Universidad de las Ciencias Informáticas

Facultad 5

Título: Núcleo Matemático para un Simulador Eléctrico.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS

Autores:

Jabier Guerra Fonseca

Addsel Perdomo Vázquez

Tutora

Ing. Iliana Pérez Pupo

Co-Tutora

Ing. Adisley Reyes Crespo

Junio 2010

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 _____

_____ _____

Addsel Perdomo Vázquez

Autor

Jabier Guerra Fonseca

Autor

_____

Ing. Iliana Pérez Pupo

Tutor

_____

Ing. Adisley Reyes Crespo

Co-Tutora

Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad.

Jabier

A nuestra Revolución por ser una obra tan generosa y darme la posibilidad de formarme como profesional y como persona en la mejor de las universidades.

A mi mamita (María Luisa Fonseca), que ha sido ejemplo, guía, fuerza, empeño, tesón, empuje, valor, confianza, que me ha enseñado el mundo tal y como es, a ti gracias por existir, por respirar, sin ti la vida no tiene sentido.

A mi papá (José Guerra), por ser el mejor del mundo, el más preocupado, por ser mi ejemplo, por tu dedicación, tu voluntad, tu carisma, a ti que me has enseñado a enfrentar los más grandes retos sin mirar el tamaño, sino sus debilidades, para atacarlos desde ese ángulo. Gracias por todo.

A mi hermano preferido (Emilio, único que tengo), que espero grandes logros de él, por cómo es conmigo, por su cariño, por su amor, por obligarme a ser mejor cada día, pues me tiene como ejemplo y quiero ser el mejor para él.

A mi abuela, Sara, por luchar tantos años seguidos conmigo, por soportar mis pesadeces, por ser mi segunda madre, por su amor, por su ternura, a ti gracias vieja linda, espero que no te vayas nunca.

A mis abuelos Isabel y Pepe, que han sido ejemplo para mí y para todos. Si me dieran a escoger abuelos, sin pensarlo los escogería a ustedes. Los quiero mucho.

A mis tíos Isael, Julián, Arquímedes, por el cariño y el amor que me han dado, me siento orgullosos de ustedes.

A mi tío Heriberto, por estar siempre ahí, por su confianza, su ejemplo, su desinterés, su dedicación, su firmeza. A ti Gracias.

A mis tías Zaida, Damaris y Susana (chuchi), las quiero mucho, mucho, mucho,

por estar en todo momento para mí, por su dulzura, su amor, su cariño y ejemplo.

A mi prima Evelyn, que más que mi prima es como una hermana, por todo lo que ha hecho por mí, por su amor, su sonrisa, su calidad humana, nunca cambies.

A mis primas Diana, Leyanet y Rachel, las quiero mucho, mucho y espero mucho de ustedes.

A mis primos Camilo, Juliancito, Albertico, Raulito, los quiero mucho.

A Aracelis, Ramiro, Mireidis, Irene, Ramirito, por ser mi familia durante mi niñez, de ustedes aprendí mucho y siempre los voy a llevar conmigo en donde quiera que esté.

A mis maestros de primaria: Julia, Abel, Radel, Amada, Ofelia, Zaida, Mellizo, por educarme y enseñarme mucho de lo que se hoy en día.

A mis vecinos, Mayra, Alberto, Paula, Rigo, Noris, Niurka, Novelia, Goche, Papi Eli, Mirna, Yanet, Yunel, Yurian, Madelaine, Matha, Jilberto, ustedes son los mejores vecinos que se puede tener, gracias por estar ahí para mí y para mi familia.

A Pablo José, por ser un hermano para mí durante estos años, sabes que esta amistad no se acaba aquí, seremos hermanos para toda la vida.

A Yamilet, por soportarme todos estos años, por ser mi apoyo, por celebrar mis victorias y levantarme en mis reveses. A ti gracias.

A Addsel, por ser mi compañero incansable de tesis, y además mi hermano, por su confianza y por su dedicación. A mis compañeros de aula de primer año Amides, Ihordan, Pavel, Jesús, Dayrel, Fabián, Ernesto Pérez, Hung, Yoel (-),

Yuniet, Alexeis, Abigail, Yanesita, Ernesto Castro, Leyner, Yadeilis, gracias por compartir estos años conmigo, por estar ahí cuando los necesité.

Addsel

Primeramente tengo que agradecer a esta Revolución por haberme permitido nacer tantas veces. Son muchas las personas involucradas, pero cuando ya no quedaba esperanza habia personas que se mantuvieron firmes. A mi papá que ha sido un amigo incondicional, fue como un bastón y una luz en los dias más difíciles, a mi mamá y a mi hermano por ayudarme durante todo este tiempo, los quiero mucho, a Calixto por ser más que un amigo de la familia, a Pura Avilés que sin sus manos yo no estaria aquí graduándome.

A lo mejor que me ha pasado en la vida, a mi amor Geidys, que no por estar lejos de mí llegando a México ha dejado de estar preocupada por mí, por la tesis.

A las personas del ITM por su preocupación y ayuda brindada, a Cacha.

A mi compañero de tesis, a mi amigo, a mi hermano, Jabier, por su entrega en este trabajo.

A mis compañeros que han transitado conmigo durante estos 5 años, a mis compañeros de cuarto, a la gente del piquete: al negrito de la virgen (Dayrel), a Tunas (Jesús), a Boll (Pablo), al pelú (Ihordan), al lagarto (Pavel).

A Adisley que su ayuda fue enorme, a Betty.

A la gente de Pinar, a las Jimaguas que son tan niñas todavía.

A la gente de Santa Fé, al Coco, a los vecinos de Bellavista. A mi primo Leonardo (el negro). A todos muchas gracias.

Jabier

Dedico este trabajo a mi mamá, a mi papá y a mi hermano.

A mi abuela Sara, mis tías y a mi tío Heriberto.

A mis abuelos paternos, mis tíos y mis primos.

A mis vecinos.

Addsel

Dedico este trabajo a mi familia, a mi papá, a mi mamá, y a mi hermano.

A mi novia.

A mi familia allá en oriente.

A mis tíos que ya no están y a los que están.

A Pérez y familia, a Calixto y familia.

A la gente de Pinar.

A Pura Avilés.

A la gente de Santa Felicia.

Resumen:

En la actualidad existen muchos simuladores que se utilizan para simular procesos, y así ahorrar recursos y tiempo a la hora de tomar decisiones; al mismo tiempo permite estudiar los procesos por complicados que estos sean. Un simulador no puede por sí sólo hacer todo ese trabajo sin un analizador matemático, éste le da el sustento tanto físico como matemático de todos los procesos que se van a estudiar en dichas simulaciones. Por lo que se hace necesario crear una herramienta para que concluya dicha simulación, en vista a ese objetivo se crea un núcleo matemático para el simulador eléctrico de los Sistemas de Adquisición de Datos y Control Supervisado (SCADA) el cual permite concluir los estudios de los fenómenos eléctricos que pueden ocurrir en la realidad, obteniendo resultados para cada simulación.

Palabras claves: Núcleo Matemático y Simulador eléctrico

Índice:

Introducción

En la actualidad, la sociedad cubana se encuentra enfrascada en su informatización, jugando un papel fundamental para el avance de la misma la Universidad de las Ciencias Informáticas, la cual se encuentra estructurada por centros productivos en los que se desarrollan software y se brindan servicios. Uno de los centros que se encuentran enfrascado en esta tarea de la informatización es el Centro de Desarrollo de Informática Industrial (CEDIN) el cual está organizado por líneas de desarrollo como son: la Línea de Seguridad, la Línea de Persistencia, la Línea de Software Empotrado y la Línea de Modelación y Procesamiento.

La Línea Modelación y Procesamiento, amplía las funcionalidades del (SCADA) permitiendo la ejecución y control de algoritmos especializados, creados por los usuarios, que puedan hacer uso de los servicios del SCADA. Esta aplicación le da al SCADA una planificación de acciones controladas en forma de calendarios que pueden ser configuradas por los usuarios para automatizar acciones, como la generación de reportes, aplicación de recetas, optimización de algoritmos y otros.

En esta línea se emplea la simulación como una de las vías para el desarrollo de software, ya que con ella se obtienen soluciones a problemas que se puedan dar en una situación determinada. Con la simulación se analizan los diferentes caminos de un estado inicial a otro siguiente, hasta llegar al final sin que afecten los resultados que se han logrado hasta ese momento, disminuyendo el riesgo de afectar los sistemas reales.

La **simulación** es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias dentro de los límites impuestos por un cierto criterio o un conjunto de ellos, para el funcionamiento del sistema.[1]

Para los efectos de simulación de sistemas, se considera un **modelo** a una descripción matemática de un sistema físico que puede obtenerse a partir de la evaluación de su conducta, basado en mediciones estimadas, observadas o realizadas directamente sobre el sistema que se pretende modelar. [2] **Modelar** significa construir una representación de algo, es una representación de estructuras. [3] El término **modelo** se refiere a la generalización conceptual que se abstrae de un grupo de experiencias con el

propósito de categorizar y sistematizar nuevas experiencias. [4]

En el CEDIN, dentro de la Línea de Modelación y Procesamiento se encuentra en desarrollo un Simulador Eléctrico, con el cual se simulará el funcionamiento de los circuitos eléctricos empleados por la línea. Esta línea está compuesta por los siguientes módulos: Repositorio de modelos, Módulo de simulación, Módulo de edición de modelos, y un Núcleo matemático.

La necesidad actual del proyecto es mejorar los procesos dentro del CEDIN con la simulación de los mismos y el análisis probabilístico de los eventos indeseables y de calidad de las variables a partir del resultado que arroje cada simulación para ayudar a la toma de decisión y control avanzado de los eventos de cada proceso en el CEDIN. Se necesita la obtención de una salida correcta dado un valor de entrada de cada componente que se utilizará en dicha simulación, mediante el comportamiento matemático asociado a ésta. Se mejorarían los procesos que se automatizan dentro del centro a partir de la simulación de los mismos, el análisis de los eventos que puedan surgir en la misma y se facilitaría la toma de decisiones.

Por eso surge la necesidad del desarrollo del proyecto Simulador Eléctrico de la línea “Modelación y Procesamiento” del CEDIN, el cual no posee una herramienta donde se puedan definir ecuaciones matemáticas que rijan la simulación del modelo eléctrico definido. Así mismo, tampoco se cuenta con una biblioteca que permita el chequeo de sintaxis usada durante la confección de una ecuación. Con la ausencia de esta herramienta, no se podrá completar la simulación de un proceso donde se necesite de análisis matemático para su operación, y por consiguiente, no se podrán estudiar los procesos reales para la toma de decisiones sin afectar el sistema real mediante la simulación del mismo.

A partir de lo descrito anteriormente se define el siguiente **problema científico**:

¿Cómo definir y calcular las ecuaciones que se generan a partir de las simulaciones de los circuitos eléctricos que se realizan en el Simulador Eléctrico de la línea de Modelación y Procesamiento del CEDIN?

Se define como **objeto de estudio**: Los Núcleos matemáticos para Simuladores.

El **campo de acción** se enmarca en: Los Núcleos Matemáticos para Simuladores Eléctricos.

Para dar respuesta al problema de la investigación se define como **objetivo general**: Desarrollar un núcleo matemático que permita la definición de las ecuaciones que rigen la simulación de los modelos definidos en el Simulador Eléctrico para la línea Modelación y Procesamiento del CEDIN.

Para dar cumplimiento al objetivo general se derivan las siguientes **tareas:**

- Análisis del estado del arte de los editores matemáticos existentes en el mundo para conocer en que se especializan y cuáles son las características de los mismos, así como también de simuladores eléctricos.
- Determinación de las diferentes bibliotecas (open source, código abierto) para el trabajo de lectura e interpretación de modelos matemáticos.
- Definición de los requisitos funcionales y no funcionales del sistema.
- Desarrollo del diagrama de caso de uso del sistema.
- Descripción de los casos de uso del sistema.
- Realización de los CU-Diseño.
    - Diagramas de Clases del Diseño.
    - Diagramas de Secuencia del Diseño.
- Elaboración de la Vista Lógica.
- Elaboración del diagrama de despliegue.
- Realización de los diagramas de componentes del sistema.
- Implementación de la aplicación que cumpla con las especificaciones necesarias para el cumplimiento de los requerimientos.
- Realización de las Pruebas de Unidad.

- Documentación de los resultados de las pruebas realizadas.

Resultados esperados:

- Definición de una biblioteca para el análisis semántico y sintáctico de ecuaciones matemáticas para el núcleo matemático de la línea Simulación.

- Obtención de la herramienta núcleo matemático.

El presente trabajo de diploma consta de cuatro capítulos, introducción, conclusiones y bibliografía.

En el **Capítulo 1 Fundamentación Teórica**: Se describe el estado del arte actual. Se hace referencia a los diferentes editores matemáticos y simuladores eléctricos que se utilizan en el mundo así como también las herramientas, metodologías y bibliotecas que se utilizan.

En el **Capítulo 2 Descripción del Sistema:** Se definen los requerimientos funcionales y no funcionales del núcleo matemático. Se describen los casos de uso para la comprensión de las funcionalidades, así como el modelo de dominio.

En el **Capítulo 3 Diseño de la solución:** En este capítulo se describe la fase de diseño, representando los diagramas de clases del diseño, los diagramas de secuencia del diseño y el diagrama de despliegue.

En el **Capítulo 4 Implementación y prueba del Sistema:** Aborda la programación realizada partiendo de los requerimientos identificados y el diagrama de clases del diseño elaborado. Además se muestran ejemplos de las validaciones del sistema realizadas.

Capítulo 1: Fundamentación Teórica.

En este capítulo se describe el estado del arte actual. Se hace referencia a los diferentes editores matemáticos y simuladores eléctricos que se utilizan en el mundo así como también las herramientas, metodologías y bibliotecas que se utilizan.

1.1 Simuladores para circuitos eléctricos.

Un **simulador eléctrico** es un programa con el que se puede simular casi cualquier circuito común, permitiendo ver si se cometen errores y fallas que se producen y ver cómo solucionarlas.

Existen 3 formas de simular procesos utilizando herramientas digitales. La primera está constituida por entornos abiertos donde uno puede generar sus propias simulaciones. El segundo tipo, lo constituyen los Shockwave o applets, programas disponibles en Internet y que abordan simulaciones muy concretas de una manera rápida y versátil; y la tercera corresponde a aplicaciones más sofisticadas que tratan temas más complejos, en este caso los circuitos eléctricos, a los cuales está dirigida la investigación.

Algunos de los elementos simples que se usan en un simulador eléctrico son las resistencias, las fuentes, las bobinas, los pulsadores entre otros; hay otros más complejos como los variadores de velocidad de corriente alterna (CA) y corriente continua o directa (CC o CD).

Los sistemas de simulación tradicionales, utilizan principalmente señales simuladas a través de funciones o datos pre-capturados, mientras que los lenguajes de programación permiten el uso de señales reales analógicas y digitales, junto a las simuladas. El manejo de señales reales con los modelos de simulación permite que el modelo sea optimizado y ajustado en tiempo real, de acuerdo a los patrones de comportamiento de las señales de entrada y salida del sistema. Es decir, el modelo puedo ser ajustado en base a la respuesta del proceso o sistema real a las señales de estímulo generadas por el modelo mismo.

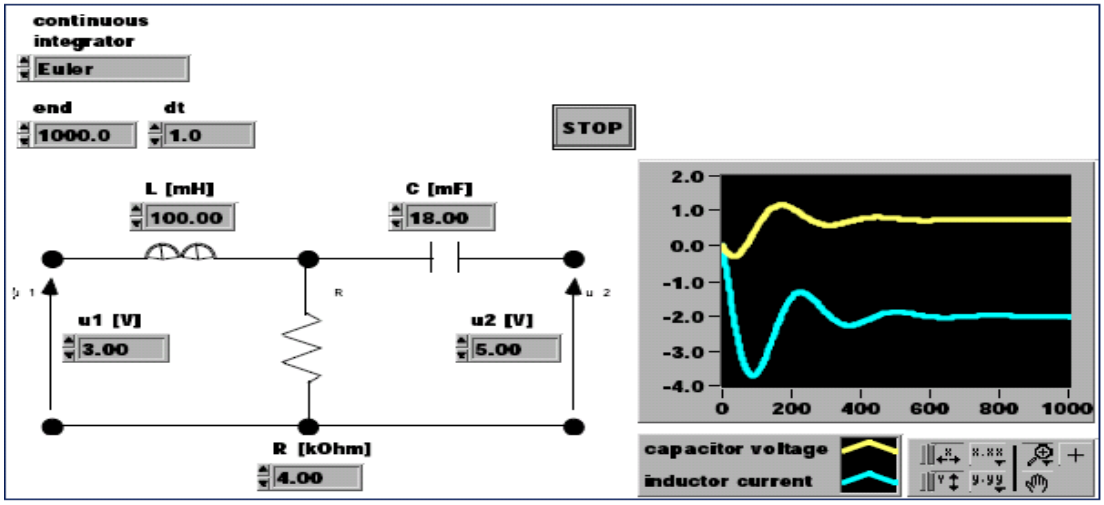

Figura 1.1 Ejemplo de un Proceso Simulado.

A continuación se caracterizan algunos de los simuladores eléctricos soportados por Linux:

- **Simulador Oregano:**

  Oregano es un simulador de circuitos eléctricos y electrónicos que permitirá crear esquemas tanto con resistencias, condensadores, bobinas y elementos más avanzados como diodos, diodos zener, tiristores, diacs, triacs, potenciómetros, transistores (P-MOS, N-MOS), bombillas, leds, amplificadores operacionales, puesta a tierra, fusibles, pulsadores y otros componentes electrónicos.

