Universidad de las Ciencias Informáticas

Facultad 6

Trabajo de diploma para optar por el título de Ingeniero Informático

Análisis y diseño de un simulador para el sistema inmune

Autores

Yeniley Matos

Jorge A González

Tutores

Lic. Noel Moreno Lemus

Dr. Kalet León

Ciudad de La Habana, Junio 2007

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 de _____ del año _____.

<nombre autor> <nombre tutor>

_____ _____

Firma del Autor Firma del Tutor

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA.

El Trabajo de Diploma titulado “Sistema de Manejo de Datos de Ensayos Clínicos: Módulo de Administración”, fue desarrollado en la Universidad de las Ciencias Informáticas. Esta entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface:

Totalmente ____
Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes:

_____
_____
_____
_____
_____
_____
_____

Y para que así conste, se firma la presente a los ___ días del mes de _____ del año _____

_____ _____
Representante de la entidad Cargo

_____ _____
Firma Cuño

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Sistema de Manejo de Datos de Ensayos Clínicos: Módulo de Administración
Autores: Linet Cobo Barreras y Yordany Lima Acosta

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

<Aquí el tutor debe expresar cualitativamente su opinión y medir (usando la escala: muy alta, alta, adecuada) entre otras las cualidades siguientes:
- Independencia
- Originalidad
- Creatividad
- Laboriosidad
- Responsabilidad >

< Además, debe evaluar la calidad científico-técnica del trabajo realizado (resultados y documento) y expresar su opinión sobre el valor de los resultados obtenidos (aplicación y beneficios) >

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de <nota>. <Además, si considera que los resultados poseen valor para ser publicados, debe expresarlo también>

_____ _____

Firma

Fecha

“Debes amar la arcilla que está en tus manos,
debes amar su arena hasta la locura”
Silvio Rodriguez.

Dedicatoria

A mis hermanos Helaine y Dayrel porque esta es nuestra meta común, y a mi abuelo Raúl por ser excelente padre de familia y esperar tanto este momento.

Yeni

Este trabajo se lo dedico a todos aquellos profesores, amigos y compañeros que a lo largo de estos cinco años han compartido junto a mí los momentos vividos en nuestra universidad y en especial a mis padres y demás familiares que me han apoyado en todo para poder llegar a esta culminación exitosa de estudios universitarios

Jorge Ariel

Agradecimientos

Agradezco a la “mamita” de la “hijita” por ser tan dulce, tan buena y apoyarme siempre

A mi papá por enseñarme en mis años de infancia a procurar y valorar el conocimiento, el amor y la mistad

A mi hermanita Arlenys, que me enseñó de pequeña a ver la vida de otra forma.

A mi familia toda -cardenense y guantanamera- por tanto apoyo y cariño.

A Andresej por enseñarme a pensar y motivarme siempre.

A Sonia, Dictinio, Brito, Tony, Héctor, Mariño y Noel por contribuir en mi formación.

A Jorge y Marixa, por su apoyo y cariño.

A mis amigos todos que hemos soñado juntos este momento y que hemos formado una familia en la UCI.

A mis vecinos que tanto han esperado de mí y han sido como familia siempre.

A mis amigos de infancia y a los que aún lejos, me apoyan y me quieren

Yeni

Mi agradecimiento especial para mi familia y amistades todas.

Le agradezco mucho a Héctor y su familia que tanto me han apoyado durante mi carrera.

A quienes me han apoyado durante mis estudios y en especial en este trabajo de diploma con el que finalmente me gradúo: a mi compañera de trabajo, tutores y otros compañeros que han aportado una ayuda valiosa para la terminación del mismo.

Jorge Ariel

Resumen

La investigación propone el análisis y diseño de una herramienta computacional para la simulación del Sistema Inmune. Dicho sistema constituye el mecanismo de defensa de los organismos frente a agentes extraños, por lo que es constante objeto de estudio de diferentes especialistas. Existen herramientas que permiten a los mismos apoyarse en sus investigaciones, pero los criterios en los cuales están basadas no son compartidos por todos, de ahí que en la investigación se proponga un modelo que permita flexibilizar las herramientas utilizadas. Debido a la complejidad en las interacciones entre los diferentes agentes (células, moléculas, entre otras) que componen dicho sistema, el modelo está basado en ecuaciones diferenciales y estocásticas, autómatas celulares y sistema basado en agentes.
La investigación muestra además los resultados del análisis y diseño de la herramienta en cuestión, se hace un análisis comparativo acerca de las tecnologías existentes para desarrollar este tipo de aplicación, se seleccionan las más apropiadas y se muestran los resultados del diseño de la propuesta de la herramienta.

Palabras claves: Sistema Inmune, Sistemas Biológicos, Células, Moléculas, Receptores, Antígenos, Autómatas celulares, Agentes.

Índice

Introducción

En los últimos años la investigación en el área de la Biología ha estado sufriendo una drástica transformación. Los avances experimentales obtenidos mediante el desarrollo de los proyectos de secuenciación genómica están transformando a la Biología en una disciplina rica en datos donde pueden empezar a descifrarse algunos de los complejos mecanismos de evolución y función celular. Esto ha motivado el nacimiento de una nueva disciplina denominada **"**Biología de Sistemas", una combinación de disciplinas diversas como la Biología Molecular, la Bio-Matemática, la Física, la Dinámica de Sistemas o la Bioinformática.

Dentro de la Bioinformática, el procesamiento de datos, la modelación y la simulación de procesos biológicos han sido las líneas de investigación por las que más interés han mostrado científicos y desarrolladores del mundo entero debido a la utilidad práctica que representan, pues devienen en herramientas computacionales sumamente útiles para el estudio del comportamiento de enfermedades y de posibles fármacos que las contrarresten. Dichas herramientas que describen y/o simulan los procesos biológicos se basan generalmente en el uso de diferentes técnicas de modelación como Ecuaciones Diferenciales, Redes Neuronales, Autómatas Celulares (AC) y Sistemas Basados en Agentes.

Por su parte el Sistema Inmune (SI) es un mecanismo de defensa que han desarrollado los organismos para protegerse ante la presencia de microorganismos extraños. Este sistema es sumamente complejo en su funcionamiento, de ahí que el desarrollo de herramientas computacionales que contribuyan con su estudio sea necesario, atractivo, aunque también complejo por la cantidad de incógnitas que aun prevalecen alrededor del SI.

En nuestro país los estudios biológicos gozan de un prestigio reconocido mundialmente gracias a la preparación y la competencia de nuestros científicos así como al Gobierno Cubano que tanto respaldo y prioridad les ofrece. El Centro de Inmunología Molecular (CIM) una de las instituciones científicas más prestigiosas del Polo Científico del Oeste de La Habana, 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.

Este centro ha propuesto una investigación conjunta con nuestra universidad específicamente con la facultad número seis (F6) cuyo perfil es la bioinformática -facultad de la Universidad de Ciencias Informáticas (UCI)- de cuya propuesta surgió el proyecto “Sistemas Biológicos” (BioSys) con el objetivo de crear herramientas que faciliten la modelación y/o simulación de los mismos.

Específicamente para el caso del SI, existen algunas herramientas desarrolladas que lo simulan, como el IMMSIM, el C-IMMSIM y IMMSIM++. Todas con la limitante común de tener pre-definida las entidades que representarían al sistema, es decir, un conjunto determinado de componentes del SI con interacciones también definidas, siguiendo criterios propios del equipo de desarrolladores que pueden no ser compartidos por todos los especialistas.
De ahí la necesidad de diseñar una herramienta que permita simular el SI de manera genérica, donde el investigador pueda definir sus propias entidades y las interacciones entre ellas.

Por lo cual la investigación se plantea el siguiente **problema científico**: ¿Cómo flexibilizar, la simulación del sistema inmune?
El **objeto de estudio** es la simulación del sistema inmune, lo cual define el **campo de acción** como la simulación del sistema inmune mediante autómatas celulares y sistemas de agentes.

Por lo que la investigación se plantea como **objetivo general** realizar el análisis y diseño de una herramienta para la simulación del SI y como **objetivos específicos** se ha definido:

.

- ✓ Determinar un modelo genérico que permita simular el sistema inmune.
- ✓ Realizar el análisis de una herramienta para la simulación del sistema inmune.
- ✓ Realizar el diseño de una herramienta para la simulación del sistema inmune.

Y para alcanzarlo se han propuesto como **tareas** las siguientes:

- ✓ Investigación sobre el funcionamiento del sistema inmune, sus características y componentes.
- ✓ Investigación sobre la situación actual de la simulación del sistema inmune mediante herramientas computacionales.

- ✓ Determinación de los requerimientos funcionales y no funcionales del sistema.
- ✓ Determinación de un modelo genérico que permita simular el sistema inmune.
- ✓ Realización del análisis de la herramienta de simulación.
- ✓ Realización del diseño de la herramienta de simulación.

Estructuración del contenido por capítulos:

Capítulo 1: Fundamentación teórica, se definen los principales conceptos relacionados con el objeto de estudio así como se describen las principales herramientas y metodologías utilizadas durante la investigación.

Capítulo 2: Análisis del Sistema, se determina y describe un modelo para simular el SI, así como el modelo de dominio, los requerimientos funcionales, no funcionales y se presenta el diagrama de casos de usos del sistema.

Capítulo 3: Análisis y diseño, se describe el análisis y diseño de la aplicación, se presentan los diagramas que lo caracteriza así como los patrones de diseño que se utilizaron.

Capítulo 1 Fundamentación Teórica

Introducción

En este capítulo se definen las bases conceptuales de la investigación. Se describen algunos conceptos relacionados con los sistemas biológicos, la biología de sistemas y algunas técnicas que utiliza esta última para modelar lo primero. Se generalizan las principales características del SI y las herramientas existentes que lo simulan. Se describen además las diferentes herramientas y metodologías utilizadas y/o consideradas para desarrollar la investigación.

1.1 Biología de sistemas y modelación de sistemas biológicos

1.1.1 Definición de Biología de Sistemas y Sistemas Biológicos

La biología de sistemas es un área de investigación bastante joven que se define como una combinación de diversas ciencias como la matemática, la computación, la física y la biología molecular aplicadas al estudio de sistemas biológicos.

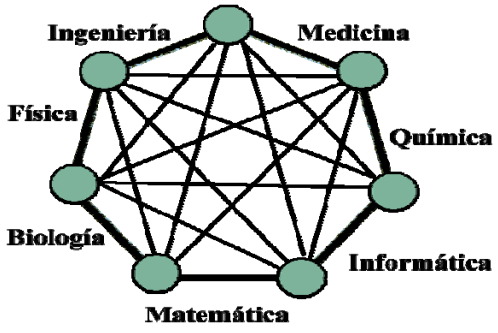


Fig. 1.1 Biología de Sistemas

Por su parte *los sistemas biológicos* pueden ser definidos como sistemas abiertos que operan en condiciones alejadas del equilibrio termodinámico, con muchas y fuertes interacciones no lineales entre sus muchos elementos. [11]

Luego, la biología de sistemas se centra en estudiar y analizar sistemas biológicos en su totalidad, usando herramientas de modelación, simulación, y comparación. [12]
La modelación y el análisis de los sistemas biológicos desarrollan modelos y herramientas tecnológicas para entender la complejidad de dichos sistemas biológicos. Para estos usos, en la integración de interacciones a través de niveles y de escalas y la aparición de los patrones que ayudan a explicar comportamiento del sistema, están las últimas metas para aplicar estas herramientas.

1.1.2 Estrategia de las 4M

Las 4M es una estrategia que resulta muy útil para la BS, pues integra lo experimental y lo computacional en un ciclo iterativo de inter-relación que permite obtener nuevas hipótesis que serán entonces modeladas, simuladas y validadas nuevamente por los resultados experimentales.

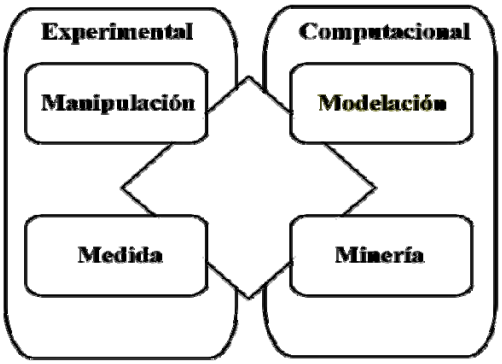


Fig. 1.2 Estrategia de las 4M

La Medición genera una significativa cantidad de datos que pueden ser analizados mediante Minería para determinar relaciones y comportamientos, mediante este análisis es posible entonces definir un Modelo que describa esos comportamientos, simularlos y validarlos mediante la Manipulación (la

Manipulación controla los cambios funcionales y estructurales de su entorno). Nuevas hipótesis son generadas mediante la Manipulación y re-medidas y así continúa el ciclo iterativo. [34, 35]

1.1.3 Modelación y simulación de sistemas biológicos.

Según el diccionario de la lengua española, modelar es ajustarse a un modelo, y un modelo es un objeto que se reproduce o se imita. Esta definición puede resultar insuficiente para entender lo que significa modelar sistemas complejos, pero en esencia describe su significado. Hitt, desde un punto de vista matemático señala que "a través de las funciones podemos modelar matemáticamente un fenómeno de la vida real, describir y analizar relaciones de hechos sin necesidad de hacer a cada momento una descripción verbal o un cálculo complicado de cada uno de los sucesos que estamos describiendo".
La *modelación* relacionada con sistemas de representaciones integra: símbolos, signos, figuras, gráficas y construcciones geométricas. Éstos expresan el concepto y suscriben en sí mismos el modelo con el cual es posible interpretar y predecir comportamientos de fenómenos físicos. La simulación y la modelación son representaciones de un objeto matemático que está vinculado a una situación física o real.

Una *simulación* es un intento por imitar o aproximarse a algo, es la acción de diseñar y dar solución a un modelo para un objeto o proceso con el objetivo de lograr una mejor comprensión del objeto o proceso en estudio (la simulación computacional hace uso de las computadoras para la solución del modelo propuesto); por su parte, modelar significa construir una representación de algo. La diferencia semántica reside en que un modelo es una representación de estructuras, mientras que una simulación infiere un proceso o interacción entre las estructuras del modelo para crear un patrón de comportamiento.

En la búsqueda de nuevas vías para realizar investigaciones biológicas y producir nuevos medicamentos se ha abierto el camino hacia la modelación matemática de los sistemas biológicos. Esta herramienta unida a la simulación computacional de los modelos obtenidos proporciona una serie de ventajas en comparación con los métodos clásicos, que ha posibilitado que muchos científicos de todo el mundo vean, en esta dirección, una alternativa para llevar a cabo estudios biomédicos. [13, 14]

1.1.4 Ventajas y desventajas

Son muchas las ventajas de estos nuevos métodos de investigación biológica. Casi todas ellas se derivan del hecho de que los tiempos requeridos para realizar un experimento son sumamente pequeños. En un laboratorio de investigación para realizar los estudios *in vivo* se requiere de meses de trabajo. Cada vez que un animal de laboratorio es inyectado con un producto se hace necesario esperar horas y en determinadas ocasiones hasta días, para tomar las muestras de sangre y hacer los estudios farmacocinéticos. Y en cada caso, si se quiere variar la concentración del medicamento es necesario repetir todo el ensayo otra vez. Estas debilidades de los métodos clásicos, constituyen la fortaleza de los métodos de simulación. Los ensayos que durarían meses y consumirían miles de dólares en reactivos químicos y animales de laboratorio, en un simulador sólo necesitan del cambio de un parámetro en las ecuaciones del modelo.

Resumiendo, la simulación de sistemas biológicos:

- ✓ Aceleran el descubrimiento de posibles drogas
- ✓ Minimizan los tiempos de obtención de un nuevo producto (2-3 años)
- ✓ Reducen los costos de investigación y producción
- ✓ Eliminan los problemas éticos vinculados al uso de humanos como animales de laboratorio
- ✓ Posibilitan la predicción de nuevos modelos a partir de los ya existentes
- ✓ Brindan facilidades para variar parámetros y condiciones experimentales. [14]

1.2 El sistema inmune, sus componentes y técnicas de modelación.

El **sistema inmune** es un extraordinario sistema de defensa que se ha desarrollado en los animales para defenderse de la invasión de microorganismos patogénicos y del cáncer. Es capaz de generar una enorme variedad de células y moléculas que pueden reconocer y eliminar gran variedad de invasores extraños. [1]

La capacidad de defensa se adquiere antes de nacer y se madura y consolida en los primeros años de la vida fuera del seno materno.

La *respuesta inmune inespecífica* es la primera barrera defensiva del organismo y no requiere sensibilización previa. Este tipo de respuesta es mediada por células con capacidad fagocítica y células asesinas naturales. Los mecanismos de defensa inespecíficos aportan un buen sistema de protección. Sin embargo, en muchas ocasiones no es suficiente para defender eficazmente al organismo.

La *respuesta específica o adquirida* se desarrolla sólo frente a la sustancia que indujo su iniciación y en ella participan prioritariamente los linfocitos y los elementos solubles liberadas por los mismos, anticuerpos y linfocinas. Todas las sustancias que se comportan como extrañas a un organismo frente a las cuales éste desarrolla una respuesta inmune específica, se conocen como antígenos. Los linfocitos son de dos tipos: linfocitos B y linfocitos T, éstos últimos, a su vez, pueden ser linfocitos T colaboradores (Th), linfocitos T citotóxicos (Tc) y por algunos autores también se proponen los linfocitos supresores/reguladores (Ts).

La respuesta inmune especifica, se considera que puede ser de dos tipos: humoral y celular. En general se considera que cuando el elemento efector final son las inmunoglobulinas formadas por los linfocitos B se trata de una respuesta tipo humoral, mientras que cuando participan los linfocitos T tanto colaboradores (Th) como citotóxicos (Tc), se trata de una respuesta tipo celular. Para que se inicie una u otra respuesta inmune se requiere el reconocimiento del antígeno y activación de los linfocitos. (Ver Anexo 1).

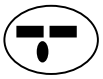
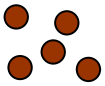
	Inmunidad Humoral	Inmunidad mediada por células	
Antigeno	 Bacterias extracelulares	 Microorganismos intracelulares en el macrófago.	 Microorganismos intracelulares (ej virus) replicándose en una célula infectada.
Linfoncitos respondedores	 Linfoncito B	 Linfoncito T	 Linfoncito T
Mecanismos efectores	 Anticuerpos secretados Eliminación de las bacterias	 Activación del macrófago muerte del microorganismo	 Lisis de la célula infectada
Transferido por	Sueros (anticuerpos)	Linfoncitos	Linfoncitos

Tabla 1.1 Inmunidad mediada por células.

Características de la respuesta inmune especifica:

La respuesta inmune específica se caracteriza por ser de carácter clonal, especifica, desarrollar memoria y ser regulable.

Especificidad. Se sabe que cada antígeno estimula solo a aquel linfocito o grupo de linfocitos que han desarrollado y en consecuencia poseen en su membrana los receptores capaces de reconocer y unirse específicamente a él. Estos receptores, tal como se ha indicado anteriormente, son las inmunoglobulinas de superficie cuando se trata de linfocitos B o el TcR cuando se trata de linfocitos T.

Clonalidad. Cuando un linfocito o grupo de linfocitos es activado, este prolifera y se diferencia en múltiples cedulas derivadas, todas ellas con idénticos receptores de superficie. Se dice entonces que todas estas células constituyen lo que se denomina clon celular. Tanto la especificidad como la clonalidad de la respuesta inmune, fueron originariamente definidos en los años cincuenta por varios inmunólogos entre los que se encontraba Burnet, y se conoció después por la teoría de selección clonal de Burnet. Esta teoría decía que cada antígeno estimulará a aquel linfocito o grupo de linfocitos que poseen en su membrana receptores capaces de reconocer y unirse específicamente a él y que como consecuencia se producía su proliferación y diferenciación en células con las mismas características de reconocimiento que los linfocitos originales.

Memoria Inmunológica. Otra característica importante de este tipo de repuesta es que el organismo mantiene memoria de un estímulo a otro cuando son de la misma índole. Eso se debe a la permanencia de linfocitos sensibilizados de larga vida después de un estímulo antigénico.

Regulación. Este tipo de respuesta dispone de mecanismos internos de control, de tal forma que la intensidad de la misma se regula por acción de diversos tipos de moléculas entre las que destacan las inmunoglobulinas y sobre todo las citocinas. (Las citocinas son una serie de sustancias producidas por células en respuesta a una gran variedad de estímulos y que son capaces de regular el funcionamiento de otras células.) [32]

Órganos que intervienen en el SI

A los órganos que forman parte del sistema inmunológico se les llama órganos linfoides, los cuales afectan el crecimiento, el desarrollo y la liberación de linfocitos. Los vasos sanguíneos y los vasos linfáticos son partes importantes de los órganos linfoides debido a que son los encargados de transportar los linfocitos hacia y desde diferentes áreas del cuerpo. Cada órgano linfoide desempeña un papel en la producción y activación de los linfocitos [33]

1.2.1 Simulación del sistema inmune

Al constituir el sistema defensivo frente a los organismos extraños, el SI es constante objeto de investigación por parte de los científicos, de ahí que propiciar herramientas de simulación que permitan su estudio es sumamente importante.

