Universidad de las Ciencias Informáticas
Facultad 3

Título: Análisis y Diseño de un Nodo Virtual de Procesos.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores: Mailen Edith Escobar Pompa

Leydis Andis Ortiz Azahares

Tutor: Ing. Yalice Gámez Batista

Co-tutor: Ing. Carlos Yasmiany Hidalgo García

Asesores: Ing. Sergio Jesús García de la Puente

Ing. José Ramón Fernández Pérez

Ciudad de la Habana

Junio 2008

El hombre puede hacer de sí mismo muchas cosas
producto de su propio esfuerzo físico y espiritual,
el que se proponga cultivar la virtud la cultiva,
el que se proponga alcanzar los más altos niveles
de conocimientos los alcanza.
Fidel Castro Ruz.

DECLARACIÓN DE AUTORÍA

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

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

Mailen Edith Escobar Pompa

_____

Firma del Autor

Leydis Andis Ortiz Azaharez

_____

Firma del Autor

Yalice Gámez Batista

_____

Firma del Tutor

Agradecimientos

Un especial agradecimiento a mi mami por estar siempre presente y apoyarme en todas y cada una de las decisiones, siempre lista para dar un buen consejo, por los innumerables sacrificios que puede hacer una madre por un hijo y por brindar tanto amor incondicionalmente.

A mi papá por confiar y estar de mi lado siempre que lo necesité, por saberme proteger, cuidar y malcriar, sé que todos los días te sentirás orgulloso de tener a tus dos hijos con una profesión justa y digna.

A mi hermano que ha sido para mí la mano dura por la cual hoy puedo levantarme luego de caer y seguir adelante, siempre vas a ser un ejemplo para mí.

A mi sobrino que lo quiero mucho y al que viene en camino muy pronto que estoy deseosa por conocerlo.

A Yuleidy por saber tener mucha paciencia, gracias por tu apoyo y cariño.

A mis tías y tíos todos: Dora, Anneida, Annelis, Annolvis, Adalis, Juanita, Elaida, Elenita, Marcelino, mis tíos Chino y Choli y a sus respectivos hijos que siempre han contribuido en alguna etapa de mi vida para mi formación y por preocuparse tanto.

A mi abuelita que aunque hoy no está conmigo siempre sueño con su mano sobre mi cabeza, sé que si me viera estaría más que orgullosa porque aunque somos bastante nietos siempre nos quiso a todos por igual.

A Linnet por estar siempre pendiente y pelearme en los momentos exactos para que recapacite, y a May y Susa que siempre estuvieron de mi lado en los momentos más difíciles y supieron comprenderme.

A mis amigos que me han acompañado en este y todos los tiempos, siempre los recordaré, aunque no los pueda mencionar por falta de espacio.

A nuestra tutora Yalice, Carlos: a quien le debo mucho de la persona que soy hoy, a Sergio y José Ramón por su ayuda desinteresada.

A la persona tan grande que ha creado esta obra hermosa que es la Batalla de Ideas y levantó ante los ojos del imperio una universidad que es un verdadero sueño.

Leydis

Agradecimientos

A Su por sus alones de moños tan necesarios. Por ser arquitecta, amiga y hermana, y estar siempre disponible para participar en las Tormentas de Ideas.
A mi tutora Yalice por darme la oportunidad de ser parte de este trabajo.
A mi cotutor Carlos porque a pesar de la distancia siempre estuvo presente, por su preocupación y su apoyo incondicional.
A mis asesores Sergito y Jose, por todo el trabajo extra que les di, por sacar tiempo de donde no había para aclarar mis dudas, por sus consejos y su apoyo.
A Arturo, Yankiel y Leandro, quienes sin apenas conocerme me ayudaron.
A Luis Jiménez, amigo en todo momento.
A toda mi patrulla, Arianna, Susana, Leydis, Lisset, Daimi, Yoandro, Ainier y Luis, por todos los momentos buenos, por quererme y aceptarme como soy.
A mi grupo, mi antiguo 3107, porque en el aprendí el verdadero concepto de compañerismo, solidaridad y unidad.
A todos los que de una forma u otra han contribuido en mi educación y mi carrera.
A todos los amigos.

Mailen

Dedicatoria

…A mi mamá, mi papá, mi hermano...

Leydis.

…A mis padres, porque sin ellos no hubiera llegado hasta aquí, este es nuestro sueño, y este éxito es tanto mío como de ustedes.

…A mi bebe Eddy, por quereme tanto. Espero lo mismo de ti.

…A mi Pepy, por estar siempre, a pesar de la distancia y las ocupaciones, por ser mi luz y mi fuerza, por la fe que siempre ha tenido en mí.

…A mis suegros, por quererme como una hija, por todo su apoyo en este tiempo.

…A mi familia.

…A todos los amigos.

May.

Resumen

Gracias a la introducción de las Tecnologías de la Información y las Comunicaciones se ha perseguido que la mayoría de la población sea capaz de socializar su aprendizaje de forma general y así contribuir a la llamada informatización de la sociedad. Es por ello que el trabajo continuo debe convertirse en un reto inigualable para muchos y así poder vincular todas las esferas sociales y todo ello se debe al auge que ha tenido el surgimiento de Internet.

El caudal de servicios que brinda, posibilita la creación de modelos, procesos y aplicaciones lo que hace que se pueda llevar el control informático de toda una serie de actividades que simulan actividades reales o experimentales. Para poder llevar a cabo estas acciones de forma virtual, se propone la realización de un proyecto que se basa en la realización de un Nodo Virtual de Procesos Industriales con el cual se persiguen objetivos fundamentales tales como: implementar modelos de distintos procesos industriales, agrupar estos modelos por tipos dando la posibilidad de realizar simulaciones con los mismos o probando aplicaciones en tiempo real. La puesta en marcha de este nodo se basa en la tecnología cliente-servidor, donde aplicaciones clientes podrán conectarse al nodo utilizando un protocolo establecido.

Para modelar el análisis y diseño de un nodo virtual de procesos industriales se hará uso de la metodología RUP, lenguaje UML y la herramienta Visual Paradigm, con un Gestor de Base de Datos MySQL.

PALABRAS CLAVE

"Nodo Virtual, Simulación de Procesos, Procesos Industriales, Análisis, Diseño"

TABLA DE CONTENIDOS

Introducción

Actualmente las Tecnologías Informáticas son muy usadas a nivel mundial sobre todo para el desarrollo de software. El trabajo continuo para con las nuevas tecnologías y la computación, parece ser la tarea de primer orden que exige que se convierta en un reto para muchos. Lograr que nuestro entorno quede completamente inmerso en el mundo de los ordenadores y manipularlo empleando la menor fuerza humana posible, sigue siendo una utopía, pues todavía queda mucho por explorar y aprender, pero la situación actual está propiciando que lleguemos a tener una sociedad completamente informatizada.

Es difícil encontrar en nuestros días alguna esfera de la vida en la que no estén vinculadas las Tecnologías de la Información y las Comunicaciones (TIC). Ellas están presentes en la economía, los negocios, la cultura, la sociedad, la salud, la educación. Un factor influyente en la proliferación de su uso y explotación fue el surgimiento y desarrollo de Internet. Con su integración, se fueron desarrollando aplicaciones que, utilizando la red como medio de comunicación, son capaces de brindar cualquier tipo de servicio desde y hacia cualquier parte del mundo. La Modelación y Simulación Computacional (MCS) es un ejemplo de servicio que constituye un eslabón fundamental para llevar a cabo proyectos que requieran de su uso.

La MCS se ha venido desarrollando desde la década de los 40. Un ejemplo de ello es la investigación realizada para la creación y construcción de las primeras bombas atómicas, pero adquieren auge e inusitado impulso a partir de los años 80 en que se ha visto un notable aumento en el desarrollo de la industria.

Estos modelos computacionales han sido empleados de forma más intensa en unos campos que en otros, cuestión perfectamente lógica, ya que el MSC es el único método de investigación y análisis disponible para aquellos fenómenos y procesos, que no pueden ser reproducidos o experimentados en la realidad, o que su costo o peligrosidad hacen de la experimentación real un hecho poco practicable, conjuntamente con la connotación de obtener alguna solución plausible. [2]

En la actualidad existen herramientas informáticas que permiten realizar simulaciones de procesos, pero todas ellas tienen limitaciones y restricciones que las hacen parcialmente ineficientes, por lo que para solucionar estos problemas se han implementado otras aplicaciones que hacen uso de nodos virtuales en los cuales se simulan los procesos.

Estados Unidos, Inglaterra y Alemania son los países que mayor progreso han logrado en el desarrollo de este tipo de software: la Universidad de Stuttgart en Alemania

desarrolló una aplicación llamada Network Emulation Testbed con el objetivo de simular redes. Por otra parte, en Inglaterra, la Universidad de Nottingham Trent creó una aplicación para la simulación de sistemas de agua, y el laboratorio Nacional de Lawrence Berkeley en EUA desarrolló un simulador para pozos de petróleo todas basadas en nodos virtuales.

Se conoce del desarrollo de Sistemas de Control Industrial (ICS) en nuestro país, por ejemplo en el año 1991 fue implementado por ingenieros de la Unión del Níquel en la provincia de Holguín un ICS conocido como EROS que se usó en 27 plantas de diferentes tipos y en la actualidad, se está desarrollando en la Universidad de las Ciencias Informáticas (UCI) el sistema SCADA (Supervisory Control and Data Adquisition) para el trabajo con imágenes utilizando una plataforma virtual de procesamiento. También se conoce de la existencia en dicha universidad del Proyecto SIMPRO el cual se dedica al desarrollo de simuladores.

Por todo lo anteriormente expresado podemos concluir que tanto a nivel mundial como en nuestro país no se tienen grandes experiencias en el desarrollo de nodos virtuales para la simulación de procesos; a esto se une el hecho de que la carencia de una aplicación que realice estas actividades, trae consigo problemas para el desarrollo y comprensión de los contenidos docentes en la carrera de ingeniería automática, pues en esta se imparten una serie de contenidos que proveen a los estudiantes de toda la teoría necesaria para ejercer como profesionales de la rama, sin embargo no siempre estos conocimientos son fundamentados desde el punto de vista práctico con herramientas que fortalezcan la comprensión de los mismos.

Tomando todo lo anterior podemos definir nuestra **situación problémica** consistente en que los estudiantes de la carrera de Ingeniería Automática no cuentan con las herramientas adecuadas para fortalecer los conocimientos adquiridos durante la carrera desde el punto de vista práctico, por lo que tienen la necesidad de una nueva herramienta que de manera remota o local les permita interactuar con diferentes procesos industriales y con los que puedan comprobar sus estrategias de control.

Según lo descrito anteriormente se puede llegar al siguiente **problema**:

¿Cómo realizar el análisis y diseño de un nodo virtual de procesos que permita que se facilite y mejore la calidad del aprendizaje de los estudiantes de la carrera de Ingeniería Automática?

Por consiguiente el **objeto de la investigación** sería, el modelado de procesos por tipos, centrando la atención en el análisis y diseño de un nodo virtual de procesos, identificado como campo de acción de la investigación.

Lo que conlleva a la siguiente **hipótesis**:

Si se realiza un correcto análisis y diseño para un NVP, se podrá implementar el mismo, de modo que los estudiantes de la carrera de Ingeniería Automática puedan adquirir y fortalecer los contenidos asociados con procesos.

A partir de ésta se formulan los **objetivos** de este trabajo, los cuales se precisan a continuación:

Objetivo General

- Realizar el análisis y diseño de un Nodo Virtual que simule procesos industriales.

Objetivos Específicos

- Caracterizar el entorno actual teniendo en cuenta otros softwares similares y la bibliografía al respecto.

Tarea 1: Realizar una búsqueda con el fin de extraer las características de las aplicaciones o sistemas similares.

Tarea 2: Analizar toda la bibliografía existente sobre el tema a investigar.

- Realizar el levantamiento de requisitos.

Tarea 1: Utilizar métodos que permitan obtener toda la información necesaria acerca de lo que se quiere realizar.

Tarea 2: Confeccionar las plantillas correspondientes.

- Obtener artefactos del análisis para los casos de uso críticos.

Tarea 1: Realizar el diagrama de CU del sistema a partir de los requisitos funcionales.

Tarea 2: Realizar el diagrama de análisis tomando como punto de partida lo obtenido en el paso anterior.

Tarea 3: Confeccionar los diagramas de colaboración.

- Obtener el diseño de los casos de uso críticos.

Tarea 1: Seleccionar patrones de diseño a utilizar.

Tarea 2: Realizar el diseño de acuerdo a los artefactos obtenidos en el análisis.

Con el propósito de desarrollar las tareas planteadas, se utilizaron los **métodos** de investigación siguientes:

Teóricos:

La presente investigación se fundamenta en el **método dialéctico materialista** como método basado en el conocimiento de las leyes más generales del desarrollo. Se potencia la autorregulación de la actividad cognoscitiva de cada usuario lo que posibilita el logro de nuevos resultados a través de las conexiones y contradicciones manifestadas en los procesos. La aplicación, Nodo Virtual de Procesos, que se

propone en unidad, interrelación e independencia con todos sus módulos facilitará desplegar un proceso de interacción de los usuarios con los diferentes procesos.

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

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

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

Empíricos

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

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

Métodos Particulares:

Las entrevistas y la toma de criterios de expertos:

Propiciaron recoger las opiniones que sobre el tema de investigación poseen distintos especialistas conocedores de la rama tratada.

La tormenta de ideas:

Herramienta utilizada para posibilitar la generación de un elevado número de ideas, por parte de un grupo, y la presentación ordenada de éstas.

Como aporte esperado de esta investigación:

Aporte Teórico: Diseño del modelo de análisis de la aplicación Nodo Virtual de Procesos y obtención de los casos de uso.

Para la confección del trabajo de diploma que se propone desarrollar, se van a realizar 4 capítulos, estructurados de la siguiente forma:

En el capítulo 1 se realizará una breve reseña de los nodos virtuales y autómatas. Se llevará a cabo un estudio sobre las metodologías y herramientas a utilizar, para establecer una comparación que nos permita escoger las más adecuadas. Además se estudiará los principales conceptos y características de los flujos de análisis y diseño.

En el capitulo 2 se comienza a construir el sistema que se desea desarrollar, elaborando para ello los artefactos correspondientes al primer flujo de trabajo que se analiza: Requerimientos. Aquí se realizará la presentación de los procesos del negocio a través de un Modelo de Dominio, se hará el levantamiento de los Requisitos Funcionales y No Funcionales del Sistema, se definirán los actores y casos de uso del sistema, los cuáles se integrarán en el Diagrama General de Casos de Uso, y además se harán las descripciones por cada caso de uso para una mayor comprensión de las acciones en cada uno.

En el capítulo 3 se continúa con el flujo de Análisis y Diseño del sistema. Aquí se van a representar lo artefactos correspondientes a este flujo como son: los diagramas de clases del análisis para cada caso de uso, los diagramas de interacción correspondientes, el modelo de diseño, los diagramas de clases del diseño por subsistemas de diseño, la realización de los casos de uso, y los diagramas de interacción por casos de uso. Todos estos artefactos servirán para que se tenga un mejor entendimiento del funcionamiento del sistema a construir.

Finalmente en el capítulo 4 se realizará la validación de los resultados. Mediante la aplicación de métricas se medirá la calidad del trabajo realizado.

Capítulo 1: Fundamentación Teórica

Introducción

Como una de las propuestas de solución a la problemática planteada en la introducción de este trabajo se pensó en un Nodo Virtual, ya que permite correr diferentes instancias del software en un único nodo (nodo físico) y cada instancia del software trabaja en un entorno de ejecución independiente (nodo virtual). Por tanto en este capitulo se pretende abordar los aspectos y conceptos relacionados con el Nodo Virtual. Se hará un análisis de las aplicaciones existentes, de las tecnologías y tendencias actuales utilizadas en productos de este tipo. Se hará un estudio de las metodologías de desarrollo existentes así como de los patrones de diseños, pretendiendo dejar sentadas las bases teóricas para un correcto análisis y diseño del software. Se explicarán las herramientas escogidas para dar solución a la problemática propuesta.

1.1 Nodo Virtual de Simulación de Procesos.

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

Esta filosofía de trabajo permite la simulación simultánea de diferentes procesos sin que interfieran uno con otros. Se denomina simulación a la reproducción del comportamiento dinámico de un sistema real en base a un sistema con el fin de llegar a conclusiones aplicables al mundo real. Simulación es también el proceso de diseñar el modelo de un sistema real a través de experimentos basados en computadoras para describir, explicar y predecir el comportamiento del sistema real.

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:

- El parámetro más importante es minimizar los gastos de virtualización para preservar los recursos destinados al 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. [22]

Para ello se establece como nodo a aquella estructura a la cual se interconectan varios elementos. Por otra parte se conoce como proceso industrial a una operación que transforma los aportes de material, energía e información en productos, como parte de un sistema de producción industrial.
Para el establecimiento de pruebas se necesita de una fracción de recursos de un nodo de prueba y un número de aplicaciones dirigidas a dispositivos de pocos recursos de forma tal que se puedan ejecutar varios procesos en este nodo de prueba más conocido como nodo físico o pnodo y que cada proceso provea un entorno de ejecución del mismo de manera separada, a esto se le conoce como nodo virtual o vnodo.

1.2 Análisis de otras soluciones existentes

Actualmente existen herramientas informáticas que permiten realizar simulaciones de procesos (en tiempo real o no) pero todas estas 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, simulan los procesos o permiten probar aplicaciones reales, nunca las dos prestaciones.
Para la simulación de redes, en la Universidad de Stuttgart en Alemania, instituto especializado en sistemas distribuidos y paralelos, se creó la aplicación “Network Emulation Testbed”. Esta aplicación va dirigida a simular un entorno de redes configurable que permita reproducir un escenario real de cientos de nodos en comunicación. De esta manera posibilita medir de manera comparativa el comportamiento de una aplicación en diferentes entornos de redes o de varias aplicaciones en un mismo entorno de red. Esta aplicación es utilizada tanto en redes tradicionales como en un entorno de redes ad hoc inalámbrica. Permite a partir de una red de 64 computadoras traducirlo a un escenario de más de 1920 nodos. [22]
La principal limitante de esta aplicación está en que es diseñada para simular redes por lo que no es posible utilizarla para la simulación de procesos que tienen una dinámica un tanto más compleja y variada.
En la Universidad de Nottingham Trent, se creó un software para la simulación de sistemas de agua. Surge por la necesidad que tenían en la industria del agua de adquirir y almacenar datos de estaciones remotas para la inspección ingenieril. A medida que fue creciendo el sistema fue además incrementándose la acumulación de

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

Esta aplicación no es la solución pues no está concebida para simular procesos independientes, sino que simula subprocesos de un sistema general, para luego integrarlo en el proceso general como un todo. [15]

En el laboratorio Nacional de Lawrence Berkeley en EUA, se enfrentaban al problema de tratar las condiciones límites en los pozos de petróleo en el momento de formular y codificar un simulador numérico multifase de la reserva. Esto se debe a la complejidad de las ecuaciones diferenciales que gobiernan el flujo de la superficie que son una mezcla de tipo hiperbólica-parabólica que provocan problemas de convergencia computacional. El método convencional de tratamiento de pozos geotermales o de reservas de petróleo no es lo suficientemente riguroso y puede dar lugar a soluciones físicamente incorrectas para las diferentes capas del pozo. Por esta razón se utilizó el método de nodos virtuales donde a cada capa se le asignó un nodo virtual cuyo tratamiento está dado en dependencia de las características de la capa a la que represente para los cálculos del flujo. La solución en el pozo se obtendrá de resolver las ecuaciones del balance de masa para el nodo del pozo. De esta manera se provee de un procedimiento numérico eficiente y consistente de manera física para el manejo de los problemas del flujo en los pozos. [31]

A similitud del anterior este simulador fue diseñado para el estudio de las diferentes capas de un pozo petrolífero para luego integrarlo en el modelo del pozo.

En la Universidad de Stanford se creó un algoritmo de nodos virtuales para el trabajo con imágenes en tres dimensiones. Cuando un elemento es fragmentado para crear varias replicas del elemento y se le asigna una porción real del material a cada réplica. De esto resultan elementos que contienen material real y regiones vacías. El material faltante está contenido en otra u otras copias. El algoritmo de nodo virtual determina automáticamente el número de réplicas así como la asignación de material para cada

una. Provee además los grados de libertad requeridos para simular el material parcial o completamente fragmentado en una imagen consistente con la geometría. Aprovecha las posibilidades de la simulación de una geometría compleja con una simple mezcla. Permite manejar tanto cuerpos rígidos como colisiones en la simulación. [25]

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

1.3 Autómatas

El autómata programable industrial (API) nació como solución al control de circuitos complejos de automatización. Por lo tanto, se puede decir que un API no es más que un aparato electrónico que sustituye los circuitos auxiliares o de mandos en los sistemas automáticos. A el se conectan los captadores por una parte y los actuadores por la otra. [3]

Los autómatas leen la información del entorno a través de detectores, y en función de lo que se le programe pondrá en marcha las salidas. El Sistema de Entradas y Salidas recoge la información del proceso controlado (entradas) y envía las acciones de control del mismo (salidas). Los dispositivos de entrada pueden ser pulsadores, interruptores, termostatos, presostatos, etc. y los de salida pueden ser pilotos, indicadores, relés, bombillas, contactores, válvulas, etc.

De manera general los autómatas están formados por dos partes fundamentales [29]:

- La unidad central de procesos (CPU)
- Sistema de entrada y salida (E/S)

La CPU es la encargada del control del autómata, realiza tanto el control interno como externo. Esta tiene prácticamente los mismos elementos que la CPU de una computadora: un microprocesador, una memoria RAM , una memoria ROM, unos circuitos electrónicos que sirven para unir estos componentes con unas conexiones externas, que lo conectan con el mundo exterior [1]. La interpretación de las instrucciones del programa de memoria se divide en dos bloques: la de lectura (ROM) y la de lectura y escritura (RAM).

1.4 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. [21]

Las metodologías de desarrollo de software constituyen un factor importante para lograr productos con calidad en el tiempo requerido. Actualmente existen múltiples metodologías, todas con muchas características comunes, pero que comparten también significativas diferencias, por lo que se recomienda hacer un estudio de estas para realizar una correcta selección, garantizando así el éxito del proyecto de software.

Las ventajas de tener una metodología son:

Mejora de los procesos de desarrollo.

- Todas las personas del proyecto trabajan bajo un marco común.
- Estandarización de conceptos, actividades y nomenclaturas.
- Actividades de desarrollo apoyadas por procedimientos y guías.
- Resultados del desarrollo predecibles.
- Uso de herramientas de Ingeniería Software.
- Planificación de actividades en base a un conjunto de tareas definidas y a la experiencia en otros proyectos.
- Recopilación de mejores prácticas para proyectos futuros.

Mejora de los productos.

- Se asegura que los productos cumplen con los objetivos de calidad propuestos.
- Detección temprana de errores.
- Se garantiza la trazabilidad de los productos a lo largo del desarrollo.

Mejora de las relaciones con el cliente.

- El cliente percibe el orden en nuestros procesos.
- Facilita al cliente el seguimiento de la evolución del Proyecto.
- Se establecen mecanismos para asegurar que los productos desarrollados cumplen con las expectativas del cliente. [13]

A continuación se realizará un pequeño estudio de posibles metodologías a usar en el desarrollo del sistema a construir.

1.4.1 Programación Extrema

La Programación Extrema (XP), es una de las metodologías de desarrollo de software con más éxito en la actualidad. Es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado. Se utiliza en proyectos con equipos de desarrollo pequeños y con plazos de entrega corto. La metodología consiste en una programación rápida o extrema. Una particularidad es tener como miembro del equipo al usuario final. Esta metodología tiene las siguientes características:

- **Pruebas Unitarias:** Se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.
- **Refabricación:** Se basa 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.

Lo fundamental de XP es:

- **La comunicación**: Entre los usuarios y los desarrolladores.
- **La simplicidad**: Al desarrollar y codificar los módulos del sistema.
- **La retroalimentación**: Concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

El desarrollo bajo XP tiene características que lo distinguen claramente de otras metodologías:

- Los diseñadores y programadores se comunican efectivamente con el cliente y entre ellos mismos.
- Los diseños del software se mantienen sencillos y libres de complejidad o pretensiones excesivas.
- Se obtiene retroalimentación de usuarios y clientes desde el primer día gracias a las baterías de pruebas.
- El software es liberado en entregas frecuentes tan pronto como sea posible.
- Los cambios se implementan rápidamente tal y como fueron sugeridos.
- Las metas en características, tiempos y costos son reajustadas permanentemente en función del avance real obtenido.

Esta metodología ha sido diseñada para solucionar el eterno problema del desarrollo de software por encargo: entregar el resultado que el cliente necesita a tiempo.

1.4.2 SCRUM

Dentro de las metodologías ágiles se encuentra SCRUM y se enfoca fundamentalmente hacia la planeación iterativa y el seguimiento del proceso. Sus características son muy cercanas a las de XP, sin embargo presenta procesos de iteración con un período de 30 días, denominadas carreras cortas.
La metodología SRUM se divide de forma tal que cuando se vaya a hacer una primera carrera se debe definir la funcionalidad para la misma y luego se deja al equipo de desarrolladores para que haga entrega de la primera iteración. La actividad principal en esta primera corrida es la de estabilizar los requisitos. Para ello la gerencia se da a la tarea de reunir al equipo todos los días durante 15 min para discutir las tareas del próximo día, haciendo que no se pierda el objetivo perseguido durante la carrera: la comunicación entre todos. Durante esta reunión se muestran todos los inconvenientes que impidan el progreso de la carrera y la gerencia debe resolver los mismos. Al igual la gerencia debe tener toda la información diaria que se genera como una forma de mantener la actualización diaria del proyecto. 18]

¿Qué es SCRUM?

SCRUM es una metodología que surge ajena al desarrollo del software. Sus principios fundamentales están basados en los procesos de re ingeniería hacia la década de 1980 por Goldratt, Takeuchi y Nonaka, sin embargo esta no fue aplicada hasta 1993, por Jeff Sutherland. [24]
Las metodologías ágiles se centran principalmente en el factor humano o el producto de software, dándole una mayor prioridad al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Aquí se muestra la efectividad en los proyectos con requisitos cambiantes, pero con la diferencia de tiempo reducido y el mantenimiento de la calidad del mismo.

1.4.3 Proceso Unificado de Desarrollo (RUP)

El Proceso Unificado de Desarrollo (RUP), propuesto por Jacobson, Booch y Rumbaugh se publicó en 1999.
Es un proceso de desarrollo de software, definido como un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software [16].

Sin embargo, el proceso unificado es más que un proceso de trabajo, es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones y diferentes niveles de aptitud.

Está basado en componentes y utiliza UML (Lenguaje de Modelado Unificado) para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. Está **dirigido por casos de uso** (forma en como un actor opera con el sistema en desarrollo), porque con éstos se especifican las funcionalidades que el sistema proporciona al usuario. Los casos de uso representan los requisitos funcionales y fuerzan a pensar en términos de importancia para el usuario, y no sólo en términos de qué funciones sería bueno tener. Los casos de uso no sólo son una herramienta para especificar los requisitos del sistema, también guían su diseño, implementación y pruebas, es decir, guían todo el desarrollo software.

Se **centra en la arquitectura**, pues la manera en que se organiza el sistema depende de los casos de uso clave y debe tener en cuenta la comprensibilidad, la facilidad de adaptación al cambio y la reutilización. Los casos de uso clave son aquellos que dotan al sistema con la funcionalidad fundamental para los usuarios y sin los cuales, los demás casos de uso no tienen sentido.