  Una vez diseñado el circuito se marcan los nodos que se quieren medir y se establecen los parámetros de simulación. Luego de ejecutada se mostrará una gráfica con las tensiones en los nodos marcados en función del tiempo de simulación. [5]

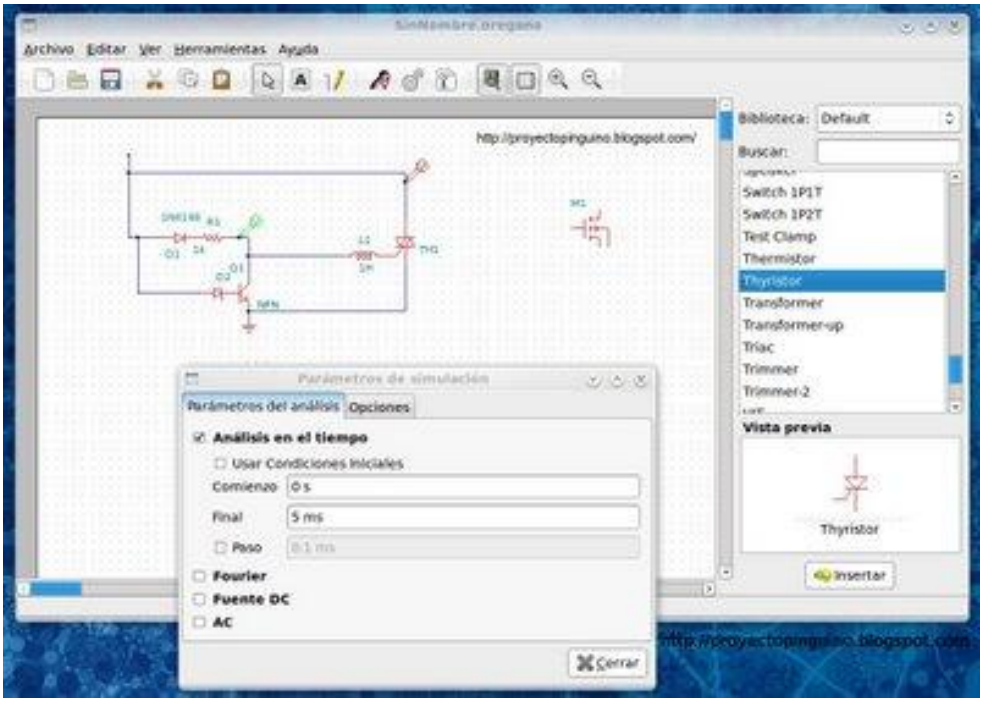

Figura 1.2 Simulador Oregano.

- **Simulador KSimus:**

Este simulador está enfocado a procesos técnicos y circuitos electrónicos que ofrece una buena diversidad de bloques para añadir al montaje: puertas lógicas, condicionales, funciones aritméticas, conversores, entradas/salidas booleanas y triestado. También se le puede añadir bloques extras que vengan en paquetes separados. [5]

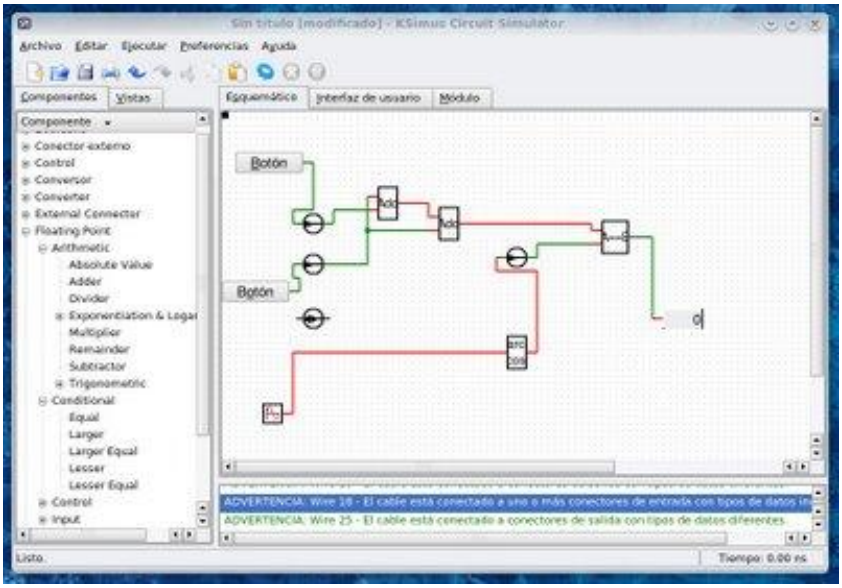

Figura 1.3 KSimus Circuit Simulator.

- **Simulador Qucs:**

  Qucs es un simulador eléctrico que se le puede ir añadiéndole componentes al dibujo e ir juntándolos por cables. Cuenta con resistencias, condensadores, bobinas, puestas a tierra, transformadores, bloques para corriente continua, amplificadores, atenuadores, bobinas, sondas de corriente y de tensión y conmutadores.

  La librería de componentes cuenta con muchos más bloques: varios tipos de Mosfets, amplificadores operacionales, Leds de varios colores, transistores, distintos diodos Zener y diodos convencionales. En cuanto a la simulación, se puede ver la gráfica de las tensiones respecto al tiempo, usar diagramas de tiempos y tablas de verdad. [5]

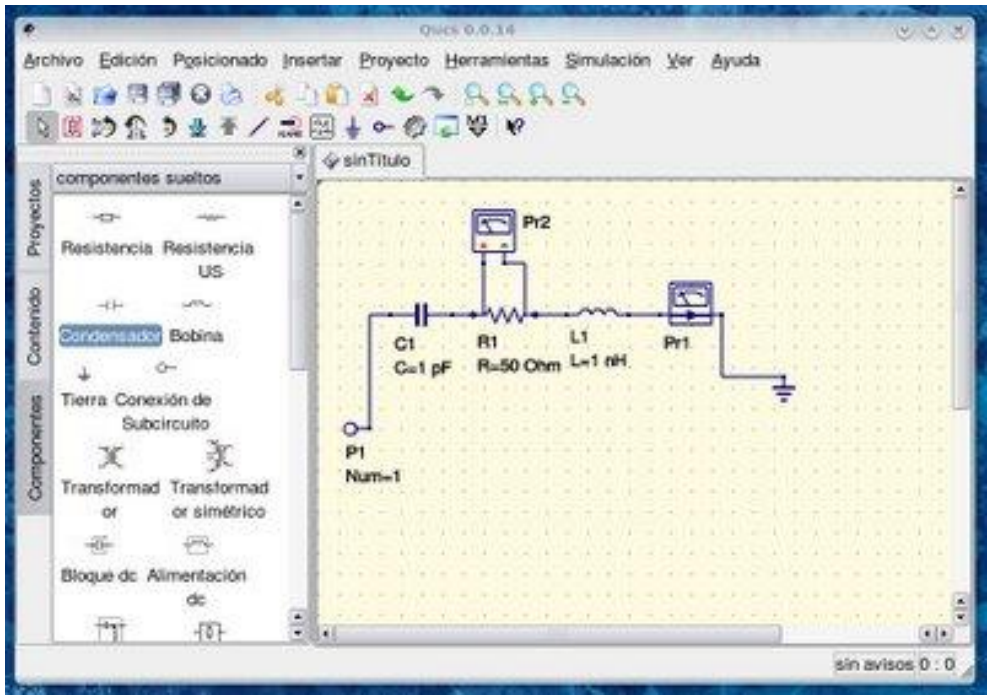

Figura 1.4 Simulador Qucs.

- **Simulador RCSim:**

  RCSim es un simulador de circuitos muy elemental. Dispone de una librería muy limitada en la que hay componentes resistivos, fuentes de tensión y de corriente y cables para hacer las conexiones. Permite el diseño del circuito directamente en pantalla mediante selección del componente y ubicación en la hoja de dibujo. Cuenta con instrumentos de medición que muestran los valores de voltaje y corriente mientras se ejecuta la simulación. Este programa es de libre distribución. [5]

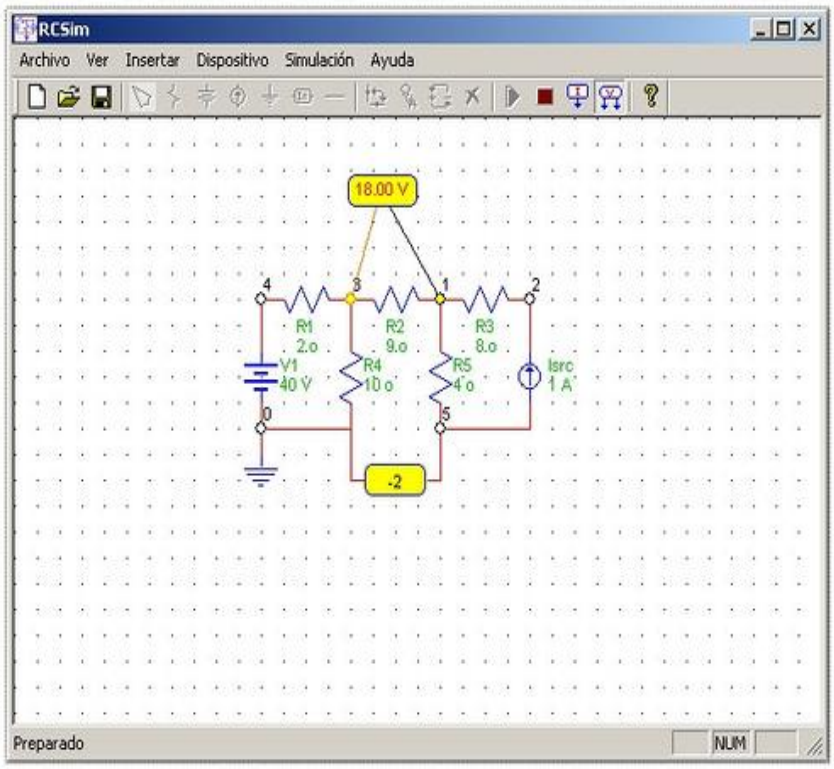

Figura 1.5 Simulador RCSim.

1.2 Circuitos Eléctricos.

Un circuito eléctrico no es más que un grupo de elementos o componentes eléctricos, tales como resistencias, inductancias, condensadores, fuentes, y/o dispositivos electrónicos semiconductores, conectados eléctricamente entre sí con el propósito de generar, transportar o modificar señales eléctricas.

1.2.1 Corriente Directa (CD).

La CD o CC es aquella cuyas cargas eléctricas o electrones fluyen siempre en el mismo sentido en un circuito eléctrico cerrado, moviéndose del polo negativo hacia el polo positivo de una fuerza electromotriz (FEM), tal como ocurre en las baterías, las dinamos o en cualquier otra fuente generadora de ese tipo de corriente eléctrica.

Es importante conocer que ni las baterías, ni los generadores, ni ningún otro dispositivo similar crea cargas eléctricas pues, de hecho, todos los elementos conocidos en la naturaleza las contienen, pero para establecer el flujo en forma de corriente eléctrica es necesario ponerlas en movimiento.

Las cargas eléctricas se pueden comparar con el líquido contenido en la tubería de una instalación hidráulica. Si la función de una bomba hidráulica es poner en movimiento el líquido contenido en una tubería, la función de la tensión o voltaje que proporciona la FEM, es precisamente, bombear o poner en movimiento las cargas contenidas en el cable conductor del circuito eléctrico. Los elementos o materiales que mejor permiten el flujo de cargas eléctricas son los metales y reciben el nombre de “conductores”. [6]

1.2.2 Corriente Alterna (CA).

Además de la CD existe la CA, que se diferencia de la primera por el cambio constante de polaridad que efectúa por cada ciclo de tiempo. La característica principal de una corriente alterna es que durante un instante de tiempo un polo es negativo y el otro positivo, mientras que en el instante siguiente las polaridades se invierten tantas veces como ciclos por segundo o hertz posea esa corriente. No obstante, aunque se produzca un constante cambio de polaridad, la corriente siempre fluirá del polo negativo al positivo, tal como ocurre en las fuentes de FEM que suministran corriente directa.

Un ejemplo es que si ponemos una pila a girar a una determinada velocidad se producirá un cambio constante de polaridad en los bornes donde hacen contacto los dos polos de dicha pila. Esta acción hará que se genere una corriente alterna tipo pulsante, cuya frecuencia dependerá de la cantidad de veces que se haga girar la manivela a la que está sujeta la pila para completar una o varias vueltas completas durante un segundo.[6]

1.2.3 Leyes físicas de circuitos eléctricos.

Muchos científicos han emitido teoremas o leyes que se usan frecuentemente en el análisis y resolución de los circuitos eléctricos, tales como Kirchhoff, Thevenin, Norton y Ohm. Estos teoremas o leyes se aplican a varias ramas de la electricidad, es decir, son muchos los científicos que han demostrado sus

estudios en las diversas ramas de la misma, se explicarán varios de estos a continuación para entender mejor cómo funcionan en un circuito. Hay infinidad de fórmulas dentro de un circuito, ya que cada componente ya sea un transformador o un condensador, le da un resultado distinto a cada valor si esta insertado en el circuito, que se busca en el mismo.

Los circuitos se rigen por varias leyes, ya sea de CC o de CA, las fórmulas no son iguales, por lo que el valor de las variables se halla de maneras distintas, sin estas no se sabría el valor exacto de cada variable en cada lugar del circuito y en cada momento que se quiere analizar. Algunas de estas leyes son la **Ley de Ohm** y las **Leyes de Kirchhoff,** estas últimas están dentro de los teoremas de redes. **El teorema de Superposición** que no es más que la respuesta de una red lineal que contiene varias fuentes independientes se halla, considerando cada generador por separado y sumando después las respuestas parciales. Cuando se calcula la respuesta debida a una fuente, cada uno de los generadores independientes se sustituye por su impedancia interna. [7]

Los teoremas de Thevenin y Norton plantean que una fuente de tensión V en serie con una impedancia Z es equivalente a una fuente de corriente I en paralelo con Z, siempre que I=V/Z. Estas leyes que dieron a conocer son solo aplicables a circuitos lineales, pero son válidas aunque se hallen fuentes independientes. [7]

Existen también muchas fórmulas físicas que relacionan a los elementos del circuito; todo ello para calcular matemáticamente y físicamente el comportamiento de dichos elementos que se quieran calcular. A continuación se hace mención a las Leyes de Ohm y de Kirchhoff, pero hay muchas más, como por ejemplo para calcular Capacidad eléctrica, para calcular el Efecto joule, la Resistencia de los conductores, la Potencia en CC, la Frecuencia eléctrica, y tantas otras en dependencia de la complejidad del circuito que se vaya a simular.

Algunas de las fórmulas para circuitos de CC son:

Ley de Ohm: plantea que la corriente eléctrica es directamente proporcional al voltaje e inversamente proporcional a la resistencia eléctrica.

- ✓ I=V/R.

Donde I es la corriente eléctrica, V la diferencia de potencial y R la resistencia eléctrica.

- ✓ La corriente o intensidad eléctrica es el flujo de carga por unidad de tiempo que recorre un material. Se debe a un movimiento de los electrones en el interior del material.
- ✓ El voltaje, tensión o diferencia de potencial es la presión que ejerce una fuente de suministro de energía eléctrica o FEM sobre las cargas eléctricas o electrones en un circuito eléctrico cerrado, para que se establezca el flujo de una corriente eléctrica.

Se denomina **resistencia eléctrica**, simbolizada habitualmente como R, a la dificultad u oposición que presenta un cuerpo al paso de una corriente eléctrica para circular a través de él. En el Sistema Internacional de Unidades, su valor se expresa en ohmios, que se designa con la letra griega omega mayúscula, Ω.

Con un multímetro se puede calcular las 3 variables anteriores, para calcular resistencia ponemos la llave selectora en Ω; para medir corriente se pone A o si es muy baja mA y para el voltaje esta la corriente directa y la alterna y la unidad es voltios.

Esta expresión toma una forma más formal cuando se analizan las ecuaciones de Maxwell, sin embargo puede ser una buena aproximación para el análisis de circuitos de corriente continua. [8]

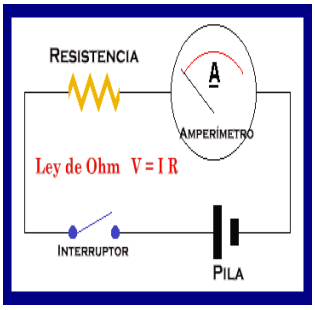

Figura 1.6 Ejemplo Ley de Ohm.

Cabe señalar que la ley de Ohm presenta algunas limitaciones como son:

1.- Se puede aplicar a los metales pero no al carbón o a los materiales utilizados en los transistores.

2.- Al utilizarse esta ley debe recordarse que la resistencia cambia con la temperatura, pues todos los materiales se calientan por el paso de corriente.

3.- Algunas aleaciones conducen mejor las cargas en una dirección que otra. [9]

Leyes de Kirchhoff.

