficios tangibles son las ventajas económicas cuantificables que obtiene la organización a través del uso del sistema de información. [24] Ejemplo de beneficios tangibles serían:

- El incremento en la velocidad de proceso
- La obtención de información con mayor puntualidad que en el pasado
- Aprovechar el mayor poder de cálculo de las computadoras
- Reducir el tiempo requerido por los empleados para concluir una tarea especifica

Aunque la medición no siempre es fácil los beneficios tangibles pueden estimarse en términos de pesos, recursos o tiempo ahorrados.

Los beneficios tangibles que se obtendrán en este proyecto son:

- Disminuirá considerablemente el tiempo del proceso de modelados de sistemas biológicos
- A la postre influirá en la disminución de los costos de producción de biofármacos, y facilitará el almacenamiento de información biológica.

5.4 Análisis de costos y beneficios

Al desarrollo de todo producto informático va asociado un costo, el justificarlo depende de los beneficios tangibles e intangibles que produce.

La utilización de este nuevo software para desarrollar el análisis de sistemas biológicos puede traer muchos beneficio económicos debido a que los costos de producción de medicamentos por el método tradicional son muy elevados, llegando a la suma de 500 millones de dólares, y el costo de esta aplicación es despreciable al compararla con la suma mostrada anteriormente, por lo que se considera factible la implementación de la misma.

5.5 Conclusiones

En este capítulo se hizo el análisis de factibilidad del desarrollo del sistema y se determinó que se necesitan 662,25 horas con cinco hombres trabajando y que el costo total de la aplicación asciende a $ 7760,7421 teniendo en cuenta que el salario promedio es de $ 250. Se describieron además los beneficios tangibles e intangibles de la implementación del software.

Conclusiones

1. Se ha realizado una investigación que abarca lo referente a los sistemas biológicos y las interacciones dentro de los mismos, concluyendo cuales son los componentes necesarios para la implementación de un software que permita modelar dichos sistemas.
2. Se ha realizado el análisis y diseño de una herramienta que permitirá modelar y almacenar la información de un sistemas biológico, haciendo además un análisis de la factibilidad de la implementación de la misma.

Recomendaciones

Se Recomienda:

1. Poner a prueba el sistema un periodo de tiempo considerable para comprobar su correcta implementación y correspondencia con lo requerido.
2. Elaborar una funcionalidad que permita almacenar un fichero con extensión propia para poder trabajar con dichos ficheros en el futuro y así ampliar la aplicabilidad del sistema.
3. Incorporar a esta aplicación un módulo de cálculos que permita además de modelar, realizar simulaciones de los sistemas biológicos en estudio

Referencias bibliográficas

1. http://es.wikipedia.org/wiki/Edward_Jenner febrero 2006
2. http://es.wikipedia.org/wiki/Cronolog%C3%ADa_de_las_vacunas febrero 2006
3. http://www.hygiene-educ.com/sp/profs/histoire/guide/presentation3.htm
4. Osorio Ramírez, Karen: *Plataforma Computacional para el Desarrollo de la Biología de Sistemas*, Tesis de grado en opción al título de Licenciado en Ciencia de la Computación, Universidad de la Habana, julio 2004.
5. http://www.cim.sld.cu/ febrero 2006
6. http://www.biouml.org/ marzo 2006
7. http://www.esse.ou.edu/models/stella_using.html marzo 2006
8. http://www.clikear.com/manuales/uml/introduccion.asp marzo 2006
9. www.dcc.uchile.cl/~luguerre/cc61j/recursos/clase2.ppt marzo 2006
10. http://www.willydev.net/descargas/articulos/general/cualxpfddrup.PDF marzo 2006
11. www.vico.org/TallerRationalRose.pdf marzo 2006
12. http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Delphi marzo 2006
13. http://es.wikipedia.org/wiki/Borland marzo 2006
14. http://www.agapea.com/Eclipse-3-para-desarrolladores-Java-n228608i.htm marzo 2006
15. www.fing.edu.uy/inco/grupos/coal/investigacion/proyectos/lead/docs/IntroduccionEclipse.pdf marzo 2006
16. Jacobson, Ivar y Booch, Grady y Rumbaugh, James. *El proceso unificado de software*. Primera edición. Pearson Educación, S.A. 2000.

17. http://www.creangel.com/uml/casouso.php marzo 2006
18. www.unab.edu.co/editorialunab/revistas/rcc/pdfs/r11_art4_c.pdf marzo2006
19. http://www.tejedoresdelweb.com/307/article-5809.html marzo 2006
20. http://www.xml.com/pub/r/1072 marzo 2006
21. http://sbml.org/specifications/sbml-level-2-v1.pdf marzo 2006
22. www.itba.edu.ar/capis/rtis/rtis-6-1/estimaci%F3n-del-esfuerzo-basada-en-casos-de-usos.pdf abril 2006
23. http://www.methodsandtools.com/archive/archive.php?id=25 abril 2006
24. http://www.monografias.com/trabajos24/seleccion-plataforma/seleccion-plataforma.shtml mayo 2006
25. http://www.mathworks.com/products/simbiology/description1.html mayo 2006

Glosario de términos

SIDA: Síndrome de Inmunodeficiencia Adquirida.

CIM: Centro de Inmunología Molecular de la Habana.

In silico: investigaciones donde se aplican los implementos computacionales

SB: Sistema Biológico.

RUP: Proceso Unificado (Rational Unified Process) metodología para el desarrollo de sistemas informáticos, dirigidos por casos de uso.