Existen diversos modelos que pudieran utilizarse para simular el SI, las ecuaciones diferenciales (que han sido bastante utilizadas), son un ejemplo pero presentan algunas desventajas que mencionamos a continuación [4]:

✓ No representan la verdadera complejidad de las células y las moléculas que lo componen (cada una de ellas tiene un comportamiento y una historia única que define su interacción particular con el ambiente).
✓ La segunda limitación es que las ecuaciones dan solamente el comportamiento medio de un sistema.
✓ Finalmente, los modelos del sistema inmune implican a menudo la no-linealidad, que hacen la solución de ecuaciones diferenciales difícil.

Sin embargo estas limitaciones podrían solucionarse modelando mediante otras técnicas de simulación como un Autómata Celular que básicamente es una herramienta computacional que hace parte de la Inteligencia Artificial basada en modelos biológicos, el cual está básicamente compuesto por una estructura estática de datos y un conjunto finito de reglas que son aplicadas a cada nodo o elemento de la estructura. [3]

De los autómatas y sus características se comenta en próximo epígrafe, ellos presentan varias ventajas, entre ellas que puede no ser necesariamente determinísticos, sino estocásticos, lo cual permite en nuestro caso estimar la distribución de los comportamientos exhibidos por el sistema, no apenas el promedio. También es fácil para modificar la complejidad de las interacciones sin introducir muchas dificultades para solucionar el modelo porque las no linealidades no son difíciles de modelar en autómatas celulares. Finalmente, el autómata puede representar los componentes y los procesos del interés en lengua biológica de modo que las aproximaciones en el modelo sean más biológicas en carácter, que matemáticas. [4]

1.2.2 Consideraciones sobre sistemas previos implementados para la simulación del sistema inmune.

Alrededor del funcionamiento del sistema inmune actualmente existen todavía algunas interrogantes. De manera general, aunque los descubrimientos en este campo han ido avanzando progresivamente, no se puede predecir con exactitud como reaccionan los organismos ante factores extraños a ellos. Esto crea la necesidad y la motivación por parte de inmunólogos y otros especialistas, por desarrollar herramientas que contribuyan a determinar patrones de comportamiento de este complejo sistema mediante la simulación.

IMMSIM

Este software fue propuesto por F. Celada y P.E. Seiden in 1992 fue programado en APL2. Los autores diseñaron el modelo que seguirían luego Kleinstein y Fillippo para hacer sus versiones en lenguajes C y C++ respectivamente. Este primer software solo representaba la respuesta humoral (basada en células B), y aunque la última versión realizada -el IMMSIM3- incluye la respuesta celular (basada en células T) presenta la limitación de que solo simula para un conjunto determinado de células según la consideración de los autores. La interfaz además, no es amigable.

C-IMMSIM

Esta aplicación es una nueva versión del IMMSIM desarrollado sobre lenguaje C, aunque incluye la respuesta celular y humoral, las entidades están previamente determinadas según el modelo de Seiden y Celada.

IMMSIM++

Este software tiene como antecedente el IMMSIM. Esta nueva versión realizada en incluye mejoras, pero solo modela entidades de la inmunidad humoral, lo cual supone una limitación para el investigador. Además se utilizan entidades definidas de la inmunidad humoral, lo cual hace al sistema un tanto rígido pues sólo es posible modelar las interacciones entre dichas entidades. Incluye la representación visual

de la simulación y además permite determinar que graficas representar y controlar los parámetros mientras esta corriendo la simulación. [28, 29]

De lo anterior descrito, se puede determinar fundamentalmente dos cosas, la primera, la efectividad del modelo que diseñaron Seiden y Celada que ha sido reutilizado por más de un desarrollador, y lo segundo la rigidez de dicho modelo, que resulta un tanto contradictorio para los especialistas puesto que no todos conciben el funcionamiento del SI de la misma manera. Es decir, asumir que una célula X, tiene un comportamiento Y ante cierta situación, sin dar posibilidad de variar el criterio, no es algo con lo que todos los investigadores estén de acuerdo. Y es eso constituye precisamente, la limitante principal de cada uno de estos sistemas.
De ahí la necesidad de diseñar una herramienta que permita flexibilizar el estudio del SI por vía computacional, es decir lograr hacer una herramienta genérica que permita al investigador representar sus propios conceptos.

1.3 Autómatas Celulares

Alrededor de 1940, John von Neumann estuvo trabajando en una teoría general de autómatas para el procesamiento de información que fuese aplicable tanto a sistemas biológicos como a aparatos tecnológicos. Fue Stanislaw Ulam quién sugirió a Von Neumann que utilizara un arreglo cuadriculado con un autómata de estados finitos en cada celda. Cada una de estas celdas estaría en un estado y utilizaría información sobre los estados de las celdas a su alrededor para pasar al siguiente estado. Von Neumann no llego a terminar su prueba escrita sobre este trabajo, perdiéndose así el interés por los autómatas celulares hasta que en los años 70 se hicieron populares por la publicación de llamado “Juego de la vida” de John H. Conway [18].

1.3.1 Descripción de Autómatas Celulares.

El mundo de los autómatas celulares es un arreglo uniforme de celdas, donde cada sitio contiene cierta información y donde el tiempo trascurre en pasos discretos. Las leyes de este universo consisten en las reglas que determinan cual será el próximo estado de la celda.

Este arreglo puede tener n dimensiones, aunque con más de tres dimensiones son muy difíciles de representar. Como ya habíamos dicho cada celda actualiza su estado tomando información de ella misma y de un grupo de celdas a su alrededor que constituyen un vecindario; los más comunes son el de Neumann, donde son tomados los 4 vecinos ortogonales (norte, sur, este, oeste) además de el mismo (Fig. 1.3 a), y el de Moore, que además de los vecinos ortogonales tiene en cuenta los diagonales y el mismo (Fig. 1.3 b) [15].

Fig. 1.3 Autómatas celulares de 4 y 6 vecinos.

La cantidad de estados en que puede estar una celda tiene que ser finito. Las reglas que determinan el siguiente estado de la celda se basan en los estados de las celdas que componen el vecindario, de esta forma si k es el número de estados posibles y n el número de celdas en el vecindario, existen kn posibles reglas de transición.

1.3.1.2 Autómatas celulares probabilísticos.

Los AC probabilísticos pueden definirse de dos formas; una donde las reglas del autómata son aplicadas uniformemente a todas las celdas del espacio celular, existiendo para cada cambio de estado una probabilidad de ocurrencia, a este caso se le denomina AC probabilístico uniforme (ACPU); y otra donde las reglas no son uniformes y cada celda puede aplicar una regla u otra con distintas probabilidades, denominándose este caso AC probabilístico no uniforme (ACPN)

Un ACPN se define de la siguiente manera:

M = {A, Q, Δ, N, P}

Donde A, Q, Δ, N tiene el mismo significado que en los AC, P se define como una función $P: \Delta \square Q$. La probabilidad de que una transición δ origine el nuevo estado q la denotaremos por p(δ, q). Se cumple que:

$$\forall \delta \in \Delta, \sum_{q \in Q} p(\delta(q) = 1$$

Autómata Celular Probabilístico Uniforme.

Un ACPU se define análogamente a los ACPN redefiniendo la función probabilística $P : \Delta XQ \rightarrow Q$. La probabilidad de que la transición δ produzca cambio de estado de q a q' se denota por p (δ, q' | q) y la probabilidad de que no se produzca el cambio es 1 - p(δ, q' | q).

1.4 Sistemas basados en agentes

El estudio de los agentes es un área de investigación en desarrollo. A pesar de que todavía no existe un consenso para definir los Agentes Inteligentes se han hecho ya aproximaciones a la definición que involucran la definición de Agentes y Agentes de Software.

Un *Agente*, de manera general es todo aquello que pueda considerarse que percibe su entorno mediante sensores y que responde o actúa mediante efectores.

Un *Agente de Software* es, un programa de computación que se ejecuta en un ambiente, y que es capaz de realizar acciones dentro de éste, con la finalidad de alcanzar los objetivos para los cuales fue diseñado. Para un Agente de Software sus percepciones y acciones vienen dadas por instrucciones de programas en algún lenguaje de programación. Un tipo de estos agentes son los llamados Agentes Inteligentes.

Los *Agentes Inteligentes* realizan tres funciones continuamente: perciben las condiciones de su entorno, actúan con el objetivo de modificarlo y razonan. Estos interpretan las percepciones, resuelven problemas y determinan las acciones a realizar. [36]

Los agentes son capaces de acometer acciones autónomas de forma flexible. La flexibilidad implica que estos agentes están dotados de un carácter reactivo, ya que pueden responder de forma instantánea a los cambios, y también de un carácter pro-activo, ya que su comportamiento está dirigido a lograr los objetivos mediante la toma de iniciativas. Las propiedades principales de los agentes se pueden resumir en cuatro: autonomía, habilidad social, reactividad y proactividad, propiedades que los hacen idóneos para la representación de sistemas de gran dinamismo.

Características y/o atributos de los Agentes	
Agente	✓ Ejecución autónoma. ✓ Comunicación con otros agentes y usuarios. ✓ Supervisión del estado del entorno de ejecución.
Agente Inteligente	✓ Capaz de utilizar símbolo y abstracciones. ✓ Capaz de explorar cantidades significantes del conocimiento del domino ✓ Capaz de adaptarse a un comportamiento orientado a objetivos.
Agente Inteligente Ideal	✓ Capaz de aprender de su entorno ✓ Capaz de tolerar errores ✓ Capaz de comunicarse usando lenguaje natural

Tabla 1.2 Características de los Agentes

1.5 Tendencias y tecnologías actuales.

Para lograr el objetivo general de la investigación se hace necesario el uso de una metodología que guíe el proceso de desarrollo así como de herramientas y lenguajes que permitan el diseño del mismo.

1.5.1 Metodología de desarrollo y lenguaje de modelado.

Un proceso de desarrollo de software define *quién* está haciendo *qué*, *cuándo* y *cómo* alcanzar un determinado objetivo; en la ingeniería de software el objetivo es construir un producto de software o mejorar uno existente con calidad. [6]
El alcance y complejidad de los sistemas informáticos que se desarrollan hoy en día, hace necesario el uso de una metodología de desarrollo que permita organizar y controlar los procesos de su producción y mantenimiento. En este sentido ha habido muchas propuestas, teniendo gran impacto en la actualidad el Proceso Unificado de Desarrollo de software (RUP).

Esta, es una metodología basada en un pequeño grupo de principios claves: el equipo de un proyecto de software debe planificar el desarrollo; debe conocer hacia donde se dirige; debe documentar el proyecto de una manera perdurable y extensible. Además incorpora el concepto de "mejores prácticas" para la ingeniería de software, definido por cinco características fundamentales [7]:

- ✓ Dirigido por casos de uso. El desarrollo está dirigido a satisfacer las necesidades de los usuarios del sistema expresadas en casos de uso.
- ✓ Centrado en la arquitectura. El desarrollo se centra en una arquitectura bien definida, con relaciones claras entre sus distintos componentes.
- ✓ Iterativo. El problema y la solución se organizan en pequeñas piezas, de manera que cada iteración se dirige específicamente al desarrollo de un conjunto de ellas.
- ✓ Incremental. Cada iteración se construye sobre la base creada por las iteraciones anteriores, agregándole capacidades al sistema.
- ✓ Controlado. El proceso se planifica y en cada momento está claro lo que debe hacerse

Existen otras como XP y FDD que son llamados “métodos ligeros”, centrados fundamentalmente en las relaciones interpersonales y la velocidad de reacción; intentando 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.

¿Por qué RUP?

Porque por sus características de ser iterativo e incremental, centrado en la arquitectura y por casos de usos permite un desarrollo eficiente para proyectos extensos y complejos como el que propone la investigación. Las fases de desarrollo que propone, el trabajo por roles y el cumplimiento de hitos en cada una de las fases garantiza que se desarrolle el proyecto de manera efectiva y eficiente.

Lenguaje de modelado

Como lenguaje de modelado utilizaremos UML (Unified Modeling Language) que es un lenguaje para especificar, construir, visualizar y documentar los artefactos (información que es utilizada o producida mediante un proceso de desarrollo de software) de un sistema de software orientado a objetos, que por su potencialidad se ha convertido en un estándar.
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.[22]

1.5.2 Herramienta CASE

Rational Rose Enterprise Edition

A medida que los sistemas que hoy se construyen se tornan más y más complejos, las herramientas de modelado con UML ofrecen muchos beneficios para todos los involucrados en un proyecto, por ejemplo,

administrador del proyecto, analistas, arquitectos, desarrolladores y otros. Las herramientas CASE de modelado con UML nos permiten aplicar la metodología de análisis y diseño orientados a objetos y abstraernos del código fuente, en un nivel donde la arquitectura y el diseño se tornan más obvios y más fáciles de entender y modificar. Cuanto más grande es un proyecto, es más importante utilizar una herramienta CASE. [9]

Rational Rose (RR) es una herramienta nos permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable. 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). Es la herramienta líder en el mundo para el modelado de sistemas complejos y de tiempo real. Para los analistas de negocios, ofrece la capacidad de modelar y visualizar sus procesos de negocios y destacar oportunidades para aumentar la eficiencia. Para los analistas de datos, el modelado de su diseño de base de datos en RR, mejora la comunicación entre el cliente y los desarrolladores, se asegura que la solución sea creada con el usuario en mente. Unifica así, a los analistas de negocios, sistemas y datos al permitirles crear y administrar modelos en una herramienta con un solo lenguaje de modelado. [8, 10]

Además utiliza un proceso de desarrollo iterativo, donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración, primero se identifican los riesgos y después se prueba la aplicación para que éstos se hagan mínimos.
Permite generar código en distintos lenguajes de programación a partir de un diseño en UML y proporciona mecanismos para realizar la denominada Ingeniería Inversa.

Visual Paradigm

Por su parte Visual Paradigm (VP) es una herramienta CASE poderosa y fácil de usar que utiliza UML como lenguaje de modelado. Permite representar todo tipo de diagramas UML, aplicar ingeniería inversa, generar códigos entre otras funcionalidades.

Visual Paradigm ofrece [19, 20]:

- ✓ Entorno de creación de diagramas para UML 2.0
- ✓ Producto de calidad.
- ✓ Soporta aplicaciones web.
- ✓ Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Capacidades de ingeniería directa (version profesional) e inversa.
- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo
- ✓ Disponibilidad de múltiples versiones, para cada necesidad.
- ✓ Disponibilidad de integrarse en los principales IDEs.
- ✓ Disponibilidad en múltiples plataformas
- ✓ Fácil de instalar y actualizar.
- ✓ Compatibilidad entre ediciones.

¿Por qué Visual Paradigm?

Como se pudo apreciar RR es una herramienta verdaderamente potente y muy usada pero es software propietario y no corre sobre Linux. Visual Paradigm por su parte es multiplataforma, también poderosa, se integra con NetBean y es software libre, características que nos hacen seleccionarla como herramienta CASE para el desarrollo de la investigación.

1.5.3 Entorno integrado de desarrollo (IDE)

NetBeans

NetBeans es un entorno de desarrollo o IDE (Integrated Development Environment). Tiene editor de código sensible al contenido con soporte para autocompletar el código, coloreado de etiquetas, autotabulación y uso de abreviaturas para varios lenguajes de programación.

Incluye además:

- ✓ Soporte para Java, C, C++, XML y lenguajes HTML.
- ✓ Soporte para JSP, XML, RMI, CORBA, JINI, JDBC y tecnologías Servlet
- ✓ Incluye CVS (control de versiones) y Ant (compilación avanzada)
- ✓ Posibilidad de utilizar otras versiones de compiladores, depuradores,...
- ✓ Creación visual de componentes gráficos
- ✓ Herramientas con asistentes para facilitar la escritura de código
- ✓ Dispone de todo un entorno para crear documentación javadoc

Para su uso se requiere disponer en Windows XP de 125 MB en disco duro, 500 MB de RAM y procesador Pentium III 500 Mhz. En Linux de 125 MB en disco duro, procesador Pentium III 500 Mhz y 256 MB RAM. [21]

El IDE NetBeans es un producto libre y gratuito sin restricciones de uso. Soporta el desarrollo de todos los tipos de aplicación Java (J2SE, Web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring.

Esta plataforma es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- ✓ Administración de las interfaces de usuario (ej. menús y barras de herramientas)
- ✓ Administración de las configuraciones del usuario
- ✓ Administración del almacenamiento (guardando y cargando cualquier tipo de dato)
- ✓ Administración de ventanas
- ✓ Framework basado en asistentes (diálogos paso a paso)[16]

Eclipse

El entorno integrado de desarrollo (IDE) de Eclipse emplea módulos (plug-in) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software. Adicionalmente le permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Phyton, permite a Eclipse trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y Sistema de gestión de base de datos. La arquitectura plugin permite escribir cualquier extensión deseada en el ambiente, como seria Gestión de la configuración. Se provee soporte para Java y CVS en el SDK de Eclipse. Y no tiene porque ser usado únicamente para soportar otros lenguajes de programación

En cuanto a las aplicaciones clientes, eclipse provee al programador con frameworks muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, etc.
El SDK de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código. El IDE también hace uso de un espacio de trabajo, permitiendo modificaciones externas a los archivos en tanto se refresque el espacio de trabajo correspondiente.
La versión actual de Eclipse dispone de las siguientes características:

- ✓ Editor de texto
- ✓ Resaltado de sintaxis
- ✓ Compilación en tiempo real
- ✓ Pruebas unitarias con JUnit
- ✓ Control de versiones con CVS
- ✓ Integración con Ant
- ✓ Asistentes (wizards): para creación de proyectos, clases, tests, etc.
- ✓ Refactorización
- ✓ Asimismo, a través de "plugins" libremente disponibles es posible añadir:
- ✓ Control de versiones con Subversion, vía Subclipse.

- ✓ Integración con Hibernate, vía Hibernate Tools. [17]
- ✓ La misma tiene varios beneficios como son [23, 24]:
- ✓ 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.

¿Por qué NetBeans?

Aunque son dos herramientas que se equilibran bastante en una balanza, se determinó utilizar NetBeans porque tiene mejor interfaz gráfica y es un poco más rápido en la compilación.

1.5.4 Lenguaje de programación

C++

El C++ es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C.
Las principales características del C++ son el soporte para programación orientada a objetos y el soporte de plantillas o programación genérica (templates). Se puede decir además que es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel:

- ✓ Posibilidad de redefinir los operadores (sobrecarga de operadores)
- ✓ Identificación de tipos en tiempo de ejecución (RTTI)

C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo es a su vez uno de los que menos automatismos trae (obliga a hacerlo casi todo manualmente al igual que C) lo que "dificulta" mucho su aprendizaje y utilización.[25]

Java

Java supone un significativo avance en el mundo de los entornos software, y esto viene avalado por tres elementos claves que diferencian a este lenguaje desde un punto de vista tecnológico:

✓ Es un lenguaje de programación que ofrece la potencia del diseño orientado a objetos con una sintaxis fácilmente accesible y un entorno robusto y agradable.
✓ Proporciona un conjunto de clases potente y flexible.
✓ Pone al alcance de cualquiera la utilización de aplicaciones que se pueden incluir directamente en páginas Web (aplicaciones denominadas applets).

Cuenta con determinadas características como:

✓ Orientación a objetos
✓ Riqueza semántica
✓ Robusto
✓ Fácil aprendizaje
✓ Interactivo y animado
✓ Multiplataforma

En Java podemos crear los siguientes tipos de aplicaciones:
✓ Aplicaciones: Se ejecutan sin necesidad de un navegador.
✓ Applets: Se pueden descargar de Internet y se observan en un navegador.
✓ JavaBeans: Componentes software Java, que se puedan incorporar gráficamente a otros componentes.

✓ JavaScript: Conjunto del lenguaje Java que puede codificarse directamente sobre cualquier documento HTML
✓ Servlets: Módulos que permiten sustituir o utilizar el lenguaje Java en lugar de programas CGI (Common Gateway Interface) a la hora de dotar de interactividad a las páginas Web.[26]

¿Por qué Java?

Porque es un lenguaje fácil de aprender, potente, con diversas funcionalidades y características que lo hacen muy popular y se alejan de la complejidad propia del C++.

1.5.5 Computación Grid

Un grid computacional es una infraestructura de hardware y software que suministra al que la utiliza acceso seguro, consistente, penetrante y barato, a unas elevadas capacidades computacionales. Es un conjunto de recursos (ciclos de CPU, datos, sensores, etc.), y todos esos recursos necesitan una interconexión hardware y un control software para que estén ensambladas en un grid. Esta infraestructura debe proporcionar a los usuarios un servicio seguro a todos los niveles: capacidad de cómputo, de integridad de datos y de seguridad de acceso. El servicio debe ser consistente, basado en estándares, y de esta manera el acceso y las operaciones sobre el grid estarán definidas por los mismos evitando la heterogeneidad [30]

1.5.6 Herramienta BioSysModelador

BiosysModelador es una herramienta diseñada por un grupo de desarrolladores de la facultad de Bioinformática de la Universidad de Ciencias Informáticas (UCI) que permite el modelado gráfico de SB. Tiene un conjunto de funcionalidades que le permiten al usuario modelar sistemas biológicos para luego realizar simulaciones a partir de él.