Primera ley:

$$\sum_{k=1}^{n} I_k = I_1 + I_2 + I_3 \ldots + I_n = 0$$

Segunda ley:

$$\sum_{k=1}^{n} V_k = V_1 + V_2 + V_3 \ldots + V_n = 0$$

La primera ley de Kirchhoff describe con precisión la situación del circuito: La suma de las tensiones en un bucle de corriente cerrado es cero. Es decir la suma de corrientes que entran a un nudo es igual a la suma de las que salen. Las resistencias son sumideros de potencia, mientras que la batería es una fuente de potencia, por lo que la convención de signos descrita anteriormente hace que las caídas de potencial a través de las resistencias sean de signo opuesto a la tensión de la batería. La suma de todas las tensiones da cero. En el caso sencillo de una única fuente de tensión, una sencilla operación algebraica indica que la suma de las caídas de tensión individuales debe ser igual a la tensión aplicada. Y la segunda consiste en: la suma de todas las caídas de tensión es igual a la suma de todas las subidas de tensión. [10]

Algunas de fórmulas para circuitos de CC son:

- Reactancia inductiva.

La reactancia inductiva XL de una inductancia L a una frecuencia f y una velocidad angular w:

X_L = wL = 2wfL.

Si una corriente sinusoidal i de amplitud I y velocidad angular w pasa por una inductancia L, la tensión e a través de la inductancia:

e = L di/dt = L d(I sen wt)/dt = wLI cos wt = XLI cos wt

La corriente que atraviesa una inductancia está retrasada 90° con respecto a la tensión.

- Potencia en una impedancia serie.

Si una tensión V (tomada como referencia) se aplica a una impedancia Z formada por una resistencia R en serie con una reactancia X, la corriente I y j es el ángulo formado entre el cateto contiguo y la hipotenusa:

I = VY = V(R / |Z|2 - jX / |Z|2) = VR / |Z|2 - jVX / |Z|2 = IP - jIQ

La corriente activa IP y la corriente reactiva IQ valen:

IP = VR / |Z|2 = |I|cos Θ
IQ = VX / |Z|2 = |I|sen Θ

El valor de la potencia aparente S, la potencia activa P, y la potencia reactiva Q es:

S = V|I| = V2 / |Z| = |I|2|Z|

P = VIP = IP2|Z|2 / R = V2R / |Z|2 = |I|2R = V|I|cos Θ

Q = VIQ = IQ2|Z|2 / X = V2X / |Z|2 = |I|2X = V|I|sen Θ

El factor de potencia cos Θ resulta:

cos Θ = IP / |I| = P / S = R / |Z|.

1.3 Editores.

Desde hace mucho tiempo se utilizan los editores para mejorar el trabajo que se hace, crear o modificar

archivos digitales, existen editores de texto que lo que hace es leer un número binario y representarlo como una letra en la pantalla. También encontramos los editores gráficos que su función es desarrollar aplicaciones gráficas, además de mejorar y modificar imágenes. También están los editores matemáticos que aborda en esta investigación, con algunos ejemplos.

Hay editores que son los encargados de darle credibilidad a los video juegos ya sea de vuelos tanto de combate como comercial, de carreras de autos y muchos otros. Los editores son los encargados de darle el comportamiento físico y matemático, ya sea un aumento de velocidad, o un detenimiento brusco de la velocidad, todo ello relacionado directamente con los principios físicos y matemáticos.

1.4 Herramientas para la edición y construcción de fórmulas matemáticas.

Un **editor matemático** no es más que un programa que permite escribir expresiones matemáticas usando un lenguaje propio y reconociendo los errores para que el usuario los rectifique. En los últimos años los ordenadores han incrementado de forma drástica su capacidad para resolver grandes problemas procedentes de los más diversos campos de la ciencia debido, por una parte al portentoso avance que ha sufrido el hardware (ordenadores más potentes y rápidos) y por otra, al creciente desarrollo de software con un elevado nivel de sofisticación.[11]

Son múltiples los sistemas que se encargan de hacer cálculos complejos, entre los que se encuentran Derive. También algunos se especializan en calcular área y volumen en el espacio, con figuras geométricas. Existen también los editores matemáticos que están interrelacionados con los simuladores eléctricos, sin estos no funcionarían los simuladores dado que no se podría definir el comportamiento matemático de los componentes a simular. Algunos ejemplos son SimuLink y MatLab, que dentro de todos lo que son capaces de hacer está la simulación de circuitos eléctricos. [11]

A continuación se describen algunos de los editores matemáticos que existen:

1.4.1 Maple

Permite obtener soluciones analíticas exactas de los problemas matemáticos, se pueden calcular límites, derivadas e integrales de funciones, resolver sistemas de ecuaciones de forma exacta y encontrar

soluciones de ecuaciones diferenciales. [12]

1.4.2 Wiris

Es de acceso libre y gratuito. Opera con números enteros, racionales, radicales, decimales, reales y complejos. Funciones de divisibilidad (mcm, mcd, primos y factorización) con enteros. Funciones trascendente de variables reales (trigonométricas, exponenciales y logarítmicas). [12]

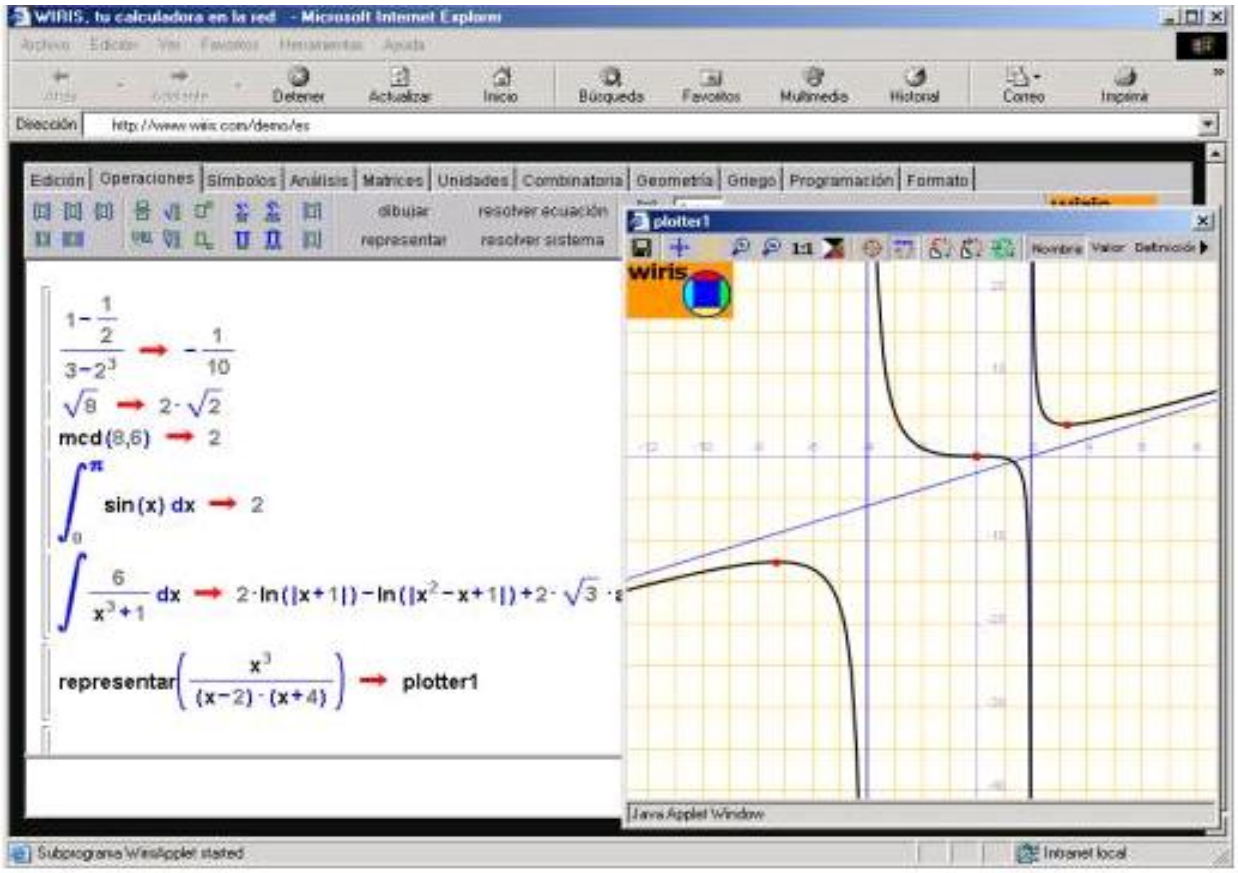

Figura 1.7 Interfaz del Wiris.

1.4.3 KFormula

Este programa es un editor de fórmulas matemáticas, con el cual de manera sencilla se pueda componer

una fórmula matemática y en forma de imagen se pueda presentar en una página Web o en un documento y no requiere sacrificar mucho tiempo para aprender a emplearlo. Tiene una sintaxis avanzada y exporta a Latex e importa de MathML.

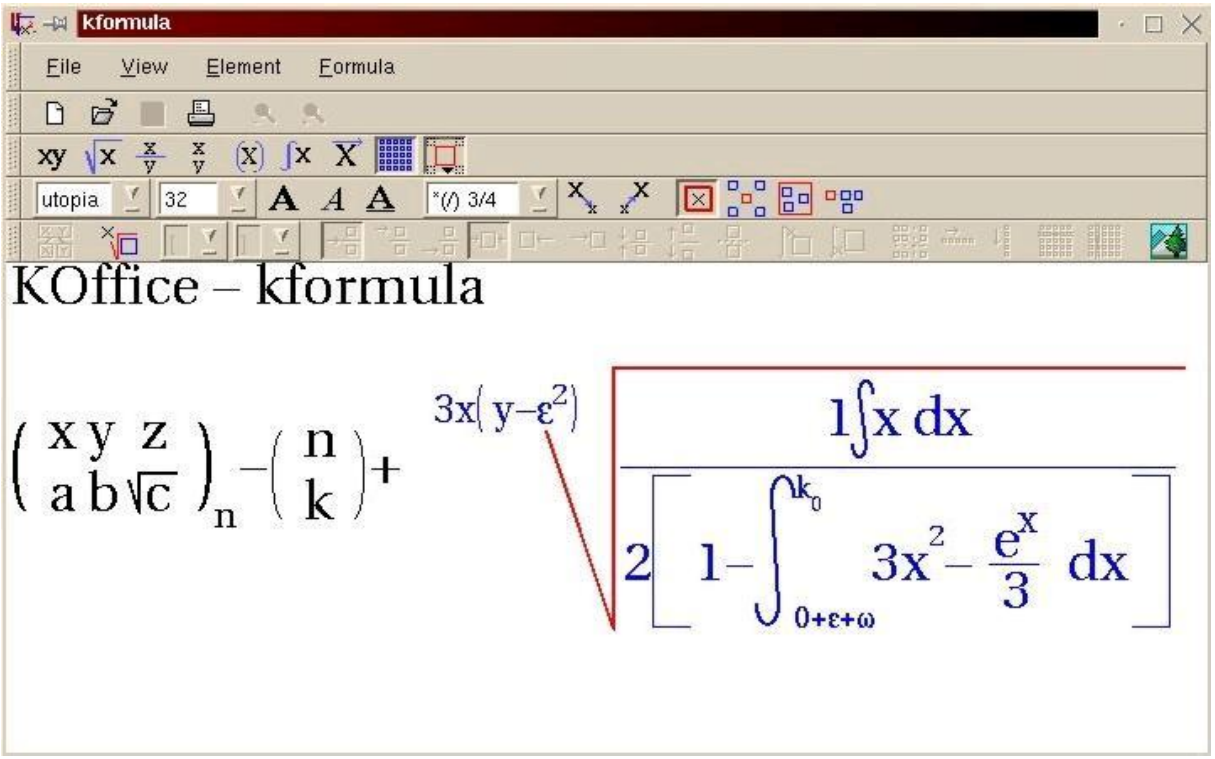

Figura 1.8 Interfaz del editor Kformula.

1.5 Bibliotecas para la interpretación y análisis de ecuaciones matemáticas.

Actualmente una biblioteca puede solucionar dificultades tanto del gráfico en los juegos, como en otras aplicaciones. En los simuladores y especialmente en los editores matemáticos permite que se llegue a un resultado veraz en las operaciones sin importar la complejidad de las mismas.

Muchas bibliotecas se han desarrollado incluso para brindar una equidad entre los distintos lenguajes de programación, por ejemplo GTK+ y GLib para desarrollar y trabajar con interfaces gráficas. El problema de estas bibliotecas es que no pueden trabajar en conjunto todas a la vez pues crean contradicciones. Muchos editores las usan para hacer los cálculos más rápido, como por ejemplo la math.h que contiene

las funciones matemáticas comunes, string.h para el trabajo con caracteres, y otras muchas. Pero las hay un poco más complejas, en esta investigación se estudiaron varias de estas bibliotecas.

1.5.1 SciMath

Es la más poderosa biblioteca matemática que se usa hoy en día. Esta biblioteca es adecuada para el desarrollo de aplicaciones en muchas áreas de las ciencias, como son las Matemáticas, la Física y la Ingeniería. SciMath contiene cientos de poderosas funciones matemáticas que permiten realizar cálculos con un alto grado de complejidad. Esta biblioteca se compila en formato de tipo .lib, similar a las bibliotecas usadas por las computadoras. Las aplicaciones a las cuales se le indexa esta biblioteca pueden hacer uso de cualquiera de los cientos de funciones que esta contiene. También contiene un manual de usuario donde se describe con detalles cada una de estas funciones. SciMath está disponible para compiladores de 16 y 32 bits como son SUN-OS (Sistemas Operativos SUN).

1.5.2 GSL (GNU Scientific Library)

Es una biblioteca numérica que se usa en programas de C y C++. Contiene funcionalidades que incluyen números complejos, raíces y polinomios, vectores y matrices, permutaciones y combinaciones, ordenamiento, álgebra lineal, números aleatorios, ecuaciones diferenciales, interpolación y otras funciones matemáticas importantes. Esta biblioteca fue liberada bajo la licencia GNU GPL y ha sido probada en distintos Sistemas Operativos tales como: SUN-OS, Linux, Windows, Mac OS y Unix.

Dado el análisis de las bibliotecas anteriormente expuestas, se ha seleccionado la GSL para el desarrollo del Núcleo Matemático, ya que se uso es libre y contiene las funciones matemáticas necesarias para su implementación.

1.6 Herramientas y Metodologías

Para la realización de una aplicación de software siempre se deben definir los tipos de tecnologías a utilizar así como las herramientas y metodologías que serán de mayor utilidad para su implementación y así lograr un producto final con la calidad requerida. A continuación se realiza un estudio de las tecnologías existentes en el proyecto y se explica en detalle cada una de la selección realizada con el fin

de llevar a cabo la implementación del sistema.

1.6.1 Metodología de Desarrollo: OpenUp

OpenUp es un FrameWork de procesos de desarrollo de software de código abierto. Es un proceso modelo y extensible, dirigido a la gestión y desarrollo de proyectos de software basados en desarrollo iterativo, ágil e incremental; y es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo.

Preserva la esencia del Proceso Unificado (RUP Rational Unified Process), que es la metodología más conocida por los estudiantes y profesores vinculados a la producción debido a que fue la estudiada en los cursos de Ingeniería de Software I y II, asignatura obligatoria de la carrera; y por demás, utilizada por la mayoría de los proyectos de la universidad.

Este proceso fue desarrollado por IBM y reconocido mundialmente como uno de los procesos de desarrollo de software de mayor calidad, basándose en los principios de adaptación, importancia a los involucrados e interesados en los resultados del proyecto; colaboración, valor a la iteración; y calidad continua.

OpenUp permite un abordaje ágil al proceso de desarrollo de software, con sólo proveer un conjunto simplificado de contenidos, fundamentalmente relacionados con orientación, productos de trabajo, roles, y tareas.

Características Generales:

- ✓ Preserva la esencia del Unified Process.
- ✓ Desarrollo iterativo e incremental.
- ✓ Desarrollo dirigido por Casos de Uso.
- ✓ Centrado en la Arquitectura. [13]

Procura un equilibrio entre las necesidades de los involucrados con los resultados del proyecto y los costos técnicos, con el fin de maximizar el valor de los involucrados y las guías del proceso de desarrollo.

Desarrolla un ciclo de vida interactivo que mitiga el riesgo a tiempo y ofrece demostrar resultados en curso al cliente del proyecto. [14]

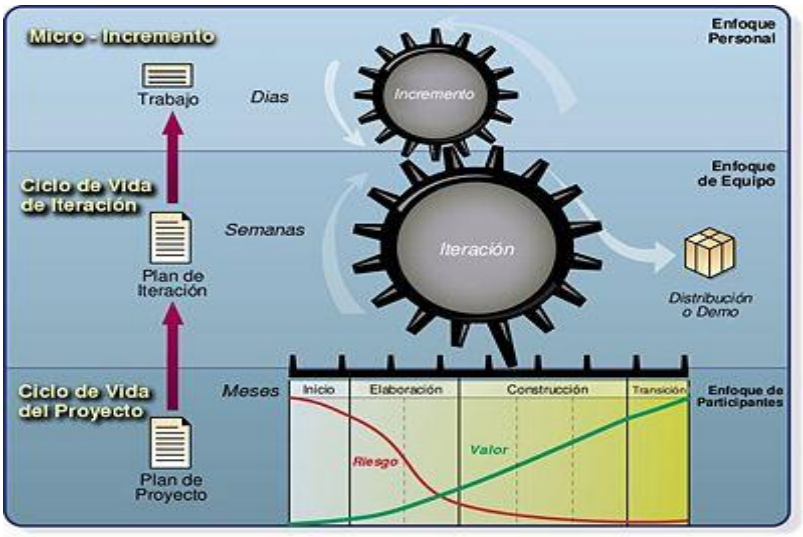

Figura 1.9 Ciclo de vida Open UP.

