Universidad de las Ciencias Informáticas

Facultad 3

Título: Desarrollo de un módulo de sincronización de procesos concurrentes en cascada para un Nodo Virtual de Procesos.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Nivaldo Faraco Díaz.

Julio César Bellón Rodríguez.

Tutor(es): MSc. Yalice Gámez Batista.

Co-Tutor(es): Ing. Julio Jesús García Guevara.

Ciudad de La Habana, 16 de noviembre de 2013

"Año 54 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Nivaldo Faraco Díaz. Julio César Bellón Rodríguez.

Firma del Autor Firma del Autor

_____ _____

MSc. Yalice Gámez Batista. Ing. Julio Jesús García Guevara.

Firma del Tutor Firma del Co-Tutor

DATOS DE CONTACTO

Nivaldo Faraco Díaz.

Correo Electrónico: nfaraco@estudiantes.uci.cu

Julio César Bellón Rodríguez.

Correo Electrónico: jcbellón@estudiantes.uci.cu

Tutor:

MSc. Yalice Gámez Batista

Correo Electrónico: yaliceg@uci.cu

Co-Tutor:

Ing. Julio Jesús García Guevara.

Correo Electrónico: jjguevara@uci.cu

AGRADECIMIENTOS

Nivaldo:
Agradecer a todas esas personas que han estado ahí cuando las he necesitado, que me hayan apoyado y que hayan contribuido a que hoy yo sea la persona que soy.
A mi madre por estar siempre presente y brindarme ese apoyo y esos consejos que tanto me han servido.
A mi padre por confiar y estar de mi lado siempre que me hizo falta, y por siempre estar ahí.
A mi hermano por sus palabras, por sus preguntas, por su forma de ser y por ser ese hermano que necesitaba.
A mis abuelos por su preocupación, por su sacrificio, por sus consejos y por su apoyo.
A mis tíos y a mis tías, a mi tía Yusa por ser la persona que es.
A mi novia por su ayuda y su comprensión.
A mi tutora Yálice Gámez Batista por haber confiado en nosotros y habernos ayudado tanto, sin usted no hubiera sido posible.
A mi Co-tutor Julio que siempre lo vi como otro tutor más, gracias hermano por haber estado ahí con nosotros en todo el proceso y habernos ayudado tanto.
Agradezco también a esos profesores que han marcado tanto durante mi trayectoria como estudiante no solo en la universidad sino también en las otras enseñanzas, gracias a ustedes por encaminarme en el buen camino.
A mis amigos de la cuadra, del "verde", del pre a todos gracias por influir de manera positiva en mi como persona.
A los amigos de aquí de la universidad para que, ustedes saben que tengo mucho que agradecerles, a Allian la gente del basquet, al Javie, a mis compañeros de clases, a Doris, a Marta, a Scofee, a el Danie, el Carli, el Luisma a los que no son de aula también, a mis compañeros de estudio el Don, Cesar, Diógenes, los Full, Raul, Amilkar, Isidro, los que no pudieron y que no quisieron llegar hasta aquí con nosotros, el menda, el Fide, el Raulo, Lisset, Ariadna, y también a los Pudines que tanto hemos vivido aquí, tantos recuerdos, tantas asaderas que no es fácil. Ustedes saben el 01, el magic, recfull, el 03, al guaya mi herma, naruto.
A todos ustedes gracias por todos esos momentos buenos y malos que hemos pasado en esta Universidad, a todos en general porque son tantos que no es fácil, a todos gracias por haberlos conocidos y por haber compartido tanto en esta universidad.
Y a mi compañero de tesis que sin él tampoco hubiera sido posible este momento, gracias brother por haberte conocido y más que un amigo tener otro hermano más.

AGRADECIMIENTOS

Julio Cesar:
A mi madre por el cariño y apoyo que siempre he recibido de ti y con el cual he logrado culminar mi esfuerzo, terminando así mi carrera profesional, que es para mí la mejor prueba de cariño y amor. Mamá todo lo que pudiera agradecerte hoy es poco porque tú eres mi mayor tesoro. Hoy hicimos un sueño realidad porque este sueño era tanto mío como tuyo.
A Mami por ser mi segunda mamá y a Papi por ser un ejemplo a seguir.
A mi hermano José por ser tan buen hermano. Sé sin dudas que serás ingeniero igual que yo.
A mi padre por estar siempre pendiente de mí.
A mis tías Marley y Madey por darme toda su confianza. A mis tíos Juan Miguel y Arley que se han convertido en verdaderos tíos. A mi abuelo José y a Tata por su cariño y ser como una abuela para mí.
A mi abuela Nilda por ser tan buena conmigo y complacerme todas mis malacrianzas. A mis tías Nildita, Nancy y Minerva por preocuparse por mí y ser tan cariñosas conmigo.
A mi jevita linda Geisy, por apoyarme siempre y por quererme como soy. Te quiero infinito sin muela.
A mi hermano Tito que este título es tanto mío como tuyo.
A mi tutora Yalice Gámez Batista y a mi Co-tutor Julio por haber confiado en nosotros y habernos ayudado tanto, sin ustedes no hubiera sido posible.
A mis amigos de la cuadra, del "verde", del IPI, a todos gracias porque de todos pude aprender algo positivo que me ha ayudado a mejorar como persona.
A mis amigos y amigas de la UCI de verdad súper agradecidos con todos: Scofi, Mailenis, Greisy, Ariadna, Lisset, Mari, Ailyn, Laura, Maybel, Jimeno, Leidy, Yessi, Doris, Martha, Daniel, Carlos, Luisma, al loco de Fidel, Humberto, el Javico, Yasiel alias el Board. A los míos Raulin, Cesar, Isidro, Luiso alias el Don, Raidel, Diogenes. A los locos del edificio 9, Amilkar, Buti, Ricardo, el Chino, Duvergel. A los pudines que los quiero de verdad Franko, Yordan, Raulo alias el Guaya, Juan David, Ariel, Reidis, Naruto. A todos ustedes gracias por todos esos momentos buenos y malos que hemos pasado, a todos gracias por haberlos conocidos y por haberme brindado su amistad los quiero a todos.
A mi compañero de tesis que sin él no hubiera sido posible que mi gran sueño se hiciera realidad, gracias por existir y más que un amigo eres mi hermano y sabes que conmigo puedes contar para lo que sea, te quiero.

DEDICATORIA

Nivaldo:

A mi familia completa, por siempre haberme apoyado en todo y por esa confianza que siempre me tuvieron y la cual no espero defraudar nunca.

A mi mamá por ser esa madre que todos quisieran pero que solo yo tengo, por los sacrificios que has hecho por mí, por la educación que me has dado y sobre todo por haberme guiado por buen camino para convertirme en la persona que soy hoy.

A mi papá por siempre estar ahí, por inculcarme esos valores que debe tener una persona para enfrentarse a la vida y por ser un ejemplo a seguir como persona y como padre.

A mi hermano por las alegrías que siempre me has dado y por esa forma de ser única que te hace tan especial y tan importante.

A mi abuelos, por siempre estar presentes en mi vida, por su preocupación, por su amor y por el privilegio de poder ser su nieto y de poder tenerlos conmigo, a pesar de que la distancia no permita que estén aquí hoy.

A mi tía Yusa que no hay forma de que faltara por ese amor, por esa preocupación y por ser esa grandísima persona que eres, que admiro y que quiero.

A mi novia que ha sabido acompañarme y quererme como soy, y que es esa persona tan importante para mí.

Y a todas esas personas que me quieren y que esperan mucho de mí.

DEDICATORIA

Julio Cesar:

A mi madre por ser la mamá más linda del mundo, por darme su apoyo incondicional sin esperar nada a cambio, por atravesar conmigo momentos duros, momentos alegres, momentos en la vida en los cuales no sabes qué hacer, siempre estás ahí para guiarme para demostrarme que hay una salida y que no todo está perdido. Por ver siempre lo mejor de mí. Por ser quien eres, lo más grande que tengo en la vida.

A Mami por ser tan especial, por tus regaños dulces que sé que lo haces para enseñarme. Por malcriarme en todo, yo sé que soy tu nieto lindo y tú eres mi abuelita linda.

A Papi por ser más que un abuelo, por sus consejos y por ser mi ejemplo a seguir. Yo quisiera significar para mis nietos la mitad de lo que significas para mí, con eso estaría satisfecho.

A mi hermano José, porque sin ti, en todos estos años de mi vida me hubiera sentido solo y gracias a ti tengo fuerzas para seguir adelante, tú sabes que te quiero con la vida y que conmigo puedes contar para lo que sea. Siempre voy a estar para lo que necesites, no dudes eso nunca.

A mi padre ese loco que siempre me hace reír y sé que está orgulloso de mí. Por halarme las orejas cada vez que me portaba mal en la escuela. Con tus virtudes y defectos no quiero otro padre, te quiero a ti.

A mis tías Marley y Madey por ser las tías que todos quisieran tener. Por darme su apoyo, su confianza y siempre estar preocupadas por mí, de verdad las quiero mucho.

A mi hermosa familia, que yo quiero mucho, le dedico este triunfo en mi vida.

A mi hermano Tito que no pudo graduarse por dificultades de la vida a pesar de que conocimientos para hacerlo le sobraban. Este título es tanto tuyo como mío, gracias por ser mi amigo y más que un amigo un hermano.

Y a todas esas personas que me quieren y me tienen presente siempre.

RESUMEN

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TICs) ha potenciado muchas áreas del conocimiento, especialmente el área de la simulación. La simulación dentro de la investigación-desarrollo e innovación, ocupa un papel fundamental pues permite evaluar modelos de diversas índoles, entre los cuales se encuentran los de procesos industriales. En el curso 2011-2012 se realizó la implementación de la aplicación Nodo Virtual de Procesos (NVP). Esta aplicación permite la simulación concurrente de procesos industriales a partir de sus modelos matemáticos. De esta forma brinda a los estudiantes de Ingeniería Automática la posibilidad de simular procesos industriales simples y probar sus estrategias de control.

Este trabajo propone un módulo que permitirá la simulación de procesos en cascada en el NVP. De esta forma se podrán evaluar modelos matemáticos escalablemente complejos, balancear la carga de los nodos y sincronizar los procesos entre ellos.

Palabras claves:

Nodo Virtual de Procesos, Procesos en cascada

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la actualidad todas las esferas de la sociedad están siendo influenciadas por los más recientes adelantos científico –técnicos, especialmente los relacionados con las Tecnologías de la Información y las Comunicaciones (TICs). Una de las áreas más beneficiadas es la de simulación de procesos industriales.

La simulación industrial es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias dentro de los límites impuestos por un cierto criterio o un conjunto de ellos para el funcionamiento del sistema. (García, 2011)

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 troncales en las que se utilizan modelos de procesos industriales. Con el objetivo de dar respuesta a las limitaciones de recursos y a las necesidades de los estudiantes de poder fundamentar sus conocimientos teóricos desde el punto de vista práctico, en la Universidad de Ciencias Informáticas (UCI) en conjunto con desarrolladores del Instituto Superior Politécnico José Antonio Echeverría, se desarrolló un proyecto para la elaboración de un Simulador concurrente de procesos industriales, Nodo Virtual de Procesos (NVP).

Durante el curso 2007-2008 se realizó el “Análisis y Diseño” (Escobar, 2008), y en demás trabajos de diploma (Gonce, 2008), (Cleger, 2009), (Hernández, 2009), (Rivero, 2009), (Muñoz, 2010), (García, 2010) se solucionaron problemáticas como la arquitectura y la implementación de la misma.

Después de un estudio de la versión actual, del Nodo Virtual de Procesos (NVP) se llegó a la conclusión de que en la versión actual, debido a la complejidad técnica que se requiere para la utilización de los controladores físicos, sólo se simulan procesos simples. Esto limita el empleo de la herramienta ya que no permite la simulación concurrente de procesos industriales en cascada, reduciendo las posibilidades de experimentación con la misma.

Teniendo en cuenta la situación planteada se identifica como **problema a resolver**: La herramienta interactiva Nodo Virtual de Procesos sólo permite la simulación de procesos industriales simples lo que limita sus posibilidades de experimentación.

La investigación se enmarca en el **objeto de estudio**: La modelación y simulación de procesos industriales en cascada.

Incidiendo en el **campo de acción**: Herramientas interactivas para la simulación de procesos industriales en cascada.

Para dar respuesta al problema se propone el **objetivo general:** Desarrollar una solución Informática que permita la simulación de procesos en cascada en el Nodo Virtual de Procesos, partiendo de la última versión del mismo.

La investigación está basada en la siguiente **idea a defender**: El desarrollo de un módulo de sincronización para el Nodo Virtual de Procesos permitirá la simulación concurrente de procesos industriales en cascada.

Para darle cumplimiento al objetivo propuesto, se han derivado un conjunto de **objetivos específicos** orientados fundamentalmente a proveer los elementos necesarios para la implementación de la solución. Estos se listan a continuación:

- ✓ Caracterizar los sistemas de simulación de procesos en cascada existentes, a partir de la sistematización de los referentes teóricos relacionados con la modelación y simulación de estos procesos.
- ✓ Realizar el análisis y diseño del módulo de sincronización de la simulación de procesos concurrentes en cascada.
- ✓ Implementar el módulo de sincronización de la simulación de procesos concurrentes en cascada.
- ✓ Verificar la solución propuesta aplicando diferentes pruebas y métricas.

Para dar cumplimiento a los objetivos específicos, **las tareas de investigación** trazadas quedaron estructuradas de la siguiente forma:

- ✓ Elaboración del marco teórico de la investigación.
- ✓ Estudio de sistemas existentes para resolver problemáticas similares.
- ✓ Confección de los artefactos del análisis y diseño del módulo de sincronización.
- ✓ Implementación del diseño propuesto.