XP: Metodología para el desarrollo de sistemas informáticos que se basa en el intercambio constante con el cliente.

UML: Lenguaje de Modelamiento Unificado (Unified model languaje), Lenguaje gráfico que brinda un vocabulario y reglas para especificar, construir, visualizar y documentar los artefactos de un sistema utilizando el enfoque orientado a objetos.

Plugin: Función o utilidad específica, generalmente muy específica. Se readiciona a algún programa para ser ejecutado. Los plugins típicos tienen la función de reproducir determinados formatos de gráficos, reproducir datos multimedia, etc.

POO: Programación Orientada a Objeto, estilo de programación muy difundido en la actualidad por las ventajas que tiene.

Open-Source: Código abierto.

C++: es un lenguaje de programación, diseñado a mediados de los ochenta, por Bjarne Stroustrup, como extensión del lenguaje de programación C.

Java: es Lenguaje de programación, de alto nivel, orientado a objetos y desarrollada por Sun Microsystems.

CI : Componentes Internos.

XML: (e**X**tensible **M**arkup **L**anguage, 'lenguaje de marcado extensible') es un lenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

MathML: (*Mathematical Markup Language*) es un lenguaje de marcado basado en XML, cuyo objetivo es expresar notación matemática de forma que distintas máquinas puedan entenderla.

SBML: (*Systems Biology Markup Language*) es un lenguaje de marco en beneficio de la biología de sistemas esta basado en XML y creado para describir simulaciones en biología de sistemas.

DTD: (Document type definition) La DTD define los tipos de elementos, atributos y entidades permitidas, y puede expresar algunas limitaciones para combinarlos. Los documentos XML que se ajustan a su DTD se denominan válidos.

Anexo 1 Formato de archivo SBML

<model id="My_Model">
<listOfFunctionDefinitions>
...
</listOfFunctionDefintions>
<listOfUnitDefinitions>
...
</listOfUnitDefinitions>
<listOfCompartments>
...
</listOfCompartments>
<listOfSpecies>
...
</listOfSpecies>
<listOfParameters>
...
</listOfParameters>
<listOfRules>
...
</listOfRules>
<listOfReactions>
...
</listOfReactions>
<listOfEvents>
...
</listOfEvents>
</model>

Anexo 2 MathML, elementos de representación

$(x+y)^2$

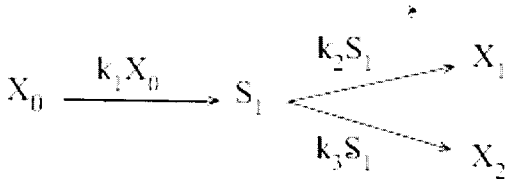
```
<math>
<msup>
<mrow>
<mo> ( </mo>
<mrow>
  <mi> x </mi>
<mo> + </mo>
<mi> y </mi>
</mrow>
<mo> ) </mo>
</mrow>
<mn> 2 </mn>
</msup>
</math>
```

Anexo 3 MathML, elementos de contenido

x+y

```
<math>
<apply>
<power/>
<apply>
  <plus/>
  <ci>x</ci>
 <ci>y</ci>
</apply>
</apply>
</math>
```

Anexo 4 Fichero SBML



```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2" level="2" version="1">
<model id="Branch">
<notes>
<body xmlns="http://www.w3.org/1999/xhtml">
<p>Simple branch system.</p>
<p>The reaction looks like this:</p>
<p>reaction-1: X0 -> S1; k1*X0;</p>
<p>reaction-2: S1 -> X1; k2*S1;</p>
<p>reaction-3: S1 -> X2; k3*S1;</p>
</body>
</notes>
<listOfCompartments>
<compartment id="compartmentOne" size="1"/>
</listOfCompartments>
<listOfSpecies>
<species id="S1" initialConcentration="0" compartment="compartmentOne"
boundaryCondition="false"/>
<species id="X0" initialConcentration="0" compartment="compartmentOne"
boundaryCondition="true"/>
<species id="X1" initialConcentration="0" compartment="compartmentOne"
boundaryCondition="true"/>
<species id="X2" initialConcentration="0" compartment="compartmentOne"
boundaryCondition="true"/>
</listOfSpecies>
```

<listOfReactions>
<reaction id="reaction_1" reversible="false">
<listOfReactants>
<speciesReference species="X0"/>
</listOfReactants>
<listOfProducts>
<speciesReference species="S1"/>
</listOfProducts>
<kineticLaw>
<math xmlns="http://www.w3.org/1998/Math/MathML">
<apply>
<times/>
<ci> k1 </ci>
<ci> X0 </ci>
</apply>
</math>
<listOfParameters>
<parameter id="k1" value="0"/>
</listOfParameters>
</kineticLaw>
</reaction>
<reaction id="reaction_2" reversible="false">
<listOfReactants>
<speciesReference species="S1"/>
</listOfReactants>
<listOfProducts>
33
<speciesReference species="X1"/>
</listOfProducts>
<kineticLaw>
<math xmlns="http://www.w3.org/1998/Math/MathML">

```
<apply>
<times/>
<ci> k2 </ci>
<ci> S1 </ci>
</apply>
</math>
<listOfParameters>
<parameter id="k2" value="0"/>
</listOfParameters>
</kineticLaw>
</reaction>
<reaction id="reaction_3" reversible="false">
<listOfReactants>
<speciesReference species="S1"/>
</listOfReactants>
<listOfProducts>
<speciesReference species="X2"/>
</listOfProducts>
<kineticLaw>
<math xmlns="http://www.w3.org/1998/Math/MathML">
<apply>
<times/>
<ci> k3 </ci>
<ci> S1 </ci>
</apply>
</math>
<listOfParameters>
<parameter id="k3" value="0"/>
</listOfParameters>
</kineticLaw>
</reaction>
```

```
</listOfReactions>
</model>
</sbml>
```