1.6.2 Lenguaje de modelado: UML

El lenguaje Unificado de Modelado (UML, por sus siglas en Ingles *Unified Modeling Language*) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. Permite la modelación de sistemas con tecnología orientada a objetos. No es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso.

UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos. El cual ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

1.6.3 Herramienta CASE: Visual Paradimg (Versión 6.4 Enterprise Edition):

Es una potente herramienta CASE (Computer-Aided Software Engineering) para visualizar y diseñar elementos de software, para ello utiliza el lenguaje UML, proporciona a los desarrolladores una plataforma que les permite diseñar un producto con calidad de una forma rápida. Facilita la interoperabilidad con otras herramientas CASE y se integra con las siguientes Aplicaciones Java: Eclipse/IBM WebSphere, Jbuilder, NetBeans IDE, Oracle Jdeveloper, BEA Weblogic. Está disponible en varias ediciones: Enterprise, Professional, Community, Standard, Modeler y Personal. [15].

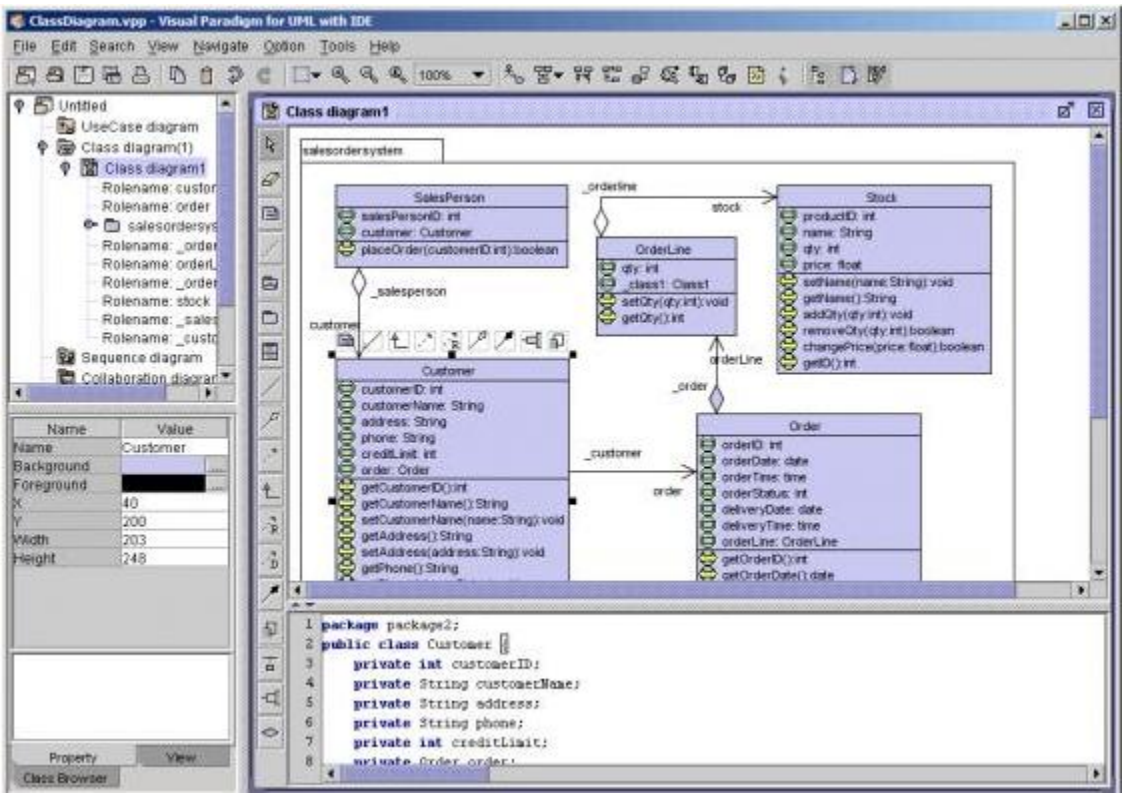

Figura 1.10 Interfaz de Visual Paradimg.

1.6.5 Lenguaje de Programación: C++

Hoy en día los entornos de desarrollo proporcionan un marco de trabajo para la mayoría de los lenguajes de programación existentes en el mercado (por ejemplo C, C++, C#, Java, Python y Visual Basic entre

otros). Además es posible que un mismo entorno de desarrollo tenga la posibilidad de utilizar varios lenguajes de programación, como es el caso de Eclipse. [15]

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina. [16]

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales. C++ permite trabajar tanto a alto como a bajo nivel. A partir de dichas razones y por estudios realizados en el proyecto sobre los diferentes lenguajes de programación posibles a utilizar en el proyecto se seleccionó dicho lenguaje.

1.6.6 IDE de Desarrollo: Eclipse (Ganymede)

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

Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones.

1.7 Conclusiones

Después de haber estudiado varios simuladores, estudiar también varios editores matemáticos, y de ver

las herramientas y tecnologías que van a utilizar, se arribaron a las siguientes conclusiones:

- Los simuladores más desarrollados son los hechos para Windows, y por tanto mucho más populares.
- Los editores de Windows de forma general tienen más aplicaciones en comparación con los de Linux, y por tanto están más desarrollados.
- Se le dio cumplimiento a las tareas previstas para este capítulo, se analizó el estado del arte de los editores matemáticos y simuladores, se conoció en qué se especializa cada uno de ellos, las operaciones que aceptan y a las que le pueden dar solución, así como determinación de la biblioteca de código abierto para el trabajo de lectura e interpretación de los modelos matemáticos.

Capítulo 2: Características del Sistema.

En este capítulo se definen los requerimientos funcionales y no funcionales del núcleo matemático. Se describen los casos de uso para la comprensión de las funcionalidades, así como un modelo de dominio.

2.1 Modelo de dominio.

Para poder entender el contexto en que se emplaza el sistema se describe el funcionamiento de la aplicación mediante una serie de conceptos, entidades y sus relaciones, agrupándolos en un modelo de dominio.

2.1.1 Modelo de dominio del Núcleo Matemático.

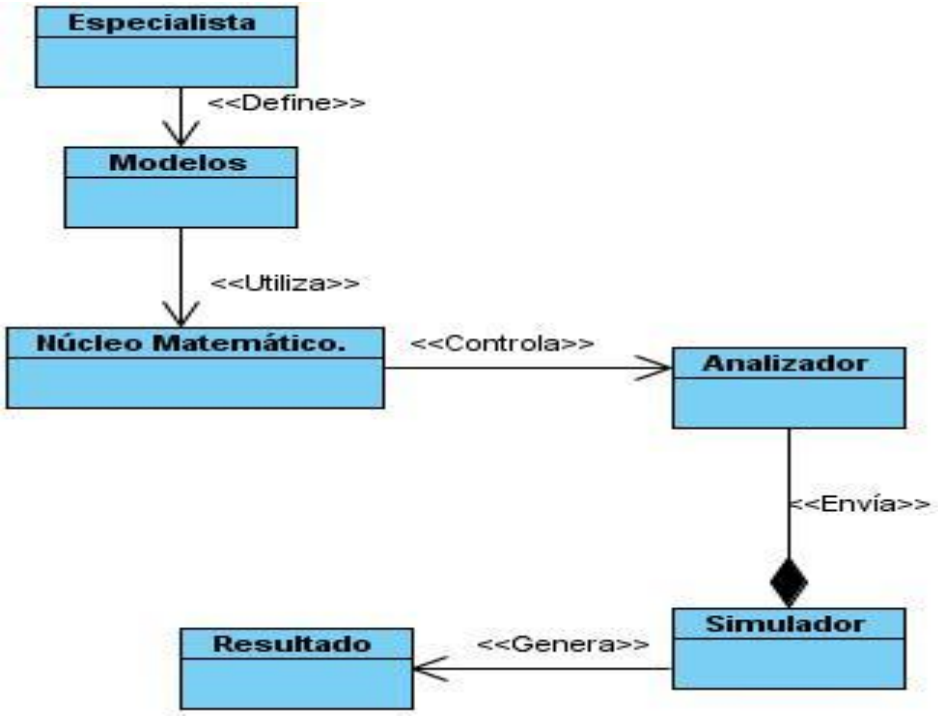

Figura 2.1 Diagrama del Modelo de dominio.

2.1.2 Glosario de términos del Dominio.

- ✓ **Núcleo Matemático:** Es parte del sistema que se encargará de hacer el enfoque matemático de lo que se quiere simular por parte del especialista.
- ✓ **Simulador:** Es parte del sistema que se encargará de hacer la correspondiente simulación que requiera el especialista u otro operario.
- ✓ **Especialista:** Es el encargado de definir las fórmulas que respaldarán el circuito que se van a simular, así como la inserción de éstas en un momento determinado.
- ✓ **Analizador:** Es la parte del sistema que se encarga de chequear todo lo que le llega al núcleo, es decir las fórmulas.
- ✓ **Modelos:** El núcleo utilizará modelos con fórmulas físicas predefinidos que van a ser usados por el especialista.

2.2 Requerimientos funcionales.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Ellos permiten determinar, de una manera clara, las responsabilidades que se propone el sistema. [17]

- ✓ R1 Analizar Lexicológicamente el modelo matemático.

  El sistema debe realizar análisis léxico de los modelos matemáticos, es decir debe verificar que los caracteres entrados por el usuario estén definidos en el sistema.

- ✓ R2 Analizar Sintácticamente el modelo matemático.

El sistema debe realizar análisis sintáctico de los modelos matemáticos que se definan, es decir verifica que la gramática de lo que está escrito, esté correcta.

- ✓ R3 Analizar Semánticamente el modelo matemático.

El sistema debe realizar análisis semántico, por ejemplo, verificar que no se le aplique un operador a un operando incompatible.

- ✓ R4 Resolver el modelo matemático.

El sistema debe de resolver los modelos matemáticos definidos generando como salida un algoritmo.

- ✓ R5 Gestionar el modelo matemático.

El sistema permitirá gestionar el modelo matemático:

- R5.1 Crear un modelo matemático.

  El sistema permitirá crear un nuevo modelo.

- ✓ R6 Importar modelo matemático.

El sistema permitirá importar modelos matemáticos.

- ✓ R7 Exportar modelo matemático.

El sistema permitirá exportar el modelo matemático.

- ✓ R8 Guardar modelo matemático.

El sistema permitirá guardar modelos matemáticos.

2.3 Requisitos no funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable, por ejemplo, pudiera desearse que el sistema responda dentro de un intervalo de tiempo especificado o que obtenga los resultados de los cálculos con un nivel de precisión dado. [17]

- ✓ Usabilidad.

El sistema podrá ser utilizado por quien tenga un nivel apropiado de conocimiento en el campo de la electricidad, pues tendrá que insertar fórmulas físicas relacionadas con la electricidad, además tiene que tener un conocimiento básico en programación.

- ✓ Fiabilidad.

El sistema debe garantizar un soporte matemático de forma tal que cualquier modelo definido en el Simulador Eléctrico funcione con la exactitud requerida.

- ✓ Reusabilidad

El sistema debe incluir una Interfaz con sistema de plugin de forma que garantice la reusabilidad del núcleo para ser usado por otros simuladores.

- ✓ Hardware.

La aplicación debe trabajar en máquinas de 1 GB de capacidad, 256 MB de RAM y procesador de más de 700 MHz de velocidad.

- ✓ Seguridad

Disponibilidad: Se garantizará que los usuarios autorizados tengan acceso al sistema para operarlo, sin retrasar al mismo.

- ✓ Software.

La aplicación deberá trabajar sobre un sistema operativo GNU/Linux.

- ✓ Requerimientos en el diseño de implementación.

- Eclipse como IDE de desarrollo.
- Visual Paradigm como herramienta CASE.
- C++ como lenguaje de programación.

- UML como lenguaje de modelado.
- Biblioteca GSL.

2.4 Descripción de actores.

Un actor puede ser un ser humano, o un software que interactúa con el Sistema que se quiere automatizar, y participa en uno o varios casos de uso.

Tabla 2.1 Descripción de los actores del sistema a automatizar.

Nombre del actor	Descripción
Especialista	Representa al usuario que se va a encargar de verificar la evaluación, y de añadirle fórmulas físicas para que la simulación se efectúe correctamente. Además se encargará de rectificar algún error que ocurra en el análisis léxico, sintáctico y semántico para resolver cualquier modelo matemático.

2.5 Diagrama de Casos de Uso del Sistema.

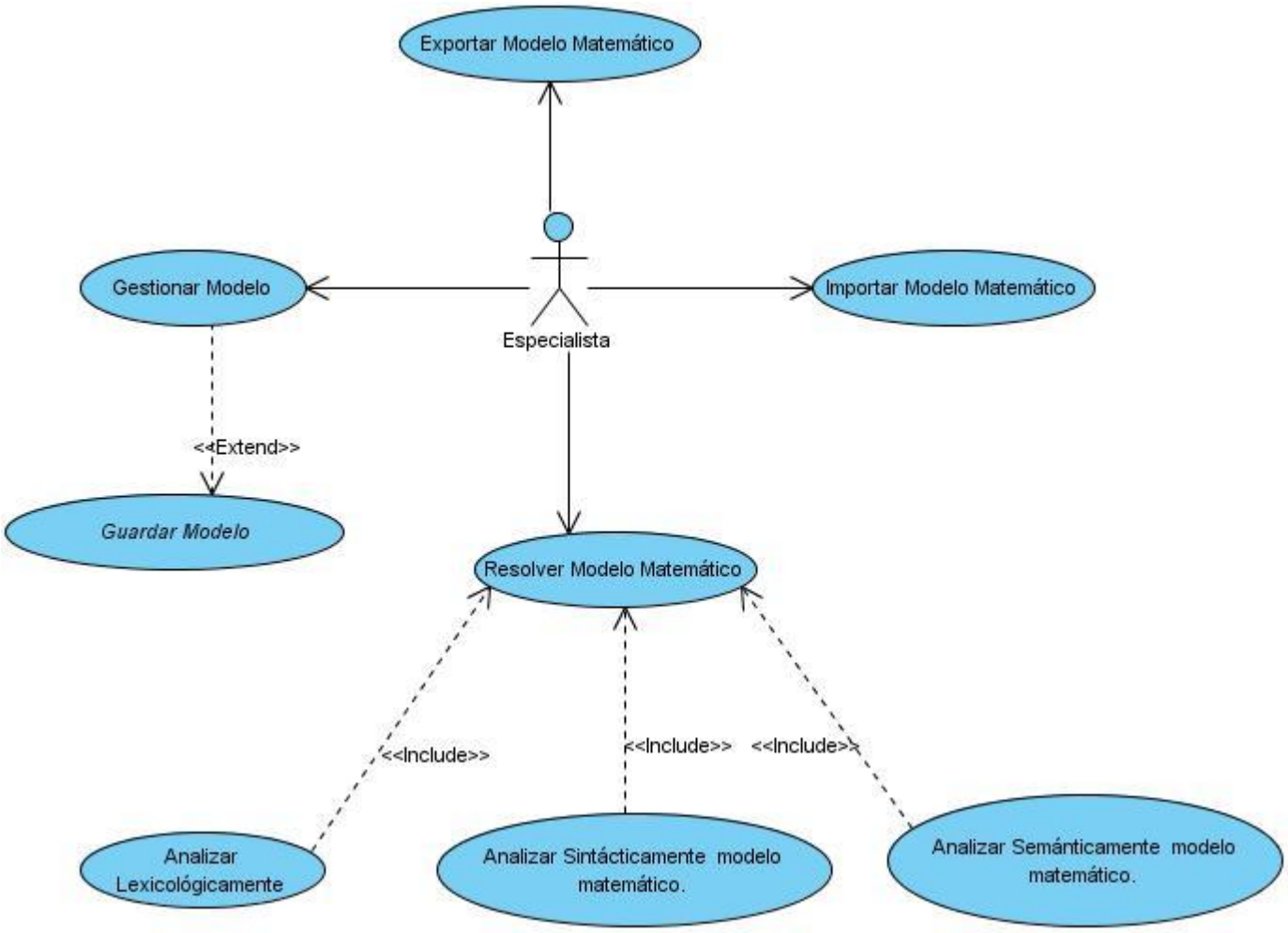

Figura 2.2 Diagrama de caso de uso del sistema.

2.6 Casos de Uso del Sistema.

1. Analizar lexicológicamente el modelo matemático.
2. Analizar sintácticamente el modelo matemático.
3. Analizar semánticamente el modelo matemático.

4. Resolver el modelo matemático.
5. Gestionar modelo matemático.
6. Importar modelo matemático.
7. Exportar modelo matemático.
8. Guardar Modelo matemático.

2.6.1 Descripción textual de los Casos de Uso (CU).

Tabla 2.2: Descripción del Caso de Uso Resolver modelo matemático.

Nombre del Caso de Uso	**Resolver modelo matemático.**
Actores	Especialista.
Resumen	El CU comienza cuando se solicita resolver el modelo matemático, el sistema después de analizar léxica, sintáctica y semánticamente el modelo, resuelve el mismo terminando así el caso de uso.
Referencias	R4
Precondiciones	Que esté creado el modelo. Que esté cargado el modelo.
Post condiciones	Se resuelve el modelo.

Requerimientos especiales	
Flujo normal de los eventos	
Acción del actor	Respuesta del Sistema
1. Solicita resolver el modelo matemático.	2. Sistema muestra los modelos existentes.
3. Selecciona el modelo matemático a resolver.	4. Carga el modelo matemático.
	5. Va al CU “Analizar Lexicológicamente”.
	6. Va al CU “Analizar Sintácticamente”.
	7. Va al CU “Analizar Semánticamente”.
	8. La biblioteca GSL se encarga de hacer los cálculos pertinentes correspondientes al modelo.
	9. Resolver matemáticamente el modelo.
	10. Termina el Caso de Uso.