- ✓ Selección de las pruebas para la verificación de la propuesta.
- ✓ Aplicación de las pruebas seleccionadas y constatación de los resultados.

Para el desarrollo del trabajo se aplican los **métodos científicos**:

Métodos teóricos:

- ✓ Hipotético-deductivo: A partir de la interpretación de la realidad se establecen posibles situaciones o resultados para llegar a conclusiones.
- ✓ 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.
- ✓ Histórico lógico: Se realiza un análisis de cómo han evolucionado las diferentes herramientas de simulación de procesos en cascada. Obteniendo una tendencia de cómo se debe comportar en la actualidad.

Métodos empíricos:

- ✓ Observación: Mediante guías de observación se le dará seguimiento al desarrollo de la aplicación.

El presente documento se estructura en 3 capítulos que incluyen los aspectos relacionados con el módulo de sincronización de procesos concurrentes en cascada para un Nodo Virtual de Procesos.

CAPÍTULO 1. Fundamentación Teórica: Contiene la base teórica para comprender el problema; es el estudio profundo del tema, a través de la indagación bibliográfica y la consecuente estructuración lógica del material y el análisis crítico del mismo. Se presentan el conjunto de las técnicas, herramientas, lenguajes y metodología involucrados en el desarrollo de la propuesta y se realiza la justificación de las seleccionadas para la solución del problema.

CAPÍTULO 2. Planificación, diseño e implementación: Contiene la especificación de elementos relacionados con la descripción y análisis del sistema informático: los requerimientos del software a través de las historias de usuario. Se describen los principales artefactos generados en estas fases como: Plan de iteraciones, Plan de Entrega, Modelo de datos, Tarjetas CRC y las Tareas de programación, además se plantea el uso de diferentes artefactos de la metodología RUP, claves para un mejor entendimiento del flujo y de la arquitectura de la

aplicación. Algunos de estos artefactos son: diagrama de clases del diseño y diagrama de despliegue.

CAPÍTULO 3. Validación de la solución propuesta: En este capítulo se abordan las métricas y pruebas empleadas para comprobar la calidad del análisis y diseño del sistema. Se trata el tema de las pruebas de software realizadas,y se diseñan y ejecutan los casos de pruebas para probar el correcto funcionamiento de los componentes implementados. Se hace un análisis de los resultados obtenidos luego de la aplicación de las métricas y realización de las pruebas, y se le da solución a las no conformidades detectadas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se abordan los conceptos claves para la comprensión de la investigación. Se realiza un análisis de los aspectos técnicos y teóricos sobre los procesos industriales en cascada y su simulación. Se realiza además un estudio de las tecnologías actuales y de las principales herramientas que constituyen opciones para el desarrollo del sistema.

1.1 Nodos Virtuales

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, 2010)

Para dar solución a esta problemática se optó por un Nodo Virtual, el cual 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, 2005)

Atendiendo a esto se define como nodo virtual de procesos, el software que permitirá implementar los modelos de diferentes procesos para su simulación o para probar aplicaciones en tiempo real. Será necesario que varios procesos se encuentren activos de manera simultánea, lo que requiere de gran cantidad de nodos y computadoras que no están disponibles generalmente. (Gámez, 2010)

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. (Hernández, 2009)

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, 2008)

Esta filosofía de trabajo permite la simulación simultánea de diferentes procesos concurrentes sin que interfieran unos 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, 2010)

- ✓ 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 Nodo Virtual de Procesos

En el mundo se han desarrollado varias aplicaciones que utilizan los nodos virtuales, con el objetivo de aprovechar al máximo los recursos de las computadoras. A pesar de que existen varias aplicaciones que hacen uso de los nodos virtuales, no se ha identificado una herramienta informática que funcione como un nodo virtual de procesos, y específicamente, que simule procesos industriales. (Gámez, 2010)

La ausencia de un software con estas características motivó el desarrollo de una herramienta interactiva para la simulación concurrente de procesos industriales, por parte de la Universidad de las Ciencias Informáticas (UCI) y del Centro Universitario José Antonio Echeverría. Esta herramienta fue desarrollada en el curso 2011-2012 para su integración en el proceso de enseñanza aprendizaje de la carrera de Ingeniería Automática.

El Nodo Virtual de Procesos se caracteriza por:

- ✓ Simular los procesos de forma concurrente con el uso de sockets e hilos de ejecución
- ✓ Gestiona los reportes relacionados con las acciones realizadas por el usuario

- ✓ Ofrece una interfaz amigable y de fácil uso
- ✓ Presenta una arquitectura en tres capas brindando una organización del código, la reutilización y la flexibilidad del mismo.

La versión existente del NVP tiene como limitación que solo simula procesos simples, siendo para esto necesario tener toda la información contenida en un solo nodo físico. Con el propósito de perfeccionar el Nodo Virtual de Procesos para que tenga un mayor rango de funcionalidades, se decide extender los modelos utilizados en esta herramienta a modelos en cascada. Con estas prestaciones el Nodo Virtual de Procesos podrá simular procesos complejos en diferentes nodos físicos, de forma tal que cada nodo podrá interactuar con otros nodos, enviando y recibiendo información de forma sincronizada. Así habrá un nodo maestro que será el encargado de distribuir la red de nodos, y será el punto de conexión entre ellos. Esto se realizará según el flujo definido por el usuario.

1.3 Modelación y simulación de procesos industriales.

La modelación y simulación de procesos industriales es de vital importancia en la actualidad. Se evidencia porque son cada vez más importantes para el desarrollo y la mejora de productos, y lograr al final la posición de mayor competitividad y margen de beneficios. La modelación y simulación basada en computadoras ha estado en constante desarrollo por alrededor de 50 años. Su desarrollo ha favorecido gran variedad de contextos industriales, tales como la industria aeroespacial, la fabricación, la agricultura, y otros. En la actualidad factores como la necesidad de lograr un mejor rendimiento del producto o proceso, la creciente complejidad de los sistemas tecnológicos avanzados, la creciente necesidad de una ventaja competitiva, la eficiencia, la economía, y la disminución de los costos, y el acoplamiento de la modelación y la simulación con otros potentes métodos basados en computadoras, son evidencias de la necesidad cada vez mayor de la modelación y la simulación.

Para una mejor comprensión es importante definir modelo y simulación(García, 2011):

- ✓ Un objeto “X” es un **modelo** del objeto “Y” para el observador “Z”, si “Z” puede emplear “X” para responder cuestiones que le interesan acerca de “Y”.

- ✓ **Simulación:** es el proceso de diseñar un modelo de un sistema real y llevar a cabo experiencias con él, con la finalidad de aprender el comportamiento del sistema o de evaluar diversas estrategias para el funcionamiento del sistema.

De forma general, la descripción de las características de interés de un sistema se conoce como modelo del sistema, y el proceso de abstracción para obtener esta descripción como modelado. (Taylor, 2011) Existen dos tipos fundamentales de modelados:

- ✓ El modelado hard que consiste en el uso de principios científicos (Ley de Newton, leyes de Kirchhov, leyes de la termodinámica, etc.) para obtener un modelo analítico. La factibilidad de hacerlo varía mucho de una disciplina a otra. Por ejemplo en disciplinas como la robótica se hace más fácil que en áreas de aplicaciones biológicas donde es muy difícil e incluso en algunos casos imposible.

- ✓ El modelado soft se sustenta en el uso de técnicas de montaje formales o informales para que coincida con las matemáticas. Modelos matemáticos del comportamiento de los datos, observaciones, el LotkaVolterra, ecuaciones de competencia para la dinámica de la población, constituyen ejemplos claros de este proceso.

Existen un gran número de métodos y paquetes de software para la identificación del modelo. El desarrollo de un modelo de sistema bien puede implicar una combinación de los enfoques anteriores. (Taylor, 2011)

- ✓ El modelado hard produce lo que se denomina “caja blanca” del modelo, y un modelo basado únicamente en la determinación de los parámetros de un modelo no físico.

- ✓ Una combinación del modelado hard y soft produce “caja gris” del modelo.

A la hora de llevar a cabo la simulación, se debe tener en cuenta algunas consideraciones principales entre las que destaca una correcta definición del sistema, para tener una representación exacta del sistema que se desea simular. Es necesario hacer primeramente un análisis preliminar de este, con el fin de determinar la interacción con otros sistemas, sus restricciones, las variables que interactúan en su interior y sus interrelaciones, las medidas de efectividad que se van a utilizar y estudiar, y los resultados que se esperan obtener del estudio. Otra consideración es formular el modelo con el cual se obtendrán los resultados deseados. En la formulación del modelo es necesario definir todas las variables que forman parte de él, sus

relaciones lógicas y los diagramas de flujo que describen en forma completa el modelo. También es importante que se definan con claridad y exactitud los datos que el modelo va a requerir para producir los resultados deseados. Con el modelo definido, el siguiente paso es decidir qué lenguaje se va a utilizar para procesarlo en la computadora.

Los siguientes elementos a tener en cuenta son la validación con el objetivo de detallar posibles deficiencias en la formulación del modelo o en los datos del modelo, y la interpretación que consiste en la interpretación de los resultados que arroja la simulación y con base a esto se toma una decisión. (Taylor, 2011)

1.3.1 Definición y clasificación de un proceso en cascada

Una de las técnicas que se han desarrollado y utilizado con frecuencia es el control en cascada. Se utiliza con el fin de mejorar el control por retroalimentación y es válido en gran número de aplicaciones. Sin embargo en muchas de ellas la sintonización del regulador se hace complicada. (Montero, 2009)

El control en cascada es definido por (Montero, 2009) como la configuración de control donde la salida de un controlador se utiliza como retroalimentación para el ajuste de al menos otro controlador.

Existen dos propósitos para usar control cascada:

1. Eliminar el efecto de algunas perturbaciones haciendo la respuesta de regulación del sistema más estable y más rápido.

2. Mejorar la dinámica del lazo de control.

La estructura de control en cascada tiene dos lazos; un lazo primario con un controlador primario también llamado “maestro” K1(s), y un lazo secundario con un controlador secundario también denominado “esclavo” K2(s), siendo la salida del primario el punto de consigna del controlador secundario. La salida del controlador secundario es la que actúa sobre el proceso. (Ver Figura 1)

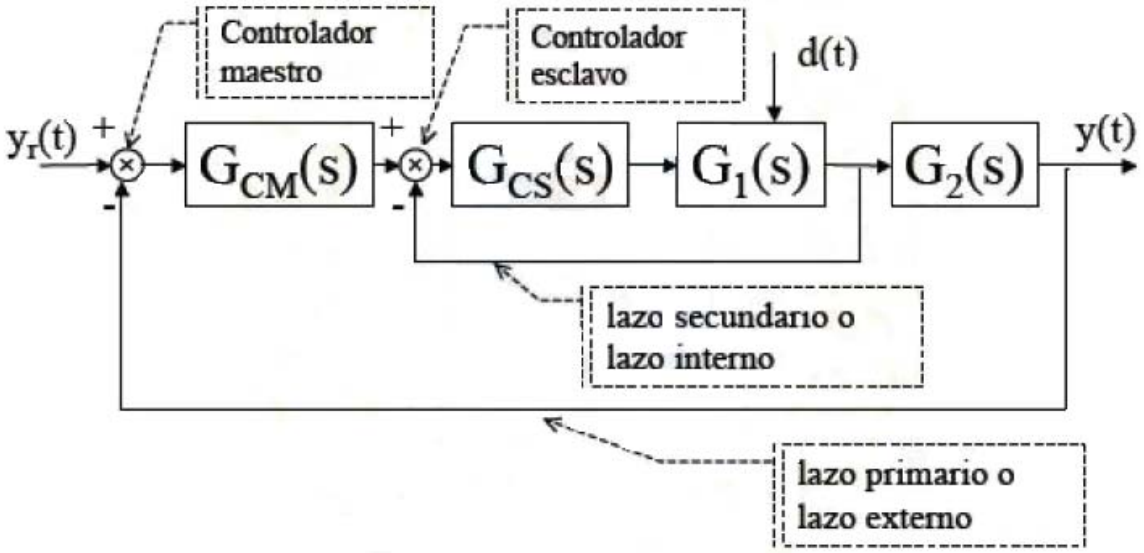


Fig 1. Estructura de control en cascada

Algunas de las ventajas del control en cascada son:

a) Produce estabilidad en la operación.

b) Las perturbaciones en el lazo interno o secundario son corregidas por el controlador secundario, antes de que ellas puedan afectar a la variable primaria.

c) Cualquier variación en la ganancia estática de la parte secundaria del proceso es compensada por su propio lazo.

d) Las constantes de tiempo asociadas al proceso secundario son reducidas drásticamente por el lazo secundario.

e) El controlador primario recibe ayuda del controlador secundario para lograr una gran reducción en la variación de la variable primaria.

f) Es menos sensible a errores de modelado.

g) Incremento de la capacidad de producción.

Algunas limitaciones del control en cascada son:

a) Es aplicable sólo cuando pueden obtenerse mediciones de variables adicionales de proceso.

b) Requiere medir las perturbaciones en forma explícita, y además es necesario un modelo para calcular la salida del controlador.

c) En algunas aplicaciones la variable controlada no puede medirse y la realimentación no puede realizarse.

Los criterios para el diseño de control en cascada son:

Puede ser considerado:

1. Cuando el control realimentado simple no provee un desempeño satisfactorio a lazo cerrado.

2. La medida de la variable es disponible.

1.4 Herramientas para la simulación de procesos industriales en cascada

En el mundo se han desarrollado diversas herramientas que permiten la simulación de procesos en cascada, con el objetivo de dar solución a las diferentes problemáticas y necesidades que se presentan. A continuación se presentan algunas de estas herramientas.