Por ser un proceso **iterativo e incremental** el trabajo se divide en partes más pequeñas llamadas iteraciones. En cada iteración se recorren los flujos de trabajo que forman el conjunto de actividades a realizar.

Los elementos del RUP son:

- **Actividades**: Son los procesos que se llegan a determinar en cada iteración.
- **Trabajadores**: Son las personas o entes involucrados en cada proceso.
- **Artefactos**: Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

RUP presenta cuatro fases [16]:

Fase de concepción: Esta fase tiene por finalidad definir la visión, los objetivos y el alcance del proyecto, tanto desde el punto de vista funcional como del técnico, obteniéndose como uno de los principales resultados, una lista de los casos de uso y una lista de los factores de riesgo del proyecto. El principal esfuerzo está radicado en el “Modelamiento del Negocio” y el “Análisis de Requerimientos”. Es la única fase que no necesariamente culmina con una versión ejecutable, si bien muchas veces se desarrollan las interfaces con el usuario, o se prueban algunos aspectos técnicos críticos (por ejemplo la factibilidad de conectarse a una determinada Base de Datos).

Fase de elaboración: Esta fase tiene como principal finalidad completar el análisis de los casos de uso y definir la arquitectura del sistema. En esta etapa se busca eliminar los principales riesgos técnicos.

Fase de construcción: Esta fase está compuesta por un ciclo de varias iteraciones, en las cuales se van incorporando sucesivamente los casos de uso, de acuerdo a los factores de riesgo del proyecto. Este enfoque permite por ejemplo contar en forma temprana con versiones del sistema que satisfacen los principales casos de uso. Los cambios en los requerimientos no se incorporan hasta el inicio de la próxima iteración.

Fase de transición: Esta fase se inicia con una primera versión del sistema y culmina con el sistema en fase de producción.

Flujos de trabajo

- **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- **Prueba (Testeo):** Busca los defectos a los largo del ciclo de vida.
- **Instalación o despliegue:** Produce release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.
- **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.

- **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

RUP es una metodología de desarrollo pensada para proyectos grandes, a largo plazo y con un equipo de desarrollo numeroso. Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

Es explícito en la definición de artefactos y su trazabilidad, es decir, contempla en relación causal de los artefactos creados desde los requerimientos hasta la implementación y pruebas.

RUP identifica claramente a los profesionales (actores) involucrados en el desarrollo del software y sus responsabilidades en cada una de las actividades. Además, explícitamente indica qué actor es responsable de qué artefacto en cada actividad.

1.5 Lenguaje y herramientas de modelado.

El Proceso Unificado es un proceso de desarrollo de software (conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software). Es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos.

El Proceso Unificado está basado en componentes. Utiliza el lenguaje unificado de modelado (UML) para preparar todos los esquemas de un sistema de software. De hecho, UML es una parte esencial de RUP, sus desarrollos fueron paralelos.

1.5.1 UML (Unified Modeling Languaje)

UML (Unified Modeling Languaje) o 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) [19]. Pueden ser artefactos: un modelo, una descripción que comprende el desarrollo de software que se basen en el enfoque Orientado a Objetos. UML usa procesos de otras metodologías, aprovechando la experiencia de sus creadores, eliminó los componentes que resultaban de poca utilidad práctica y añadió nuevos elementos. (Histchfeld) 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.

De forma general las principales características son [19]:

- Lenguaje unificado para la modelación de sistemas.
- Tecnología orientada a objetos.
- El cliente participa en todas las etapas del proyecto.
- Corrección de errores viables en todas las etapas.
- Aplicable para tratar asuntos de escala inherentes a sistemas complejos de misión crítica, tiempo real y cliente/servidor.

UML es desde finales de 1997, un lenguaje de modelado orientado a objetos estándar, de acuerdo con el *Object Management Group*, siendo utilizado diariamente por grandes organizaciones como: **Microsoft**, **Oracle**, **Rational**.

1.5.2 Herramientas de modelado

En el mundo informático se han creado varias herramientas para el desarrollo de la ingeniería de software con el objetivo de desarrollar programas. Las herramientas CASE (Computer Aided Software Engineering), Ingeniería de Software Asistida por Ordenador que son un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación pasando por el análisis y diseño, hasta la generación del código fuente de los programas y la documentación. [12]
Es conocido que desde los años 70 se comenzaron los estudios para la creación de una herramienta para crear proyectos y fue en 1984 que aparece la primera herramienta CASE con el nombre de Excelerator y es entonces que en la década de los 90 que comienza a tener auge entre la comunidad.

Las herramientas CASE tiene como objetivo primordial:

1. Mejorar la productividad en el desarrollo y mantenimiento del software.
2. Aumentar la calidad del software.
3. Mejorar el tiempo y coste de desarrollo y mantenimiento de los sistemas informáticos.
4. Mejorar la planificación de un proyecto
5. Aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos.
6. Automatizar, desarrollo del software, documentación, generación de código, pruebas de errores y gestión del proyecto.

7. Ayuda a la reutilización del software, portabilidad y estandarización de la documentación
8. Gestión global en todas las fases de desarrollo de software con una misma herramienta.
9. Facilitar el uso de las distintas metodologías propias de la ingeniería del software.

Las herramientas CASE-UML se basan en las fases del ciclo de desarrollo que cubren y se clasifican en:

- Upper Case: planificación y requerimientos.
- Middle Case: análisis y diseño.
- Lower Case: generación de código, pruebas e implementación. [12]

Entre las herramientas CASE orientadas a UML podemos encontrar: ArgoUML, Poseidon, MagicDraw UML, Visual Paradigm y Borland Together. [12]

1.5.2.1 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 todo 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.[32]
Es importante destacar que el Visual Paradigm como herramienta de modelado posee licencia gratuita. [12]

Las herramientas Visual Paradigm basado en UML se clasifica en:

- Producto de calidad.
- Soporta aplicaciones Web.
- Las imágenes y reportes generados, no son de muy buena calidad.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.

Visual Paradigm para UML Community Edition (CE VP-UML): Es una plataforma de modelado de diseño para el aprendizaje del modelado UML a través de nuestras mejores tecnologías de modelado visual UML que facilita la visualización en diferentes tipos de diagramas como los diagramas entidad-relación, diagrama de requisitos y análisis textual de datos.
Permite la creación de diagramas mucho más rápido que cualquier herramienta en el mercado a través de VP-UML con recursos más intuitivos con ubicación céntrica. Poniendo en orden su complejo y desordenado diagrama con tan sólo un clic del ratón utilizando la disposición automática de la tecnología de VP-UML y, sin embargo, el gesto de capacidad de edición de diagramas lo hace una divertida tarea.[9]

Extensibilidad y diseño personalizado de apoyo: La ampliación de las capacidades UML más allá de la última UML 2,1 ya no es un sueño. VP-UML permite diseñar nuevas notaciones o incorporar formas personalizadas mediante el uso de símbolos incrustados en forma de editor, la importación de imágenes o íconos, con capacidad de integración con Visio UML. VP-UML CE es para uso no comercial y sólo admite un diagrama por tipo de diagrama. Todos los productos de la VP-UML CE tienen una sola línea de marca de agua en la parte superior izquierda de la pantalla indicando que el software es para uso no comercial. [9]
Visual Paradigm para UML (VP-UML) es amistoso y con alternativa económica junto a la Borland IBM Rational Rose. Tiene como característica principal ser abundante más fácil de utilizar en mercado en cuanto al uso que las herramientas CASE-UML le proporcionan una buena interoperabilidad con otras aplicaciones; además de su Smart Development Environment (SDE) esta disponible en varias versiones las cuales se integran en un IDE.

A continuación aparecen 10 razones por las cuales una organización debe elegir VP-UML y SDE [34]:

1. Permite incorporar dibujos de Visio en cualquier diagrama UML: A diferencia de otras herramientas CASE-UML, VP-UML y SDE prolongan esta limitación permitiendo incorporar dibujos de Visio en cualquier diagrama de UML, pudiendo modelar un dominio específico de hardware, software, redes, componentes, etc.
2. Integración perfecta con los principales IDEs: SDE esta disponible en varios IDEs incluyendo Microsoft Visual Studio, Borland JBuilder, Eclipse/IBM WSAD, NetBeans/Sun ONE, IntelliJ IDEA y JDeveloper, lo que puede proporcionar un

entorno que integra todo el modelo de código desplegando el proceso de desarrollo de software.

3. Completa integración con Microsoft Office: VP-UML y SDE soporta la copia de diagramas como objetos OLE, para poder pegar el diagrama a cualquier documento Word, Excel, Power Point. El contenido del diagrama se puede editar directamente, sin embargo no hay que preocuparse de perder el diagrama original de la fuente.
4. Herramientas efectivas y costeables: son los más asequibles con las actuales funciones de las herramientas CASE-UML. Cuestan la octava parte del precio de otras herramientas de modelado con funcionalidades similares. Sus diversas ediciones se encuentran disponibles y se puede elegir de acuerdo a su presupuesto y necesidades.
5. Herramientas UML más fáciles de usar: Ofrece una edición de líneas para diagramas UML. Se puede crear y especificar un modelo de elementos sin cuadro de diálogos. Los recursos centrados en la interfaz ayuda, en las tareas y botones de acceso directo se muestran cuando un elemento del diagrama está seleccionado. Se realizan operaciones como mostrar/ocultar compartimientos, ajuste de tamaño y creación de elementos con un solo clic del ratón.
6. Soporta múltiples plataformas: no importa el Sistema Operativo que este en uso, VP-UML y SDE se puede ejecutar en equipo y están disponibles en muchas plataformas como Windows, Linux, Mac OS X y Java Desktop.
7. Análisis textual para la identificación de clases candidatas: el análisis textual es una técnica útil y práctica para la captura de los requisitos del sistema y la identificación de las clases candidatas. Éstas son las únicas herramientas CASE que apoyan el análisis textual. Sólo se rellena la declaración del problema y las herramientas pueden ayudarle a identificar el sistema de clases con solo arrastrar y soltar.
8. Soporte para las tarjetas CRC: Las tarjetas CRC identifican clases, responsabilidades y colaboraciones en un sistema OO. Son las únicas herramientas de modelado que apoyan las tarjetas CRC.
9. Conversión instantánea de código fuente, archivos ejecutables y binarios en modelos: Permite invertir programas de código fuente, archivos ejecutables y binarios y modelos UML de inmediato. Además de idiomas y formatos que incluyen XML, esquema XML, .NET dll o exe, Java de fuente/class/jar, origen de archivo C++ y archivos fuente CORBA IDL.

10. Diseño automático del diagrama: Con su diseño automático se pueden ordenar los diagramas desordenados con tan solo unos pocos clics con el ratón. Se puede elegir el trazado ortogonal, la disposición jerárquica o el árbol del diseño de acuerdo a la naturaleza del diagrama. Cada estilo del diseño puede afinarse con un conjunto de parámetros configurables.

Después de exponer las principales características de la herramienta CASE a utilizar, en el próximo epígrafe se hará una breve exposición de los aspectos fundamentales, conceptos y patrones que se tratan en el flujo de trabajo de Análisis y Diseño.

1.6 Análisis

La ingeniería de requisitos del software es un proceso de descubrimiento, refinamiento, modelado y especificación. En esta etapa es donde se refinan los requisitos del sistema, se crean modelos de los requisitos de datos, flujo de información y control, y del comportamiento operativo. Se analizan soluciones alternativas y el modelado completo del análisis es creado. [28].

El análisis y la especificación de los requisitos puede en ocasiones parecer una tarea sencilla, pero no es así. Durante esta etapa aparecen las malas interpretaciones o la falta de información por lo que es muy probable que haya ambigüedad. Es por ello que la situación a la que se enfrenta el ingeniero de sistemas es muy difícil e importante para el futuro desarrollo del software. [28].

El análisis de requisitos es una tarea de ingeniería del software que cubre la separación que hay entre la definición del software a nivel de sistema y el diseño del software. [28].

El modelado de análisis es independiente del paradigma de ingeniería a usar. Es realmente un conjunto de modelos que muestran las primeras presentaciones técnicas del sistema. Este modelo debe lograr 3 objetivos primarios:

- Describir lo que requiere el cliente.
- Establecer una base para la creación de un diseño de software.
- Definir un conjunto de requisitos que se pueda validar una vez que se construye el software. [28]

1.6.1 Principios del Análisis

Son conocidos los estudios realizados por investigadores donde alegan haber identificado los problemas del análisis y sus causas, además de las heurísticas para

dar solución a los mismos. Cada método de análisis tienen su punto de vista y cada uno de ellos tienen relación con un conjunto de principios operativos:

- Debe representarse y entenderse el dominio de información de un problema.
- Deben definirse las funciones que debe realizar el software.
- Debe representarse el comportamiento del software (como consecuencia de acontecimientos externos).
- Deben dividirse los modelos que representan información, función y comportamiento de manera que se descubran los detalles por capas o jerárquicamente.

El proceso de análisis debería ir desde la información esencial hasta el detalle de la implementación. [28]

Se dice que poniendo en práctica estos principios el analista de sistema va a poder acercarse un poco más al problema. Se examina el dominio para poder comprender la información; se emplean modelos para comunicar las características de la función y su comportamiento; se aplica la partición para reducir la complejidad; y son necesarias las visiones esenciales y de implementación para acomodar las restricciones lógicas y físicas de los requisitos del sistema. [28]

Además de los principios operativos de análisis, se sugieren principios directrices, los cuales si se aplican correctamente, se puede desarrollar una excelente especificación del software de forma tal que se proporcione un buen fundamento para el diseño. Los principios directrices se basan en:

- Entender el problema antes de empezar a crear el modelo de análisis.
- Desarrollar prototipos que permitan al usuario entender como será la interacción hombre-máquina.
- Registrar el origen y la razón de cada requisito.
- Usar múltiples planteamientos de requisitos.
- Dar prioridad a los requisitos.
- Trabajar para eliminar la ambigüedad.

1.6.2 El análisis como fase de elaboración.

El artefacto, modelo de análisis, está compuesto principalmente por clases del análisis y por realizaciones de casos de uso del análisis. Se pueden encontrar muchas clases del análisis, y mucha realización de casos de uso de análisis. [4]

Se puede decir que un modelo de análisis se basa en un modelo de bases conceptuales que comprende los requisitos del software y los describe en un lenguaje más técnico sin precisar como se implementa la solución. Realmente se puede o no

hacer dicho modelo, o sea, se puede pasar directamente al diseño y la implementación, sobre todo si el proyecto es muy pequeño, aunque se puede perder muchas de las ventajas que brinda un modelo del análisis. [4]

Ventajas del modelo de análisis:

- El modelo de análisis suaviza la transición hacia el diseño.
- Da apoyo en caso de tener la necesidad de realizar cambios a otra plataforma de programación.
- Sirve para tener una visión general de la propuesta del sistema.
- Sirve para planificar y dividir el diseño e implementación en pequeños módulos.
- Apoya la aplicación de reingeniería a aplicaciones ya existentes, ya que los lenguajes al ser menos técnico se pueden entender mucho más fácil la propuesta de solución.[4]

Al igual que otros modelos el de análisis está compuesto por artefactos como son las clases del análisis y la realización de los CU del análisis.

Las clases del análisis representan abstracciones de conceptos que se trabajan en un contexto determinado. Para ellas deben de incluirse atributos y operaciones pero a un nivel bastante alto, donde no se incluya el paso de parámetros ni el tipo de datos. Las clases del análisis siempre deben encajar en 3 tipos o estereotipos básicos: interfaz, entidad, control.

Las clases interfaz modelan la interacción entre el sistema y sus actores. Se encargan de clarificar y reunir los requisitos que limitan el sistema, se representan a través de ventanas, formularios, interfaces de comunicación con otros sistemas o dispositivos.

Las clases entidades modelan la información tiene larga vida en el sistema y que a veces son persistentes. Modelan la información y el comportamiento asociado a algún fenómeno o concepto del mundo real (persona, objeto, suceso, lugar, etc.).

Las clases de control coordinan el trabajo de uno o unos pocos casos de uso, coordinando las actividades de los objetos que implementan la funcionalidad del mismo. Estas clases se usan con frecuencia para encapsular el control de un caso de uso completo y también para implementar cálculos complejos.

La realización de los casos de uso del análisis son los diagramas que permiten expresar el comportamiento que tienen las clases que se han trabajado en el diagrama de clases del análisis, ello implica una trazabilidad de uno a uno con el diagrama de caso de uso.

Un diagrama de interacción expresa la comunicación entre las clases que intervienen en la realización de un caso de uso, por lo que representa la vista dinámica del sistema y se modela por medio de objetos concretos o instancias de las clases ya definidas. Estos diagramas pueden ser de dos tipos: diagrama de secuencia o

diagrama de colaboración. Estos no son excluyentes entre sí, puesto que cada uno de ellos tiene objetivos muy diferentes y muy bien establecidos. Aquí la trazabilidad no es de uno a uno, sino que se hace un diagrama de interacción por cada escenario de caso de uso.

En un diagrama de secuencia se destaca el orden de los mensajes, apareciendo así el flujo de mensajes secuenciales que se establece entre los objetos.

En un diagrama de colaboración se destaca la organización de los objetos. Este se construye colocando mensajes ente los objetos como nodos de un grafo y se conectan como si fueran arcos.

Comparación entre le modelo de análisis y el modelo del diseño: un modelo de análisis es un modelo básicamente conceptual, sin embargo un modelo de diseño es un modelo físico. El modelo de análisis es un modelo genérico con respecto al diseño, o sea puede servir para varios tipos de diseño, sin embargo el modelo de diseño es un modelo no genérico, es específico para un lenguaje de programación. En el moldeo de análisis solo se tiene 3 tipos o estereotipos conceptuales y en el modelo de diseño se puede utilizar cualquier tipo de estereotipo en dependencia del lenguaje de programación. Por otro lado tanto el modelo de análisis como el modelo de diseño es un modelo dinámico aunque tiene diferentes características internas. El modelo de análisis puede realizarse solo una vez y no refinarse en las posteriores iteraciones, sin embargo el modelo de diseño debe ser mantenido durante todo el ciclo de vida del software. Finalmente el modelo de análisis define una estructura que es una entrada esencial para la creación del modelo del diseño y el modelo del diseño da forma al sistema mientras intenta preservar la estructura definida por el modelo de análisis lo mas posible peor esto no quiere decir que no pueda cambiarse en este modelo. [4]

1.6.3 Rol de analista de sistemas

El analista de sistemas surge de la necesidad de analizar, identificar y separar en procesos toda la información referente al software que se desee construir o mejorar, es el encargado de proponer soluciones y seleccionar la idea más idónea para el problema en cuestión.

Varias de las actividades en el desempeño del rol del analista de sistemas son:

- Consultores externos para negocios.
- Experto de soporte dentro de un negocio.
- Agente de cambio en situaciones tanto internas como externas.

Los analistas poseen un amplio rango de habilidades. La primera y principal es que el analista soluciona problemas, le motiva el reto de analizar un problema y encontrar

una respuesta funcional que satisfaga al cliente. Los analistas de sistemas requieren habilidades de comunicación que les permitan relacionarse en forma significativa con muchos tipos de personas diariamente, así como habilidades de computación. Para el éxito del analista es necesario que se involucre el usuario final.

La labor del analista de sistemas comienza desde el inicio de la creación del software, en la identificación de los procesos del negocio valorando la manera en que funcionan los mismos y examinando las entradas, el procesamiento de datos y la salida de información, con el objetivo de automatizar los procesos identificando las necesidades del cliente.

En cualquier proceso de desarrollo de software es indispensable el analista de sistemas. Su labor es lograr que tanto el cliente como los desarrolladores hablen el mismo idioma. Selecciona la metodología y define la estrategia de captura de requisitos con el propósito de lograr los objetivos que se proponen.

1.7 Diseño

El diseño es una representación significativa de ingeniería de algo que se va a construir. Se puede hacer el seguimiento basándose en los requisitos del cliente, y al mismo tiempo la calidad se puede evaluar y cotejar con el conjunto de criterios predefinidos para obtener un diseño “bueno”. En el contexto de la ingeniería del software, el diseño se centra en cuatro áreas importantes de interés: datos, arquitectura, interfaces y componentes [28].

El diseño de software es uno de los pilares fundamentales de la Ingeniería de software. Es la única forma de convertir exactamente los requisitos de un cliente en un producto o sistema de software finalizado [28]. Entre las tareas fundamentales del diseño están: producir un diseño de datos, un diseño arquitectónico, un diseño de interfaz y un diseño de componentes.

1.7.1 Calidad del diseño

La importancia del diseño de software se puede describir con una sola palabra –calidad-. El diseño es el lugar en donde se fomentará la calidad en la Ingeniería de software [28]. Existen tres características fundamentales que se deben tener en cuenta a la hora de realizar un buen diseño.

Características:

- El diseño deberá implementar todos los requisitos explícitos del modelo de análisis, y deberá ajustarse a todos los requisitos implícitos que desea el cliente.

- El diseño deberá ser una guía legible y comprensible para aquellos que generan código y para aquellos que comprueban y consecuentemente, dan soporte al software.
- El diseño deberá proporcionar una imagen completa del software, enfrentándose a los dominios de comportamiento, funcionales y de datos desde una perspectiva de implementación.

Si tenemos en cuenta estas características cuando realicemos nuestro diseño estamos garantizando que este tenga calidad. Además existen otros aspectos que se deben tener en cuenta a la hora de diseñar.

Existen varios criterios por los cuales se puede evaluar la calidad del diseño:

- Un diseño deberá presentar una estructura jerárquica que facilite el trabajo con los componentes del software.
- Un diseño deberá ser modular, es decir que deberá dividirse de forma lógica en elementos que realicen funciones específicas.
- Un diseño deberá contener distintas representaciones de datos y procedimientos.
- Un diseño deberá conducir a módulos que tengan características funcionales independientes.
- Deberá conducir a interfaces que reduzcan la complejidad de las conexiones entre los módulos y el entorno externo.
- Deberá producir un diseño mediante un método que pudiera repetir la información obtenida durante el análisis de los requisitos del software.

1.7.2 Principios del diseño

Roger Pressman en su libro: “Ingeniería del Software. Un enfoque práctico” plantea una serie de principios básicos del diseño que hacen posible que el ingeniero de software navegue por el proceso de diseño, y que hay que tener en cuenta a la hora de diseñar ya que ellos garantizan el correcto funcionamiento del sistema.

- En el proceso de diseño no deberá utilizarse “orejeras”.
- El diseño deberá poderse rastrear hasta el modelo de análisis.
- El diseño no deberá inventar nada que ya esté inventado.
- El diseño deberá minimizar la distancia intelectual entre el software y el problema como si de la vida real se tratara.
- El diseño deberá presentar uniformidad e integración.
- El diseño deberá estructurarse para admitir cambios.
- El diseño deberá estructurarse para degradarse poco a poco, incluso cuando se enfrenta con datos, sucesos o condiciones de operación aberrantes.

- El diseño no es escribir código y escribir código no es diseñar.
- El diseño deberá evaluarse en función de la calidad mientras se va creando, no después de terminado.
- El diseño deberá revisarse para minimizar los errores conceptuales (semánticos).

El aplicar adecuadamente los principios descritos anteriormente proporciona la creación de un diseño el en el cual se muestran tanto los factores de la calidad externos como los internos. Los factores de la calidad externos son esas propiedades que puedes ser observadas por los usuarios (velocidad, fiabilidad, grado de corrección, usabilidad). Los factores de la calidad internos tienen importancia para los ingenieros del software, los cuales deberán comprender los conceptos de diseño básicos para poder lograr estos factores.

1.7.3 Conceptos del diseño

Los conceptos de diseño son muy importantes para el diseñador, cada uno de ellos proporcionará una base a partir de la cual podrán aplicarse los métodos de diseño. Cuando se hace un programa es importante hacerlo funcionar, pero es más importante aún hacerlo funcionar bien. Esto es lo que nos permiten los conceptos diseño de software. Pressman en una de sus publicaciones habla sobre una serie de conceptos importantes. Ver Anexo 1.

1.7.4 Patrones de diseño

Uno de los principios del diseño plantea que el diseño no deberá inventar nada que ya esté inventado. Los sistemas se construyen utilizando un conjunto de patrones de diseño, muchos de los cuales probablemente ya se han encontrado antes. Estos patrones deberán elegirse siempre como una alternativa para reinventar [28].

El concepto de patrón como solución probada a un problema dado, surgió de un área tan poco relacionada con la programación como es la arquitectura en los años 70 (el arquitecto Christopher Alexander comenzó a aplicarlos), dónde eran aplicados cuando requería enfrentarse a un problema general en un diseño [32].

Un patrón es una solución en forma de reglas para problemas conocidos [32]. Esta solución es efectiva (se ha utilizado en resolver el problema anteriormente) y reutilizable (se puede aplicar a diferentes problemas en circunstancias distintas).

Solo los patrones que han sido más utilizados son los que son elevados a la comunidad de desarrolladores, ya que han sido testeados y probados en variedad de escenarios, es por eso que los patrones de diseño proponen una forma de reutilizar la

experiencia de los desarrolladores, para ello clasifican y describen formas de solucionar problemas que ocurren de forma frecuente en el desarrollo [20].

1.7.4.1 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. [19] GRASP es un acrónimo que significa General Responsability Asignment Software Paterns (patrones generales de software para asignar responsabilidades). [19]

En los patrones GRASP están presentes algunos de los principios de diseño, los cuales son aplicados cuando se elaboran los diagramas de interacción y a la hora de asignar las responsabilidades a las clases y objetos que conforman el sistema que se esté diseñando. Es importante recalcar que responsabilidad no equivale a métodos, los métodos se ponen en práctica para cumplir con las responsabilidades. [19]

Existen nueve patrones GRASP: Experto, Creador, Alta Cohesión, Bajo Acoplamiento, Controlador, Polimorfismo, Fabricación Pura, Indirección y No Hables con Extraños. Cada uno de ellos indica la solución que se le debe dar a una problemática determinada. En la tabla siguiente se muestran estos patrones con los problemas que resuelven y las soluciones que le dan. Ver Anexo 2.

Es importante dominar estos patrones, ya que constituyen el fundamento de cómo se diseña un sistema orientado a objetos [19]. No introducen ideas nuevas, son una mera codificación de los principios básicos más usados [19]. Cada uno de los patrones expuestos soluciona problemas que contribuyen a que los sistemas sean más robustos y más flexibles. Los patrones GRASP no compiten con los patrones de diseño, sino que guían para ayudar a encontrar los patrones de diseños adecuados para aplicar al sistema que se esté elaborando [20].

1.7.4.2 Patrones Gof

Estos patrones se descubren como una forma indispensable de enfrentarse a la programación a raíz del libro “Design Patterns—Elements of Reusable Software” de Erich Gamma, Richard Helm, Ralph Jonson y John Vlissides, a partir de entonces estos patrones son conocidos como los patrones de la pandilla de los cuatro (GoF, gang of four) [32].

Los patrones de diseño tienen como características:

- Son soluciones concretas. Proponen soluciones a problemas concretos, no son teorías genéricas.
- Son soluciones técnicas. Indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO). En ocasiones tienen más utilidad

con algunos lenguajes de programación y en otras son aplicables a cualquier lenguaje.

