Universidad de las Ciencias Informáticas.

Facultad 3

Solución Informática "Nodo Virtual de Procesos" para la carrera Ingeniería Automática.

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autores: Daniel Hernández Garrigó.

Rolando Moreno Martínez.

Tutor: Ing. Yalice Gámez Batista

Co-Tutor: Dr. Valery Moreno Vega

Consultante: MSc. Yoan Martínez Márquez

Ciudad de la Habana, Junio de 2009

"Año 50 de la Revolución"

Lo que nosotros tenemos que practicar hoy, es la solidaridad. No debemos acercarnos al pueblo a decir: "Aquí estamos. Venimos a darte la caridad de nuestra presencia, a enseñarte con nuestra presencia, a enseñarte con nuestra ciencia, a demostrarte tus errores, tu incultura, tu falta de conocimientos elementales". Debemos ir con afán investigativo, y con espíritu humilde, a aprender en la gran fuente de sabiduría que es el pueblo.

Ernesto Ché Guevara

Agosto 1960

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Rolando Moreno Martínez

_____

Firma del Autor

Daniel Hernández Garrigó

_____

Firma del Autor

Yalice Gámez Batista

_____

Firma del Tutor

Agradecimientos.

- *Agradecemos a nuestros familiares que en todo momento nos han brindado su apoyo y preocupación.*
- *A nuestros amigos por tener confianza en nosotros y brindarnos su amistad.*
- *A nuestra tutora y consultante por ayudarnos siempre.*
- *A las personas que de una forma u otra contribuyeron a la realización del Trabajo de Diploma.*

Dedicatoria.

Daniel:

A mi hija por ser lo más importante de mi vida y darme fuerzas para realizar este trabajo y a mi novia por saber cuidar de ella en los momentos que me encontraba dedicado a su realización.

A mis suegros por preocuparse por mí siempre y quererme como a un hijo. A mi cuñado por ser como un hermano para mí.

A mis padres que han estado en todo momento conmigo, confiar en mí, aconsejarme en los momentos difíciles y lograr guiarme por este camino que a llegado a hacerme un profesional.

A mis amigos que han logrado darme alegrías y aconsejarme en los momentos que lo necesité en gran medida, y que me ayudaron en mi preparación como estudiante.

A las personas que de alguna forma contribuyeron a mi formación y colaboraron conmigo cuando les pedí ayuda.

A los profesores Yoan y Yalice por ser personas tan especiales que me brindaron tanto su amistad como sus consejos para mejorar en mi carácter y como futuro profesional, siempre han estado dispuestos a ayudarme y los recordaré siempre.

Rolando:

A mi sobrino por ser ese pequeño duendecito travieso que siempre está preocupado por cuando volveré a estar a su lado, y que se molesta cuando le digo que ya tengo que irme, porque para él no es tiempo suficiente para estar juntos.

A mi madre por estar siempre ahí cuando más la necesito, por ser la única que llora cuando llego de pase después de largos meses de ausencia, por ser mi sustento, mi apoyo y mi guía.

A mis dos padres por estar siempre al tanto de lo que me sucede, por sus consejos oportunos, por ayudarme a ser el hombre que soy hoy en día, por representar para mí un ejemplo a seguir.

A mi hermana por brindarme su cariño incondicional, por preocuparse siempre por su hermano más pequeño, yo.

A Yalice y Yoan por estar siempre a mi lado en los momentos más difíciles de mi carrera, por preocuparse por mí incluso en los malos momentos, por haber sido mi sustento cuando tuve problemas de enfermedad y por haberme dado siempre como es característico de ellos ese ánimo que tanto necesitamos las personas.

RESUMEN.

El presente trabajo se fundamenta en la programación de una aplicación para desarrollar la simulación de procesos industriales. Mediante el uso de modelos matemáticos; que permitirá a los estudiantes de la carrera de Ingeniería Automática probar estrategias de control. Actualmente no se cuenta con una herramienta que ofrezca semejantes prestaciones, que permita la simulación computarizada de forma remota y que brinde el máximo de interacción entre el software y el usuario. Por estas razones se desarrolla una aplicación cliente-servidor. Empleando la técnica de virtualización, y haciendo uso del framework Qt4.5.0. Bajo el entorno de desarrollo QtCreator. El trabajo parte del Análisis y Diseño de la herramienta Nodo Virtual de Procesos elaborado bajo la metodología de desarrollo Rational Unified Process (RUP), utilizando el Visual Paradigm como herramienta de modelado. Se seleccionaron como lenguaje de programación el C++ y como paradigma el Orientado a Objetos. Además se llevó a cabo un análisis del diseño propuesto y de los módulos y componentes reutilizables. Para obtener como resultado una herramienta robusta, se llevaron a cabo una serie de pruebas tanto al código como a las interfaces, que permitieron detectar los fallos de la aplicación y darles solución en posteriores iteraciones.

PALABRAS CLAVE

Simulación de Procesos Industriales, Nodo Virtual de Procesos, Virtualización, DLL.

ÍNDICE.

.

INTRODUCCIÓN

Para nadie es un secreto la relevancia que tienen en la actualidad las tecnologías de informática y las comunicaciones (TIC) en todos los ámbitos de la vida, ya sea en la Salud, la Economía, la Cultura, la Educación. Producto al avance que han tenido las TIC en los últimos tiempos y las nuevas potencialidades que ofrecen, se ha logrado desarrollar un área muy importante de la informática referida a la simulación de procesos.

La simulación de procesos se ha utilizado ampliamente en aspectos prácticos en muchas disciplinas. Con esta se accede a la capacidad de experimentar independientemente del sistema real. La simulación permite obviar los riesgos inherentes a la experimentación, alcanzar una completa independencia temporal y repetir el experimento un número de veces arbitrario.

En la actualidad existen herramientas informáticas (Hosseinzaman, et al., 1995) (Maier, et al., 2003) (Miyachi, et al., 2006) (Molino, et al., 2004) (Wu, 1999) que permiten realizar simulaciones de procesos industriales (en tiempo real o no), pero tienen limitaciones en cuanto a la cantidad de recursos que necesitan para ser implementadas, otras en cuanto al número de procesos que pueden simular simultáneamente. Y para erradicar estas limitaciones se han creado aplicaciones que usan nodos virtuales en los cuales se simulan los procesos.

En el trabajo con los nodos virtuales, los países más desarrollados en la creación de este tipo de software son, Estados Unidos, Inglaterra y Alemania. Por ejemplo: En el laboratorio Nacional de Lawrence Berkeley en EUA, se creó una herramienta para simulación numérica multifase en pozos de petróleo (Wu, 1999); en la Universidad de Stuttgart en Alemania se implementó la herramienta llamada “Network Emulation Testbed” (Maier, et al., 2003), que permite simular redes; y en Inglaterra, Universidad de Nottingham Trent, se creó una herramienta para la simulación de sistemas de agua (Hosseinzaman, et al., 1995).

En Cuba también se han desarrollado sistemas de control industrial (SCI), y un ejemplo de ello es el SCI “EROS” desarrollado en 1991 por ingenieros de la Unión del Níquel en la provincia de Holguín. En la Universidad de las Ciencias Informáticas (UCI) se está llevando a cabo un proyecto SCADA (Supervisory Control And Data Adquisition) en convenio con Venezuela para el control de los procesos que se desarrollan en las industrias, “El Guardián del Alba”. En esta misma Universidad existe además un proyecto llamado SIMPRO dedicado al desarrollo de simuladores. Sin

embargo en ninguno de estos casos se han usado los nodos virtuales para la simulación de procesos.

En el Instituto Superior Politécnico José Antonio Echeverría (CUJAE), en la facultad de Eléctrica, disciplina de Ingeniería Automática, los estudiantes reciben un grupo de asignaturas de peso en las que se utilizan modelos de procesos industriales. Para ello se cuenta con herramientas de simulación tales como el MATLAB, LabVIEW, y para correr aplicaciones en tiempo real un SCADA; pero ninguna de ellas da respuesta a las limitaciones de recursos y son herramientas propietarias.

En la UCI, durante el curso 2007-2008 se realizó el "Análisis y Diseño" (Escobar, et al., 2008), y además la "Arquitectura" (Gonce, 2008) de una herramienta para la simulación de procesos usando nodos virtuales. Pero aún no se ha implementado, por lo que se hace necesaria su ejecución.

Por estas razones no existe una herramienta para la simulación concurrente de procesos industriales, que permita a los estudiantes de la carrera de Automática, interactuar con dichos procesos y de esta manera apropiarse de los conocimientos prácticos que necesitan para enfrentarse a las situaciones prácticas que deberán enfrentar en su vida laboral.

Situación Problémica.

En la actualidad los estudiantes de la carrera de Ingeniería Automática no cuentan con una herramienta que les permita interactuar con diferentes procesos configurados por ellos o por sus profesores con la cuál puedan probar posibles estrategias de control. Provocando así que se dificulte la puesta en práctica de los conocimientos adquiridos en clase. En la UCI se cuenta con el Diseño y la Arquitectura de una herramienta que permite realizar este tipo de tareas pero aún no ha sido implementada.

Problema.

¿Cómo desarrollar una solución informática que fortalezca la base de los conocimientos prácticos en los estudiantes de la carrera Ingeniería Automática partiendo de los artefactos obtenidos durante el análisis, diseño y arquitectura del nodo virtual de procesos (NVP)?

Objetivo General.

Desarrollar, partiendo de los artefactos obtenidos durante el análisis, diseño y arquitectura, la solución informática NVP para la carrera Ingeniería Automática.

Hipótesis:

Si se desarrolla la solución informática NVP para la carrera de Ingeniería Automática entonces se fortalecerá la base de los conocimientos prácticos en los estudiantes de la carrera Ingeniería Automática.

Del objetivo general se derivan los siguientes **objetivos específicos:**

1. Estudiar el estado del arte de los nodos virtuales.
    - Tarea 1: Búsquedas bibliográficas sobre los nodos virtuales en el mundo y en Cuba.
2. Seleccionar el lenguaje de programación y el entorno de desarrollo.
    - Tarea 2: Estudio de los paradigmas de programación de acuerdo a los requerimientos del NVP.
    - Tarea 3: Estudio de los lenguajes de programación que se ajusten a los requerimientos del NVP.
    - Tarea 4: Estudio de los entornos de desarrollo que se ajusten a los requerimientos del NVP.
3. Proporcionar el soporte matemático del NVP.
    - Tarea 4: Selección del asistente matemático según los requerimientos de la aplicación.
    - Tarea 5: Implementación del mecanismo de integración de la aplicación con el asistente matemático.
    - Tarea 6: Selección de los modelos matemáticos a simular.
4. Implementar la solución propuesta.
    - Tarea 8: Estudio de las herramientas seleccionadas.
5. Validación de la solución propuesta.
    - Tarea 9: Desarrollo de los casos de prueba que permitan validar la solución propuesta.

Objeto de estudio: Proceso de desarrollo del software.

Campo de acción: Implementación del sistema nodo virtual de procesos para la carrera ingeniería automática.

Métodos teóricos:

Histórico lógico: Se realiza un análisis de cómo han evolucionado las diferentes herramientas de simulación con el uso de nodos virtuales. Obteniendo una tendencia de cómo se debe comportar en la actualidad.

Sistémico: Se tienen en cuenta todas y cada una de las relaciones que se establecen entre los módulos dentro de la aplicación y las contradicciones que generan los mismos.

Hipotético-deductivo: A partir de la interpretación de la realidad se establecen posibles situaciones o resultados para llegar a conclusiones.

Métodos empíricos:

Consulta a especialistas: Se empleó para comprobar la necesidad y la funcionalidad práctica de la aplicación, Nodo Virtual de Procesos que se propone en la presente investigación.

La observación: Mediante guías de observación se le dará seguimiento al desarrollo de la aplicación.

CAPÍTULO I.

1.1 Introducción.

En este capítulo se realiza un análisis general de las diferentes técnicas de programación, lenguajes de programación, tecnologías, frameworks y tendencias de uso para el desarrollo del nodo virtual de procesos, para lo cual se ha llevado a cabo un estudio del estado del arte de los aspectos antes mencionados.

1.2 Nodo virtual de procesos.

En la actualidad existen muchas herramientas informáticas que permiten realizar simulaciones de procesos industriales (en tiempo real o no), pero todas tienen limitaciones en cuanto al número de recursos que necesitan para su implementación o en cuanto al número de procesos simultáneos que se pueden simular. Incluso, en su mayoría, o simulan los procesos o permiten probar aplicaciones reales, nunca las dos prestaciones. (Gámez, 2008)

Para dar solución a esta problemática se optó por un Nodo Virtual. Un Nodo Virtual es una aplicación que permite ejecutar diferentes instancias del software en un único nodo (nodo físico) y cada instancia del software trabaja en un entorno de ejecución independiente (nodo virtual). (Maier, et al., 2003)

Atendiendo a esto se define como nodo virtual de procesos, a aquel software que permita implementar modelos de distintos procesos ya sea para su simulación o la prueba de aplicaciones en tiempo real, por lo que será necesario que varios procesos estén activos simultáneamente para requerir de la cantidad de nodos y computadoras (Gámez, 2008).

Para ello se establece como nodo a aquella estructura a la cual se interconectan varios elementos. No hay que pensar en un nodo como un elemento constituido solamente por una parte física, sino más bien considerarlo como una unidad funcional en donde tiene que haber tanto hardware como software.

Por otra parte, al ser el punto de conexión de dos o más elementos, el nodo por lo general tiene la capacidad de recibir información, procesarla y enrutarla a uno o varios nodos. De esta forma, un nodo puede ser el punto de conexión para transmitir los datos, el punto desde el cual se distribuyan los datos hacia otros nodos y el punto al que se transmitan los datos (Escobar, et al., 2008).

Esta filosofía de trabajo permite la simulación simultánea de diferentes procesos concurrentes sin que interfieran uno con otros. La virtualización de nodos provee una vía de regular el acceso a recursos de hardware exclusivos de un determinado número de consumidores. En este caso los consumidores son los entornos de ejecución para cada proceso, los cuales están sujetos a las propiedades de la simulación.

De ahí se derivan los siguientes requerimientos para la virtualización del nodo: (Gámez, 2008)

- El parámetro más importante es minimizar los gastos de virtualización para preservar los recursos para el proceso en ejecución.
- Cada entorno de ejecución introducido por la virtualización del nodo debe ser tan transparente como sea posible para los restantes. Esto es importante para que la medición de la implementación no sufra modificaciones en comparación con la real.

1.2.1 NVP en otros sistemas similares.

En la actualidad existen muchas aplicaciones que utilizan los nodos virtuales por las grandes ventajas que brindan en el aprovechamiento máximo de los recursos de las computadoras.

Para la simulación de redes, en la Universidad de Stuttgart en Alemania, instituto especializado en sistemas distribuidos y paralelos, se creó la aplicación “Network Emulation Testbed” (Maier, et al., 2003). Esta aplicación va dirigida a simular un entorno de redes configurable que permita reproducir un escenario real de cientos de nodos en comunicación. De esta manera posibilita medir de manera comparativa el comportamiento de una aplicación en diferentes entornos de redes o de varias aplicaciones en un mismo entorno de red. Esta aplicación es utilizada tanto en redes tradicionales como en un entorno de redes ad hoc inalámbrica. Permite a partir de una red de 64 computadoras traducirlo a un escenario de más de 1920 nodos.

La principal limitante de esta aplicación está en que es diseñada para simular redes por lo que no es posible utilizarla para la simulación de procesos que tienen una dinámica un tanto más compleja y variada.

En la Universidad de Nottingham Trent, se creó un software para la simulación de sistemas de agua (Hosseinzaman, et al., 1995). Surge por la necesidad que tenían en la industria del agua de adquirir y almacenar datos de estaciones remotas para la inspección ingenieril. A medida que fue creciendo el sistema, fue además

incrementándose la acumulación de grandes volúmenes de datos que debían ser procesados por los ingenieros. Para ayudar a reducir la carga de trabajo y aumentar la eficiencia y la efectividad del control del sistema se introdujo el software de simulación. Este software tiene como tareas analizar los parámetros del sistema y arribar a decisiones que pueden ser aceptadas o modificadas por el ingeniero responsable de estas operaciones. La integración del sistema de supervisión con el software de simulación, para lograr un sistema capaz de tomar decisiones en tiempo real, conllevó a un incremento de los requerimientos computacionales para los algoritmos de simulación con un respectivo aumento de tamaño de la red física. Una solución clásica fue dividir la red en subredes para resolver las subredes de manera aislada y de esta manera coordinar las soluciones de los subsistemas para encontrar la solución del sistema completo.

Esta aplicación no es la solución pues no está concebida para simular procesos independientes, sino que simula subprocesos de un sistema específico, para luego integrarlo en el proceso general como un todo.

En el laboratorio Nacional de Lawrence Berkeley en EUA, se enfrentaban al problema de tratar las condiciones límites en los pozos de petróleo en el momento de formular y codificar un simulador numérico multifase de la reserva. Esto se debe a la complejidad de las ecuaciones diferenciales que gobiernan el flujo de la superficie que son una mezcla de tipo hiperbólica-parabólica que provocan problemas de convergencia (Wu, 1999). El método convencional de tratamiento de pozos geotermales o de reservas de petróleo no es lo suficientemente riguroso y puede dar lugar a soluciones físicamente incorrectas para las diferentes capas del pozo. Por esta razón se utilizó el método de nodos virtuales donde a cada capa se le asignó un nodo virtual cuyo tratamiento está dado en dependencia de las características de la capa a la que represente para los cálculos del flujo. La solución en el pozo se obtendrá de resolver las ecuaciones del balance de masa para el nodo del pozo. De esta manera se provee de un procedimiento numérico eficiente y consistente de manera física para el manejo de los problemas del flujo en los pozos.

De la misma manera que el anterior este simulador fue diseñado para el estudio de las diferentes capas de un pozo petrolífero para luego integrarlo en el modelo del pozo.

En la Universidad de Stanford se creó un algoritmo de nodos virtuales para el trabajo con imágenes en tres dimensiones (Bao, et al., 2004). Un elemento es fragmentado para crear varias replicas del elemento y se le asigna una porción real del material a cada réplica. De esto resultan elementos que contienen material real y regiones

vacías. El material faltante está contenido en otra u otras copias. El algoritmo de nodo virtual determina automáticamente el número de réplicas así como la asignación de material para cada una. Provee además los grados de libertad requeridos para simular el material parcial o completamente fragmentado en una imagen consistente con la geometría. Aprovecha las posibilidades de la simulación de una geometría compleja con una simple mezcla.

Este simulador tampoco satisface las necesidades planteadas.

En Japón se creó una herramienta “StarBED” para dar solución al vacío que existe entre internet y los entornos para experimentación, atendiendo a los aspectos de escala, complejidad y realidad. Es una aplicación de pruebas basada en lotes de nodos que tiene como objetivos construir entornos de experimentaciones reales, complejos y de gran escala (Miyachi, et al., 2006). Está diseñado para 512 computadoras en las cuales corren diez computadoras virtuales. Es capaz de soportar una topología de experimentación por encima de los 512 pero resulta difícil manejar y controlar todos esos nodos. Además se implementó “SpringOS” para soportar las simulaciones de acuerdo con la configuración de los usuarios.