En la Universidad de Cádiz, en Comunidad Autónoma española de Andalucía, que se destaca por su especialización en las disciplinas de ciencias del mar, ciencias náuticas e ingenierías navales se creó **EcosimPro**. Este es un entorno de simulación de sistemas que brinda la posibilidad de tratar simultáneamente materias tan diferentes como la Electrónica, la Mecánica, Mecánica de Fluidos, la Ingeniería Térmica, o el estudio la dinámica de las reacciones Químicas, etc. Es un programa para estudiar modelos de comportamiento de sistemas que permite introducir de forma sistemática las diferentes ecuaciones algebraicas y diferenciales constitutivas del modelo global. (Lozano ,2008) Esta es una herramienta concebida para su aplicación en sistemas de alta complejidad, y su uso es exclusivo de diferentes empresas como la NASA, lo que dificulta su estudio y aplicación para la solución del problema.

La herramienta **Cadsim Plus** creada por Aurel Systems, sirve para la simulación de procesos de fabricación de pasta y papel. Estos presentan un alto grado de complejidad debido a que los

procesos implican varias operaciones unitarias, inclusión de múltiples componentes en los flujos y falta de modelos establecidos. **CADSIM Plus** es una herramienta fácil de usar, totalmente autónoma. Es una plataforma de simulación de procesos que permite al usuario dibujar rápidamente un dibujo diagrama de flujo de proceso y crear una simulación de procesos, todo al mismo tiempo. Es una única herramienta que puede equilibrar diagramas de flujo y retratar condiciones dinámicas. **CADSIM Plus** realiza cálculos precisos y balances de materia de cualquier proceso químico. Se puede utilizar para el diseño, la búsqueda de soluciones para los cuellos de botella del proceso, para realizar un seguimiento de posibles problemas de control de calidad, para refinar las estrategias de gestión de residuos de proceso, para mejorar la eficiencia del proceso. (Aurel Systems, 2013) Esta herramienta se utiliza para la simulación de procesos químicos y no está concebida para la simulación de otro tipo de procesos. Por esta razón no satisface los requerimientos del NVP.

En la Universidad Nacional de Rosario, escuela de Ingeniería Electrónica, se desarrolló el proyecto de ingeniería **Simudrives** (Simulación y Diseño Asistido de Accionamientos Eléctricos Controlados). Diseñada para la investigación de los modernos sistemas de control de movimiento (MCS), a través de simulaciones a nivel del sistema. Consiste de una librería de modelos (Simudrives) desarrollados en el entorno MATLAB / Simulink, más una serie de programas asociados (dentro del mismo entorno) para el ajuste de los controladores. Además incluye un asistente (wizard) para la creación de modelos para una posterior simulación (Interfaz Gráfica de Usuario, o GUI) llamada **Simudrives GUI**. (Felicioni, 2003) Este sistema tiene como base la implementación de una herramienta de simulación digital de accionamientos eléctricos controlados por lo que no es posible utilizarla para la simulación de procesos que tienen una dinámica un tanto diferente. Además se basa en el entorno propietario Matlab/ Simulink, que no es compatible con el NVP.

En la Universidad de la Ciencias Informáticas (UCI), se crea el software para la simulación de sistemas biológicos **Módulo de Simulación y Análisis.** Tiene como objetivo principal realizar simulaciones distribuidas de sistemas biológicos, que puedan ser modelados mediante sistemas de ecuaciones diferenciales, almacenar los resultados y realizar meta-análisis sobre estos datos. Esta aplicación facilitará a los usuarios el proceso de realizar simulaciones de sistemas biológicos, permitiéndole realizar simulaciones distribuidas de los mismos. Brindará la posibilidad de manipular toda la información referente a los sistemas biológicos. Tendrá

almacenada, en una base de datos, información correspondiente a los sistemas biológicos estudiados con anterioridad, para poder modificarlos y compararlos con otros sistemas. Uno de los aportes más importantes es que permitirá realizar el análisis de las simulaciones obtenidas de forma automatizada. (Alonso, 2007) A pesar de sus ventajas, no se considera una solución al problema planteado debido a que la simulación no genera soluciones óptimas a problemas de análisis cuantitativos, cada modelo de simulación es único. Las soluciones e inferencias no son usualmente transferibles a otros problemas, y el modelo de simulación no produce respuestas por sí mismo.

A partir del estudio de diferentes herramientas existentes para la simulación de procesos en cascada, se pudo constatar que ninguna de ellas da respuestas al problema planteado. Por estas razones se optó por el desarrollo de un módulo que de forma fácil e intuitiva permita la sincronización de procesos industriales en nodos distribuidos para el NVP, y con ello la simulación concurrente de procesos en cascada.

1.5 Metodologías de desarrollo de software.

El término **Metodología** se define como un conjunto de métodos eficientes orientados a conseguir un objetivo propuesto. Son un conjunto de procesos que organizados dan una secuencia de pasos a seguir para obtener los hitos propuestos y finalmente el producto final. (López, 2005)

La determinación de una metodología de desarrollo del software, es un factor determinante en el éxito de un proyecto. Su selección incluye un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo de software. Existen dos grandes grupos de metodologías:

Las metodologías tradicionales: Este tipo de metodología requiere de una extensa documentación durante todo el ciclo de vida, ya que pretende prever todo de antemano. Suelen ser eficaces y necesarias cuando se trata de proyectos que requieren de grandes equipos de desarrollo. Dentro de estas metodologías una de las más utilizadas es la Metodología RUP (Rational Unified Process), la cual divide el desarrollo en cuatro fases que definen su ciclo de vida y en nueve flujos de trabajo, seis de ingeniería y tres de soporte (Jacobson, 2000).

Las principales características de RUP son**:**

- ✓ Centrado en la arquitectura: la arquitectura involucra los elementos más significativos del sistema, y está influenciada entre otros por plataformas de software, sistemas operativos, manejadores de bases de datos, consideraciones de desarrollo como sistemas heredados y requerimientos no funcionales. Una vez definida la arquitectura se puede decir que el sistema tiene forma.

- ✓ Dirigido por casos de uso: tiene a los casos de uso como el hilo conductor del proceso de desarrollo. Los desarrolladores crean una serie de modelos de diseño e implementación que los llevan a cabo. Además, estos modelos se validan para que sean conformes a los casos de uso. Finalmente, los casos de uso también sirven para realizar las pruebas sobre los componentes desarrollados.

- ✓ Iterativo e incremental: esta característica propone dividir el proceso de desarrollo en partes, cada una de las cuales incluya las fases de: Requerimientos, Análisis, Diseño, Implementación y Pruebas, con el objetivo de acelerar el ritmo de desarrollo para que el producto salga al mercado en el menor tiempo posible y con mayor calidad.

- ✓ RUP no es factible para el módulo de sincronización de procesos del NVP, debido a que el sistema por sus características de ejecución paralela, se hace muy difícil su descripción en casos de uso. No obstante por su complejidad se necesita una amplia documentación por lo que se elaborarán los artefactos diagramas de despliegue y clases que propone RUP.

- ✓ Las metodologías ágiles: Para este tipo de metodologías es más importante la capacidad de respuesta ante los cambios realizados que el seguimiento estricto de un plan. Se enfatiza en la satisfacción del cliente y promueve el trabajo en equipo. Una de las metodologías más utilizadas es la Metodología XP (Extreme Programming) (Wells. 2009) la cual se basa en una serie de valores, principios y prácticas que brindan una satisfactoria productividad en el proceso de desarrollo del software.

Durante el ciclo de vida en dicha metodología aparecen cambios frecuentes, por lo que a veces el equipo que lo integra no se encuentra preparado para enfrentarlos. Ante tal situación el

equipo de desarrollo enfrenta un conjunto de valores que son importantes para el logro del producto, entre los que se encuentran:

- ✓ Comunicación
- ✓ Coraje
- ✓ Simplicidad
- ✓ Retroalimentación

Cuenta también con los siguientes principios y artefactos, los cuales son indispensables en el desarrollo de la metodología:

- ✓ Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que se adelanta en algo hacia el futuro. Se pueden hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.
- ✓ Re fabricación: se desarrolla en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- ✓ Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

Artefactos esenciales en XP:

- ✓ Historias del usuario
- ✓ Pruebas de aceptación
- ✓ Metáfora
- ✓ Tareas de ingeniería
- ✓ Pruebas unitarias y de integración
- ✓ Plan de iteraciones

- ✓ Código

Luego de haberse realizado el estudio de las metodologías, se decidió centrar el desarrollo del producto sobre la metodología ágil, Extreme Programming (XP) teniendo en cuenta que el grupo de desarrollo encargado del producto de software es muy reducido. Otras de las características que conlleva a la selección de esta metodología son el corto plazo de sus iteraciones y la constante presencia de un cliente, lo cual provoca un constante cambio, reflejándose por concerniente la flexibilidad y la alta respuesta a dichos cambios en el menor tiempo posible.

Además de seleccionar la metodología XP, la cual se propone para guiar el transcurso del desarrollo del software, el equipo de desarrollo propone el uso necesario de algunos artefactos de la metodología RUP que propiciarán un mejor entendimiento de los usuarios, concerniente al flujo de la aplicación y sus principales componentes.

Artefactos propuestos de la metodología de desarrollo RUP

- ✓ Diagrama de clases del diseño: es un tipo de diagrama estático que describe la estructura de un sistema y las relaciones entre los principales elementos, dígase clases e interfaces y las relaciones que existe entre ellos. El diagrama de clases le permite al cliente tener una buena perspectiva de cómo está estructurado el diseño de la herramienta, puesto que en XP, con el uso de las tarjetas CRC, no es apreciable de una forma visual las distintas interconexiones entre las clases, sino que se realiza por medio de tablas.

- ✓ Diagrama de despliegue: muestran las relaciones físicas entre los componentes del hardware y el software, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes del software. El diagrama de despliegue es uno de los diagramas no generado por la metodología XP. Permite al usuario ver de una forma general los principales componentes por los que está compuesta la aplicación.

1.6 Herramientas y tecnologías seleccionadas

Con el propósito de desarrollar un componente con la mayor calidad posible y que favorezca la integración con el NVP, se decidió por el equipo de desarrollo del NVP, mantener las tecnologías y herramientas seleccionadas. Estas son:

- ✓ Lenguaje de modelado, UML 8.0
- ✓ Herramienta de modelado, Visual Paradigm 5.0
- ✓ Lenguaje de programación, C++
- ✓ Entorno de desarrollo (IDE), Qt-Creator
- ✓ Estándar de conexión de la base de datos, Open Database Connectivity (ODBC)

1.6.1 Lenguaje de Modelación Unificado (UML).

UML (Unified Modeling Language), de sus siglas en español Lenguaje de Modelación Unificado, es un lenguaje gráfico para detallar, construir, visualizar y documentar las partes o artefactos (información que se utiliza o produce mediante un proceso de software). Pueden ser artefactos: un modelo, una descripción que comprende el desarrollo de software que se base en el enfoque Orientado a Objetos. (García, 2011)

UML es un lenguaje más expresivo, claro y uniforme que los anteriores, definidos para el diseño orientado a objetos, que no garantiza el éxito de los proyectos pero si mejora sustancialmente el desarrollo de los mismos, al permitir una nueva y fuerte integración entre las herramientas, los procesos y los dominios (Pressman, 2005).

1.6.2 Visual Paradigm.

Visual Paradigm para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado ayuda a una construcción más rápida de aplicaciones de calidad, mejores y a un menor coste, además de permitir el dibujo de todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación, así como una serie de tutoriales con demostraciones interactivas y proyectos. (García, 2011)

1.6.3 Lenguaje de programación para el desarrollo

Para la construcción de este proyecto se compararon numerosas alternativas de desarrollo. Desde usar un único lenguaje de alta eficiencia como C++ y basarse en bibliotecas 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. (Rodríguez, 2005)

Se optó como lenguaje de programación C++ con el objetivo de garantizar la compatibilidad del módulo y su integración con el NVP.

1.6.4 Elección del entorno de desarrollo

La selección de Qt se basa en que hace más fácil el desarrollo de aplicaciones de escritorio para programadores C++. Le permite a los desarrolladores crear aplicaciones de subprocesos múltiples que funcionan en todos los sistemas operativos, desde una única base de código fuente. También provee a los desarrolladores de herramientas fáciles e intuitivas para el desarrollo de aplicaciones avanzadas. Además incluye mecanismos directos para los subprocesos de comunicación interna, facilita y acelera la programación paralela e incluye funciones para la administración de subprocesos, objetos y datos. (García, 2011)

1.6.5 Selección del estándar de conexión de la base de datos

La conexión y el intercambio de información entre las aplicaciones y los gestores de base de datos (SGBD) han tenido su máximo exponente con el surgimiento del lenguaje de consulta estructurado (SQL). Gracias a este y al surgimiento de otras condiciones favorables, como su adopción por parte de un gran número de fabricantes de SGBD y a un mutuo consenso de estos en el diseño de los SGBD ha sido posible la creación y desarrollo de varios estándares para establecer dicha conexión. (García, 2011)

El Open Database Connectivity (Conectividad abierta a base de datos u ODBC) es un estándar de acceso a base de datos desarrollado por Microsoft Corporation, aunque se han desarrollado diferentes implementaciones del mismo, tanto libres como propietarias. Permite acceder a cualquier base de datos desde cualquier aplicación sin importar que sistema gestor de base de datos (SGBD) se utilice. Inserta una capa intermedia entre la aplicación y el SGBD, cuyo propósito es traducir las consultas de datos de la aplicación en comandos que el SGBD entienda. Para que esto funcione tanto la aplicación como el SGBD deben ser compatibles con ODBC, esto es que la aplicación debe ser capaz de producir comandos ODBC y el SGBD debe

ser capaz de responder a ellos. Desde la versión 2.0 el estándar soporta SAG y SQL. (Schultz, 2000) (Ripley, 2010)

Atendiendo a las anteriores características señaladas, se decidió utilizar ODBC para establecer la conexión a la base de datos del NVP, y de esta forma mantener la compatibilidad con el mismo. Por otra parte, es necesario destacar que el framework Qt utilizado para desarrollar el NVP, es compatible con dicho estándar.