1.5.7 Estándares utilizados

XML: Es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.
Las tecnologías XML son un conjunto de módulos que ofrecen servicios útiles a las demandas más frecuentes por parte de los usuarios. XML sirve para estructurar, almacenar e intercambiar información. [31]

Conclusiones

En este capítulo se definieron conceptos importantes relacionados con el objeto de estudio, se describieron algunos softwares que simulan el SI y se presentaron las principales características de las metodologías y herramientas. Luego de dicho estudio se determinó aplicar la metodología de desarrollo RUP; usar el lenguaje de modelado UML sobre la herramienta CASE Visual Paradigm; lenguaje de programación Java y NeatBeans como entorno de desarrollo. Se utiliza además el estándar XML, la herramienta de modelación grafica BioSys-Modelador y la computación Grid.

Capítulo 2. Análisis del sistema

Introducción

En este capítulo se presenta una propuesta de un sistema que le de solución al problema planteado. Consta de dos partes fundamentales, una primera donde se explican las consideraciones teóricas de un modelo sobre el cual se basa dicho sistema y otra donde se describe el dominio, se plantean los requerimientos funcionales y no funcionales, así como se determinan y describen los casos de uso que caracterizan el sistema.

2.1 Propuesta del sistema.

2.1.1 Consideraciones generales

En el Capítulo 1 se mencionaron algunos software que simulan el sistema inmune. Tanto el IMMSIM, como el C- IMMSIM y el IMMSIM++, siguen el modelo descrito por Seiden y Celada; todos con la limitante de tener entidades pre-definidas. En este epígrafe se propone una adaptación de dicho modelo con la idea de lograr genericidad, ya que el modelo en sí es abarcador, pero rígido. El objetivo es que sirva de base para un posterior incremento de las funcionalidades del software que aquí se diseña.

El espacio de simulación se compone por celdas y agentes. Como agentes se entienden aquellos componentes del sistema inmune que intervienen en la respuesta inmunitaria, es decir células, anticuerpos, antígenos, moléculas y como procesos aquellos sucesos que permiten a las células cambiar de estados (definidos por los propios procesos) y ejecutar acciones. Claro, que el concepto de agente que nos planteamos es reducido, es decir nuestros agentes van a estar caracterizados por su dinamismo, van a actuar en dependencia de su entorno pero no son inteligentes. Las tablas 2.1 y 2.2 muestran los procesos y agentes aquí definidos.

Procesos
Generación en la Médula Ósea
Diferenciación en el Timo
Muerte celular
Estimulación
Inhibición
División clonal
Interacción interna
Interacción
Segregación

Agentes
Células(compuesta por receptores)
Químicos
Antígenos

Para representar los agentes que definen el modelo y los procesos que ocurren entre ellos, se utiliza BioSysModelador, una herramienta que permite representar diagramas según un lenguaje de modelado biológico que fue descrita en el capítulo 1.
Las ideas generales del modelo así como las ecuaciones que se utilizan podrán ser encontradas en las referencias [4,27, 28].

2.1.2 Compartimientos y procesos del modelo

El simulador está compuesto por tres compartimientos que se corresponden con tres regiones anatómicas de los animales: la médula ósea, el timo y el nodo linfático. Cada una de estos compartimentos serán descritos a continuación, así como los diferentes procesos que van a ocurrir en ellos.

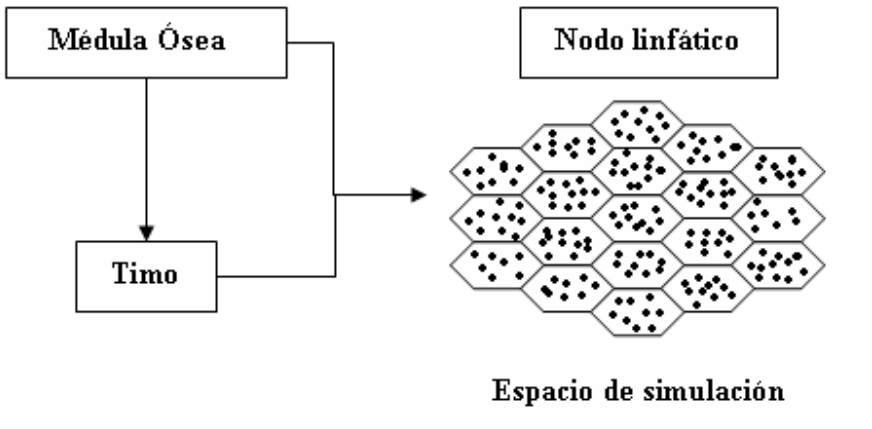


Fig 2.1 Compartimentos que componen el simulador

Médula Ósea

La médula ósea es el órgano donde se generan y regulan las células que intervienen en el sistema inmune, este compartimiento no tiene representación geométrica sino que, considerando que en este compartimiento se generan poblaciones, utilizaremos ecuaciones diferenciales estocásticas para determinar cantidades de una célula x

$dx(t) = \eta(\bar{x} - x(t)) + \sigma\ d \in (t)$, $x(0) = x_0$ que en nuestro caso significa:

$\frac{dx_i(t)}{dt} = \frac{\ln 2}{\lambda_i}(x_i(0) - x_i(t)) + \frac{d\varepsilon(t)}{dt})$ para las constantes $\bar{x}$ y x_0 .

Dado x_0 , el valor del tiempo t para el proceso x(t) es normalmente distribuido lo cual significa :

$E[x(t) \mid x_0] = \bar{x}$ y $Varx[(t) \mid x_0] + = \frac{\lambda}{2\ln 2}(1 - e^{-2\frac{\ln 2t}{}})$

Este proceso es importante en la regulación, ya que garantiza que cuando el número de células es mayor que el valor inicial el promedio de muerte aumenta y $\frac{dx_i}{dt} < 0$, por el contrario, si el número de células es menor que el valor inicial, el promedio de nacimiento aumenta y $\frac{dx_i}{dt} > 0$.

Timo

No todas las células abandonan la médula ósea listas para interactuar, algunas necesitan pasar por un proceso de maduración que se realiza en este caso en el Timo. Una vez en el Timo, deberán someterse a la selección positiva (garantiza que la célula esté lista para interactuar, elimina a la célula que no este lista para interactuar) y a la selección negativa (elimina a aquellas células que reaccionan contra el propio organismo).

Ampliando la explicación, en la selección positiva la célula tiene que interactuar con un mínimo de afinidad con otra de las células presente en el timo, de lo contrario el organismo (en este caso el sistema) las elimina, y en la selección negativa como parámetro de entrada se define un bit-string que va a representar "lo propio" (así se le denomina a los antígenos propios del organismo), y si la afinidad entre la célula y "lo propio" es muy alta, también se elimina.

El resultado de este proceso de selección es que las células que finalmente abandonen el timo y entren en circulación, habrán creado una tolerancia a las células propias del organismo (impidiendo que lo ataque) y estarán listas para interactuar

La probabilidad de que una célula pase la selección negativa esta dada por:

$\Pr\left[\prod_{j,k}\left(1 - A_{jk} * B_{jk}\right)\left(1 - C_{jk} * Dj_k\right)\right]^{ThymEff}$ donde para cada molécula P y receptor R se tiene que:

$$A_{jk} = Affinity\ (R_i, left\ (R_j)\quad right\ (P_i))$$

$$B_{ij} = match(R_i, P_i)^{1/2}$$

$$C_{jk} = Affinity(R_i, left(R_j)\quad left(P_i))$$

$$D_{jk} = macth\quad (R_j, P_i)^{1/2}$$

Interacciones

Las interacciones ocurren cuando dos células, o una célula y un antígeno coinciden en tiempo y espacio bajo ciertas condiciones (definidas por el usuario) y si son afines para interactuar.

La afinidad se define mediante el modelo de de bit-string, este modelo usa un string de bits (0 y 1, generados aleatoriamente) para representar la especificidad de los elementos. La longitud del string estaría representado por un parámetro llamado NBISTR. Dos bit-string se complementarían uno al otro (o definen un “match” perfecto) si para cada cero en uno se corresponde con un 1 en el otro o vs. De manera general, un m-bit es definido como un par donde exactamente m bits se complementan uno al otro.

La función match (a, b)= hamming (a, b) determina el número de bits que “matchean” entre dos string, y es calculado como la distancia “Haming”. Así mismo, mismatch (a, b)= NBISTR- match (a, b) determina el número de bits que no “matchean”. Se define entonces la funcion Affinity(a,b)= Affinity(match (a, b))= Affinity(m). El parámetro affLevel define el límite por debajo del cual la interacción no va a ocurrir. La figura muestra una célula B con dos receptores cada uno representado mediante un bit-string.

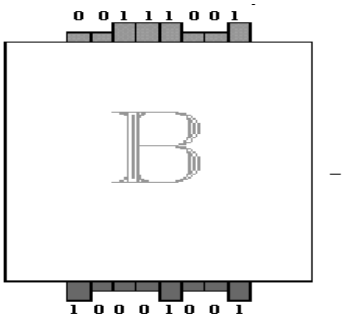


Fig 2.2 Bit-string asociados a una célula

Interacciones internas

Este tipo de interacciones caracterizan el caso de los antígenos intracelulares, es decir, aquellos que infectan a la célula desde dentro de sí misma. En este caso, la interacción ocurre de manera similar a la anterior sólo que cuando se computa, la célula tiene la información de dicho antígeno y entonces no se seleccionan dos agentes aleatorios para interactuar sino que se comporta como uno solo, se considera compuesto. Este proceso es conocido como digestión del antígeno, y la muerte de este tipo de antígeno (muerte por fagositosis) se explica más adelante.

División clonal.

La división clonal, es el proceso en el cual una célula, después de recibir una señal se divide en célula de memoria (célula de larga vida de duración) o célula efectora (reconoce y elimina un antígeno específico). Esta es la probabilidad de entrar en el ciclo mitótico.

$$\text{E Pr}\left[TH divides\right] = F_1\left(n\left(x,t\right)\right) * F_2\left(\mu\left(x,t\right)\right) \qquad 2.1$$

$$F_1 n\left((x,\ t)\right) = \exp\left(- \frac{n(x,\ t)^2}{\tau^2 \left(\sum n(x,0)\right)^2} \right)$$

$$F_2\left(\mu_{(x)}(x,\ t)\right) = 1 - \exp\left(- \left(\frac{\mu_{(x)}(x,\ t)}{n} \right)^2 \right)$$

Donde n(x,t) es el número total de células en el lugar x en un tiempo t y $\mu(x,t)$ es la concentración del químico en la celda donde ocurre la interacción.

Segregación, Estimulación, Inhibición.

Estos procesos se refieren a acción de los químicos dentro del sistema, pueden ser segregados por las células dentro del sistema o pueden considerarse en circulación sin ser segregados en la simulación; y su función puede ser estimular, inhibir procesos o destruir antígenos en caso de que se refieran a los anticuerpos. El rate de segregación, consumo y degradación caracterizan la forma en que se computa.

Vida media y muerte celular

La vida media es el tiempo promedio de vida que tiene una célula de acuerdo a sus características. Se define para célula como parámetro de entrada al sistema a consideración del investigador.

La *muerte celular por envejecimiento* se determina mediante la ecuación Pdie= $e^{\frac{-\ln 2}{\lambda}}$ donde λ es la vida media. También se computa la *muerte por fagocitosis*, esto es cuando una célula es infectada internamente con un antígeno y éste muere dentro de la misma luego de la interacción con otra célula. Cuando una célula muere esta sale completamente del sistema.

Nodo linfático:

El nodo linfático, en los organismos, es el primer sitio donde las células que intervienen en el SI interactúan con los atígenos que entran a través de tejidos [10]. En nuestro sistema, este compartimiento representa el espacio de simulación donde las células van a interactuar y se va a desatar la respuesta inmunitaria.

Para representar estas interacciones, es conveniente utilizar un autómata celular. Sus ventajas se basan fundamentalmente en que permite la inclusión de la complejidad intrínseca del sistema sin gran dificultad, y la descripción de las varias entidades e interacciones se hace en términos biológicos más bien que matemáticos.
Las reglas generales para los autómatas celulares son las siguientes:

- ✓ Número finito de celdas
- ✓ Número finito de iteraciones
- ✓ Cada celda tiene un número finito de posibles valores (estados)
- ✓ Conjunto finito de reglas determinísticas.
- ✓ Las reglas de la evolución de un determinado sitio dependen de los vecinos

Definiendo un autómata que se adecue a los requerimientos de nuestro sistema, éste sería un ACPU donde es necesario extenderlo generalizando las reglas como a continuación:

En la regla 4 “determinísticas” es cambiada por “estocásticas”. La regla 5 es cambiada por el sitio en si mismo.
Y se adicionaría una 6 regla:
Las entidades se mueven de sitio a sitio (difusión, tiende a homogeneizar el sistema).

El autómata se define con 6 posibles vecinos (forma hexagonal), tendrá X*Y celdas (X e Y definen la dimensión del espacio) y en cada una de ellas las interacciones van a ocurrir de manera independiente y aleatoria. Como se muestra en la Fig 2.3 esta estructura se representa de manera bi-dimensional de manera que al unir los bordes de la izquierda con la derecha y el límite superior con el inferior se logre

una forma toroidal que garantiza que cada celda (incluyendo los bordes y las esquinas) tenga 6 vecinos idénticos.

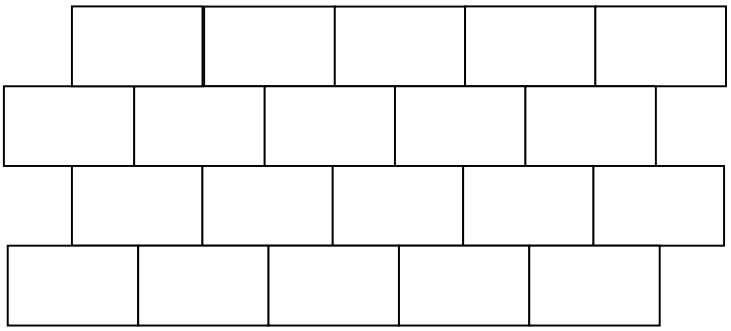


Fig 2.3 Representación 2 dimensional de celdas hexagonales.

Esto permite que en cada celda sólo interactúen agentes que estén dentro de las mismas y que luego que se ejecuten los procesos dentro de las celdas éstas puedan difundirse hacia sus celdas vecinas. Esto constituye una iteración.

De esta forma la probabilidad de las células que pertenecen a una determinada celda, de difundirse para un vecino aleatoriamente para cada iteración, sería $1 - d_x$, d_x se define como

$$d_x = \left(\frac{N_x}{\bar{M}}\right) \quad if \leq 1 \text{ , sino 1;}$$

donde

$$\bar{M} = \frac{10^9 \times n\left[cubicmicro \quad meters\right]\left[cells\right]}{L^2 \times 523{,}6\left[cubicmicro \quad meters\right]}$$

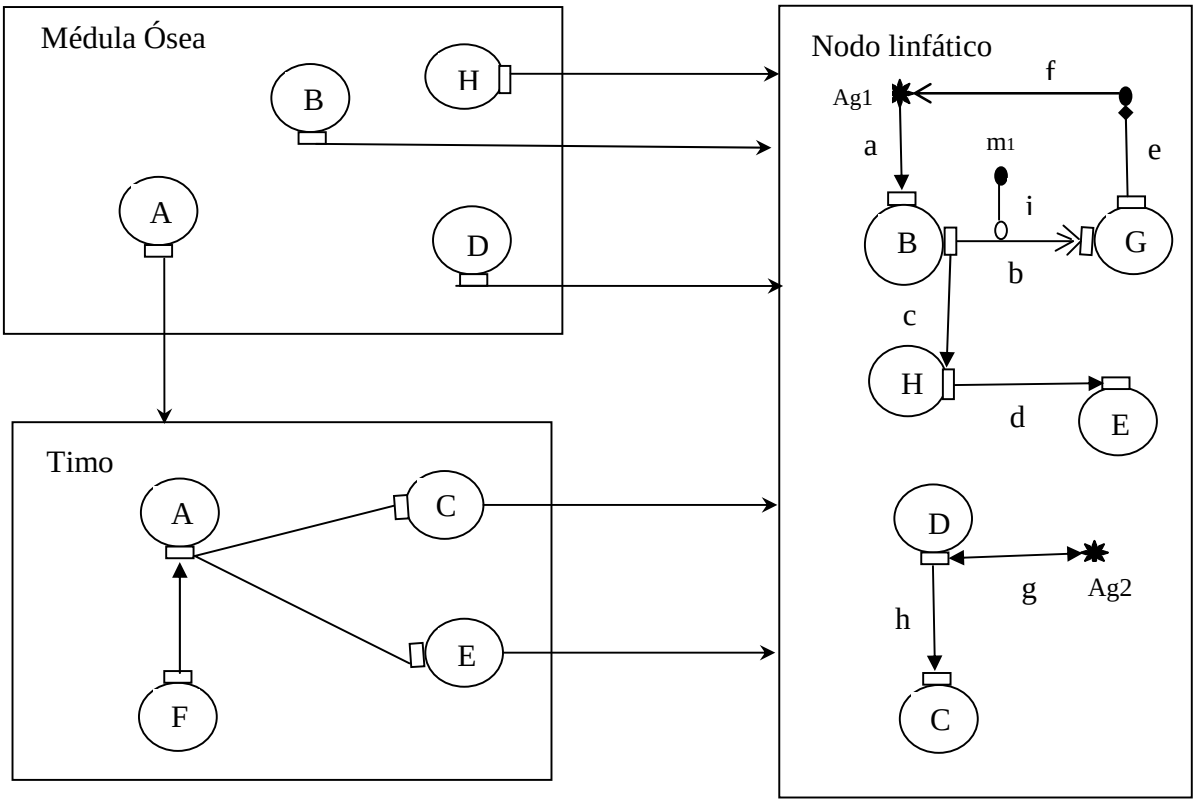
es el volumen máximo de células en cada celda y N_x es el número total de células (de todo tipo) en una celda.

Ejemplo

El componente Médula Ósea contiene las células que serán creadas en la misma, ellas pasan al espacio de simulación mediante una función f(x)=C, excepto A que debe diferenciarse en el timo primero. Una vez en el Timo, A debe reaccionar con una probabilidad p con la célula X para sobrevivir (selección positiva) y además no debe sobre accionar al péptido propio que defina el usuario (selección negativa), una vez halla pasado estas dos pruebas se diferencia en C o en E, que básicamente representarían a la misma célula A sólo que con receptores diferentes. C y E pueden entonces pasar al espacio de simulación. Una vez todo el repertorio de células se encuentren creadas y distribuidas (asignadas a una celda x) en el espacio de simulación, las celdas aleatoriamente escogen dos entidades y un proceso, dicho proceso determina si se puede ejecutar o no (el proceso es quien sabe que entidades participan en él). En caso de que se no se pueda la celda repite el procedimiento.

En el ejemplo que muestra la figura, los procesos **a**, **c**, **d**, y **h** son interacciones, **b** es una división clonal estimulada por el proceso **i**, **e** representa segregación, **f** muerte y **g** es una interacción-interna.

Cada proceso tiene definido la entidad saliente (ejecutora), la entrante (resultante) y si sobre él influye un proceso que llamamos intermedio (estimulación e inhibición). Así, antes de que un proceso sea ejecutado por una entidad, se debe comprobar que los procesos que tienen a dicha entidad como entrante hayan sido ejecutados.



Para explicar brevemente a manera de ejemplo como funcionan los procesos a, b, c, i entre las entidades involucradas en los mismos definiremos las siguientes sintaxis (cada proceso actual se definen como un proceso m)

Interacción.

Condición: si los procesos de orden m-1 se ejecutaron && si son afines && ocurre con una probabilidad p || si los procesos de orden m-1 se ejecutaron && si son afines && ocurre con una probabilidad p && estimulación || si los procesos de orden m-1 se ejecutaron && si son afines && ocurre con una probabilidad p && inhibición.

Acción: activa bandera para que se puedan ejecutar el proceso m+1|| activa la bandera de no ejecutado && no se ejecuta el proceso por inhibición.

La división clonal:

Condición: si los procesos de orden m-1 se ejecutaron && ocurre con una probabilidad p && el volumen de la celda lo permite || si los procesos de orden m-1 se ejecutaron && ocurre con una probabilidad p && estimulación && el volumen de la celda lo permite || si los procesos de orden m-1 se ejecutaron && ocurre con una probabilidad p && inhibición.
Acción: activa bandera de ejecutado && genera clones de sí misma || activa la bandera de no ejecutado && no se ejecuta el proceso por inhibición.