Tabla 2.3: Descripción del Caso de Uso Analizar lexicológicamente el modelo matemático.

Nombre del Caso de Uso	**Analizar Lexicológicamente el modelo matemático.**
Actores	CU “Resolver modelo matemático”.
Resumen	El CU comienza cuando se solicita analizar lexicológicamente el modelo, el sistema analiza léxicamente el modelo, terminando así el caso de uso.
Referencias	R1
Precondiciones	El modelo esté cargado.
Post condiciones	Modelo analizado Lexicológicamente.
Requerimientos especiales	
Flujo normal de los eventos	
Acción del actor	Respuesta del Sistema
1. Solicita analizar Lexicológicamente el modelo matemático a resolver.	2. El sistema lee los caracteres de entrada.

	3. El sistema chequea los tokens del sistema o del txt.
	4. El sistema envía un mensaje confirmando el correcto análisis léxico.
	5. Termina el Caso de Uso
Flujo alterno: Error en el chequeo de sintaxis.	
Acción del actor	Respuesta del Sistema
	3.1. El sistema detectó un error en el analizador léxico o un token no definido en el sistema.
	3.2. Se envía un mensaje informando el error léxico detectado.
	3.3. Termina el Caso de Uso

Tabla 2.4: Descripción del Caso de Uso Analizar sintácticamente el modelo matemático.

Nombre del Caso de	**Analizar Sintácticamente el modelo matemático.**

Uso	
Actores	CU “Resolver modelo matemático”.
Resumen	El CU comienza cuando se solicita analizar sintácticamente el modelo, el sistema analiza la sintaxis del modelo, terminando así el caso de uso.
Referencias	R2
Precondiciones	El modelo esté cargado.
Post condiciones	Modelo analizado sintácticamente.
Requerimientos especiales	

Flujo normal de los eventos	
Acción del actor	Respuesta del Sistema
1. Solicita analizar Sintácticamente el modelo matemático a resolver.	2. El sistema chequea que los caracteres entrados por el usuario, estén definidos.
	3. El sistema chequea ambigüedades en la gramática.

	4. El sistema envía un mensaje confirmando el correcto análisis sintáctico.
	5. Termina el Caso de Uso
Flujo alterno: Error en el chequeo de sintaxis.	
Acción del actor	Respuesta del Sistema
	2.1 El sistema detectó un error de sintaxis.
	2.2 Se envía un mensaje informando el error sintáctico detectado.
	2.3 Terminando así el Caso de Uso.
Flujo alterno: Error en el chequeo de ambigüedades.	
Acción del actor	Respuesta del Sistema
	3.1 El sistema detectó una ambigüedad en la gramática.
	3.2 Se envía un mensaje informando el error de reconocimiento detectado.
	3.3 Terminando así el Caso de Uso.

Tabla 2.5: Descripción del Caso de Uso Analizar semánticamente modelo matemático.

Nombre del Caso de Uso	**Analizar Semánticamente modelo matemático.**
Actores	CU “Resolver modelo matemático”.
Resumen	El CU comienza cuando se solicita analizar semánticamente el modelo, el sistema reconoce las operaciones, analiza la lógica entre ellas, terminando así el caso de uso.
Referencias	R3
Precondiciones	Que se haya hecho el análisis sintáctico.
Post condiciones	Modelo analizado semánticamente.
Requerimientos especiales	

Flujo normal de los eventos	
Acción del actor	Respuesta del Sistema
1. Se analiza semánticamente el modelo matemático a resolver.	2. Comprueba la semántica del lenguaje.

	3. Verifica las operaciones que se puedan aplicar.
	4. El sistema infiere el tipo de cada construcción del lenguaje.
	5. El sistema aplica las distintas reglas de inferencia descritas.
	6. El sistema envía un mensaje confirmando el correcto análisis semántico.
	7. Termina así el caso de uso.
Flujo alterno: Error en la semántica del lenguaje.	
Acción del actor	Respuesta del Sistema
	4.1 El sistema detectó un error en la semántica.
	4.2 El sistema manda un mensaje de error.
	4.3 Termina caso de uso.
Flujo alterno: Error en verificación de las operaciones.	
Acción del actor	Respuesta del Sistema

	4.4 El sistema detectó un error en la lógica de las operaciones.
	4.5 El sistema manda un mensaje de error.
	4.6 Termina el Caso de Uso.

Tabla 2.6: Descripción del Caso de Uso Gestionar modelo matemático.

Nombre del Caso de Uso	**Gestionar modelo matemático**.
Actores	Especialista
Resumen	El CU se inicia cuando el Especialista va a realizar algunas de las siguientes operaciones: • **Crear nuevo Modelo**: cuando por primera vez el especialista inserta datos necesarios para una simulación, creando así el modelo, finalizando el CU.
Referencias	R5 (R5.1)
Precondiciones	Que esté el sistema corriendo.
Post condiciones	En dependencia de la acción del Especialista: • Se crea un nuevo modelo.

Requerimientos especiales	
Flujo normal de los eventos:	
Acción del actor	Respuesta del Sistema
1.El Especialista selecciona cuál acción realizar: a. Crear nuevo modelo.	2. El sistema en dependencia de la acción solicitada por el Especialista muestra en consola. **a.** Crear nuevo Modelo: ir a la **sección "Crear nuevo Modelo".**
Sección "Crear nuevo Modelo": Flujo Normal de Eventos	
Acción del actor	Respuesta del Sistema
1. El Especialista necesita crear un modelo .txt.	2. El sistema muestra la consola para crear modelo.
3. El Especialista provee al sistema de variables que van a insertarse en un modelo nuevo.	4. El sistema verifica que no exista el modelo.
	5. Crea el modelo.

	6. Va al CU “Guardar Modelo”.
	7. Termina así el caso de uso
Sección “Crear nuevo Modelo”: Flujo Alternativo de Eventos	
	4.1 El sistema emite un mensaje de que ya estaba creado el modelo.

Tabla 2.7: Descripción del Caso de Uso Importar modelo matemático.

Nombre del Caso de Uso	**Importar modelo matemático.**
Actores	Especialista
Resumen	El CU comienza cuando el usuario selecciona la opción de importar un modelo, el sistema importa el modelo, terminando así el caso de uso.
Referencias	R6
Precondiciones	Que exista el modelo matemático.
Post condiciones	Modelo importado correctamente.
Requerimientos especiales	
Flujo Normal:”Importar modelo”	

Acción de actor	Respuesta del sistema
1. El Especialista desea importar un modelo .txt.	2. El sistema muestra un menú para importar modelo.
3. El Especialista selecciona el archivo que desea importar.	4. Verifica que el archivo escogido sea el correcto.
	5. El sistema carga el archivo.
	6. Termina así el caso de uso.

Tabla 2.8: Descripción del Caso de Uso Exportar modelo matemático.

Nombre del Caso de Uso	**Exportar modelo matemático.**
Actores	Especialista
Resumen	El CU comienza cuando el usuario selecciona la opción exportar un modelo, el sistema exporta el modelo, terminando así el caso de uso.
Referencias	R7
Precondiciones	Necesidad de exportar el modelo matemático.

Post condiciones	Que exporte correctamente el modelo matemático.
Requerimientos especiales	

Flujo Normal:"Exportar modelo"	
Acción de actor	Respuesta del sistema
1. El Especialista desea exportar un modelo .txt.	2. El sistema muestra un menú con los modelos existentes para exportar modelo.
3. El Especialista selecciona el archivo que desea exportar.	4. Verifica archivo.
	5. Carga el modelo.
	6. El sistema exporta el archivo como un fichero .txt.
	7. Terminando así el Caso de Uso.

Tabla 2.9: Descripción del Caso de Uso Guardar modelo matemático.

Nombre del Caso de Uso	**Guardar modelo matemático**.
Actores	Gestionar modelo matemático

Resumen	El CU comienza cuando el usuario desea guardar el modelo, el sistema guarda el modelo, terminando así el caso de uso.
Referencias	R8
Precondiciones	Que esté creado o modificado el modelo matemático.
Post condiciones	Modelo matemático guardado.
Requerimientos especiales	
Flujo Normal:"Guardar modelo"	
Acción de actor	Respuesta del sistema
1. El Especialista desea guardar un modelo .txt.	2. El sistema muestra un menú con los modelos existentes para guardar modelo.
3. El Especialista selecciona el archivo que desea guardar.	4. Verifica si es el archivo escogido.
	5. El sistema guarda el archivo como un fichero .txt.
	6. Terminando así el Caso de Uso.

2.7 Conclusiones

En este capítulo primeramente se describieron una serie de conceptos para la modelación de dominio, las relaciones entre ellos, mediante un diagrama de modelo de dominio y así entender mejor el sistema. Se definieron los actores que interactúan con el sistema. Se tuvieron en cuenta los requisitos funcionales y los no funcionales, y por último, se describió cada caso de uso, por tanto se llegaron a las siguientes conclusiones:

- Se le dio cumplimiento a las distintas tareas como: definición de los requisitos funcionales y no funcionales del sistema.
- Se desarrolló el diagrama de caso de uso del sistema.
- Se describió los casos de uso del sistema.

Capítulo 3: Diseño de la solución.

En este capítulo se describe la fase de diseño, representando los diagramas de clases del diseño, los diagramas de secuencia del diseño y el diagrama de despliegue.

3.1 Diagrama de clases del diseño organizado por paquetes.

Un diagrama de paquetes no es más que una agrupación de clases que contienen estos internamente, y que se especializan en una actividad en especial. El paquete Lexer se especializa en leer los caracteres de entrada y producir como salida una secuencia de componentes léxicos, el paquete Scanner tiene su función principal es reconocer una secuencia de tokens en base a una gramática es correcta y el Parser se encarga de detectar la validez de las sentencias aceptadas en el Scanner.

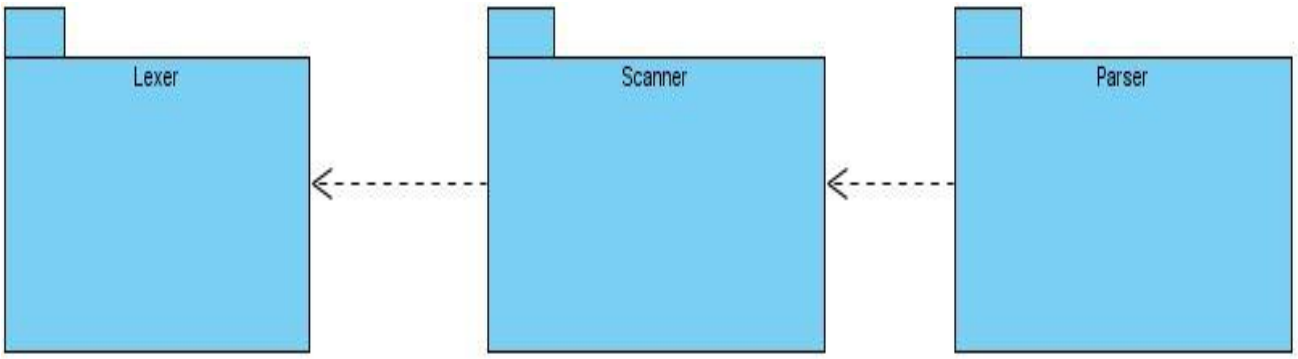

Figura 3.1 Diagrama organizado por paquetes.

3.1.1 Diagrama del paquete Lexer.

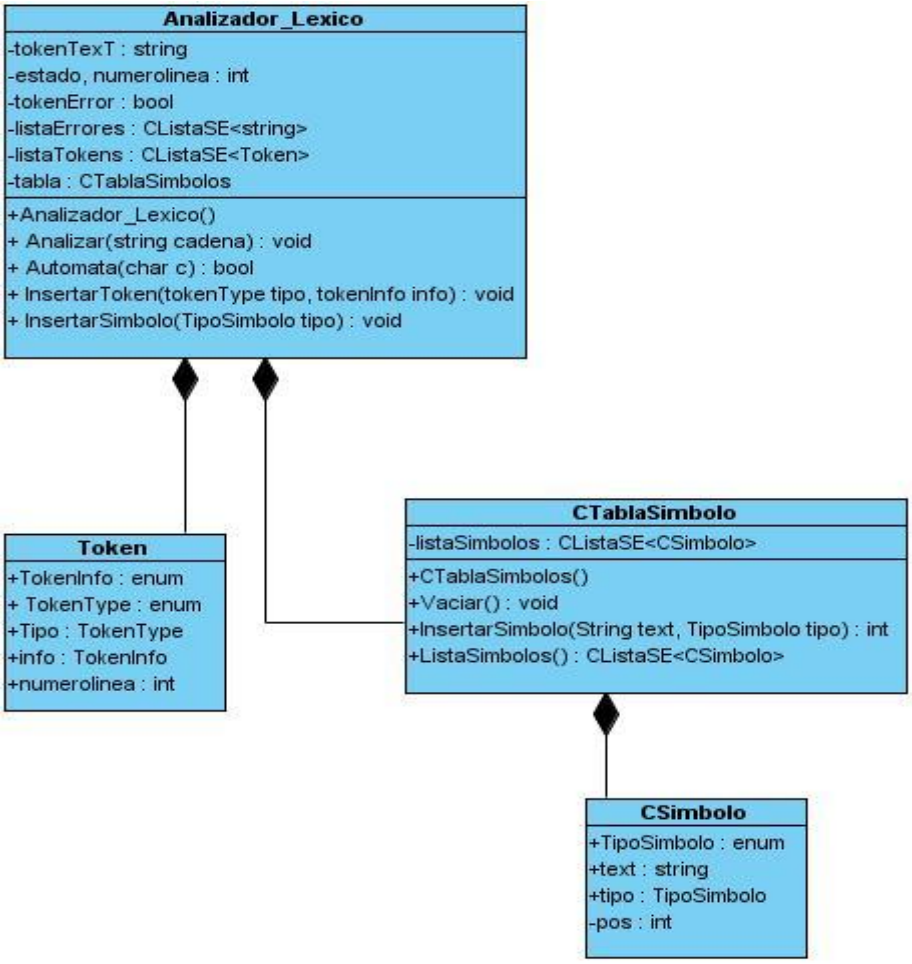

Figura 3.2 Paquete Lexer.

3.1.2 Diagrama del Paquete Scanner.

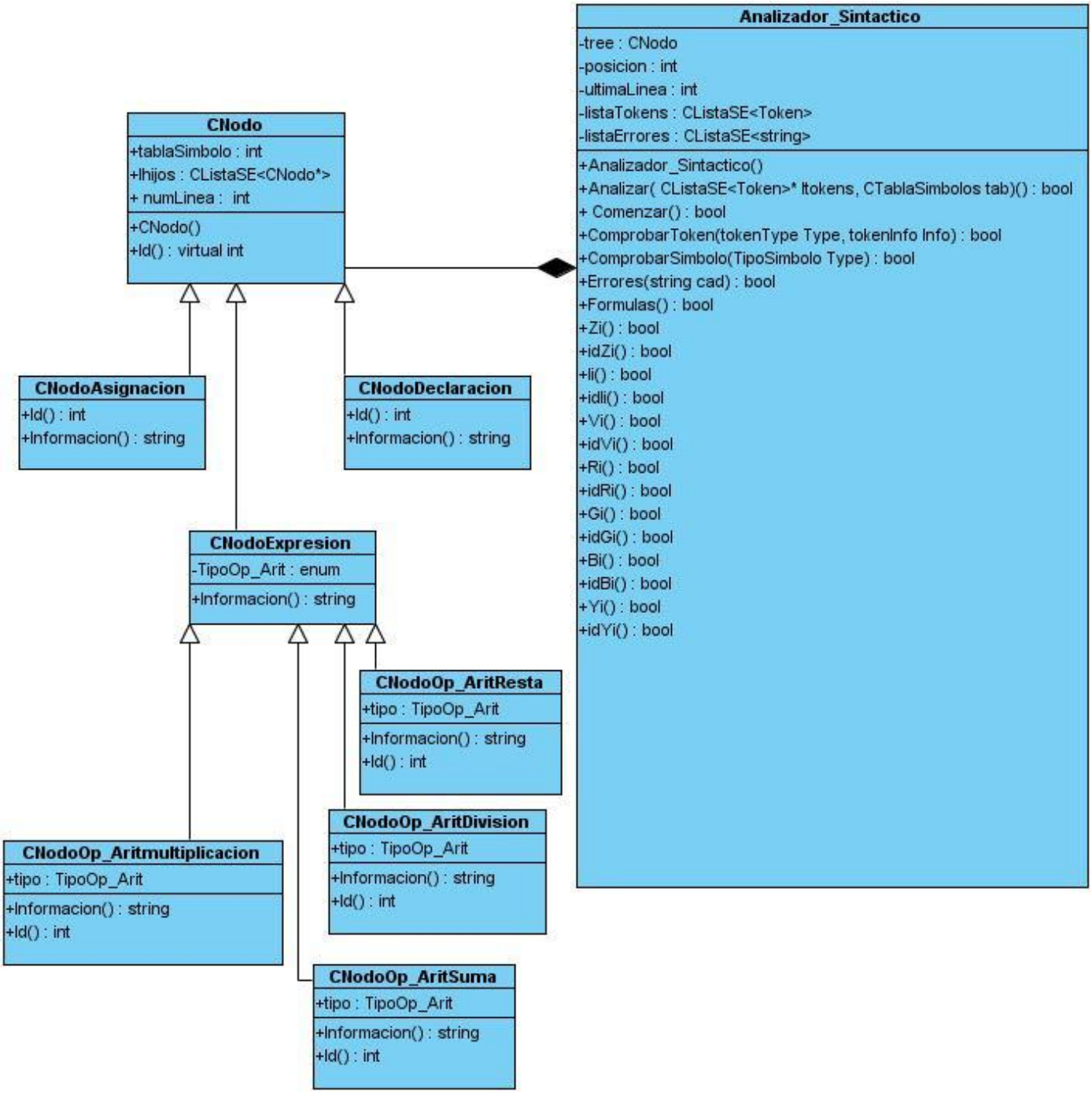

Figura 3.3 Paquete Scanner.