1.7 Conclusiones parciales.

A lo largo de este capítulo se analizaron los principales conceptos asociados con el problema planteado. Por otra parte se realizó el estudio de algunas de las herramientas que existen a nivel nacional y mundial para la simulación de procesos en cascada, determinándose la necesidad de desarrollar un módulo de sincronización que permita la simulación de procesos en cascada en el Nodo Virtual de Procesos.

Se seleccionó la metodología de desarrollo Extreme Programing (XP) con la incorporación de los artefactos de RUP: diagrama de despliegue y diagrama de clases del diseño. Se hizo con el objetivo de propiciar un mejor entendimiento por los usuarios, referente al flujo de la aplicación y sus principales componentes.

Como herramientas y tecnologías para el análisis, diseño e implementación se mantienen las seleccionadas por el equipo de desarrollo del NVP para propiciar la integración y el correcto funcionamiento del sistema. Estas son:

- ✓ **Extreme Programming:** como metodología de desarrollo
- ✓ **Visual Paradigm:** como herramienta de modelado de datos
- ✓ **QT:** como entorno de desarrollo
- ✓ **C++:** como lenguaje de programación
- ✓ **Open Database Connectivity:** como estándar de conexión de la base de datos

CAPÍTULO 2: PLANIFICACIÓN, DISEÑO E IMPLEMENTACIÓN.

En este capítulo se detallan elementos relacionados con la descripción y análisis del sistema informático. Se identifican los requerimientos del software y se reflejan a través de las historias de usuario. Se describe el desarrollo del software a través del análisis y descripción de las tres primeras fases de la metodología XP: Planificación, Diseño, e Implementación. Se dará solución al problema y se generarán los artefactos correspondientes para obtener como resultado un producto con la calidad requerida.

2.1 Descripción del sistema

El sistema consiste en un módulo para la sincronización de procesos simples y concurrentes en cascada para un nodo virtual, que se encargue de la integración e interconexión de los distintos componentes necesarios a la hora de simular un proceso de forma local o remota. De esta forma se podrán sincronizar de manera sencilla, todos los elementos que integran la simulación, encontrando los fallos o los componentes que no estén bien configurados a la hora de simular un proceso. El sistema, a petición del usuario, distribuirá los diferentes componentes en diferentes nodos o computadoras para que se realice con más rapidez la simulación de un proceso determinado.

2.2 Planificación

La metodología de desarrollo de software XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores o gerentes. Se comienza recopilando **Historias de usuarios** (HU), las que sustituyen a los tradicionales **Casos de uso**. Una vez obtenidas las **Historias de usuarios**, los programadores evalúan rápidamente el tiempo de desarrollo de cada una. Si alguna de ellas tiene riesgos que no permiten establecer con certeza la complejidad del desarrollo, se realizan pequeños programas de prueba, para reducir estos riesgos. Una vez realizadas estas estimaciones, se organiza una reunión de planificación, con los diversos actores del proyecto (cliente, desarrolladores, gerentes), a los efectos de establecer un plan o cronograma de entregas (“Release Plan”) en los que todos estén de acuerdo. Una vez acordado este

cronograma, comienza una fase de iteraciones, en dónde en cada una de ellas se desarrolla, prueba e instala unas pocas **Historias de usuarios**. (Joskowicz 2008)

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Esta fase dura unos pocos días, identificándose además, el número y tamaño de las iteraciones, al igual que se plantean ajustes necesarios a la metodología según las características del proyecto.

2.2.1 Requerimientos del sistema.

Los requisitos funcionales definen el comportamiento interno del software y otras funcionalidades que muestran cómo las historias de usuario serán llevadas a la práctica.

En cambio, los requisitos no funcionales especifican criterios que pueden usarse para juzgar la operación de un sistema. Van a constituir las propiedades o cualidades que el producto debe tener para que llegue a ser confiable, usable y rápido. De acuerdo a las clasificaciones y descripciones planteadas se han definido los siguientes requerimientos.

2.2.1.1 Requisitos funcionales del sistema.

- ✓ **HU:** Crear planta.
    - RF1: Crear planta.
- ✓ **HU:** Crear controlador interno.
    - RF2: Crear controlador interno.
- ✓ **HU:** Crear controlador externo.
    - RF3: Crear controlador externo.
- ✓ **HU:** Crear elemento de medición.
    - RF4: Crear elemento de medición.
- ✓ **HU:** Crear modelo.
    - RF5: Crear modelo.

- ✓ **HU:** Cargar modelo.
    - RF6: Cargar modelo.
- ✓ **HU:** Guardar modelo.
    - RF7: Guardar modelo.
- ✓ **HU:** Validar modelo.
    - RF8: Validar que exista IP.
    - RF9: Validar que el puerto esté disponible.
- ✓ **HU:** Configurar servidor OPC.
    - RF10: Introducir datos para la configuración.
- ✓ **HU:** Configurar método numérico en la planta.
    - RF11: Seleccionar tipo de método numérico.
- ✓ **HU:** Sincronizar componentes.
    - RF12: Crear documento XML con los datos de la sincronización.
    - RF13: Enviar documento XML al nodo maestro.

2.3.1.2 Requisitos no funcionales del sistema.

- ✓ Requerimientos de Software:
    - Sistema Operativo Windows o Linux.
- ✓ Requerimientos de Hardware:
    - Memoria RAM de 512MB.
- ✓ Requerimientos en el diseño y la implementación:
    - Lenguaje de programación: C++
    - Entorno de desarrollo: QT.
    - Metodología de desarrollo: XP.

- ✓ Requerimientos de Seguridad:
    - Los cambios en el sistema sólo pueden ser realizados por los usuarios autorizados.
    - El sistema podrá ser usado en cualquier momento por todos los usuarios autorizados.
    - El administrador es el único que tiene el control total del sistema.
    - Para realizar una operación determinada el usuario deberá estar autenticado y podrán acceder al mismo de acuerdo a los roles asignados.
- ✓ Apariencia o interfaz externa:
    - El sistema tiene que ofrecer una interfaz amigable, fácil de operar.
    - El sistema tiene que mantener la línea de diseño establecida para la institución que mantiene la uniformidad y representatividad de la misma.
- ✓ Usabilidad:
    - En caso de desconexión de la red el sistema garantizará la disponibilidad de la información.
    - El software tendrá siempre la posibilidad de ayuda disponible para cualquier tipo de usuario.

2.2.2 Historias de usuario

Las **Historias de usuarios** (User stories) sustituyen a los documentos de especificación funcional, y a los **Casos de uso**. Estas historias son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra, en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles

necesarios. Las historias de usuarios deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia. (Joskowicz 2008) Las historias de usuario proporcionaran los detalles sobre la estimación del riesgo y cuánto tiempo conllevará la implementación de la misma.

Las historias de usuario deben ser:

- ✓ Valoradas por los clientes o usuarios: Los intereses de los clientes y de los usuarios no siempre coinciden, pero en todo caso, cada historia debe ser importante para alguno de ellos más que para el desarrollador.
- ✓ Estimables: Un resultado de la discusión de una historia de usuario es la estimación del tiempo que tomará completarla. Esto permite estimar el tiempo total del proyecto.
- ✓ Pequeñas: Las historias muy largas son difíciles de estimar e imponen restricciones sobre la planificación de un desarrollo iterativo. Generalmente se recomienda la consolidación de historias muy cortas en una sola historia.
- ✓ Verificables: Las historias de usuario cubren requerimientos funcionales, por lo que generalmente son verificables. Cuando sea posible, la verificación debe automatizarse, de manera que pueda ser verificada en cada entrega del proyecto.

Durante la fase de exploración se identificaron once HU, representando cada una las funcionalidades del sistema, las cuales se describen a continuación (Tablas 1-11):

Historia de Usuario	
Número: 1	Usuario: Todos los usuarios.
Nombre de historia: Crear planta.	
Prioridad en negocio: Alta.	Riesgo en desarrollo: Alta.
Puntos estimados: 1 semana.	Iteración asignada: 1
Programador responsable: Julio César Bellón Rodríguez.	
Descripción: La aplicación permite al usuario crear una planta, especificando para ello los	

parámetros necesarios para su creación.

Tabla 1. HU Crear planta.

Historia de Usuario	
Número: 2	Usuario: Todos los usuarios.
Nombre de historia: Crear controlador interno.	
Prioridad en negocio: Baja.	Riesgo en desarrollo: Baja.
Puntos estimados: 1 semana.	Iteración asignada: 2
Programador responsable: Nivaldo Faraco Díaz.	
Descripción: La aplicación permite al usuario crear controlador interno, especificando para ello el tipo de controlador interno que se va a utilizar.	

Tabla 2. HU Crear controlador interno.

Historia de Usuario	
Número: 3	Usuario: Todos los usuarios.
Nombre de historia: Crear controlador externo.	
Prioridad en negocio: Alta.	Riesgo en desarrollo: Alta.
Puntos estimados: 1 semana.	Iteración asignada: 1
Programador responsable: Julio César Bellón Rodríguez.	
Descripción: La aplicación permite al usuario crear controlador externo, especificando para ello la ubicación del controlador que no se encuentra dentro de la red de procesos.	

Tabla 3. HU Crear controlador externo.

Historia de Usuario	
Número: 4	Usuario: Todos los usuarios.
Nombre de historia: Crear elemento de medición.	

Prioridad en negocio: Baja.	Riesgo en desarrollo: Medio.
Puntos estimados: 1 semana.	Iteración asignada: 2.
Programador responsable: Nivaldo Faraco Díaz.	
Descripción: La aplicación permite al usuario crear elemento de medición y especificar el tipo de elemento de medición.	

Tabla 4. HU Crear elemento de medición.

Historia de Usuario	
Número: 5	Usuario: Todos los usuarios.
Nombre de historia: Crear modelo.	
Prioridad en negocio: Alta.	Riesgo en desarrollo: Baja.
Puntos estimados: 1 semana.	Iteración asignada: 1.
Programador responsable: Julio César Bellón Rodríguez.	
Descripción: La aplicación permite al usuario a partir de la creación de la planta, los controladores y los elementos de medición crear un modelo.	

Tabla 5. HU Crear modelo.

Historia de Usuario	
Número: 6	Usuario: Todos los usuarios.
Nombre de historia: Cargar modelo.	
Prioridad en negocio: Alta.	Riesgo en desarrollo: Baja.
Puntos estimados: 1 semana.	Iteración asignada: 1
Programador responsable: Nivaldo Faraco Díaz.	
Descripción: La aplicación permite al usuario cargar modelo para poder simular un proceso dado dicho modelo.	

Tabla 6. HU Cargar modelo.

Historia de Usuario	
Número: 7	Usuario: Todos los usuarios.
Nombre de historia: Guardar modelo.	
Prioridad en negocio: Alta.	Riesgo en desarrollo: Baja.
Puntos estimados: 1 semana.	Iteración asignada: 1
Programador responsable: Julio César Bellón Rodríguez.	
Descripción: La aplicación permite al usuario guardar modelo una vez que se haya creado.	

Tabla 7. HU Guardar modelo.

Historia de Usuario	
Número: 8	Usuario: Todos los usuarios.
Nombre de historia: Validar modelo.	
Prioridad en negocio: Medio.	Riesgo en desarrollo: Baja.
Puntos estimados: 1 semana.	Iteración asignada: 2.
Programador responsable: Nivaldo Faraco Díaz.	
Descripción: La aplicación permite probar que la red esté bien configurada y a su vez validar que los datos puedan llegar a su destino.	

Tabla 8. HU Validar modelo.

Historia de Usuario	
Número: 9	Usuario: Todos los usuarios.
Nombre de historia: Configurar servidor OPC.	
Prioridad en negocio: Baja.	Riesgo en desarrollo: Baja.
Puntos estimados: 1 semana.	Iteración asignada: 2.
Programador responsable: Julio César Bellón Rodríguez.	

Descripción: La aplicación permite al usuario seleccionar los parámetros de ajustes del servidor OPC.

Tabla 9. HU Configurar servidor OPC.

Historia de Usuario	
Número: 10	Usuario: Todos los usuarios.
Nombre de historia: Configurar método numérico en la planta.	
Prioridad en negocio: Media.	Riesgo en desarrollo: Alta.
Puntos estimados: 1 semana.	Iteración asignada: 2.
Programador responsable: Nivaldo Faraco Díaz.	
Descripción: La aplicación permite al usuario especificar el método numérico que se va a utilizar en la planta.	

Tabla 10. HU Configurar método numérico en la planta.

Historia de Usuario	
Número: 11	Usuario: Todos los usuarios.
Nombre de historia: Sincronizar componentes.	
Prioridad en negocio: Alta.	Riesgo en desarrollo: Alta.
Puntos estimados: 2 semanas.	Iteración asignada: 1.
Programador responsable: Julio César Bellón Rodríguez.	
Descripción: La aplicación permite organizar los componentes por los nodos que se le especificaron.	

Tabla 11. HU Sincronizar componentes.