Estas herramientas fueron creadas para evitar las influencias de los servicios críticos de internet en las aplicaciones de simulación. Se utilizan para la simulación de diferentes tipologías de redes y no están concebidas para la simulación de procesos. Por esta razón no satisface los requerimientos de la aplicación planteada.

De manera general se puede concluir que en la actualidad los nodos virtuales son potencialmente usados para trabajar en aplicaciones diseñadas para la simulación de redes, para la simulación de procesos que por su complejidad no pueden ser realizadas por las herramientas tradicionales y para el tratamiento de imágenes. Por estas razones es necesario que el Nodo Virtual de Procesos incorpore muchas características que no poseen dichas aplicaciones.

1.3 Paradigmas de programación.

Los paradigmas son marcos de referencia que imponen reglas sobre cómo se deben hacer las cosas, indican qué es válido dentro del paradigma y qué está fuera de sus límites. Un paradigma distinto implica nuevas reglas, elementos, límites y maneras de pensar, o sea implica un cambio. Los paradigmas pueden ser considerados como patrones de pensamiento para la resolución de problemas. Desde luego siempre teniendo en cuenta los lenguajes de programación, según nuestro interés de estudio.

Definición:

Un paradigma de programación es un modelo básico de diseño y desarrollo de programas, que permite producir programas con unas directrices específicas, tales como: estructura modular, fuerte cohesión, alta rentabilidad, etc. (Bonaparte, et al., 2008)

Los paradigmas pueden ser considerados como patrones de pensamiento para la resolución de problemas. Desde luego siempre teniendo en cuenta los lenguajes de programación, según el interés de estudio.

No es mejor uno que otro sino que cada uno tiene ventajas y desventajas. También hay situaciones donde un paradigma resulta más apropiado que otro.

Hay multitud de ellos atendiendo a alguna particularidad metodológica o funcional

Cuando un lenguaje refleja bien un paradigma particular, se dice que soporta el paradigma, y en la práctica un lenguaje que soporta correctamente un paradigma, es difícil distinguirlo del propio paradigma, por lo que se identifica con él.

1.3.1 Tipos de Paradigmas de Programación.

Después de leer bibliografías donde autores clasifican los paradigmas de modos similares, siempre destacan el imperativo, el orientado a objetos, el funcional y el lógico. Algunos autores o profesores, mencionan paradigmas heurísticos, concurrentes, procedimentales, declarativos y demostrativos.

1.3.1.1 Paradigmas Imperativos*:* Modelo abstracto que consiste en un gran almacenamiento de memoria donde la computadora almacena una representación codificada de un cálculo y ejecuta una secuencia de comandos que modifican el contenido de ese almacenamiento. Algoritmos + Estructura de Datos = Programa.

Son aquellos que facilitan los cálculos por medio de cambios de estado, entendiendo como estado la condición de una memoria de almacenamiento. Los lenguajes estructurados en bloques, se refieren a los ámbitos anidados, es decir los bloques pueden estar anidados dentro de otros bloques y contener sus propias variables. La RAM representa una pila con una referencia al bloque que está actualmente activo en la parte superior (Rodríguez, 2008).

1.3.1.2 Paradigmas Procedimentales*:* Modelos de Desarrollo: Orientado a Objetos, a Eventos, y a Agentes. Secuencia computacional realizada etapa a etapa para resolver el problema. Su mayor dificultad reside en determinar si el valor computado es una solución correcta del problema (Rodríguez, 2008).

1.3.1.3 Paradigmas Declarativos*:* Modelos de Desarrollo: Funcional, Lógico y de Flujo de Datos. Se construye señalando hechos, reglas, restricciones, ecuaciones, transformaciones y otras propiedades derivadas del conjunto de valores que configuran la solución (Rodríguez, 2008).

1.3.1.4 Paradigmas Demostrativos*:* Modelos de Desarrollo: Genético. Cuando se programa bajo un paradigma demostrativo (también llamada programación por ejemplos), el programador no especifica procedimentalmente cómo construir una solución sino que presentan soluciones de problemas similares (Bonaparte, et al., 2008).

1.3.1.5 Paradigmas Funcionales*:* Modelo matemático de composición funcional donde el resultado de un cálculo es la entrada del siguiente, y así sucesivamente hasta que una composición produce el valor deseado (Bonaparte, et al., 2008).

Como su nombre lo dice, operan solamente a través de funciones. Cada función devuelve un solo valor, dada una lista de parámetros. No se permiten asignaciones globales, llamados efectos colaterales. La programación funcional proporciona la capacidad para que un programa se modifique así mismo, es decir que pueda aprender.

1.3.1.6 Paradigmas lógicos*:* Esta programación se basada en un subconjunto del cálculo de predicados, incluyendo instrucciones escritas en formas conocidas como cláusulas de Horn. Este paradigma puede deducir nuevos hechos a partir de otros hechos conocidos. Un sistema de cláusulas de Horn permite un método particularmente mecánico de demostración llamado resolución (Bonaparte, et al., 2008).

1.3.1.7 La programación Orientada a Objetos (POO): es una metodología de diseño de software y un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan *estado* (es decir, datos) y *comportamiento* (esto es, procedimientos o *métodos*). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que se comunican entre ellos para realizar tareas. Esto difiere de los lenguajes procedurales tradicionales, en los que los datos y los procedimientos están separados y sin relación. Estos métodos están pensados para hacer los programas y módulos más fáciles de escribir, mantener y reutilizar (Bonaparte, et al., 2008).

La tecnología orientada a objetos se define como una metodología de diseño de software que modela las características de objetos reales o abstractos por medio del

uso de clases y objetos. Hoy en día, la orientación a objetos es fundamental en el desarrollo de software, sin embargo, esta tecnología no es nueva, sus orígenes se remontan a la década de los años sesenta. De hecho Simula, uno de los lenguajes de programación orientados a objetos más antiguos, fue desarrollado en 1967.

Los programadores que emplean lenguajes procedurales, escriben funciones y después les pasan datos. Los programadores que emplean lenguajes orientados a objetos definen objetos con datos y métodos y después envían mensajes a los objetos diciendo que realicen esos métodos en sí mismos (Bonaparte, et al., 2008).

- **Objeto:**

Una unidad de software conformada por atributos y métodos específicos.

El objeto, es el concepto principal sobre el cual se fundamenta la tecnología orientada a objetos. Un objeto puede ser visto como una entidad que posee atributos y efectúa acciones.

En términos de la programación orientada a objetos, se dice que todo objeto tiene un estado (atributos) y un comportamiento (acciones). La programación orientada a objetos nos permite modelar estos objetos del mundo real en objetos de software de forma eficaz. Un objeto de software mantiene sus atributos o estado en variables e implementa las acciones o comportamientos por medio de métodos o funciones (Wesley, 2001).

- **Clases.**

Es un molde o bien prototipo en donde se definen los atributos (variables) y las acciones (métodos) comunes de una entidad.

Este es el paradigma que propone la programación orientada a objetos, la abstracción de los elementos que constituyen a un objeto del mundo físico, esto es, atributos y comportamiento, y la representación de éstos elementos por medio de objetos de software formados por variables y métodos que permitan la manipulación de tales variables. Cabe mencionar que además de representar los objetos físicos del mundo real por medio de objetos, también podemos modelar objetos abstractos, por ejemplo las acciones de un usuario al interactuar con la maquina. Como observaremos más adelante, algunas de las ventajas de la tecnología orientada a objetos son la reutilización de objetos, y la facilidad de comprensión de los programas (Wesley, 2001).

- **Abstracción en programación orientada a objetos.**

Consiste en aislar un elemento de su contexto o del resto de los elementos que lo acompañan. En programación, el término se refiere al énfasis en el "¿qué hace?" más que en el "¿cómo lo hace?". El común denominador en la evolución de los lenguajes de programación, desde los clásicos o imperativos hasta los orientados a objetos ha sido el nivel de abstracción del que cada uno de ellos hace uso (Wesley, 2001).

- **Encapsulación en programación orientada a objetos.**

En programación orientada a objetos, se denomina *encapsulación* al ocultamiento del estado, es decir, de los datos miembro, de un objeto de manera que sólo se puede cambiar mediante las operaciones definidas para ese objeto.

De esta forma el usuario de la clase puede obviar la implementación de los métodos y propiedades para concentrarse sólo en cómo usarlos. Por otro lado se evita que el usuario pueda cambiar su estado de maneras imprevistas e incontroladas (Wesley, 2001).

- **Encapsulamiento.**

El encapsulamiento consiste en poner juntos los datos y las funciones dentro de un objeto de tipo clase.

El modelo procedimental puede representarse como en la figura.

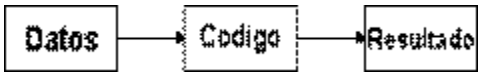


Fig. 1 Ejemplo de encapsulamiento

La encapsulación define los niveles de acceso para elementos de esa clase. Estos niveles de acceso definen los derechos de acceso para los datos, permitiéndonos el acceso a datos a través de un método de esa clase en particular, desde una clase heredada o incluso desde cualquier otra clase. Existen tres niveles de acceso:

- *público:* funciones de toda clase pueden acceder a los datos o métodos de una clase que se define con el nivel de acceso *público*. Este es el nivel de protección de datos más bajo
- *protegido:* el acceso a los datos está restringido a las funciones de clases heredadas, es decir, las funciones miembro de esa clase y todas las subclases
- *privado:* el acceso a los datos está restringido a los métodos de esa clase en particular. Este es nivel más alto de protección de datos

- **Polimorfismo en programación orientada a objetos.**

En programación orientada a objetos se denomina *polimorfismo* a la propiedad que poseen algunas operaciones de tener un comportamiento diferente dependiendo del objeto sobre el que se aplica.

Muchas veces es necesario que un mismo comportamiento o acción se realice de diferentes maneras, por ejemplo, supongamos que deseamos implementar a la clase mamíferos, supongamos también que uno de los métodos que deseamos implementar para esta clase, es el que permita a tales mamíferos desplazarse de forma natural. Nos encontraremos entonces con que para algunos mamíferos el desplazamiento se realizará por medio de caminar, como es en el caso de las personas, para otros el desplazamiento natural será nadar, como en el caso de los delfines e inclusive para otros, el desplazamiento se logrará por medio de volar, como sucede con los murciélagos. En otras palabras, un mismo comportamiento, en este caso el desplazamiento, puede tomar diferentes formas.

Dentro de la programación orientada a objetos puede modelarse esta situación del mundo real, en objetos de software, gracias al polimorfismo. El polimorfismo es la propiedad por la cual una entidad puede tomar diferentes formas. Generalmente está entidad es una clase, y la forma en que se consigue que tome diferentes formas es por medio de nombrar a los métodos de dicha clase con un mismo nombre pero con diferentes implementaciones.

- **Herencia en programación orientada a objetos.**

La herencia es uno de los mecanismos de la programación orientada a objetos, por medio de la cual una clase se deriva de otra de manera que extiende su funcionalidad. Una de sus funciones más importantes es la de proveer polimorfismo y late binding (Wesley, 2001).

- **Tipos de herencia.**

Herencia simple: Un objeto puede extender las características de otro objeto y de ningún otro, es decir, solo puede tener un padre.

Herencia múltiple: Un objeto puede extender las características de uno o más objetos, es decir, puede tener varios padres. En este aspecto hay discrepancias entre los diseñadores de lenguajes. Algunos de ellos han preferido no admitir la herencia múltiple por las posibles coincidencias en nombres de métodos o datos miembros. Por ejemplo C++ admite herencia múltiple, Java y Ada sólo herencia simple.

En la Programación Orientada a Objetos, la herencia es un concepto semejante al de la herencia genética que se da en la naturaleza (Wesley, 2001).

- **Jerarquía de clase.**

La relación primaria-secundaria entre clases puede representarse desde un punto de vista jerárquico, denominado vista de clases en árbol. La vista en árbol comienza con una clase general llamada superclase (a la que algunas veces se hace referencia como clase primaria, clase padre, clase principal, o clase madre; existen muchas metáforas genealógicas). Las clases derivadas (clase secundaria o subclase) se vuelven cada vez más especializadas a medida que van descendiendo el árbol. Por lo tanto, se suele hacer referencia a la relación que une a una clase secundaria con una clase primaria mediante la frase "es una" x o y (Wesley, 2001).

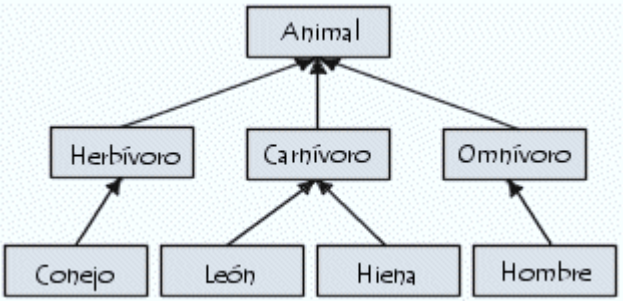


Fig. 2 Ejemplo de jerarquía de clases

- **Herencia múltiple.**

Algunos lenguajes orientados a objetos, como C++ permiten herencias múltiples, lo que significa que una clase puede heredar los atributos de otras dos superclases. Este método puede utilizarse para agrupar atributos y métodos desde varias clases dentro de una sola.

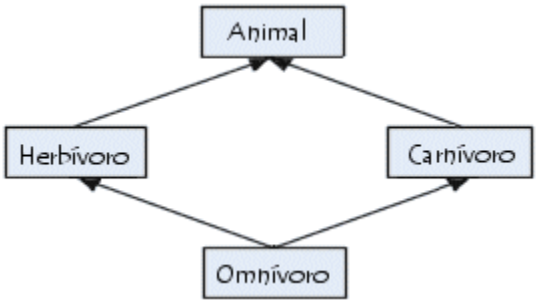


Fig. 3 Ejemplo de herencia múltiple

De acuerdo a los requerimientos del nodo virtual de procesos se selecciona el paradigma orientado a objetos por tener características tales como la herencia, polimorfismo, abstracción y encapsulamiento que brindan ventajas como la mejora del trabajo en equipo, fomenta la reutilización de código, adapta el sistema al mundo real, agiliza el desarrollo del software, permite desarrollar sistemas complejos y facilita el mantenimiento del software.

1.4 Selección del lenguaje de programación y del entorno de desarrollo.

Para el desarrollo de este proyecto se compararon numerosas alternativas de desarrollo. Desde usar un único lenguaje de alta eficiencia como C++ y basarse en librerías intermedias para lograr portabilidad, hasta lenguajes que no dependieran en absoluto de la máquina de ejecución como puede ser Java. Pero sin perder de vista que para el objetivo fundamental de las operaciones también es necesario altas capacidades de cálculo en punto flotante y una eficiente gestión de la memoria (Lucas Rodríguez, Cabello Quintero, 2005).

Dentro de los lenguajes de programación que se consideraron están Java, C++ y C#.

1.4.1 C#.

C# es un lenguaje orientado a objetos elegante y con seguridad de tipos que permite a los desarrolladores crear una amplia gama de aplicaciones sólidas y seguras que se ejecutan en .NET Framework. Puede utilizar este lenguaje para crear aplicaciones cliente para Windows tradicionales, servicios Web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos, y muchas tareas más.

C# es una versión avanzada de C y C++ y se ha diseñado especialmente para el entorno .NET. C# es un nuevo lenguaje orientado a objetos empleado por programadores de todo el mundo para desarrollar aplicaciones que se ejecuten en la plataforma .NET. De todas formas, C# no es parte del entorno .NET. C# es parte de Microsoft Visual Studio. NET 7.0. C# es un paso muy importante en la evolución de los lenguajes de programación, y es una solución ideal para las aplicaciones de alto nivel. Con C# se puede desarrollar todo tipo de proyectos de aplicaciones cliente/servidor (Charte, 2002).

C# amplía las capacidades de C, C++, Visual Basic y Java para proporcionar un completo entorno de desarrollo en el que crear aplicaciones. C# mezcla la potencia de C, las capacidades de orientación a objetos de C++ y la interfaz gráfica de Visual Basic. Además, los programas en C# y en Java se compilan a bytecode.

1.4.2 Lenguaje C++.

Versión de C orientada a objetos creada por Bjarne Stroustrup. C++ se ha popularizado porque combina la programación tradicional en C con programación orientada a objetos. Las principales características del C++ son abstracción (encapsulación), el soporte para programación orientada a objetos (polimorfismo) y el soporte de plantillas o programación genérica (Templates). Desde sus inicios, C++ intentó ser un lenguaje que incluyera completamente al lenguaje C (quizá el 99% del código escrito en C es válido en C++) pero al mismo tiempo incorpora muchas características sofisticadas no incluidas en aquél, tales como: POO, excepciones, sobrecarga de operadores, templates o plantillas. Una visión general del lenguaje se pude obtener en (Wesley, 2001). Diversos tutoriales y textos como (Eckel, 2000), (Sloter, et al., 2005) proporcionan abundante información educativa.

1.4.3 Lenguaje Java.

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principio de los años 90s. Java es un lenguaje moderno, de alto nivel, que recoge los elementos de programación que típicamente se encuentran en todos los lenguajes de programación, permitiendo la realización de programas profesionales. (Allende, 2005)

La sintaxis del lenguaje heredó características de C y C++, explícitamente eliminando aquellas que para muchos programadores (según los diseñadores) resultan excesivamente complejas e inseguras.

Con el auge de Internet, pareció natural aprovechar este lenguaje para desarrollar aplicaciones distribuidas y portables. La primera implementación de Java data de 1995 y pronto los "navegadores web" incorporaron soporte Java para la ejecución de pequeñas aplicaciones interactivas (Applets.) En la actualidad su uso es promovido para el desarrollo de aplicaciones empresariales del lado del servidor, especialmente a través del estándar J2EE, así como en dispositivos móviles (a través del estándar J2ME.)

En realidad, Java hace referencia a un conjunto de tecnologías entre las cuales el lenguaje Java es sólo una de ellas. Por tal motivo muchas veces se habla de la "plataforma Java", la cual es indesligable del lenguaje.

Sun controla los estándares de Java a través de un mecanismo de apertura parcial denominado el Java Community Process (JCP.)

1.4.4 Criterios de comparación.

Comparar lenguajes de programación nunca ha sido una tarea sencilla ni objetiva. Teniendo en cuenta los requerimientos del software se establecieron los siguientes criterios de comparación:

- Portabilidad.
- Capacidades 2D/3D.
- Matemática de precisión compleja.
- Gestión de memoria.
- Velocidad de ejecución.
- Licencia.
- Eficiencia
- Modularidad.

1.4.4.1 Portabilidad.

El lenguaje C++ aunque no es un lenguaje automáticamente distribuido en los sistemas Unix, prácticamente todos lo pueden ejecutar ya sea en una variante comercial o mediante el popular GNU GCC/G++ con lo que la disponibilidad está asegurada. En cuanto a su portabilidad, el único inconveniente notorio radica en ciertos problemas (cada vez menos frecuentes) en las implementaciones de la Standard Template Library (STL).

No obstante lo indicado, el C++ presenta importantes dificultades de portabilidad, particularmente en cuanto a los siguientes aspectos:

1. Características dependientes de la implementación: Lo que permite realizar fuertes optimizaciones en distintas arquitecturas, resulta con frecuencia una pesadilla para la portabilidad. Muchos detalles importantes son dejados a criterio de quien escribe el compilador, tales como los tamaños de diversos tipos de datos, juegos de caracteres, comportamiento ante ciertos errores, etc.