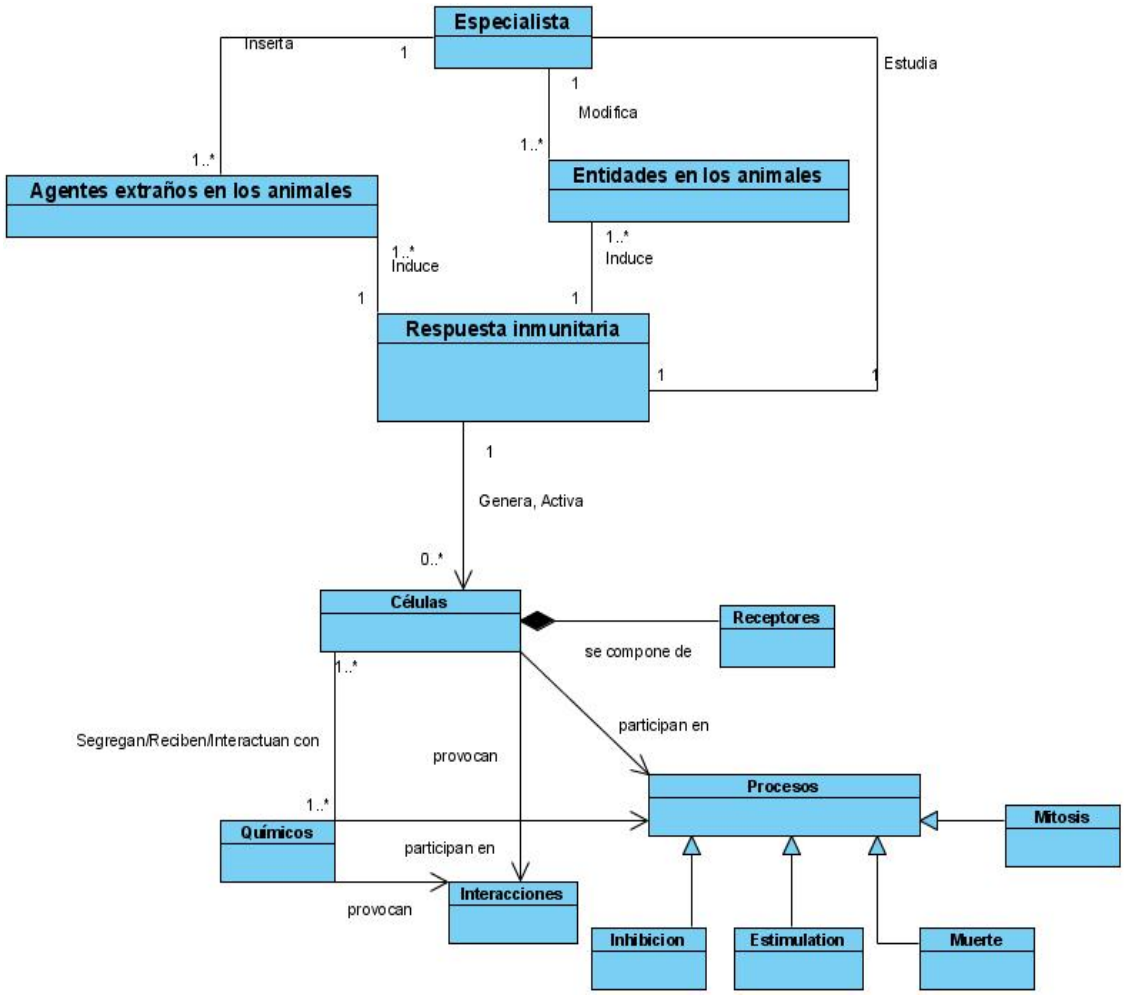
Así para el ejemplo que estamos tratando el proceso **a** puede ejecutarse libremente pues no depende de ninguno anterior, ya que la entidad saliente sólo es entrante en un proceso de muerte. Como no tiene proceso intermedio se ejecuta atendiendo a la primera variante de la condición de la sintaxis. Luego, cuando el proceso **c** valla a ser ejecutado, comprobara que **a** ya lo haya hecho, en caso negativo no podrá ejecutarse. Así **c** hará lo mismo, pero como tiene un proceso intermedio **i**, dependerá de si la cantidad de químicos1 que existan durante la ejecución de i es suficiente para comenzar la clonación.

2.2 Modelo de dominio

2.2.1 ¿Por qué modelo de 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 para el fácil entendimiento de lo que deber hacer el sistema.

Fig Diagrama del modelo de dominio



2.2.3 Descripción del modelo de dominio

Para el estudio del SI por vía experimental, el *investigador* inserta determinados *antígenos* en ratas de laboratorio (algunos experimentos son *in vitro*) o *modifica* algunas de sus células con el fin de estudiar la reacción del mismo ante la presencia del *agente extraño*. El estudio básicamente consiste en monitorear

el comportamiento o variación de distintos parámetros en las entidades que intervienen en la *respuesta inmunitaria* como son las *células*, *los receptores* de las células, *los químicos* y como i*nteractúan* entre ellos y en los *procesos* que participan. Estos parámetros pueden ser crecimientos y decrecimientos poblacionales, velocidad de respuesta de las entidades y eficiencia o deficiencia de estas en los procesos, entre otros.

Los procesos fundamentales son la *mitosis*, la *generación de células*, *diferenciación en el timo*, la *segregación* de químicos la *estimulación* e *inhibición* de las células por parte de las moléculas y la *muerte* celular. Luego se repiten los experimentos en otros organismos (o en los mismos bajo determinadas modificaciones), algunas con características similares, otras diferentes, para observar las distintas reacciones de los mismos ante la presencia del mismo factor extraño u otro diferente.

2.3 Requerimientos del sistema

Los requerimientos del software constituyen las capacidades o condiciones que el sistema debe cumplir o alcanzar. A continuación se exponen los requisitos separados en funcionales y no funcionales.

2.3.1 Requerimientos funcionales

Los requerimientos funcionales, surgen partiendo del análisis del modelo de dominio y de las necesidades que el cliente considera debe satisfacer el sistema. A continuación se enumeran.

R1 Insertar datos de las entidades

R2 Simular la respuesta inmunitaria

- R2.1 Generar células.
- R2.2 Diferenciar células en el Timo.
  - R2.2.1 Determinar selección positiva y negativa de los linfocitos en el timo.
- R2.3 Inyectar antígeno.
- R2.4 Simular el proceso de mitosis (división celular)
- R2.5 Simular interacciones entre entidades.

R2.5.1 Interacciones extracelulares.

R2.5.2 Interacciones intracelulares.

R2.6 Simular muerte celular.

R2.6.1 Muerte por fagocitosis

R2.6.2 Muerte por envejecimiento

R2.7 Determinar afinidad entre células y moléculas

R2.8 Simular la difusión celular.

R.3 Graficar simulaciones.

R.4 Gestionar ficheros

R4.1 Salvar fichero

R4.2 Abrir fichero

R.5 Introducir parámetros generales

2.3.2 Requerimientos no funcionales

Los requerimientos no funcionales constituyen propiedades o cualidades que el producto final debe tener. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto. Los numeramos a continuación.

✓ Apariencia o interfaz externa.

La interfaz de manera general es sencilla, debe permitir iconos de acceso directo para las opciones de simular, salvar y guardar.

✓ Usabilidad.

La aplicación se usará en el CIM o en cualquier centro del polo que presice de su uso. Podrá usarla cualquier investigador capacitado. Contendrá un manual de usuario con las descripciones básicas sobre

como trabajar con la aplicación incluyendo las especificaciones o características del módulo de representación gráfica de las entidades y los procesos. No tiene restricciones de usuarios.

✓ Rendimiento.

La rapidez de la simulación dependerá del volumen de información que necesite procesar, aun así tiende a demorar un poco en dar respuesta, por eso se propuso la integración a una Grid que distribuirá el trabajo haciéndolo más eficaz.

✓ Soporte.

Una vez terminada la aplicación se instalará en CIM para realizar pruebas piloto del software y de despliegue y luego se instalará en cualquier centro del polo que presice de su uso. A los posibles usuarios se les explicará como instalar la aplicación y se actualizará a medida que se diseñen las mejoras.

✓ Seguridad.

No requiere requisitos especiales de seguridad

✓ Políticos y Culturales

La aplicación empleará el español como idioma. Algún cambio que se quiera hacer, será a través de la dirección del proyecto de los Centros vinculados a este y será entonces tramitado con la Vicerrectora de Producción de la Universidad de la Ciencias Informáticas (UCI).

✓ Confiabilidad

Este sistema será administrado por el personal de calidad de cada centro en el cual se esté usando, lo usarán los usuarios que estén asignados para trabajar con él. Por lo tanto, toda la información que fluya de dicho sistema, será la emitida por los diferentes grupos, departamentos y directivos en general.
Si se necesitara por parte de algún directivo de otra área visualizar cierta información, se acudirá al los directivos del proyecto.

✓ De Software

Para poder instalar este sistema se debe disponer con un sistema operativo superior o igual a de Windows XP, o GNU\Linux en cualquiera de sus distribuciones. Además se requiere tener instalada la máquina virtual de Java 1.4 o superior.

✓ De Hardware

Para instalar la aplicación se necesitan 2 GB de disco duro disponibles, máquinas con 256 MB de RAM y procesadores a 3.0 Ghz.

✓ Restricciones en el Diseño y la Implementación

Se usará como lenguaje de programación Java.
Como herramienta CASE Visual Paradigm para el modelado de los artefactos que se generen en cada uno de los flujos de trabajo.
Se utiliza UML como lenguaje de modelado.
Se implementará en la IDE NetBeans v5.0.
Se utilizan el estándar XML.
Se utilizara la librería de modelado visual BioSysModelador, así como Grafication.

2.4 Definición y descripción de los casos de uso del sistema

2.4.1 Definición de los actores

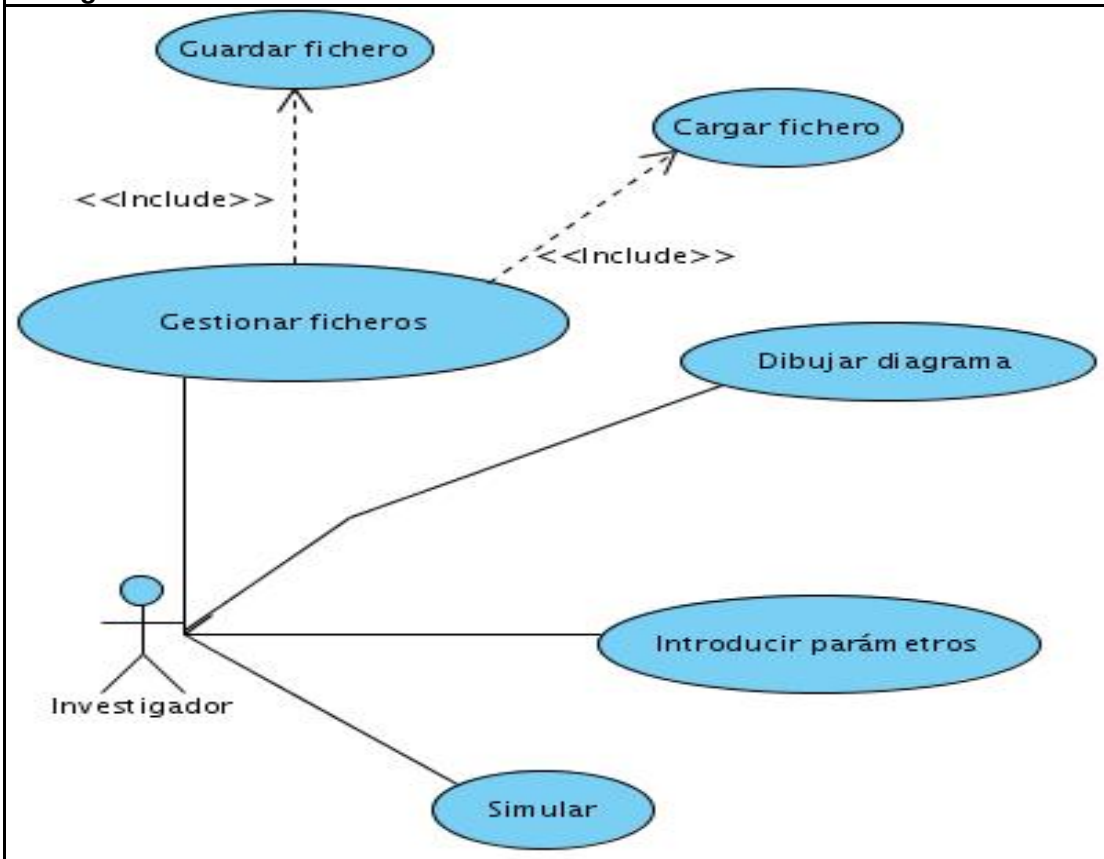
El término actor significa el rol que algo o alguien juega cuando interactúa con el sistema. Un candidato a actor del sistema es cualquier individuo, grupo, organización o máquina que interactúa en los casos de uso. De acuerdo con esta idea un actor del sistema representa un tipo particular de usuario 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. [6]

Actores	**Justificación**
Investigador	Es la persona que interactúa con el sistema. Es el encargado de dibujar el diagrama que define las entidades y los procesos que intervienen en la simulación y los parámetros generales.

2.4.2 Diagrama de casos de uso del sistema

El diagrama de casos de uso del sistema nos ayuda a comprender gráficamente los procesos del sistema y su interacción con los actores.

Diagrama de Casos de Uso del Sistema



2.4.3 Descripción de los casos de uso del sistema

Para entender la funcionalidad asociada a cada caso de uso, no es suficiente con la representación gráfica del Diagrama de casos de uso, por tal motivo se han descrito textualmente cada uno de ellos a continuación.

Caso de Uso	**Realizar simulación**
Actores:	Investigador
Propósito:	Simular como reacciona el sistema inmune bajo determinadas condiciones.
Descripción:	El caso de uso se inicia cuando el investigador desea simular una respuesta inmunitaria y selecciona la opción simular
Referencias:	R2, R3
Precondiciones:	Haber definido las propiedades de cada uno de los agentes (células, químicos y atígenos), los parámetros generales y los procesos que intervienen en la simulación
Postcondiciones	Graficar la simulación y almacenar los resultados en un fichero
Requerimientos especiales:	
Curso Normal de los eventos	
1. El investigador accede a la aplicación para simular la respuesta inmunitaria.	1.1 Muestra la interfaz correspondiente. (Ver anexo 2.1)
2. El investigador define las propiedades del espacio de simulación, las reglas para la difusión así como el repertorio de las entidades	2.1 Interpreta el diagrama que representa las relaciones entre las células, las moléculas y los procesos previamente definidos así como sus propiedades, valores y comportamientos. 2.2 Crea el espacio de simulación según las propiedades definidas por el usuario 2.3 Ejecuta la simulación según las reglas lógicas que se derivan de la interpretación del diagrama 2.4 Muestra mediante gráficas el comportamiento de las entidades una vez concluida la simulación.

Caso de Uso	**Gestionar ficheros**
Actores:	Investigador
Propósito:	Guardar en fichero el resultado de la simulación o el diagrama que define las entidades, los procesos y los órganos que intervienen; así como cargar un fichero del mismo tipo de los antes mencionados.
Descripción:	El caso de uso se inicia cuando el investigador desea abrir un fichero que contenga una simulación que desee modificar o guardar una simulación concluida o no
Referencias:	R4
Precondiciones:	
Postcondiciones	Se creará un fichero con la información guardada o el sistema cargará una simulación
Requerimientos especiales:	El fichero debe tener el formato requerido

Curso Normal de los eventos	
Acciones del Actor	Respuesta del sistema
El investigador en la aplicación tiene la opción de salvar o guardar un fichero	Si decide cargar un fichero ver la Sesión 1, si decide salvar un fichero ver la Sesión 2 (Ver Anexo 1.1, 1.2)
Sesión 1	
1. El investigador abre la aplicación y decide cargar un fichero	1.1 Pide la localización del fichero
2. El investigador define la localización del fichero a cargar. 3. Selecciona el fichero a cargar	3.1 Chequea que la localización del fichero sea válida. 3.2 Chequea que el formato del fichero sea válido. 3.3 Carga el fichero en la aplicación mostrando su contenido
Flujo alterno de eventos	
	3.1 Si la localización del fichero no es accesible muestra un mensaje pidiendo una correcta 3.2 Si el formato del fichero no es válido muestra un mensaje pidiendo uno correcto.
Sesión 2	
1 El investigador decide salvar los datos de una simulación en un fichero.	1.1 Pide la localización para salvar el fichero
2 Determina una localización para salvar el fichero	2.1 Chequea que la localización sea válida 2.2 Guarda el fichero
Flujo alterno de eventos	
	2.1 Si la localización determinada no es accesible muestra un mensaje pidiendo una correcta

Caso de Uso	**Insertar parámetros generales**
Actores:	Investigador
Propósito: Introducir parámetros generales de la simulación	
Descripción: El caso de uso se inicia cuando el investigador decide introducir los parámetros generales de la simulación.	
Referencias:	R5
Precondiciones:	
Postcondiciones:	

Requerimientos especiales:	
Curso Normal de los eventos	
Acciones del Actor	**Respuesta del sistema**
1. El investigador decide introducir los parámetros generales de la simulación.	1.1 Muestra la interfaz correspondiente (Ver Anexo 1.3)
2. El investigador introduce los parámetros generales de la simulación.	2.1 El sistema chequea la validez de los datos 2.2 Si los datos están correctos el sistema los almacena.
Curso alterno de eventos	
	2.1 Si los datos no son válidos muestra un mensaje de notificación y no guardará los parámetros hasta que no sean válidos.

CU-1	Representar diagrama
Actores	Administrador
Descripción: El administrador es el encargado de crear un diagrama donde represente las entidades, los procesos entre ellas y sus propiedades. La herramienta de modelado le permite al usuario un grupo de opciones como la de representar componentes, células, receptores, químicos y relacionarlos según las asociaciones definidas en la mismas de manera gráfica. (Ver Anexo 1.4)	
Referencias	R1

Conclusiones

En este capítulo se determinó un modelo para simular el SI de manera genérica. Se hizo además una descripción de las características del sistema a través de la modelación del negocio propuesto: el enuncio de las reglas del negocio que deben ser consideradas, la identificación de los actores,

trabajadores y los casos de uso correspondientes, de los cuales se realizó su correspondiente descripción textual y los diagramas necesarios para su correcta modelación; el planteamiento de los requisitos funcionales y no funcionales de la aplicación que se va a desarrollar y la modelación de la misma en términos de casos de uso de sistema.

Capítulo 3 Análisis y Diseño

Introducción

En este capítulo se presenta el diagrama de clases del análisis y del diseño, constituyendo este último el artefacto más importante del presente flujo de trabajo según la metodología RUP y para el cual se tuvo en cuenta los patrones GRASP. También se presentan los diagramas de secuencia del diseño así como sus descripciones correspondientes.