2.2.3 Estimación de esfuerzos por historias de usuario

Se realizó una estimación del esfuerzo que costará implementar cada una de las historias de usuario identificadas, para el buen desarrollo del sistema propuesto, llegando a los resultados que se muestran a continuación (Tabla 12):

Historia de Usuario	Punto de estimación (semanas)
Crear planta.	1
Crear controlador interno.	1
Crear controlador externo.	1
Crear elemento de medición.	1
Crear modelo.	1
Cargar modelo.	1
Guardar modelo.	1
Validar modelo.	1
Configurar servidor OPC.	1
Configurar método numérico en la planta.	1
Sincronizar componentes.	2

Tabla 12. Estimación de esfuerzos por historias de usuario.

2.2.4 Plan de iteraciones

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación. Asimismo, para cada historia de usuario se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes

iteraciones no han afectado a las anteriores. Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir. (Joskowicz 2008) A continuación se describen cada una de las iteraciones propuestas, donde la duración total de iteraciones en días se obtiene a partir del esfuerzo en días estimado por el desarrollador para implementar cada historia de usuario:

Iteración # 1: Se tuvieron en cuenta aquellas HU de mayor importancia en cuanto a la funcionalidad que describen cada una, es decir, aquellas HU con prioridad alta, entre las cuales se encuentran las que se mencionan a continuación:

HU Crear planta.

HU Crear controlador externo.

HU Crear modelo.

HU Cargar modelo.

HU Guardar modelo.

HU Sincronizar componentes.

Iteración # 2: Se tuvieron en cuenta aquellas funcionalidades del sistema con menos prioridad, es decir, aquellas HU que presentan prioridad media o baja. Por otra parte en esta iteración se corregirán los errores encontrados en la iteración anterior, obteniéndose una nueva versión del sistema:

HU Crear controlador interno.

HU Crear elemento de medición.

HU Validar modelo.

HU Configurar servidor OPC.

HU Configurar método numérico en la planta.

2.2.5 Plan de duración de las iteraciones

En este plan (Tabla 13) se reflejan las HU que se desarrollarán en cada iteración, la duración de cada una y el orden en que serán implementadas en dichas iteraciones.

Nro. de iteración.	Historia de usuario a implementar.	Duración total de la iteración (semanas).
1	Crear planta. Crear controlador externo. Crear modelo. Cargar modelo. Guardar modelo. Sincronizar componentes.	7
2	Crear controlador interno. Crear elemento de medición. Validar modelo. Configurar servidor OPC. Configurar método numérico en la planta.	5

Tabla 13. Plan de duración de las iteraciones.

2.2.6 Plan de entregas

El cronograma de entregas establece qué historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas. Este cronograma será el resultado de una reunión entre todos los actores del proyecto (cliente, desarrolladores, etc.). XP denomina a esta reunión "Juego de planeamiento" ("Planning game"), pero puede denominarse de la manera que sea más apropiada al tipo de empresa y cliente (por ejemplo, Reunión de planeamiento, "Planning meeting" o "Planning workshop") Típicamente el cliente ordenará y agrupará según sus prioridades las historias de usuario. El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores. Luego de algunas iteraciones es recomendable realizar nuevamente una reunión con los actores del proyecto,

para evaluar nuevamente el plan de entregas y ajustarlo si es necesario. (Joskowicz 2008) A continuación (Tabla 14) se presenta el plan de entrega elaborado por el equipo de desarrollo, donde se reflejan las fechas de entregas para las primeras versiones de las historias de usuario:

Nro. de iteración.	Historia de usuario a implementar.	Fecha de entrega.
1	Crear planta. Crear controlador externo. Crear modelo. Cargar modelo. Guardar modelo. Sincronizar componentes.	15/05/2013
2	Crear controlador interno. Crear elemento de medición. Validar modelo. Configurar servidor OPC. Configurar método numérico en la planta.	22/05/2013

Tabla 14. Plan de entregas.

2.2.7 Arquitectura propuesta

La arquitectura es el esqueleto o base de una aplicación, donde se analiza la aplicación desde varios puntos de vista. En ella aparecen los artefactos más importantes y diferentes para establecer un esquema de cómo deben ser los próximos artefactos a construir. De obtenerse un artefacto demasiado diferente de los demás formaría parte de la arquitectura. (Jacobson, 2000).

Se escogió la arquitectura en n capas. Por las ventajas que presenta el uso de la misma relativas a la organización del código, la reutilización y la flexibilidad.

- **Capa de presentación:** Es la que interactúa directamente con el usuario. Captura la información de entrada (realiza un filtrado previo para comprobar que no hay errores de formato) y hace las peticiones a la capa inferior, mostrando al usuario la respuesta proveniente de esta. Únicamente se comunica con la capa de negocio.
- **Capa de negocio:** Está conformada por el subsistema Gestión, el cual se ajusta a los requisitos. Desde el punto de vista de diseño, esta capa es contenedora de las clases entidades y controladoras. Únicamente se comunica con la capa de Acceso a Datos.
- **Capa de Acceso a Datos:** Contiene clases que interactúan con los datos persistentes y permiten, utilizando los procedimientos de almacenamiento generados, realizar todas las operaciones de forma transparente para la capa de negocio.
- **Capa de Datos:** Es donde se encuentran todos los archivos persistentes que complementan la capa de negocio, conformados por los archivos de configuración del sistema que se almacenan en un fichero XML; y los elementos compilados relacionados con los modelos matemáticos de los procesos industriales que se utilizan en la simulación, y que se guardan como bibliotecas dinámicas.

2.3 Diseño

La metodología XP hace especial énfasis en los diseños simples y claros, para de esta forma lograr que sean de fácil entendimiento en la fase de implementación. En esta fase los elementos más importantes a tener en consideración son;

- simplicidad: un diseño simple se implementa más rápido que uno complejo.
- soluciones: pequeños programas de prueba para explorar diferentes soluciones.
- recodificación: consiste en escribir nuevamente parte del código de un programa, sin cambiar su funcionalidad.
- metáforas: una manera sencilla de explicar el propósito del proyecto, y guiar la estructura y arquitectura del mismo. Describe cómo debería funcionar el sistema, a través del diseño de las tarjetas CRC (Class, Responsibilities and Collaboration por sus

siglas en inglés). Es importante que el cliente y el grupo de desarrollo estén de acuerdo y compartan esta metáfora.

2.3.1 Aplicación de patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo del software y otros ámbitos referentes al diseño de interacción o interfaces, ya que brindan una solución a un problema de diseño. De los mismos ya debe estar comprobada su efectividad, resolviendo problemas similares en otras ocasiones y que sea aplicable a diferentes problemas de diseño en distintas circunstancias. Para la descripción de los objetos y clases definidos, se utilizarán algunos patrones de diseños con el objetivo de solucionar un problema de diseño general.

Los patrones GRASP (General Responsability Assigment Software Patterns) constituyen patrones generales de software para asignación de responsabilidades, y son considerados como una serie de buenas prácticas de aplicación recomendable en el diseño de software. Entre ellos se destacan los que se mencionan a continuación.

- ✓ **Experto en información:** Será utilizado debido a que es el principio básico de asignación de responsabilidades. Nos indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento).

- ✓ **Controlador:** Se asocia con operaciones del sistema y respuestas a sus eventos. El controlador delega en otros objetos el trabajo que se necesita hacer, pero coordina o controla la actividad. En otras palabras, el controlador no realiza estas actividades, sino que las delega en otras clases con las que mantiene un modelo de alta cohesión. Sirve como intermediario entre una clase interfaz y el algoritmo que la implementa ya que recibe los datos del usuario y los envía a las diversas clases.

- ✓ **Creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que conecte con el objeto producido en cualquier evento.

2.3.2 Tarjetas CRC

Las tarjetas CRC (Class, Responsibilities and Collaboration por sus siglas en inglés) identifican y organizan las clases orientadas a objetos que son relevantes para el incremento actual de software. Estas tarjetas representan objetos, para los cuales se especifican la clase a la que pertenece dicho objeto, las responsabilidades u objetivos que debe cumplir y las clases que colaboran con cada responsabilidad. A continuación (Tablas 15 y 16) se muestran las tarjetas CRC principales asociadas al sistema propuesto

Clase:

Responsabilidad	Clases relacionadas
Crear Sensor	DiagramItem DiagramScene Sensor Controladora
Crear Controlador	DiagramItem DiagramScene Controlador Controladora
Crear Planta	DiagramItem DiagramScene Planta Controladora

Eliminar Sensor	DiagramItem DiagramScene Controladora
Eliminar Controlador	DiagramItem DiagramScene Controladora
Eliminar Planta	DiagramItem DiagramScene Controladora
Cargar Modelo	Controladora
Eliminar Modelo	Controladora
Validar Modelo	Controladora
Configurar método numérico en la planta	Planta Controladora
Sincronizar componentes	Controladora

Tabla 15. Tarjeta CRC: Controladora

Responsabilidad	**Clases relacionadas**
Crear Sensor	DiagramItem DiagramScene Sensor Controladora
Crear Controlador	DiagramItem

	DiagramScene Controlador Controladora
Crear Planta	DiagramItem DiagramScene Planta Controladora
Eliminar Componentes	DiagramItem DiagramScene Componentes

Tabla 16. Tarjeta CRC: Scene

2.3.3 Artefactos propuestos por RUP

Diagrama de clases del diseño

Los diagramas de clases del diseño son los diagramas principales en el flujo de trabajo análisis y diseño. En cada uno se especifica la estructura de clases del sistema y sus relaciones. Durante el análisis del sistema, los diagramas se desarrollan buscando una solución ideal. Durante el diseño, se modifican esos diagramas para satisfacer los detalles de las implementaciones. (García, 2011). En la Figura 2 se muestra el diagrama de clases del diseño propuesto.

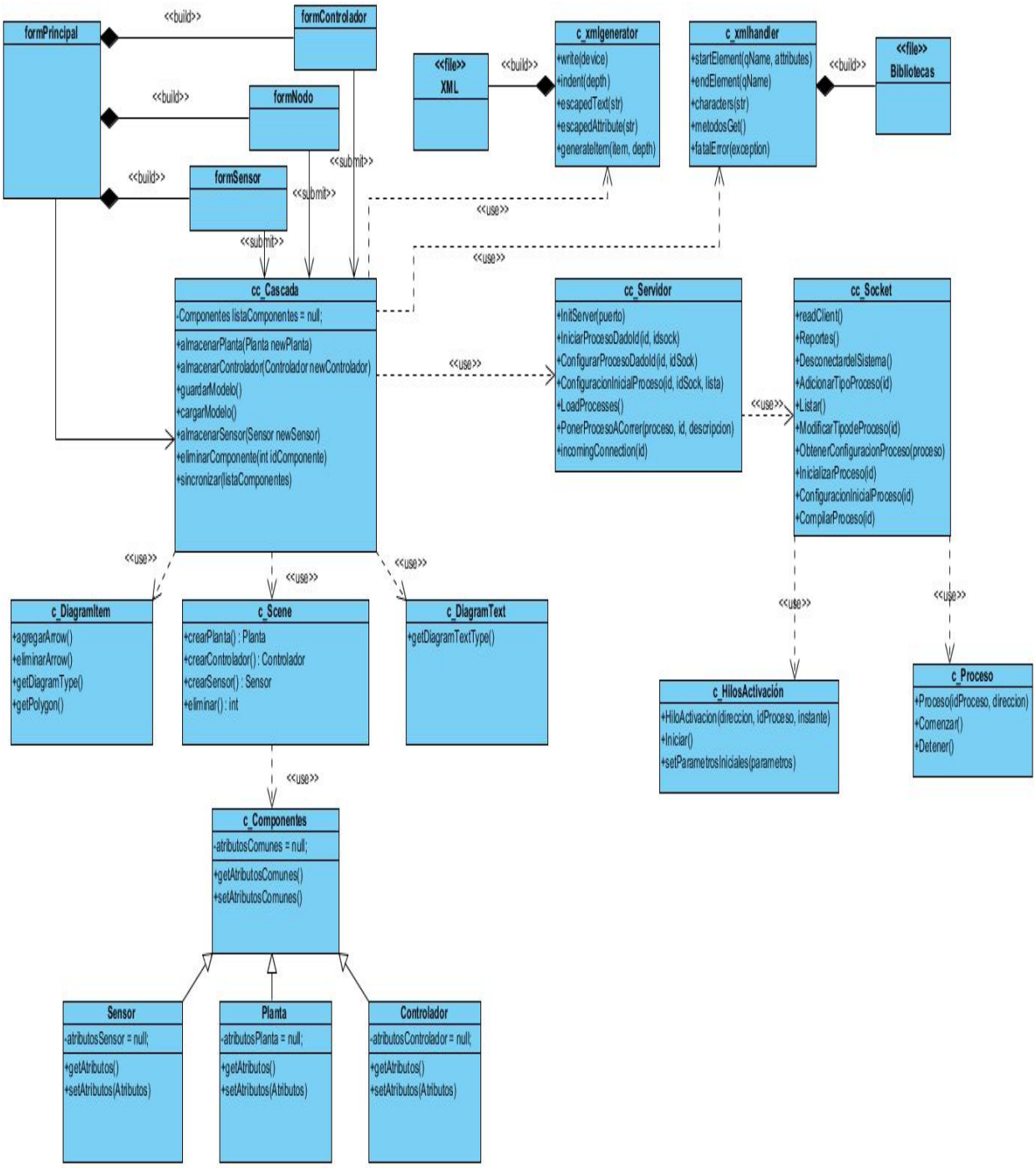


Fig 2. Diagrama de clases del diseño

Diagrama de despliegue

Los Diagramas de Despliegue muestran las relaciones físicas entre los componentes del hardware y el software, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes del software. Un Diagrama de Despliegue es un grafo de nodos, unidos por conexiones de comunicación, donde un nodo puede contener instancias de componentes. Es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuyen las funcionalidades entre los nodos de cómputo. (García, 2011). En la Figura 3 se puede observar el diagrama de despliegue correspondiente.