2. Acceso a librerías del sistema operativo: Las interfaces y librerías principales no han seguido un proceso de estandarización tan riguroso como el lenguaje, lo que ha traído como consecuencia diversas soluciones incompatibles para los mismos problemas.

Estrictamente este no es un problema del lenguaje, sino más bien de la plataforma utilizada (por ejemplo, las variantes de Unix.)

En ese sentido Java introdujo un enfoque radical (aunque predecible) al diseñar un lenguaje prácticamente sin características dependientes del implementador (potencialmente algo menos eficiente), y con una extensa librería utilitaria cuya interfaz de programación está muy fuertemente estandarizada. Esto trajo consigo la famosa promesa: “write once, run everywhere” (escribir una sola vez, ejecutar en cualquier lugar) la cual ha sido muchas veces objeto de mofa debido a diversos errores de implementación y especificaciones poco claras.

En el caso de C#, al separar los pasos de compilación a código intermedio de la ejecución de este código intermedio, tiene la posibilidad de crear entornos de ejecución propios de las diferentes arquitecturas. La plataforma ha tomado la orientación de suministrar un entorno de ejecución (run-time) y un entorno de desarrollo (kit de desarrollo).

Con todo, la portabilidad alcanzada es cualitativamente superior a la que se puede obtener con el lenguaje C/C++, y se consigue de manera automática por cualquier desarrollador (Gámez, 2008).

1.4.4.2 Capacidades 2D/3D.

C++ depende en este apartado por completo en bibliotecas externas al propio lenguaje, esto supone si se quiere portabilidad usar capas intermedias como vxWindows para 2D.

Java tiene soporte bastante completo para 2D, pero sus capacidades tridimensionales están limitadas a x86. A pesar de todo existen algunos componentes que permiten generar gráficos 3D sin usar código nativo, pero están bastante limitados.

C# tiene muy buen soporte para 2D, pero muy limitado en el aspecto 3D. Puede tener posibilidades de mejora.

1.4.4.3 Matemática de precisión compleja.

En este punto el mejor lenguaje es FORTRAN con diferencia, de hecho es el único lenguaje con soporte nativo para números complejos. Además permite usar tipos de datos de enorme precisión (hasta 8 bytes).

Le sigue a cierta distancia C++, mucho más estructurado y “moderno” pero menos especializado en este ámbito. Es muy frecuente usar bibliotecas que internamente hacen uso de código FORTRAN.

Java posee una máquina virtual muy limitada en este aspecto y casi todo el peso recae sobre código software. Es muy deficitario en este aspecto. Aunque existen bastantes utilidades y bibliotecas de clases para suplir este aspecto, dejan bastante que desear.

En el caso de C# el nivel de desarrollo alcanzado en este aspecto es pobre, pero la estructura de la plataforma de programación (.NET), al no definir una máquina virtual como hace java, permite radicales optimizaciones en este aspecto.

1.4.4.4 Gestión de memoria.

En este aspecto C++ es lo más eficiente, ya que se tiene control absoluto de la memoria. Se pueden usar todas las técnicas de manejos de punteros y conteo de referencias.

Java y C# tiene recolectores automáticos de memoria que facilitan enormemente la programación. No es necesario reservarla ni liberarla. Pero esta sencillez de utilización no es crítica ya que con un buen diseño y metodología en C++, pueden crearse estructuras complejas con respecto al uso de memoria, pero muy fáciles de usar en el resto del programa (Lucas Rodríguez, Cabello Quintero, 2005). Además, los recolectores automáticos generan una carga de tiempo adicional al procesador, afectando el tiempo de ejecución de la aplicación, un recurso crítico en el NVP.

1.4.4.5 Velocidad de ejecución.

C++ es bastante eficiente, sobre todo si se trata de un compilador capaz de realizar optimizaciones sobre el código.

Java y C# no generan código nativo, sino para una capa intermedia que necesita ser interpretada/compilada. El enfoque de Java es definir una arquitectura hardware virtual. Una máquina con sus instrucciones y niveles de pila.

C# define un lenguaje intermedio semánticamente mucho más rico, pero con igual capacidad teórica de resolver problemas (máquina de Turing). Pero la realidad es que al ser el lenguaje intermedio de C# mucho más expresivo, permite definir instrucciones de más alto nivel. En ambos enfoques durante la ejecución se transforma ese nivel intermedio en código máquina, pero en C#, los resultados experimentales dan mejor

rendimiento (Lucas Rodríguez, Cabello Quintero, 2005). No obstante, el tener que traducir el código en esa capa intermedia, una vez más, agrega una carga de tiempo en la ejecución del código, afectando el tiempo global de ejecución de la aplicación.

1.4.4.6 Licencia.

Las implementaciones de Java pueden adquirirse a varias compañías (en lugar de una única como es .NET). Java tiene un largo camino andado en relación al desarrollo de su arquitectura sobre diferentes plataformas. La tecnología Java es abierta y se basa en gran medida en estándares de organizaciones de normalización y "de facto". (Gámez, 2008)

En el caso del C++ existen variantes tanto propietarias como abiertas.

1.4.4.7 Eficiencia.

Prácticamente todos los computadores ejecutan los programas mediante una o más unidades centrales de procesamiento (CPU) las cuales (dependiendo de la marca y el modelo) sólo comprenden el llamado "lenguaje máquina" o "código máquina", el cual consiste de una serie de operaciones relativamente elementales o de muy "bajo nivel" tales como escribir bytes en memoria, sumar un par de números, leer bytes de un dispositivo externo, etc.

Por lo tanto, todos los lenguajes de programación deben ser "traducidos" en algún momento a "lenguaje máquina" para que los programas sean ejecutados; simplificando, a este proceso se le suele denominar "compilación" y tanto el lenguaje C como el lenguaje C++ siguen este esquema de ser "compilados" al "lenguaje máquina" del procesador en el que se van a utilizar.

Java fue creado desde el inicio para ser ejecutado en cualquier clase de dispositivo o CPU, y uno de sus aspectos más interesantes es que no se compila directamente en el lenguaje máquina del CPU en uso, sino en un "pseudo lenguaje máquina" denominado "bytecode". Este Java compilado en "bytecode" puede ser transportado a cualquier computador en el cual se dispone de un programa especial encargado de la traducción del "bytecode" al verdadero "lenguaje máquina" del CPU en uso.

En otras palabras, este programa especial "interpreta" el "bytecode", efectivamente ejecutando la aplicación Java original. Este programa intérprete se conoce (simplificando un poco) como "Java Virtual Machine" (JVM) o "Java Runtime Enviroment".

Un programa compilado en “bytecode” en tanto debe ser además traducido (interpretado) en lenguaje máquina, en general resulta algo más lento que un programa ya traducido al lenguaje máquina del CPU donde este paso adicional ya no se requiere.

Un segundo inconveniente, particularmente en aplicaciones relativamente pequeñas, radica en los recursos de memoria que típicamente utiliza el Java Virtual Machine; si bien esto suele ser configurable, dichos ajustes no suelen ser sencillos ni bien documentados.

Un tercer inconveniente para ciertas clases de aplicaciones se encuentra en la impredecibilidad del “garbage collector”, el cual en muchas ocasiones no realiza su trabajo en el momento más apropiado y suele consumir mucho tiempo de CPU en su análisis, contribuyendo a la lentitud. Afortunadamente los implementadores de las JVM han optimizado mucho la inteligencia del garbage collector al punto que en la actualidad esto sólo es un problema en casos excepcionales.

Un programa en Java suele ser notoriamente más lento si la tarea principal consiste en operaciones lógico/matemáticas, mientras que la eficiencia suele ser ligeramente inferior a la correspondiente a C/C++ para aplicaciones que hacen uso de muchos otros componentes y librerías auxiliares (AmericaTI EIRL, 2006).

C# en este aspecto es muy similar al Java.

1.4.4.8 Modularidad.

En la referencia original (Tucker, 1992) este criterio estaba referido a la posibilidad de desarrollar componentes de manera independiente los que eventualmente interactuarían.

Es ese sentido, los tres lenguajes analizados permiten desarrollar funciones, clases, y paquetes de modo independiente, cada cual con sus convenciones particulares.

1.4.5 Elección del lenguaje a utilizar.

Para tomar una decisión en la elección del lenguaje, se realizó una tabla resumen (tabla 1.1) donde se le asignaron valores a los diferentes criterios de selección en función de su importancia para el desarrollo de la aplicación. Obteniéndose los siguientes resultados:

Criterio de selección	Peso	C++	C#	Java
Portabilidad	5	3	4	5
Capacidad 2D/3D	5	3	2	2
Matemática de precisión	5	3	2	2
Gestión memoria	5	5	3	3
Velocidad	10	10	8	6
Licencia	10	10	0	10
Eficiencia	10	10	6	6
Modularidad	5	4	4	4
Total		48	29	38

Tabla.1 Comparación entre los lenguajes de programación.

Teniendo en cuenta estos resultados se optó por escoger como lenguaje de programación el C++ para aprovechar su velocidad de ejecución, eficiencia y todas las potencialidades que ofrece de manera general.

1.4.6 Elección del entorno de programación.

Para la elección de entorno de desarrollo integrado (IDE) se tuvo en cuenta que fuera una aplicación sobre software libre por las ventajas que conlleva. Entre estas se destacan:

- Evita la dependencia tecnológica de empresas foráneas.
- Ahorros por pagos de licencias de softwares.
- Posibilidad de revisar el código fuente.

Entre los IDE que existen para Linux los más populares son:

- *KDevelop:* Surgió en 1998 con el fin de desarrollar un IDE fácil de usar para KDE(K Desktop Environment). Desde entonces está públicamente disponible bajo licencia GPL y soporta lenguajes de programación como: C, C++, Java, Ada, SQL, Python, Perl y Pascal. Sólo corre en sistemas Linux y otros sistemas Unix. Su última versión es la 3.5 y salió el 16 de octubre del 2007. Tiene como limitantes principales que su entorno gráfico es muy pobre y sólo corre sobre plataforma Linux.(www.kdevelop.org).
- *NetBeans:* Es una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a su vez un IDE desarrollado usando dicha plataforma. Sun MicroSystems fundó el proyecto de código abierto en junio 2000, aunque fue en el año 1996 que surgieron las primeras ideas para su desarrollo en la República Checa, y continúa siendo el patrocinador principal. Sus limitantes fundamentales radican en su entorno gráfico que no es muy amigable y que tiene menos facilidades de programación que el Eclipse. (www.netbeans.org).
- *Eclipse:* Es un IDE multiplataforma desarrollado por IBM. En la actualidad lo mantiene la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos, capacidades y servicios complementarios. Pese a que esté escrito en su mayor parte en Java (salvo el núcleo), se ejecute sobre máquina virtual de ésta y su uso más popular sea como un IDE para Java, Eclipse es neutral y adaptable a cualquier tipo de lenguaje, por ejemplo C/C++, Cobol, C#, XML, etc. (Furmankiewicz, 2008). Sus mayores ventajas radican en su gran comunidad de desarrollo que lo ubican como el mejor IDE Java. (www.eclipse.org).
- *GTK* (GIMP ToolKit, http://www.gtk.org). Librería realizada en C (orientado a objetos) creada por Peter Mattis,Spencer Kimball y Josh MacDonald. Inicialmente creada para desarrollar el programa de manipulación de imágenes GIMP, ahora usada en múltiples aplicaciones, GNOME por ejemplo.

Características:

Multiplataforma (HPUX, SUN-Solaris, Linux,Windows, BeOs).

Libre, licencia GPL

Tiene como objetivos, Corrección, Rendimiento, Usabilidad.

- *Qt: Framework C++ para Desarrollo Multiplataforma:* La librería Qt es desarrollada por la empresa noruega TrollTech, con licencias tanto libre como comercial, desarrollada en C++ con añadidos. Qt a la fecha en sus versiones 4.X es un toolkit

maduro que cuenta aproximadamente con 500 clases, más de 9000 funciones y 500.000 líneas de código y que le da al programador mucha de la potencia que le brindan lenguajes como C# o Java con la eficiencia de código compilado en C++.

Qt, al igual que otros toolkits no sólo cuentan con clases para la construcción de interfaces de usuario, también incluyen soporte para dibujo en 2D, hilos, red, bases de datos, etc. La diferencia fundamental entre Qt y las otras librerías es que Qt le agrega al C++ estándar los conceptos de *signal* y *slot*, los cuales son similares en funcionalidad al concepto de callback (puntero a una función *X* que es pasado a otra para que ésta última llame a *X* en el momento adecuado).

Una de las características de las interfaces de usuario actuales es la flexibilidad, y si visualizamos cada uno de los elementos de la interfaz de usuario como un objeto lo que tenemos es un montón de objetos intercambiando información, es precisamente allí donde la magia de las Qt tiene lugar. Se define una ***señal/signal*** como un aviso que un objeto puede emitir cuando le ocurre algo (un cambio de estado importante, también denominado evento), un ***slot*** es un método de un objeto que puede ser llamado cuando se genera una señal particular. Las clases que provee Qt poseen signals y slots predefinidos, sin embargo es muy fácil crear propios en nuevos tipos que deriven de la clase QObjetct.

Se seleccionó Qt debido a que simplifica el desarrollo de aplicaciones ricas para equipos desktop para programadores C++. El marco ofrece amplia funcionalidad y herramientas intuitivas para el desarrollo de aplicaciones avanzadas. Con Qt, los desarrolladores pueden crear potentes aplicaciones de subprocesos múltiples que funcionan en todos los sistemas operativos principales desde una única base de código fuente. Las características potentes de Qt, que incluyen funciones para la administración de subprocesos, objetos y datos, como también mecanismos directos para los subprocesos de comunicación interna, facilitan y aceleran la programación paralela.

1.5 Selección del asistente matemático

La selección de un asistente matemático se hace necesaria para el trabajo con sistemas de ecuaciones que serán la base matemática de la simulación de los procesos. El uso de un asistente matemático ahorra el trabajo de tener que implementar los mecanismos matemáticos que proporcionan estas herramientas.

Para la selección se tuvieron en cuenta herramientas como: *Scilab*, *Matlab*, *Maxima*, *Octave y DLLs Con métodos numéricos compilados*.

Scilab: es un software de cálculo científico orientado a la computación numérica. Posee una extraordinaria versatilidad y capacidad para resolver problemas de matemática aplicada, física, ingeniería, procesamiento de señales y otras muchas aplicaciones. Su base la constituye un sofisticado intérprete formado por cientos de rutinas de cálculo matricial, análisis numérico y visualización gráfica. El programa está concebido como un software abierto, el usuario puede ampliarlo añadiendo sus propias primitivas o modificando las existentes. Emplea un entorno de ventanas amigable que recuerda mucho a los paquetes Matlab, Maple, Mathematica, etc. Elaboradas estructuras de datos (matrices polinómicas, racionales y de texto, listas, sistemas lineales multivariable).

- Intérprete sofisticado y lenguaje de programación con una sintaxis similar a Matlab.
- Cientos de funciones matemáticas desarrolladas (pueden ser añadidas fácilmente nuevas primitivas).
- Fantásticos gráficos (2d, 3d, animaciones).
- Estructura abierta (interfaz sencilla con Fortran y C a través de un enlace dinámico).
- Muchas librerías desarrolladas:
- Álgebra lineal (incluyendo matrices dispersas, forma de Kronecker, ordenación Schur).
- Control (Clásico, LQG, H-infinity).
- Paquete para optimización LMI (Inecuaciones con matrices lineales).
- Procesado de señal.
- Simulación (varios ode's, dassl).
- Optimización (diferenciable y no-diferenciable, soluciona LQ).
- Scicos, un entorno interactivo para modelización y simulación de sistemas dinámicos.
- Metanet (Análisis y optimización de redes).
- Capacidades simbólicas a través de un interfaz Maple.
- Scilab paralelo.

Scilab es una herramienta con muchas potencialidades en el campo científico que ofrece una gran cantidad de funcionalidades por lo que se hace un poco más complejo su uso, además se hace necesario un mayor uso de recursos de hardware para su

ejecución. Según la filosofía del nodo virtual, lo que se busca es una mayor eficiencia y rapidez en la ejecución de los procesos.

Matlab: es un lenguaje de computación técnica de alto nivel y un entorno interactivo para desarrollo de algoritmos, visualización de datos, análisis de datos y cálculo numérico. Con Matlab, puede resolver problemas de cálculo técnico más rápidamente que lenguajes de programación tradicionales, tales como C, C++ y FORTRAN. Se puede usar Matlab en una amplia gama de aplicaciones que incluyen procesamiento de señales e imágenes, comunicaciones, diseño de sistemas de control, sistemas de prueba y medición, modelado y análisis financiero y biología computacional. Los conjuntos de herramientas complementarios las colecciones de funciones de MATLAB para propósitos especiales, que están disponibles por separado amplían el entorno de Matlab permitiendo resolver problemas especiales en estas áreas de aplicación.

Características del Matlab:

- Lenguaje de alto nivel para cálculo técnico
- Entorno de desarrollo para la gestión de código, archivos y datos
- Herramientas interactivas para exploración, diseño y resolución de problemas iterativos
- Funciones matemáticas para álgebra lineal, estadística, análisis de Fourier, filtraje, optimización e integración numérica
- Funciones gráficas bidimensionales y tridimensionales para visualización de datos
- Herramientas para crear interfaces gráficas de usuario personalizadas
- Funciones para integrar los algoritmos basados en MATLAB con aplicaciones y lenguajes externos, tales como C/C++, FORTRAN, Java, COM y Microsoft Excel.

Matlab es una herramienta muy potente que brinda muchas funcionalidades y aplicaciones en el cálculo matemático lo que la hace una de las herramientas más usadas en el mundo. Se decidió no usar esta herramienta por ser una aplicación de software propietario y puede ofrecer limitaciones en su uso.

Maxima: es un sistema para la manipulación de expresiones simbólicas y numéricas, incluyendo diferenciación, integración, expansión en series de Taylor, transformadas de Laplace, ecuaciones diferenciales ordinarias, sistemas de ecuaciones lineales, y

vectores, matrices y tensores. Maxima produce resultados con alta precisión usando fracciones exactas y representaciones con aritmética de coma flotante arbitraria. Adicionalmente puede graficar funciones y datos en dos y tres dimensiones.

El código fuente de Maxima puede compilarse sobre varios sistemas incluyendo Windows, Linux y MacOS X. El código fuente para todos los sistemas y los binarios precompilados para Windows y Linux

Maxima, entre muchas otras funciones, permite:

- Calcular Límites
- Calcular Derivadas
- Calcular Integrales
- Resolver Ecuaciones Diferenciales
- Resolver Transformadas de Laplace
- Resolver Sumatorias
- Resolver Sistemas de Ecuaciones Lineales
- Desarrollar Series de Taylor
- Realizar Cálculo Matricial
- Dibujar Curvas y Superficies

Maxima es una herramienta muy usada en el cálculo simbólico pero no es muy potente en el cálculo numérico, para el desarrollo del sistema el cálculo numérico es muy importante por lo que, no se seleccionó esta por sus posibles limitantes en el procesamiento numérico.