- Se utilizan en situaciones frecuentes. Debido a que se basan en la experiencia acumulada al resolver problemas reiterativos.
- Favorecen la reutilización de código. Ayudan a construir software basado en la reutilización, a construir clases reutilizables. Los propios patrones se reutilizan cada vez que se vuelven a aplicar.
- El uso de un patrón no se refleja en el código. Al aplicar un patrón, el código resultante no tiene por que delatar el patrón o patrones que lo inspiró. No obstante últimamente hay múltiples esfuerzos enfocados a la construcción de herramientas de desarrollo basados en los patrones y frecuentemente se incluye en los nombres de las clases el nombre del patrón en que se basan facilitando así la comunicación entre desarrolladores.
- Es difícil reutilizar la implementación de un patrón. Al aplicar un patrón aparecen clases concretas que solucionan un problema concreto y que no será aplicable a otros problemas que requieran el mismo patrón.

Se clasifican según su propósito en:

Patrones de Creación

Se encargan de la creación de instancias de los objetos. Abstraen la forma en que se crean los objetos, permitiendo tratar las clases a crear de forma genérica, dejando para después la decisión de que clase crear o cómo crearla.

Según donde se tome dicha decisión se pueden clasificar los patrones de creación en: patrones de creación de clases (la decisión se toma en los constructores de las clases y usan la herencia para determinar la creación de las instancias)

Los patrones de creación son [27]:

- *Abstract Factory (Fábrica Abstracta):* Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.
- *Builder (Constructor virtual):* Abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto.
- *Factory Method (Método de fabricación):* Centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que se crea.
- *Prototype (Prototipo):* Crea nuevos objetos clonándolos de una instancia ya existente.

- *Singleton (Instancia única):* Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Patrones Estructurales

Son los que plantean las relaciones entre clases, las combinan y forman estructuras mayores. Tratan de conseguir que los cambios en los requisitos de la aplicación no ocasionen cambios en las relaciones entre los objetos. Lo fundamental son las relaciones de uso entre los objetos, y éstas están determinadas por las interfaces que soportan los objetos. Estudian cómo se relacionan los objetos en tiempo de ejecución. Sirven para diseñar las interconexiones entre los objetos.

Los patrones estructurales son [27]:

- *Adapter (Adaptador):* Adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla.
- *Bridge (Puente):* Desacopla una abstracción de su implementación.
- *Composite (Objeto compuesto):* Permite tratar objetos compuestos como si de uno simple se tratase.
- *Decorator (Envoltorio):* Añade funcionalidad a una clase dinámicamente.
- *Facade (Fachada):* Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.
- *Flyweight (Peso ligero):* Reduce la redundancia cuando gran cantidad de objetos poseen idéntica información.
- *Proxy:* Mantiene un representante de un objeto.

Patrones de Comportamiento

Plantea la interacción y cooperación entre las clases. Los patrones de comportamiento estudian las relaciones entre llamadas entre los diferentes objetos, normalmente ligados con la dimensión temporal.

Los patrones de comportamiento son [27]:

- *Chain of Responsibility (Cadena de responsabilidad):* Permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada.
- *Command (Orden):* Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.
- *Interpreter (Intérprete):* Dado un lenguaje, define una gramática para dicho lenguaje, así como las herramientas necesarias para interpretarlo.
- *Iterator (Iterador):* Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos.

- *Mediator (Mediador):* Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto.
- *Memento (Recuerdo):* Permite volver a estados anteriores del sistema.
- *Observer (Observador):* Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él.
- *State (Estado):* Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno.
- *Template Method (Método plantilla):* Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos, esto permite que las subclases redefinan ciertos pasos de un algoritmo sin cambiar su estructura.
- *Visitor (Visitante):* Permite definir nuevas operaciones sobre una jerarquía de clases sin modificar las clases sobre las que opera.
- *Strategy (Estrategia):* Permite disponer de varios métodos para resolver un problema y elegir cuál utilizar en tiempo de ejecución.

1.8 Sistemas de Gestión de Bases de Datos

El software que permite la utilización y/o actualización de los datos almacenados en una o varias bases de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina sistema de gestión de bases de datos (SGBD).
El objetivo fundamental de un SGBD consiste en suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos, o sea, de forma que no le sea necesario conocer el modo de almacenamiento de los datos en la computadora, ni el método de acceso empleado.
Los programas de aplicación operan sobre los datos almacenados en la base utilizando las facilidades que brindan los SGBD, los que, en la mayoría de los casos, poseen lenguajes especiales de manipulación de la información que facilitan el trabajo de los usuarios.

1.8.1 MySQL

MySQL es una idea originaria de la empresa opensource MySQL AB establecida inicialmente en Suecia en 1995 y cuyos fundadores son David Axmark, Allan Larsson, y Michael "Monty" Widenius. El objetivo que persigue esta empresa consiste en que

MySQL cumpla el estándar SQL, pero sin sacrificar velocidad, fiabilidad o usabilidad [5].

Michael Widenius en la década de los 90 trató de usar mSQL para conectar las tablas usando rutinas de bajo nivel ISAM, sin embargo, mSQL no era rápido y flexible para sus necesidades. Esto lo conllevó a crear una API SQL denominada MySQL para bases de datos muy similar a la de mSQL pero más portable [5].

MySQL es muy utilizado en aplicaciones web y en plataformas como Linux y Windows. Su popularidad esta muy ligada a PHP, con el cual aparece muy a menudo en combinación.

Se han desarrollado varias versiones, siendo la ultima la 5.0 y una de las más usadas en el mundo. Entre las características disponibles en las últimas versiones podemos destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

MySQL tiene también características distintivas que sólo son implementadas en él, entre las que encontramos:

- Múltiples motores de almacenamiento, permitiendo al usuario escoger la que sea más adecuada para cada tabla de la base de datos.
- Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.

1.8.2 PostgreSQL

PostgreSQL es un servidor de base de datos objeto relacional libre, liberado bajo la licencia BSD. Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su

desarrollo, dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group) [6].

PostgreSQL ha tenido una larga evolución, comenzando con el proyecto Ingres en la Universidad de Berkeley. Este proyecto, liderado por Michael Stonebraker, fue uno de los primeros intentos en implementar un motor de base de datos relacional. Después de haber trabajado un largo tiempo en Ingres y de haber tenido una experiencia comercial con el mismo, Michael decidió volver a la Universidad para trabajar en un nuevo proyecto sobre la experiencia de Ingres, dicho proyecto fue llamado post-ingres o simplemente POSTGRES [6].

Entre las principales características de PostgreSQL están:

- Alta concurrencia: Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.
- Amplia variedad de tipos nativos: PostGreSQL provee nativamente soporte para:
    - Números de precisión arbitraria.
    - Texto de largo ilimitado.
    - Figuras geométricas (con una variedad de funciones asociadas).
    - Direcciones IP (IPv4 e IPv6).
    - Bloques de direcciones estilo CIDR.
    - Direcciones MAC.
    - Arrays.

Otras características que presenta PostGreSQL son las claves ajenas, denominadas también llaves ajenas o llaves foráneas y los disparadores.

..Un disparador se define en una acción específica basada en algo ocurrente dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica. [6]

Conclusiones

En este capitulo se hace una análisis de los principales aspectos y conceptos relacionados con el Nodo Virtual, así como de las aplicaciones existentes actualmente. También se analizan temas y conceptos de importancia dentro del análisis y diseño de software, pretendiendo dejar sentadas las bases teóricas del trabajo a desarrollar. Se realiza un estudio de las posibles herramientas a utilizar.

Como resultado de este análisis se llego a las siguientes conclusiones:

- Es necesaria la construcción de un Nodo Virtual de Procesos, ya que esta herramienta será de valor incalculable para que estudiantes desarrollen las habilidades y los conocimientos adquiridos.
- La metodología a utilizar en el desarrollo del proyecto será Rational Unified Process (RUP), debido a que es una metodología robusta que resulta más adaptable para los proyectos a largo plazo que necesiten un desarrollo iterativo capaz de mantener un buen control sobre los cambios. En base a esta decisión se define como lenguaje de modelado UML.
- Una buena práctica de diseño es la utilización de patrones como soluciones que ya han sido probadas satisfactoriamente en otras ocasiones.
- Como necesidad de desarrollar un producto en software libre, además de las características y ventajas que presentan estas herramientas se utilizara como Gestor de Base de Datos MySQL 5.0.24a, haciendo uso de la suite de herramientas AppServ 2.5.9, y como herramienta de modelado Visual Paradigm en su versión 6.0.

Capítulo 2: Características del Sistema

Introducción

En el presente capítulo se describe la propuesta de solución. Se centrará en el modelo de dominio, permitiendo con este mostrarle al usuario los principales conceptos que se manejan en el dominio del problema. Se plantean los requisitos funcionales y no funcionales de la aplicación a desarrollar y se modela la misma en términos de casos de uso del sistema para obtener mejores resultados.

2.1 Propuesta del sistema

Se propone realizar el análisis y diseño de un nodo virtual en el cual se implementen los modelos de procesos industriales que permitan la simulación de estos, así como probar aplicaciones en tiempo real. Entre las características principales del software a desarrollar se encuentran:

- Permitir la simulación concurrente de procesos industriales.
- Permitir la conexión con dispositivos físicos (autómatas) para probar estrategias de control mediante la implementación de protocolos de comunicación industriales.
- Permitir la supervisión y control por parte del profesor sobre los procesos que están corriendo simultáneamente y las acciones de los clientes (alumnos).

Toda la información de los procesos industriales que han sido simulados va a encontrarse tabulada y gráfica para facilitar un mejor entendimiento del mismo. Es importante garantizar una interfaz que facilite la visualización de la información y permita que esta sea consultada desde cualquier equipo de cómputo o autómata conectado a la red de manera directa o remota.

Se utilizara MySQL Server como gestor de la base de datos que contendrá toda la información referente al software a realizar. La aplicación brindará un conjunto de reportes para mostrar los datos del comportamiento de la simulación en cada uno de los procesos activados, a los cuales podrán acceder los usuarios del sistema que estén registrados en ese proceso. Además brindarán una serie de reportes que ayudaran al Administrador del sistema a controlar el buen funcionamiento de este.

Las múltiples tablas que serán diseñadas en la base de datos, contendrán toda la información de los usuarios del sistema, así como de los modelos de cada uno de los procesos industriales a simular, y los resultados de cada una de las simulaciones realizadas.

De esta forma quedaran automatizados los servicios de simulación de procesos que van ayudar a mejorar el sistema de aprendizaje de los estudiantes de la carrera de Ingeniería Automática.

2.2 Objetivos estratégicos específicos

- Brindar la posibilidad de que los estudiantes puedan poner en práctica los conocimientos adquiridos en clases de forma más eficiente.
- Permitir que el profesor pueda llevar la supervisión y control de la práctica que realizan los estudiantes.
- Brindar mejores condiciones de trabajo para estudiantes y profesores.

2.3 Modelo de Dominio

El negocio es el conjunto de servicios que una entidad, organización o empresa brinda a un conjunto de clientes o usuarios con el propósito de satisfacer las necesidades de estos. Al emplear la metodología basada en RUP, el modelamiento del negocio plantea la identificación de los procesos del negocio y su completo análisis, el cual servirá como base para la identificación de probables candidatos a sistemas informáticos que soporten el negocio total o parcialmente [10].

Dependiendo del escenario o situación que se presente, se determina cual opción es más adecuada para desarrollar este proceso, si Modelo de Negocio o Modelo de Dominio.

En el presente trabajo se decidió realizar modelo del dominio debido a que no se puede lograr determinar el proceso del negocio con fronteras bien establecidas donde se logren ver claramente, quienes son las personas que lo inician, quienes son los beneficiados con cada uno de estos procesos, pero además quienes son las personas que desarrollan las actividades en cada uno de estos procesos.

El modelo de Dominio es una de las alternativas que brinda RUP para la identificación de requisitos y la comprensión del contexto cuando existe poca estructuración en los procesos de negocio, y con la que se le puede mostrar al usuario de manera visual los principales conceptos que se manejan en el dominio del sistema, sus partes y sus relaciones. Esto les permite a todos los que de alguna manera están involucrados en el proceso de desarrollo del producto manejar un vocabulario común que posibilite el entendimiento del contexto en que se sitúa el sistema [16].

2.3.1 Descripción del negocio actual

Poner en práctica los conocimientos adquiridos por los estudiantes ha sido el talón de Aquiles dentro de la carrera de Ingeniería Automática, de ahí que se tomara la decisión de implementar una herramienta que permita simular procesos industriales o correr aplicaciones en tiempo real, como una forma de mejorar el aprendizaje de los estudiantes y el trabajo de profesores.

En la carrera de Ingeniería Automática se imparten asignaturas de peso para el desarrollo de futuros profesionales como son Modelación y Simulación, Teoría de Control, Sistemas de Mediciones, Medios Técnicos de Automatizaciones, Control de Procesos, y Automática que es la asignatura integradora. En todas ellas se trabaja con los modelos de procesos industriales, de ahí la importancia que tiene que los estudiantes interactúen con estos.

Actualmente para realizar la parte práctica de estas asignaturas se utilizan dos herramientas de simulación LabVIEW y MATLAB. LabVIEW es una herramienta gráfica para pruebas, control y diseño mediante la programación. Tiene un modulo para el diseño, simulación y programación de sistemas de control, mediante el cual se pueden simular sistemas lineales, no lineales y discretos con una amplia variedad de soluciones.

MATLAB (*MATriz LABoratory*) es un software matemático muy versátil que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Dentro del paquete MATLAB se dispone de una herramienta adicional llamada Simulink. Esta herramienta proporciona un entorno gráfico al usuario que facilita enormemente el análisis, diseño y simulación de sistemas (de control, electrónicos, etc.), al incluir una serie de rutinas que resuelven los cálculos matemáticos de fondo, junto con una sencilla interfaz para su uso.

Para correr aplicaciones en tiempo real actualmente se utiliza la herramienta SCADA (Supervisory Control and Data Acquisition, en español, Control supervisor y adquisición de datos). SCADA comprende todas aquellas soluciones de aplicación para referirse a la captura de información de un proceso o planta industrial, para que con esta información, sea posible realizar una serie de análisis o estudios con los que se pueden obtener valiosos indicadores que permitan una retroalimentación sobre un operador o sobre el propio proceso. Esta herramienta no esta creada con fines docentes. Actualmente se usa en la carrera para que los estudiantes además de realizar sus prácticas se vayan familiarizando con una herramienta que van a utilizar cuando ejerzan su profesión dentro de una industria.

En algunas universidades también existe la posibilidad de llevar a cabo prácticas físicas, pero estas son muy limitadas dado que no se cuenta con los recursos necesarios.

Al desarrollar esta herramienta se pretende reunir en ella características que eliminan las limitaciones que se encuentran en otros softwares utilizados para la simulación, como son la simulación concurrente de varios procesos, una supervisión y control centralizados que permitan a un usuario determinado poder supervisar desde el software las acciones de otro usuarios, y la conexión de dispositivos físicos para probar estrategias de control.

2.3.2 Glosario de términos

A continuación se presenta la descripción de las clases del dominio.

Usuario: Clientes que van a trabajar con la aplicación.

Nodo Virtual: Servidor donde se va ha encontrar toda la información necesaria para realizar la simulación de procesos industriales.

Autómata: Dispositivo físico para controlar procesos industriales.

Tipo de Proceso: Clasificación en la que se agrupan los procesos.

Modelo de Proceso: Conjunto de datos de un proceso; como son modelo matemático, variables de entrada y salida, método numérico y controlador.

Proceso: Conjunto de datos configurados para realizar la simulación.

Proceso Activado: Proceso que esta siendo simulado en esos momentos dentro del servidor.

Proceso Inactivo: Proceso que está sin activar dentro del servidor.

El usuario se conecta al nodo virtual, servidor donde se encuentran los procesos industriales con los que puede interactuar. Para entrar a un proceso, el usuario debe pulsar sobre el listado de procesos, a continuación se desplegará un menú con los tipos de procesos que existen dentro de la aplicación; al pulsar sobre un tipo de proceso se mostrarán los procesos que están comprendidos dentro de este. Los procesos se pueden encontrar de dos formas dentro de la aplicación, activos e inactivos; se podrán diferenciar por el icono que tiene cada uno, si el icono esta en rojo es que el proceso está activo, y si el icono está en amarillo es que el proceso esta inactivo.

Si el usuario decide entrar en un proceso activo, el sistema, después de verificar disponibilidad de conexión en ese proceso, le mostrará la interfaz donde se está simulando el mismo. El usuario solo podrá ser espectador del mismo.

De forma contraria, si el usuario entra en un proceso inactivo, el sistema le asigna el rol de usuario maestro de ese proceso, dándole la posibilidad de activarlo e interactuar completamente con el mismo.

Para activar el proceso primeramente el usuario maestro debe de configurarlo. Al pulsar la opción de configurar el sistema mostrará los datos del modelo de ese proceso. El usuario debe de entrar los valores de las variables de entrada, escoger de qué forma va a expresar el modelo matemático, y que método numérico y que controlador va a utilizar. El modelo matemático no es más que la ecuación matemática que representa el proceso y podrá ser expresada de seis formas distintas: Ecuación diferencial, Función transferencial, Modelo de estado, Diagramas en bloques, Diagrama de flujo de señales y Modelos dinámicos discretos. El método numérico es el que se va a utilizar en la resolución de la ecuación matemática del proceso. Hasta el momento se tienen definido dentro de la aplicación seis métodos numéricos: Método de Euler, Método de Adams-Bashford, Método de Serie de Taylor, Método Runge-Kutta, Método Predictor-Corrector y Método de Gear. Los controladores son ecuaciones matemáticas, definidas dentro del sistema, encargadas de controlar el mismo.

Estos datos son los necesarios para la simulación y después de configurados serán guardados por el sistema dentro de un objeto proceso. Después de realizada la configuración el usuario maestro puede pulsar el botón Simular.

Cuando el proceso comience la simulación el sistema abrirá una interfaz gráfica mostrando como se desarrolla el mismo y cuáles son los valores que toman las variables de salida. Durante la simulación el usuario maestro puede ir variando los valores de las variables de entrada, ya sea porque el controlador se lo indica o por que quiere interactuar con el proceso. Cuando lo desee puede terminar la simulación.

Si el usuario maestro lo desea el proceso puede ser probado de forma real. Para esto se cuenta con un dispositivo físico llamado autómata que realizará el control del proceso. En este caso el usuario maestro no escogerá un controlador de los que le brinda el modelo del proceso, y en lugar de simular pulsará el botón Probar Aplicaciones. La simulación se llevará a cabo normalmente, con la única diferencia en que el autómata es el que va a estar controlando el proceso.

2.3.3 Modelo de objeto del negocio

El objetivo del modelo de dominio es capturar los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que trabaja el sistema [16].

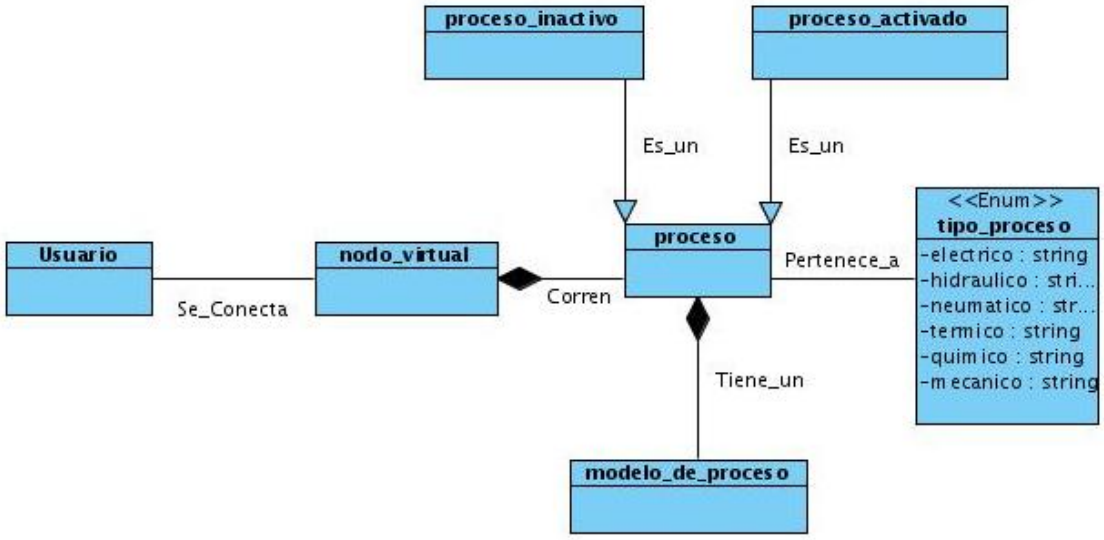


Figura 1 Modelo de Dominio

2.4 Modelo de Sistemas

Durante el ciclo de vida de un proceso de desarrollo de software es importante la etapa en la cual el proyecto está en la captura de requerimientos y el modelado del sistema, donde se van a identificar las necesidades de los clientes y se expresan los requisitos como casos de uso formándose así el modelo del sistema. Aquí es donde se identifican los actores del sistema y se conforman los casos de uso a partir de las actividades que fueron detectadas para ser automatizadas en el negocio.

2.4.1 Especificación de los requisitos de Software.

Los requisitos de software son descripciones orientadas al cliente de las funciones del sistema y de las restricciones en su operación. Para el levantamiento y seguimiento de los requisitos existe una amplia gama de técnicas y métodos. Una de las más utilizadas, de acuerdo a la metodología escogida son las de [14]:

- Entrevistas y cuestionario.
- Comparación con sistemas existentes.
- Grabaciones de video y de audio.
- Brainstorming (tormenta de ideas).
- Prototipos.
- Análisis de matriz DAFO (Debilidades, Amenazas, Fortalezas, Oportunidades).
- Glosario de términos.
- Checklist (lista de verificación).

El método más efectivo para la realización del levantamiento de los requisitos fue el de las entrevistas, donde los analistas de sistemas pudieron presentar las necesidades de los clientes de forma concreta y directa, así como verificar las respuestas que se recibieron por su parte de acuerdo a las preguntas realizadas, teniendo un grado de aceptación bastante aceptable entre el cliente y el sistema que se desea diseñar.

2.4.1.1 Definición de los requisitos funcionales

Los Requisitos Funcionales son capacidades que el sistema cumplirá. Es una tarea simple enunciada con un solo verbo y se corresponde con futuras opciones, acciones ocultas y condiciones extremas a determinar por el software. [14]

A continuación se presentarán los requisitos funcionales:

R1. Conexión Sistema

R1.1. Conectarse al sistema.

R1.2. Registrarse

R1.3. Autenticarse

R1.4. Asignar roles.

R1.5. Desconectarse del sistema.

R2. Conexión Proceso

R2.1. Conectarse a un proceso.

R2.2. Desconectarse de un proceso.

R3. Desconectar a un usuario de un proceso.

R4. Modificar contraseña

R5. Gestionar rol de administrador

R5.1. Adicionar administrador.

R5.2. Modificar datos de un administrador.

R5.3. Eliminar un administrador.

R6. Establecer Límite

R6.1. Limitar el número de usuarios conectados al sistema.

R6.2. Limitar el número de usuarios conectados por procesos.

R6.3. Limitar el número de procesos activos.

R7. Gestionar Tipos de Procesos

R7.1. Adicionar un tipo de proceso.

R7.2. Modificar un tipo de proceso.

R7.3. Eliminar un tipo de proceso.

R8. Gestionar Modelo de Procesos

R8.1. Adicionar un modelo de un proceso.

R8.2. Actualizar un modelo de un proceso.

R8.3. Eliminar un modelo de un proceso.

R9 Configurar Proceso

R9.1. Introducir los valores de las variables de entrada de un proceso.

R9.2. Escoger un método numérico.

R9.3. Escoger una forma de expresar el modelo.

R9.4. Escoger un controlador.

R10. Simular Proceso

R10.1. Activar un proceso.

R10.2. Mostrar los valores que van tomando las variables de salida.

R10.3. Cambiar los valores de las variables de entrada de un proceso.

R10.4. Mostrar la simulación de forma gráfica.

R10.5. Mostrar el resultado final de la simulación.

R10.6. Terminar un proceso.

R11. Probar Aplicaciones

R11.1. Activar un proceso.

R11.2. Realizar la conexión con el autómata.

R11.3. Mostrar los valores que van tomando las variables de salida del proceso.

R11.4. Mostrar los valores de salida del autómata.

R11.5. Cambiar los valores de las variables de entrada de un proceso.

R11.6. Mostrar la simulación de forma gráfica.

R11.7. Mostrar el resultado final de la simulación.

R11.8. Terminar un proceso.

R12. Adicionar controladores de procesos.

R13. Activar automáticamente uno o varios procesos.

R14. Priorizar el número de procesos activos por tipo de proceso.

R15. Mostrar un listado de los tipos de procesos.

R16. Mostrar un listado se los modelos de proceso dado un tipo de proceso.

R17. Mostrar un listado de procesos activos.

R18. Mostrar un registro de todos los usuarios conectados.

R19. Mostrar un registro con los usuarios conectados a un proceso.

R20. Mostrar un registro de los procesos activos de un cliente.

R21. Mostrar un registro con los valores de las variables de salida de un proceso durante una simulación. (Reporte de Simulación)

R22. Imprimir informes.

2.4.1.2 Definición de los requisitos no funcionales

Los Requisitos No Funcionales son propiedades o cualidades que el producto de software debe tener, como restricciones del entorno o la implementación, rendimiento, mantenimiento, facilidades, extensibilidad o fiabilidad. Los Requisitos No Funcionales pueden ser específicos de cada caso de uso o generales para todo el sistema, como es el caso del presente trabajo. [14]

A continuación se presentarán los requisitos no funcionales.

- **Hardware**

  Servidor

  1. Discos duros: SCCSI
  2. Microprocesador: 300 MHz como mínimo
  3. Memoria RAM:1Gb
  4. PC PENTIUM IV o superior
  5. 3 GB de espacio libre en el disco duro.

  PC clientes

  1. Microprocesador: 300 MHz como mínimo
  2. Memoria RAM: 256 Mb
  3. PC PENTIUM IV o superior

- **Software**

  Servidor

  1. Sistema Operativo: Linux

2. Gestor de Base de datos: My SQL

PC clientes

1. Sistema Operativo: Windows 2000 o superior, Linux.

- **Seguridad**

1. Cada usuario accederá solo a la información que le corresponda según su nivel de acceso.
2. Se llevara a cabo tratamiento de excepciones.
3. Programación de disparadores (Triggers) en la Base de Datos para no permitir la manipulación de los datos directamente en el SGBD.
4. El tipo de tecnología para la configuración que tendrán los discos duros en el servidor será Raid.

- **Redes**

1. La red existente en las instalaciones debe de soportar la tecnología multicast.
2. Las transacciones y recuperación de los datos en la comunicación servidor - pc cliente, se realizara a través del protocolo TCP/ IP.
3. Protocolo de comunicación con autómatas.

- **Portabilidad**

1. La aplicación deberá correr sobre cualquier Sistema Operativo.

- **Restricciones de diseño e implementación**

1. Debe implementarse usando una plataforma libre
2. Para la implementación del sistema se usara el lenguaje C++
3. El IDE a usar será Eclipse
4. Debe implementarse siguiendo la filosofía de la Programación Orientada a Objetos (POO).