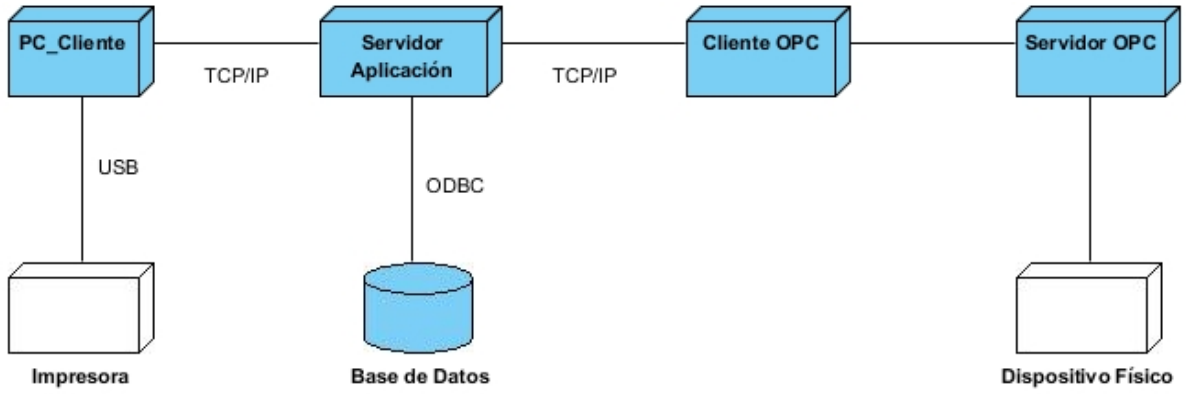


Fig 3. Diagrama de despliegue

2.4 Implementación

La implementación del sistema es la parte más importante dentro del desarrollo del proyecto en la metodología XP. Esta propone una serie de prácticas que sirven y benefician el desarrollo del mismo, tales como:

- ✓ el desarrollo de las iteraciones según el plan de iteraciones
- ✓ entrega en iteraciones pequeñas
- ✓ un diseño simple
- ✓ las pruebas continuas
- ✓ el código es revisado y discutido mientras se escribe

- ✓ propiedad del código compartida

- ✓ hacer entregas frecuentes
- ✓ refactorización del código
- ✓ corrección de todos los errores antes de añadir una nueva funcionalidad

2.4.1 Tareas de programación por historia de usuarios.

Las tareas de programación se definen con el objetivo de proporcionar una guía para un mejor desarrollo y cumplimiento de cada una de las tareas que serán desarrolladas por los programadores.

Las tareas de programación son actividades que los programadores conocen que el sistema debe hacer. Deben ser estimables, y poder ser implementadas entre uno y tres días ideales. La mayoría de las tareas de programación se derivan directamente de las historias de usuario. (Beck., K. Extreme Programming Explained. Embrace Change. s.l.: Addison-Wesley, 1999.)

En la Tabla 17 se relacionan las HU con sus respectivas tareas de ingeniería.

Historias de Usuario	Tareas de Ingeniería
Crear planta.	Permitir seleccionar la planta. Permitir seleccionar las características de dicha planta.
Crear controlador interno.	Permitir seleccionar el controlador interno. Permitir seleccionar las características de dicho controlador interno.
Crear controlador externo.	Permitir seleccionar el controlador externo. Permitir seleccionar la ubicación de dicho

	controlador externo.
Crear elemento de medición.	Permitir seleccionar el elemento de medición. Permitir seleccionar el tipo de elemento de medición.
Crear modelo.	Permitir a partir del elemento seleccionado crear modelo.
Cargar modelo.	Mostrar la interfaz de los modelos creados. Permitir seleccionar el modelo a cargar.
Guardar modelo.	Permitir guardar el modelo.
Validar modelo.	Verificar que la red esté configurada adecuadamente. Validar que los datos puedan llegar a su destino.
Configurar servidor OPC.	Permitir seleccionar los parámetros de ajustes del servidor OPC.
Configurar método numérico en la planta.	Permitir seleccionar el método numérico.
Sincronizar componentes.	Permitir organizar los componentes por los nodos que se le especificaron.

Tabla 17. Distribución de las tareas por cada Historia de Usuario

En las Tablas de la 18 a la 34 se describen las tareas de cada una de las HU.

HU: Crear planta.

	Tarea de ingeniería
Número Tarea: 1	**Número Historia de Usuario:** 1.
Nombre Tarea: Permitir seleccionar la planta.	
Tipo Tarea: Desarrollo	**Puntos estimados (días):** 3.
Fecha inicio: 04/02/2013	**Fecha fin:** 07/02/2013
Programador responsable: Julio César Bellón Rodríguez.	
Descripción: El sistema permite al usuario seleccionar el componente planta para describir la dinámica del proceso.	

Tabla 18: Descripción de la tarea de ingeniería #1.

	Tarea de ingeniería
Número Tarea: 2	**Número Historia de Usuario:** 1.
Nombre Tarea: Permitir seleccionar las características de dicha planta.	
Tipo Tarea: Desarrollo	**Puntos estimados (días):** 3.
Fecha inicio: 08/02/2013	**Fecha fin:** 11/02/2013
Programador responsable: Julio César Bellón Rodríguez.	
Descripción: El sistema permite al usuario seleccionar las características correspondientes de dicha planta.	

Tabla 19: Descripción de la tarea de ingeniería #2.

HU: Crear controlador interno.

	Tarea de ingeniería
Número Tarea: 3	**Número Historia de Usuario:** 2.
Nombre Tarea: Permitir seleccionar el controlador interno.	

Tipo Tarea: Desarrollo	**Puntos estimados (días):** 3.
Fecha inicio: 15/02/2013	**Fecha fin:** 18/02/2013
Programador responsable: Nivaldo Faraco Díaz.	
Descripción: El sistema permite al usuario seleccionar el controlador interno, entre los preestablecidos por el NVP	

Tabla 20: Descripción de la tarea de ingeniería #3.

		Tarea de ingeniería
Número Tarea: 4	**Número Historia de Usuario:** 2.	
Nombre Tarea: Permitir seleccionar las características de dicho controlador interno.		
Tipo Tarea: Desarrollo		**Puntos estimados (días):** 3.
Fecha inicio: 20/02/2013		**Fecha fin:** 23/02/2013
Programador responsable: Nivaldo Faraco Díaz.		
Descripción: El sistema permite al usuario seleccionar las características correspondientes de dicho controlador.		

Tabla 21: Descripción de la tarea de ingeniería #4.

HU: Crear controlador externo.

		Tarea de ingeniería
Número Tarea: 5	**Número Historia de Usuario:** 3.	
Nombre Tarea: Permitir seleccionar el controlador externo.		
Tipo Tarea: Desarrollo		**Puntos estimados (días):** 3.
Fecha inicio: 25/02/2013		**Fecha fin:** 28/02/2013
Programador responsable: Julio César Bellón Rodríguez.		
Descripción: El sistema permite al usuario seleccionar el controlador externo, el cual no se encuentra en la red de procesos.		

Tabla 22: Descripción de la tarea de ingeniería #5.

Tarea de ingeniería

Número Tarea: 6	**Número Historia de Usuario:** 3.
Nombre Tarea: Permitir seleccionar la ubicación de dicho controlador externo.	
Tipo Tarea: Desarrollo	**Puntos estimados (días):** 3.
Fecha inicio: 01/03/2013	**Fecha fin:** 04/03/2013
Programador responsable: Julio César Bellón Rodríguez.	
Descripción: El sistema permite al usuario seleccionar la ubicación del controlador externo.	

Tabla 23: Descripción de la tarea de ingeniería #6.

HU: Crear elemento de medición.

	Tarea de ingeniería
Número Tarea: 7.	**Número Historia de Usuario:** 4.
Nombre Tarea: Permitir seleccionar el elemento de medición.	
Tipo Tarea: Desarrollo	**Puntos estimados (días):** 3.
Fecha inicio: 06/03/2013	**Fecha fin:** 09/03/2013
Programador responsable: Nivaldo Faraco Díaz.	
Descripción: El sistema permite al usuario seleccionar el elemento de medición.	

Tabla 24: Descripción de la tarea de ingeniería #7.

	Tarea de ingeniería
Número Tarea: 8.	**Número Historia de Usuario:** 4.
Nombre Tarea: Permitir seleccionar el tipo de elemento de medición.	
Tipo Tarea: Desarrollo	**Puntos estimados (días):** 3.
Fecha inicio: 11/03/2013	**Fecha fin:** 15/03/2013
Programador responsable: Nivaldo Faraco Díaz.	
Descripción: El sistema permite al usuario, una vez seleccionado el elemento de medición, especificar el tipo de elemento.	

Tabla 25: Descripción de la tarea de ingeniería #8.

HU: Crear modelo.

	Tarea de ingeniería
Número Tarea: 9	**Número Historia de Usuario:** 5.
Nombre Tarea: Permitir a partir del elemento seleccionado crear modelo.	
Tipo Tarea: Desarrollo	**Puntos estimados (días):** 6.
Fecha inicio: 17/03/2013	**Fecha fin:** 23/03/2013
Programador responsable: Julio César Bellón Rodríguez.	
Descripción: El sistema permite al usuario seleccionar modelo.	

Tabla 26: Descripción de la tarea de ingeniería #9.

HU: Cargar modelo.

	Tarea de ingeniería
Número Tarea: 10	**Número Historia de Usuario:** 6.
Nombre Tarea: Mostrar la interfaz de los modelos creados.	
Tipo Tarea: Desarrollo	**Puntos estimados (días):** 3.
Fecha inicio: 25/03/2013	**Fecha fin:**28/03/2013
Programador responsable: Nivaldo Faraco Díaz.	
Descripción: El sistema muestra una interfaz con los modelos existentes.	

Tabla 27: Descripción de la tarea de ingeniería #10.

	Tarea de ingeniería
Número Tarea: 11.	**Número Historia de Usuario:** 6.
Nombre Tarea: Permitir seleccionar el modelo a cargar.	
Tipo Tarea: Desarrollo	**Puntos estimados (días):** 3.
Fecha inicio: 02/04/2013	**Fecha fin:** 05/04/2013
Programador responsable: Nivaldo Faraco Díaz.	
Descripción: El sistema permite al usuario cargar modelo para reutilizar un modelo existente y llevar a cabo su simulación y sincronización.	

Tabla 28: Descripción de la tarea de ingeniería #11.

HU: Guardar modelo.

	Tarea de ingeniería
Número Tarea: 12.	**Número Historia de Usuario:** 7.
Nombre Tarea: Permitir guardar el modelo.	
Tipo Tarea: Desarrollo	**Puntos estimados (días):** 6.
Fecha inicio: 06/04/2013	**Fecha fin:** 12/04/2013
Programador responsable: Julio César Bellón Rodríguez.	
Descripción: El sistema permite guardar el modelo una vez se hayan definido todos los parámetros necesarios.	

Tabla 29: Descripción de la tarea de ingeniería #12.

HU: Validar modelo.

	Tarea de ingeniería
Número Tarea: 13.	**Número Historia de Usuario:** 8.
Nombre Tarea: Verificar que la red esté configurada adecuadamente.	
Tipo Tarea: Desarrollo	**Puntos estimados (días):** 3.
Fecha inicio: 14/04/2013	**Fecha fin:** 17/04/2013
Programador responsable: Nivaldo Faraco Díaz.	
Descripción: Se verifica que la red esté configurada adecuadamente, comprobando que las direcciones ip que se le han asignado a los diferentes componentes se encuentren en la red.	

Tabla 30: Descripción de la tarea de ingeniería #13.

	Tarea de ingeniería
Número Tarea: 14	**Número Historia de Usuario:** 8.
Nombre Tarea: Validar que los datos puedan llegar a su destino.	
Tipo Tarea: Desarrollo	**Puntos estimados (días):** 3.
Fecha inicio: 20/04/2013	**Fecha fin:** 23/04/2013
Programador responsable: Nivaldo Faraco Díaz.	

Descripción: Se verifica que los puertos dados a cada uno de los componentes se encuentren habilitados para el envío de datos.

Tabla 31: Descripción de la tarea de ingeniería #14.

HU: Configurar servidor OPC.

	Tarea de ingeniería
Número Tarea: 15.	**Número Historia de Usuario:** 9.
Nombre Tarea: Permitir seleccionar los parámetros de ajustes del servidor OPC.	
Tipo Tarea: Desarrollo	**Puntos estimados (días):** 6.
Fecha inicio: 25/04/2013	**Fecha fin:** 01/05/2013
Programador responsable: Julio César Bellón Rodríguez.	
Descripción: El sistema permite al usuario seleccionar los parámetros de ajustes del servidor OPC.	

Tabla 32: Descripción de la tarea de ingeniería #15.

HU: Configurar método numérico en la planta.

	Tarea de ingeniería
Número Tarea: 16.	**Número Historia de Usuario:** 10.
Nombre Tarea: Permitir seleccionar el método numérico.	
Tipo Tarea: Desarrollo	**Puntos estimados (días):** 6.
Fecha inicio: 05/05/2013	**Fecha fin:** 11/05/2013
Programador responsable: Nivaldo Faraco Díaz.	
Descripción: El sistema permite al usuario seleccionar el método numérico.	

Tabla 33: Descripción de la tarea de ingeniería #16.

HU: Sincronizar componentes.

	Tarea de ingeniería
Número Tarea: 17	**Número Historia de Usuario:** 11.

Nombre Tarea: Permitir organizar los componentes por los nodos que se le especificaron.	
Tipo Tarea: Desarrollo	**Puntos estimados (días):** 12.
Fecha inicio: 03/05/2013	**Fecha fin:** 15/05/2013
Programador responsable: Julio César Bellón Rodríguez.	
Descripción: El sistema permite sincronizar componentes.	