3.1.3 Diagrama del paquete Parser.

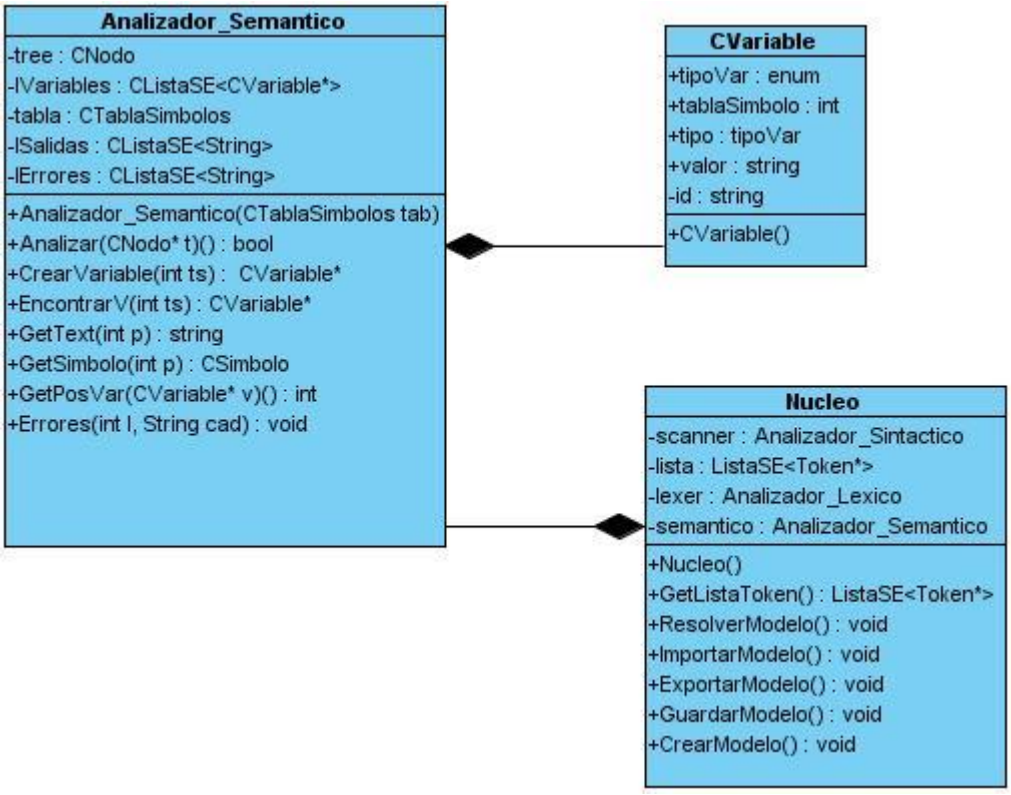

Figura 3.4 Paquete Parser.

3.1.4 Diagrama de clases general

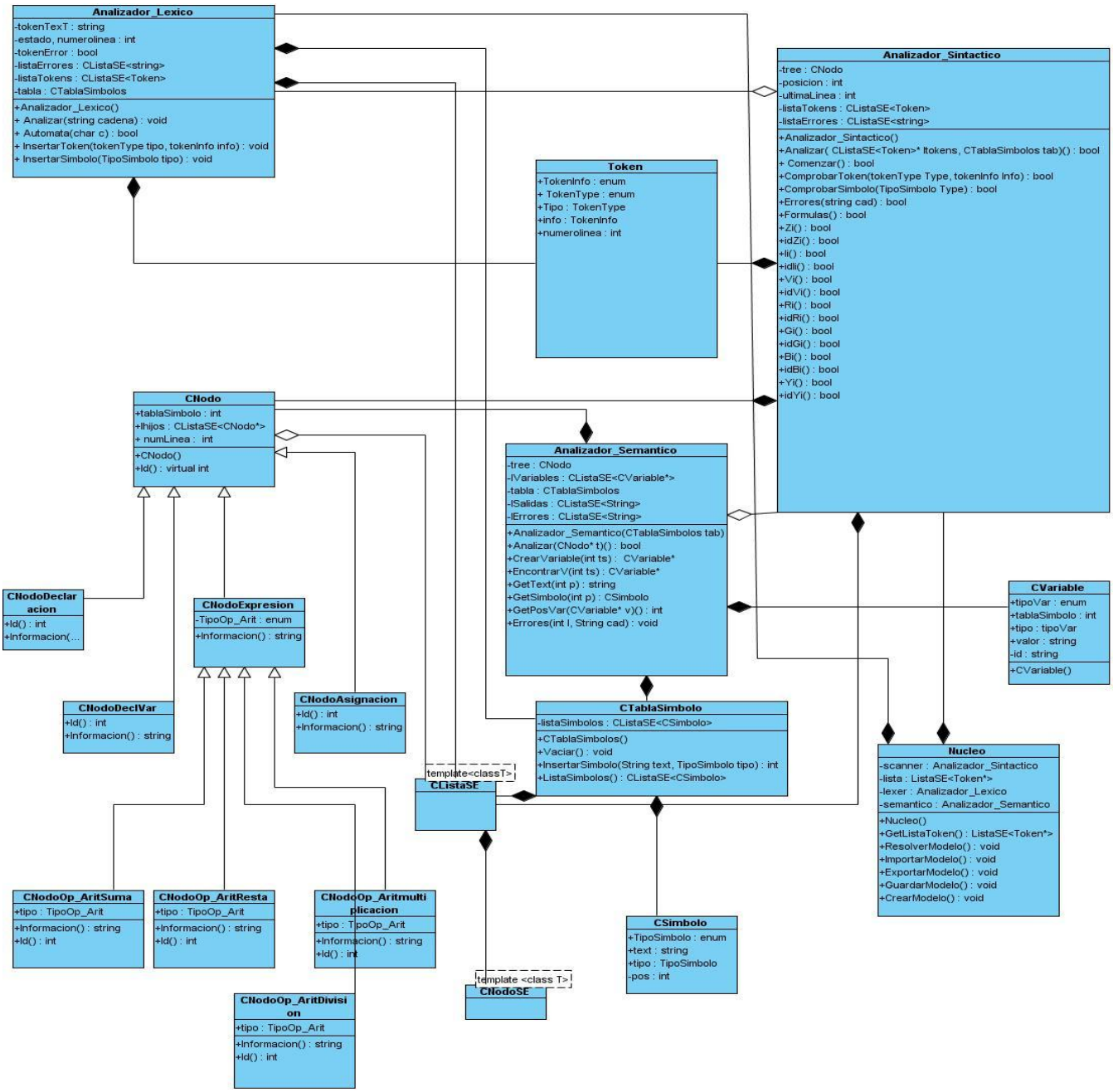

Figura 3.5 Diagrama de Clases.

3.2 Diagrama de Vista Lógica.

En este diagrama se encuentran las clases más importantes del sistema, es decir, las que tienen que ver más estrechamente con los casos de uso más importantes del diagrama de caso de uso del sistema.

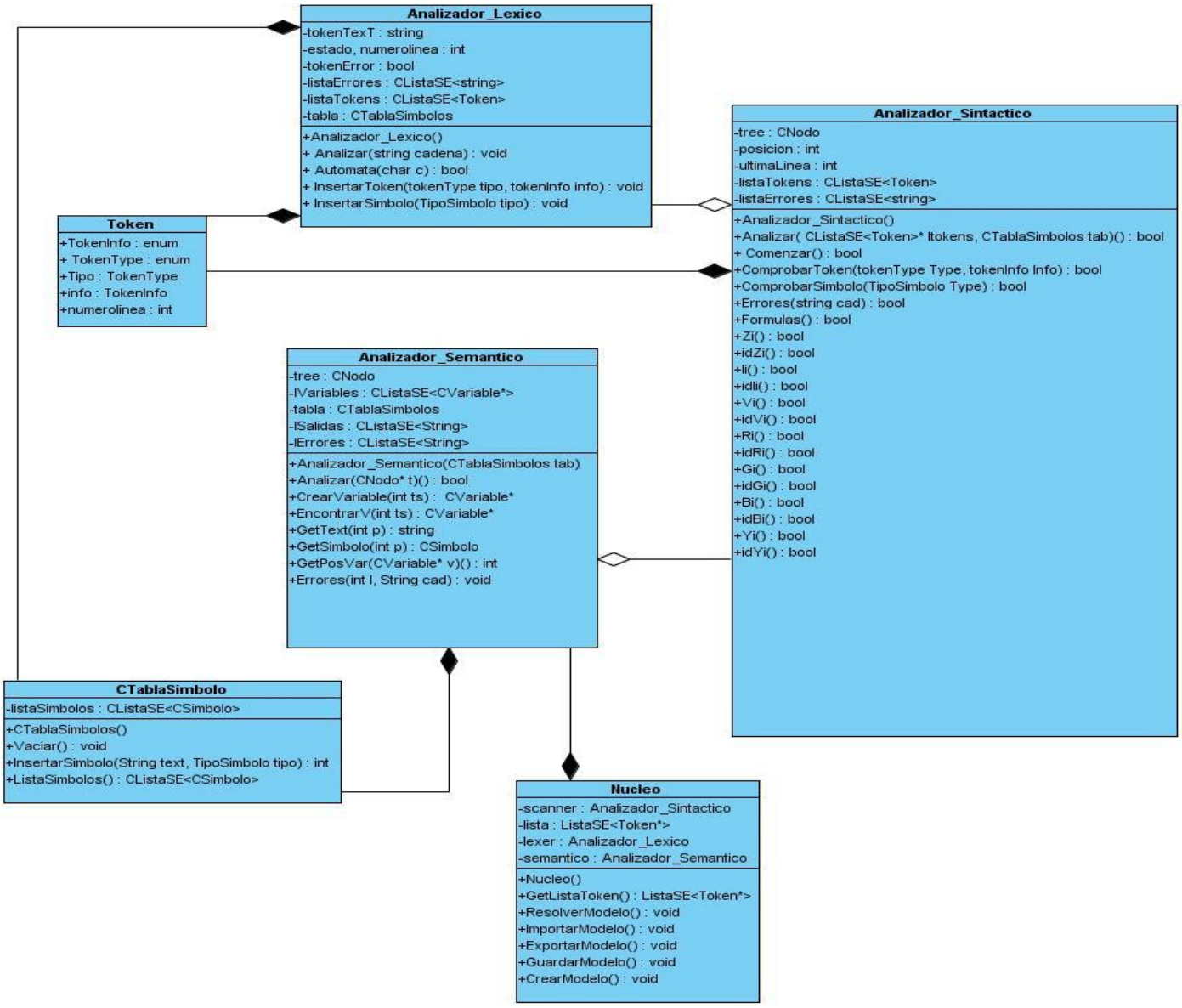

Figura 3.6 Vista Lógica del Sistema.

3.2.1 Descripción de las principales clases del sistema.

Ver anexo 1.

3.3 Diagrama de Casos de Uso Arquitectónicamente Significativos.

Un diagrama de casos de uso del sistema representa gráficamente los procesos y su interacción con los actores. Se representan los casos de uso críticos que son los más importantes para los usuarios porque cubren las principales tareas o funciones que el sistema ha de realizar. Definen la arquitectura básica. [17]

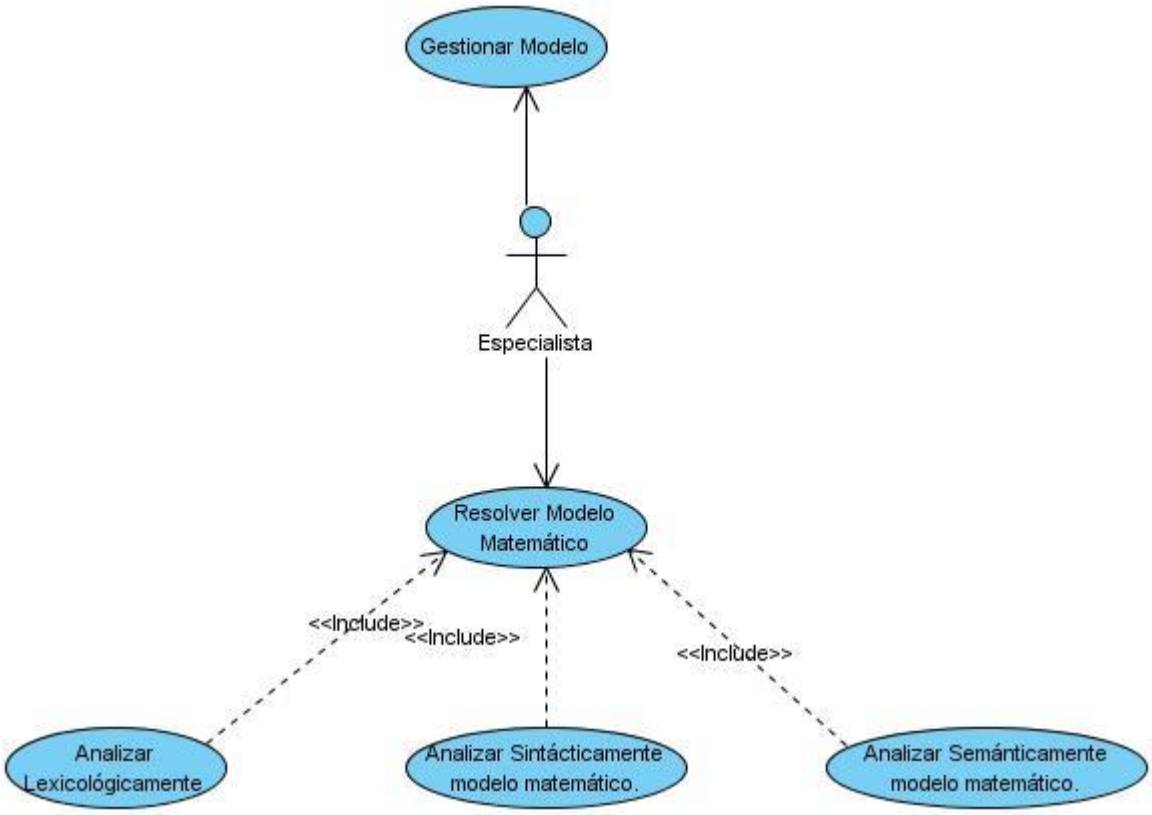

Figura 3.7 Diagrama Caso uso arquitectónicamente significativos.

3.4 Diagramas de interacción del diseño.

- **Diagramas de secuencia.**

Diagrama de secuencia del Caso de uso Analizar Léxicamente.

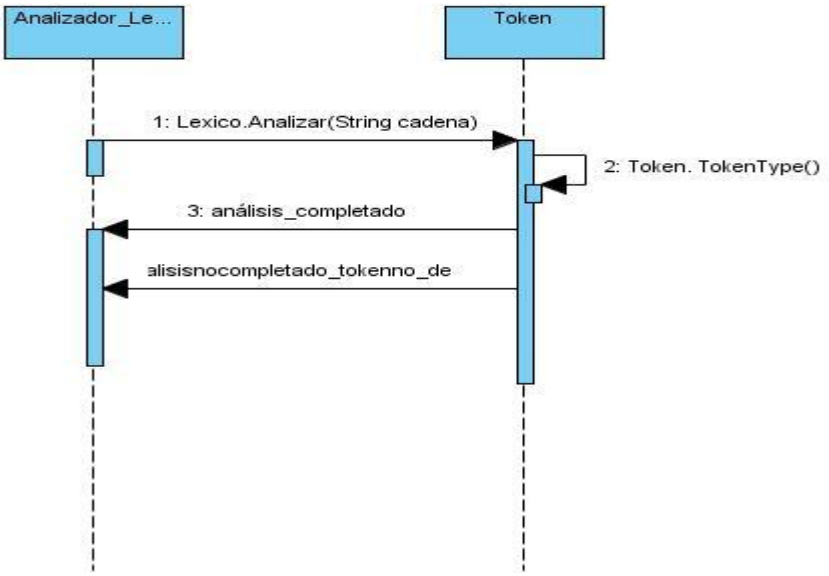

Figura 3.7 Diagrama de Secuencia Analizar Léxicamente.

Diagrama de secuencia del Caso de uso Analizar Sintácticamente.