2.5 Definición de los casos de uso del sistema

Los casos de uso son descripciones narrativas de los procesos en un formato estructurado de prosa. Constituyen un paso preliminar muy útil, porque describen las especificaciones del sistema. Estos especifican o incluyen tácticamente los requerimientos. [17]

Para establecer la definición de los casos de uso se usaron una serie de patrones que modelan reglas que hacen más comprensibles y correctos los modelos. Algunos de estos patrones y los que se seleccionaron a la hora de modelar los casos de uso fueron los siguientes:

- Extensión o Inclusión Concreta: Extensión: este patrón consiste en dos casos de uso y la relación de extensión entre ellos, es aplicable cuando un flojo puede extender el flujo de otro caso de uso.
- Extensión o Inclusión Concreta: Inclusión: este patrón consiste en la relación de incluido entre el caso de uso base y el caso de uso incluido. Es aplicable cuando un flujo puede incluir el flujo de otro caso de uso al igual que la realización del mismo. Tipo: Patrón de estructura.
- CRUD: Completo: Este patrón consiste en un caso de uso base llamado Información CRUD (o Administrador de información), el cual modela los distintos tipos de operaciones que pueden realizarse en un segmento de información de cierto tipo como: Crear, Eliminar, Actualizar, Modificar etc. Tipo: Patrón de estructura.

2.5.1 Actores del sistema. Descripción

Los actores son el conjunto de usuarios, personas, máquinas, software, hardware, etc. que interactúan con el sistema, y que “pueden intercambiar información o pueden ser un recipiente pasivo de información”.

A continuación se describen los actores del sistema.

Tabla 2 Relación de actor- función.

Nombre del actor	**Justificación**
Administrador	Es el encargado de controlar el funcionamiento de la aplicación.
Usuario Maestro	Es el primer usuario que se conecta a un proceso. Es el encargado de configurarlo y activarlo.
Usuario	Todos los usuarios que trabajan con la aplicación ya

	sea administradores o clientes.

2.5.2 Diagramas de casos de uso del sistema

A continuación se presenta el diagrama general de los casos de uso del sistema.

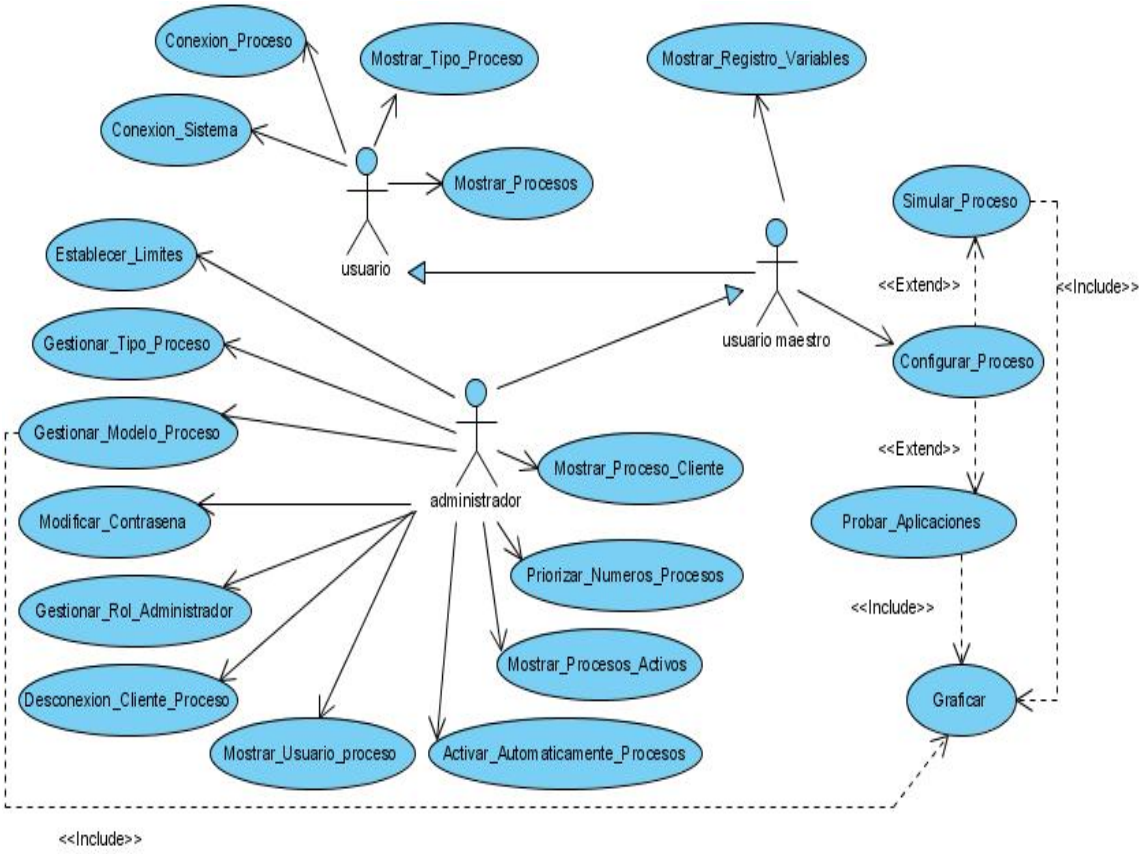


Figura 2 Diagrama General de Casos de Uso

Los casos de uso críticos según RUP son aquellos que reflejan una funcionalidad o tarea esencial para el cliente. A continuación les mostramos el diagrama de los casos de uso críticos del sistema.

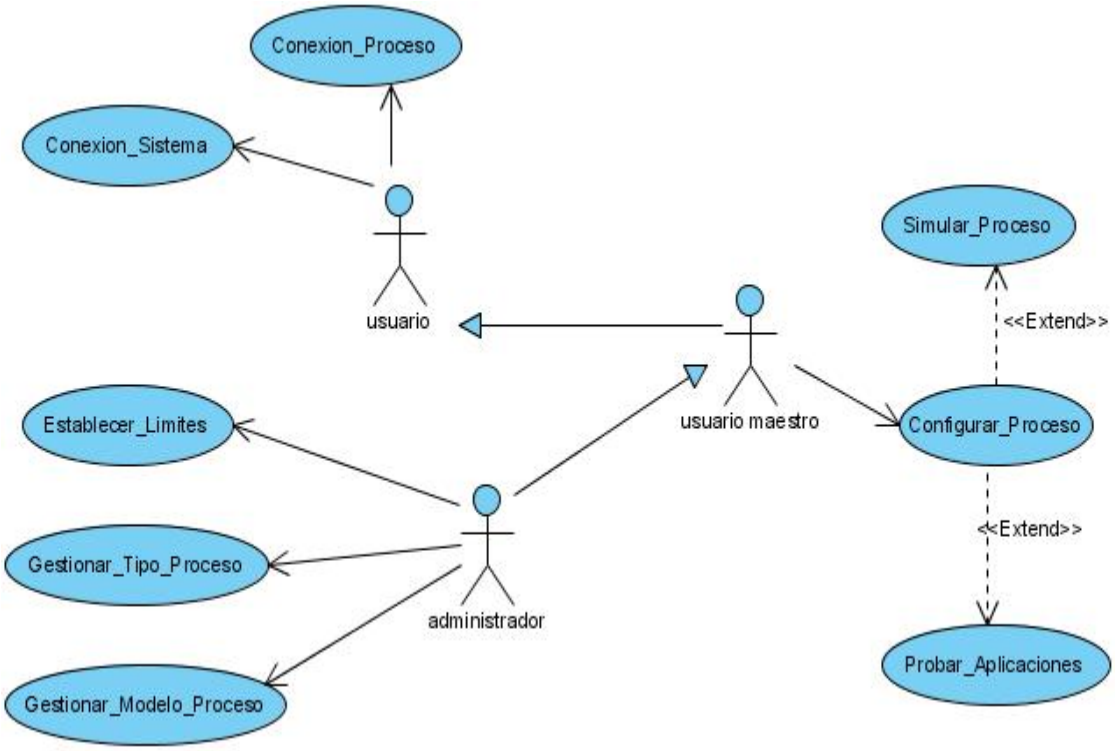


Figura 3 Diagrama de Casos de Uso Críticos del Sistema

2.5.3 Descripción de los casos de uso

Cada uno de los casos de uso que se identificaron, se expondrán a continuación análogamente con su descripción básica para poder facilitar una mayor comprensión de los mismos.

CUS 1: Gestionar_Conexion_Sistema

Caso de Uso – 1	**Gestionar_Conexion_Sistema**
Actores	Usuario
Propósito	Da la posibilidad de que un usuario lleve a cabo la acción de conectarse y desconectarse del sistema.
Resumen	El CUS se inicia cuando el usuario de la aplicación solicita la conexión o desconexión del sistema, el sistema verifica las solicitudes, realiza la acción seleccionada por el usuario, y termina el CUS.
Descripción	
Poscondiciones	1. Usuario conectado al sistema. 2. Usuario desconectado del sistema.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
El usuario selecciona la opción de entrar o de salir del sistema.	El sistema muestra una interfaz con las opciones de Registrarse y Autenticarse, o muestra un mensaje confirmando que el usuario quiere salir del sistema.
Escenario 1: Registrarse	
1. El Usuario selecciona la opción de registrarse. 2. El Usuario introduce los datos solicitados por el sistema.	1.1. El sistema verifica disponibilidad de conexión de usuarios. 1.2. Si hay disponibilidad el sistema muestra el formulario para que el usuario se registre. 2.1. El sistema verifica los datos introducidos por el usuario. 2.2 Si el sistema no encuentra datos semejantes en la base de datos, crea un nuevo usuario con los datos introducidos. 2.3. El sistema introduce al usuario al módulo de usuarios y culmina el caso de uso.
Cursos Alternos	1.2. Si no hay disponibilidad el sistema muestra un mensaje informándole al usuario que no hay disponibilidad de conexión, y culmina el caso de uso. 2.2 Si el sistema encuentra datos semejantes muestra un mensaje al usuario informándole que ese usuario ya esta conectado al sistema e indica al usuario retornar a la acción dos.
Escenario 2: Autenticarse	
1. El Usuario selecciona la opción de autenticarse. 2. El Usuario introduce los datos solicitados por el sistema.	1.1. El sistema muestra el formulario para la autenticación del Usuario. 2.1. El sistema verifica los datos introducidos por el Usuario. 2.2. Si los datos introducidos son correctos el sistema le da acceso al Usuario al módulo de administrador y termina el caso de uso.

Cursos Alternos:	2.2. Si los datos introducidos por el usuario son incorrectos el sistema muestra un mensaje de error e indica al usuario retornar a la acción 2.
Escenario 3: Desconexión del sistema	
1. El Usuario solicita salir del sistema. 2. El Usuario confirma o no que quiere salir del sistema.	1.1. El sistema verifica que el Usuario no este conectado a un proceso. 1.2. Si el usuario no esta conectado a un proceso, el sistema muestra un mensaje de advertencia para la acción a realizar. 2. Si el Usuario confirma que desea salir, el sistema lo desconecta del sistema y termina el caso de uso.
Cursos Alternos:	1.2. Si el usuario esta conectado a un proceso, el sistema le informa que no puede desconectarse mientras no se desconecte del proceso antes, y lo envía a la acción 1. 2. Si el Usuario cancela el mensaje termina el caso de uso sin ejecutar ninguna acción.
Referencia:	R1
Prioridad:	Crítico

Ver **Anexos 17, 18 y 19.**

CUS 2: Gestionar_Conexion_Proceso

Caso de Uso – 2	**Gestionar_Conexion_Proceso**
Actores	Usuario
Propósito	Da la posibilidad de que un usuario lleve a cabo la acción de conectarse y desconectarse de un proceso.
Resumen	El CUS se inicia cuando el usuario de la aplicación solicita la conexión o desconexión a un proceso, el sistema verifica las solicitudes y realiza la acción seleccionada por el usuario, y termina

	el CUS.
Descripción	
Poscondiciones	1. Usuario conectado al proceso. 2. Usuario desconectado de un proceso. 3. Usuario maestro asignado a un proceso.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario solicita la opción de conectarse o desconectarse de un proceso.	1.1 El sistema muestra una interfaz con los procesos, tanto activos como inactivos o muestra un mensaje confirmando que el usuario quiere salir del proceso.
Escenario 1: Conexión a un proceso activo	
1. El Usuario solicita la conexión a un proceso activo.	1.1. El sistema verifica la disponibilidad de conexión para dar acceso al Usuario. 1.2 Si hay disponibilidad de conexión el sistema da acceso al Usuario introduciéndolo a la interfaz donde se esta simulando el proceso, y culmina el caso de uso.
Cursos Alternos	1.2 Si no hay disponibilidad de conexión el sistema va a mostrar un mensaje al Usuario informándole que no hay capacidad de conexión, que lo intente más tarde, y el caso de uso termina sin ninguna acción.
Escenario 2: Conexión a un proceso inactivo	
1. El Cliente solicita la conexión a un proceso inactivo.	1.1. El sistema da conexión al usuario. 1.2. El sistema asigna al usuario la responsabilidad de usuario maestro del proceso. 1.3. El sistema introduce al usuario en la interfaz de Configurar el proceso, y culmina el caso de uso.
Cursos Alternos:	

Escenario 3: Desconexión de un proceso	
1. El Usuario solicita la desconexión del proceso. 2. El Usuario confirma o no la opción de desconexión.	1.1 El sistema muestra un mensaje de advertencia para la acción a realizar. 2.1. Si el Usuario acepta el sistema busca si el usuario es usuario maestro del proceso. 2.2. Si el usuario es usuario maestro el sistema lo desconecta del proceso y asigna un nuevo usuario maestro para el proceso, y culmina el caso de uso.
Cursos Alternos:	2. Si el usuario cancela el mensaje de advertencia, culmina el caso de uso sin realizar ninguna acción. 2.1. Si el usuario no es usuario maestro, el sistema lo desconecta del proceso y culmina el caso de uso.
Referencia:	R2
Prioridad:	Crítico.

Ver **Anexos 20 y 22.**

CUS 3: Desconectar_Usuario_Proceso. Ver **Anexo 3.**
CUS 4: Modificar_Contraseña. Ver **Anexo 4.**
CUS 5: Gestionar_Rol_Administrador. Ver **Anexo 5.**

CUS 6: Establecer_Limites

Caso de Uso – 6	Establecer_Limites
Actores	Administrador
Propósito	Permitir al Administrador modificar los datos de los límites en cuanto a la cantidad de usuarios conectados al sistema o a un proceso y la cantidad de procesos activos simultáneamente.
Resumen	El CUS se inicia cuando el Administrador de la aplicación selecciona la opción de Establecer Límite, luego selecciona el tipo de límite que

	va a modificar, introduce los datos necesarios, el sistema realiza la acción seleccionada por el Administrador y termina el CUS.
Descripción	
Poscondiciones	1. Información de los límites adicionada a la Base de Datos.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador selecciona la opción de Establecer Límite.	1.1 El sistema muestra las opciones de Limitar cantidad de usuarios del sistema, Limitar cantidad de usuarios en proceso y Limitar cantidad de procesos activos.
Escenario 1: Limitar cantidad de usuarios del sistema	
1. El administrador selecciona la opción de limitar el número de usuarios conectados. 2. El administrador entra los datos solicitados por el sistema.	1.1. El sistema muestra un formulario a llenar para hacer la limitación. 2.1. El sistema verifica los datos entrados. 2.2. Si los datos están correctos guarda la información en la Base de Datos correspondiente, y termina el caso de uso.
Cursos Alternos	2.2. Si los datos introducidos por el administrador son incorrectos el sistema muestra un mensaje de error indicando donde está el dato erróneo e indica al Administrador retornar a la acción 2.
Escenario 2: Limitar cantidad de usuarios en proceso.	
1. El administrador selecciona la opción de limitar el número de usuarios conectados a un proceso. 2. El administrador entra los datos para establecer límites en los procesos que desee.	1.1. El sistema muestra un listado con los modelos de los procesos existentes en el sistema. 2.1. El sistema verifica los datos entrados. 2.2. Si los datos están correctos guarda la información en la Base de Datos correspondiente, y termina el caso de uso.
Cursos Alternos:	2.2. Si los datos introducidos por el administrador son incorrectos el sistema muestra un mensaje de error indicando

	donde está el dato erróneo e indica al Administrador retornar a la acción 2.
Escenario 3: Limitar cantidad de procesos activos.	
1. El administrador selecciona la opción de limitar el número de procesos activos. 2. El administrador entra los datos solicitados por el sistema.	1.1. El sistema muestra un formulario a llenar para hacer la limitación. 2.1. El sistema verifica los datos entrados. 2.2. Si los datos están correctos guarda la información en la Base de Datos correspondiente, y termina el caso de uso.
Cursos Alternos:	2.2. Si los datos introducidos por el administrador son incorrectos el sistema muestra un mensaje de error indicando donde está el dato erróneo e indica al Administrador retornar a la acción 2.
Referencia:	R6
Prioridad:	Crítico

Ver **Anexo 23.**

CUS 7: Gestionar_Tipo_Proceso

Caso de Uso – 7	Gestionar_Tipo_Proceso
Actores	Administrador
Propósito	Permite al administrador gestionar (adicionar, modificar o eliminar) tipos de procesos.
Resumen	El CUS se inicia cuando el administrador de la aplicación selecciona la opción de Gestionar Tipo de Proceso, luego selecciona el tipo de gestión, introduce los datos necesarios, el sistema realiza la acción seleccionada por el administrador y termina el CUS.
Descripción	
Poscondiciones	1. Información del tipo de proceso adicionada a la Base de Datos. 2. Información del tipo de proceso modificada en la base de Datos.

	3. Información del tipo de proceso eliminada de la base de Datos.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador selecciona la opción de Gestionar Tipo de Proceso.	1.1 El sistema muestra las opciones de Adicionar Tipo de Proceso, Modificar Tipo de Proceso y Eliminar Tipo de Proceso.
Escenario 1: Adicionar Tipo de Proceso	
1. El Administrador selecciona la opción de Adicionar Tipo de Proceso. 2. El administrador entra los datos solicitados por el sistema.	1.1. El sistema muestra el formulario a completar para la adición de un nuevo tipo de proceso. 2.1. El sistema verifica los datos introducidos por el administrador. 2.2. Si los datos introducidos son correctos el sistema adiciona dicha información en la Base de Datos correspondiente y termina el caso de uso.
Cursos Alternos	2.2. Si los datos introducidos por el administrador son incorrectos el sistema muestra un mensaje de error indicando donde está el dato erróneo e indica al usuario retornar a la acción 2.
Escenario 2: Modificar Tipo de Proceso	
1. El Administrador selecciona la opción de Modificar Tipo de Proceso. 2. El Administrador selecciona el tipo de proceso a modificar. 3. El Administrador realiza los cambios necesarios a los datos	1.1. El sistema muestra un listado con los tipos de procesos existentes en la base de datos. 2.1. El sistema localiza los datos del tipo de proceso y los muestra, listos para modificar. 3.1. El sistema verifica los datos modificados por el administrador. 3.2. Si los datos están correctos el sistema actualiza los datos del tipo de proceso en la Base de Datos correspondiente y termina el caso de uso.

Cursos Alternos:	3.2. Si los datos introducidos por el administrador son incorrectos el sistema muestra un mensaje de error indicando donde está el dato erróneo e indica al usuario retornar a la acción 3.
Escenario 3: Eliminar Tipo de Proceso	
1. El Administrador selecciona la opción de Eliminar Tipo de Proceso. 2. El Administrador selecciona el Tipo de Proceso a eliminar. 3. El Administrador selecciona la opción Eliminar Tipo de Proceso. 4. El Administrador confirma si quiere o no eliminar el Tipo de Proceso.	1.1. El sistema muestra un listado con los Tipos de procesos existentes en la Base de Datos. 2.1. El sistema localiza los datos del Tipo de proceso y los muestra, listos para eliminar. 3.1. El sistema muestra un mensaje de advertencia para la acción a realizar. 4.1. Si el administrador acepta el sistema elimina los datos del Tipo de proceso seleccionado y culmina el caso de uso.
Cursos Alternos:	3.1 Si el administrador cancela la acción se culmina el caso de uso sin ejecutar ninguna acción.
Referencia:	R7
Prioridad:	Crítico

Ver **Anexo 24**.

CUS 8: Gestionar_Modelo_Proceso

Caso de Uso – 8	Gestionar_Modelo_Proceso
Actores	Administrador
Propósito	Permite al administrador gestionar (adicionar, modificar o eliminar) modelos de procesos.
Resumen	El CUS se inicia cuando el administrador de la aplicación selecciona la opción de Gestionar Modelo de Proceso, luego selecciona el tipo de gestión, introduce los datos necesarios, el sistema realiza la

	acción seleccionada por el administrador y termina el CUS.
Descripción	
Poscondiciones	1. Información del modelo de proceso adicionado a la Base de Datos. 2. Información del modelo de proceso modificado en la base de Datos. 3. Información del modelo de proceso eliminada de la base de Datos.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador selecciona la opción de Gestionar Modelo de Proceso.	1.1 El sistema muestra las opciones de Adicionar Modelo de Proceso, Modificar Modelo de Proceso y Eliminar Modelo de Proceso.
Escenario 1: Adicionar Modelo de Proceso	
1. El Administrador selecciona la opción de Adicionar Modelo de Proceso. 2. El Administrador entra los datos solicitados por el sistema.	1.1. El sistema muestra el formulario a completar para la adición de un nuevo modelo de proceso. 2.1. El sistema verifica los datos introducidos por el administrador. 2.2. Si los datos introducidos son correctos el sistema adiciona dicha información en la Base de Datos correspondiente y termina el caso de uso.
Cursos Alternos	2.2. Si los datos introducidos por el administrador son incorrectos el sistema muestra un mensaje de error indicando donde está el dato erróneo e indica al usuario retornar a la acción 2.
Escenario 2: Modificar Modelo de Proceso	
1. El Administrador selecciona la opción de Modificar Modelo de Proceso. 2. El Administrador selecciona el modelo de	1.1. El sistema muestra un listado con los modelos de procesos existentes en la base de datos.

proceso a modificar. 3. El Administrador realiza los cambios necesarios a los datos	2.1. El sistema localiza los datos del modelo de proceso y los muestra, listos para modificar. 3.1. El sistema verifica los datos modificados por el administrador. 3.2. Si los datos están correctos el sistema actualiza los datos del modelo del proceso en la Base de Datos correspondiente y termina el caso de uso.
Cursos Alternos:	3.2. Si los datos introducidos por el administrador son incorrectos el sistema muestra un mensaje de error indicando donde está el dato erróneo e indica al usuario retornar a la acción 3.
Escenario 3: Eliminar Modelo de Proceso	
1. El Administrador selecciona la opción de Eliminar Modelo de Proceso. 2. El Administrador selecciona el Modelo de Proceso a eliminar. 3. El Administrador selecciona la opción Eliminar Modelo de Proceso. 4. El Administrador confirma si quiere o no eliminar el Modelo de Proceso.	1.1. El sistema muestra un listado con los Modelos de procesos existentes en la Base de Datos. 2.1. El sistema localiza los datos del Modelo de proceso y los muestra, listos para eliminar. 3.1. El sistema muestra un mensaje de advertencia para la acción a realizar. 4.1. Si el administrador acepta el sistema elimina los datos del Modelo de proceso seleccionado y culmina el CUS.
Cursos Alternos:	3.1 Si el administrador cancela la acción se culmina el CUS sin ejecutar ninguna acción.
Referencia:	R8
Prioridad:	Crítico

Ver **Anexos 25, 26, 27, 28 y 29.**

CUS 9: Configurar_Proceso

Caso de Uso – 9	Configurar_Proceso	
Actores	Usuario Maestro.	
Propósito	Permitir al Usuario Maestro configurar un proceso	
Resumen	El CUS se inicia cuando el Usuario Maestro selecciona la opción de Configurar Proceso, introduce los datos necesarios, y el sistema realiza la acción seleccionada por el Usuario y termina el CUS.	
Descripción		
Poscondiciones	Proceso Configurado.	
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Sistema**
1. El Usuario Maestro selecciona la opción de Configurar Proceso. 2. El Usuario Maestro configura e introduce los datos solicitados por el sistema.		1.1 El sistema localiza los datos del modelo del proceso donde se encuentra el Usuario Maestro. 1.2 El sistema muestra los datos del modelo del proceso para ser configurados. 2.1 EL sistema verifica los datos introducidos por el Usuario Maestro. 2.2 Si los datos introducidos son correctos el sistema adiciona dicha información a la Base de Datos correspondiente y termina el caso de uso.
Cursos Alternos:		2.2 Si los datos entrados por el Usuario Maestro son incorrectos el sistema muestra un mensaje de error indicando donde esta el dato erróneo e indica al Usuario Maestro retornar a la acción 2.
Referencia:	R9	

Prioridad:	Crítico

Ver **Anexo 30.**

CUS 10: Simular_Proceso

Caso de Uso – 10	Simular_Proceso (extendido de Configurar_Proceso)
Actores	Usuario Maestro.
Propósito	Permite al Usuario Maestro simular un proceso.
Resumen	El CUS se inicia cuando el Usuario Maestro de un proceso selecciona la opción de Simular Proceso, el sistema realiza la acción seleccionada por el Usuario Maestro y termina el CUS.
Descripción	
Poscondiciones	Proceso Simulado.
Flujo Normal de Eventos	

Acción del Actor	**Respuesta del Sistema**
1. El Usuario Maestro selecciona la opción de Simular Proceso.	1.1. El sistema localiza los datos del proceso donde se encuentra el Usuario Maestro y verifica que el proceso este configurado. 1.2. Si el proceso esta configurado, el sistema verifica si se puede efectuar la simulación. 1.3. Si la simulación se puede efectuar el sistema muestra un mensaje para comenzar la simulación.
2. El Usuario Maestro confirma si quiere comenzar la simulación o no.	2.1. Si el Usuario Maestro acepta el sistema comienza la simulación. 2.2. El sistema va a mostrar de forma gráfica los valores que van tomando las variables de salida del proceso a medida que transcurre la simulación.
3. El Usuario Maestro puede o no cambiar los valores de las variables de entradas.	3.1. Si el Usuario Maestro cambia los valores

4. El Usuario Maestro selecciona la opción de terminar la simulación.	de las variables el sistema chequea si los nuevos datos son correctos. 3.2. Si los datos son correctos el sistema guarda los cambios y continúa la simulación con los nuevos valores de las variables de entradas. 4.1. El sistema termina la simulación y culmina el caso de uso.
Cursos Alternos:	1.2 Si el proceso no esta configurado el sistema muestra un mensaje indicándole al Usuario maestro que debe configurar el proceso antes, y lo envía a la acción 1. 1.3. Si no se puede efectuar la simulación el sistema muestra un mensaje informándole al Usuario Maestro que la simulación no se puede efectuar, que intente mas tarde, y termina el caso de uso sin realizar ninguna acción. 2.1 Si el Usuario Maestro cancela la opción se termina el caso sin realizar ninguna acción. 3.1. Si el Usuario Maestro no cambia las variables de entradas la simulación continua sin ningún cambio. 3.2. Si los nuevos datos entrados por el Usuario Maestro son incorrectos el sistema muestra un mensaje de error indicando donde esta el dato erróneo e indica al Cliente Maestro retornar a la acción 3.
Referencia:	R10
Prioridad:	Crítico

Ver **Anexo 32.**

CUS 11: Probar_Aplicacion

Caso de Uso – 11	Probar_Aplicacion. (extendido de Configurar_Proceso)
Actores	Usuario Maestro
Propósito	Permitir al Usuario Maestro probar aplicaciones en tiempo real.
Resumen	El CUS se inicia cuando el Usuario Maestro de un proceso selecciona la opción de Probar aplicaciones, introduce los datos necesarios, el sistema realiza la acción seleccionada por el Usuario Maestro y termina el CUS.
Descripción	
Poscondiciones	Aplicación probada
Flujo Normal de Eventos	