Anexo5 Prototipo de interfaz

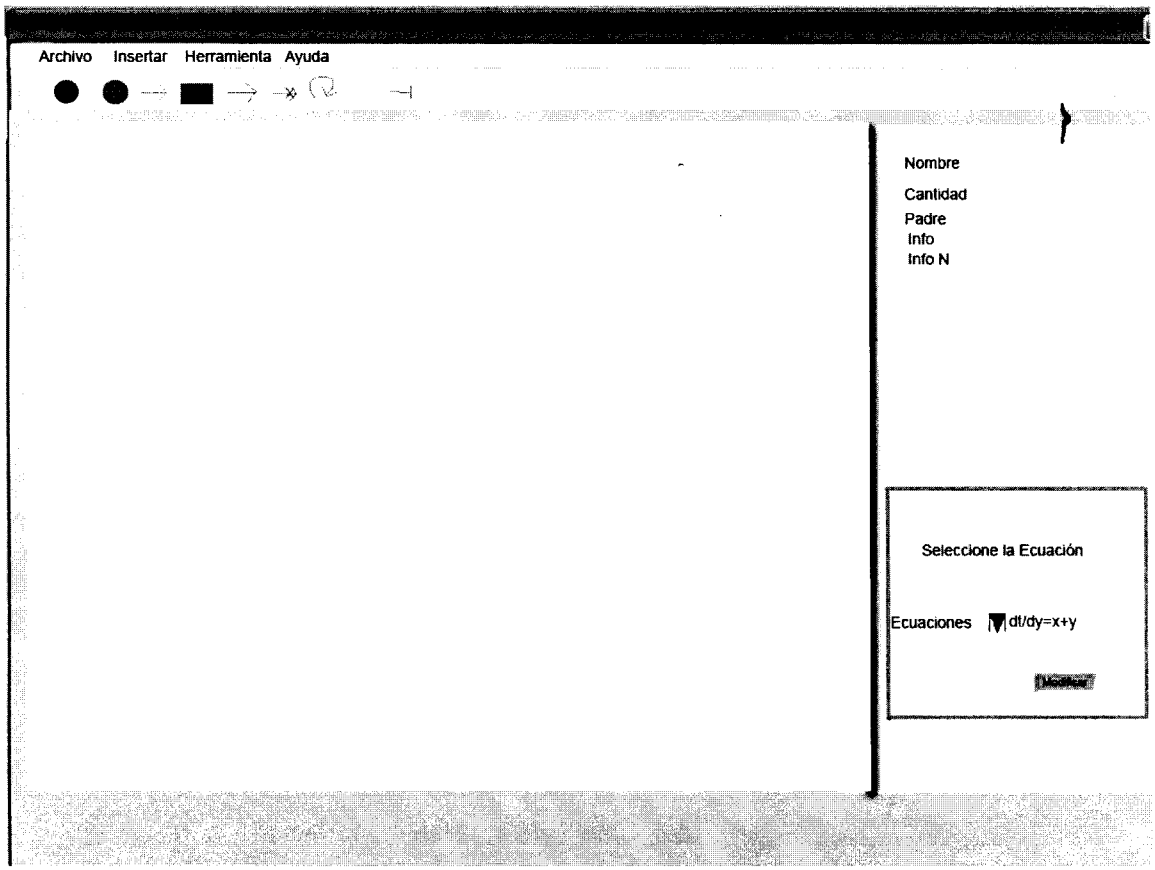


Figura 9 Prototipo e interfaz

Anexo 6 Diagrama de clases del Caso de uso editar sistema biológico

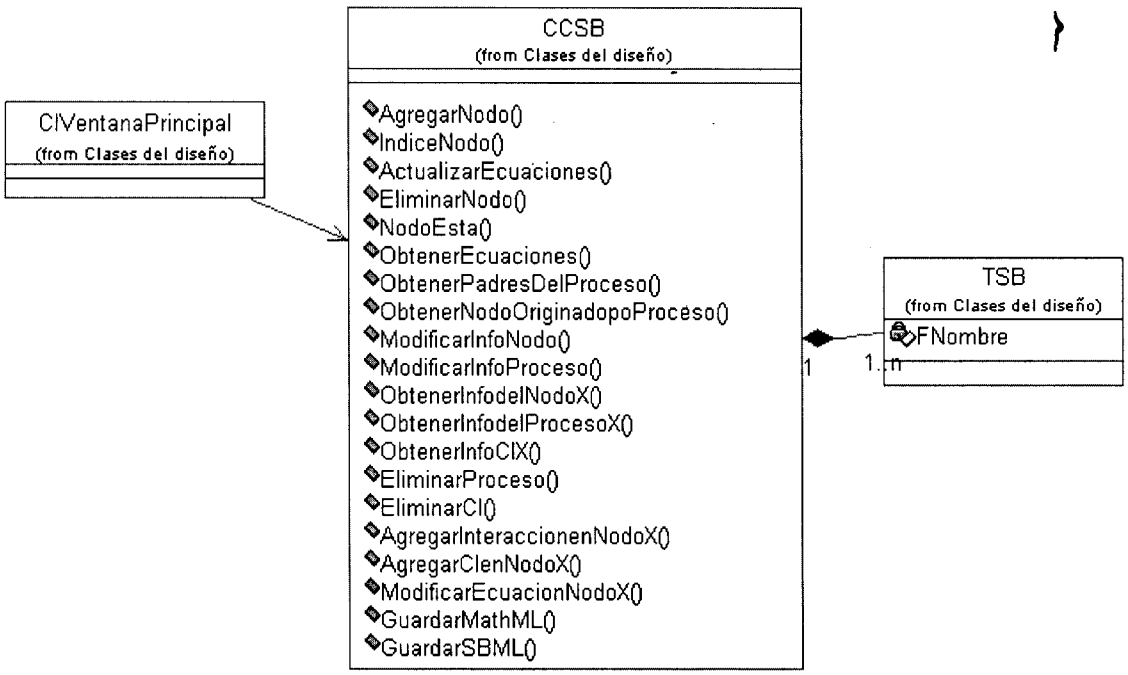


Figura 10 Diagrama de clases del Caso de uso editar sistema biológico