3.1 Análisis

El modelo de análisis ayuda a refinar los requisitos y permite razonar sobre los aspectos internos del sistema, ya que se escribe utilizando el lenguaje de los desarrolladores. Puede considerarse como una primera aproximación al modelo de diseño, y es por tanto, una entrada fundamental cuando se da forma al sistema en el diseño y la implementación.
Por su parte, una clase del análisis representa una abstracción de una o varias clases y/o subsistemas del diseño del sistema y siempre encajan en uno de tres estereotipos básicos: de interfaz, de control o de entidad (cada estereotipo implica una semántica específica). [6]

Diagrama de clase del análisis: CU Introducir Parámetros Generales

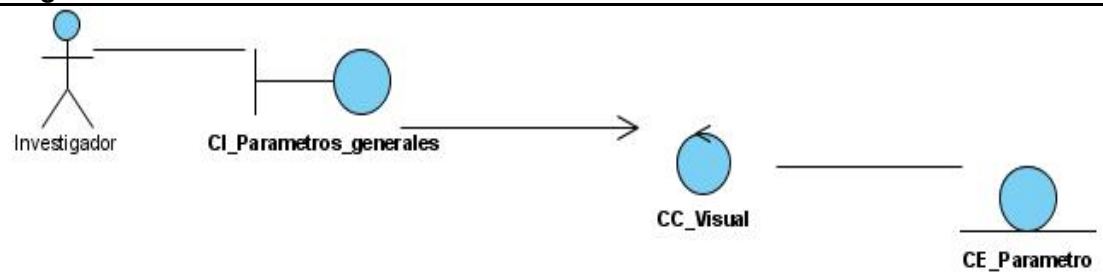


Diagrama de clase del análisis: CU Gestionar fichero

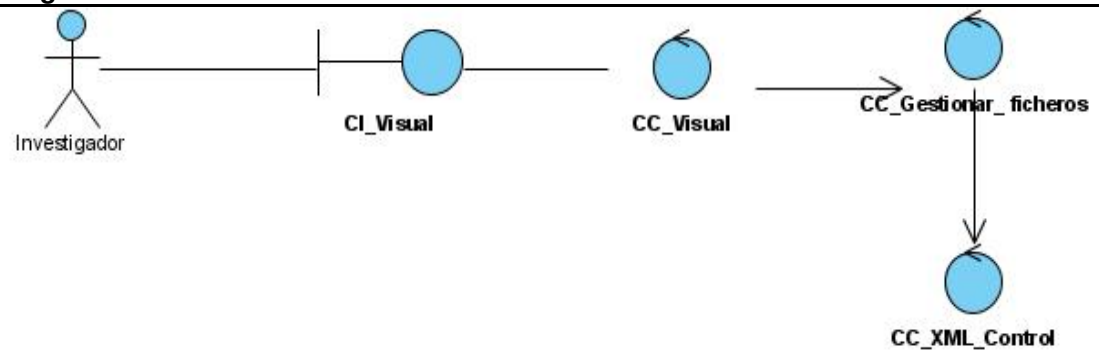
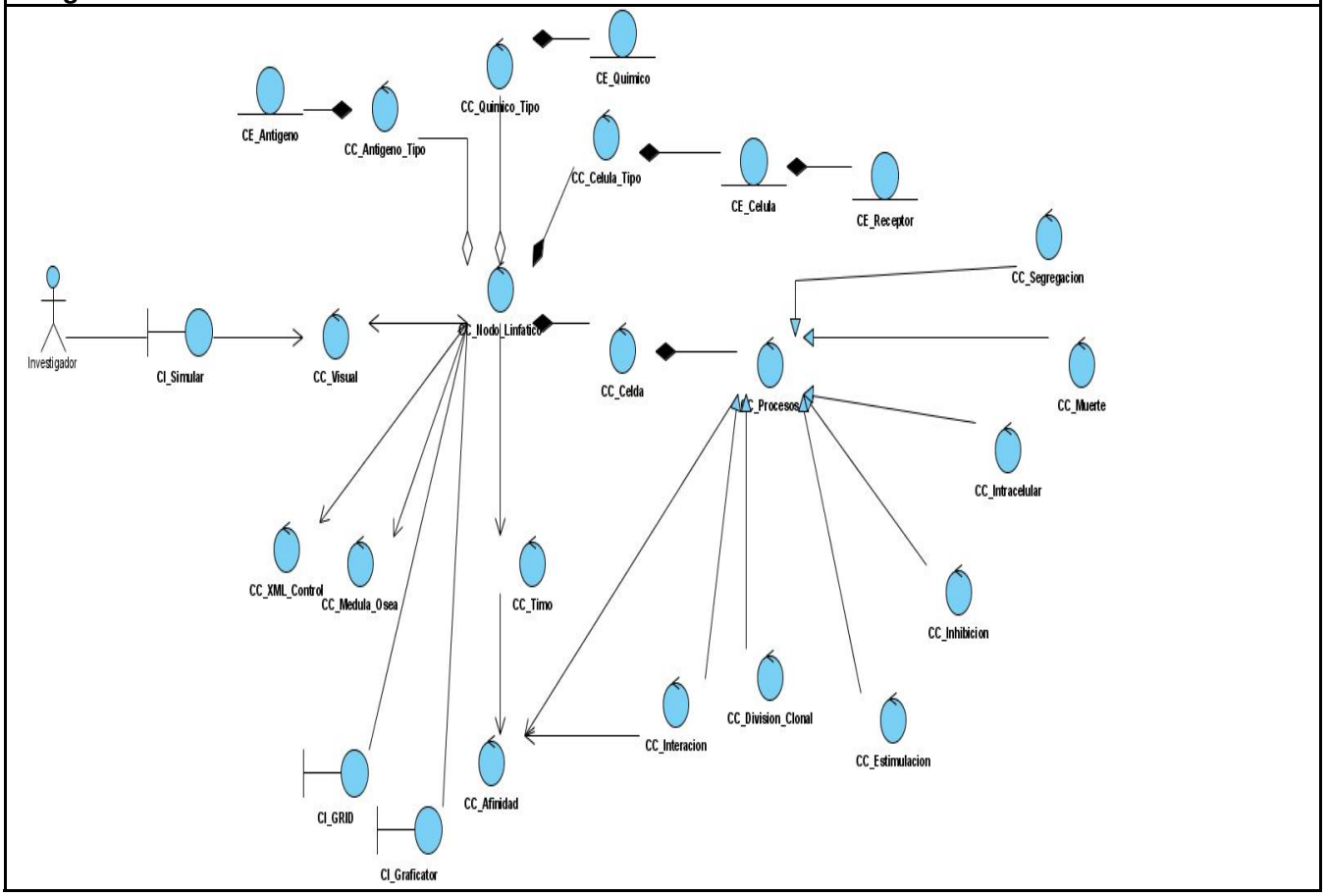


Diagrama de clase del análisis: CU Simular



3.2 Diseño

3.2.1 Patrones de diseño

Definición de un patrón de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Identifica: Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades.

Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades). Estos fueron los patrones que se aplicaron para realizar el diseño de la aplicación que propone la investigación.

- ✓ **Experto**: Es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen.
- ✓ **Creador**: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.
- ✓ **Alta Cohesión**: Asignar una responsabilidad de modo que la cohesión siga siendo alta. Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. La ventaja que significa una gran funcionalidad también soporta un aumento de la capacidad de reutilización.

- ✓ **Bajo Acoplamiento**: asignar una responsabilidad para mantener bajo acoplamiento (una clase con bajo (o débil) acoplamiento no depende de muchas otras; una clase con alto (o fuerte) acoplamiento recurre a muchas otras.). Este patrón estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento.

- ✓ **Controlador**: asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. La mayor parte de los sistemas reciben eventos de entrada externa, En cualquiera de los casos que puedan existir, si se recurre a un diseño orientado a objetos, hay que elegir los controladores que manejen esos eventos de entrada.

3.2.3 Diagrama de clases del diseño

Una clase del diseño es una abstracción sin costuras de una clase o construcción similar en la implementación del sistema. Es decir, el lenguaje utilizado para especificar una clase del diseño es lo mismo que el lenguaje de programación; los métodos de una clase del diseño tienen correspondencia con directa con el correspondiente método en la implementación de las clases; una clase de diseño puede realizar interfaces…entre otras características. [6]

Diagrama de clase del diseño: Gestionar ficheros

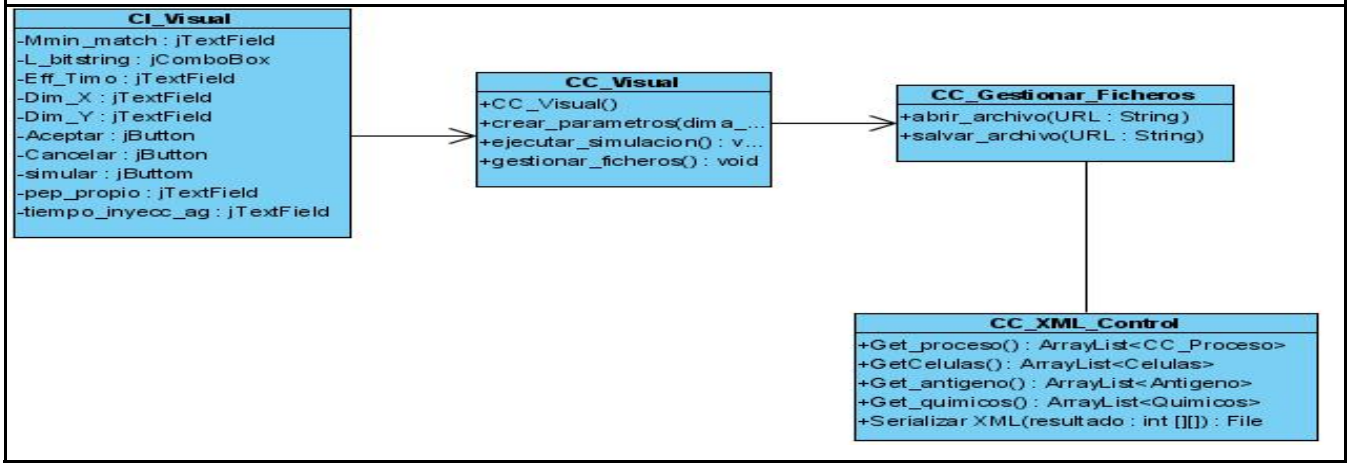


Diagrama de clase del diseño: CU Gestionar parámetros

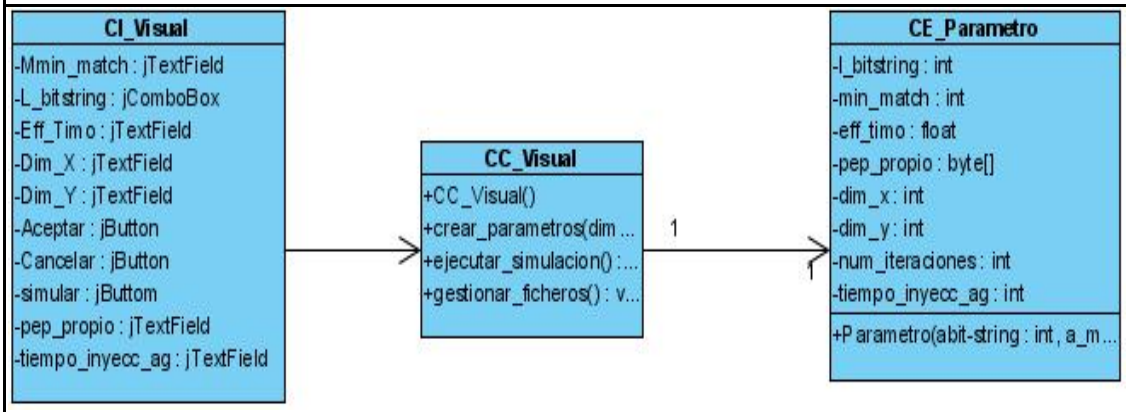
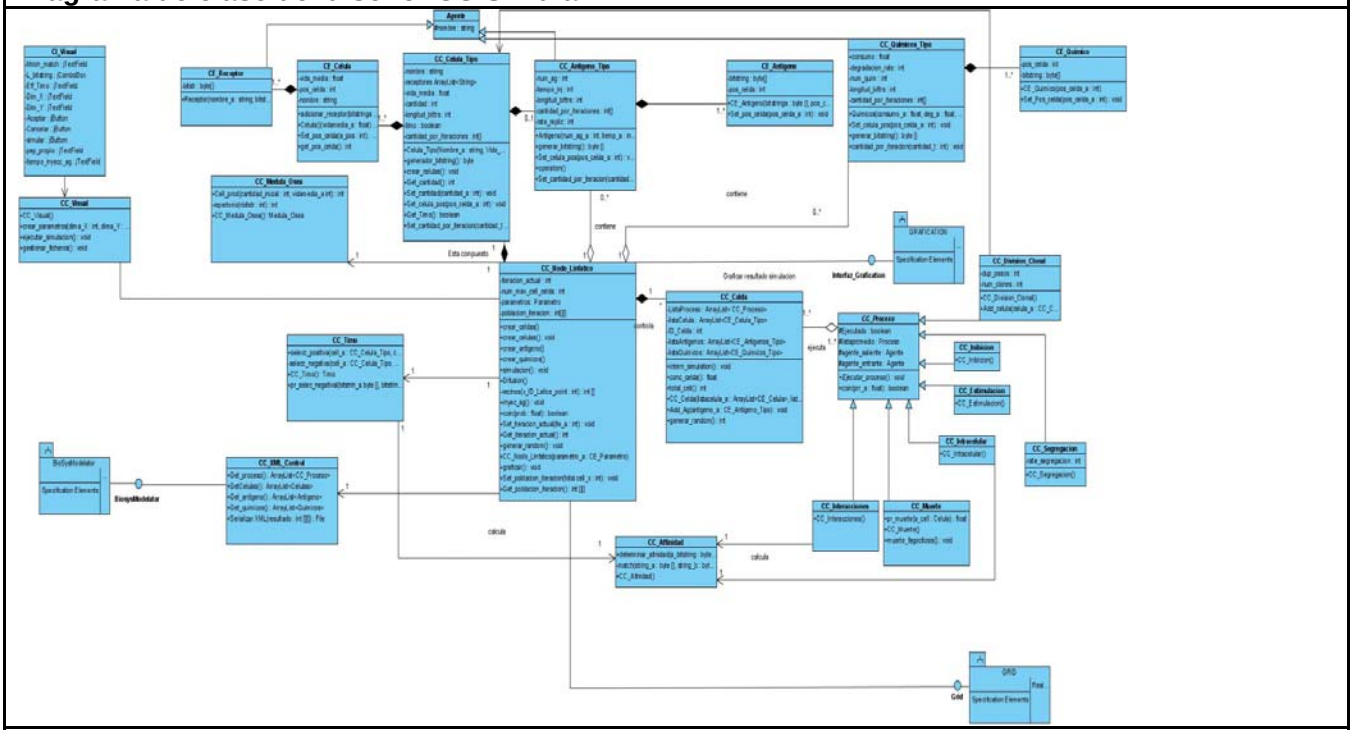


Diagrama de clase del diseño: CU Simular



Como el diagrama de clases correspondiente al CU Simular está bastante ilegible a continuación lo mostramos dividido por subdiagramas que contienen clases del diagrama estrechamente relacionadas.

Diagrama de clase del diseño: CU Simular (Subdiagrama 1)

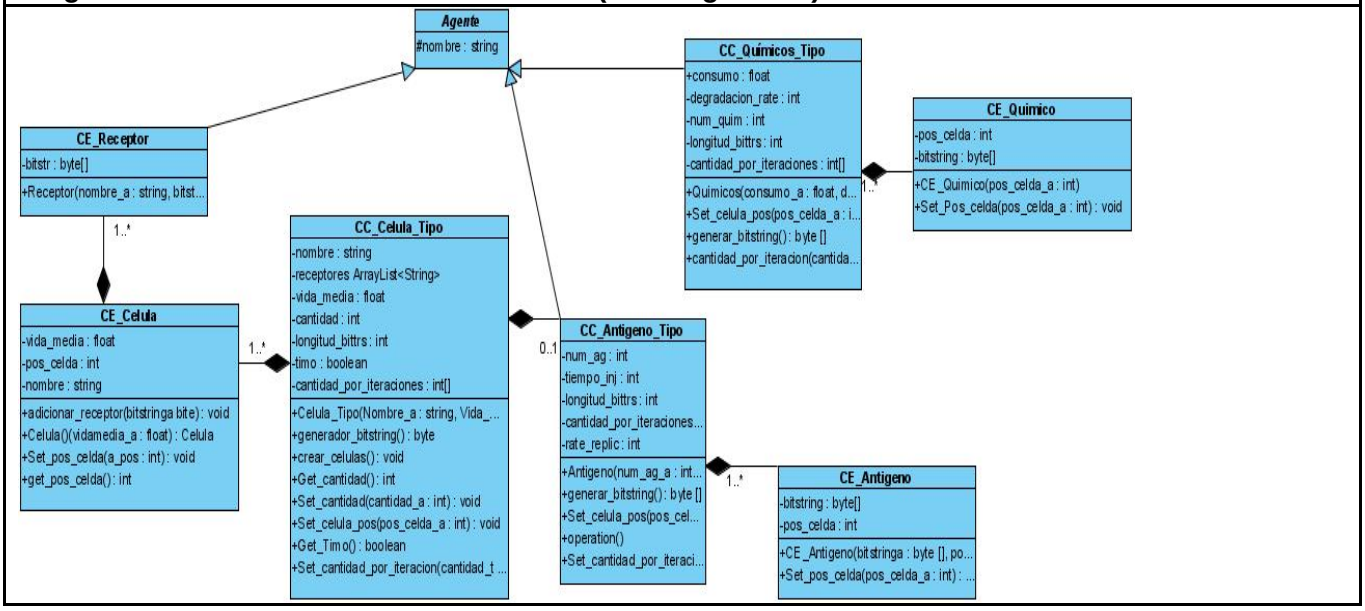


Diagrama de clase del diseño: CU Simular (Subdiagrama 2)

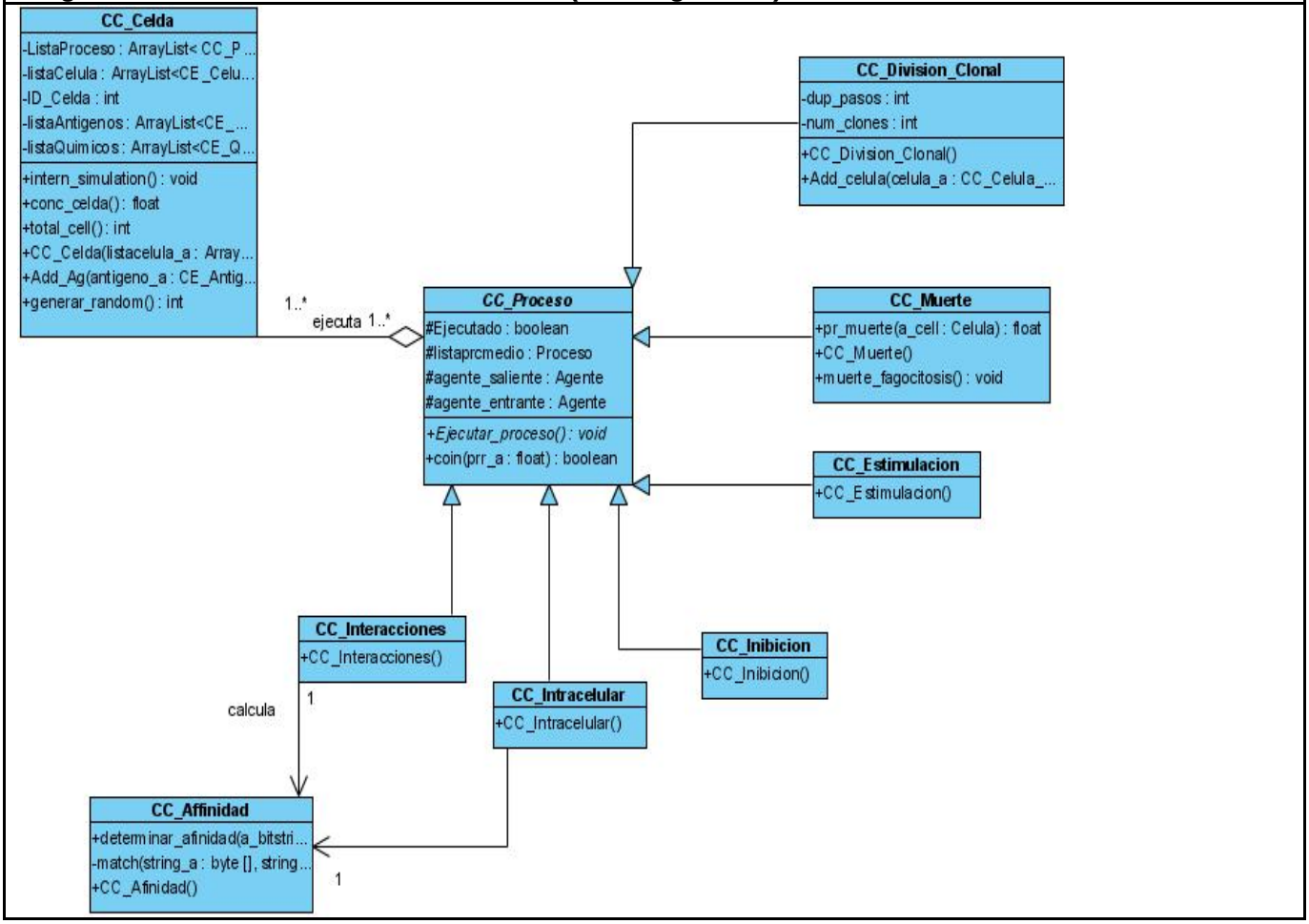
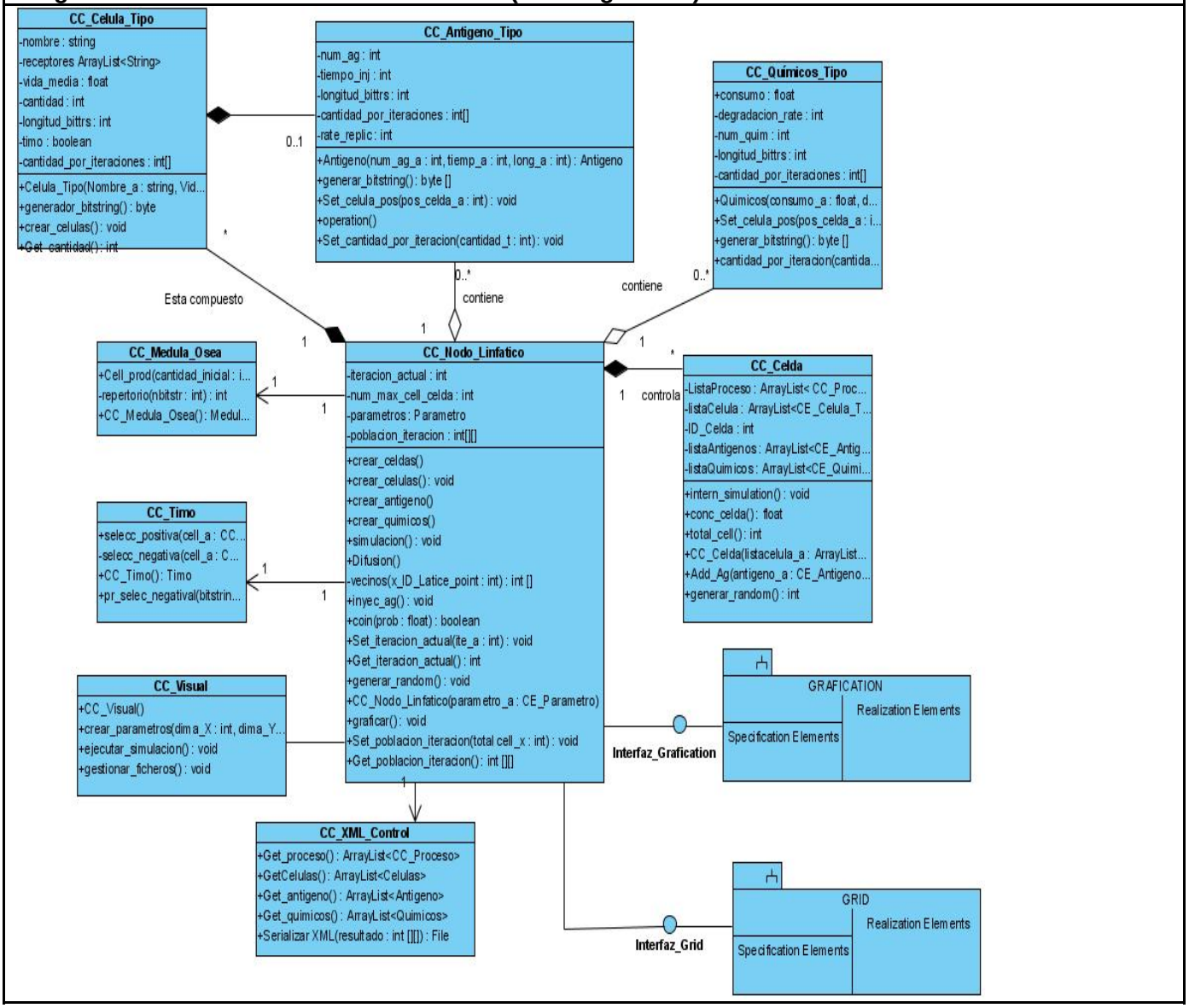


Diagrama de clase del diseño: CU Simular (Subdiagrama 3)



3.2.4 Diagrama de interacción del diseño

Los diagramas de interacción son modelos que describen cómo grupos de objetos colaboran para conseguir algún fin. Estos diagramas muestran objetos, así como los mensajes que se pasan entre ellos dentro del caso de uso, cuyo comportamiento capturan.