Acción del Actor	**Respuesta del Sistema**
1. El Usuario Maestro selecciona la opción de Probar Aplicaciones en tiempo real.	1.1. El sistema localiza los datos del proceso donde se encuentra el Usuario Maestro. 1.2. El sistema verifica que el proceso este configurado. 1.3. Si el proceso esta configurado, el sistema verifica si se puede efectuar la simulación. 1.4. Si la simulación se puede efectuar el sistema establece conexión con el autómata. 1.5. Si la conexión fue establecida el sistema muestra un mensaje de advertencia para la acción a realizar.
2. El Usuario Maestro confirma si quiere comenzar a probar la aplicación o no.	2.1. Si el Usuario Maestro acepta el sistema comienza a correr la aplicación con el autómata. 2.2. El autómata va a realizar el control del proceso a medida que transcurre este. 2.3. El sistema va a mostrar de forma gráfica los valores que van tomando las variables de

	salida a medida que transcurre la simulación.
3. El Usuario Maestro puede o no cambiar los valores de las variables de entradas.	3.1. Si el Usuario Maestro cambia los valores de las variables el sistema chequea si los nuevos datos son correctos. 3.2. Si los datos son correctos el sistema guarda los cambios y continúa la simulación con los nuevos valores de las variables de entradas.
4. El Usuario Maestro selecciona la opción de terminar el proceso.	4.1. El sistema termina el proceso y culmina el caso de uso.
Cursos Alternos:	1.3. Si el proceso no esta configurado el sistema muestra un mensaje indicándole al Usuario maestro que debe configurar el proceso antes, y lo envía a la acción 1. 1.4. Si no se puede efectuar la simulación el sistema muestra un mensaje informándole al Usuario Maestro que la simulación no se puede efectuar, que intente mas tarde, y termina el caso de uso sin realizar ninguna acción. 1.5. Si no se pudo conectar con el autómata el sistema mostrara un mensaje de error, e indicara al Usuario Maestro volver a la acción 1. 2.1. Si el Usuario Maestro cancela la opción, el caso termina sin realizar ninguna acción. 3.1. Si el Usuario Maestro no cambia las variables de entradas la simulación continua sin ningún cambio. 3.2. Si los nuevos datos entrados por el Usuario Maestro son incorrectos el sistema muestra un mensaje de error indicando

	donde esta el dato erróneo e indica al Cliente Maestro retornar a la acción 3.
Referencia:	R11
Prioridad:	Crítico

Ver **Anexo 32.**

CUS 12: Graficar. Ver **Anexo 6.**

CUS 13: Gestionar _Controlador

Caso de Uso – 13	Gestionar _Controlador
Actores	Administrador
Propósito	Permite al administrador gestionar (adicionar o eliminar) controladores.
Resumen	El CUS se inicia cuando el administrador de la aplicación selecciona la opción de Gestionar Controlador, luego selecciona el tipo de gestión, introduce los datos necesarios, el sistema realiza la acción seleccionada por el administrador y termina el CUS.
Descripción	
Poscondiciones	1. Información del controlador adicionada a la Base de Datos. 2. Información del controlador eliminada de la Base de Datos.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador selecciona la opción de Gestionar Controlador.	1.1 El sistema muestra las opciones de Adicionar Controlador y Eliminar Controlador.
Escenario 1: Adicionar Controlador	
1. El Administrador selecciona la opción de Adicionar Controlador. 2. El Administrador entra los datos solicitados por el sistema.	1.1. El sistema muestra el formulario a completar para la adición de un nuevo controlador. 2.1. El sistema verifica los datos introducidos

	por el administrador. 2.2. Si los datos introducidos son correctos el sistema adiciona dicha información en la Base de Datos correspondiente y termina el caso de uso.
Cursos Alternos	2.2. Si los datos introducidos por el administrador son incorrectos el sistema muestra un mensaje de error indicando donde está el dato erróneo e indica al usuario retornar a la acción 2.
Escenario 2: Eliminar Controlador	
1. El Administrador selecciona la opción de Eliminar Controlador. 2. El Administrador selecciona el Controlador a eliminar. 3. El Administrador selecciona la opción Eliminar Controlador. 4. El Administrador confirma si quiere o no eliminar el Controlador.	1.1. El sistema muestra un listado con los Controladores existentes en la Base de Datos. 2.1. El sistema localiza los datos del Controlador y los muestra, listos para eliminar. 3.1. El sistema muestra un mensaje de advertencia para la acción a realizar. 4.1. Si el administrador acepta el sistema elimina los datos del Controlador seleccionado y culmina el CUS.
Cursos Alternos:	3.1 Si el administrador cancela la acción se culmina el CUS sin ejecutar ninguna acción.
Referencia:	R12
Prioridad:	Crítico

Ver **Anexo 31.**

CUS 14: Activar _Procesos. Ver **Anexo 7.**

CUS 15: Priorizar_Numero_Proceso. Ver **Anexo 8.**

CUS 16: Mostrar_Tipos_Procesos. Ver **Anexo 9.**

CUS 17: Mostrar_Procesos. Ver **Anexo 10.**

CUS 18: Mostrar_Procesos_Activos. Ver **Anexo 11.**

CUS 19: Mostrar_Registro_Usuarios. Ver **Anexo 12.**

CUS 20: Mostrar_Usuario_Proceso. Ver **Anexo 13.**

CUS 21: Mostrar_Proceso_Cliente. Ver **Anexo 14.**

CUS 22: Mostrar_Registro_Variables. Ver **Anexo 15.**

Conclusiones

Luego de un estudio previo de las características del sistema a construir y de realizar todas las actividades previstas para el flujo de Requerimientos como parte del desarrollo del proyecto se puede concluir que:

- La utilización de estrategias de captura de requisitos: como las Entrevistas, permitieron captar todas las necesidades de los clientes y usuarios, lográndose un buen refinamiento de los requisitos.
- Se identificaron los actores que serán los encargados de interactuar con el sistema. Los procesos que resultaron automatizables se modelaron en forma de casos de uso.
- La aplicación de patrones de casos de uso facilitó el modelado y la interpretación de los mismos, obteniéndose un Modelo de Sistema que permitirá al equipo de desarrollo, comprender los procesos, facilitándose el modelado y la implementación del sistema.

Capítulo 3: Análisis y Diseño del sistema

Introducción

En el presente capítulo estarán reflejados los distintos diagramas de clases de análisis y del diseño, así como los diagramas de interacción correspondientes al flujo de trabajo según la metodología RUP. Durante esta etapa es donde se analizan los requisitos descritos, se refinan y estructuran a una comprensión que sea fácil de mantener y ayude al equipo de desarrolladores para poder estructurar el sistema completo, incluyendo además la arquitectura del mismo.

También se abordarán las características técnicas del Nodo Virtual. Es importante decir que el diseño esta hecho para implementarse en el IDE Eclipse, y como lenguaje de programación C++.

Se escogió la Arquitectura en tres capas. Las interfaces de usuario y los formularios que permiten la interacción con el usuario se encuentran en la capa de Aplicación. Las clases encargadas de implementar las reglas del negocio y las restricciones que desea el cliente se encuentran en la capa Lógica o de Negocio, estas son las clases controladoras y entidades que permiten la realización de los casos de uso.

3.1 Análisis

Debido a la complejidad del análisis es importante centrarse en los distintos modelos que van a describir la realización de los requisitos funcionales con mayor profundidad y sin entrar en muchos detalles, de forma tal que se pueda utilizar el lenguaje de los desarrolladores para poder describir los resultados.

A continuación se presentará una breve comparación entre el modelo de casos de uso con el modelo de análisis donde se refleja la trazabilidad directa que hay entre ambos modelos [16]. Ver Anexo 16.

El modelo de análisis está compuesto por artefactos como son: clases del análisis y por realizaciones de casos de uso del análisis.

Las clases del análisis van a representar abstracciones de conceptos, en las cuales deben incluirse atributos y operaciones a un nivel alto, por lo que no se incluye el paso de parámetros ni el tipo de datos.

La realización de los casos de uso del análisis son los diagramas que expresan el comportamiento de las clases del análisis, presentando una trazabilidad de uno a uno. En la realización de casos de uso se expresan los diagramas de interacción donde se comunican las clases, y representa la vista dinámica del sistema. Estos diagramas pueden ser de dos tipos: diagrama de secuencia y de colaboración.

3.1.1 Clases del Análisis

Dentro del artefacto que se genera durante el análisis, las clases del análisis presentan 3 estereotipos básicos que están estandarizados en UML y se utilizan para ayudar al equipo de desarrolladores, como son: las clases de interfaz, las clases controladoras y las clases de entidad, donde cada una de ellas tiene su propio símbolo y funcionalidad. [16]

A continuación se muestra una visión de los diagramas de clases del análisis modelado para cada caso de uso que se incluye en el sistema.

CU Gestionar_Conexion_Sistema. Ver Anexos 33 y 34.

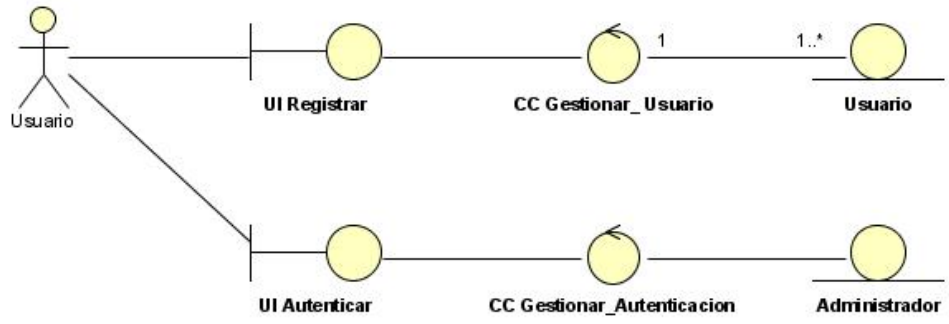


Figura 3 Diagrama de Clases de Análisis. CU Gestionar_Conexion_Sistema

CU Gestionar_Conexión_Proceso. Ver Anexos 35 y 36.

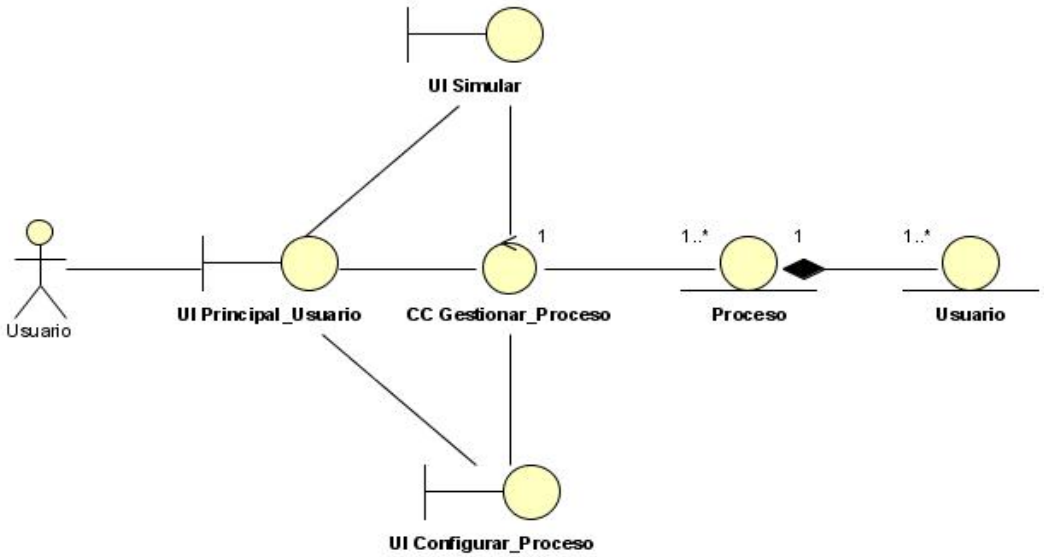


Figura 4 Diagrama de Clases de Análisis. CU Gestionar_Conexion_Proceso

CU Establecer_Limites. Ver Anexo 37.

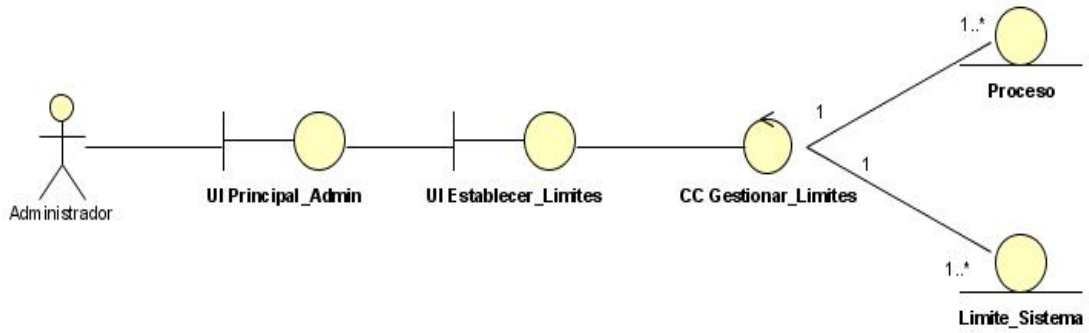


Figura 5 Diagrama de Clases de Análisis. CU Establecer_Limite

CU Gestionar_Controlador. Ver Anexos 38 y 39.

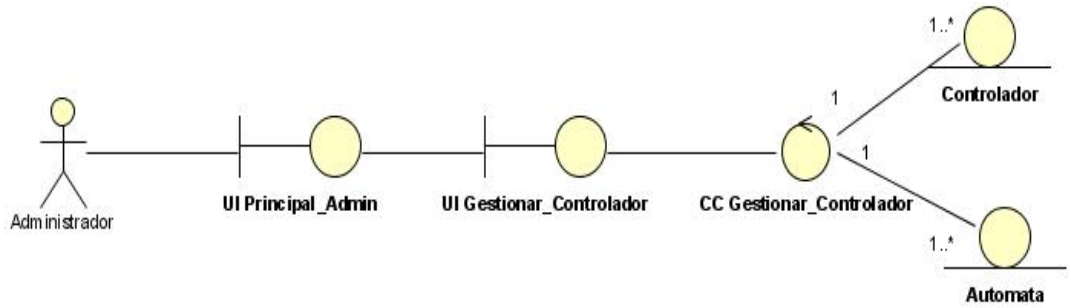


Figura 6 Diagrama de Clases de Análisis. CU Gestionar_Controlador

CU Gestionar_Modelo_Proceso. Ver Anexos 40, 41 y 42.

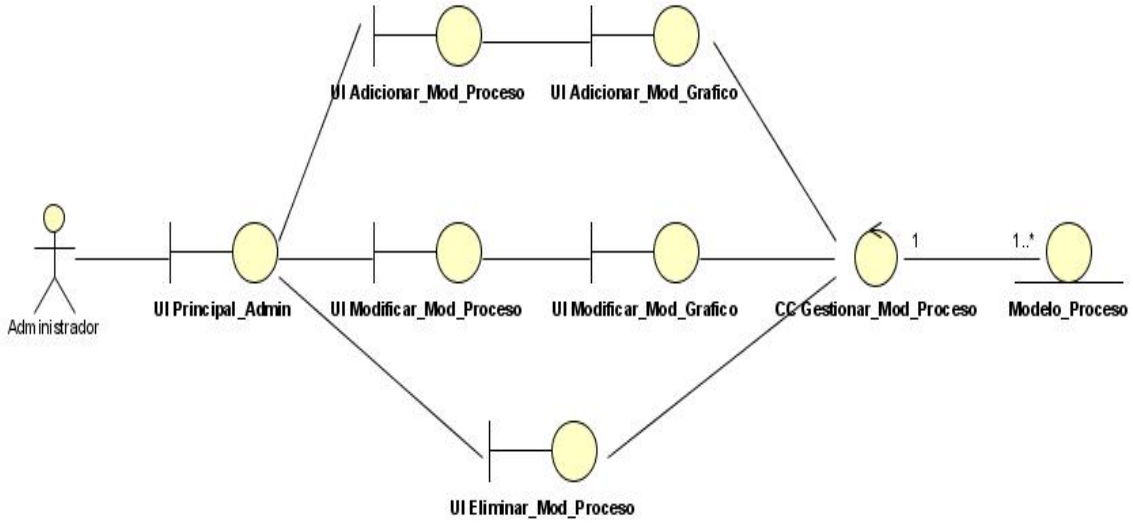


Figura 7 Diagrama de Clases de Análisis. CU Gestionar_Modelo_Proceso

CU Gestionar_Tipo_Proceso. Ver Anexo 43.

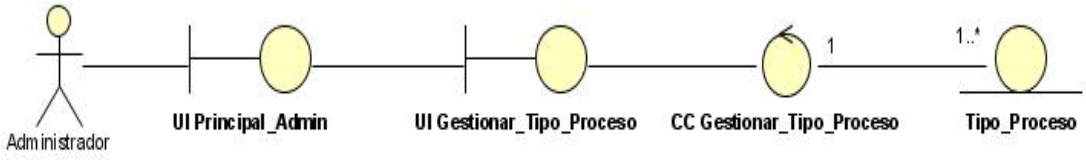


Figura 8 Diagrama de Clases de Análisis. CU Gestionar_Tipo_Proceso

CU Simular_Proceso. Ver Anexo 44.

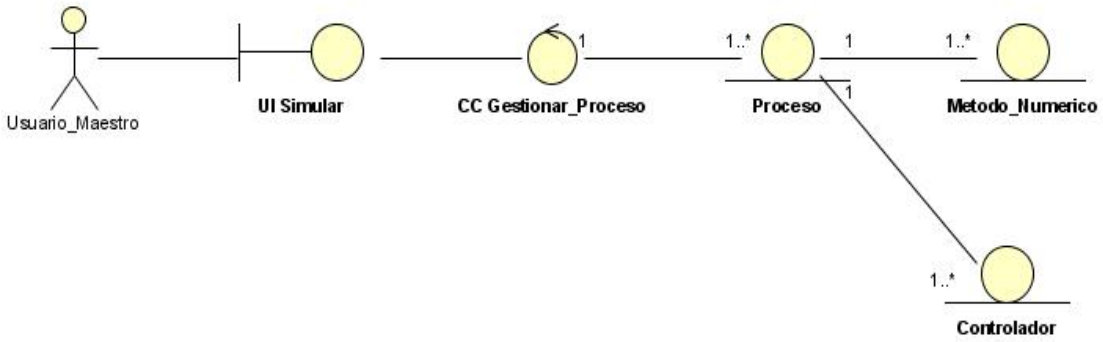


Figura 9 Diagrama de Clases de Análisis. CU Simular_Proceso

CU Configurar_Proceso. Ver Anexo 45.

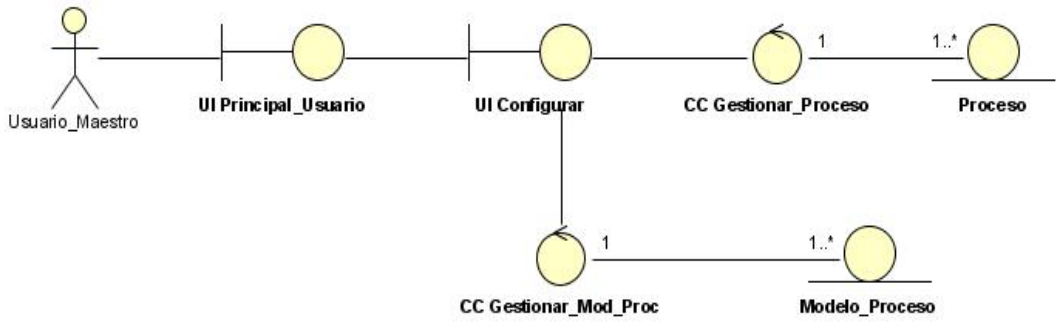


Figura 10 Diagrama de Clases de Análisis. CU Configurar_Proceso

CU Probar_Aplicaciones. Ver Anexo 46.

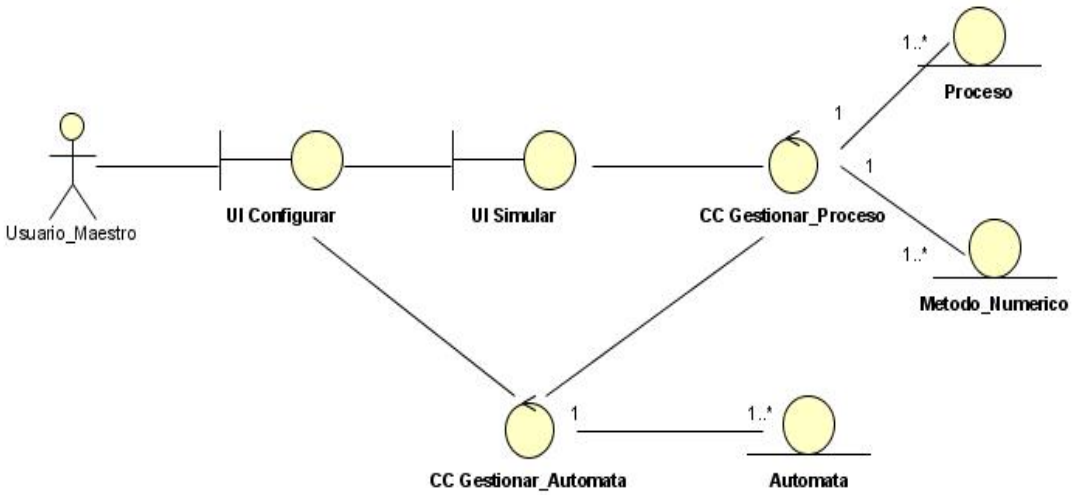


Figura 11 Diagrama de Clases de Análisis. CU Probar_Aplicaciones

3.1.2 Realización de casos de uso del análisis

La realización de los casos de uso del análisis es otro de los artefactos que se generan en el mismo, y colaboran de forma tal que describen como se lleva a cabo y se ejecuta un caso de uso en términos de las clases del análisis y de sus objetos en interacción, por tanto es el que proporciona una trazabilidad directa hacia un caso de uso del modelo de casos de uso.

Por su parte un diagrama de interacción va a expresar la comunicación de las clases del análisis, representando la vista dinámica del sistema y modelado a través de objetos concretos o instancias de las clases ya definidas. Dentro de los diagramas de interacción se pueden destacar los diagramas de secuencia y los de colaboración.

Los diagramas de secuencia ordenan los mensajes de forma tal que colocan los objetos que participan en la iteración y los mensajes que envían y reciben estos objetos en orden de sucesión y en el tiempo. Los diagramas de colaboración son los que construyen los objetos como un nodo de un grafo y a continuación se representan los mensajes que conectan a dichos objetos como si fueran arcos de este grafo, los cuales se envían y se reciben a través de los objetos. [16] Ver Anexos del 33 al 46.

3.2 Diseño

En el diseño al mismo tiempo que se modela el sistema se le da forma a la arquitectura, soportando así todos los requisitos, reglas y restricciones. Un artefacto importante a tener en cuenta como punto de partida para llegar al modelo de diseño es el modelo de análisis. Aunque RUP plantea que no es obligatoria la confección del modelo de análisis, este resulta de gran ayuda y sirve como base para crear el diseño orientado a diferentes plataformas y lenguajes de programación. [16]

3.2.1 Arquitectura definida para el sistema

En el Análisis y Diseño que plantea RUP, trabajan en conjunto para lograr un correcto modelo de diseño y demás artefactos generados en esa disciplina dos trabajadores: el diseñador del sistema y el arquitecto. Es en este momento que se generan tres de las cinco vistas arquitectónicas del sistema, por lo que no se pueden ver separados el diseño y la arquitectura de un software. Es ella quien le da forma al software para que soporte todos los requisitos. [20]

Se necesita una arquitectura robusta, que guíe el proceso de desarrollo, y que defina de manera abstracta, los componentes que llevan a cabo alguna tarea, sus interfaces y la comunicación ente ellos. [20]

Con el objetivo de aislar el negocio de la interfaz del usuario y del manejo de la persistencia aplicaremos en la implementación de la aplicación el estilo arquitectura en capas.

Para la solución del problema estableceremos 3 capas:

Capa de presentación

Es la que interactúa directamente con el usuario, captura la información entrada por éste (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 ésta. Únicamente se comunica con la capa de negocio.

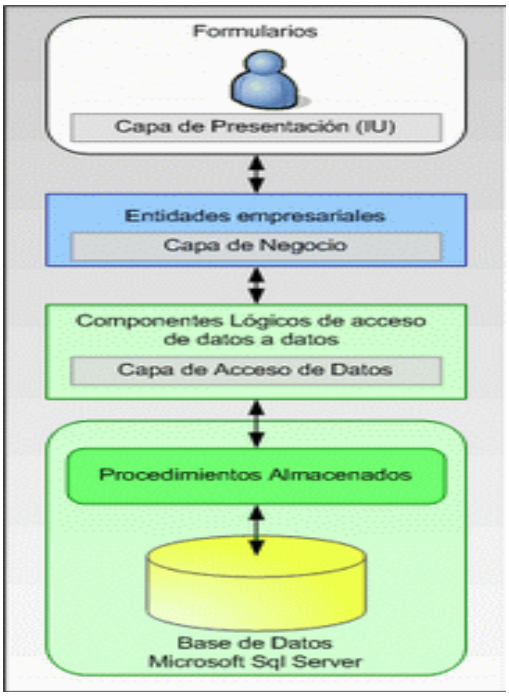
Capa de negocio

Está conformada por los subsistemas, los cuales se ajustan a los requisitos y casos de uso arquitectónicamente significativos. 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 la base de datos y permiten, utilizando los procedimientos almacenados generados, realizar todas las operaciones con la base de datos de forma transparente para la capa de negocio.

Figura 12 Capas de la Arquitectura



En el caso específico del Nodo Virtual la capa de aplicación estaría compuesta por todos los elementos de la interfaz: representaciones gráficas de los diferentes procesos que se simulan, tablas con el historial de los valores de entrada y salida, gráficos que muestren el comportamiento del proceso como modelo matemático. La capa de negocio abarca los subsistemas de Simulación, Asignar rol, Reporte, Conexión y Gestión. Por otro lado todo lo concerniente a la gestión de la persistencia de los datos estaría enmarcado en la capa de acceso a datos.

3.2.2 Descripción de los módulos principales

La modularidad es una de las principales características que debe tener un buen diseño. Es necesario prestar mucha atención cuando se divide el sistema en partes más pequeñas para no incurrir en gastos innecesarios.

El Nodo Virtual va a estar dividido en 5 módulos principales, los cuales interactúan entre ellos para llegar a un propósito final, simular procesos industriales. Los módulos son Conexión, Simulación, Gestión, Reporte y Graficar.

3.2.2.1 Módulo de Conexión

Su función es gestionar la conexión dentro del sistema, tanto la conexión a los procesos como la conexión a las distintas partes del software. Esta conexión se realizará mediante los procesos de registrarse y autenticarse. Si el usuario cuando entra al sistema se autentica, pasara a la interfaz de los administradores, la cual tiene conexión con las principales funcionalidades de este rol; si por el contrario se registra, el sistema lo envía a la interfaz de los usuarios.

3.2.2.2 Módulo de Simulación

Es el módulo principal del sistema. Su función es realizar todo el proceso para simular un proceso industrial o probarlo en tiempo real. Primeramente el usuario (que en este caso es un usuario maestro) tendrá que configurar el proceso que desea simular, luego elegirá entre simularlo o probarlo en tiempo real. Teniendo en cuenta los datos introducidos por el usuario en la configuración, este módulo se encarga de realizar los cálculos pertinentes para llevar a cabo la simulación. En caso de probar aplicaciones, el usuario además de los datos de la configuración adicionara el numero IP del autómata y el puerto por donde se establecerá la conexión. Permitir la interacción del usuario con el proceso que esta siendo simulado o probado, así como mostrar los resultados de la simulación a medida que esta ocurre son funcionalidades de este módulo. Una vez terminada la simulación se brinda la posibilidad de ver un reporte donde se encuentran los datos del proceso durante esta.