Tabla 34: Descripción de la tarea de ingeniería #17

2.5 Conclusiones parciales

Con la realización de las fases de Planificación, Diseño e Implementación correspondientemente se obtuvieron:

Se identificaron once Historias de Usuario:

- ✓ Crear planta
- ✓ Crear controlador interno
- ✓ Crear controlador externo
- ✓ Crear elemento de medición
- ✓ Crear modelo
- ✓ Cargar modelo
- ✓ Guardar modelo
- ✓ Validar modelo
- ✓ Configurar servidor OPC
- ✓ Configurar método numérico en la planta
- ✓ Sincronizar componentes

Se estimó el total del tiempo de desarrollo en cada iteración en base a la estimación de esfuerzos por cada Historia de Usuario.

Se diseñaron las tarjetas CRC, con el objetivo de definir las entidades que están involucradas en el funcionamiento del sistema.

Se definieron las tareas de programación por cada Historia de Usuario.

CAPÍTULO 3: VERIFICACIÓN DE LA SOLUCIÓN.

El proceso de pruebas constituye otro de los elementos claves de la metodología XP. Esta metodología propone que se realicen pruebas constantemente. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad y evitar efectos no deseados a la hora de realizar modificaciones y refactorizaciones.

3.1 Métricas para la especificación de requisitos

Un elemento clave de cualquier proceso de ingeniería es la medición. Las medidas se emplean para entender mejor los atributos de los modelos que se crean. Pero se emplean fundamentalmente para valorar la calidad de los productos de ingeniería o de los sistemas que se construyen. (Pressman, 2005)

Para medir la calidad de la especificación de requisitos de software, se propone una lista de características, algunas de estas son: especificidad (ausencia de ambigüedad), compleción, corrección, comprensión, y la capacidad de verificación. (Overmyer, 1993)

Para determinar la ***especificidad*** (ausencia de ambigüedad) de los requisitos se sugiere una métrica basada en la consistencia de la interpretación de los revisores para cada requisito: (Overmyer, 1993)

$$Q1 = \frac{Rii}{Rt} = \frac{26}{26} = 1$$

Rii : Número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.

Rt : Total de los requisitos.

La ***compleción*** de los requisitos funcionales puede determinarse calculando la relación:

$$Q2 = \frac{na}{na + nb} = \frac{26}{26 + 0} = 1$$

na : Número de requisitos funcionales completos.

nb : Número de requisitos funcionales pobremente especificados.

Una especificación se considera correcta cuando cada requisito contenido en ella represente una característica que el sistema debe poseer. La ***corrección*** de los requisitos se define usando la siguiente ecuación:

$$Q3 = \frac{Rc}{Rc + Rnv} = \frac{26}{26 + 0} = 1$$

Rc : Número de requisitos que se han validado como correctos.

Rnv : Número de requisitos que no se han validado como correctos todavía.

La ***comprensión*** de los requisitos se determina a partir de la relación que se muestra a continuación.

$$Q4 = \frac{Rbc}{Rt} = \frac{26}{26} = 1$$

Rbc : Número de requisitos que todos los revisores entienden.

Para realizar el proceso de medición de la especificación de requisitos practicándole estas métricas, fue necesario calcular el número total de requisitos. Al realizar la suma de los requisitos funcionales y no funcionales se obtuvieron 26 requisitos en total. Luego de aplicadas estas métricas se concluye, que la especificación cuenta con la calidad requerida, ya que a medida que los resultados de las métricas se acercan a uno se alcanza mayor calidad, lo cual contribuye en gran medida a lograr un mejor entendimiento entre clientes y desarrolladores.

Todos los requisitos fueron interpretados de la misma forma, ya que los revisores involucrados en el proceso coincidieron en todas las interpretaciones, cuyo resultado garantiza la ausencia de ambigüedad. Además el alto grado de compleción, a pesar de ser un factor difícil de medir, su valor indica que todos los requisitos que el software debe cumplir han sido incluidos y especificados.

Por otra parte la corrección y la comprensión, obtuvieron un valor óptimo, demostrando que se está en presencia de una especificación que fue bien comprendida por los revisores y que todos los requisitos representan una capacidad o cualidad que debe estar presente en el sistema a construir.

3.2 Métricas para validar diseño

Las métricas del software son una medida cuantitativa de evaluar la calidad de los atributos internos de un sistema. Se emplean con el objetivo de llevar el control de la calidad del producto que se está desarrollando, evaluar la efectividad del proceso y mejorar la calidad del trabajo. Las métricas proporcionan los conocimientos necesarios para crear modelos efectivos de análisis y diseño, un código sólido y pruebas exhaustivas (Pressman R. S., 2005).

3.2.1. Tamaño Operacional de Clase (TOC)

Consiste en medir el tamaño general de una clase tomando el valor de la cantidad de operaciones. Si el resultado obtenido indica valores grandes, significa que la clase posee un alto grado de responsabilidad, debido a esto se reducirá la reutilización, se hará mucho más difícil la implementación y la realización de pruebas de dicha clase. Por tanto mientras menor sea el valor para el TOC se hará mucho más fácil la reutilización de dicha clase dentro del sistema.

La métrica TOC fue aplicada a las nueve clases de la aplicación. En la Figura 4 se muestra la cantidad de procedimientos presentes en las clases agrupados en intervalos, así como en las Figuras de la 5 a la 7 se muestran los resultados de la evaluación de la métrica TOC en los atributos responsabilidad, complejidad y reutilización respectivamente.

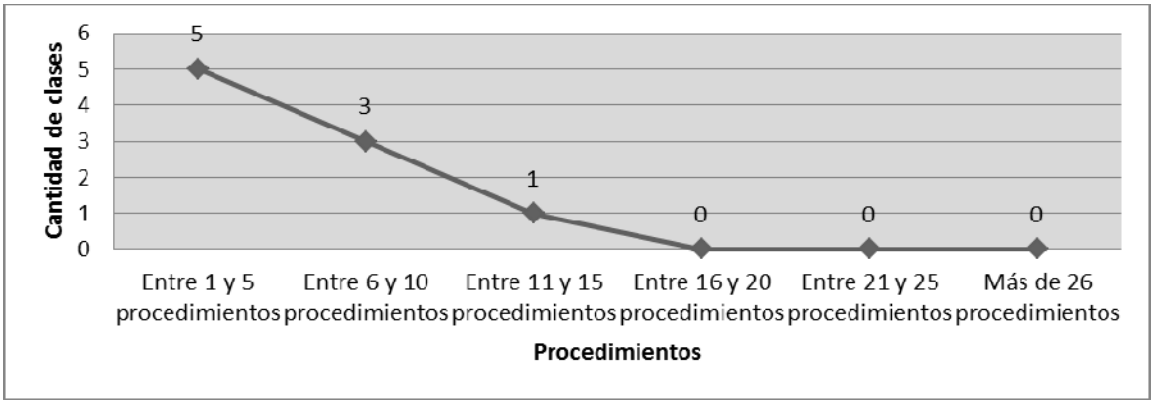


Figura 4.Representación de cantidad de procedimientos por clases

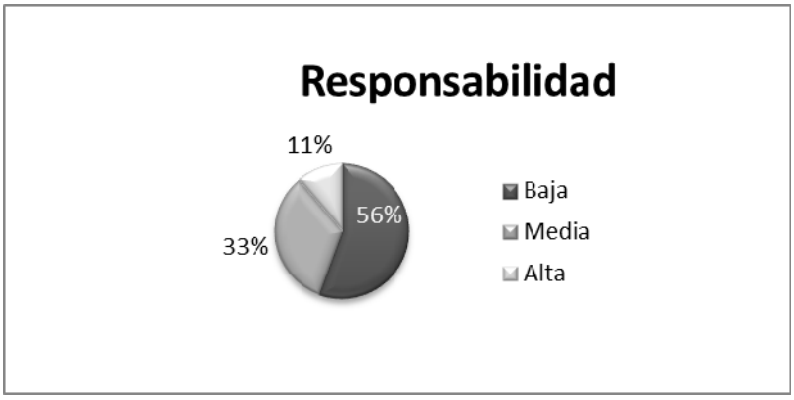


Figura 5. Representación de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad

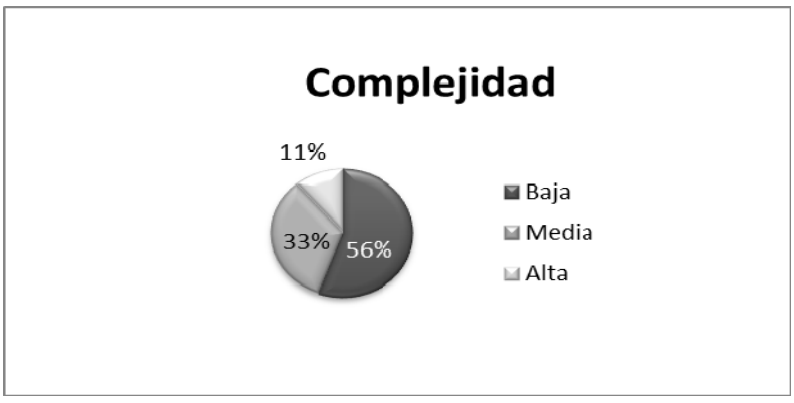


Figura 6. Representación de los resultados de la evaluación de la métrica TOC en el atributo Complejidad

Figura 7. Representación de los resultados de la evaluación de la métrica TOC en el atributo Reutilización

Los resultados arrojados por la métrica TOC fueron positivos ya que evidencian que de las nueve clases solo el once porciento presenta una alta complejidad, responsabilidad y baja reutilización. Lo que es favorable, pues implica que el sistema cumple con los parámetros de diseño demandados por las exigentes prestaciones del NVP.

3.2.2. Relaciones entre Clases (RC)

Las relaciones entre clases están dadas por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas. La métrica de diseño RC fue aplicada a las nueve clases de la aplicación obteniéndose los siguientes resultados. En la Figura 8 se muestran las cantidades de clases según sus dependencias y en las Figuras de la 9 a la 12 se muestran los resultados de la evaluación de la métrica RC en los atributos acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización respectivamente.

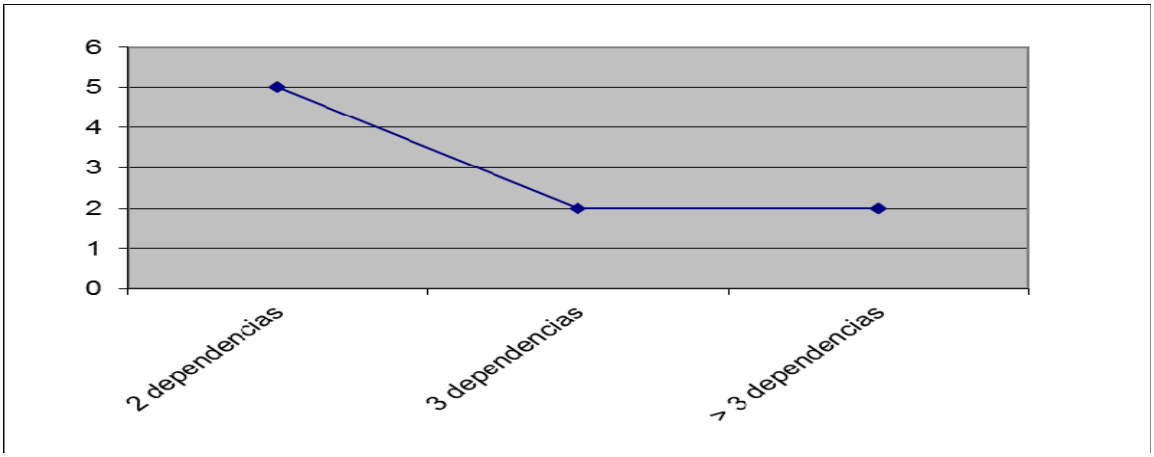


Figura 8. Cantidad de clases por dependencias

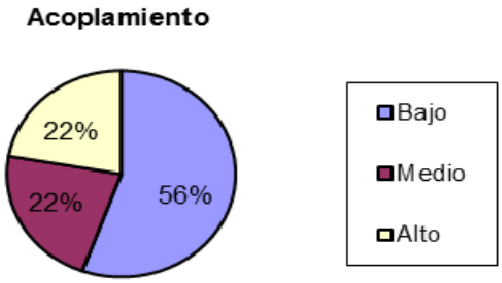


Figura 9. Representación de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento

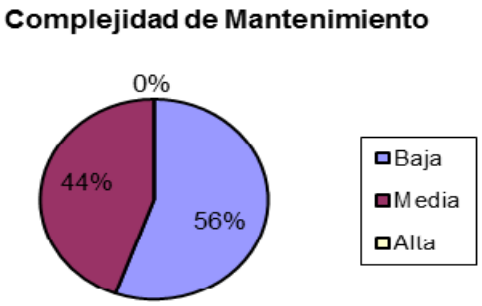


Figura 10. Representación de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento

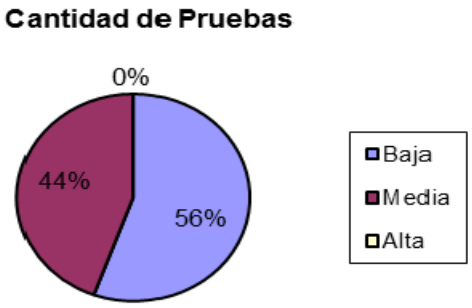


Figura 11. Representación de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.