Los diagramas de interacción se expresan de dos maneras: diagramas de secuencia y diagramas de colaboración. El Diagrama de Secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia se modela para cada caso de uso. Mientras que el diagrama de caso de uso permite el modelado de una vista del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes pasados entre los objetos.
Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como vectores horizontales.

Diagrama de secuencia del diseño: Introducir parámetros

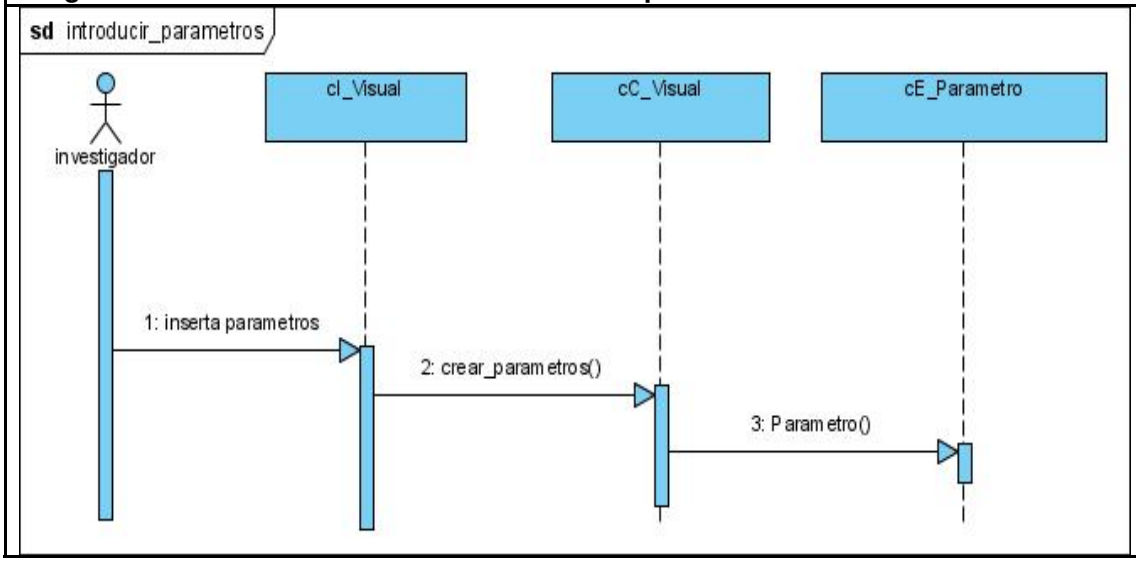


Diagrama de secuencia del diseño: Abrir fichero

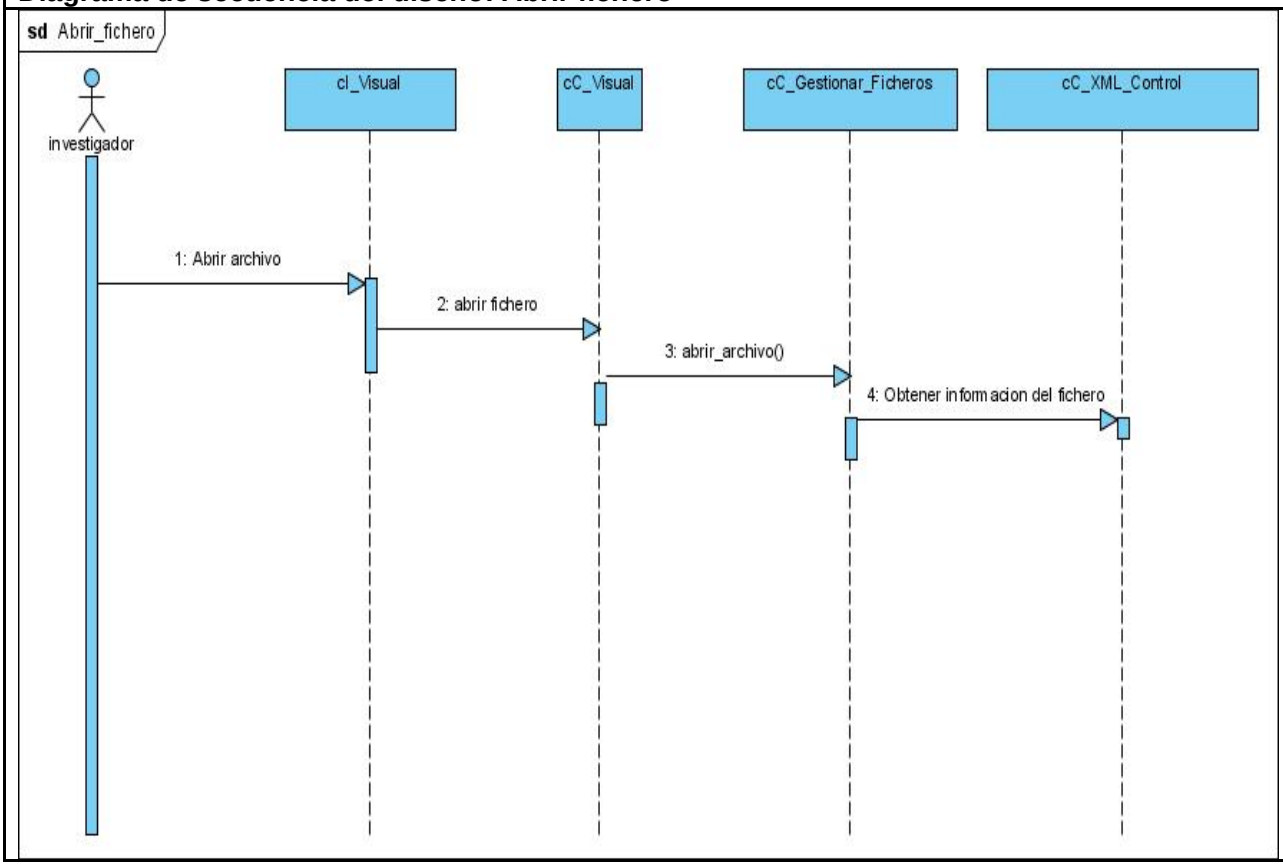


Diagrama de secuencia del diseño: Guardar ficheros

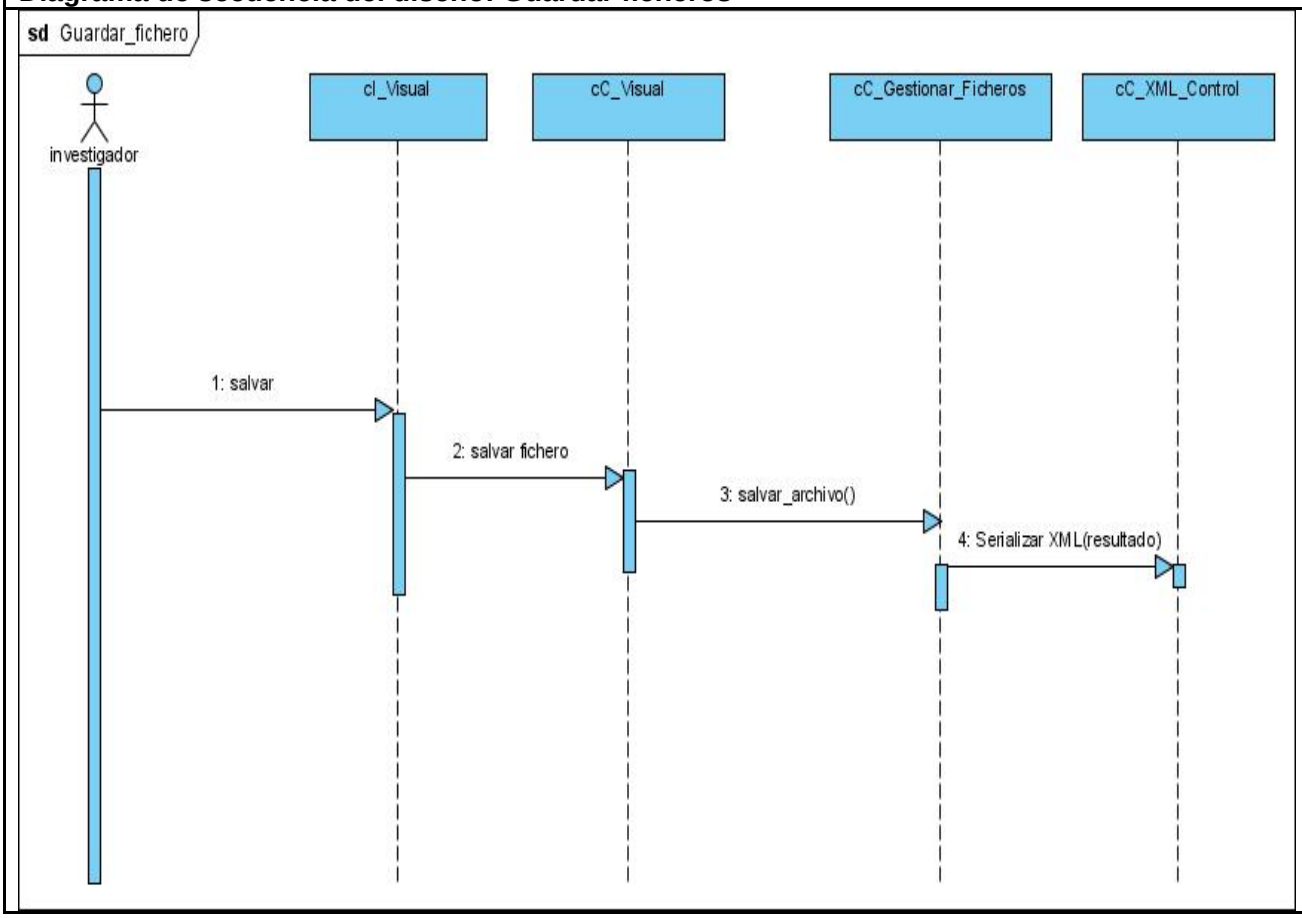


Diagrama de secuencia del diseño: Flujo alterno para diferenciar en el timo

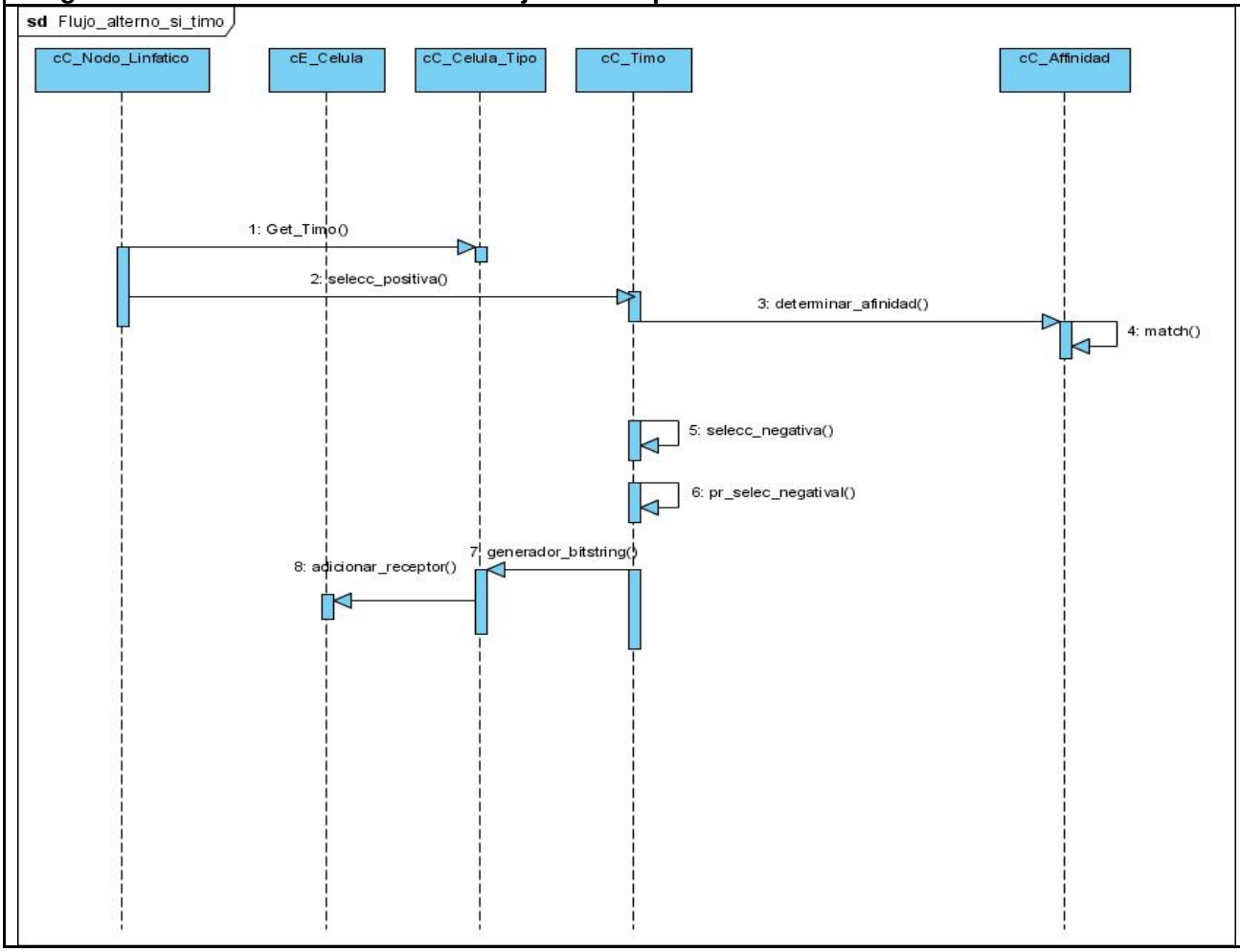


Diagrama de secuencia del diseño: Simular (Subdiagrama 1)

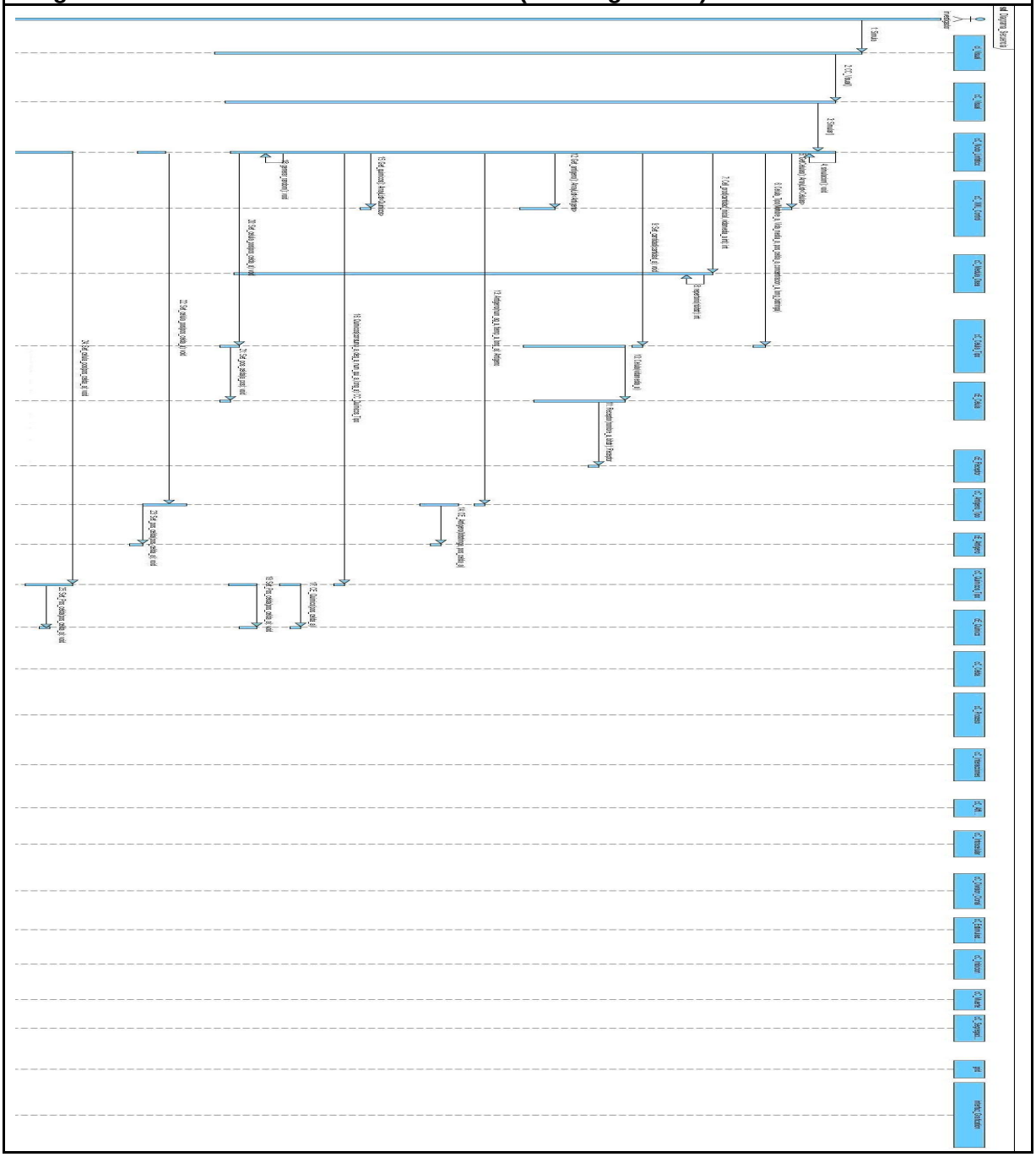
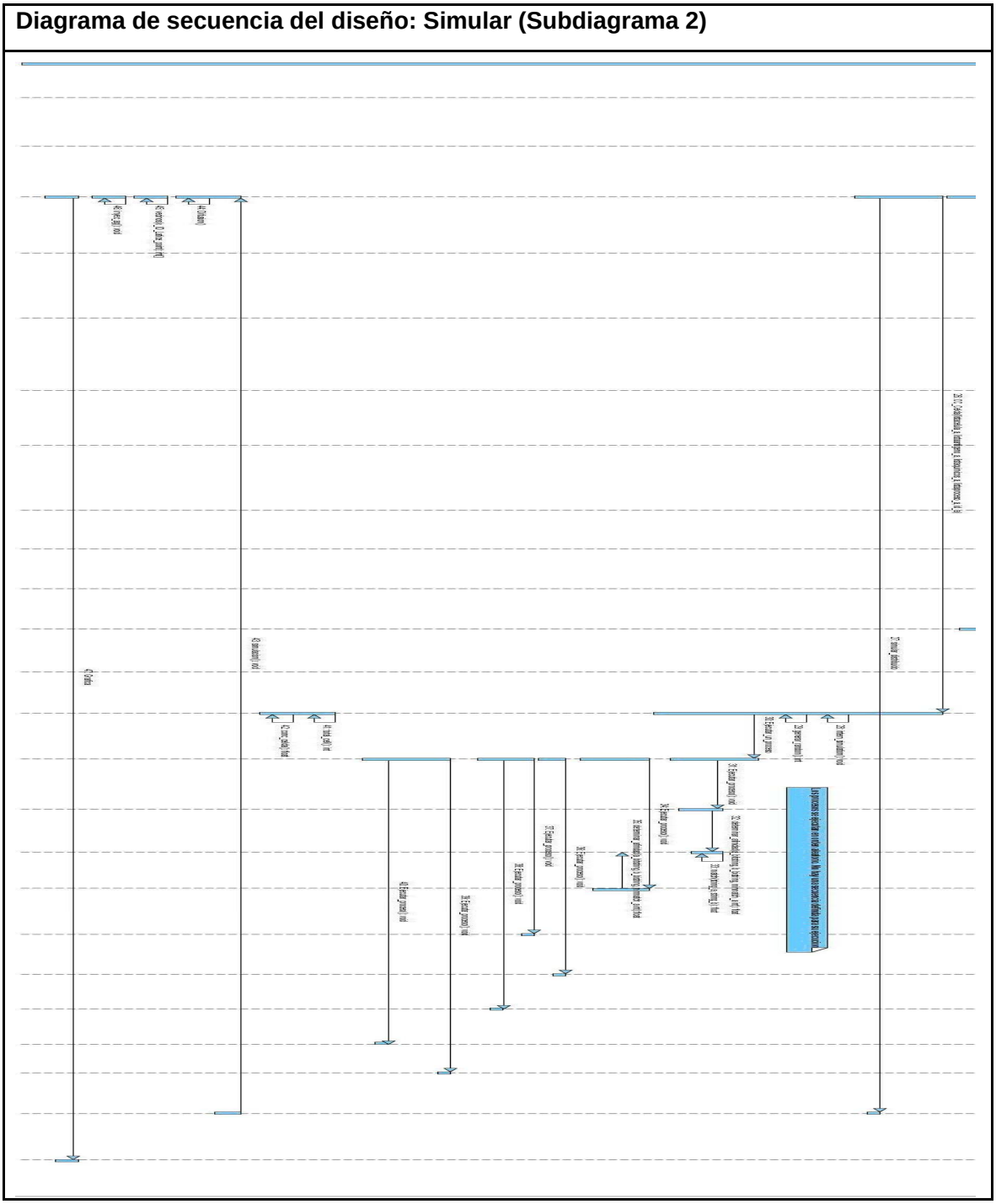