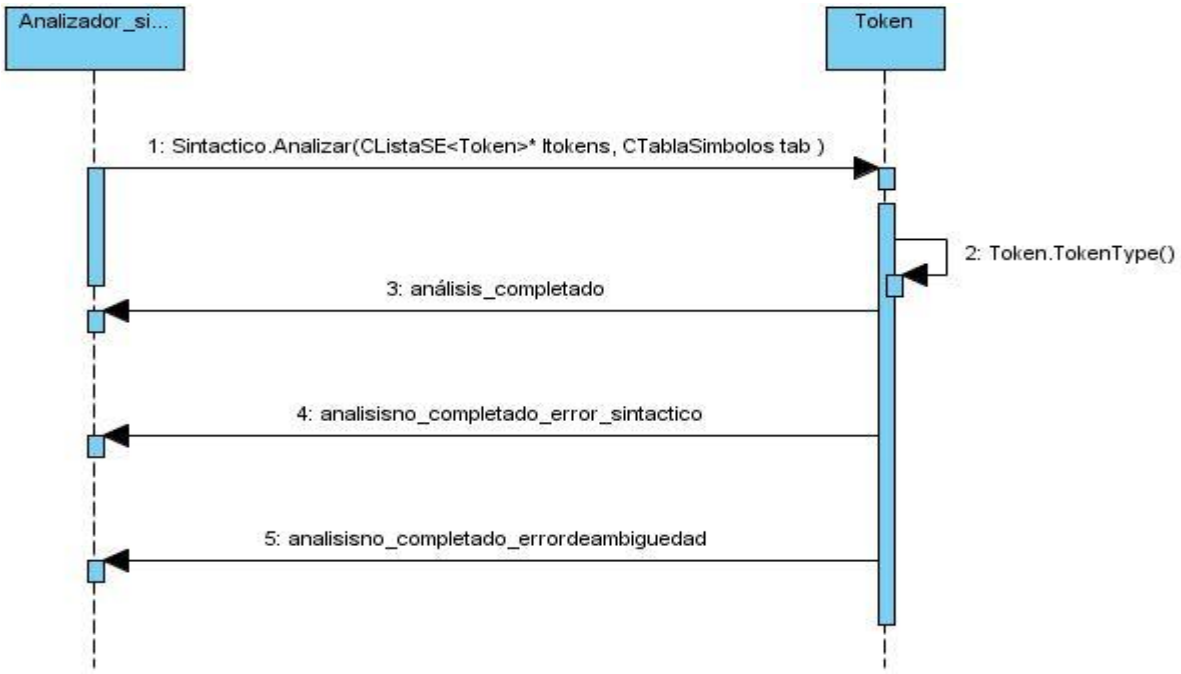

Figura 3.8 Diagrama de Secuencia Analizar Sintácticamente.

Diagrama de secuencia del Caso de uso Analizar Semánticamente.

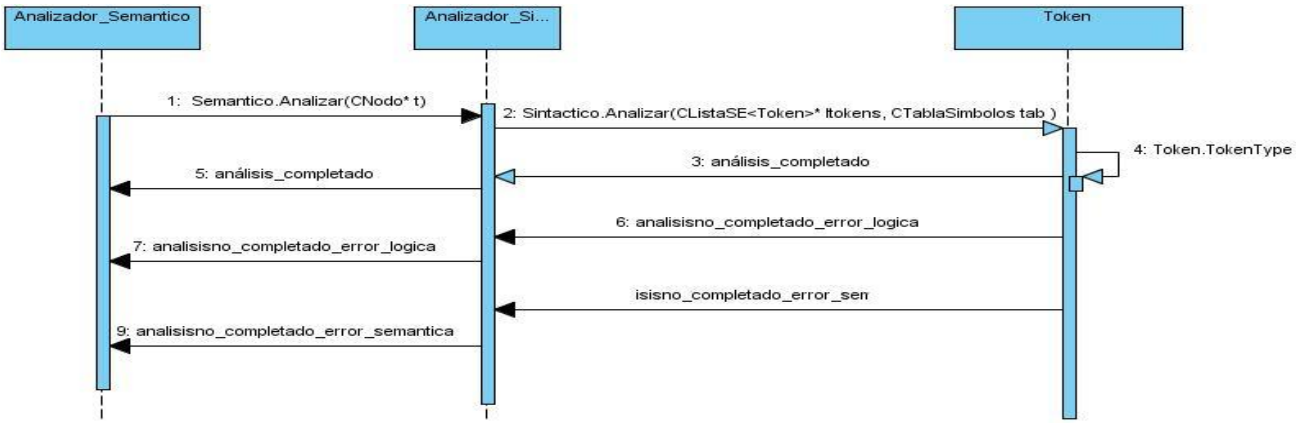

Figura 3.9 Diagrama de Secuencia Analizar Semánticamente.

Diagrama de secuencia del Caso de uso Resolver Modelo Matemático.

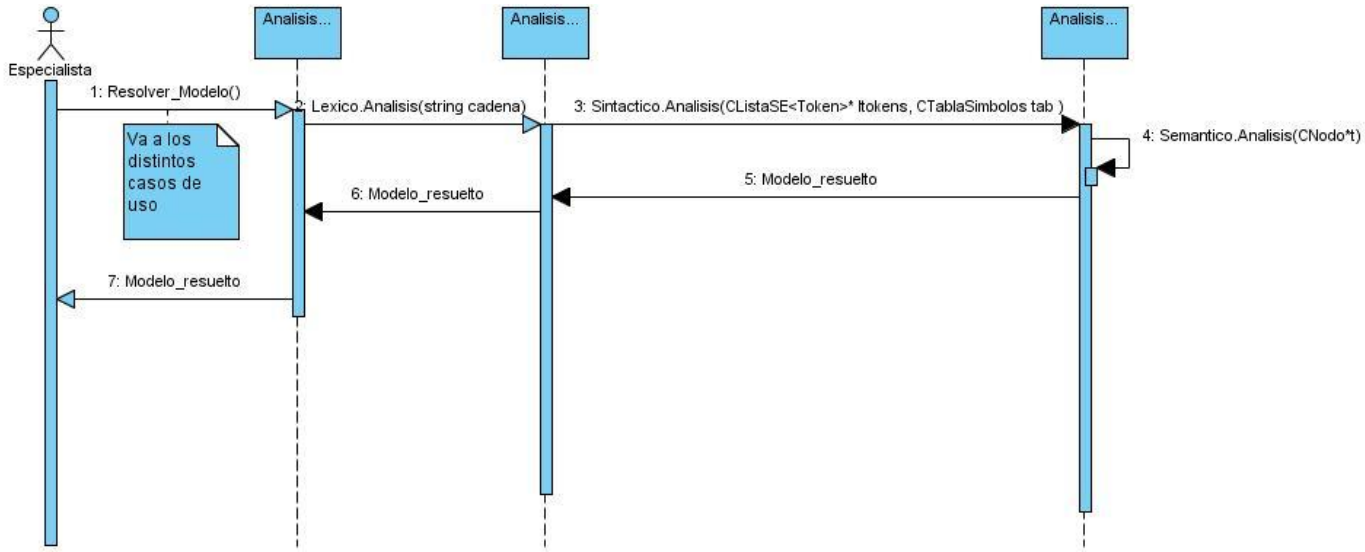

Figura 3.10 Diagrama de Secuencia Resolver modelo Matemático.

Diagrama de secuencia del Caso de uso Crear Modelo.

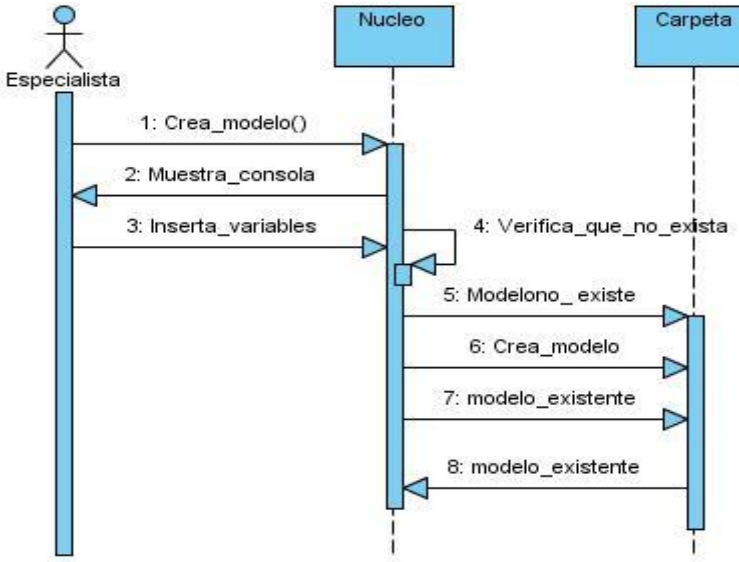

Figura 3.11 Diagrama de Secuencia Crear Modelo.

Resto de los diagramas en Anexo 2.

3.3 Patrones de diseño.

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Un sistema bien estructurado está lleno de patrones. Un patrón 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. [18]

Proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifica y describe formas de solucionar problemas que ocurren de forma frecuente en el desarrollo. Están basados en la recopilación del conocimiento de los expertos en desarrollo de software. [18]

3.3.1 Patrones utilizados GRASP.

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

En el desarrollo del sistema se aplicaron los patrones básicos GRAPS:

- ✓ Experto: asignar una responsabilidad al experto en información, la clase que cuenta con la información necesaria para cumplir la responsabilidad.
- ✓ Alta Cohesión: Asignar a las clases responsabilidades que trabajen sobre una misma área de la aplicación y que no tengan mucha complejidad.
- ✓ Bajo Acoplamiento: Asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible. [18]

3.4 Estilo arquitectónico. Diagrama de Despliegue.

3.4.1 Estilo Arquitectónico.

La Arquitectura de Software, también denominada *Arquitectura lógica*, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del Software. Es una vista estructural de alto nivel, ocurre muy tempranamente en el ciclo de vida y define los estilos o grupos de estilos adecuados para cumplir con los requerimientos no funcionales. [18]

La arquitectura de software es importante como disciplina debido a que los sistemas de software crecen de forma tal que resulta muy complicado que sean diseñados, especificados y entendidos por un solo individuo. Uno de los aspectos que motivan el estudio en este campo es el factor humano, en términos de aspectos como inspecciones de diseño, comunicación a alto nivel entre los miembros del equipo de desarrollo, reutilización de componentes y comparación a alto nivel de diseños alternativos. [19]

Este sistema utilizará una arquitectura por Plugins; ya que con esta arquitectura se puede empaquetar y distribuir un conjunto de archivos del proyecto. Al igual que un proyecto, un plugin puede tener clases, helpers, configuraciones, tareas, módulos, esquemas, e incluso recursos Web. El primer uso de los plugins es facilitar el intercambio de código entre aplicaciones, o incluso entre distintos proyectos. Los Plugins dan una manera de compartir más componentes entre aplicaciones. Pueden anexarse a otro programa para aumentar las funcionalidades, sin afectar las ya existentes ni complicar el desarrollo del programa principal. Un plugins es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande. [20]

3.4.2 Diagrama de Despliegue.

El diagrama de despliegue no es más que una representación de cómo va a quedar el sistema, todos los recursos que va a utilizar por ejemplo: servidores, PC clientes y otros. No es más que la distribución física de dicho sistema, distribuyendo las funcionalidades del mismo entre los nodos que va utilizar.

Figura 3.12 Diagrama de Despliegue.

3.5 Conclusiones.

En este capítulo se hizo una descripción de la solución propuesta, utilizando la programación orientada a objeto, se presentaron los diagrama de clases, de secuencia del diseño y el diagrama de despliegue así como los de Vista Lógica y de Vista de Casos de Uso, se llegaron a las conclusiones siguientes:

- Se le dio cumplimiento a las tareas de este capítulo.
- Los patrones utilizados fueron los GRAPS y dentro de estos el experto, bajo acoplamiento y el de alta cohesión.
- Se definieron los diagramas vista de caso de uso y el de despliegue, así como también el de Vista Lógica que va a tener un total de 6 clases.

Capítulo 4: Implementación y prueba del Sistema.

En este capítulo sea aborda la programación realizada partiendo de los requerimientos identificados y el diagrama de clases del diseño elaborado. Además se muestran ejemplos de las validaciones del sistema realizadas.

4.1 Diagrama de Componentes.

Un diagrama de componentes incluye a los componentes y archivos que se utilizan para ensamblar y hacer disponible el sistema físico; y un componente es una parte física y reemplazable de un sistema que se conforma con un conjunto de interfaces y proporciona la realización de dicho conjunto. Se usan para modelar los elementos físicos que pueden hallarse en un nodo por lo que empaquetan elementos como clases, colaboraciones e interfaces. Desde el punto de vista del diagrama de componentes, se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. A continuación se muestra el diagrama de componentes del software en cuestión:

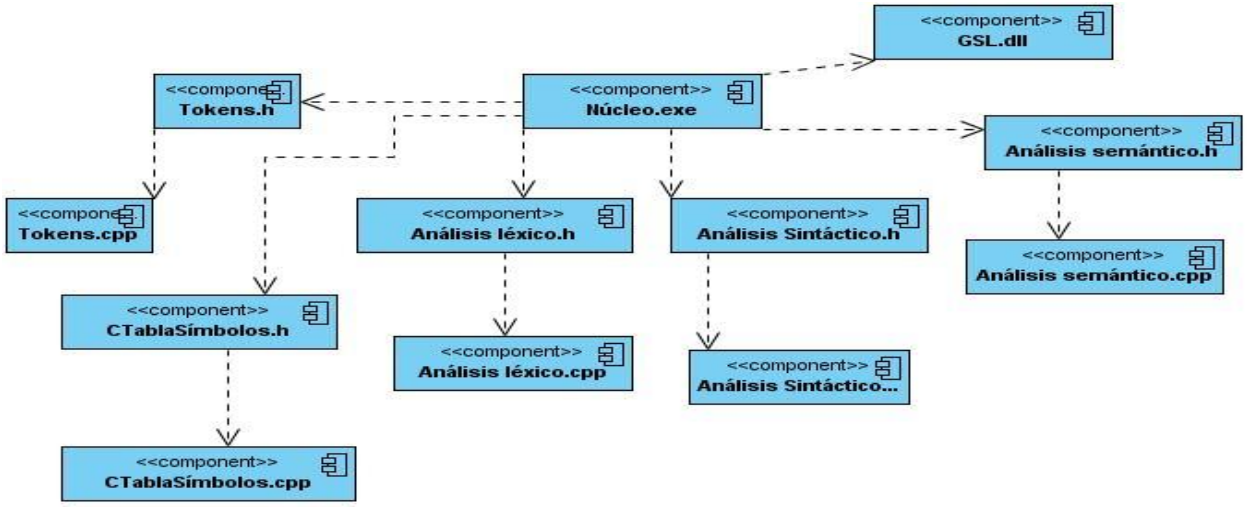

Figura 4.1 Diagrama de Componentes.

4.2 Pruebas del Sistema.

A la aplicación se le realizaron pruebas de caja negra para verificar que el programa cumple adecuadamente con los requisitos especificados. Se le entran un número de datos que ayuden a la ejecución de estos casos de prueba, sin importar los detalles internos del programa, estos datos pueden ser incorrectos o no a fin de encontrar un error o que de un resultado esperado.

Fue tomado un ejemplo resuelto del Departamento de Teoría de la Señal y Comunicaciones de la Universidad de Alcalá, donde se exponen varios problemas resueltos sobre corriente eléctrica en los transformadores.[21]

Esta prueba se hizo basada en el caso de uso “Resolver modelo matemático”, obteniéndose el siguiente resultado:

Nombre del escenario	Flujo donde empieza	Alternativo
Escenario 1 “Resultado exitoso”	Flujo Básico	

Tabla # 1 Matriz Parcial de escenarios

Id del escenario	Escenario	Fórmula	Variable1	Variable2	Respuesta del Sistema
EC 1	Escenario 1: “Resultado exitoso”	V	V	V	El sistema muestra el resultado obtenido al especialista.

Tabla # 2 Matriz de Casos de Prueba

Id del escenario	Escenario	Fórmula	Variable1	Variable2	Resultado	Respuesta del Sistema	Resultado de la prueba
EC 1	Escenario 1: "Resultado exitoso"	$X=\sqrt{(Z^2-R^2)}$	Z= 87.12 Ω (Impedancia)	R= 10.68 Ω (Resistencia)	X= 86.4629 Ω (Reactancia)	El sistema muestra el resultado obtenido al especialista.	Al realizar la prueba, se obtuvieron los valores esperados.

Tabla # 3 Matriz de Casos de Prueba con los valores de los datos.

Importancia del caso de uso(CU)	Descripción	Observación
3	Alta	El resultado tomado del ejemplo, se muestra con dos cifras decimales luego de la coma, sin embargo el resultado obtenido del núcleo matemático es el mismo, pero con cuatro cifras decimales luego de la coma, obteniéndose un valor de mayor exactitud.

Tabla # 4 Clasificación de las no conformidades atendiendo al impacto que provocan.

Esta prueba se hizo basada en el caso de uso "Resolver modelo matemático", obteniéndose el siguiente resultado:

Nombre del escenario	Flujo donde empieza	Alternativo
Escenario 1: "Resultado defectuoso"	Flujo Básico	

Tabla # 5 Matriz Parcial de escenarios

Id del escenario	Escenario	Fórmula	Variable1	Variable2	Respuesta del Sistema
EC 1	Escenario 1: “Resultado defectuoso”	F	V	V	El sistema muestra un error al especialista, dándole a conocer un error de gramática.

Tabla # 6 Matriz de Casos de Prueba

Id del escenario	Escenario	Fórmula	Variable1	Variable2	Resultado	Respuesta del Sistema	Resultado de la prueba
EC 1	Escenario 1: “Resultado defectuoso”	I=V*R	V= 10.3 V (Voltaje)	R= 10.68 Ω (Resistencia)	I= (Intensidad)	El sistema muestra un error de sintaxis.	Al realizar la prueba, se lanzó un error indicando que la fórmula estaba escrita incorrectamente.

Tabla # 7 Matriz de Casos de Prueba con los valores de los datos.

Importancia del CU	Descripción	Observación
3	Alta	Se observó con la prueba anterior que si el especialista no escribe la fórmula correctamente, no puede calcular el resultado.