Anexo 7 Diagrama de clases del Caso de uso Editar Población

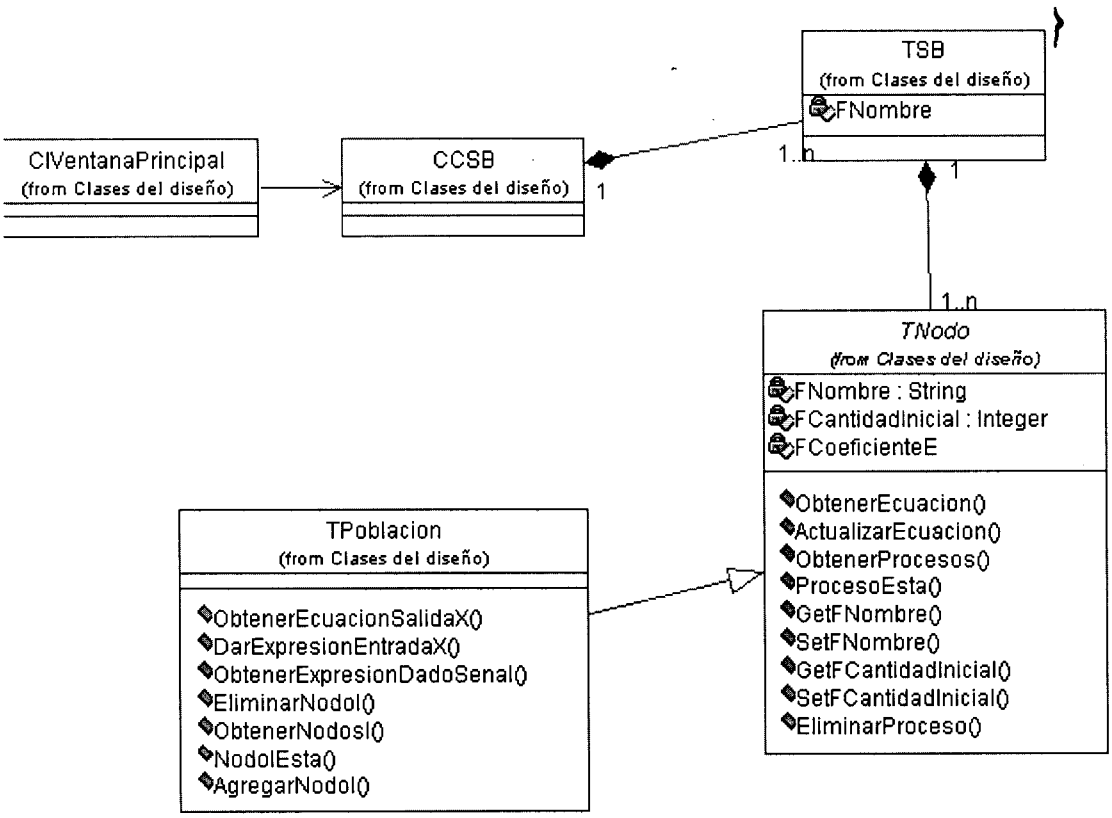


Figura 11 Diagrama de clases del Caso de uso Editar Población

Anexo 8 Diagrama de clases del Caso de uso Editar Compuesto Químico

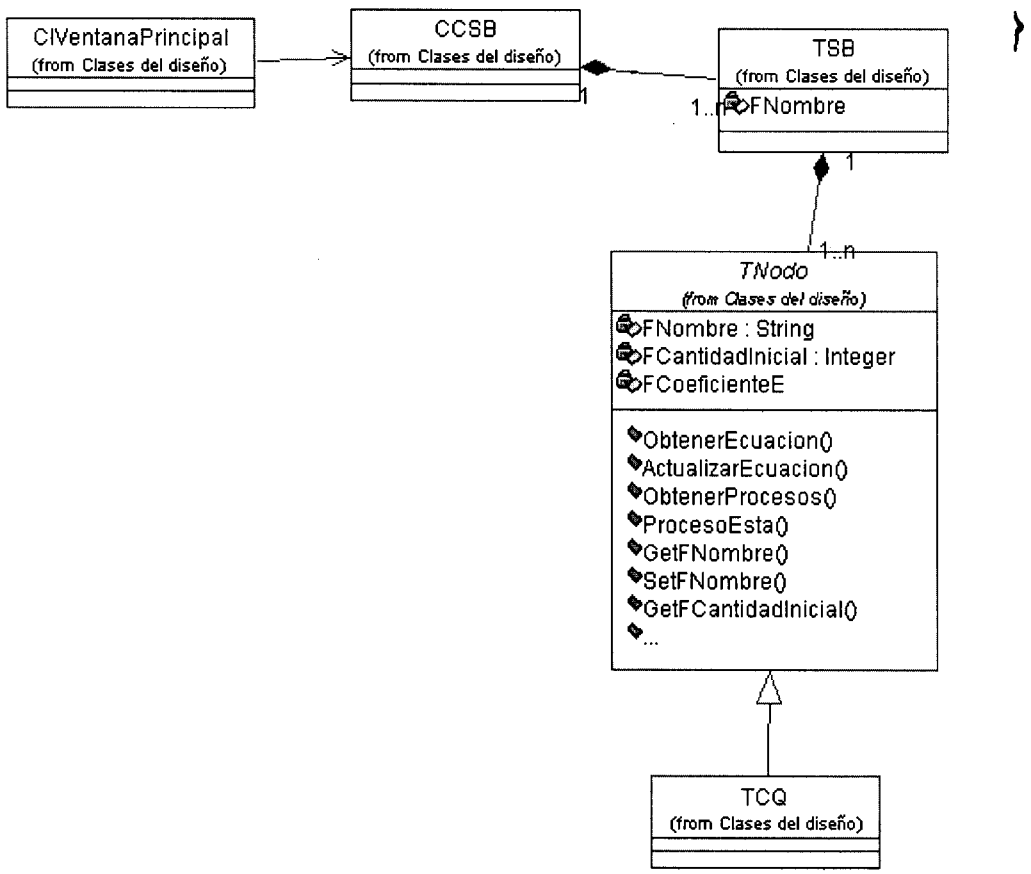