Diagrama de secuencia del diseño: Simular (Subdiagrama 2)



3.2.5 Descripción de las clases del diseño

Nombre: CC_Afinidad	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Determinar_afinidad	Determinar la afinidad entre dos entidades
match	Determina el numero de bits que matchean entre 2 bist-string
Afinidad	Constructor de la clase

Nombre: CE_Receptor	
Tipo de clase: Entidad	
Atributo	**Tipo**
nombre	string
bitstr	byte
Para cada responsabilidad:	
Receptor	Constructor de la clase

Nombre: CE_Celula	
Tipo de clase: Entidad	
Atributo	**Tipo**
nombre	String
vida media	Flota
pos_celda	Int
concentración	float
Para cada responsabilidad:	
adicionar_receptor	Construye receptor
Celula	Constructor célula
Set_pos_celda(a_pos:int)	Asigna una celda a la célula
Get_pos_celda()	Obtiene la celda en que se encuentra

Nombre: CC_Medula_ósea	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Cell_prod	Calcula la cantidad de células que pueden ser generadas
Repertorio	Determina el repertorio de células posibles
Cc_medula_osea()	Constructor de la medula ósea

Nombre: CC_XML_Control	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Get_celula	Permite obtener la información sobre las células que se obtuvo del XML
Get_proceso	Permite obtener la información sobre los procesos que se obtuvo del XML
Get_Antigeno	Permite obtener la información sobre los antígenos que se obtuvo del XML
Get_Quimico	Permite obtener la información sobre los químicos que se obtuvo del XML
CC_XML_Control	Constructor de la clase

Nombre: CC_Timo	
Tipo de clase: Controladora	
Atributo	**Tipo**
tymEff	float
Para cada responsabilidad:	
Selecc_positiva	Elimina las células que no reaccionen con determinada célula en el Timo
Selecc_negativa	Elimina las células que sobre reaccionen a lo propio
rr_selec_negativa	Reasigna receptor a las nuevas células que allí se generan.
CC_Timo()	Constructor de la clase Timo

Nombre: CE_Quimicos	
Tipo de clase: controladora	
Atributo	**Tipo**
Pos_celda	int
Bitstring	byte
Para cada responsabilidad:	
CE_Quimico	Constructor de la clase
Set_Pos_celda	Asigna la posición de un químico en una celda

Nombre: CE_Antigeno	
Tipo de clase: Entidad	
Atributo	**Tipo**
Bitstring	Byte
Pos_celda	Int
Para cada responsabilidad:	
CE_Antígeno	Constructor de la clase.
Set_pos_celda_a	Asigna la posición de un antígeno en una celda

Nombre: CC_intracelular	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
CC_Intracelular()	Constructor de la clase

Nombre: CC_estimulación	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
CC_Eestimulación	Constructor de la clase

Nombre: CC_Celda	
Tipo de clase: Controladora	
Atributo	**Tipo**
listaProceso	**ArrayList**
listaCelula	**ArrayList**
listaAntigenos	**ArrayList**
listaQuiicos	**ArrayList**
id_celda	int
Para cada responsabilidad:	
intern_simulacion	Simula los procesos internos de la celda
conc_celda	Determina la concentración de la celda
total_cell	Determina el número total de células en la celda
CC_Celda	Constructor de la clase
add_Ag	adiciona la inyección de un antígeno e un tiempo t
generar_random	Genera número aleatorio

Nombre: CC_inibicion	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
CC_inibicion	Constructor de la clase

Nombre: CC_Nodo_linfatico	
Tipo de clase: Controladora	
Atributo	**Tipo**
parametros	parametro
iteración_actual	int
num_max_cell_celda	int
población iteracion	Int[][]
Para cada responsabilidad:	
crear_ celdas	Crea las celdas.
crear_ celulas	Crea las células_tipo.
crear_ antigeno	Crea los antígenos_tipo
vecinos	Determina los vecinos para una celda dada
crear_ quimicos	Crea los químicos_tipo.
inyec_ag	Controla la inyección de un antígeno en un tiempo t.
coin	Determina si determinado evento se ejecuta para determinada probabilidad
Set_iteracion_actual	Actualiza la iteración actual
Get_iteracion_actual	Determina la iteración actual
Generar_random	Genera número aleatorio
CC_Nodo_Linfatico	Constructor de la clase
graficar()	Cuando finaliza la simulación envía los datos de cada iteración (poblaciones al final de cada iteración) a graficar
simulacion	Metodo principal que controla la simulacion
difusión	Determina la probabilidad de difusión y difunde los agentes entre las celdas vecinas

Nombre: CC_Division_clonal	
Tipo de clase: Controladora	
Atributo	**Tipo**
Dup_pasos	Int
Num_clones	int
Para cada responsabilidad:	
CC_Division_clonal()	Constructor de la clase
add_celula	Cuando una célula es clonada la adiciona a la lista de células.

Nombre: CC_interaciones	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
CC_interacciones()	Constructor de la clase

Nombre: CC_muerte	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
CC_Muerte	Constructor de la clase
Pr_muerte()	Determina la probabilidad de que una célula muera por envejecimiento
Muerte_Fagocitosis()	Elimina el antigeno que se encuentra dentro de la célula

Nombre: CC_proceso	
Tipo de clase: Controladora	
Atributo	**Tipo**
#Ejecutado	Boolean
#listaprcmedio	Proceso
#agente_saliente	agente
#agente_entrante	agente
Para cada responsabilidad:	
Ejecutar_proceso	Método virtual que se implementa en cada una de las clases hijas de acuerdo a la función particular de cada una de ellas
Coin(prr_a:float)	Método virtual que se implementa en cada una de las clases hijas para determinar la probabilidad de ocurrencia del proceso.

Nombre: CI_Visual	
Tipo de clase: Interfaz	
Atributo	**Tipo**
Mmin_match	jtextField
L_bitstring	jComboBox
Eff_Timo	jtextField
Dim_x	jtextField
Dim_y	jtextField
Aceptar	jButton
Cancelar	jButton
simular	jButton
Pep_propio	jtextField
Tiempo_inyecc	jtextField

Nombre: CC_Visual	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
CC_Visual	Constructor de la clase
crear_parametros	Crea la clase parámetros con los datos que define el usuario
ejecutar_simulacion()	Llama al método simulación de la clase Nodo Linfático
gestionar_ficheros()	Se encarga de llamar a la clase Gestionar Ficheros

Nombre: CC_Segregacion	
Tipo de clase: Controladora	
Atributo	**Tipo**
Rate_segregacion	int
Para cada responsabilidad:	
CC_Segregacion	Constructor de la clase

Nombre: CC_Antigeno_tipo	
Tipo de clase: Controladora	
Atributo	**Tipo**
Num_ag	int
Longuitud_bittrs	int
Cantidad_por_iteraciones	Int[]
Tiempo_inj	int
Para cada responsabilidad:	
Antigeno	Constructor de la clase
generar_bitstring()	Genera bit_string aleatorios para asignar a cada uno delos antígenos que se creen.
Set_celula_pos	Asigna celda a cada antígeno
Set_cantidad_por_iteracion	Almacena la cantidad de antígenos generados en cada iteración.

Nombre: CC_Celula_tipo	
Tipo de clase: Controladora	
Atributo	**Tipo**
nombre	string
receptores	ArrayList<String>
cant	int
timo	boolean
cantidad_por_iteraciones	Int[]
Para cada responsabilidad:	
CC_Celula_tipo	Constructor de la clase
generardor_bitstring()	Genera bit_string aleatorios para asignar a cada uno delos antígenos que se creen.
Set_celula_pos	Asigna celda a cada célula
Set_cantidad_por_iteracion	Almacena la cantidad de células generadas en cada iteración.
Get_cantidad	Permite acceder a la cantidad total de células por tipos
Set_celula_pos	Asigna posición a cada célula en una celda determinada
Get_timo	Permite saber si la célula se diferencia o no en el timo

Nombre: CC_Quimicos_tipo	
Tipo de clase: Controladora	
Atributo	**Tipo**
nombre	string
Longuitud_bittrs	int
Cantidad_por_iteraciones	Int[]
Num_quim	int
Para cada responsabilidad:	
CC_Quimicos_tipo	Constructor de la clase
generar_bitstring()	Genera bit_string aleatorios para asignar a cada uno delos quimicos que se creen.
Set_celula_pos	Asigna celda a cada químico
Set_cantidad_por_iteracion	Almacena la cantidad de químicos generados en cada iteración.

Nombre: CC_gestionar_ficheros	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Abrir_archivo(URL:String)	Carga un fichero XML con los datos de una simulación
Salvar_archivo(URL:String)	Guarda Carga un fichero XML con los datos de una simulación

Conclusión

En el capitulo que concluye se mostraron los diagramas del análisis, del diseño (para los cuales se utilizaron patrones GRASP) y de secuencia del diseño, correspondientes a la herramienta que propone la investigación.

Conclusiones Generales

En la investigación, a partir de un estudio de las herramientas existentes que simulan el SI, se determinó un modelo que permitiera flexibilizar su estudio mediante una herramienta computacional. Sobre la base de este modelo y del modelo de dominio, (es decir los principales conceptos sobre el estudio experimental del SI en cuestión) se determinaron los requerimientos funcionales a partir de los cuales se realizó el análisis y el diseño de una herramienta para simular el SI.
Por lo que podemos llegar a la siguiente conclusión general:

- ✓ Se realizó el análisis y diseño de una herramienta para la simulación del Sistema Inmune

Y específicamente:

- ✓ Se determinó un modelo genérico que permita simular el sistema inmune.
- ✓ Se realizó el análisis de una herramienta para la simulación del sistema inmune.
- ✓ Se realizó el diseño de una herramienta para la simulación del sistema inmune.

Recomendaciones

La investigación cumplió con los objetivos propuestos, pero debido a la complejidad propia del SI, algunas funcionalidades requieren ser estudiadas y añadidas al modelo y a la herramienta. Por lo que se recomienda lo siguiente:

✓ Implementar la herramienta que fue diseñada en la investigación para que finalmente sea de utilidad a los investigadores para el estudio del SI.

✓ Profundizar más en la simulación del SI mediante herramientas computacionales.

✓ Perfeccionar el modelo que propuso la investigación agregando la hipermutación de anticuerpos y la maduración de la afinidad.

✓ Añadir funcionalidades que no fueron incluidas en la investigación como la simulación de la respuesta inmune ante el virus del VIH-SIDA.

Bibliografía

Conferencias

- UCI. Fase de Inicio. Flujo de trabajo de requerimientos (2005-2006).
- UCI. Flujo de Análisis y Diseño. Modelo de Análisis. (2005-2006).
- UCI. Flujo de trabajo Análisis & Diseño. (Modelo de diseño). (2005-2006).
- UCI. Patrones de diseño, (2005-2006).

Libros

- Goldsby R. A., Kindt T. J., Osborne B. A., Kuby J.; Immunology 5 Edición 2003.
- Colectivo de autores. Inmunología celular y molecular Primer Parte.
- Jacobson, Ivar y Booch, Grady y Rumbaugh, James. El proceso unificado de software. Primera edición. Pearson Educación, S.A. 2000.

Tesis

- De Camino Beck, Tomás. Un lenguaje para la especificación de autómatas celulares con aplicaciones
- Diaz, Dayanara T; Lechiguero Cristian C. “Prototipo basado en agentes inteligentes para la identificación de especies del género Pacinun, familia de las gramíneas”. Universidad Central de Venezuela. Venezuela, 2003. 128
- Karel Osorio Ramirez “Plataforma Computacional para el Desarrollo de la Biología de Sistemas”. Facultad de Matemátic -Computación Universidad de la Habana, Ciudad de la Habana, 2004. 64

Artículos

- D.G. Malleta, L.G. De Pillisb, “A cellular automata model of tumor–immune system interactions”, *Journal of Theoretical Biology*, Diciembre 2006, p.334–350 [Consulta: Abril, 2007; Disponible en ttp://eprints.qut.edu.au/archive/00005278/01/5278.pdf]
- Planchart Márquez, Orlando “La Modelación Matemática: alternativa didáctica en la enseñanza de precálculo”, *InterPonce*, Junio 2005 [Consulta: 23 Marzo, 2007; Disponible en http://cremc.ponce.inter.edu/1raedicion/modelacion.htm]
- Stefania Bandini, Sara Manzoni, Giuseppe Vizzari, “Situated Cellular Agents and Immune System Modelling”, Departamento de Informática, Universidad de Milano–Bicocca, Italy, 6 [Consulta: Marzo 2007, Disponible en: http://lia.deis.unibo.it/books/woa2003/pdf/02.pdf
- Tadmor, Brigitta; Tidor, Bruce; “Interdisciplinary research and education at the biology–engineering–computer science interface: a perspective”. Drug Discovery Today. Vol 10, num 17, septiembre 2005 [Consulta: 23 Febrero, 2007, Disponible en: http://csbi.mit.edu/website/whatis/communities/Tadmor_Tidor_DDT_Sept05]
- Lilia Alberghina1 and Anna Maria Colangelo1, “The modular systems biology approach to investigate the control of apoptosis in Alzheimer's disease neurodegeneration”, *PubMed Central [*Consulta: 22 Febrero 2007, Dsiponible en: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1775042]

Referencias Bibliográficas

[1] Goldsby R. A., Kindt T. J., Osborne B. A., Kuby J.; Immunology 5 Edición 2003.

[2] Colectivo de autores. Inmunología celular y molecular Primer Parte.

[3] Una Introducción a los Autómatas Celulares [Consulta: 23 Enero, 2007, Disponible en: http://yupana.autonoma.edu.co/publicaciones/yupana/005/autocelular/Automatas.html]

[4] IMMSIM-C [Consulta: Diciembre 13, 2006; Disponible en http://www.immsim.org/]

[5] Monografias.com, UML [Consulado 25,marzo 2007, Disponible en: http://www.monografias.com/trabajos5/insof/insof.shtml]

[6] Jacobson, Ivar y Booch, Grady y Rumbaugh, James. El proceso unificado de software. Primera edición. Pearson Educación, S.A. 2000.

[7]. Evans, Gary. “A simplified approach to RUP” http://www-106.ibm.com/developerworks/ rational/library/354.html

[8] http://www.infosgroup.com/paginas/v4/publico/soluciones/soluciones_producto/rational/

[9] ApexNet, Comparación de Herramientas de modelado UML: Enterprise Architect y Rational Rose [Consultado 29,marzo 2007, Disponible en ttp://www.apexnet.com.ar/index.php/news/main/38/event=view]

[10] Monografias.com, Ingeniería de SoftwareUML [Consultado: abril, 2007, Disponible en:http://www.monografias.com/trabajos5/insof/insof.shtml]

[11] Torres, Nestor V , Caos en Sistemas Biológicos[Consultado: 16 febrero, 2007; Disponible en: http://webpages.ull.es/users/imarrero/sctm04/modulo2/8/ntorres.pdf]

[12] Open WetWare, Biología Sintética [Consultado 18 febrero 2007, Disponible en: http://openwetware.org/wiki/Biolog%C3%ADa_Sint%C3%A9tica /

[13] Planchart Márquez, Orlando “La Modelación Matemática: alternativa didáctica en la enseñanza de precálculo”, InterPonce, Junio 2005 [Consulta: 23 Marzo, 2007; Disponible en http://cremc.ponce.inter.edu/1raedicion/modelacion.htm]

[14] Karel Osorio Ramirez “Plataforma Computacional para el Desarrollo de la Biología de Sistemas”. Facultad de Matemática -Computación Universidad de la Habana, Ciudad de la Habana, 2004. 64

[15] De Camino Beck, Tomás. Un lenguaje para la especificación de autómatas celulares con aplicaciones en biología. Instituto tecnológico de Costa Rica, 2000.

[16] Netbeans, Basic IDE Concepts [Consulta: 23 Abril, 2007, Disponible en: http://www.netbeans.org/kb/55/using-netbeans/project_setup.html#pgfld-1020595]

[17] Eclipse, Eclipse in Action: A Guide for the Java Developer [Consulta: 23 Abril, 2007; Disponible en: http://www.eclipse.org/resources/]

[18] Java en Castellano [Consulta: 24 Abril, 2007, Disponible en: http://javascript.programacion.net/java/]

Basic Java Programming Learning Trail http://www.netbeans.org/kb/trails/java-se.html

[19] Practicas de Ingeniería de Software, Una herramienta CASE para ADOO: Visual Paradigm [Consulta: marzo 29, 2007: Disponible en: http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr1_VP.pdf.]

[20] Visual Paradigm, Why Visual Paradigm for UML? [Consulta: 24 Abril, 2007, Disponible en: http://www.visual-paradigm.com/product/vpuml/]

[21] NetBeans 3.51, Guía Rápida [Consulta: 24 Abril 2007, Disponible en: http://www.jorgesanchez.net/programacion/NetBeans.pdf]

[22] Clikear, UML [Consulta: 23 Abril, 2007 Disponible en: http://www.clikear.com/manuales/uml/introduccion.asp]

[23] Agapea [Consulta: 19 marzo, 2007, Disponible en:http://www.agapea.com/Eclipse-3-para-desarrolladores-Java-n228608i.htm]

[24] INCO: Instituto de Computación, Introducción Eclipse [Consulta: Abril 2007, Disponible en: www.fing.edu.uy/inco/grupos/coal/investigacion/lead/docs/IntroduccionEclipse.pdf]

[25] Monografias.com [Consulta: 27 abril, 2007, Disponible en: http://www.monografias.com/trabajos5/visualcurso/visualcurso.shtml#intro21]

[26] Junta de Castilla y Leon, Guía de Iniciación al Lenguaje Java, [Consulta: Abril 27, 2007, Disponible en: http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/I_3.htm]

[27] H. F. J. Dullens1 , M. W. M. Van Der Tol, R. A. De Weger1 and W. Den Otter1, Abstract: "A survey of some formal models in tumor immunology", *Cancer Immunology Immunotherapy*, November 17, 2004 [Consulta: 27 Marzo, 2007, Disponible en: http://www.springerlink.com/content/r71w2647j730vv33/]

[28] IMMSIM-C [Consulta: Diciembre 13, 2006; Disponible en http://www.immsim.org/]

[29] IMMSIM++ [Consulta: Enero 19, 2007; Disponible en http://www.cs.princeton.edu/immsim/download.html]

[30] Centro de difusión de tecnologías ETSITU-UPM [Consulta: Abril 2007, Disponible en:http://www.ceditec.etsit.upm.es/grid_computing.php?pagina=2]

[31] W3C Site, TecnologiasXML [Consulta: 26 Abril 2007, Disponible en: http://www.w3c.es/Divulgacion/Guiasbreves/TecnologiasXML]

[32] UNINET, Capítulo 8. 1. Inmunidad: Inmunodeficiencias [Consulta : 21 Febrero, 2007, Disponible en : http://tratado.uninet.edu/c080101.html]

[33] University of Virginia, IMMSIM [Consulta: Disponible en: http://www.healthsystem.virginia.edu/uvahealth/peds_infectious_sp/abtis.cfm]

[34] Tadmor, Brigitta; Tidor, Bruce; "Interdisciplinary research and education at the biology–engineering–computer science interface: a perspective". Drug Discovery Today. Vol 10, num 17, septiembre 2005 [Consulta: 23 Febrero, 2007, Disponible en: http://csbi.mit.edu/website/whatis/communities/Tadmor_Tidor_DDT_Sept05]

[35] Lilia Alberghina1 and Anna Maria Colangelo1, "The modular systems biology approach to investigate the control of apoptosis in Alzheimer's disease neurodegeneration", *PubMed Central [*Consulta: 22 Febrero 2007, Dsiponible en: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1775042]

[36] Diaz, Dayanara T; Lechiguero Cristian C. "Prototipo basado en agentes inteligentes para la identificación de especies del género Pacinun, familia de las gramíneas". Universidad Central de Venezuela. Venezuela, 2003. 128

Anexos

Anexo 1 Visualización de algunos conceptos sobre el funcionamiento del Sistema Inmune.