3.2.2.4 Módulo de Gestión

Este módulo es el encargado de gestionar toda la información necesaria para un buen funcionamiento del sistema. En el se van a encontrar las funcionalidades para gestionar los tipos de procesos, los modelos de procesos y los controladores. Es el encargado de establecer los límites de conexión al sistema y a los procesos, así como desconectar a los usuarios que no tengan un comportamiento adecuado dentro del sistema, de esta forma se vela porque el sistema tenga un buen rendimiento.

3.2.2.5 Módulo de Reporte

La función de este módulo es mostrar la información que brinda el sistema, tanto importante para los usuarios, como son los tipos de procesos que se encuentran en el sistema y los procesos que se encuentran dentro de ellos, como importante para el administrador, como son el listado de los usuarios conectados, el listado de los usuarios maestros, el listado de los procesos activos, el listado de los usuarios por procesos, entre otros. Esta información ayudará al administrador a llevar el control del funcionamiento del sistema.

3.2.2.6 Módulo de Graficar

Este módulo es el encargado de gestionar toda la información gráfica del sistema. Los modelos de los procesos van a tener asociados a ellos un modelo gráfico en el cual se va a ver representado el proceso. Una vez comenzada la simulación este modelo debe ir mostrando de forma grafica los cambios que ocurren en el, haciendo mas dinámica la aplicación y mas amena la interacción con el usuario.

3.2.3 Justificación de los Patrones de Diseño utilizados

En el modelo de Diseño se proponen utilizar patrones pertenecientes a los Gof, los que se pueden encontrar en cualquier bibliografía que trate de ese tema, y que fueron analizados en el capitulo 1. Es importante decir que se tuvieron en cuenta los patrones de asignación de responsabilidades (GRASP), para que el diseño tuviese Bajo Acoplamiento, Alta Cohesión y que cada clase experta en algún tipo de información fuese la encargada de realizar las operaciones con la misma, logrando con eso un mejor funcionamiento del sistema.

Después de realizar un estudio de los patrones, ver los problemas que resuelven y teniendo en cuanta las características del sistema, se seleccionaron tres patrones que encajan para darle solución a la problemática de diseñar el sistema haciendo uso de soluciones probadas satisfactoriamente en casos anteriores.

Los patrones Gof usados en el diseño del sistema son:

3.2.3.1 Patrón Solitario (Singleton)

Con este patrón se garantiza una única instancia de aquellas clases que se desee tener una sola en toda la aplicación, proporcionando un punto de acceso global a dichas clases. Tiene como ventajas que reduce el espacio de nombres y es una mejora sobre las variables globales. Es usado debido a la necesidad de trabajar con el mismo objeto en distintos momentos y distintos módulos. Específicamente, para realizar la simulación es necesario configurar el modelo del proceso que se desea

simular, creándose un objeto llamado proceso, y esa misma instancia va a ser llamada desde los módulos Gestión y Conexión para realizar funcionalidades como conexión a un proceso o establecer los limites de usuarios por proceso. Se utiliza también en las clases fachadas de los subsistemas, garantizando una única instancia de ellas, esto suele hacerse cuando se aplica el Patrón Fachada, que explicaremos a continuación.

3.2.3.2 Patrón Fachada (Facade)

Proporciona una interfaz sencilla unificada para un conjunto de clases o subsistemas, siendo más fácil de usar. Permite reducir la complejidad y minimizar las dependencias, el acceso de los clientes a los subsistemas es por medio de la clase fachada, ella es la encargada de reenviar las peticiones a los objetos de los subsistemas, por lo que no se accede directamente a los mismos, ocultando la complejidad de ellos. Este patrón favorece a un Bajo Acoplamiento entre los clientes y los subsistemas, respondiendo a uno de los patrones GRASP, va a permitir variar las clases internas, de manera transparente a los clientes que las usan; favoreciendo de esta forma a la división en capas de la aplicación. Como se dijo anteriormente estas clases van a utilizar el patrón Solitario lo que va a permitir un acceso global a ellas.

3.2.3.1 Patrón Observador (Observer)

Su propósito es definir una dependencia uno a muchos entre objetos, de modo que cuando un objeto cambia de estado, todos los que de él dependen son notificados y actualizados automáticamente. Se mantiene la dependencia entre los objetos sin necesidad de conocer al otro. Se usó por la existencia de una abstracción con dos aspectos, uno dependiente del otro, y se necesitaba encapsularlos en objetos separados para reutilizarlos de forma independiente. En el nodo virtual la clase encargada de almacenar la información de los usuarios es observadora de la clase encargada de almacenar la información de los procesos. En la clase que guarda los datos de los procesos (Proceso) se encuentra una lista de los usuarios conectados a cada proceso, el usuario 0 en esa lista va a llevar la responsabilidad de ser el usuario maestro del proceso. Cada vez que el proceso cambia de usuario maestro se debe notificar a la clase Usuario para que se actualice. Dichas clases implementan y heredan respectivamente las clases Subject y Observer, quienes son los objetos abstractos y las primeras son los objetos concretos. El patrón Observador también proporciona Bajo Acoplamiento al igual que el Fachada.

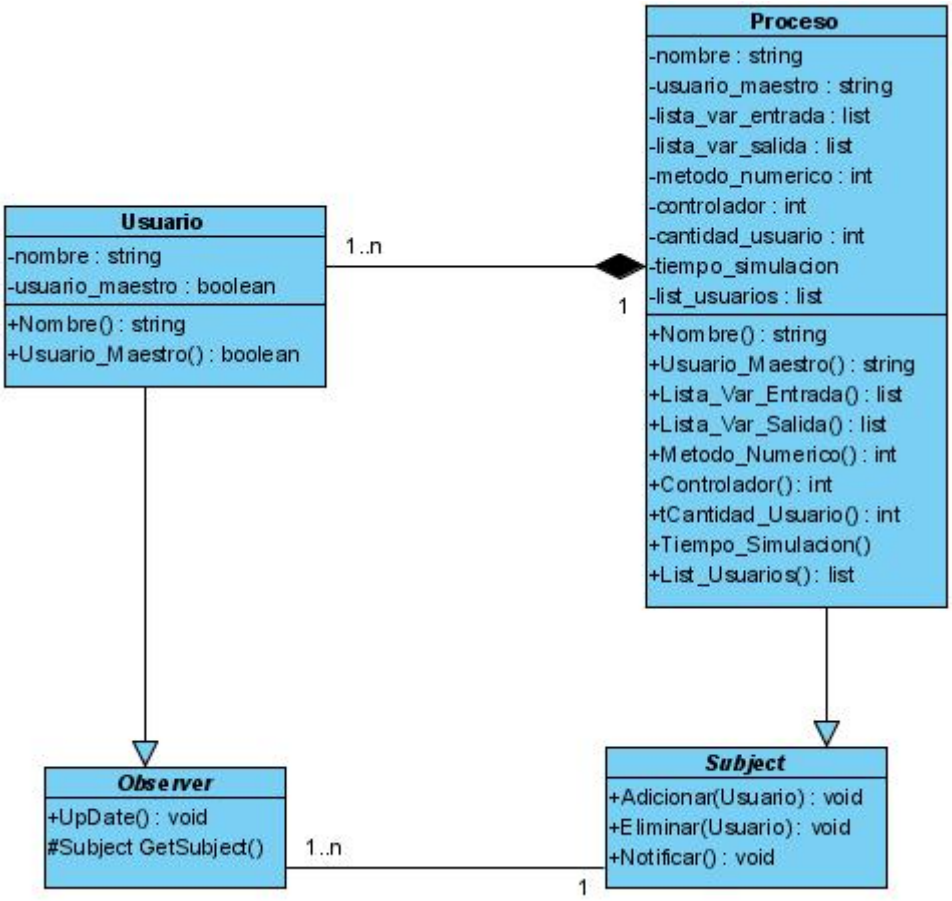


Figura 13 Modelo de Clases de Diseño. Patrón Observador

3.2.4 Modelo de Diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tiene impacto en el sistema a considerar. Además, el modelo de diseño sirve de abstracción a la implementación del sistema y es, de ese modo, utilizada como una entrada fundamental a las actividades de implementación. [16]

En este caso el modelo de diseño esta conformado por un sistema de diseño que a su vez lo conforman varios subsistemas.

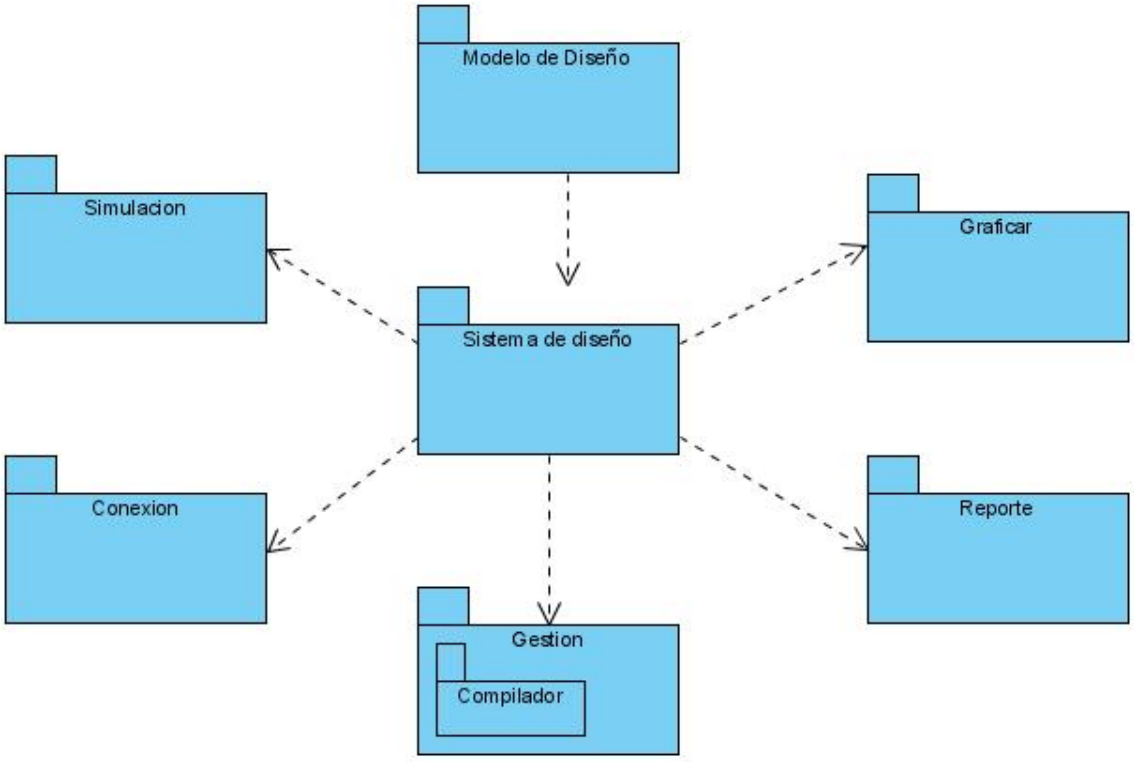


Figura 14 Modelo de Diseño del Sistema.

3.2.4.1 Subsistemas de diseño

El Modelo de Diseño está compuesto por subsistemas de diseño que interactúan entre si para realizar los casos de uso.

A continuación se explica la funcionalidad de los principales subsistemas, se muestra el diagrama de las clases que lo conforman y se describen las clases más importantes. Los subsistemas Graficar y Reportes serán desarrollados en una próxima iteración del diseño.

Subsistema Presentación

Este subsistema se encuentra en la capa de presentación de la arquitectura del sistema, en él solamente se encuentran aquellos componentes que permiten interactuar con el cliente, es decir formularios o ventanas para mostrar o capturar datos.

El diseño propuesto para este subsistema, al igual que los demás, está vinculado estrechamente a la plataforma y lenguaje de programación en el cual se va a implementar. Se va a hacer uso de las vistas para mostrar diferentes interfaces. Esto permite que las vistas sean llamadas con la ocurrencia de un evento y se muestren dentro de una parte específica de la forma principal, evitando así la creación de varias formas.

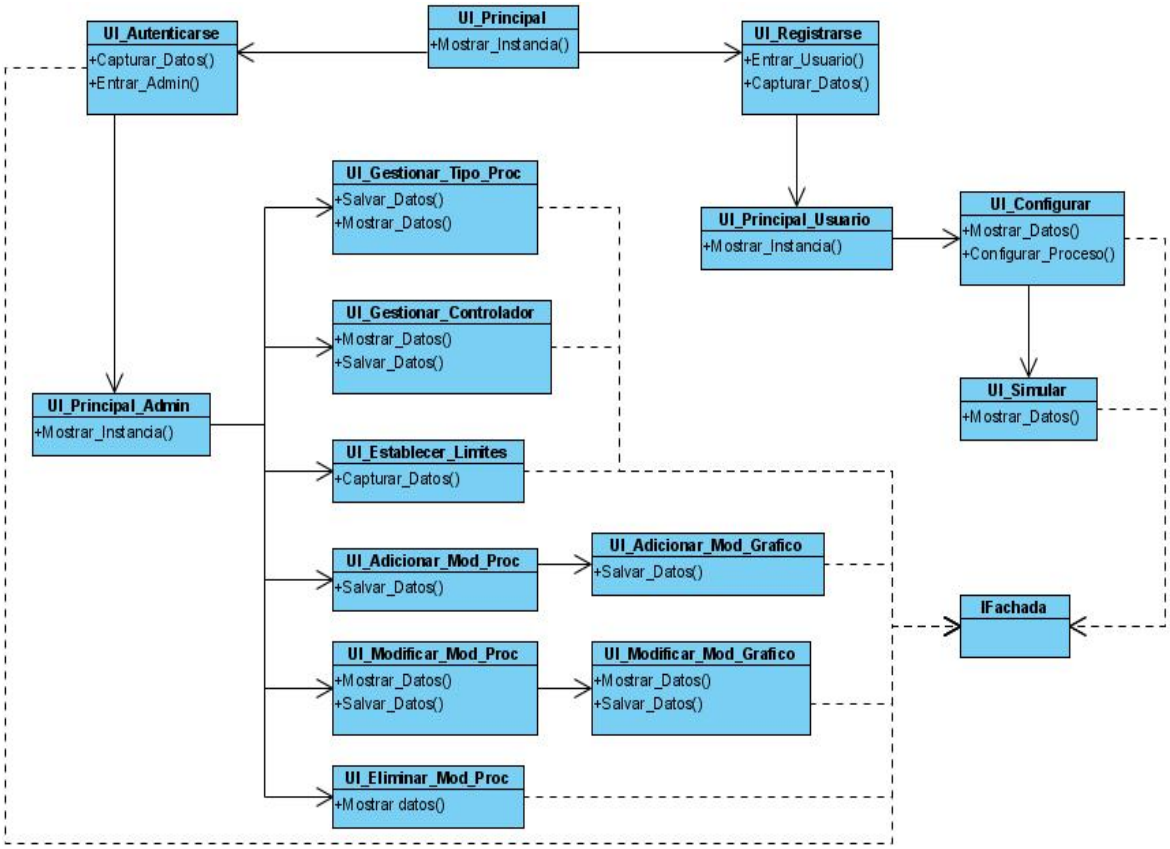


Figura 15 Diagrama de Clases de Diseño. Subsistema Presentación

- UI_Principal: Su función es mejorar la interfaz gráfica y hacer más amena la interacción con el usuario. Es un splash que carga al iniciar la aplicación con una imagen representativa del software. Dentro de ella se encuentran dos hipervínculos que dan paso a las interfaces para registrarse y autenticarse. Ver **Anexo 17**.
- UI_Registrarse: La función de esta forma es el control de acceso de los usuarios al sistema. Debido a que cualquier usuario puede tener acceso al mismo, esta forma solo tendrá un TextBox que permite entrar el nombre del usuario, así como un botón que manda a adicionar al usuario, responsabilidad de otras clases que se encuentran en otras capas del sistema. Ver **Anexo 18**.
- UI_Autenticarse: Los administradores tienen acceso a información que es importante para el rendimiento del sistema y que no es visible para los usuarios, es por eso que la función de esta forma es el control de acceso de los administradores al sistema. Tiene dos TextBox para introducir el nombre y la contraseña, así como un botón que manda a comprobar la autenticidad de los datos, responsabilidad de otras clases que se encuentran en otras capas del sistema. Ver **Anexo 19.**

- UI_Principal_Usuario: Es la forma principal de la aplicación que se le muestra al usuario. Contiene un menú a la izquierda con las distintas funcionalidades que puede realizar el usuario dentro del sistema. Mediante los eventos On-Click de cada uno de los componentes del menú se muestran las vistas y formas para capturar o mostrar información. Ver **Anexo 20.**
- UI_Configurar: Es la forma que se invoca desde un evento de UI_Principal_Usuario, tiene como función brindar la posibilidad de realizar la configuración de un proceso. Posee componentes como RadioButtons, TextBox y Botones encargados de capturar los datos y enviarlos hacia otros subsistemas para su procesamiento y posteriormente ser usados para realizar la simulación del proceso. Ver **Anexo 30.**
- UI_Simular: Esta forma se puede invocar desde dos eventos, uno desde UI_Principal_Usuario y otro desde UI_Configurar. Tiene como función mostrar la simulación del proceso de manera gráfica. Posee componentes como TextBox y ListView encargados de mostrar los datos de la simulación que se esta realizando. Ver **Anexo 32.**
- UI_Principal_Admin: Es la forma principal de la aplicación que se le muestra al administrador. Contiene un menú a la izquierda con las distintas funcionalidades que puede realizar el administrador dentro del sistema. Mediante los eventos On-Click de cada uno de los componentes del menú se muestran las vistas y formas para capturar o mostrar información. Ver **Anexo 21.**
- UI_Gestionar_Tipo_Proc: Esta vista se invoca desde UI_Principal_Admin, tiene como función brindar la posibilidad de adicionar un nuevo tipo de proceso, modificar uno existente o eliminarlo. Tiene componentes como TextBox, CommoBox y Botones encargados de capturar los datos y enviarlos a otros subsistemas para su procesamiento. Ver **Anexo 24.**
- UI_Gestionar_Controlador: Esta vista se invoca desde UI_Principal_Admin, tiene como función brindar la posibilidad de gestionar (adicionar o eliminar) un controlador o un autómata. Tiene componentes como TextBox y Botones que se encargan de capturar los datos y enviarlos a otros subsistemas para su procesamiento. Ver **Anexo 31.**
- UI_Establecer_Limites: Esta vista se invoca desde UI_Principal_Admin, tiene como función brindar la posibilidad de establecer los límites al sistema. Tiene componentes como TextBox y Botones que se encargan de capturar los datos y enviarlos a otros subsistemas para su procesamiento. Ver **Anexo 23.**

- UI_Adicionar_Mod_Proc: Esta forma se invoca desde UI_Principal_Admin, tiene como función entrar los datos para adicionar un nuevo modelo de proceso. Tiene componentes como TextBox, CheckBox, RadioButtons y Botones encargados de capturar los datos que serán procesados en otros subsistemas. Ver **Anexo 25.**
- UI_Adicionar_Mod_Grafico: Esta forma se invoca desde UI_Adicionar_Mod, su función es crear el modelo grafico del proceso. Contiene un menú a la izquierda donde se va a encontrar la librería grafica de la cual se van a seleccionar los componentes que van a integrar el modelo. Ver **Anexo 26.**
- UI_Modificar_Mod_Proc: Esta forma se invoca desde UI_Principal_Admin, su función es mostrar el modelo del proceso para que se le puedan hacer modificaciones. Ver **Anexo 27.**
- UI_Modificar_Mod_Grafico: Esta forma se invoca desde UI_Modificar_Mod_Proc, su función es mostrar el modelo grafico del proceso para que se le hagan modificaciones. Ver **Anexo 28.**
- UI_Eliminar_Proc: Esta vista se invoca desde UI_Principal_Admin. Se muestra una lista con los modelos de los procesos que existen en el sistema, de ahí el administrador escogerá el que quiere eliminar. Ver **Anexo 29.**

Subsistema Simulación

Se encuentra en la capa de negocio, su función es realizar la simulación de los procesos industriales. Las principales funciones que realiza son configurar el proceso y simularlo o probarlo en tiempo real. Va a estar conformado por 4 clases, dos clases controladoras y dos clases entidades.

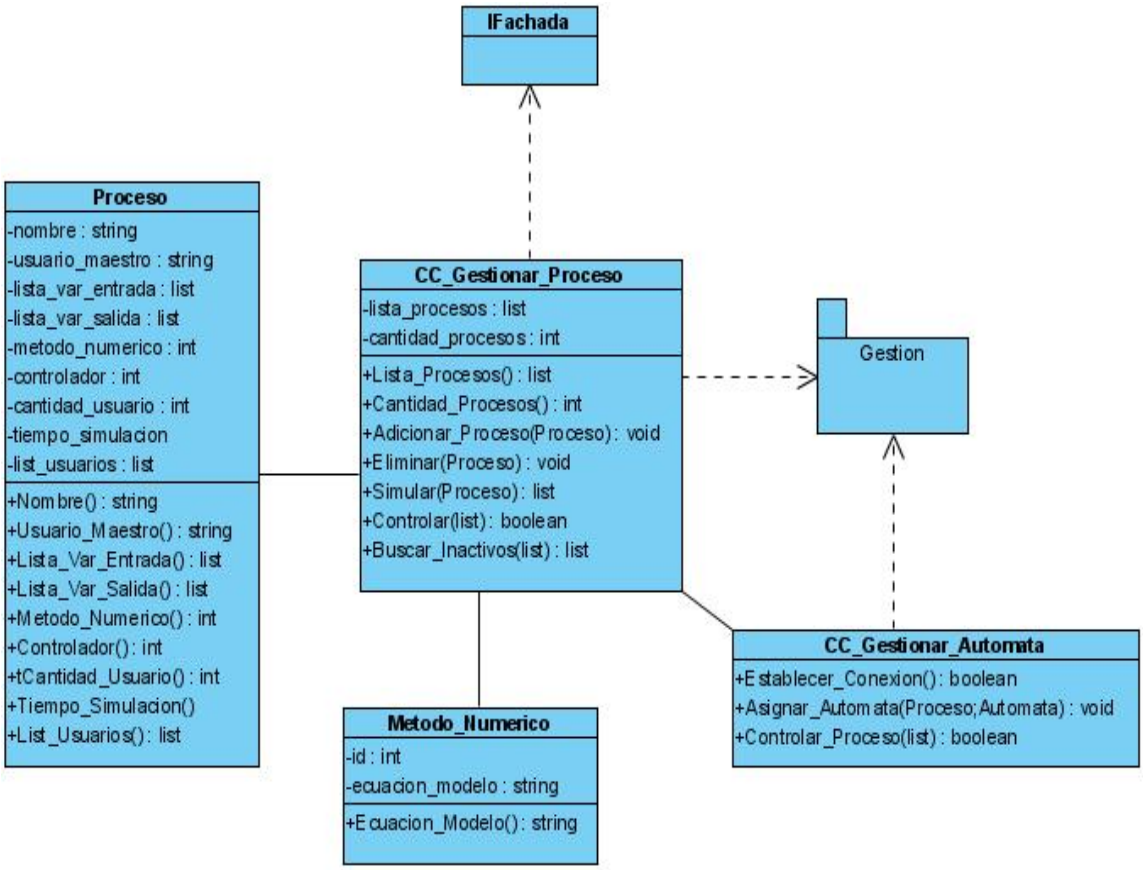


Figura 16 Diagrama de Clases de Diseño. Subsistema Simulación

La relación entre CC_Gestionar_Proceso y CC_Gestionar_Automata con Gestión se debe a clases que existen en Gestión que son necesarias para que CC_Gestionar_ Proceso y CC_Gestionar_Automata realicen sus funcionalidades. En este caso va a realizar una llamada a las funcionalidades de CC_Gestionar_Mod_Proc para poder realizar la configuración del modelo del proceso.

A continuación se explican que funciones cumplen las clases del subsistema.

- CC_Gestionar_Proceso: Es la encargada de gestionar la simulación de un proceso, comenzando por guardar la configuración de este. Luego gestionará todos los datos necesarios para realizar la simulación de un proceso y el control de este, y los enviara a la clase Compilador donde serán interpretados y calculados. Va a tener como atributos un listado de procesos, donde se van a ir adicionando los procesos a medida que se activen en el sistema y una variable con el número de procesos. Implementa el patrón Singleton.
- Proceso: Es la clase que se encarga de guardar los datos de los procesos, una ves realizado el proceso de configuración.

- Método _ numérico: Se encarga de guardar los métodos numéricos que se utilizan en la simulación de los procesos. Va a tener dos atributos, un id para identificar el método numérico y un string para guardar la ecuación del mismo, y un método para devolver la ecuación cuando se requiera de ella.
- CC_Gestionar_Automata: Esta clase se encarga de gestionar la conexión con los autómatas. Entre sus funcionalidades se encuentran el establecer conexión con el autómata, asignar un autómata a un proceso que lo requiera, y llevar el flujo de datos generados por el autómata en el control del proceso.

Subsistema Gestión

Se encuentra en la capa de negocio, su función es gestionar la información importante para el sistema. Las principales funciones que realiza es gestionar la información de los tipos de procesos, gestionar la información de los modelos de proceso, gestionar la información de los controladores y establecer límites necesarios para el buen funcionamiento del sistema. Va a estar integrado por diez clases, cuatro clases controladoras y seis entidades. Su funcionamiento depende de otros dos subsistemas, debido a que la clase CC_Gestionar_Limite necesita datos de la clase Proceso, perteneciente a Simulación y la clase CC_Gestionar_Mod_Proc necesita del subsistema Graficar para adicionar y modificar el modelo gráfico del proceso.