Figura 12 Diagrama de clases del Caso de uso Editar Compuesto Químico

Anexo 9 Diagrama de clases del Caso de uso Editar Interacción

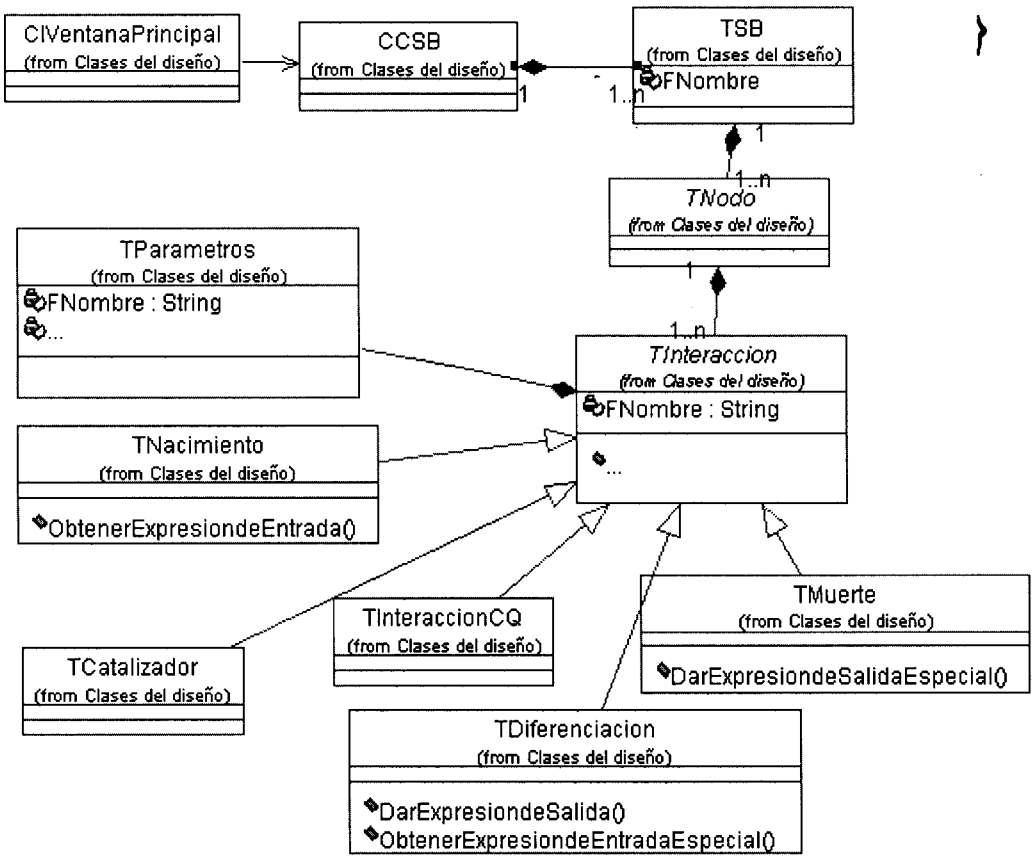


Figura 13 Diagrama de clases del Caso de uso Editar Interacción