A1.1 Respuesta mediada por células.

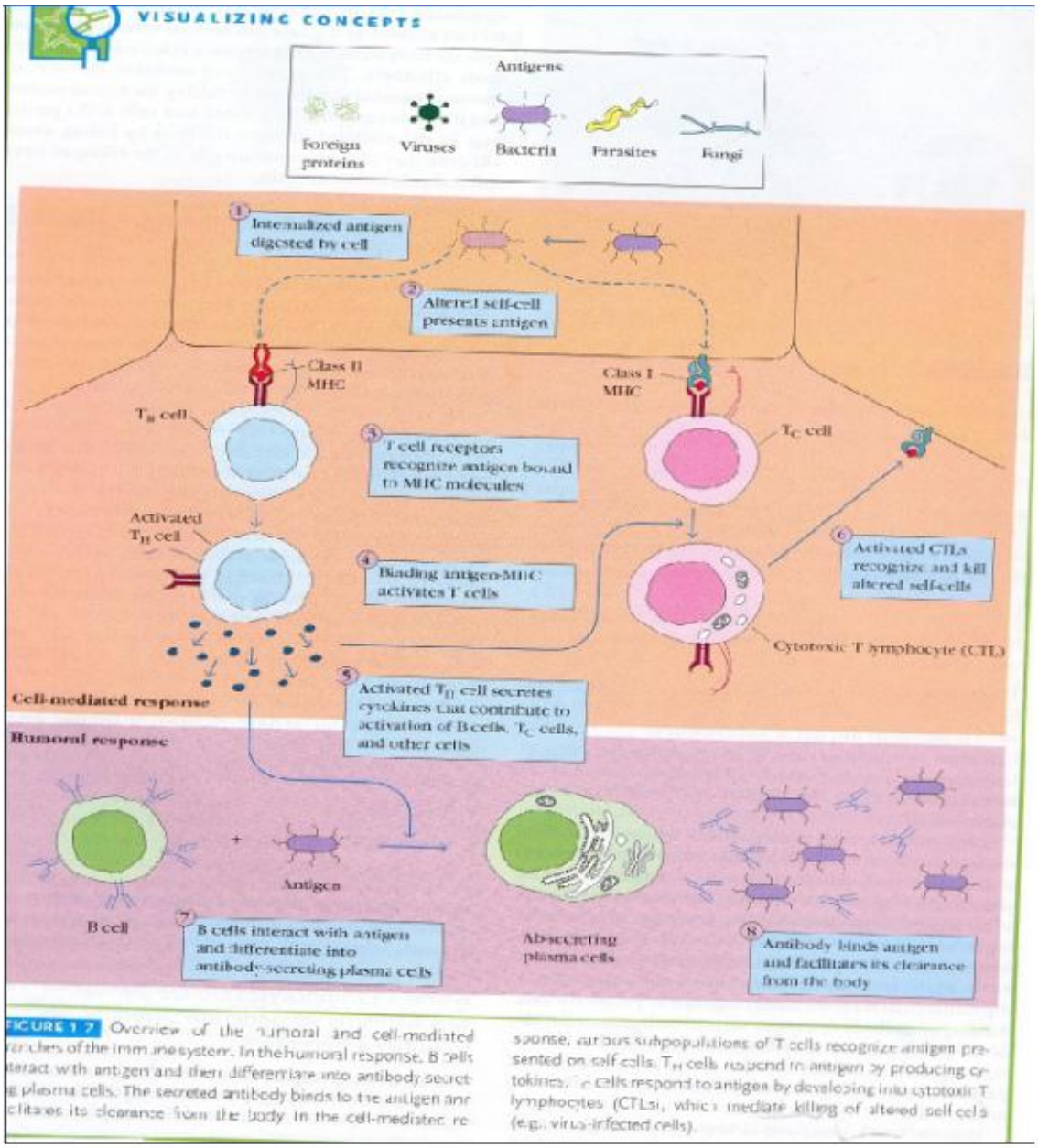


FIGURE 1.7 Overview of the humoral and cell-mediated branches of the immune system. In the humoral response, B cells interact with antigen and then differentiate into antibody secreting plasma cells. The secreted antibody binds to the antigen and facilitates its clearance from the body. In the cell-mediated response, various subpopulations of T cells recognize antigen presented on self cells. T_H cells respond to antigen by producing cytokines. T_C cells respond to antigen by developing into cytotoxic T lymphocytes (CTLs), which mediate killing of altered self cells (e.g., virus-infected cells).

A1.2 Presentación de Antígenos

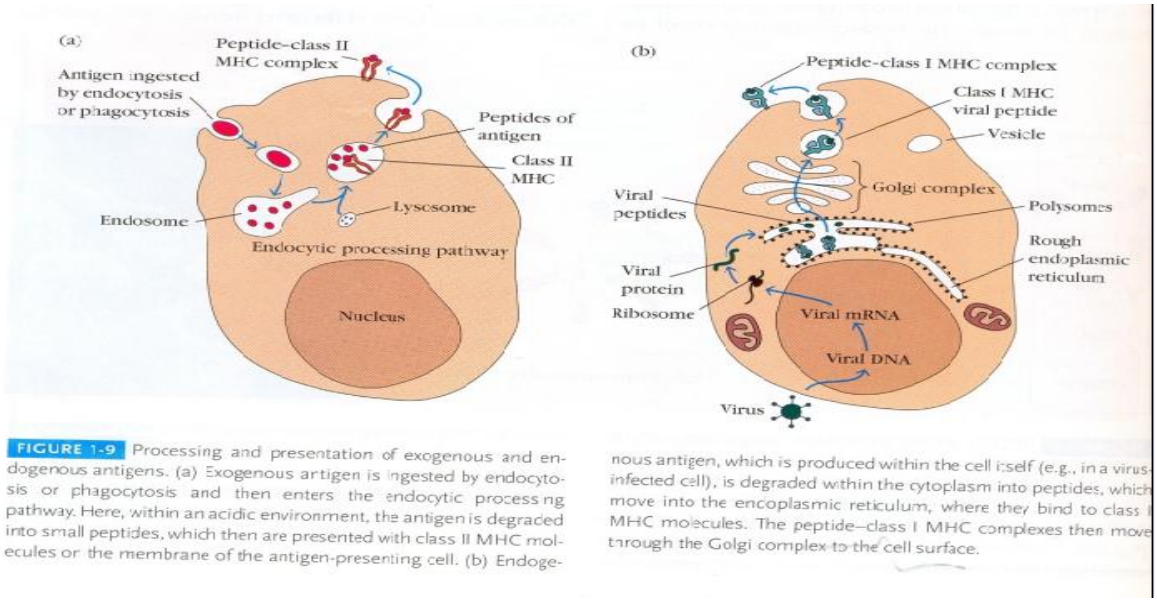


FIGURE 1-9 Processing and presentation of exogenous and endogenous antigens. (a) Exogenous antigen is ingested by endocytosis or phagocytosis and then enters the endocytic processing pathway. Here, within an acidic environment, the antigen is degraded into small peptides, which then are presented with class II MHC molecules on the membrane of the antigen-presenting cell. (b) Endogenous antigen, which is produced within the cell itself (e.g., in a virus-infected cell), is degraded within the cytoplasm into peptides, which move into the encoplasmic reticulum, where they bind to class I MHC molecules. The peptide–class I MHC complexes then move through the Golgi complex to the cell surface.

A1.3 Reconocimiento de las células mediante receptores MHC.

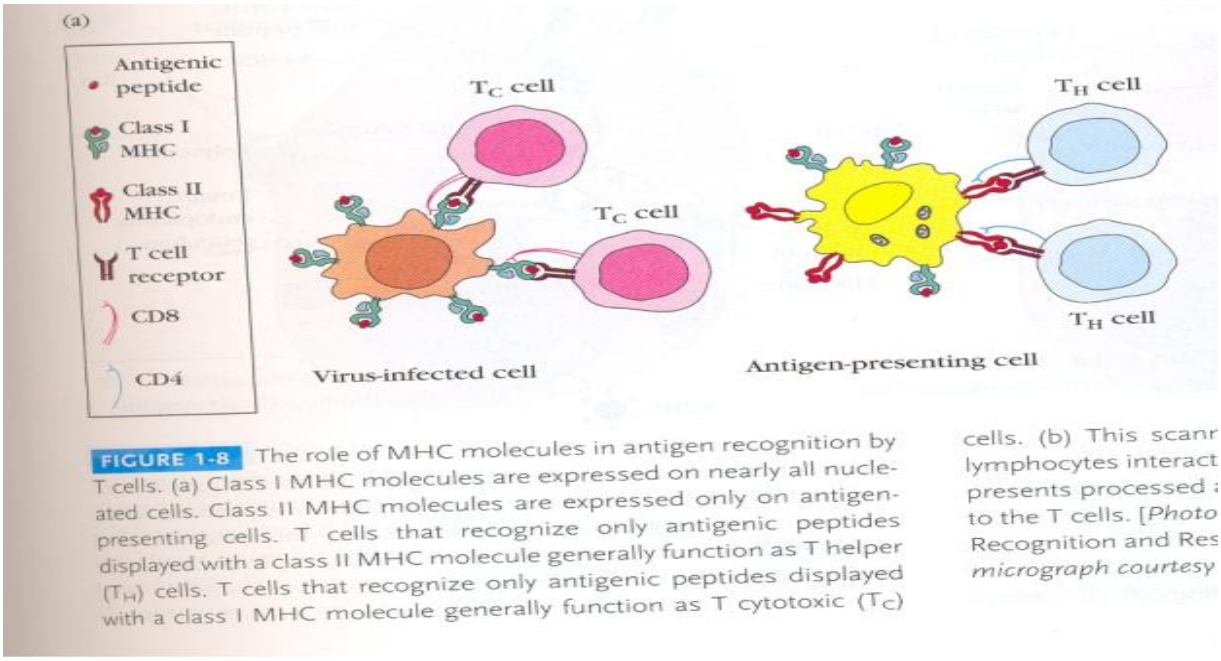
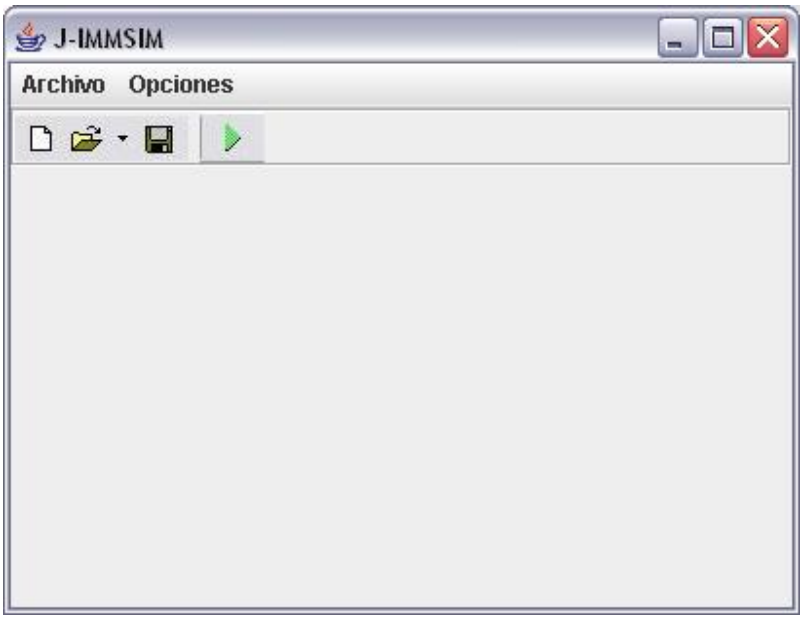


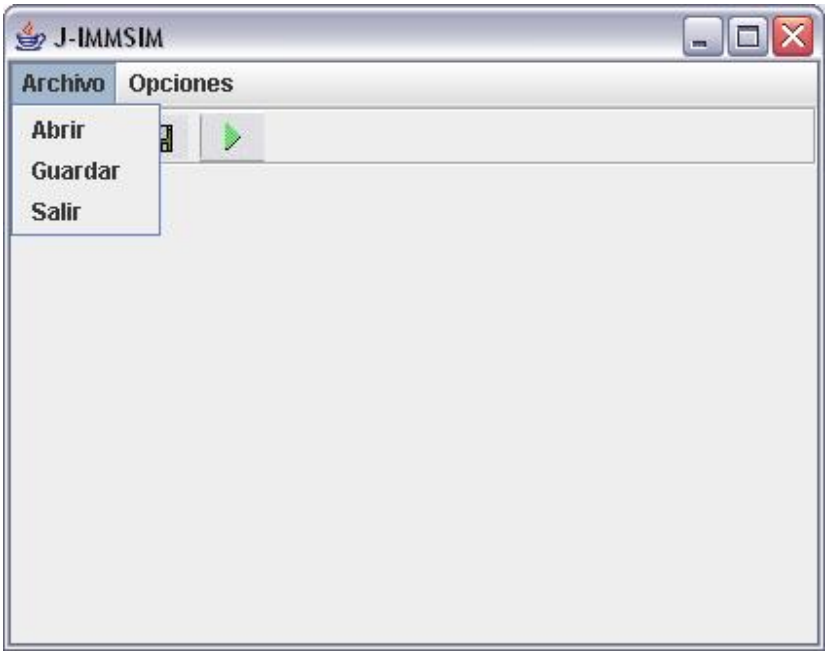
FIGURE 1-8 The role of MHC molecules in antigen recognition by T cells. (a) Class I MHC molecules are expressed on nearly all nucleated cells. Class II MHC molecules are expressed only on antigen-presenting cells. T cells that recognize only antigenic peptides displayed with a class II MHC molecule generally function as T helper (T_H) cells. T cells that recognize only antigenic peptides displayed with a class I MHC molecule generally function as T cytotoxic (T_C) cells. (b) This scanr lymphocytes interact presents processed to the T cells. [*Photo* Recognition and Res *micrograph courtesy*

Anexo 2 Prototipos no funcionales

A 2.1



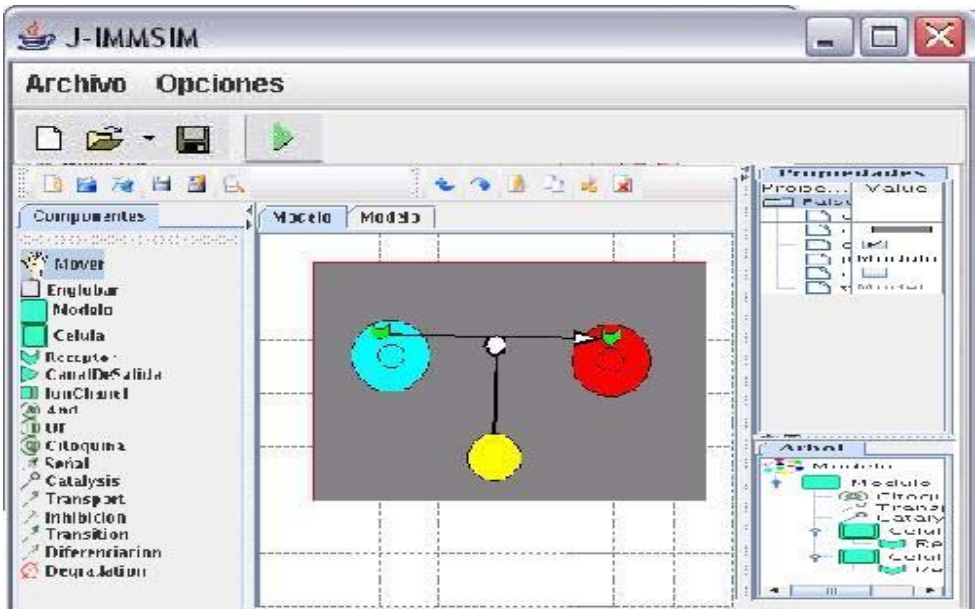
A2.2


J-IMMSIM
Archivo
Opciones
Abrir
Guardar
Salir

A2.3


Parámetros Generales
Generales
Microlitos
Long BitString
MinMatch
Timo
Eficiencia
PP
Propio
Nodo Linfático
Dim X
Dim Y
MaxFactLocal
Aceptar
Cancelar

A2.4


J-IMMSIM
Archivo Opciones
Componentes
Modelo
Celula
Catalysis
Transport
Inhibition
Transition
Arbol

Glosario de Términos

Anticuerpos: Tipo de proteína elaborada por las células plasmáticas (tipo de glóbulos blancos) en respuesta a un antígeno (sustancia extraña). Cada anticuerpo puede unirse a un solo antígeno específico. El propóstio de esta unión es ayudar a destruir el antígeno. Los anticuerpos trabajan de varias formas, según la naturaleza del antígeno. Algunos anticuerpos destruyen los antígenos en forma directa. Otros le facilitan la tarea a los glóbulos blancos para que estos destruyan el antígeno.

Antígeno: Es una sustancia que induce la formación de anticuerpos, debido a que el sistema inmune la reconoce como una amenaza. Esta sustancia puede ser extraña (no nativa) proveniente del ambiente (como químicos) o formada dentro del cuerpo (como toxinas virales o bacterianas).

Bit-string: Es una cadena de bits utilizada para identificar receptores, antígenos y otras moléculas.

Célula: Es la unidad más esencial que tiene todo ser vivo. Es además la estructura funcional fundamental de la materia viva según niveles de organización biológica, capaz de vivir independientemente como entidades unicelular, o bien, formar parte de una organización mayor, como un organismo pluricelular.

Citocinas: Son proteínas que regulan la función de las células que las producen u otro tipos celulares. Son los agentes responsables de la comunicación intercelular, inducen la activación de receptores específicos de membrana, funciones de proliferación y diferenciación celular, quimiotaxis, crecimiento y modulación de la secreción de inmunoglobulinas.

Fagositosis: Proceso por el cual un fagocito (tipo de glóbulo blanco) rodea y destruye las sustancias destruye las sustancias extrañas (como bacterias) y extrae las células muertas.

Inmunoglobinas: Las Inmunoglobulinas son proteínas anticuerpo altamente específicas que son producidas en respuesta a antígenos específicos. Los anticuerpos o inmunoglobulinas son producidos por los linfocitos B en su forma unida a la membrana. Este anticuerpo unido a la membrana constituye el receptor de antígenos de la célula B

Linfoncitos: Los linfocitos son células de alta jerarquía en el sistema inmune, principalmente encargadas de la inmunidad específica o adquirida.

Lisis: En biología, lisis se refiere al deterioro de una célula causado por una lesión en su membrana plasmática (exterior). La lisis puede ser causada por medios químicos o físicos (por ejemplo, detergentes fuertes u ondas sonoras de alta energía) o por una infección.

Macrófagos: Son fagocitos que viajan a través del cuerpo en busca de patógenos invasores. La función principal de los macrófagos es la de fagocitar todos los cuerpos extraños que se introducen en el organismo como las bacterias y sustancias de deshecho de los tejidos

Molécula: Partícula más pequeña de una sustancia que tiene todas las propiedades físicas y químicas de esa sustancia. Las moléculas están compuestas por uno o más átomos. Si contienen más de un átomo, los átomos pueden ser iguales (una molécula de oxígeno tiene dos átomos de oxígeno) o distintos (una molécula de agua tiene dos átomos de hidrógeno y un átomo de oxígeno). Las moléculas biológicas, como las proteínas y el ADN, pueden estar compuestas por muchos miles de átomos.

Receptor: Molécula en el interior o la superficie de una célula que se une con una sustancia específica que causa un efecto fisiológico específico en la célula.