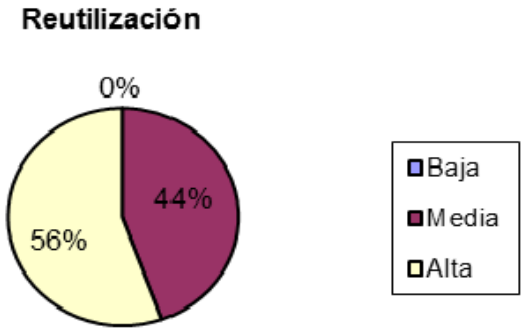


Figura 12. Representación de los resultados de la evaluación de la métrica RC en el atributo Reutilización

Los resultados arrojados por la métrica RC fueron positivos ya que el sistema presenta un bajo acoplamiento, baja complejidad de mantenimiento, baja cantidad de pruebas y una alta reutilización . Lo que es propicio, pues indica que las relaciones entre las clases están en correspondencia con los requerimientos de la aplicación NVP.

3.3 Pruebas de caja blanca

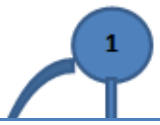
Las pruebas de caja blanca realizan un seguimiento del código fuente según se van ejecutando los casos de prueba, de manera que se determinan de manera concreta las instrucciones, bloques, etc, en los que existen errores. Cuando se pasan casos de prueba al programa que se está probando, es conveniente 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 del camino básico: Le permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental, y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba desarrollados garantizarán que durante la prueba se ejecute por lo menos una vez cada sentencia del programa. (García, 2011)

Para la aplicación de estas pruebas se seleccionó el método toolBarra. Este método es el encargado de mostrar y ocultar la tabla de propiedades genéricas de cada componente

- ✓ **Técnica del Camino Básico:** Para la aplicación de esta técnica se siguieron una serie de pasos lógicos:

  - Se construye el grafo de flujo a partir del código fuente del método a probar que se observa en la figura once.
  - Se determina la complejidad ciclomática V(G) del grafo G. Para calcular la complejidad ciclomática hay tres formas:
    - V(G) = Aristas - Nodos + 2.
    - V(G) = Nodos de predicado + 1.
    - V(G) = Número de regiones del grafo.
  - A partir del valor de la complejidad ciclomática se obtiene el número de caminos independientes, que nos dan un valor límite para el número de pruebas que tenemos que diseñar (Ver Figura 13).

Complejidad Ciclomática	
V(G) = (A - N)+2	

V(G) = (5 – 5) + 2	
V(G) = 2	
Posibles Caminos	
1. 1-2-5	
2. 1-3-4-5	

Figura 13: Grafo de Flujo.

A cada camino obtenido de la prueba de camino básico se le realiza un caso de prueba.

- ✓ Caso de prueba para el Camino básico #1:
  - Condición de ejecución: Que la variable de entrada sea verdadera.
  - Resultados esperados: Se muestra la tabla con las propiedades del componente.
- ✓ Caso de prueba para el Camino básico #2:
  - Condición de ejecución: Que la variable de entrada no sea verdadera.
  - Resultados esperados: Se cierra la tabla de propiedades del componente.

3.4 Pruebas de caja negra

Las pruebas de Caja Negra parten de los requisitos funcionales, a muy alto nivel, para diseñar pruebas que se aplican sobre el sistema sin necesidad de conocer cómo está construido por dentro. Estas permiten determinar si la función está siendo desempeñada correctamente por el sistema bajo prueba, aplicando sobre el sistema un conjunto de datos de entrada y observando las salidas que se producen. Estas pruebas permiten encontrar: (Pressman, 2005)

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a las base de datos externas.
- ✓ Errores de rendimiento.

- ✓ Errores de inicialización y terminación.

Para preparar los casos de prueba hacen falta un número de datos que ayuden a la ejecución de los casos, y que permitan que el sistema se ejecute en todas sus variantes. Pueden ser datos válidos o inválidos para el programa según lo que se desea, si es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa se ejecute bien. Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están: (Pressman, 2005)

- ✓ **Técnica de la Partición de Equivalencia**: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- ✓ **Técnica del Análisis de Valores Límites**: esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- ✓ **Técnica de Grafos de Causa-Efecto**: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Una de las pruebas a desarrollar, utilizando la técnica de caja negra, es la de “Validación”, que es realizada sobre la interfaz del sistema. Para realizar dichas pruebas se diseñaron casos de prueba para cada historia de usuario del sistema con el objetivo de encontrar errores en las funcionalidades implementadas.

Descripción del caso de prueba Editar Procesos de la Planta

La funcionalidad comienza cuando el Administrador del sistema desea configurar la Planta, luego de la correcta selección de la funcionalidad en el menú de edición se procede a entrar al sistema los elementos de configuración. La funcionalidad termina cuando el sistema guarda definitivamente la operación correspondiente (Tablas 35 y 36).

Condiciones de Ejecución:

- El usuario del sistema debe seleccionar la opción "Editor de Proceso de la Planta".

Escenario	Variable 1 (Editor de Fórmulas)	Variable 2 (Nombre)	Variable 3 (Descripción)	Variable 4 (Clasificación)	Variable 5 (Vectores)	Variable 6 (Fórmula)	Respuesta del sistema
EP 1.1: Configurar mensajería.	V	V	V	V	V	V	El sistema almacena los elementos de configuración. □Se muestra un mensaje de información "Se creo la Planta correctamente".
	Checked Unchecked	Proceso_1	Primer Proceso	Electrónico	A - B - C - D	A*B-(C-D)	
EP 1.2: Campos vacíos.	V	V	V	V	V	V	El sistema muestra un mensaje de error para que llene los campos vacíos "Inserte nombre del proceso".
	Checked Unchecked	(Vacío)	(Vacío)	Electrónico	A - B - C - D	A*B-(C-D)	
EP 1.3 Campos incorrectos	V	V	V	V	V	F	El sistema muestra un mensaje de error "Fórmula Incorrecta".
	Checked Unchecked	Proceso_1	Primer Proceso	Electrónico	A - B - C - D	A*B-()C-D	
EP 1.4: Cancelar operación	V	V	V	V	V	V	Se cierra la ventana
	N/A	N/A	N/A	N/A	N/A	N/A	

Tabla 35: Escenarios a probar en el HU Editar Procesos de la Planta.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Editor de Fórmulas	CheckBox	N/A	Activa o desactiva el editor de fórmulas.
2	Nombre	Campo de texto	No	Nombre del proceso.
3	Descripción	Campo de texto	Si	Descripción del proceso.
4	Clasificación	ComboBox	No	Clasificación
5	Vectores	ListView	No	Vectores que tiene el proceso.
6	Fórmula	Label	Si	Fórmula creada.

Tabla 36: Descripción de las variables.

3.5 Conclusiones parciales

Luego de finalizada la fase de Pruebas se concluye:

- ✓ El desarrollo de las pruebas aseguró la ejecución correcta de la solución en todo el período de desarrollo.

- ✓ Las pruebas de aceptación realizadas se efectuaron de manera exitosa demostrando que el sistema reúne las condiciones para dar solución al problema planteado cumpliendo con los parámetros de calidad establecidos.

CONCLUSIONES GENERALES

Al finalizar el presente trabajo investigativo se arribó a las siguientes conclusiones:

El estudio realizado de las características de la aplicación NVP existente y de la modelación y simulación de procesos industriales, permitió constatar la necesidad que tiene la aplicación de simular procesos en cascadas.

Con el estudio realizado, se identificaron varias herramientas que simulan procesos, pero ninguna de ellas específicamente simula procesos industriales de relativa complejidad, de forma genérica y en cascada, confirmando la necesidad de un módulo de sincronización de procesos que permita al NVP la simulación de procesos industriales en cascada.

El estudio de los requisitos permitió que se desarrollaran un total de trece funcionalidades, reflejadas en once historias de usuarios.

Se diseñaron un total de tres tablas CRC, en las cuales quedó plasmado el diseño de la aplicación.

Se aplicaron un total de tres patrones de diseño, demostrando la alta flexibilidad de las clases y la robustez del sistema.

Para validar la solución se realizaron varias iteraciones de pruebas de caja blanca y caja negra, auxiliadas por métricas para comprobar la calidad del producto y estimar la efectividad de los procesos, mostrando resultados satisfactorios.

Con el desarrollo de este trabajo se le dio cumplimiento al objetivo general de la investigación propuesta: desarrollar una solución informática que permita la simulación de procesos en cascada al Nodo Virtual de Procesos, partiendo de la última versión del mismo.

.

RECOMENDACIONES

Una vez concluido el presente trabajo se recomienda:

- ✓ Realizar pruebas de carga en el NVP con herramientas profesionales, para comprobar su capacidad de soportar las peticiones concurrentes
- ✓ Valorar la inclusión de clústeres para adicionar potencialidades de procesamiento al NVP.
- ✓ Implantar el sistema para enriquecer el desarrollo de prácticas en la especialidad de Ingeniería Automática.

BIBLIOGRAFÍA

1. Gámez, Yalice Batista. (2010). Herramienta interactiva para la enseñanza en la carrera de Ingeniería Automática: Nodo Virtual de Procesos. La Habana: s.n., 2010.

2. Maier S., Herrscher D. (2005).” On Node Virtualization for Scalable Network Emulation”, University of Stuttgart, Institute of Parallel and Distributed Systems (IPVS), Germany.

3. Hernández, D., Moreno, R. (2009). Solución Informática “Nodo Virtual de Procesos” para la carrera Ingeniería Automática, tesis en opción al grado de ingeniero en ciencias informáticas, Universidad de las Ciencias Informáticas, La Habana, Cuba.

4. Escobar, M., Ortiz, L. (2008). “Análisis y Diseño de un Nodo Virtual de Procesos.”, tesis en opción al grado de ingeniero en ciencias informáticas, tesis en opción al grado de ingeniero en ciencias informáticas, Universidad de las Ciencias Informáticas, La Habana, Cuba.

5. Hosseinzaman, A Bargiela, A. (1995). ADA’s Virtual Node base d Water System. Department of Computing Nottingham Trent University Burton Street: Reino Unido.

6. Molino N., Bao Z. (2004).”A Virtual Node Algorithm for Changing Mesh Topology During Simulation”, Stanford University.

7. Miyachi T., Chinen K. (2006). “StarBED and SprinOS: Large scale general purpose network testbed and supporting software”, Japan Advanced Institute of Science and Technology, Ishikawa, Japan.

8. Cuevas L. (2007). “Sistema informático de gestión para actividades docentes y extra docentes en la facultad 3. Rol Analista de Sistemas”, tesis en opción al grado de ingeniero en ciencias informáticas, Universidad de las Ciencias Informáticas, La Habana, Cuba.

9. López Barrio, C. (2005). “Metodología de Desarrollo: Programación Extrema”. Disponible en http://www- lsi.die.upm.es/~carreras/ISSE/programacion_extrema_1.x2.pdf. Consultado: 17 de febrero del 2013.

10. Jacobson, I., Booch, G. and Rumbaugh, J. (2004). “El Proceso Unificado de Desarrollo de Software”. La Habana: Félix Varela.

11. García Guevara, Julio J., Seija Rodríguez, Yelina (2011). "Rediseño e implementación de una herramienta interactiva para la simulación de procesos industriales: Nodo Virtual de Procesos.", tesis en opción al grado de ingeniero en ciencias informáticas, tesis en opción al grado de ingeniero en ciencias informáticas, Universidad de las Ciencias Informáticas, La Habana, Cuba. Pressman, R. (2005). "Ingeniería del Software. Un enfoque práctico". La Habana: Félix Varela.

12. SlideShare. (2009). Visual Paradigm for UML. [Online] 2009. Disponible en: http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml. Consultado: 22 de enero del 2013.

13. Rodríguez F, Lucas Quintero A, Cabello. (2005). Simulación de procesos de información y criptografía cuántica. Departamento Física Aplicada I, Sevilla: s.n., 2005.

14. Schultz, Rick. (2000). Using the Oracle ODBC Drivers with Third Party Products. 2000.

15. Ripley, Brian.(2010). ODBC Connectivity. Department of Statistics, University of Oxford: s.n., 2010.

16. Felicioni, Flavia E., (2003). Simulación y Diseño Asistido de Accionamientos Eléctricos Controlados.

17. Letelier, Patricio and Penadés, Mª Carmen. Metodologías ágiles para el desarrollo de software. s.l. : Universidad Politécnica de Valencia.

18. Joskowicz, José (2008). "Reglas y Prácticas en eXtreme Programming", Doctorado de Ingeniería telemática, Universidad de Vigo, España.

19. Alonso Delgado, Yanet., González Mulet, Yunet (2007). "Software para la Simulación de Sistemas.", tesis en opción al grado de ingeniero en ciencias informáticas, tesis en opción al grado de ingeniero en ciencias informáticas, Universidad de las Ciencias Informáticas, La Habana, Cuba.

20. Nokia Qt Reference Documentation (Open Source Edition). – (2009).

21. Overmyer, Davis S. (1993). *Identifying and measuring quality in software requirements specification.* California: Los Alamitos: s.n., 1993.

22. Taylor H, James (2001). "Modeling & Simulation of Dynamic Systems -- a Tutorial", plenary lecture, Proc. IMACS/IFAC Fourth International Symposium on Mathematical Modelling and Simulation in Agricultural and Bio-Industries, Haifa, Israel.

23. Montero, C., Molleda, J. (2009) “Control en Cascada.”

24. Lozano H, J., Mesa V. (2008). “Simulación multidisciplinar de sistemas de control con ECOSIMPRO”. Departamento de Ingeniería de Sistemas y Automática. Escuela Politécnica Superior de Algeciras.

25. Aurel Systems. (2013). Chemical Engineering Software & Services. [Online] 2013. Disponible en: http://www.aurelsystems.com/. Consultado: 15 de enero del 2013.