Anexo10 Diagrama de clases del Caso de uso Editar Componente Interno

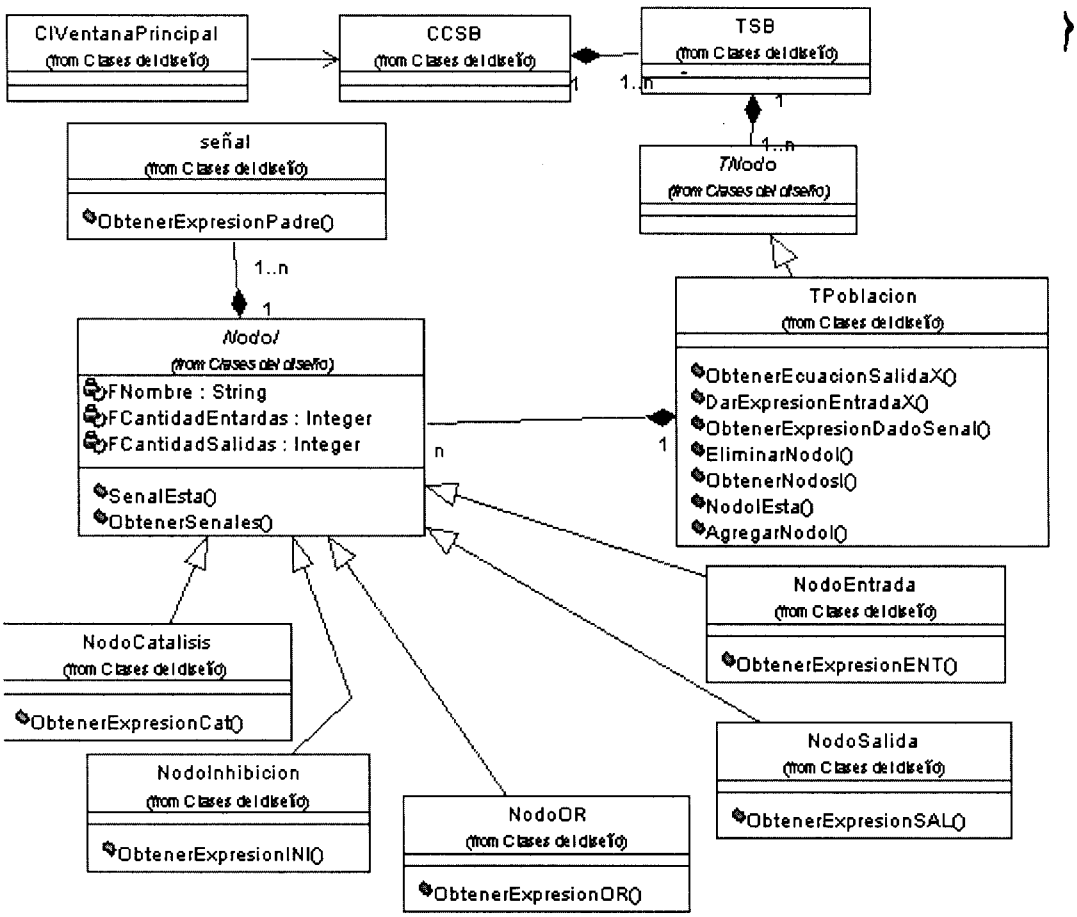


Figura 14 Diagrama de clases del Caso de uso Editar Componentes Internos

Anexo11 Diagrama de clases del Caso de uso Modificar Ecuaciones

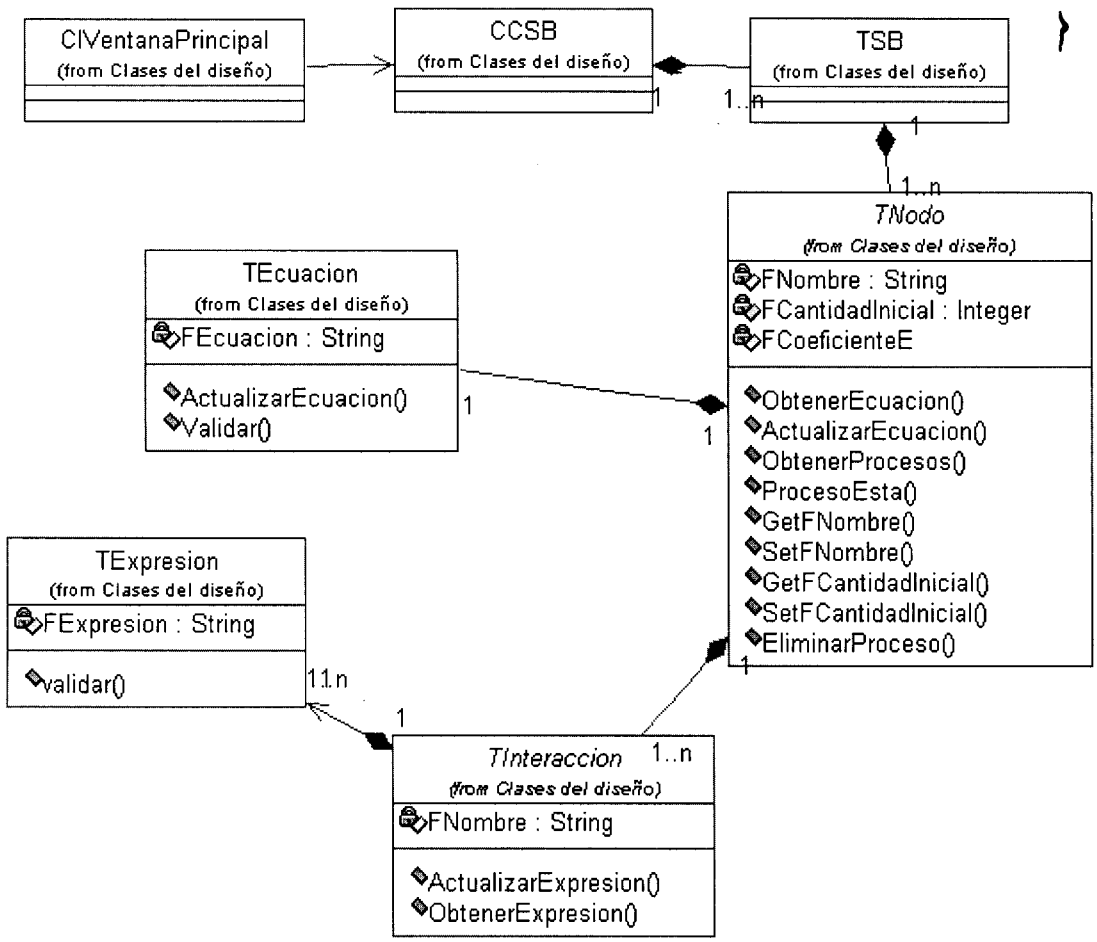