Octave: GNU Octave es un lenguaje de alto nivel, inicialmente pensado para la computación numérica. Octave proporciona una interfaz de línea de comandos para resolver problemas lineales y no lineales de manera numérica, y desarrollar otros experimentos numéricos utilizando para ello un lenguaje que en su mayoría es compatible con Matlab. También se puede utilizar como un lenguaje de lotes (batch-oriented language). Octave tiene una gran cantidad de herramientas para resolver problemas de álgebra numérica comunes, encontrar las soluciones de ecuaciones no lineales, realizar integrales de funciones ordinarias, manipular polinomios, e integrar ecuaciones diferenciales ordinarias y ecuaciones diferenciales algebraicas. Es fácil de extender y modificar a través de funciones definidas por el usuario escritas en el propio lenguaje de Octave, o utilizando módulos cargados dinámicamente escritos en otros lenguajes como C, C++, Fortran, etc.

Este software puede ser libremente redistribuible bajo los términos de GNU General Public License GPL publicado por la fundación de software libre, Octave se puede definir como un lenguaje de alto nivel inspirado en el software comercial MATLAB.

DLLs Con métodos numéricos compilados: En el instituto superior politécnico José Antonio Echeverría se lleva a cabo una investigación (tesis) en la cual se implementan un conjunto de DLLs o librerías que implementan los métodos numéricos más usados para la simulación de proceso en dicha institución. Para su implementación se usa el lenguaje C++ lo cual permite una posibilidad mayor de integración con la aplicación (López, et al., 2009).

Para proporcionar el soporte matemático del sistema se tuvo en cuenta que el uso de asistentes matemáticos como el Octave, Maxima, Matlab y Scilab pueden ser usados para la resolución de los modelos matemáticos, pero al ser herramientas tan robustas en el cálculo numérico cuentan con un conjunto bastante significativo de funcionalidades que no se usan en la simulación de procesos industriales. También se tuvo en cuenta que la integración con estas herramientas puede ser compleja de implementar. Con el uso de las DLLs, en cambio se puede asegurar una mejor integración con la aplicación y en estas se tienen implementados los métodos numéricos que necesita la aplicación sin exceso de funcionalidades innecesarias. Por lo antes planteado se decidió usar las *DLLs con métodos numéricos compilados* para brindar el soporte matemático de la aplicación.

1.6 Selección de los modelos matemáticos a simular.

El comportamiento dinámico de un sistema físico puede ser representado por una ecuación o un sistema de ecuaciones, esto es el modelo matemático de un proceso. Un grupo de ecuaciones matemáticas (o modelo) al resolverse, arrojan resultados que son el equivalente a los parámetros físicos que se incluyeron en el modelado. La base para la simulación por computadoras de procesos, es la solución numérica de la ecuación o las ecuaciones que conforman su modelo. Esto se denomina proceso de simulación.

Para llegar a las ecuaciones que describen un proceso, se requiere un conocimiento profundo de las características del proceso y una base de herramientas para escribir en lenguaje matemático, una aproximación lo más exacta posible del comportamiento físico de un sistema, un proceso e incluso una planta conformada por varios. Alternativamente pueden ser construidos modelos de sistemas cuyas características físicas son desconocidas mediante experimentación, midiendo la respuesta del

proceso ante diferentes estímulos en la entrada. El proceso descrito en el párrafo anterior se denomina proceso de modelado.

Por las características del NVP, con el uso de las DLL con métodos numéricos compilados, se optó por agregarles los modelos matemáticos a simular como DLL. Esto permite que la aplicación pueda simular cualquier modelo matemático expresado en ecuaciones diferenciales, ganando en flexibilidad. Por lo que no se hace necesario la selección de dichos modelos.

1.7 Estándares de codificación.

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente.

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. Además, si se aplica de forma continuada un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas, y, posteriormente, se efectúan revisiones del código de rutinas, caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener.

Según Bjarne StrousTrup, creador del C++ (Sutter, et al., 2004), no existe un estándar de codificación para todos los usos y todos los usuarios. Para una aplicación determinada, usar un mal estándar de codificación es peor que no usar estándar de codificación. Por tal razón se debe elegir las reglas de codificación cuidadosamente y con un conocimiento solido del área donde se desarrolla la aplicación.

La adopción de un estándar de codificación sólo es viable si se sigue desde el principio hasta el final del proyecto de software. No es práctico, ni prudente, imponer un estándar de codificación una vez iniciado el trabajo.

Para el desarrollo de la aplicación NVP se tomó en cuenta el estándar de codificación propuesto Sutter y Alexandrescu en su libro “C++ Coding Standards“el cual sirvió de guía para mantener una adecuada codificación.

1.8 Conclusiones.

En este capítulo se realizó un estudio de diferentes elementos necesarios para la realización de la investigación como son: Los paradigmas de programación, los lenguajes de programación, los frameworks y los asistentes matemáticos. Para ello se tuvieron en cuenta algunas características dentro de las cuales destacan por su importancia, la portabilidad, velocidad, licencia libre, eficiencia, modularidad entre otros. Además se utilizó un estándar de programación para hacer el código más legible y reutilizable.

Capítulo II.

2.1 Introducción:

En este capítulo se realiza un análisis del diseño de clases propuesto para el sistema Nodo Virtual de Procesos. Además una descripción de las clases seleccionadas y se hace mención de los componentes reutilizables utilizados en el sistema, realizando una descripción de métodos novedosos usados como la programación paralela para la virtualización, con la finalidad de plantear la solución propuesta para la implementación del sistema.

2.2 Valoración crítica del diseño propuesto.

El diseño de clases propuesto por el Análisis y Diseño del Nodo Virtual, está compuesto por diversos subsistemas los cuales se muestran en la siguiente figura:

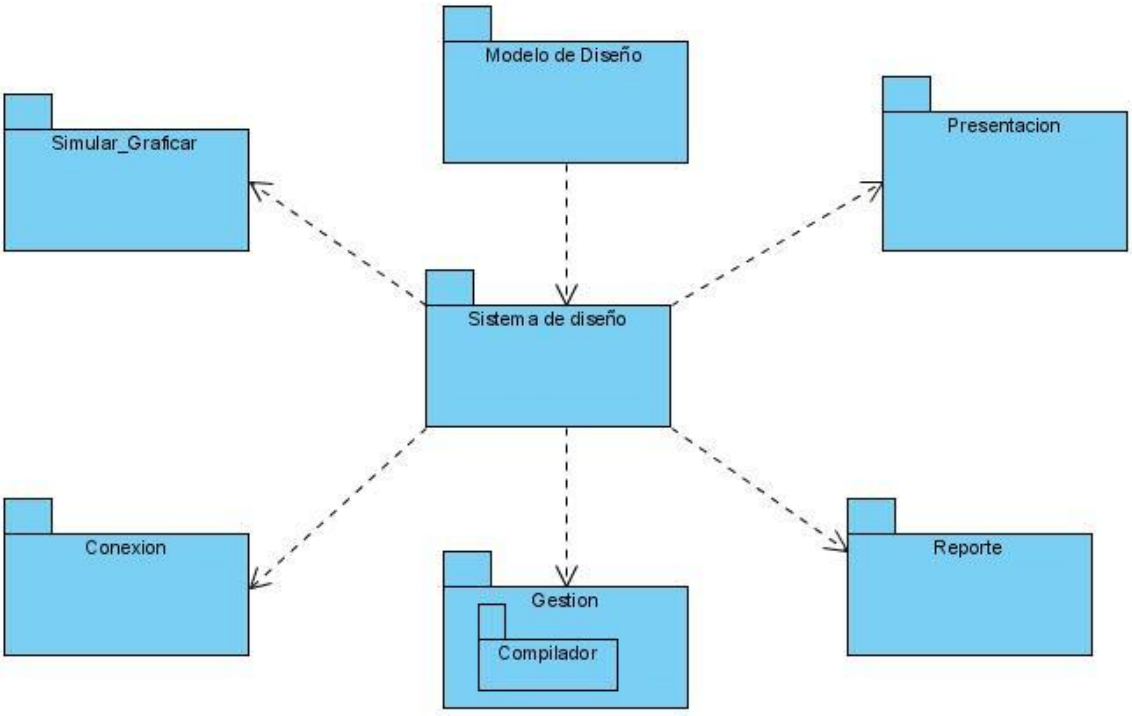


Fig. 4 Anterior Modelo de Diseño del Sistema.

Entre los cambios que se llevaron a cabo se pueden destacar:

Fusión de los subsistemas Gestión, Conexión y Reporte para lograr una mejor comprensión y simplicidad del sistema por parte de los desarrolladores.

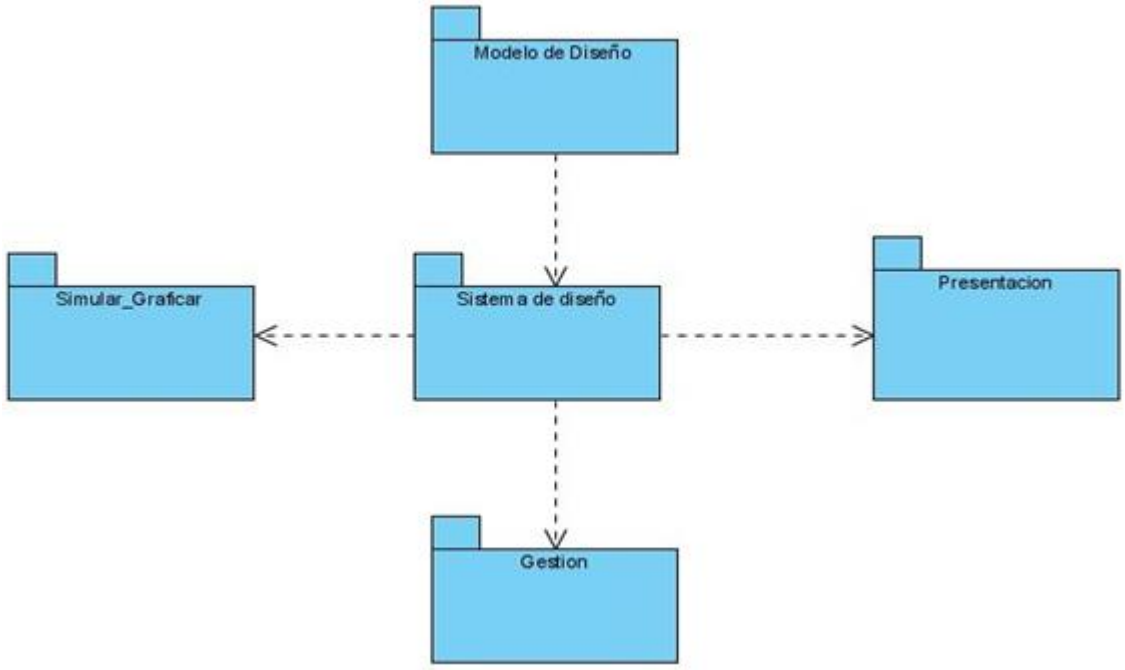


Fig. 5 Actual Modelo de Diseño del Sistema.

En el Subsistema Gestión se descartó la posibilidad de usar un compilador para en su lugar utilizar librerías compartidas (DLLs), las cuales permiten resolver los modelos matemáticos. Además se eliminó la clase entidad CE_Administrador debido a que el administrador tiene los mismos atributos que el usuario, y no cumple objetivo tener herencia en este caso. Se agruparon las funcionalidades correspondientes a la gestión de la conexión al sistema, la gestión de los reportes y la gestión de los procesos en la clase CC_Gestionar_Nodo como se muestra en la figura 6. El Subsistema Presentación y el Subsistema Simular Graficar no sufrieron cambios debido a que presentan un diseño coherente. Cabe destacar que el diseño realizado en realidad no era incorrecto y presentaba de cierta forma más robustez al tener mayor cantidad de subsistemas, lo que lo hacía más complejo para el entendimiento de los desarrolladores por lo que se optó por el diseño aquí propuesto.

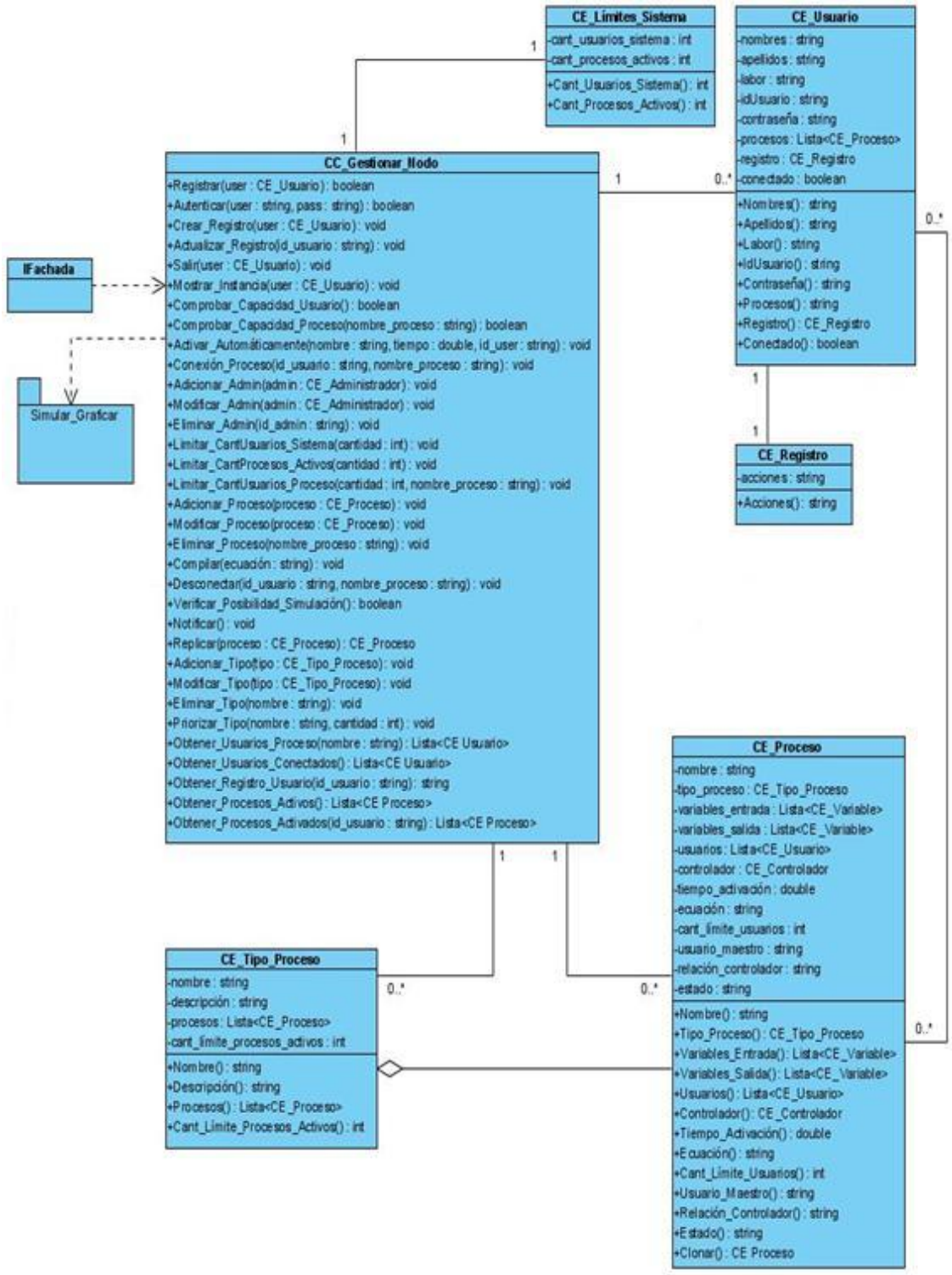


Fig. 6 Subsistema Gestión.

Subsistema Gestión: Contiene las clases que intervienen en el proceso de autenticación y registro de los usuarios al sistema y las interfaces relacionadas con sus privilegios. Además permite la conexión a las diferentes partes de la aplicación y a los procesos. En este subsistema se gestiona toda la información necesaria para el

correcto funcionamiento del software. Permite obtener reportes de la información referente a los procesos y al sistema. También permite a los administradores llevar un mejor control del funcionamiento del sistema.

CC_Gestionar_Nodo: Esta clase es la encargada de gestionar todas las funcionalidades del sistema basándose en las reglas del negocio. Entre los métodos más significativos se destacan: Registro y autenticación en el sistema, conectarse a un proceso, desconectarse de un proceso.

CE_Limites_Sistema: Clase entidad que almacena el límite del sistema.

CE_Usuario: Clase entidad que permite almacenar un usuario.

CE_Registro: Clase entidad que posibilita registrar las acciones de un usuario.

CE_Tipo_Proceso: Clase entidad usada para crear un tipo de proceso.

CE_Proceso: Clase entidad que permite crear un proceso.

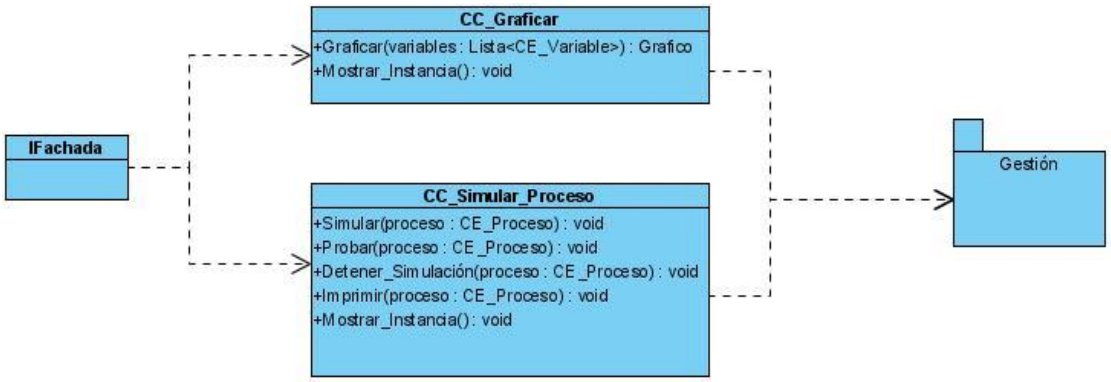


Fig.7 Subsistema Simular-Graficar.

Subsistema Simular-Graficar: En este subsistema se realizan las operaciones de simulación de los procesos. Posibilita la interacción del usuario con el proceso simulado, permitiéndole monitorizar su realización.

CC_Simular_Proceso: Contiene los métodos relacionados con la simulación de un proceso.

CC_Graficar: Posee los métodos y funciones necesarias para graficar la simulación de un proceso.