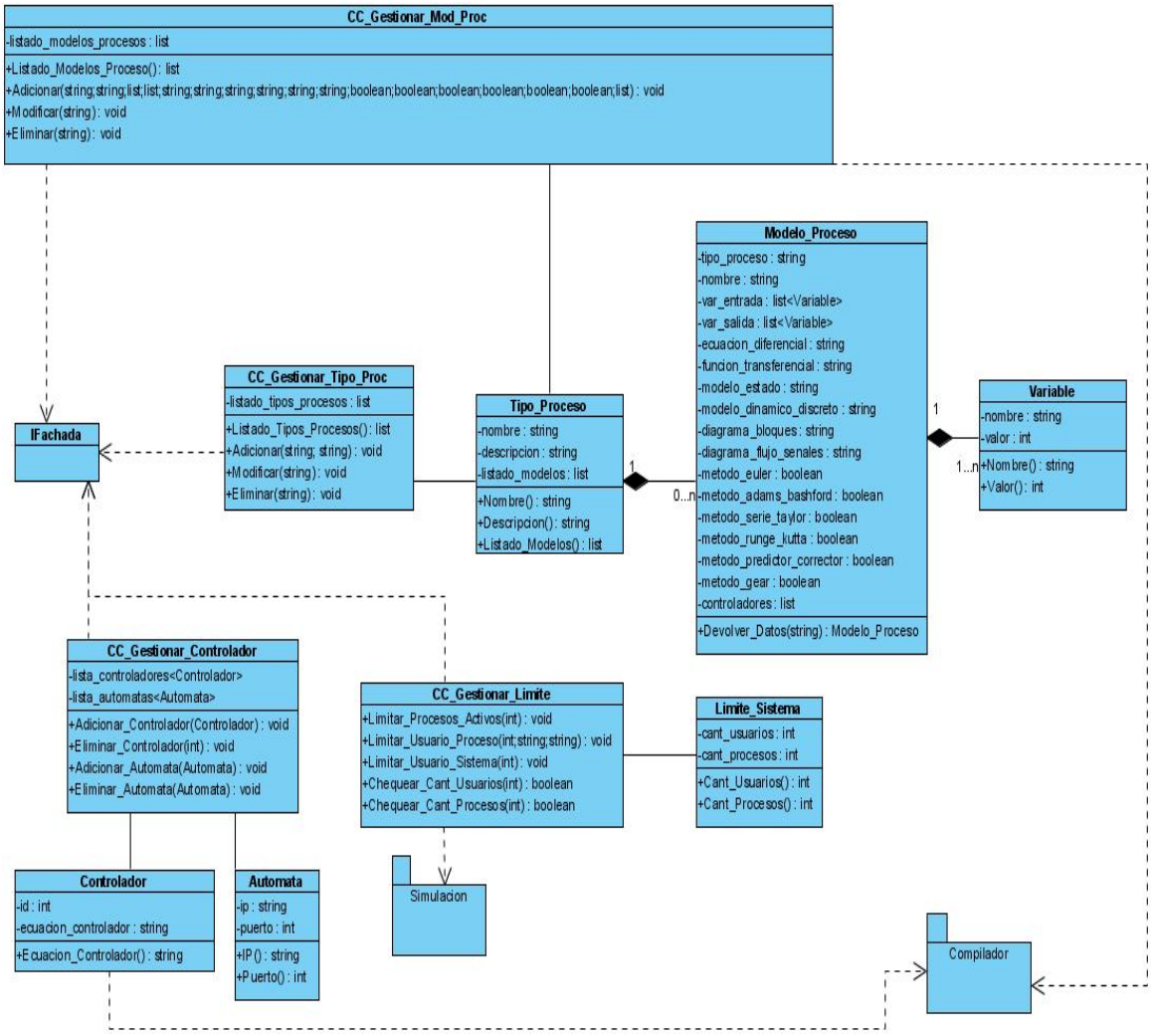


Figura 17 Diagrama de Clases de Diseño. Subsistema Gestión

- CC_Gestionar_Mod_Proc: Es la encargada de gestionar la información de los modelos de los procesos. Va a tener como atributo un listado de los modelos de proceso que existen en el sistema. Entre sus funcionalidades principales se encuentran Adicionar un modelo, modificarlo o eliminarlo.
- Modelo_Proceso: Su función es guardar los datos de los modelos de procesos, los cuales serán utilizados para la configuración de los procesos a simular.
- Variable: Es la encargada de guardar los datos de una variable. Las variables serán utilizadas dentro del modelo del proceso.
- CC_Gestionar_Tipo_Proc: Su función es gestionar la información de los tipos de proceso. Va a tener como atributo un listado de los tipos de proceso que existen en el sistema. Entre sus funcionalidades principales se encuentran Adicionar un tipo de proceso, modificarlo o eliminarlo.
- Tipo_Proceso: Es la encargada de guardar los datos de los tipos de proceso que se encuentran dentro del sistema.

- CC_Gestionar_Limite: Es la encargada de gestionar los limites necesarios para un buen rendimiento del sistema. Va a contar con tres métodos que expresan sus principales funcionalidades, limitar la cantidad de procesos activos en el sistema, limitar la cantidad de usuarios del sistema y limitar la cantidad de usuarios por proceso.
- Limite_Sistema: Es la encargada de guardar los limites del sistema. Para un mejor rendimiento de este se va a establecer límites en cuanto a la cantidad de usuarios conectados simultáneamente y la cantidad de procesos activos.
- CC_Gestionar_Controlador: Es la encargada de gestionar la información de los controladores y los autómatas. Va a tener como atributos un listado de controladores y otro de autómatas. Sus funcionalidades principales son adicionar y eliminar un controlador, y adicionar y eliminar un autómata.
- Autómata: Esta clase se encarga de guardar los datos de un autómata. Tiene como atributos el numero IP donde se va a encontrar el autómata y el puerto por el cual se conectará este.
- Controlador: Se encarga de guardar los controladores que se utilizan, como indica su nombre, para controlar el desarrollo de un proceso. Va a tener dos atributos, un id para identificar el controlador y un string para guardar la ecuación del mismo, y un método, para devolver la ecuación cuando se requiera de ella.

Subsistema Compilador

Dentro del Subsistema Gestión encontraremos el subsistema Compilador, que va a ser el encargado de interpretar las ecuaciones matemáticas con las que se trabaja dentro del proceso de simulación. Este subsistema va a contar con una clase fachada IFachada_Compilador encargada mediante la implementación del patrón fachada que solo se tenga acceso al subsistema mediante ella. Entre las principales clases de encuentran Scanner, Parser, Simbolo, Symtable,Token, Control_Errores y Error, cada una de ellas tiene una responsabilidad en específico para llevar las cadenas string de las ecuaciones matemáticas a lenguajes más sencillos, y verificar si hay errores dentro de ellas. Ver **Anexo 47**.

Subsistema Conexión

Se encuentra en la capa de negocio, su función es gestionar la conexión tanto al sistema como a los procesos. Va a estar integrado por cuatro clases, dos controladoras y dos entidades. Su funcionamiento va a depender de los subsistemas

Simulación y Gestión debido a que la clase CC_Gestionar_Usuario hace uso de clases que se encuentran dentro de ellos.

0.1.n

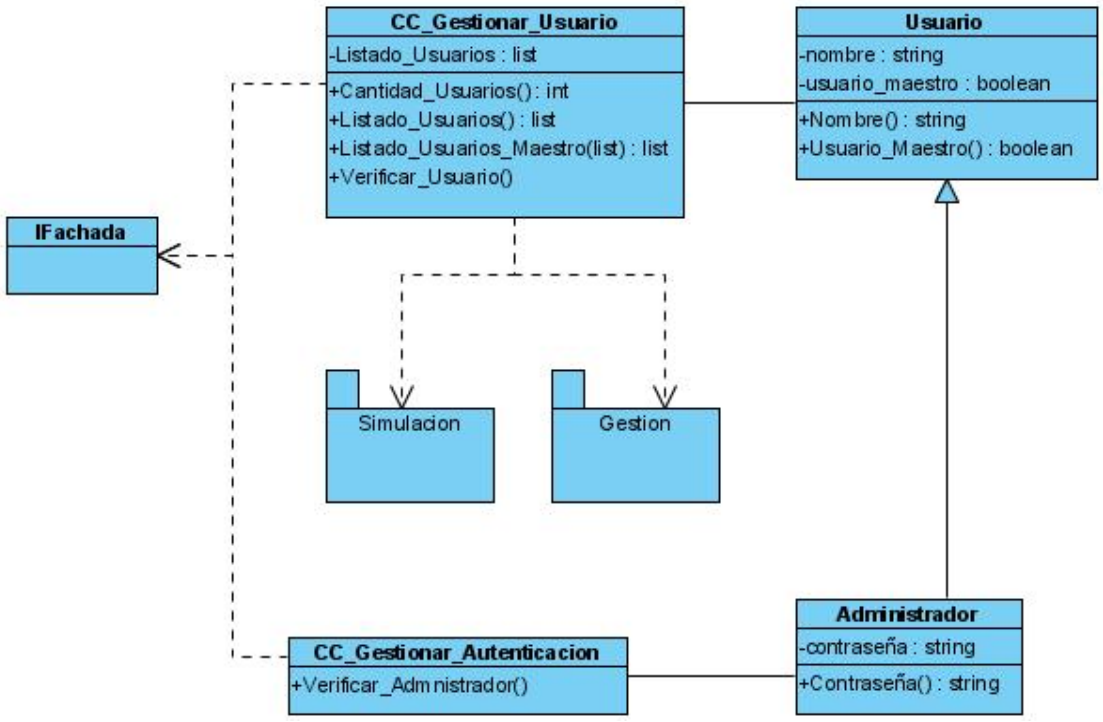


Figura 18 Diagrama de Clases de Diseño. Subsistema Conexión

- CC_Gestionar_Usuario: Es la encargada de gestionar la información de los usuarios que se conectan al sistema. Como atributos tiene un listado de los usuarios. Como funcionalidades principales además de llevar el control sobre los usuarios tiene el listar los usuarios maestros que se encuentran en el sistema.
- Usuario: Su función es guardar los datos de los usuarios que se van a conectar el sistema, de los cuales solo se va a tener el nombre y una variable booleana para saber si ese usuario esta dentro del rol de usuario maestro en un proceso o no.
- CC_Gestionar_Autenticacion: Es la encargada de realizar el proceso de autenticación de un administrador. Su función principal es verificar los datos para dar acceso al modulo de administrador.
- Administrador: Su función principal es guardar los datos de los administradores. Esta clase hereda de usuario, además de tener los atributos de esta clase, tendrá un atributo contraseña necesario para la realización del proceso de autenticación.

Se presentaron y se explicaron cada una de las clases según los subsistemas que pertenecen, pero no se pone de manifiesto las relaciones entre ellas en su totalidad para la realización de las funcionalidades del Nodo Virtual. En el epígrafe que viene a continuación, donde se realizan los casos de uso se pueden observar mejor las relaciones entre las clases interfaces de usuario, controladoras y entidades.

3.2.4.2 Realización de Casos de Uso

Para describir las realizaciones de cada caso de uso se explicara brevemente en que consiste cada uno y se presentará el diagrama de las clases que participan.

Caso de Uso Gestionar_Conexion_Sistema

Este caso de uso es el encargado de controlar el acceso al sistema. Va a tener dos escenarios, conexión al sistema como usuario (mediante el proceso de registrarse) y conexión al sistema como administrador (mediante el proceso de autenticación).

Cualquier usuario puede conectarse al sistema si se registra en él. Para registrarse el usuario solamente deberá entrar su nombre, si el sistema verifica que el usuario no esta conectado, le dará acceso a las funcionalidades propias de los usuarios. Si el usuario no entra los datos para registrase no pasará de la forma principal, siendo imposible que trabaje con el sistema.

El administrador va a tener responsabilidades importantes para la gestión y control del sistema, por eso para su conexión con el mismo se establece el proceso de autenticación. Para autenticarse el usuario entrara los datos correspondientes, si estos son correctos, una vez que el usuario se conecta tiene acceso a las funcionalidades del administrador, pero en caso contrario no va a pasar de la forma de autenticación, y le va a ser imposible trabajar con el sistema.

Este caso de uso va a permitir llevar un control de las operaciones que van a realizar los distintos usuarios. En la figura se muestra el diagrama de clases, como se puede observar va a estar compuesto por tres interfaces de usuario, dos clases controladoras y dos clases entidades. Las interfaces de usuario son formas, de la principal se da acceso a las otras dos. Ver **Anexos 48 y 49**.

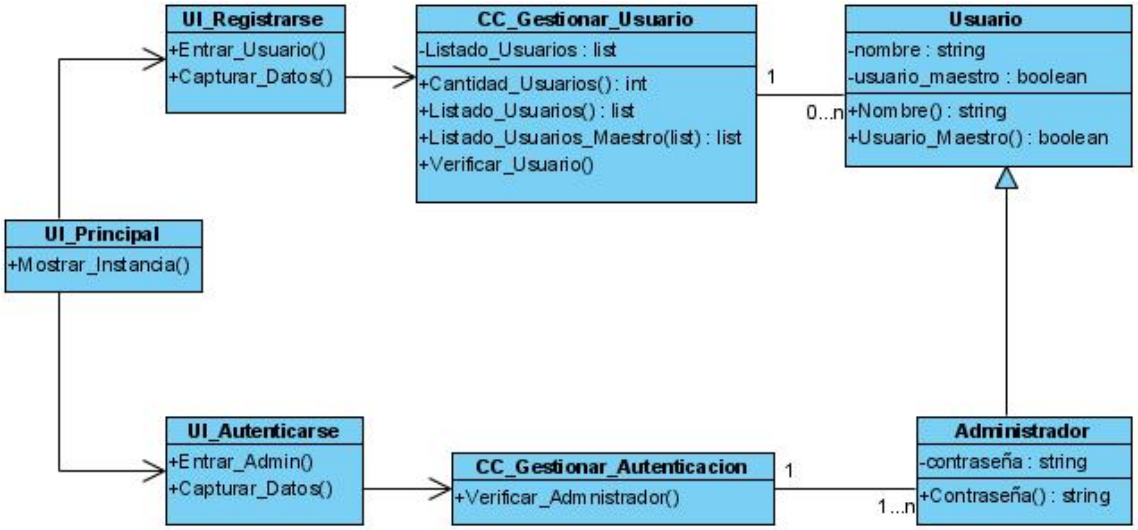


Figura 18 Diagrama de Clases de Diseño. CU Gestionar_Conexion_Sistema

Caso de Uso Gestionar_Conexion_Proceso

Este caso de uso es el encargado de controlar la conexión a los procesos. Al igual que Gestionar_Conexion_Sistema va a tener dos escenarios, cuando el usuario se conecta a un proceso activo, y cuando el usuario se conecta a un proceso inactivo. De acuerdo al tipo de proceso al que se conecte el usuario, el sistema lo enviara a dos interfaces distintas. Ver **Anexos 50 y 51.**

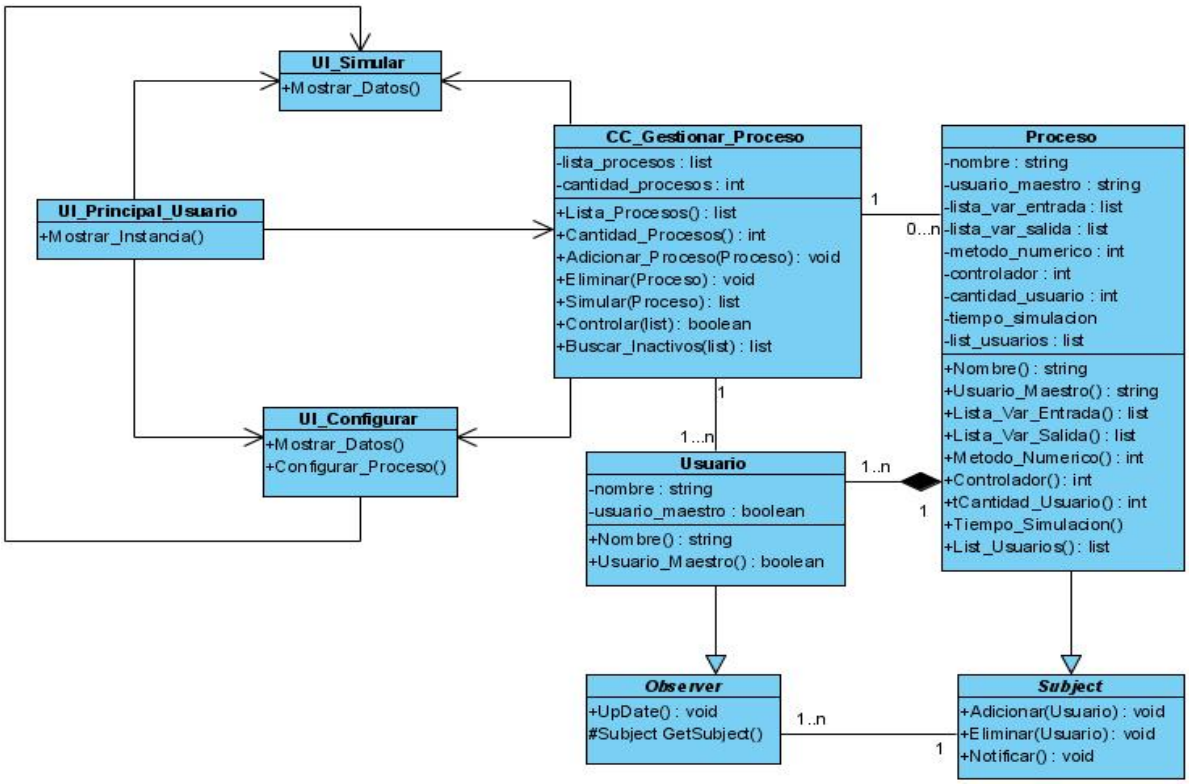


Figura 19 Diagrama de Clases de Diseño. CU Gestionar_Conexion_Sistema

Caso de Uso Establecer_Limites

Este caso de uso es el encargado de controlar los limites del sistema, la cantidad de usuarios que van a conectarse, la cantidad de procesos activos. Una vez establecidos estos límites se guardaran dentro de una clase entidad para ser utilizados posteriormente. También se establecerán los límites para la cantidad de usuarios conectados por procesos. Este caso de uso ayuda a gestionar el buen funcionamiento del sistema, mediante él se evita la sobrecarga en el mismo. Ver **Anexos 52, 53 y 54.**

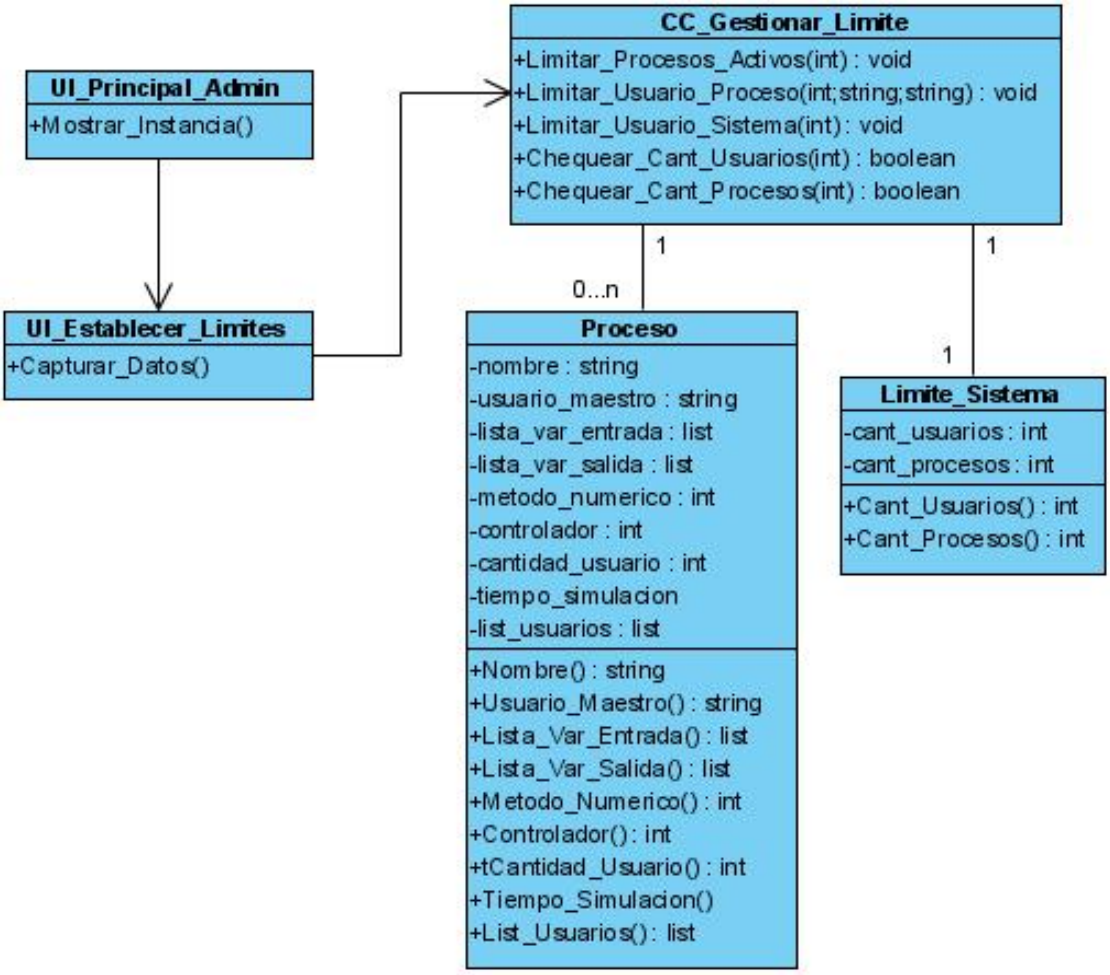


Figura 20 Diagrama de Clases de Diseño. CU Establecer_Limites

Caso de Uso Gestionar_Tipo_Proceso

Este caso de uso es el encargado de gestionar (adicionar, modificar. eliminar) toda la información de los tipos de procesos que podemos encontrar en el sistema, importante para la estandarización de los procesos. Ver Anexos **55, 56 y 57**.

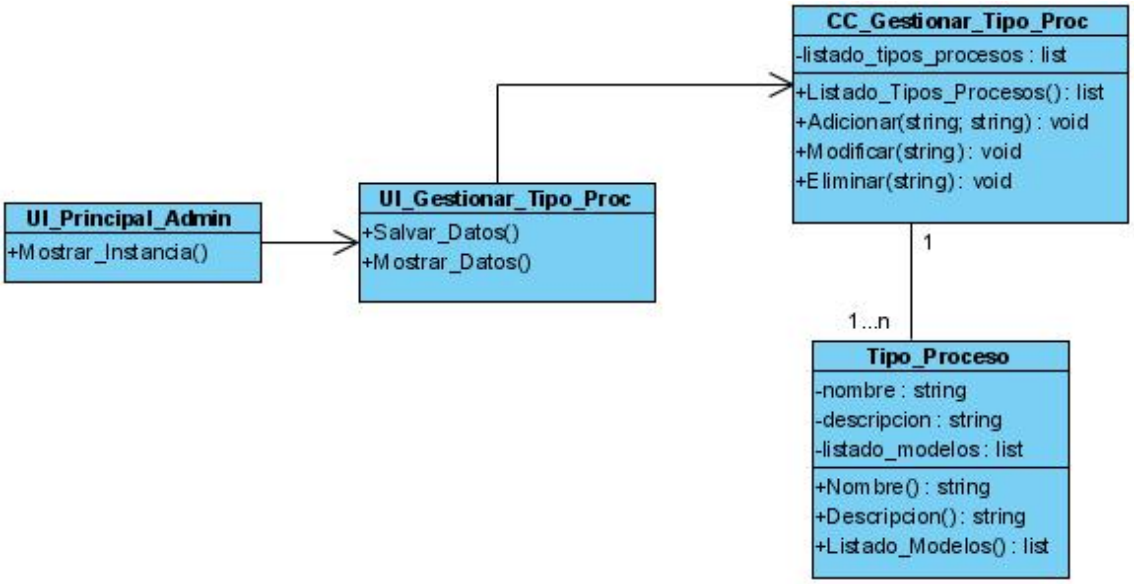


Figura 21 Diagrama de Clases de Diseño. CU Gestionar_Tipo_Proceso

Caso de Uso Gestionar_Modelo_Proceso

Este caso de uso es sumamente importante, ya que en él es donde se gestiona (adiciona, modifica, elimina) los modelos de los procesos, los cuales van a contener toda la información relacionada con el proceso, modelo matemático, variables de entrada y de salida y método numéricos que se pueden utilizar. Esta información es vital para realizar la simulación de un proceso. Ver **Anexos 58, 59 y 60.**

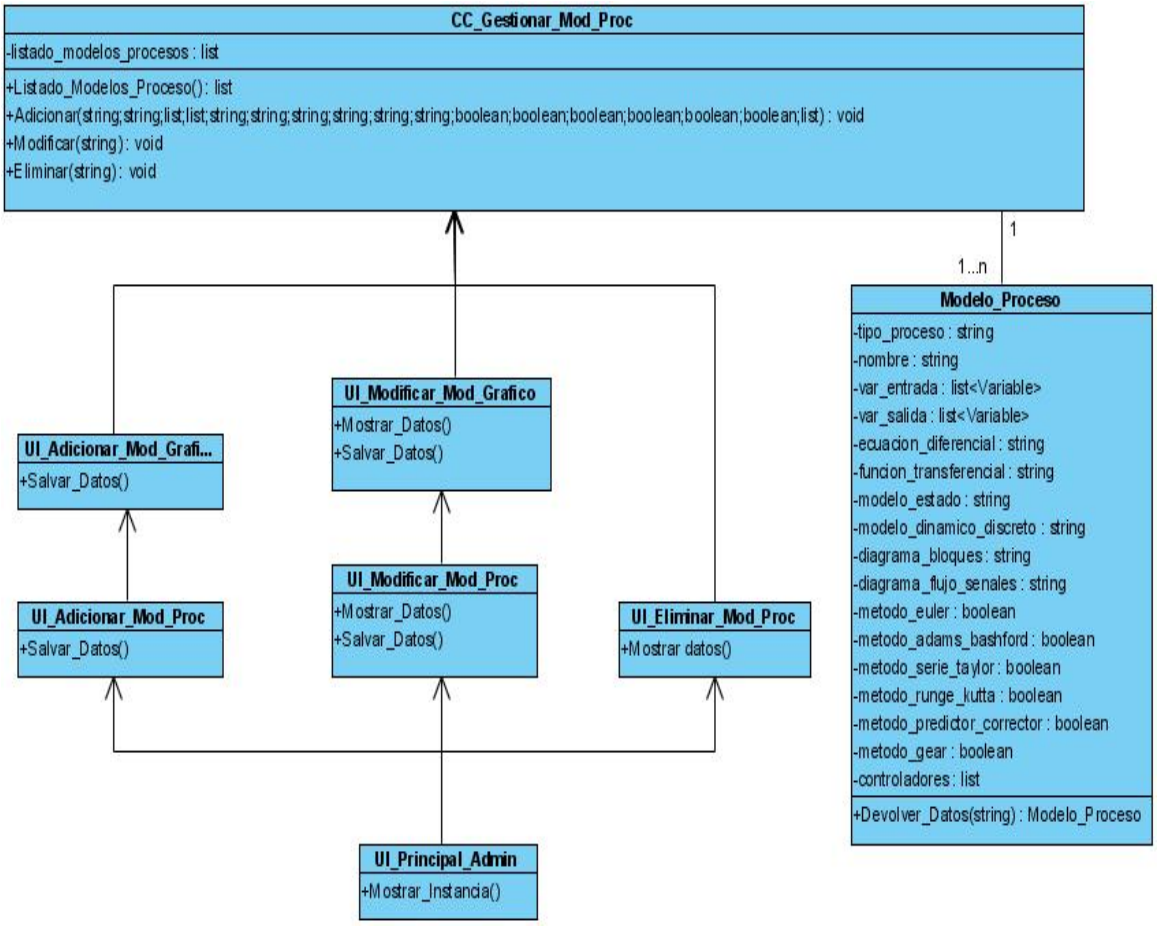


Figura 22 Diagrama de Clases de Diseño. CU Gestionar_Modelo_Proceso

Caso de Uso Configurar_Proceso

Este caso de uso es de vital importancia, en él se realizará el proceso de configuración, proceso indispensable para que se pueda hacer la simulación o probar aplicaciones. Se mostrará el modelo del proceso y el usuario maestro entrara los valores de las variables de entrada, escogerá la forma en que estará expresada la ecuación matemática del proceso, así como el método numérico y el controlador que se va a utilizar, todos estos datos me permitirán crear un proceso listo para simular o probarlo. Ver **Anexo 61.**