Figura 15 Diagrama de clases del Caso de uso Modificar Ecuaciones

Anexo 12 Diagrama de clases del Caso de uso Guardar Archivo MathML

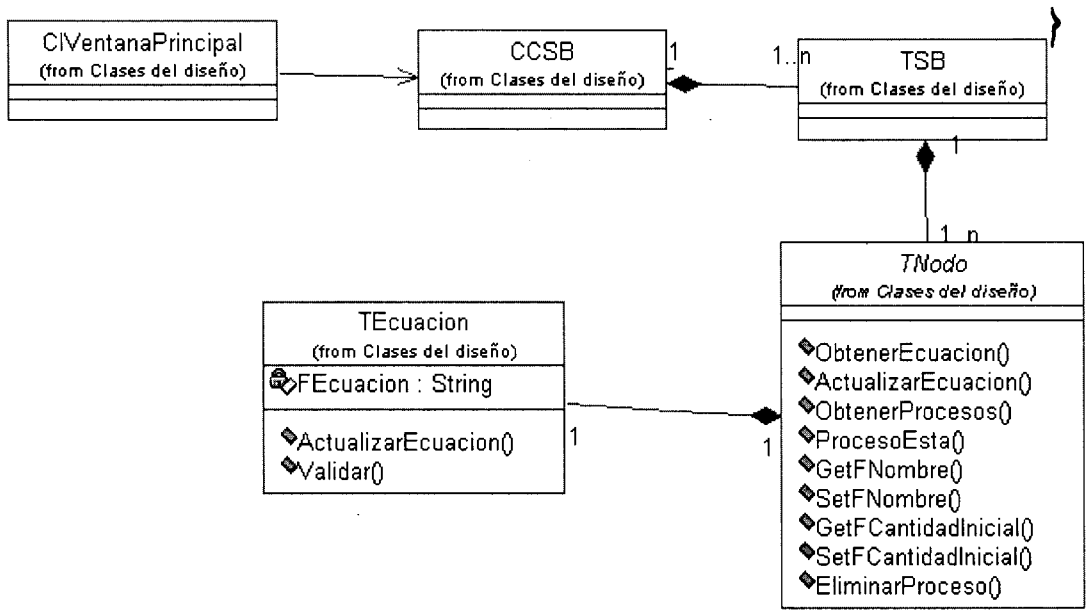


Figura 16 Diagrama de clases del Caso de uso Guardar archive MathML

Anexo 13 Diagrama de clases del Caso de uso Guardar Archivo SBML