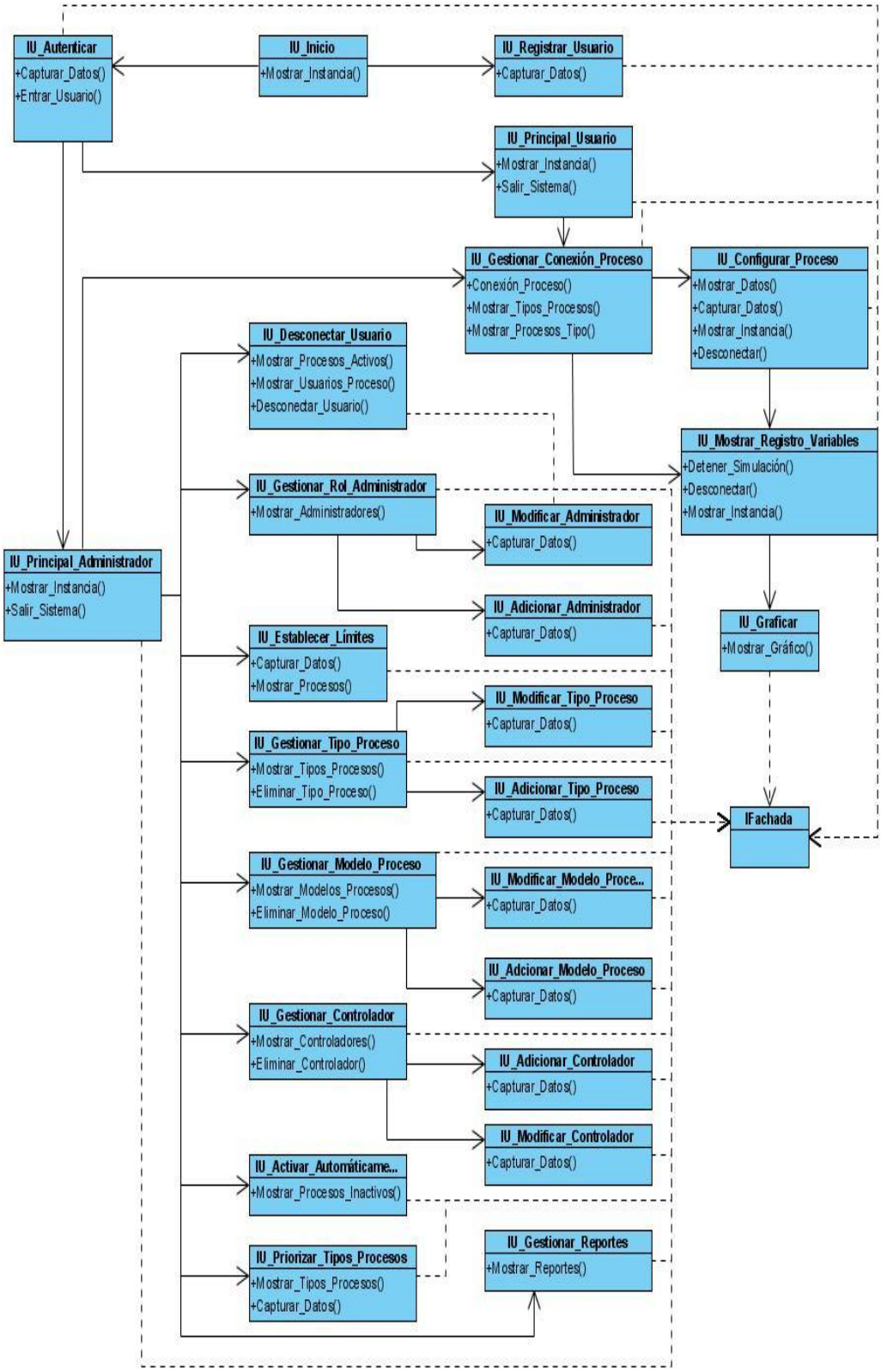


Fig.7 Subsistema Presentación.

Subsistema Presentación: Este subsistema contiene las Interfaces de Usuario que se generan durante la aplicación.

El Subsistema de Presentación está formado por varias clases interfaces como se muestra en la figura.

IU_Inicio: Por medio de esta interfaz el usuario puede registrarse en el sistema y en caso de que se ya encuentre registrado permite autenticarse para poder tener acceso al mismo.

IU_Registrar: En esta enterfaz se toman los datos del usuario para registrarlo en el sistema y se le da acceso a este.

IU_Autenticar: En esta interfaz se toman el usuario y contraseña de un usuario ya registrado en el sistema y se le da acceso al sistema en caso de estar registrado.

IU_Principal_Usuario: En esta interfaz se muestran las opciones que permite realizar el sistema como son, ver los procesos por tipo, conectarse a un proceso, simularlo, configurar proceso y salir del sistema.

IU_Gestionar_Conexion_Proceso: Esta interfaz le muestra al usuario los tipos de procesos que hay, la lista de procesos de ese tipo que se encuentran en el sistema y le permite al usuario conectarse a un proceso que seleccione de la lista de procesos.

IU_Configurar_Proceso: Esta interfaz le muestra al usuario los datos de un proceso seleccionado y le permite cambiar los datos del proceso y desconectarse del sistema.

IU_Mostrar_Registro_Variables: Esta interfaz le permite al usuario detener la simulación, desconectar del sistema, detener la simulación, mostrar los datos del proceso y mostrar una instancia de la graficación.

IU_Graficar: Esta interfaz se encarga de graficar el resultado de la simulación.

IU_Fachada: Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.

IU_Principal_Administrador: En esta interfaz se muestran las operaciones que puede realizar un administrador como son, conectarse a un proceso, desconectar un usuario del sistema, establecer los límites del sistema, administrar los tipos de procesos, administrar modelos, administrar controladores, activar un proceso automáticamente, priorizar un tipo de proceso determinado, ver los reportes y salir del sistema.

IU_Desconectar_Usuario: Esta interfaz le permite a un administrador ver una lista de procesos y los usuarios conectados al proceso permitiéndole desconectar un usuario de un proceso.

IU_Gestionar_Rol_Administrador: Permite a un administrador adicionar o eliminar un usuario.

IU_Adicionar_Administrador: Se capturan los datos del administrador para ser adicionado.

IU_Modificar_Administrador: Se capturan los datos del administrador para ser modificado.

IU_Estableces_Limites: Muestra los límites del sistema y permite al administrador cambiar los valores límites.

IU_Gestionar_Tipo_Proceso: Permite al administrador adicionar un nuevo tipo de proceso, eliminar un tipo de proceso y modificar un tipo de proceso.

IU_Modificar_Tipo_Proceso: Captura los datos del tipo de proceso para ser adicionado.

IU_Adicionar_Tipo_Proceso: En esta interfaz se capturan los datos del tipo de proceso para ser adicionado.

IU_Gestionar_Modelo_Proceso: Esta interfaz permite al administrador adicionar, eliminar y modificar un modelo de proceso.

IU_Modificar_ Modelo _Proceso: Obtiene los datos del modelo de proceso para ser modificado.

IU_Adicionar_ Modelo _Proceso: Captura los datos del modelo de proceso para ser adicionado.

IU_Activar_Automaticaminte: Permite al administrador ver la lista de los tipos de procesos y los procesos inactivos de un tipo seleccionado, y permite además establecer un tiempo para activar un proceso determinado.

IU_Priorizar_Tipo_Proceso: Permite al administrador ver los tipos de procesos que hay en el sistema y darle prioridad a los procesos de un tipo.

IU_Reportes: Permite al administrador ver los reportes generados en el sistema.

Para el mayor entendimiento del sistema para su futura implementacion se realizó el diagrama de componetes que agrupa los componentes necesarios para su correcto funcionamiento.

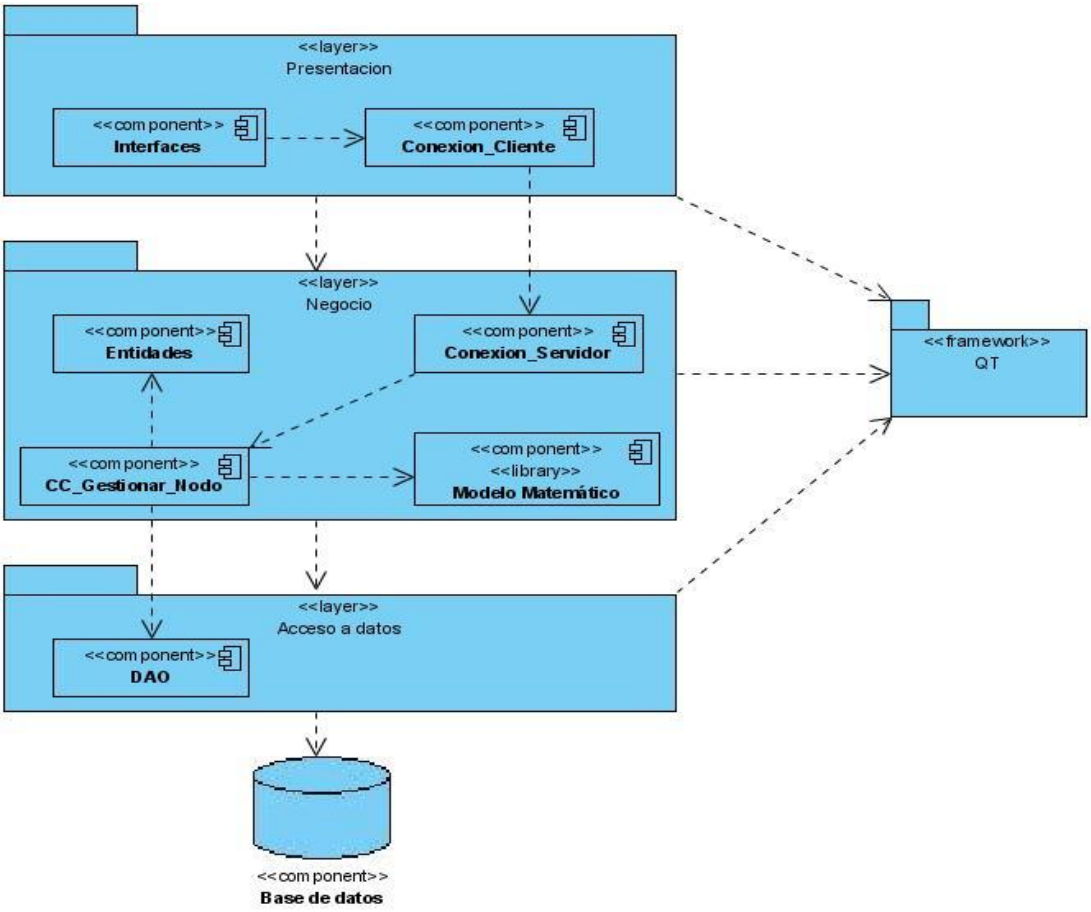


Fig.8 Diagrama de componentes.

Para poder especificar la plataforma sobre la que se ejecuta el software del sistema, y para poder manejar la frontera entre el hardware y el software cuando se trata de la relación entre hardware y software; se hizo necesario realizar el diagrama de despliegue el cual permite razonar sobre la topología de procesadores y dispositivos sobre los que se ejecuta el software.

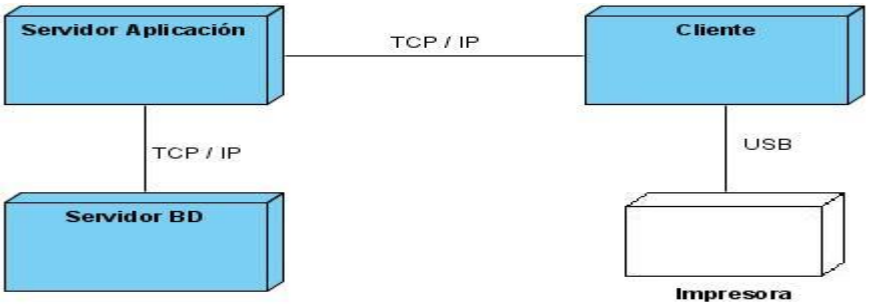


Fig.9 Diagrama de despliegue.

2.3 Análisis de posibles implementaciones, componentes o módulos ya existentes.

2.3.1 Clases y métodos usados para el trabajo multihilos:

Para la simulación concurrente de procesos se optó por la alternativa de la programación multihilos, en la cual cada proceso se desarrolla dentro de un hilo o entorno de ejecución independiente. Esta estrategia también es conocida como programación paralela y en la actualidad se hace cada vez más útil, con el uso de las computadoras multiprocesadoras para la virtualización.

Para el trabajo con hilos se incluyó la clase QThread brindada por QT para el trabajo con hilos.

Se creó una clase llamada Hilo la que hereda de QThread, permitiendo obtener funcionalidades como start() para iniciar un hilo, terminate() para terminar la ejecución de un hilo, isRunning() para saber si se encuentra ejecutándose.

Dentro de la clase Hilo se creó un atributo Proceso que va a guardar un objeto proceso, asegurando de esta forma que el proceso se simule dentro del hilo de ejecución. Además lleva un atributo volátil bool detenido para poder detener y correr el hilo, se declara volátil para poder ser accedida desde diferentes hilos.

```
#include "CE_Proceso.h"
#include <QThread>
class Hilo : public QThread
{
  Q_OBJECT
  public:
    Hilo();
    void detener(); //funcion que detiene la ejecucion de un hilo
    void run(); //funcion que llama a la ejecucion de un hilo
    Proceso* get_proceso();
 private:
   CE_Proceso* proceso;
```

```
    volatile bool detenido;
};
```

Se redefine la función run() para realizar dentro de ella las operaciones a realizar dentro del hilo, y la función detener(), para detener el hilo.

```
void Hilo::run()
{
    while (!detenido)
   {
        CE_Proceso* p = new CE_Proceso();
        //se realizan las operaciones del proceso y se emite la salida
        sleep(3); // se usa para dar un tiempo extra al cpu de procesamiento
    }
}
void Hilo::detener()
{
     detenido = true;
}
```

Para iniciar la ejecución de un hilo desde la clase controladora se procede a crear el hilo como atributo de esta. Dentro de una función de la controladora se llama la función start() heredada de QThread. Para terminar su ejecución se llama a la función terminate(), heredada también de QThread.

```
class Controladora
{
Q_OBJECT
public:
    Controladora();
   ~Controladora();
    Hilo* gethiloA();
```

```
    Hilo* gethiloB();
private slots:
     void iniciar_HiloA();
     void terminar_HiloA();
private:
    Hilo* hiloA;
    Hilo* hiloB;
}
Controladora::Controladora(){
    hiloA = new Hilo();
    hiloB = new Hilo();
}
void Controladora::iniciar_HiloA(){
     hiloA->start();
}
void Controladora:: terminar_HiloA(){
     hiloA->terminate();
}
```

2.3.2 Clases y métodos usados para la graficación:

La clase graficar hereda de la clase QMainWindow de QtCreator, para mostrar una ventana con la gráfica de los valores de la simulación. La filosofía para los gráficos en 2D en Qt difiere de la de otros ambientes de desarrollo. Todo gráfico es creado en un espacio no visible conocido como escena, donde están todos los objetos gráficos según se vayan incluyendo.

Ejemplo:

```
typedef struct {
qreal a;
qreal b;
}range;


class CC_Graficar: public QMainWindow
{
```

```
Q_OBJECT
public:
CC_Graficar();
private:
QGraphicsView *grView;
QAction *closeAct;
QAction *saveAct;
//QAction *zoomInAct;
//QAction *zoomOutAct;
QMenu *fileMenu;
QGraphicsScene *scene;
int step;
int count;
float *data;
range vertical_range;
range horizontal_range;
void DrawScene();
void AddData(double);
void SetStep(double);
private slots:

void save();
void close();
};
```

Esto se implementa en una clase llamada “QGraphicsScene“ que provee la superficie para manejar una gran cantidad objetos gráficos. Los objetos que se incluyen son o heredan del tipo “QGraphicsItem”. Estas están estrechamente ligadas con una clase que provee un “widget” (en inglés: Windows gadget) para mostrar los objetos de una escena: “QGraphicsView” (Vista Gráfica de Qt).

Ejemplo:

```
QGraphicsScene escena;
 escena.addText("Hola, mundo!");

 QGraphicsView vista(&escena);
```

```
vista.show();
```

En el ejemplo se añade un texto a la escena que es mostrado después en la vista gráfica. El método “Dibujar_Escena()” se encarga de insertar en la escena los datos para el gráfico y de mostrarlos en la vista posteriormente. También dibuja los ejes, las cuadrículas y los números más significativos de la escena. Los datos son cargados por otro método como se muestra en la definición de la clase.

En la figura se muestra una vista de los gráficos. Además la interface permite salvar la imagen para un archivo.

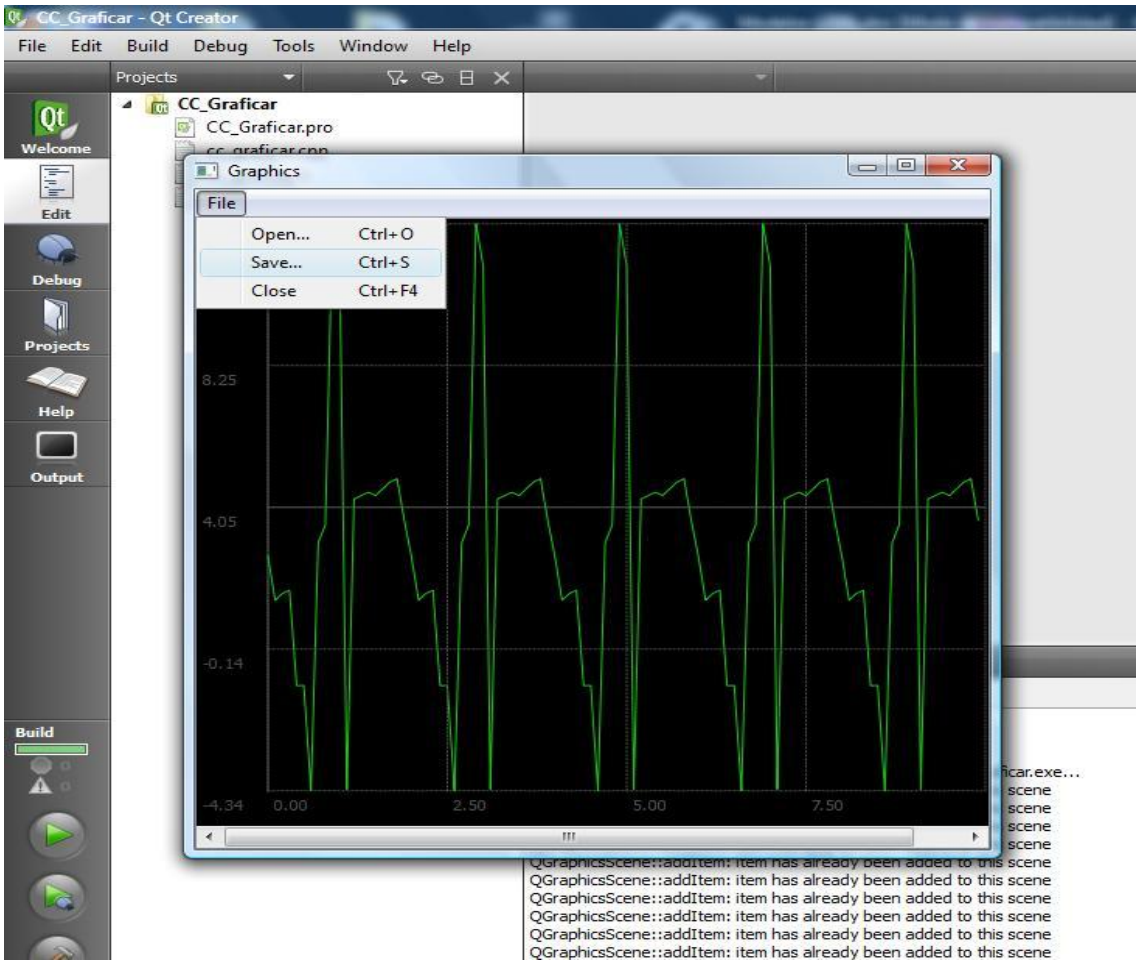


Fig. 9 Vista de los gráficos

2.3.3 Clases y métodos usados para el trabajo con DLLs:

Para el manejo de las librerías se incluyó una clase con métodos especializados en estas funciones, que usan los métodos que incluye Qt en su clase QLibrary. Para cargar la librería se utiliza:

QLibrary::QLibrary (const QString & *fileName*, QObject * *parent* = 0)

Que crea un objeto librería y carga la librería de nombre “fileName” para un “parent”(padre) dado. La ventaja consiste en que la librería se carga por el nombre, sin importar la extensión, haciendo así portable el código para diversas plataformas. En la tabla se muestran las plataformas y los sufijos válidos.