Tabla # 8 Clasificación de las no conformidades atendiendo al impacto que provocan.

Esta prueba se hizo basada en el caso de uso “Resolver modelo matemático”, obteniéndose el siguiente resultado:

Nombre del escenario	Flujo donde empieza	Alternativo

Escenario 1 “Resultado exitoso”	Flujo Básico	

Tabla # 9 Matriz Parcial de escenarios

Id del escenario	**Escenario**	**Fórmula**	**Variable1**	**Variable2**	**Respuesta del Sistema**
EC 1	Escenario 1: “Resultado exitoso”	F	V	V	El sistema muestra el resultado obtenido al especialista.

Tabla # 10 Matriz de Casos de Prueba

Id del escenario	**Escenario**	**Fórmula**	**Variable1**	**Variable2**	**Resultado**	**Respuesta del Sistema**	**Resultado de la prueba**
EC 1	Escenario 1: “Resultado defectuoso”	Y= $(R-X)/(R^2+X^2)$	R= 28.64 Ω (Resistencia)	X= 11.32 Ω (Reactancia)	Y= 2.5304 Ω (Admitancia)	El sistema muestra el resultado obtenido al especialista.	Al realizar la prueba, se obtuvieron los valores esperados.

Tabla # 11 Matriz de Casos de Prueba con los valores de los datos.

Importancia del CU	**Descripción**	**Observación**
3	Alta	Este caso de prueba también da un resultado esperado y con la misma exactitud que el primer caso de prueba.

Tabla # 12 Clasificación de las no conformidades atendiendo al impacto que provocan.

Esta prueba se hizo basada en el caso de uso “Resolver modelo matemático”, obteniéndose el siguiente resultado:

Nombre del escenario	Flujo donde empieza	Alternativo
Escenario 1 “Resultado defectuoso”	Flujo Básico	

Tabla # 13 Matriz Parcial de escenarios

Id del escenario	Escenario	Fórmula	Variable1	Variable2	Respuesta del Sistema
EC 1	Escenario 1: “Resultado exitoso”	F	V	V	El sistema muestra un error de sintaxis.

Tabla # 14 Matriz de Casos de Prueba

Id del escenario	Escenario	Fórmula	Variable1	Variable2	Resultado	Respuesta del Sistema	Resultado de la prueba
EC 1	Escenario 1: “Resultado exitoso”	B-X/(R^2+X^2)	R= 28.64 Ω (Resistencia)	X= 11.32 Ω (Reactancia)	B= (Admitancia)	El sistema muestra un error.	Al realizar la prueba, pudieron obtener los resultados esperados.

Tabla # 15 Matriz de Casos de Prueba con los valores de los datos.

Importancia del CU	Descripción	Observación

3	Alta	No se pudo obtener ningún resultado porque faltó el signo igual.

Tabla # 16 Clasificación de las no conformidades atendiendo al impacto que provocan.

Se realizaron un total de 4 casos de pruebas, obteniéndose resultados satisfactorios. En el primer caso, se tomó como muestra un caso de estudio real y se obtuvo un resultado mejor que el esperado, pues el valor obtenido fue de mayor exactitud. En el segundo caso de estudio se obtuvo el valor esperado y en los 2 casos restantes se entraron fórmulas incorrectas, de las cuales el sistema mostró el error de sintaxis correspondiente.

4.3 Conclusiones.

- Se desarrolló el diagrama de componentes, donde se recogen los ficheros de las principales clases del sistema, así como la biblioteca que va a hacer los cálculos.
- Se desarrolló una propuesta de arquitectura donde se definen los principales casos de uso y clases arquitectónicamente significativos para el núcleo matemático del simulador eléctrico.
- Se implementó la aplicación que cumple con las especificaciones necesarias para el cumplimiento de los requerimientos, así como la realización de pruebas en busca de errores.

Conclusiones Generales.

1. Se realizó el análisis del sistema propuesto, obteniéndose los requisitos funcionales y no funcionales del mismo.
2. Se desarrolló el diseño e implementación, obteniéndose como resultado una versión funcional del producto, que se materializó en una biblioteca.
3. Se realizaron pruebas unitarias sobre diferentes fórmulas físicas, con el fin de validar las respuestas del sistema ante fórmulas erróneas y verificar el grado de exactitud de los valores calculados, evidenciando la operatividad del sistema.
4. La utilización de este núcleo matemático contribuye a la solución de los modelos eléctricos que se definen en el Simulador Eléctrico del CEDIN.

Con la propuesta y el estudio realizado se materializan los objetivos planteados al inicio de esta investigación: desarrollar un núcleo matemático que permita la definición de las ecuaciones que rigen la simulación de los modelos definidos en el Simulador Eléctrico.

Recomendaciones.

1. Generar una gramática más compleja, que abarque todas las fórmulas físicas eléctricas, para darle mayores posibilidades al simulador de aumentar el rango de simulación de circuitos.
2. Desarrollar el Simulador Eléctrico del CEDIN, para poner en funcionamiento el núcleo matemático y así poder culminar la simulación deseada.
3. Se recomienda el estudio de la biblioteca “matheval”, la cual contiene varias de las funcionalidades del sistema mucho más avanzadas, con el objetivo de mejorar la aplicación.

Referencias Bibliográficas

1. Shannon, R.E.

2. Marquez, O. [cited 2010 2/2010]; Available from: http://omarsanchez.net/conceptomod.aspx.

3. Steed, M. 1991.

4. 1991, C.e.S., *Von Glasersfeld & Steefe*. 1987. 191.

5. 02/2009 3/2010]; Available from: http://proyectopinguino.blogspot.com/2009/02/simuladores-de-circuitos-electricos-y.html.

6. José Antonio E. García Álvarez, junio 2007. Available from: asifunciona.com

7. Millman and Taub, eds. *Circuitos de pulsos digitales y de conmutación*. 1982: La Habana.

8. Hidalgo, U.M.d.S.N.d. 28/5/2009 3/2010]; Available from: http://dieumsnh.qfb.umich.mx/ELECTRO/ley%20de%20ohm.htm.

9. Soto, P.L.; Available from: http://www.mitecnologico.com/Main/LeyDeOhm.

10. 2004 3/2010]; Available from: www.electronicafacil.net/tutoriales/Leyes-Kirchoff.php.

11. Pérez, M.T. and O. Arratia.

12. Sanz, A.P. 13/06/09; Available from: http://platea.pntic.mec.es/aperez4/catalogo/Catalogo-software

13. Rosales, R.G. and D.C. Fleites, *Adaptación de Open Up para el polo Productivo de BioInformática*. 2009, UCI: La Habana.

14. 2/9/2009; Available from: http://cbasqa.wordpress.com/2008/09/02/proceso-de-desarrollo-openup.

15. Perdomo, A. and J. Pérez, *Sistema de agente de monitoreo de recursos de las computadoras del MININT*. 2009, UCI: La Habana. p. 130.

16. Reyes, E., *Sistema de autenticación para el módulo Seguridad del proyecto Guardián del ALBA*. 2009, Universidad de las Ciencias Informáticas: C.Habana.

17. Conferencia 6, Fase de Inicio. Disciplina de Requisitos, 2009-2010 Asignatura Ingeniería de software.

18. aprendizaje., E.V.d. *Conferencia 2 Arquitectura*. 2007-2008.

19. Kasman 1996.

20. symfony, avaible from: *www.symfony-project.org/jobeet/1_3/Propel/es/*

21.http://agamenon.tsc.uah.es/Asignaturas/itiei/mqe/apuntes/PME2.pdf

Bibliografía.

1. Shannon, R.E.

2. Marquez, O. [cited 2010 2/2010]; Available from: http://omarsanchez.net/conceptomod.aspx.

3. Steed, M. 1991.

4. 1991, C.e.S., *Von Glasersfeld & Steefe*. 1987. 191.

5. 02/2009 3/2010]; Available from: http://proyectopinguino.blogspot.com/2009/02/simuladores-de-circuitos-electricos-y.html.

6. José Antonio E. García Álvarez, junio 2007. Available from: asifunciona.com

7. Millman and Taub, eds. *Circuitos de pulsos digitales y de conmutación*. 1982: La Habana.

8. Hidalgo, U.M.d.S.N.d. 28/5/2009 3/2010]; Available from: http://dieumsnh.qfb.umich.mx/ELECTRO/ley%20de%20ohm.htm.

9. Soto, P.L.; Available from: http://www.mitecnologico.com/Main/LeyDeOhm.

10. 2004 3/2010]; Available from: www.electronicafacil.net/tutoriales/Leyes-Kirchoff.php.

11. Pérez, M.T. and O. Arratia.

12. Sanz, A.P. 13/06/09; Available from: http://platea.pntic.mec.es/aperez4/catalogo/Catalogo-software

13. Rosales, R.G. and D.C. Fleites, *Adaptación de Open Up para el polo Productivo de BioInformática*. 2009, UCI: La Habana.

14. 2/9/2009; Available from: http://cbasqa.wordpress.com/2008/09/02/proceso-de-desarrollo-openup.

15. Perdomo, A. and J. Pérez, *Sistema de agente de monitoreo de recursos de las computadoras del MININT*. 2009, UCI: La Habana. p. 130.

16. Reyes, E., *Sistema de autenticación para el módulo Seguridad del proyecto Guardián del ALBA*. 2009, Universidad de las Ciencias Informáticas: C.Habana.

17. Conferencia 6, Fase de Inicio. Disciplina de Requisitos, 2009-2010 Asignatura Ingeniería de software.

18. aprendizaje., E.V.d. *Conferencia 2 Arquitectura*. 2007-2008.

19. Kasman 1996.

20. symfony, avaible from: *www.symfony-project.org/jobeet/1_3/Propel/es/*

21.http://agamenon.tsc.uah.es/Asignaturas/itiei/mqe/apuntes/PME2.pdf

22. **Miguel Céspedes, José Díaz Duque.** *Transformada de Laplace.* Guantánamo : Pueblo y Educación, 1980.

23. *Conferencia 4: Fase de Análisis Léxico.* **UCI, Colectivo de autores de la catedra de programación de la.** Ciudad Habana : s.n., 2009.

24. *Conferencia5:Análisis Sintáctico.* **UCI, Colectivo de autores de la catedra de programación de la.** Ciudad Habana : s.n., 2009.

25. *Conferencia 8: Análisis Semántico .* **UCI, Colectivo de autores de la catedra de programación de la.** Ciudad Habana : s.n., 2009.

Anexo 1. Descripción de las principales clases del sistema.

Tabla 1: Descripción de la clase Token.

Nombre: Token	
Descripción: Encargada de contener la información, el tipo y el número de la línea donde se encuentra el token.	
Atributo	**Tipo**
token_type	enum
token_info	enum
numerolinea	int
Responsabilidades	
1. Nombre	public Token_Type()
Descripción.	Encargado de devolver el tipo del token.
2. Nombre	public Token_Info()
Descripción.	Encargado de devolver la información del token.
3. Nombre.	Int Numero_Linea()
Descripción.	Encargado de devolver el valor del atributo numerolinea.

Tabla 2: Descripción de la clase Analizador_Lexico.

Nombre: Analizador_Lexico.	
Descripción: Encargada de realizar el análisis léxico de la cadena que se le pasa como parámetro.	
Atributo	**Tipo**
token_text	string
numerolinea	int
listaTokens	CListaSE<Tokens>
tabla	CTablaSimbolo
Responsabilidades	

1. Nombre	string Analizar(string cadena)
Descripción.	Encargado de analizar la cadena que se le pasa por parámetro.
2. Nombre	bool Automata (char c)
Descripción.	Encargado de verificar si el caracter que se pasa por parámetro pertenece al alfabeto.
3. Nombre.	void Insertar_Token(TokenType tipo, Token_Info info)
Descripción.	Encargado de insertar el caracter en la lista de tokens.

Tabla 3: Descripción de la clase Analizador_Sintactico.

Nombre: Analizador_ Sintactico.	
Descripción: Encargada de realizar el análisis sintáctico de a la lista de tokens.	
Atributo	**Tipo**
tree	CNodo
posicion	int
ultimaLinea	int
listaToken	CListaSE<Tokens>
Responsabilidades	
1. Nombre	bool Analizar(CListaSE<Tokens>* ltoken, CTablaSimbolo tab)
Descripción.	Encargado de analizar sintácticamente cada posición de la lista de tokens.
2. Nombre	bool Comenzar ()
Descripción.	Encargado de comenzar a reconocer si el token que se está analizando, pertenece a la gramática predefinida.

Tabla 4: Descripción de la clase Analizador_Semantico.

Nombre: Analizador_ Semantico.	
Descripción: Encargada de realizar el análisis semántico.	
Atributo	**Tipo**
tree	CNodo

tabla	CTablaSimbolos
ultimaLinea	int
listaSalida	CListaSE<string>
Responsabilidades	
1. Nombre	bool Analizar(CNodo * t)
Descripción.	Encargado de analizar semánticamente el árbol que se le pasa como parámetro.
2. Nombre	CSimbolo GetSimbolo(int p)
Descripción.	Encargado de devolver el símbolo dada una posición.

Tabla 5: Descripción de la clase CTablaSimbolo.

Nombre: CTablaSimbolo.	
Descripción: Encargada de realizar el análisis semántico.	
Atributo	**Tipo**
listaSimbolos	CListaSE<CSimbolos>
Responsabilidades	
1. Nombre	void Vaciar()
Descripción.	Encargado de eliminar los elementsos de la tabla de símbolos.
2. Nombre	Void InsertarSimbolo(string text, TipoSimbolo tipo)
Descripción.	Encargado de insertar un símbolo en la tabla de símbolos.

Anexo 2. Diagramas de secuencia.

Diagrama de secuencia del Caso de uso Importar Modelo.

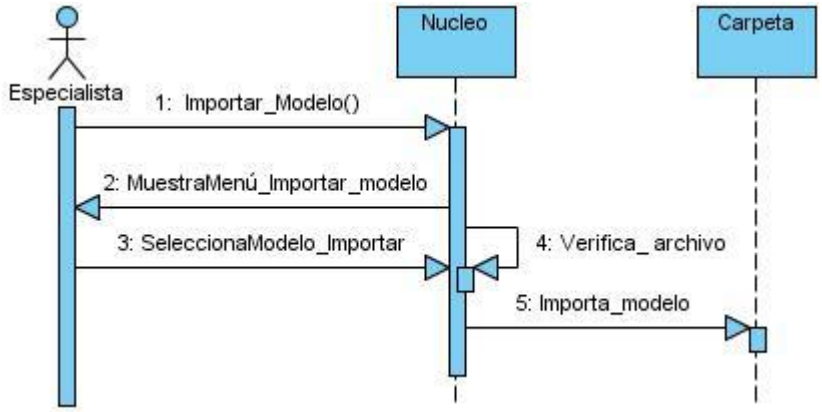

Figura 1: Diagrama de Secuencia Importar modelo.

Diagrama de secuencia del Caso de uso Exportar Modelo.

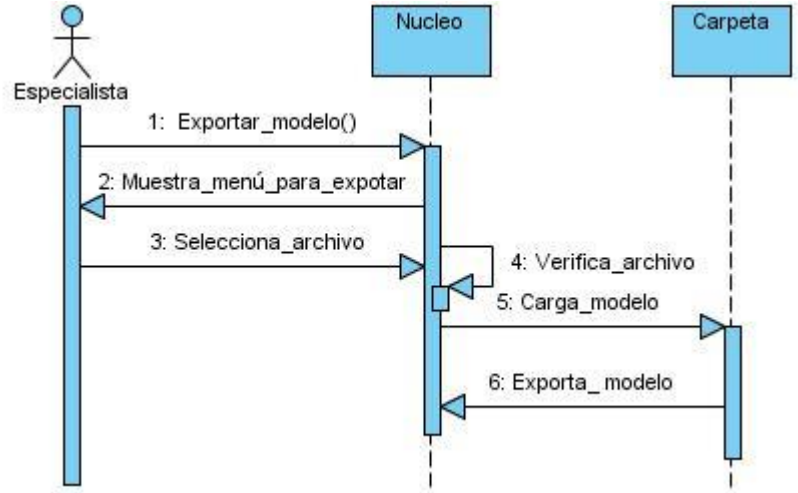

Figura 2: Diagrama de Secuencia Exportar modelo.

Diagrama de secuencia del Caso de uso Guardar Modelo.

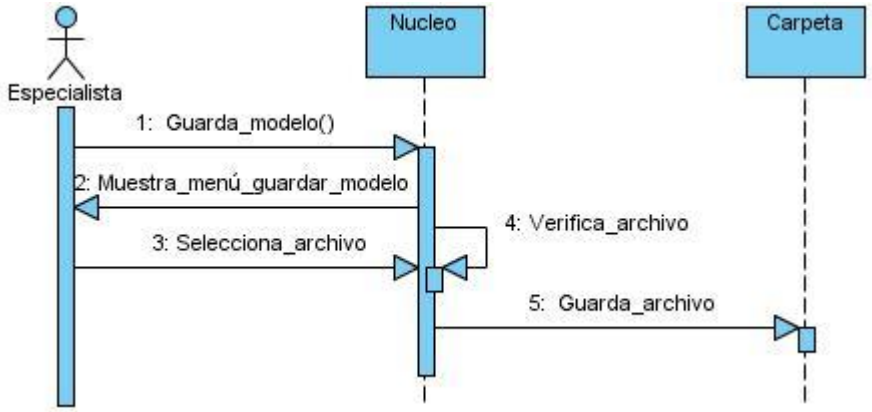

Figura 3: Diagrama de Secuencia Guardar modelo.