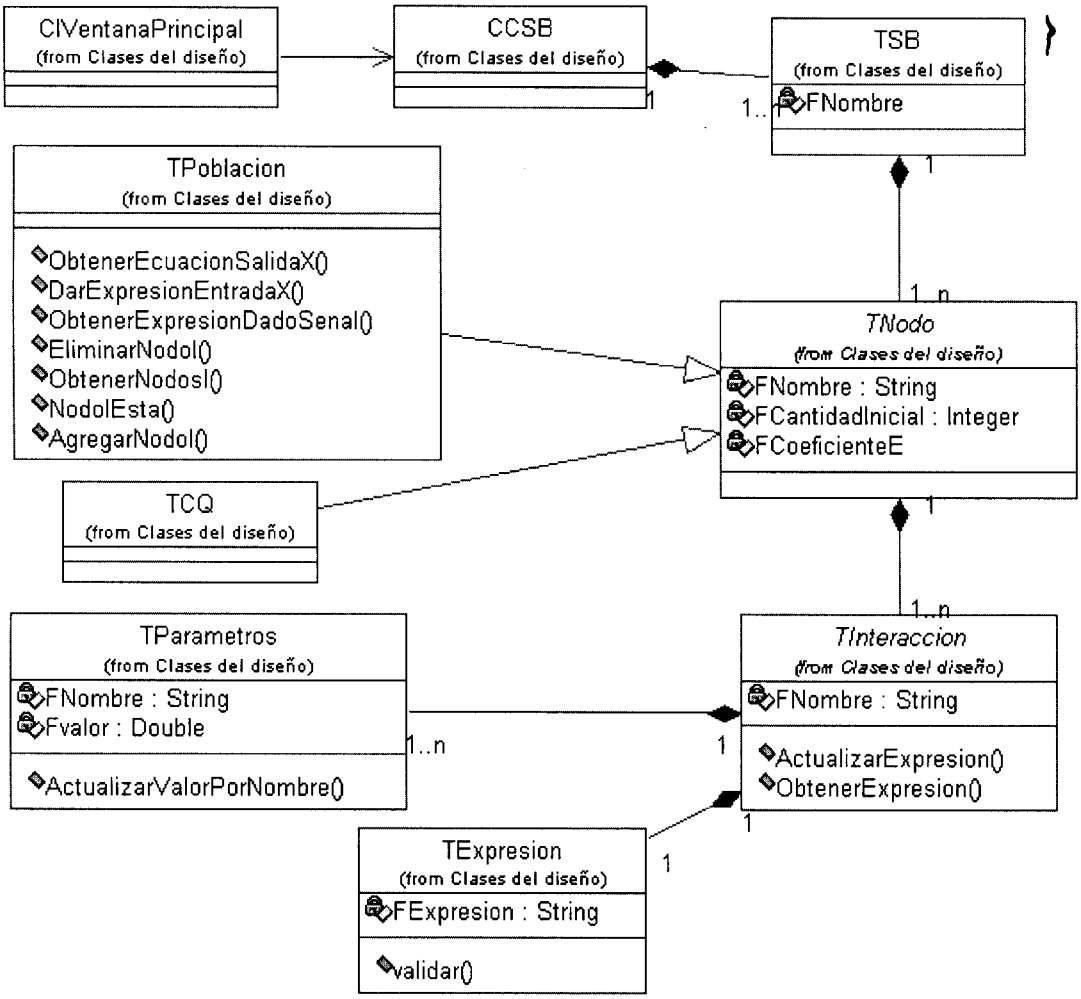


Figura 17 Diagrama de clases del Caso de uso Guardar Archivo SBLM

Anexo 14 Diagrama de clases general del diseño

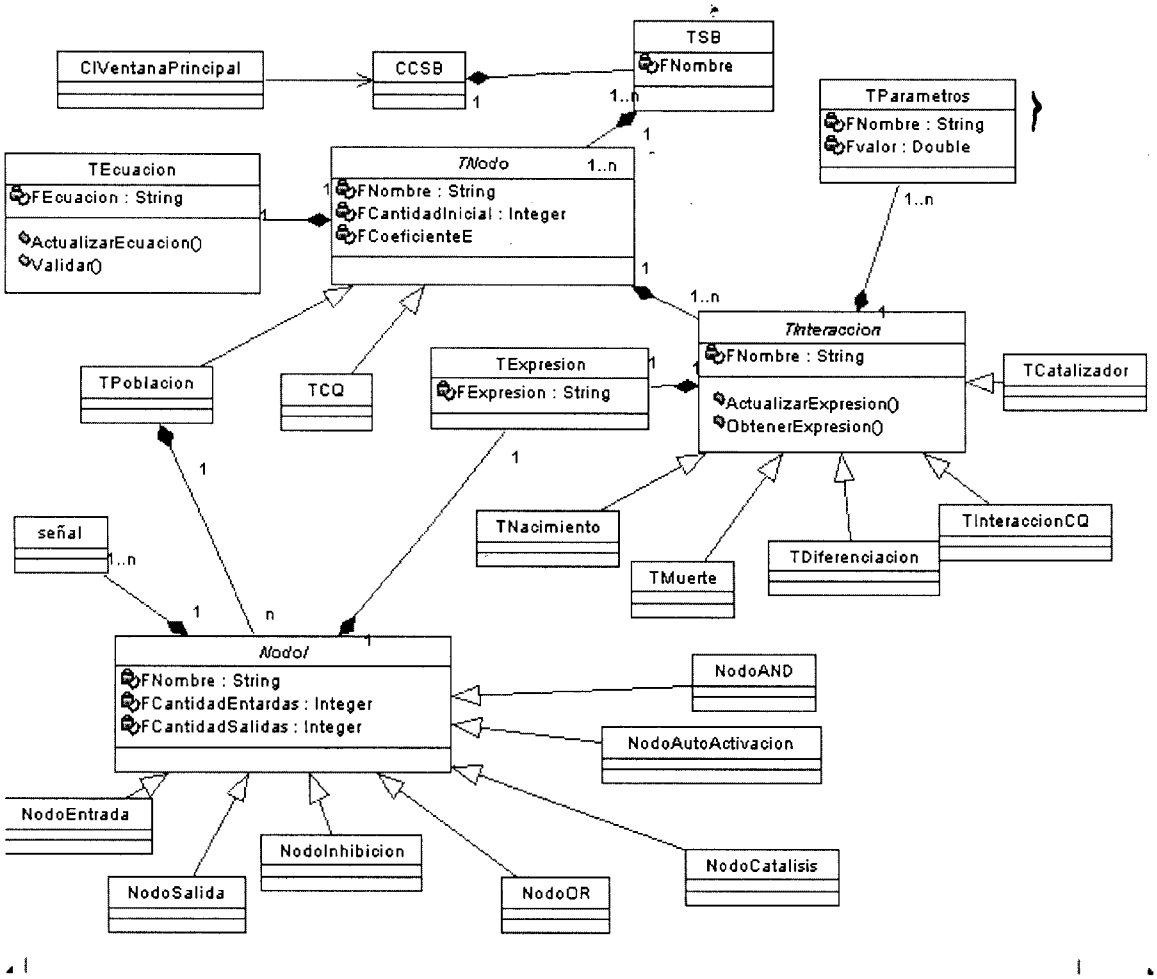


Figura 18 Diagrama de clases general del diseño