Plataforma	Sufijos Válidos
Windows	.dll
Unix/Linux	.so
AIX	.a
HP-UX	.sl, .so (HP-UXi)
Mac OS X	.dylib, .bundle, .so

Para obtener las funciones se utiliza:

void * QLibrary::resolve (const char * *symbol*)

Que resuelve una función de nombre symbol que se encuentre en la librería cargada y devuelve un puntero a void que puede ser formateado con la forma preestablecida de las funciones, por ejemplo:
typedef double (*FUNC) (double, double*, double*);

Obteniendo con esta las funciones, por ejemplo:

```
FUNC funcion = (FUNC) library->resolve("NombreFuncion");
if (funcion)
    return f1 (5.3, arreglo1, arreglo2);
else
    return -1;
```

La clase controladora de librerías que incluye ambos métodos quedó de la siguiente manera:

```
typedef double (*FUNC) (double, double*, double*);          //Prototipo de función
typedef int (*CANTEQ)(void);                                 //Prototipo de función
typedef int (*CANTPAR)(void);                                //Prototipo de función
typedef char * (*DESC)(void);                                //Prototipo de función
```

```
class CC_Libreria_Dinamica
{
  private:
      FUNC *funciones;                          //Apuntador a las funciones cargadas
      int contador_funciones;
      int cant_paramteros;                       //Cantidad de parámetros
      char nombre_Dll[];
  public:
     CC_Libreria_Dinamica();
    bool CargarDll();                        //carga el modelo matemático
    int Obtener_Cantidad_Parametros();       //devuelve la cantidad de parametros
    int Obtener_Cantidad_Ecuaciones();       //devuelve la cantidad de funciones
    FUNC *Obtener_Funciones();
    bool Obtener_Descripcion(char *Nombre_Dll, char *_descripcion);
    void Set_Nombre_Dll(char *);
};
```

De esta manera son cargadas las librerías que contienen las ecuaciones diferenciales que describen los modelos matemáticos de los procesos para ser resueltas por los diferentes métodos numéricos.

Ejemplo del uso de las DLLs para la implementación de los modelos matemáticos:

Sea el problema de Cauchy de orden 1:

$f'(x) = f(x, y);$

Que consiste en hallar la función f = f(x) que satisface la ecuación diferencial ordinaria anterior en el punto f(x0) = f0.

Y para la ecuación diferencial de orden n:

$F^{(n)} = f(x, y, x', y', \ldots, y^{(n-1)});$

Donde se requiere hallar la solución que satisface las condiciones iniciales:

$f(x0) = f0$

$f'(x0) = f0'$

.

$f^{(n-1)}(x0) = y0^{(n-1)}$

Que se puede reducir a un sistema de ecuaciones de primer orden introduciendo nuevas variables, adecuadas para la solución numérica.

Se hizo necesario entonces encontrar la manera de escribir dicho sistema y hacerlo de una manera que los métodos numéricos pudieran invocar, o sea estar las ecuaciones previamente compiladas, pero que al mismo tiempo se pudieran incluir nuevos modelos sin tener que recompilar toda la aplicación.

Para esto se optó por incluir las ecuaciones o modelos en librerías dinámicas que pudieran ser encontradas y cargadas por el programa para llamarlas. Porque los métodos numéricos de solución (en este caso Runge Kutta) requieren la invocación de las ecuaciones en reiteradas ocasiones en el proceso de solución, o sea, tienen que evaluar la función en un punto determinado. Las funciones se definen en la librería de manera que puedan ser como una función de C para los ambientes Windows.

Un ejemplo:

```
extern "C" __declspec(dllexport) int avg(int a, int b)
{
    return (a + b) / 2;
}
```

El nodo virtual de procesos está formado por una aplicación cliente a través de la cual un usuario puede simular procesos que se encuentran ejecutándose en una aplicación servidor que se encuentra en una personal computer (PC) remota.

2.3.4 Comunicación entre estas dos aplicaciones mediante el protocolo TCP.

Se utilizaron las clases QTcpSocket y QTcpServer brindadas por Qt.

La clase QTcpSocket es utilizada por la aplicación cliente para establecer una conexión TCP y enviar datos a través de esta. Esta clase cuenta con funciones como connectToHost(…) para la conexión al servidor, write(…) para enviarle información, y señales como readyRead() para recibir los datos de respuesta del servidor.

El servidor usa la Clase QTcpServer, la cual permite aceptar conexiones TCP entrantes al servidor y usa funciones como listen(…) para esperar la conexión de un cliente y la señal newConnection() para servir una nueva conexión del cliente. El servidor también hace uso de la clase QTcpSocket.

2.4 Conclusiones.

Para el buen desempeño de los programadores es necesario llevar a cabo un análisis detallado del diseño propuesto. Por ello, en este capítulo se realizó esta tarea conjuntamente con el análisis de posibles complementos reutilizables y métodos novedosos usados. Como el trabajo con hilos, librerías y sockets con sus principales funcionalidades. Esto permitió el uso de la técnica de virtualización, la implementación de los modelos matemáticos y la comunicación cliente servidor.

CAPÍTULO III

3.1 Introducción.

Las pruebas de software son un elemento crítico para la garantía de la calidad del software, y representa una revisión final de las especificaciones, del diseño y de la codificación. Una vez generado el código fuente, el software debe ser probado para descubrir (y corregir) el máximo de errores posibles antes de su entrega al cliente.

Para ello se diseñaron una serie de casos de prueba con una alta probabilidad de encontrar errores, y se hace necesario aplicar técnicas de pruebas de software. Estas técnicas facilitan una guía sistemática para diseñar pruebas desde dos perspectivas:

1-) Comprobar la lógica interna de los componentes de software, que se realiza utilizando técnicas de diseño de casos de prueba de “caja blanca”.

2-) Verificar los dominios de entrada y salida del programa para descubrir errores en la funcionalidad, el comportamiento y rendimiento, utilizando técnicas de diseño de casos de prueba de “caja negra”. En ambos casos, se intenta encontrar el mayor número de errores con la menor cantidad de esfuerzo y tiempo.

Al diseñar los casos de prueba Se determinan los resultados esperados y se guardan los resultados realmente obtenidos (Pressman, 2002).

3.2 Pruebas estructurales o de caja blanca.

Las pruebas de caja blanca realizan un seguimiento del código fuente según se van ejecutando los casos de prueba, de manera que se determinan de manera concreta las instrucciones, bloques, etc. en los que existen errores.

Cuando se pasan casos de prueba al programa que se está probando, es conveniente conocer qué porcentaje del programa se ha ejecutado, de manera que se esté próximo a asegurar que todo él es correcto (evidentemente, es imposible alcanzar una certeza del 100%).

Pruebas de Unidad: Consisten en probar todas las secciones de un proyecto independientemente, para asegurarse que funcionan de acuerdo a las especificaciones. Cuando se reúnen todas las partes probadas, se asegurará que todas las secciones trabajan de la manera correcta, haciendo la tarea de prueba más fácil.

El concepto de pruebas de unidad ha recibido especial atención recientemente porque es una parte fundamental del concepto de desarrollo de software ágil. Las pruebas de unidad permiten cambiar funciones en la implementación, si las pruebas son pasadas, el código seguirá funcionando para el resto de la aplicación (Thelin, 2007).

3.3 Módulo de prueba para la realización de pruebas de unidad.

Qt trae un módulo de prueba ligero, el módulo QtTest que es también conocido como framework QTestLib, proporcionado por Nokia, para realizar pruebas de unidad en aplicaciones basadas en las librerías Qt además de extensiones para probar interfaces de usuario. Entre sus características fundamentales se encuentran:

Ligero: Consiste en aproximadamente 6000 líneas de código y 60 símbolos exportados.

Auto contenido: Requiere pocos símbolos de la librería Qt Core para pruebas que no son de interfaz.

Probado rápido: No necesita corredores de prueba especiales, ni estar registrado para poder realizar las pruebas.

Prueba repetida de datos: Una prueba puede ser ejecutada múltiples veces con diferentes datos de prueba.

Probado básico de interfaz: QTestLib ofrece funcionalidad para la simulación de ratón y teclado.

Fácilmente extensible: Tipos predefinidos pueden ser fácilmente adicionados a la información del test y a la salida del test. (Nokia, 2009).

En las primeras pruebas realizadas se detectaron errores en algunas de las funcionalidades principales y de mayor uso en la aplicación y se obtuvo el resultado que muestra en la figura 10.

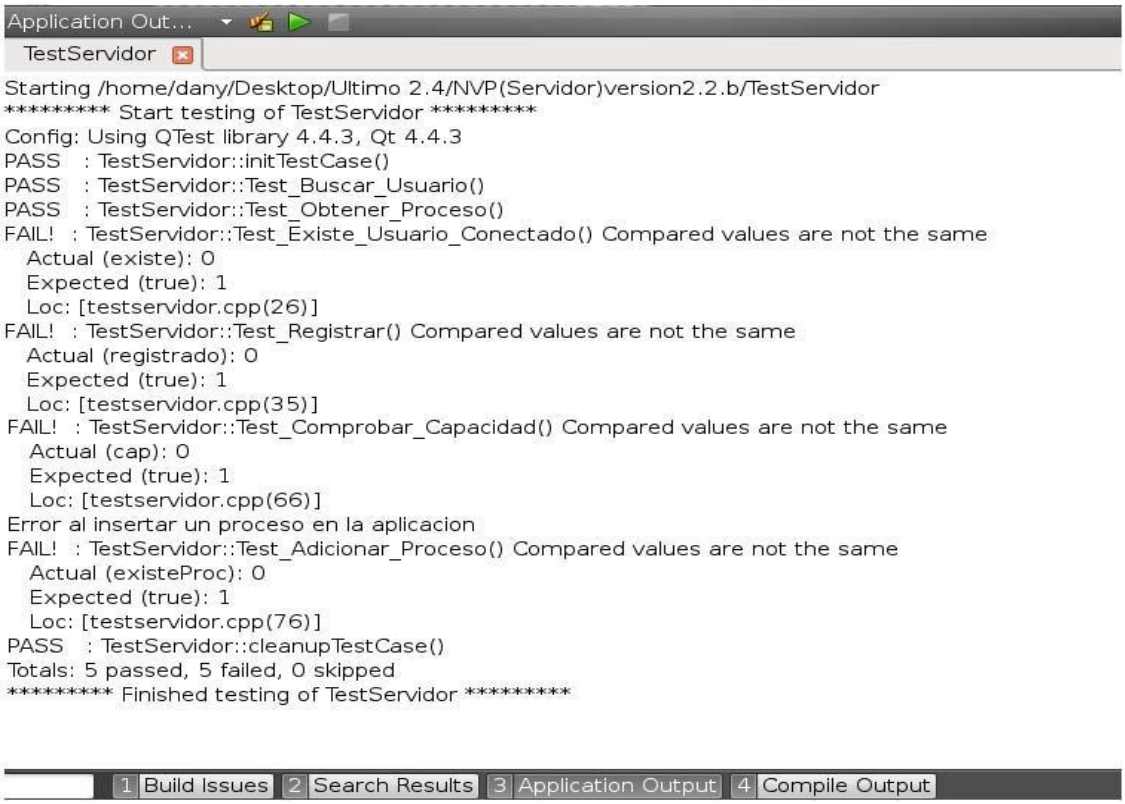


Fig.10 Pruebas primera iteración CC_Gestionar_Nodo.

Para dar solución a los métodos insatisfactorios detectados se realizó el análisis y corrección de las funcionalidades en las que se detectaron errores, hasta lograr la factibilidad de todos los métodos, como se muestra en la figura 11.

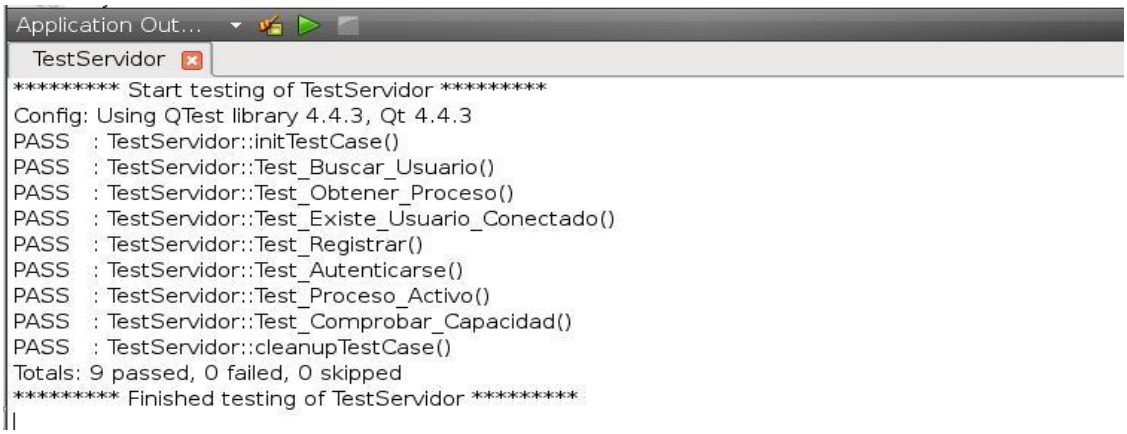


Fig.11 Posteriores Iteraciones CC_Gestionar_Nodo.

Para una mayor entendimiento de los resultados obtenidos durante el desarrollo de las pruebas de unidad aplicadas al software se utiliza una gráfica (ver figura 12) donde se muestran los resultados obtenidos en porciento.

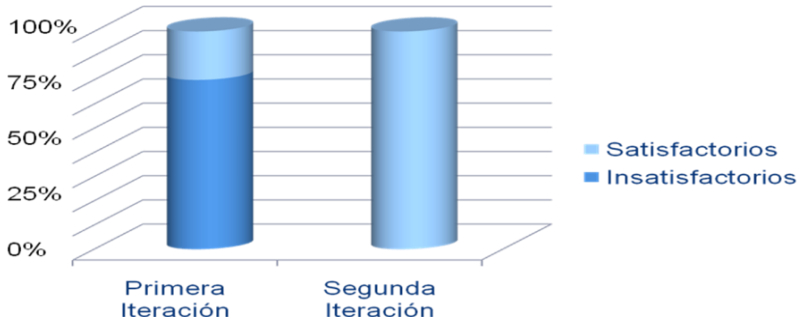


Fig 13. Resultado de las pruebas de unidad por iteraciones.

3.4 Pruebas de Caja Negra.

Son pruebas funcionales. Se parte de los requisitos funcionales, a muy alto nivel, para diseñar pruebas que se aplican sobre el sistema sin necesidad de conocer como está construido por dentro. Las pruebas se aplican sobre el sistema empleando un determinado conjunto de datos de entrada y observando las salidas que se producen, para determinar si la función está siendo desempeñada correctamente por el sistema bajo prueba. Las herramientas básicas son observar la funcionalidad y contrastar con la especificación.

Ejemplos típicos de pruebas de caja negra son la comprobación de valores límites, pruebas de integridad de la base de datos, pruebas de situaciones de excepción, o pruebas de rendimiento del sistema. Presentan una limitación en cuanto a que es prácticamente imposible reproducir todo el espectro por la innumerable cantidad de combinaciones de entrada posibles, agravada por el desconocimiento de la lógica interna (Pressman, 2002).

Permiten obtener conjuntos de condiciones de entrada que ejecuten todos los requisitos funcionales de un programa. Las pruebas de caja negra no son una alternativa a las técnicas de prueba de caja blanca. Es un enfoque complementario.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo Central**
[SC1: Registrarse]	EC1.1 Usuario no registrado	En este escenario se prueba que el usuario pueda registrarse en el sistema	1. El usuario selecciona la opción registrarse 2. El sistema muestra un formulario para que el usuario se registre. 3. El usuario introduce los datos 4. El sistema verifica los datos 5. El sistema crea un usuario con los datos introducidos
	EC 1.2: Usuario ya registrado	En este escenario se prueba que el sistema muestre un mensaje indicándole al usuario que ya se encuentra en la base de datos	1. El usuario selecciona la opción registrarse 2. El sistema muestra un formulario para que el usuario se registre. 3. El usuario introduce los datos 4. El sistema verifica los datos 5. El sistema muestra un mensaje indicando que el usuario ya estaba registrado

[SC2:Autenticarse]	EC 2.1: Sistema con capacidad.	En este escenario se prueba que el sistema tenga capacidad para poder autenticar un usuario.	1. El usuario selecciona la opción autenticarse. 2. El sistema verifica que exista disponibilidad de conexión. 3. El sistema muestra el formulario para la autenticación del usuario.
	EC 2.2: Sistema sin capacidad.	En este escenario se prueba que el sistema muestre un mensaje indicándole al usuario que no hay capacidad en el sistema.	1. El usuario selecciona la opción autenticarse. 2. El sistema verifica que exista disponibilidad de conexión. 3. El sistema muestra un mensaje indicando que no hay de capacidad de conexión.
	EC 2.3: Usuario registrado.	En este escenario se prueba que un usuario que se encuentra registrado se puede conectar al sistema cuando existe capacidad de conexión.	1. El usuario selecciona la opción autenticarse. 2. El sistema verifica que exista disponibilidad de conexión. 3. El sistema muestra el formulario para la autenticación del usuario. 4. El usuario introduce los datos requeridos. 5. El sistema da acceso al usuario.

	EC 2.4: Usuario no registrado	En este escenario se verifica que el sistema le impide el acceso al sistema a un usuario que no se encuentre registrado.	1.El usuario selecciona la opción autenticarse 2.El sistema verifica que exista disponibilidad de conexión 3. El sistema muestra el formulario para la autenticación del usuario. 4.El usuario introduce los datos requeridos 5. El sistema muestra un mensaje de error indicando que debe registrarse para poder acceder.
	EC 2.5: Usuario registrado rol usuario	En este escenario se verifica que a un usuario que se encuentra registrado con rol de usuario tenga acceso al módulo de usuario luego de introducir los datos correctos.	1. El usuario selecciona la opción autenticarse. 2. El sistema verifica que exista disponibilidad de conexión. 3. El sistema muestra el formulario para la autenticación del usuario. 4. El usuario introduce los datos requeridos. 5. El sistema da acceso al módulo de usuario.
	EC 2.6: Usuario registrado rol administrador	En este escenario se verifica que un usuario que se encuentra registrado con rol de administrador tenga acceso al módulo de usuario luego de introducir los datos correctos.	1. El usuario selecciona la opción autenticarse. 2. El sistema verifica que exista disponibilidad de conexión. 3. El sistema muestra el formulario para la autenticación del usuario. 4. El usuario introduce los datos requeridos.

			5. El sistema da acceso al módulo de administrador.
[SC3: Conexión a Proceso]	EC 3.1: Selección del proceso a conectarse	En este escenario se comprueba que un usuario pueda seleccionar un proceso para luego conectarse	1. El usuario selecciona la opción conectarse a proceso. 2. El sistema muestra una interfaz con los tipos de procesos existentes. 3.El sistema muestra los procesos del tipo seleccionado 4.El usuario selecciona un proceso
	EC 3.2: Conexión a proceso con capacidad	En este escenario se prueba que el sistema permita conectarse a un proceso que tiene disponibilidad de conexión	1. El usuario selecciona la opción conectarse a proceso. 2. El sistema muestra una interfaz con los tipos de procesos existentes. 3. El sistema muestra los procesos del tipo seleccionado 4. El usuario selecciona un proceso y acepta conectase al proceso seleccionado 5. El sistema conecta al usuario al proceso seleccionado.

EC 3.3: Conexión a proceso sin capacidad	En este escenario se prueba que el sistema muestre un mensaje al usuario informando que no hay capacidad de conexión al proceso	1. El usuario selecciona la opción conectarse a proceso. 2. El sistema muestra una interfaz con los tipos de procesos existentes. 3.El sistema muestra los procesos del tipo seleccionado 4.El usuario selecciona un proceso y acepta conectarse al proceso seleccionado 5. El sistema muestra un mensaje informando que no hay disponibilidad de conexión al proceso.
EC 3.4: Conexión a proceso activo con capacidad como usuario común	En este escenario se verifica que el sistema conecta al usuario como usuario común al proceso activo seleccionado	1. El usuario selecciona la opción conectarse a proceso. 2. El sistema muestra una interfaz con los tipos de procesos existentes. 3.El sistema muestra los procesos del tipo seleccionado 4.El usuario selecciona un proceso y acepta conectarse al proceso seleccionado 5. El sistema conecta al usuario como usuario común del proceso.

	EC 3.5: Conexión a proceso inactivo con capacidad como usuario maestro	En este escenario se verifica que el sistema muestra un mensaje de elección de ser usuario común o maestro del proceso.	1. El usuario selecciona la opción conectarse a proceso. 2. El sistema muestra una interfaz con los tipos de procesos existentes. 3. El sistema muestra los procesos del tipo seleccionado 4. El usuario selecciona un proceso y acepta conectarse al proceso seleccionado 5. El sistema muestra un mensaje de elegir tipo de usuario del proceso
	EC3.6: Conexión a proceso inactivo como usuario común	En este escenario se prueba que un usuario que elige conectarse como usuario maestro de un proceso el sistema lo envía a la interfaz correspondiente al CUS Configurar _Proceso.	1. El usuario selecciona la opción conectarse a proceso. 2. El sistema muestra una interfaz con los tipos de procesos existentes. 3.El sistema muestra los procesos del tipo seleccionado 4.El usuario selecciona un proceso y acepta conectarse al proceso seleccionado 5.El sistema muestra un mensaje de elegir tipo de usuario del proceso 6. El usuario elige ser usuario maestro del proceso.
SC4: Desconexión de Proceso]	EC4.1: Solicitud de desconexión	En este escenario se prueba que cuando el usuario solicita desconectarse de un proceso el sistema le muestra un mensaje de confirmación.	1. El Usuario solicita la desconexión del proceso. 1.1. El sistema muestra un mensaje de advertencia para confirmar la acción a realizar.

	EC4.2: Solicitud de desconexión aceptada	En este escenario se prueba que cuando el usuario confirma la desconexión de un proceso el sistema lo desconecta del proceso	1. El Usuario solicita la desconexión del proceso. 1.1 El sistema muestra un mensaje de advertencia para confirmar la acción a realizar. 1.2 El usuario confirma la desconexión 1.3 El sistema desconecta al usuario del proceso

Id del escenario	**Escenario**	**Variable 1**	**Variable 2**	**Variable N**	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1.1	Selección del proceso a conectarse	I	V	V	El sistema crea un usuario con los datos introducidos	El programa funcionó de la manera correcta
		V	I	V		
		V	V	I		
EC 1.2	Usuario ya registrado	I	V	V	El sistema muestra un mensaje	El programa pasó la prueba satisfactoria
		V	I	V		

		V	V	I		mente
EC 2.1	Sistema con capacidad.	I	V	V	El sistema muestra el formulario para la autenticación del usuario.	Se detectó una no conformidad
		V	I	V		
		V	V	I		
EC 2.2	Sistema sin capacidad.	I	V	V	El sistema muestra un mensaje indicando que no hay capacidad de conexión.	Se obtuvo el resultado esperado
		V	I	V		
		V	V	I		
EC 2.3	Usuario registrado.	I	V	V	5. El sistema da acceso al usuario.	El programa pasó la prueba
		V	I	V		
		V	V	I		
EC 2.4	Usuario no registrado	I	V	V	El sistema da acceso al usuario.	Se obtuvo una no conformidad
		V	I	V		
		V	V	I		
EC 2.5	Usuario registrado rol usuario	I	V	V	El sistema da acceso al módulo de usuario.	Se obtuvo el resultado adecuado
		V	I	V		
		V	V	I		
EC 2.6	Usuario registrado rol administrador	I	V	V	El sistema da acceso al módulo de usuario.	Se obtuvo una no conformidad
		V	I	V		
		V	V	I		
EC 3.1	Selección	I	V	V	El usuario	Resultados

	del proceso a conectarse	V	I	V	selecciona un proceso	satisfactorios
		V	V	I		
EC 3.2	Conexión a proceso con capacidad	I	V	V	El sistema conecta al usuario al proceso seleccionado.	Se obtuvo una no conformidad
		V	I	V		
		V	V	I		
EC 3.3:	Conexión a proceso sin capacidad	I	V	V	El sistema muestra un mensaje informando que no hay disponibilidad de conexión al proceso.	Se obtuvo el resultado esperado
		V	I	V		
		V	V	I		
EC 3.4	Conexión a proceso activo con capacidad	I	V	V	El sistema conecta al usuario como usuario común del proceso.	El resultado fue satisfactorio
		V	I	V		
		V	V	I		

EC 3.5:	Conexión a proceso inactivo con capacidad como usuario común	I	V	V	El sistema muestra un mensaje de elegir tipo de usuario del proceso	Se obtuvo una no conformidad
		V	I	V		
		V	V	I		
EC3.6:	Conexión a proceso inactivo con capacidad	I	V	V	El usuario elige ser usuario maestro del proceso.	Se obtuvo una no conformidad
		V	I	V		
		V	V	I		
EC4.1:	Solicitud de desconexión	I	V	V	El sistema muestra un mensaje de advertencia para confirmar la acción a realizar.	Se obtuvo el resultado esperado
		V	I	V		
		V	V	I		
EC4.2:	Solicitud de desconexión aceptada	I	V	V	El sistema desconecta al usuario del proceso	Se obtuvo el resultado esperado
		V	I	V		
		V	V	I		

Registro de defectos y dificultades detectados

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativ	Recomendacid	Estado NC	Respuesta del Equipo Desarrollo
Autenticarse	1	El sistema no comprueba en la ventana de inicio si hay capacidad de conexión	Elegir autenticarse, Aceptar	Pruebas de validación		X	La comprobación de capacidad se puede hacer luego de introducirse los datos de autenticación como se hace usualmente en muchas aplicaciones	[PD] 21-5-09	Se valorará con el cliente si es realmente necesario que se realice como aparece en las especificaciones
Autenticarse	2	El sistema autentica a un usuario que no está registrado	Elegir autenticarse, Aceptar Entrar datos Aceptar	Pruebas de validación	X		Debe arreglarse para una mayor seguridad de la aplicación los usuarios deben estar registrados	[PD] 21-5-09	Los desarrolladores se mostraron receptivos
Autenticarse	3	El sistema no muestra a un usuario administrador la interfaz correspondiente	Elegir autenticarse, Aceptar Entrar datos Aceptar	Pruebas de validación	X		Debe cambiarse para cumplir con los requisitos del nodo virtual	[PD] 21-5-09	Los desarrolladores se mostraron receptivos
Conexión a Proceso	4	El sistema da un error si no se selecciona el proceso a conectarse	Seleccionar tipo Opción conectarse a proceso	Pruebas de validación	X		Validar para que muestre un mensaje advirtiendo al usuario que debe seleccionar un proceso	[PD] 21-5-09	Los desarrolladores se mostraron receptivos
Conexión a proceso	5	El sistema no muestra el mensaje de confirmación para elegir tipo de usuario cuando el proceso está inactivo	Seleccionar tipo Opción conectarse a proceso	Pruebas de validación	X		El sistema debe cumplir con los requisitos del dodo virtual	[PD] 21-5-09	Los desarrolladores se mostraron receptivos
Conexión a proceso	6	El sistema no envía al usuario a la interfaz correspondiente cuando elige ser maestro de proceso	Seleccionar Tipo Opción Conectarse a proceso Opción maestro	Pruebas de validación	X		El sistema debe cumplir con los requisitos del nodo virtual	[PD] 21-5-09	Los desarrolladores se mostraron receptivos

Luego de realizarse las pruebas de caja negra sobre el software y detectarse no conformidades en el sistema se hizo necesario la respuesta por parte de los desarrolladores a estas no conformidades. En el presente trabajo los desarrolladores corrigieron los defectos y luego se aplicaron las pruebas de caja negra nuevamente en

una segunda iteración para asegurar que los programadores hayan solucionado los problemas encontrados.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo Central**
[SC2:Autenticarse]	EC 2.2: Sistema con capacidad.	En este escenario se comprueba que el sistema muestre un mensaje indicándole al usuario que no hay capacidad en el sistema.	1. El usuario selecciona la opción autenticarse. 2. El sistema verifica que exista disponibilidad de conexión. 3. El sistema muestra un mensaje indicando que no hay de capacidad de conexión.
	EC 2.4: Usuario no registrado	En este escenario se verifica que el sistema le impida el acceso al sistema a un usuario que no se encuentre registrado.	1.El usuario selecciona la opción autenticarse 2.El sistema verifica que exista disponibilidad de conexión 3. El sistema muestra el formulario para la autenticación del usuario. 4.El usuario introduce los datos requeridos 5. El sistema muestra un mensaje de error indicando que debe registrarse para poder acceder.

	EC 2.6: Usuario registrado rol administrador	En este escenario se verifica que un usuario que se encuentra registrado con rol de administrador tenga acceso al módulo de usuario luego de introducir los datos correctos.	1. El usuario selecciona la opción autenticarse. 2. El sistema verifica que exista disponibilidad de conexión. 3. El sistema muestra el formulario para la autenticación del usuario. 4. El usuario introduce los datos requeridos. 5. El sistema da acceso al módulo de administrador.
[SC3: Conexión a Proceso]	EC 3.2: Conexión a proceso con capacidad	En este escenario se prueba que el sistema permita conectarse a un proceso que tiene disponibilidad de conexión.	1. El usuario selecciona la opción conectarse a proceso. 2. El sistema muestra una interfaz con los tipos de procesos existentes. 3. El sistema muestra los procesos del tipo seleccionado 4. El usuario selecciona un proceso y acepta conectase al proceso seleccionado 5. El sistema conecta al usuario al proceso seleccionado.

	EC 3.5: Conexión a proceso inactivo con capacidad como usuario maestro	En este escenario se verifica que el sistema muestra un mensaje de elección de ser usuario común o maestro del proceso.	1. El usuario selecciona la opción conectarse a proceso. 2. El sistema muestra una interfaz con los tipos de procesos existentes. 3. El sistema muestra los procesos del tipo seleccionado 4. El usuario selecciona un proceso y acepta conectarse al proceso seleccionado 6. El sistema muestra un mensaje de elegir tipo de usuario del proceso 7. El usuario elige ser usuario maestro del proceso. 8. El sistema muestra la interfaz de usuario maestro del proceso
	EC3.6: Conexión a proceso inactivo como usuario común	En este escenario se prueba que un usuario que elige conectarse como usuario maestro de un proceso el sistema lo envía a la interfaz correspondiente al CUS Configurar _Proceso.	1. El usuario selecciona la opción conectarse a proceso. 2. El sistema muestra una interfaz con los tipos de procesos existentes. 3.El sistema muestra los procesos del tipo seleccionado 4.El usuario selecciona un proceso y acepta conectarse al proceso seleccionado 5.El sistema muestra un mensaje de elegir tipo de usuario del proceso 6.El sistema muestra la interfaz de usuario común del proceso

Id del escenario	Escenario	Variable 1	Variable 2	Variable N	Respuesta del Sistema	Resultado de la Prueba
EC 2.2	Sistema con capacidad.	I	V	V	El sistema muestra un mensaje indicando que no hay capacidad de conexión.	Se obtuvo una no conformidad
		V	I	V		
		V	V	I		
EC 2.4	Usuario no registrado	I	V	V	El sistema muestra un mensaje de error indicando que debe registrarse para poder acceder.	Se obtuvo el resultado adecuado
		V	I	V		
		V	V	I		
EC 2.6	Usuario registrado rol administrador	I	V	V	El sistema da acceso al módulo de usuario.	Se obtuvo el resultado adecuado
		V	I	V		
		V	V	I		
EC 3.2	Conexión a proceso con capacidad	I	V	V	El sistema conecta al usuario al proceso seleccionado,	Se obtuvo el resultado adecuado
		V	I	V		
		V	V	I		
EC 3.5:	Conexión a proceso inactivo con capacidad	I	V	V	El sistema muestra un mensaje de elegir tipo de	Se obtuvo el resultado adecuado
		V	I	V		

		V	V	I	usuario del proceso	
EC3.6:	Conexión a proceso inactivo con capacidad como usuario maestro	I	V	V	El sistema muestra la interfaz de usuario maestro del proceso	Se obtuvo el resultado adecuado
		V	I	V		
		V	V	I		

Registro de defectos y dificultades detectados

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
Autenticarse	1	El sistema no comprueba en la ventana de inicio si hay capacidad de conexión	Elegir autenticarse, Aceptar	Pruebas de validación		X		[PD] 29-5-09	El cliente piensa que no es necesario que se realice como aparece en las especificaciones

Para una mayor entendimiento de los resultados obtenidos durante el desarrollo de las pruebas de caja negra aplicadas al software se utilizó una gráfica (ver figura 12) donde se muestran los resultados obtenidos en porciento.

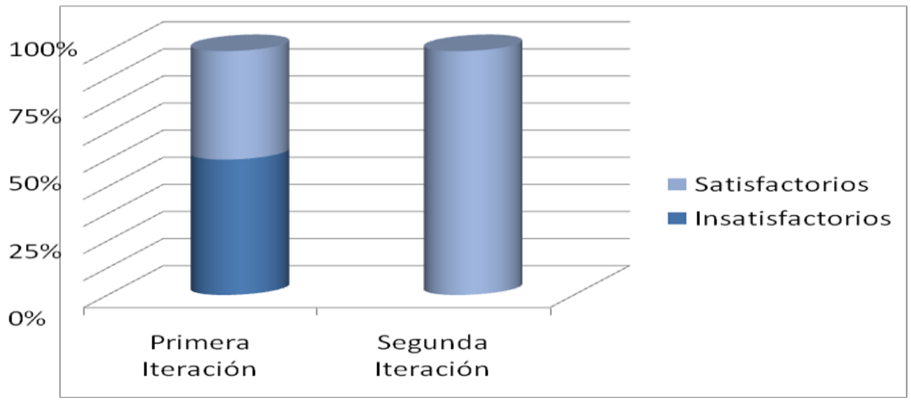


Fig 13. Resultado de las pruebas de caja negra por iteraciones.

3.5 Conclusiones.

A lo largo del capítulo se demostró la importancia que tienen las distintas pruebas de software aplicadas para validar el correcto funcionamiento de la aplicación permitiendo la detección de errores durante la implementación. Para ello se usaron las técnicas de prueba: unidad y caja negra. Las pruebas de unidad usando el módulo de pruebas QTestLib, permitió una mayor agilización del proceso de pruebas. Haciéndolo más sencillo aunque cuando son métodos complejos se hace un poco más difícil el diseño de los casos de prueba. Por otra parte las pruebas de caja negra permitieron detectar el porciento de no conformidades teniendo en cuenta las descripciones de los casos de usos a los cuales se les dio solución una vez detectados.

CONCLUSIONES GENERALES.

Este trabajo permitió brindar a los estudiantes de la carrera de ingeniería automática una herramienta que les permita probar estrategias de control. Ayudando así, al fortalecimiento de la actividad práctica necesaria para el buen desempeño en su vida laboral.

- Se realizó el estudio del arte de los Nodos virtuales en Cuba y el mundo, y se determinó que no existía ninguna aplicación que respondiera a la necesidad que dio origen a esta investigación.
- Atendiendo a los requisitos del Nodo Virtual de Procesos se seleccionó como lenguaje de programación el C++ y como entorno de desarrollo el QTCreator.
- Se proveyó a la aplicación de un soporte matemático a través de DLLs con métodos numéricos compilados lo que permitió una mejor integración con la aplicación y la implementación de los métodos numéricos que necesita.
- Y por último se validó la implementación realizada a través de las técnicas de pruebas de unidad y caja negra que arrojaron resultados satisfactorios en un 100%.

RECOMENDACIONES

Se recomienda para versiones futuras del software:

- La inclusión del plugin de Qt necesario para la conexión con la base de datos de postgreSQL que permitiría usar el servidor en Windows.
- La creación y fomento de una comunidad de desarrollo de programación en Qt en la Universidad de las Ciencias Informáticas, ya que a lo largo del presente trabajo se pudo apreciar que Qt facilita en gran medida el trabajo con C++, haciéndolo más ameno y fácil de usar. Esto se debe a que incluye una gran gama de clases para el trabajo con C++, las c uales le brindan al lenguaje una mayor sencillez y a la vez una gran potencia, además de una excelente documentación. Otro aspecto importante es que es multiplataforma y presenta licencia libre, lo que será una forma de contribuir con el movimiento de software libre en la Universidad.

Bibliografía:

Allende J. Java 2 [Libro]. - [s.l.] : McGraw-Hill, 2005. - 2ª edición.

Bao Z., R. Fedkiw A. y Molino N. Virtual Node Algorithm for Changing Mesh Topology During Simulation [En línea]. - 2004.

Bonaparte U. [y otros] Universidad Tecnológica Nacional Facultad Regional Tucumán [En línea]. - Universidad Tecnológica, Facultad Regional Tucumán, 2008. - 12 de 2008. - http://www.frt.utn.edu.ar/sistemas/paradigmas/poo.htm.

Charte F. Visual C# .net [Libro]. - [s.l.] : Anaya Multimedia, 2002.

Eckel B. Thinking in C++ [Libro]. - 2000.

Escobar M. y Ortiz L. Análisis y Diseño de un Nodo Virtual de Procesos [Libro]. - Universidad de las Ciencias Informáticas. Habana : [s.n.], 2008.

Gámez Y. "Herramienta interactiva para la enseñanza en la carrera de Ingeniería Automática: Nodo Virtual de Procesos." [Libro]. - Instituto Superior Politécnico "José Antonio Echeverría". Habana : [s.n.], 2008.

Gonce S. Arquitectura de un Nodo Virtual de Procesos. - Ciudad Habana : [s.n.], 2008.

Herrscher K. y Maier S. On node virtualization for scalable Network Emulation [Libro]. - 2003.

Hosseinzaman A. y Bargiela A. [En línea] // ADA's Virtual Node based Water System. - 1995. - http://www.intelligentmodelling.org.uk/Papers/rttg-publ10.pdf.

Jiménez A. SCILAB. COMPUTACION NUMERICA BAJO LINUX Y WINDOWS [Informe]. - [s.l.] : Universidad de Cádiz, 2007.

López A. y Rodríguez A. DLLs Con métodos numéricos compilados [Libro]. - 2009.

Maier S. y Herrsche K. Simulation of Discrete Stochastic Systems [Libro]. - Chicago, Illinois : Science Research Associates, 2004.

Maier S. y Herrscher K. On node virtualization for scalable Network Emulation [En línea]. - 2003. - http://www.ipvs.uni-stuttgart.de/abteilungen/vs/abteilung/mitarbeiter/eigenes/maiersn/start/maier05node.pdf.

Miyachi T. y Chinen K. StarBED and SprinOS: Large scale general purpose network testbed and supporting software [Informe]. - 2006.

Nokia Qt Reference Documentation (Open Source Edition). - 2009.

Perez S. y Baltasar J. Guía de estilo de programación en C++ [Libro]. - [s.l.] : Escuela Superior de Ingeniería Informática Universidad de Vigo, 2005.

Pressman Roger S. Ingeniería de Software un enfoque práctico [Libro]. - 2002.

Rodríguez P. Paradigmas de Programación [Libro]. - Ciudad de México : Universidad Nacional Autónoma de México, 2008.

Sloter A. y Kleper S. Professional C++ [Libro]. - [s.l.] : Wrox Pr Inc, 2005.

Wesley A. The Annotated C++ Reference Manual [Libro]. - 2001.

Wu Y A virtual node method for treatment of wells in modeling multiphase Flow in reservoirs [Informe]. - 1999.

Glosario de Términos:

Bytecode: Es el código independiente del hardware generado por el compilador Java y ejecutado por su intérprete.

DLL (dynamic-link library)**:** Biblioteca de vínculos dinámicos. Es un archivo ejecutable que actúa como una biblioteca compartida de funciones.

Hilo: Es un subprograma que se invoca a través de la ejecución de un programa principal y a partir de allí se considera independiente.

Métodos numéricos: Solución numérica de sistemas de ecuaciones.

Procesos industriales: Es un conjunto de procesos físicos y químicos interrelacionados, implementados por medios físicos. Todo sistema de procesos tiene entradas y salidas. Entradas pueden ser materia prima, temperatura, concentración, presión, etc. Un sistema está sujeto usualmente a señales o perturbaciones que para compensarlas se hace uso de correcciones o acciones de control.

Signal: Aviso que un objeto puede emitir cuando le ocurre algo (un cambio de estado importante, también denominado evento),

Slot: Método de un objeto que puede ser llamado cuando se genera una señal particular.

Sockets: Es el punto final de un flujo de comunicación bidireccional a través de una

Red de computadoras basada en protocolo IP.

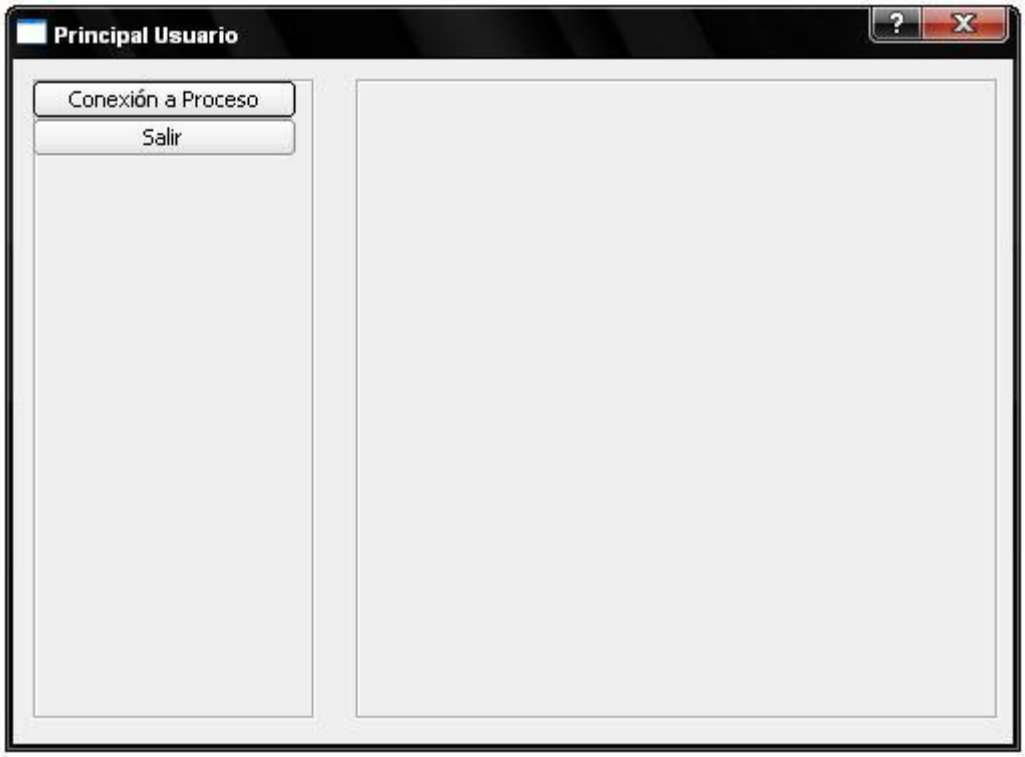
Virtualización: Representación lógica de los recursos más allá de sus limitaciones físicas. Es el método para aumentar la eficiencia en el uso de las TI compartiendo recursos entre muchos sistemas y aplicaciones.

Anexo 1.

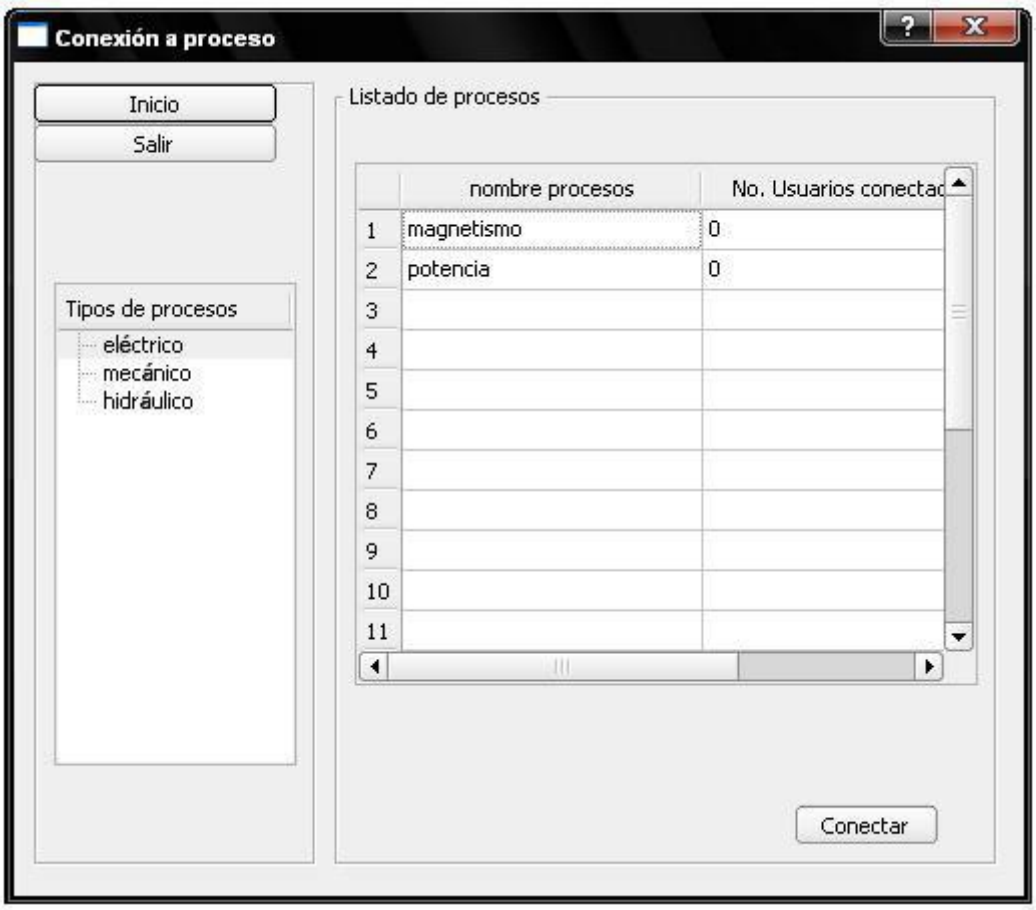
Anexo 2.

Anexo 3.

Anexo 4.



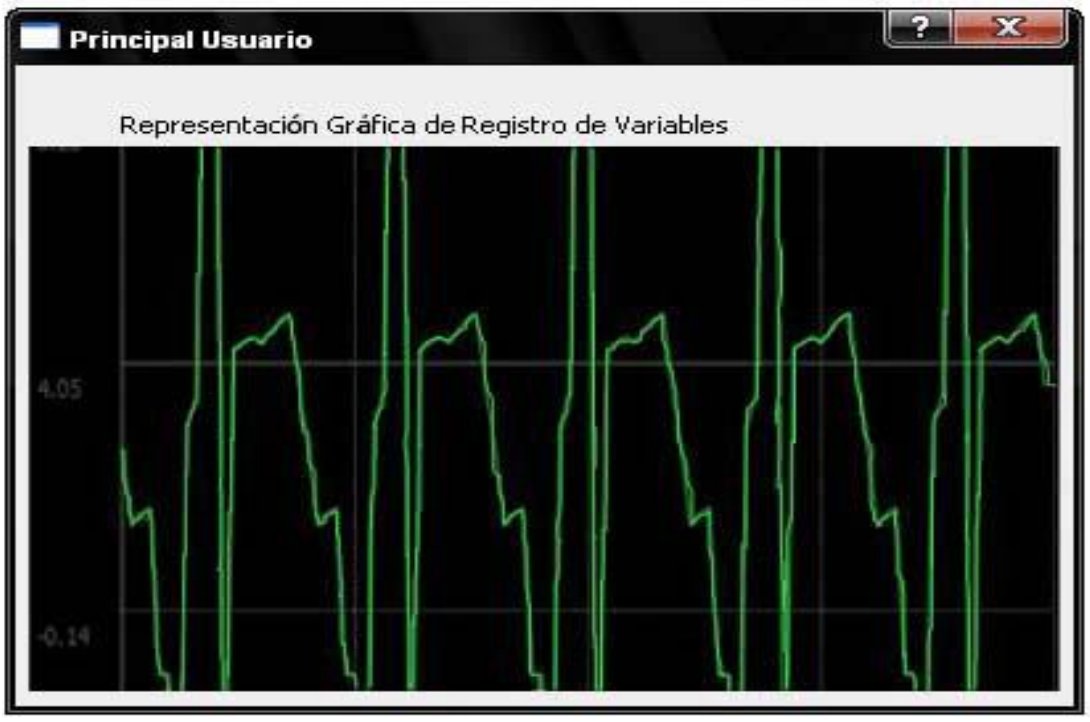
Anexo 5.



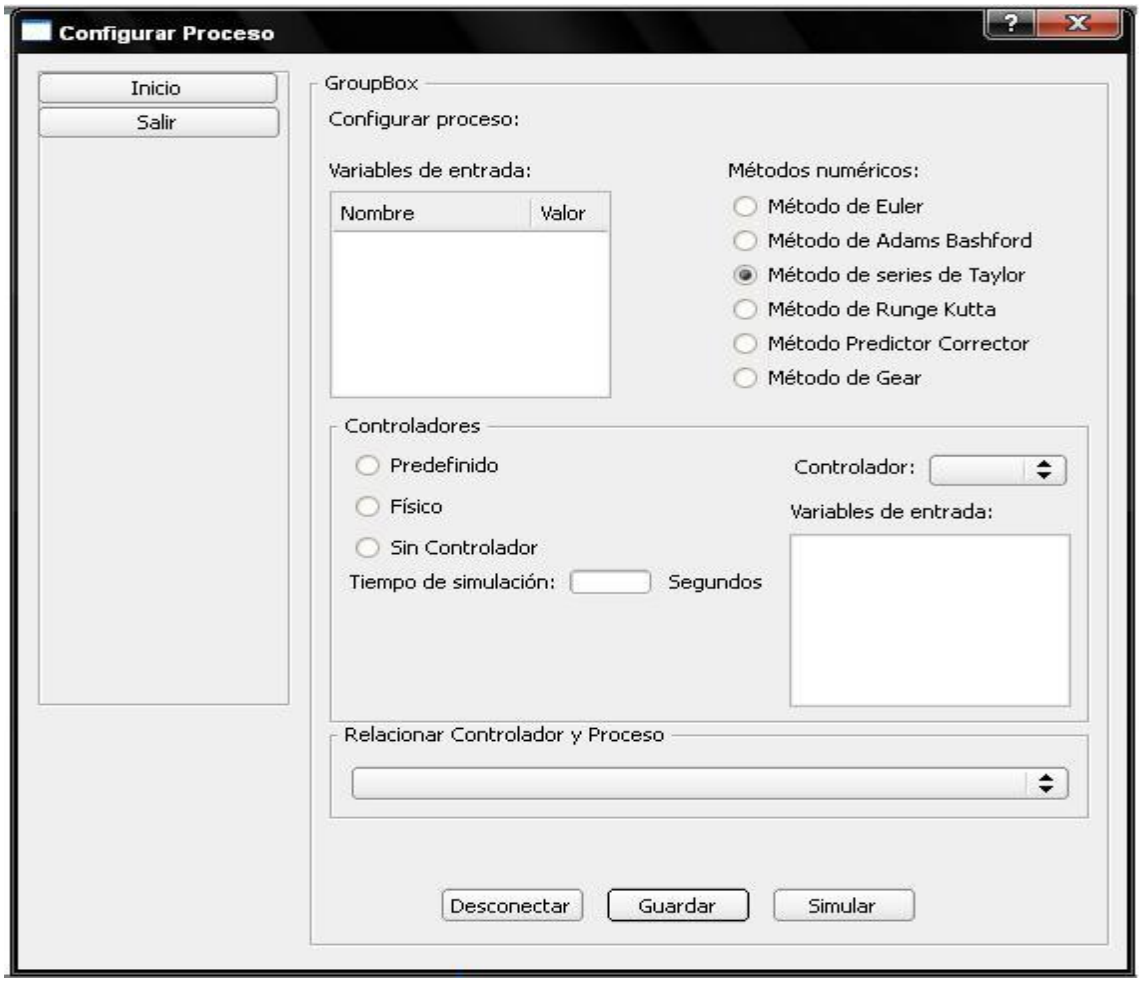
Anexo 6.

Anexo 7.

Anexo 8.

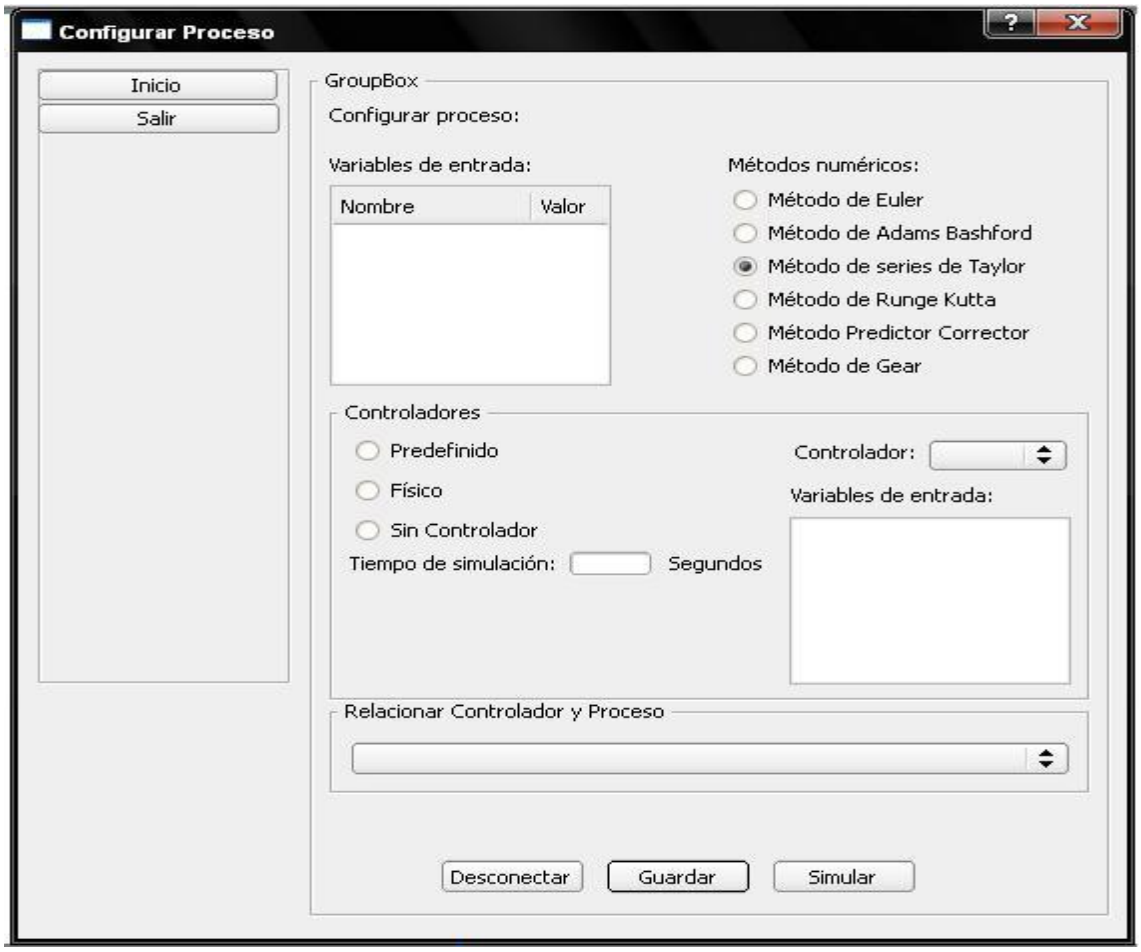


Anexo 9.



Anexo 10.

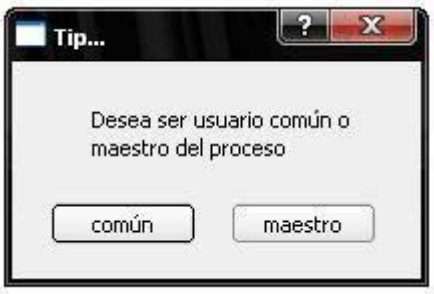
Anexo 11.



Anexo 12.

Anexo 13.

Anexo 14.



Anexo 15.

.