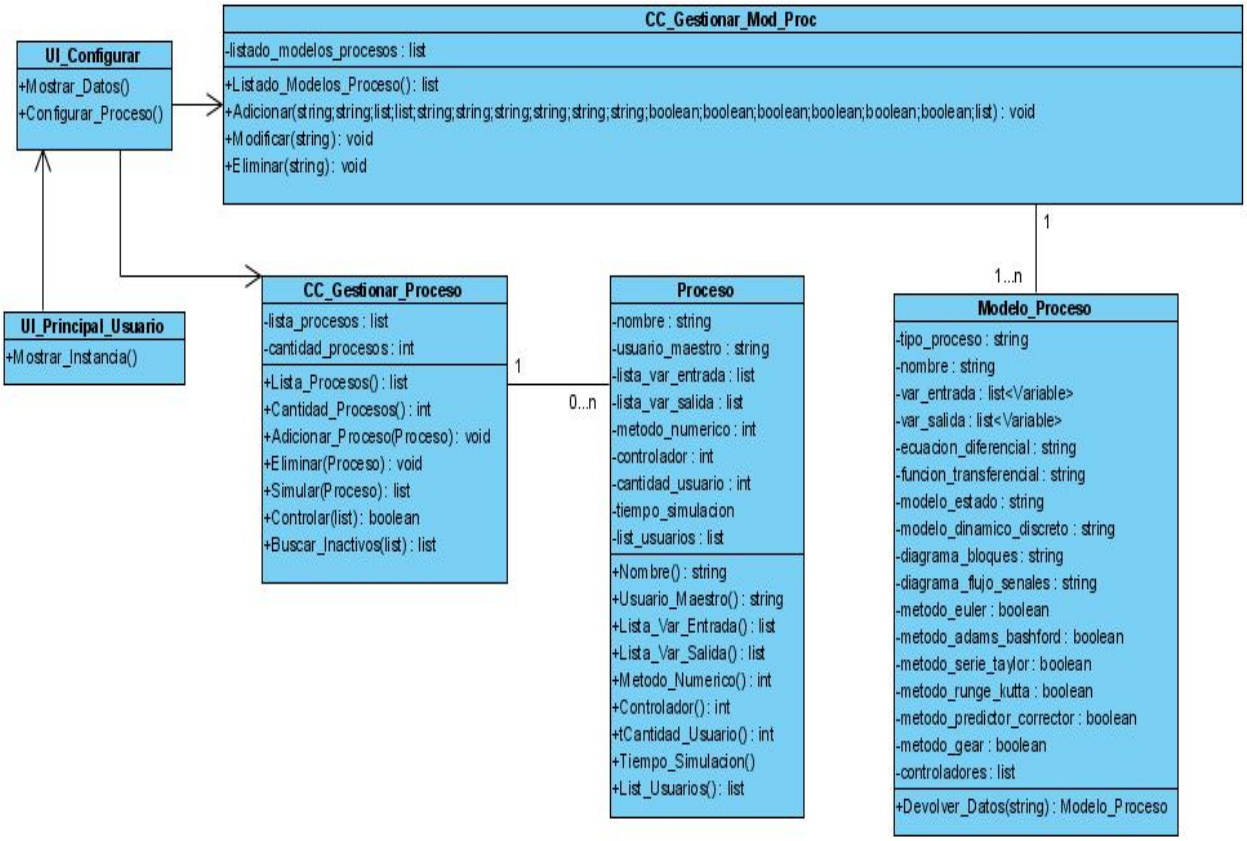


Figura 23 Diagrama de Clases de Diseño. CU Configurar_Proceso

Caso de Uso Simular_Proceso

Este caso de uso es el encargado de realizar la simulación de un proceso. Utilizando los procesos que se crean en la configuración de los modelos, realiza todo el proceso de simulación de los mismos. Ver **Anexo 62.**

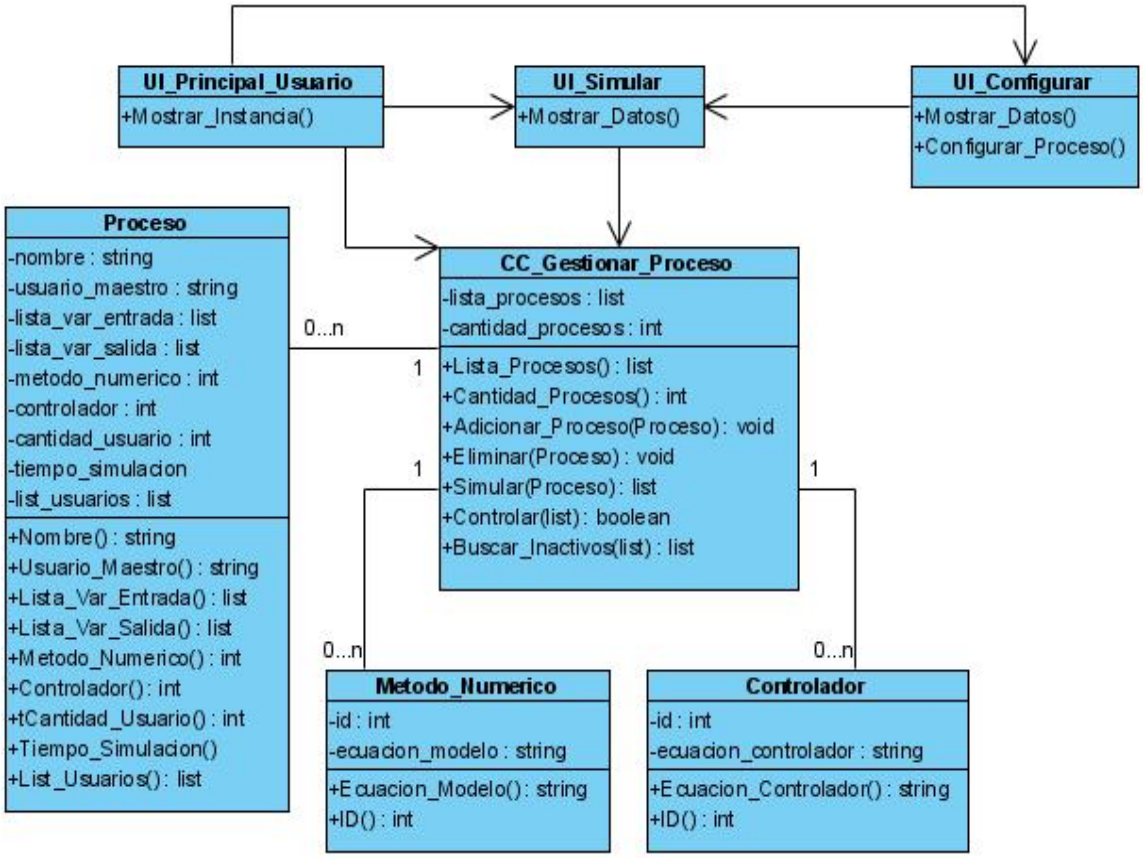


Figura 24 Diagrama de Clases de Diseño. CU Simular_Proceso

Caso de Uso Probar_Aplicacion

Este caso de uso se encarga de establecer la conexión con los autómatas para probar las aplicaciones en tiempo real. Utilizando los procesos que se crean en la configuración de los modelos, comienza el proceso de simulación, lo que en este caso de uso el control de ese proceso va a ser realizado por un autómata. Ver **Anexo 63.**

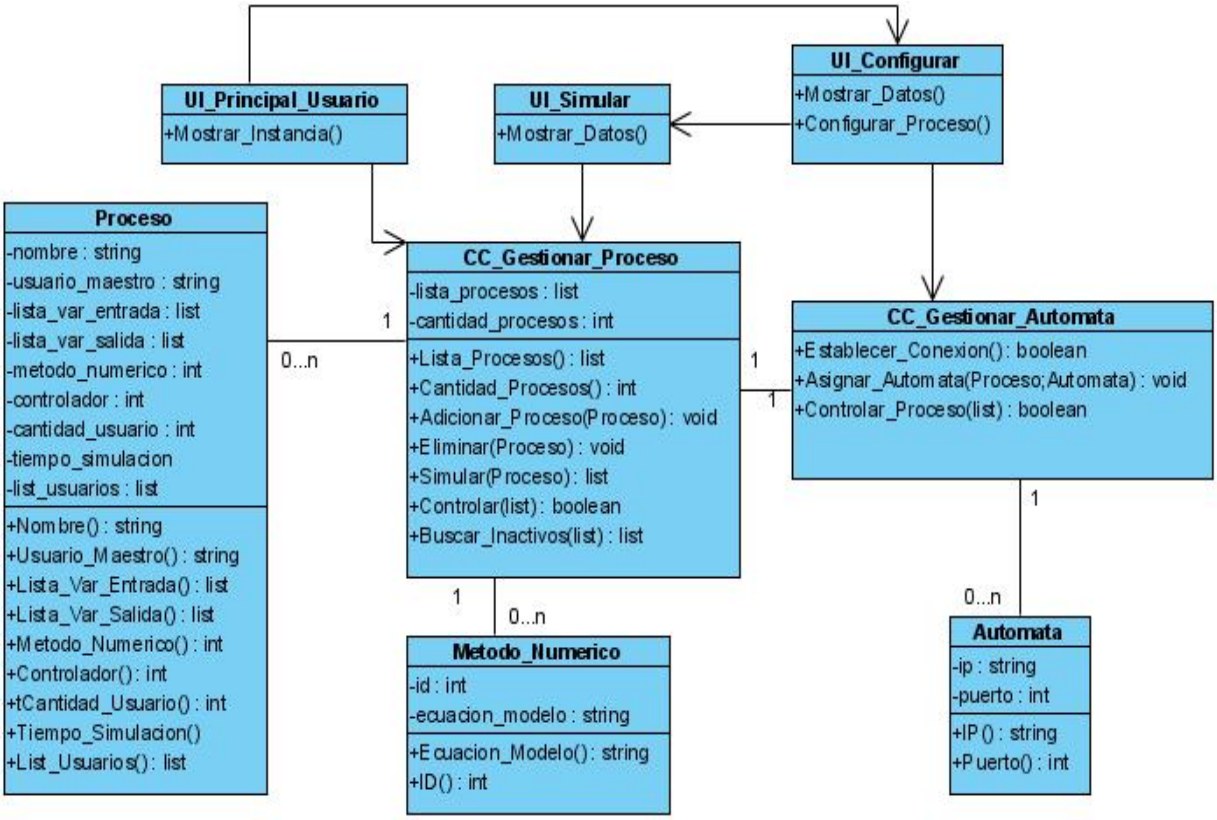


Figura 25 Diagrama de Clases de Diseño. CU Probar_Aplicacion

Caso de Uso Gestionar_Controlador

Este caso de uso es el encargado de gestionar toda la información relacionada con los controladores, tanto los controladores del sistema, como los externos que en este caso son los autómatas. Ver **Anexos 64, 65, 66 y 67.**

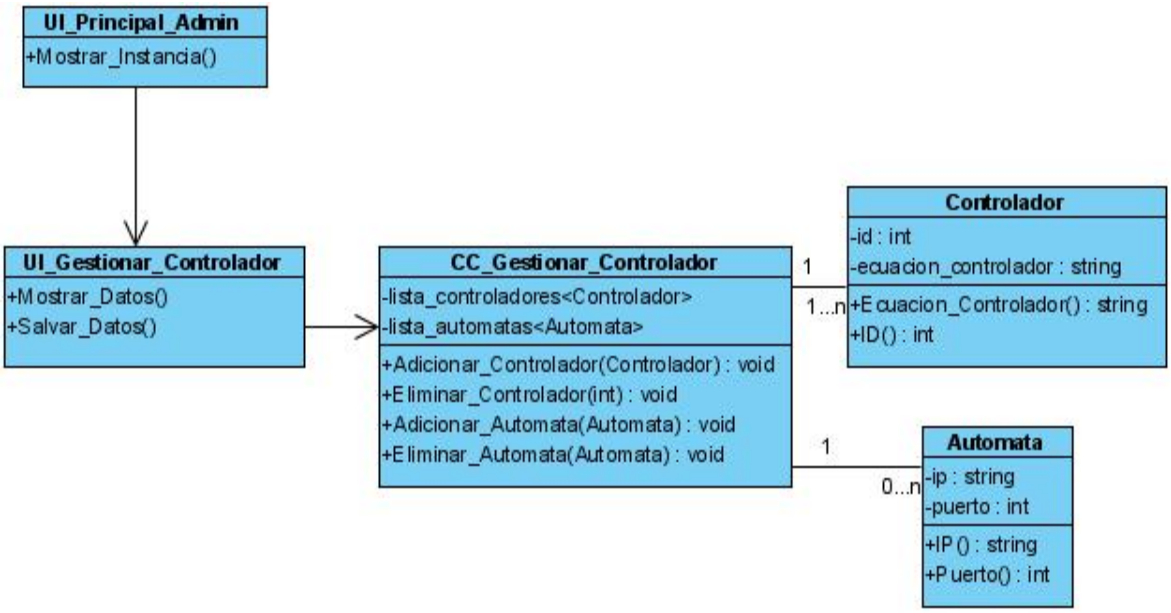


Figura 26 Diagrama de Clases de Diseño. CU Gestionar_Controlador

Conclusiones

En este capitulo se trato todo lo referente al análisis y diseño del sistema. Se realizó la representación de los casos de uso en clases del análisis. Se hizo referencia al tipo de arquitectura a utilizar, se justifican los patrones de diseño que se usaron, se explicaron los módulos que componen el Nodo Virtual y los diferentes subsistemas que forman el Modelo de Diseño. Todo esto ayudo a llegar a las siguientes conclusiones:

- La realización del modelado de análisis ayuda a lograr un diseño con mayor calidad.
- Los patrones de diseño aplicados hacen más robusto, fuerte y flexible el sistema.
- Los diferentes subsistemas diseñados permiten trabajar de manera independiente y desarrollar al mismo tiempo varios de ellos.
- Las realizaciones de los casos de uso brindan a los programadores información que les permite programar distintas clases para que el sistema tenga las funcionalidades necesarias.
- El analista y diseñador del sistema desarrollaron correctamente los artefactos generados en este flujo.

Capítulo 4: Validación de los resultados.

Introducción

Un factor fundamental para efectuar la evaluación de los productos y procesos de software en los diferentes dominios de aplicación lo constituyen las métricas, las cuales abarcan un gran escenario en cuanto a medidas de software computacional se refieren. Las métricas suelen ser aplicadas a muchas organizaciones, procesos y productos que estén relacionados y afecten a la estimación de costo.

Como se conoce la medición es muy común en el mundo de software y las mismas pueden englobarse en dos categorías principales: medidas directas y medidas indirectas. En las primeras está reflejado el costo y le esfuerzo aplicado, las líneas de código producidas, velocidad de ejecución, el tamaño de memoria, así como los defectos observados en un período de tiempo determinado. Las segundas abarcan la funcionalidad, calidad, complejidad, eficiencia, fiabilidad, facilidad de mantenimiento de un software, etc.

De manera general las métricas del software son las que están relacionadas con el desarrollo del software como tal como son: complejidad, funcionalidad y eficiencia. Dentro de ellas se pueden encontrar las métricas técnicas, de calidad, de productividad, orientadas a la persona, orientadas al tamaño y a la función, así como el establecimiento de la estimación como actividad crucial del proceso de gestión.

A raíz del trabajo realizado se obtuvieron numerosos artefactos que van desde la fase de inicio, pasando por las etapas de análisis y el diseño de software, llevados a cabo por el equipo de desarrolladores. Para dar soporte a las actividades realizadas y validar los artefactos se emplearon las métricas de la calidad de especificación de los requisitos que propone Presuman, las Métricas de diseño arquitectónico propuestas por Card y Glass para calcular la complejidad de los módulos y las métricas Orientadas a Clases.

4.1 Proceso o fase inicial. Modelo de métricas.

Con la realización de los procesos del negocio y con ello el modelo del dominio propuesto se pudo comprender la estructura y dinámica del entorno organizacional. A partir del artefacto de modelo de dominio se logró identificar el glosario de términos proporcionando un mejor entendimiento así como una buena comprensión entre las entidades que interactúan. Se pudieron identificar y proponer mejoras en los requerimientos del sistema que posibilitaron minimizar el esfuerzo de los especialistas.

De manera general se obtuvieron artefactos como el Modelo de Dominio, Glosario de Términos, proporcionado una trazabilidad directa hacia el flujo de trabajo de Requerimientos y posterior a ello, Modelo de Casos de Uso.

4.1.1 Métrica de la calidad de especificación de los requisitos

Para realizar la validación de los requisitos existe toda una lista de características que sugieren el uso de una o mas métricas como son: especificidad (ausencia de ambigüedad), corrección, compleción, comprensión, capacidad de verificación, consistencia externa e interna, capacidad de logro, concisión, trazabilidad, capacidad de modificación, exactitud y capacidad de reutilización. [28]. Debido a las particularidades de los requisitos capturados, se va a aplicar la métrica que mide la especificidad en los mismos, haciendo que los clientes puedan entender los requisitos de una manera fácil y se puedan probar.

Para llevar a cabo este proceso se tiene que: n_r representa el número de requisitos del sistema:

$$n_r = n_f + n_{nf}$$

Donde n_f es el número de requisitos funcionales y n_{nf} es el número de requisitos no funcionales.

Luego se procede a medir la especificidad de los requisitos con la siguiente fórmula:

$$Q = n_{ui} / n_r$$

Donde n_{ui} es el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas. El valor de Q a medida que se acerca a 1, se va disminuyendo la ambigüedad de la especificación.

Para darle soporte a lo planteado anteriormente y así proceder a la evaluación de la métrica propuesta se hicieron 2 revisiones, con 3 revisores consultados, todo ello con la primicia de obtener la menor ambigüedad posible y mayor claridad en los requisitos de forma tal que se cubrieran todas las necesidades de los clientes y este a su vez quedara lo mas satisfecho posible.

En la primera revisión efectuada se pudo constatar que algunos de los requisitos funcionales y no funcionales no cumplían con la redacción correspondiente y presentaban problemas de ambigüedad. Ellos son: de los requisitos funcionales, representaban el 0.16% con ambigüedad, no siendo así con los requisitos no

funcionales. Por lo tanto para un total de 67 requerimientos, los revisores técnicos tuvieron la misma interpretación para 50 de ellos.

Q = 50 / 67
Q = 0.75

En la segunda y última revisión se detectaron menos problemas que en la anterior, sin embargo se realizó un estudio de los requisitos mucho más profundo y solo se encontró error en el 0.12% de los requisitos funcionales. Por lo que para una totalidad de 73 requerimientos, los revisores tuvieron la misma interpretación para 67 de ellos.

Q = 67 / 73
Q = 0.92

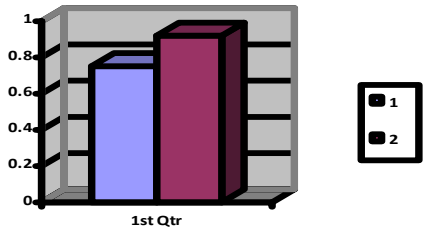


Figura: Gráfico de control de la calidad de la especificación de los requisitos

Luego de realizar un estudio cuantitativo de los resultados arrojados posterior a las revisiones, se puede llegar a la conclusión que los requisitos que se obtienen tienen en su mayoría un grado de ambigüedad bastante bajo, por lo que se puede ver que el valor de Q es bastante cercano a 1. Así se pueden presentar el nivel de calidad que deben tener los requerimientos, artefacto más que fundamental para desarrollar con éxito un software que sea capaz de cumplir con las expectativas y necesidades de los clientes.

4.2 Técnica de prototipado

Se conoce como prototipo a la implementación concreta de un sistema que se crea para explorar cuestiones sobre aspectos muy diversos durante el desarrollo de un sistema. [28] Ellos permiten la comunicación y participación que puede haber entre el

equipo y los clientes, haciendo de la calidad y las especificaciones una herramienta primordial para comprobar los requisitos del software.

Un prototipo no funcional es el diseño de la posible interfaz del software a construir, esta se utiliza para tener un mejor entendimiento del problema y validar los requisitos que el usuario solicitó. [29] Para la elaboración y construcción de la interfaz gráfica se hizo uso de la herramienta Microsoft Visio 2007, modelo que satisfizo la mayoría de las expectativas y necesidades de los clientes.

La generación de los prototipos es una forma de validar los requerimientos funcionales que fueron capturados durante la etapa de Requerimientos. La principal ventaja que presentan las interfaces de usuarios es la de reflejar la presentación e interacción de las necesidades del usuario final en un entorno amigable y fácil de entender.

Luego de la construcción de la interfaz principal, se hace necesaria la fabricación de otras con pequeñas modificaciones que complementen el producto del software final.

4.3 Complejidad de los módulos

La complejidad de los módulos puede calcularse utilizando las Métricas de Diseño Arquitectónico. Estas son métricas de diseño de alto nivel que se concentran en las características de la arquitectura, haciendo especial énfasis en la estructura arquitectónica y en al eficiencia de los módulos. [28] Para aplicarlas no se requiere ningún conocimiento del trabajo interno de un modulo en particular del sistema, en este sentido son consideradas métricas de caja negra.

Card y Glass definen tres medidas de la complejidad del diseño del software: complejidad estructural, complejidad de datos y complejidad del sistema. [28]

La *complejidad estructural*, *S(i)*, de un módulo *i* se define de la siguiente manera:

S(i) = f^2 out(i)

donde *f out(i)* es la expansión del modulo *i*.

La *complejidad de datos*, *D(i)*, proporciona una indicación de la complejidad en la interfaz interna de un módulo *i* y se define como:

D(i)=v(i)/ [f out(i)+1]

donde *v(i)* es el número de variables de entrada y salida que entran y salen del módulo *i*.

La *complejidad del sistema*, *C(i)*, se define como la suma de las complejidades estructural y de datos, y se define como:

C(i)=S(i)+D(i)

A medida que crecen los valores de complejidad, la complejidad arquitectónica o global del sistema también aumenta. A continuación, según las formulas planteadas, se calcula la complejidad de los módulos desarrollados en el presente trabajo

Módulo Conexión.

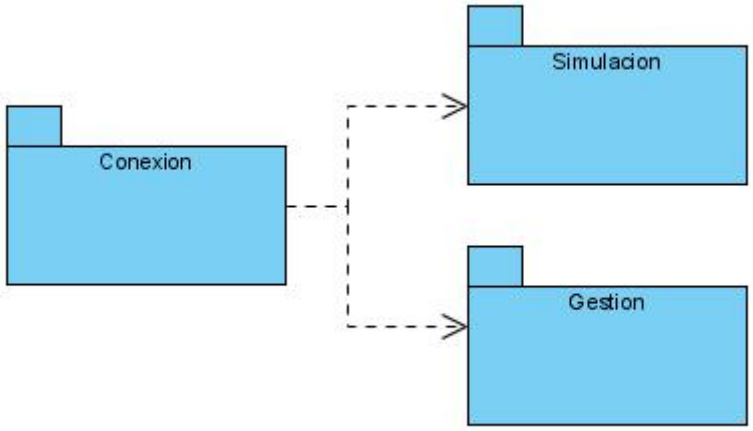


Figura 27 Expansión del Módulo Conexión

Tabla 5 Complejidad Módulo Conexión

Complejidad estructural *S(i)*	**Complejidad de datos** *D(i)*	**Complejidad del sistema** *C(i)*
S(i)=2 ^2=4	*D(i)*=2/ [4+1]= 2/5=0.4	*C(i)*= 4+0.4=4.4

Módulo Simulación

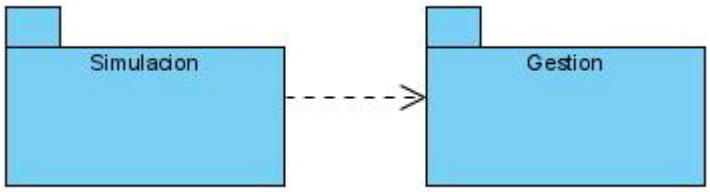


Figura 28 Expansión del Módulo Simulación

Tabla 6 Complejidad Módulo Simulación

Complejidad estructural *S(i)*	**Complejidad de datos** *D(i)*	**Complejidad del sistema** *C(i)*
S(i)=1 ^2=1	*D(i)*=0/ [1+1]= 0/2=0	*C(i)*= 1+0=1

Módulo Gestión

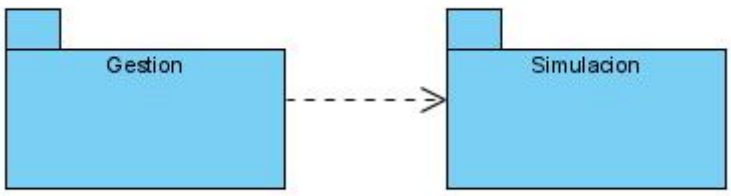


Figura 29 Expansión del Módulo Gestión

Tabla 7 Complejidad del Módulo Gestión

Complejidad estructural *S(i)*	**Complejidad de datos** *D(i)*	**Complejidad del sistema** *C(i)*
S(i)=1 ^2=1	*D(i)*=25/ [1+1]= 25/2=14.5	*C(i)*= 1+14.5=15.5

Resultados

Tabla 8 Complejidad de los módulos

Módulo Conexión	**Módulo Simulación**	**Módulo Gestión**
4.4	1	15.5

Como se puede observar, los módulos Conexión y Simulación no son muy complejos. Los valores de la complejidad en ellos son muy bajo, sin embargo no sucede lo mismo con el Módulo Gestión. La complejidad el mismo viene dada a que requiere muchos parámetros de entrada.

Es necesario mencionar que la mayoría de los autores que tratan el tema de las métricas de diseño arquitectónico proponen tres tipos de sistema según su complejidad:

- No muy complejo.
- Complejo.
- Muy complejo.

Para saber cuando un sistema esta dentro de alguno de los tres grupos se proponen umbrales. Para los no muy complejos tiene que cumplirse que *D(i)* <= 7 y *S(i)* <= 32. A pesar que el *S(i)* del sistema es menor que 32, el *D(i)* sobrepasa del umbral, por lo que se puede llegar a la conclusión de que el sistema es complejo.

4.3 Métricas orientadas a clases.

4.3.1 Tamaño de Clase (TC).

Para medir el tamaño de clase se tienen en cuenta los siguientes aspectos:

- Total de operaciones, ya sean las propias o las heredadas de las clases padres e interfaces que implementen.
- Cantidad de atributos, tanto los de ella, como lo de los padres.

- Promedio general de los dos anteriores para el sistema completo.

Para evaluar las métricas son necesarios los umbrales. En este caso las clases se clasifican en tres grupos según su tamaño, los que se representan en la siguiente tabla junto con los umbrales seleccionados para su clasificación.

Tabla 9 Valores de los umbrales para TC

Clasificación	**Valores de los umbrales**
Pequeño	<=20
Medio	>20 y <=30
Grande	>30

La tabla 10 ilustra las clases del sistema aplicándole la métrica seleccionada.

Tabla 10 Tamaño de las clases

No	**Nombre**	**Cantidad Atributos**	**Cantidad Operaciones**	**Tamaño**
1	CC_Gestionar_Usuario	1	4	Pequeño
2	Usuario	2	4	Pequeño
3	CC_Gestionar_Autenticacion	0	1	Pequeño
4	Administrador	3	3	Pequeño
5	CC_Gestionar_Proceso	2	7	Pequeño
6	Proceso	9	12	Medio
7	Observer	0	2	Pequeño
8	Subject	0	3	Pequeño
9	CC_Gestionar_Mod_Proc	1	4	Pequeño
10	Modelo_Proceso	17	1	Pequeño
11	CC_Gestionar_Limite	0	5	Pequeño
12	Limite	2	2	Pequeño
13	CC_Gestionar_Controlador	2	4	Pequeño
14	Controlador	2	2	Pequeño
15	Automata	2	2	Pequeño
16	CC_Gestionar_Tipo_Proc	1	4	Pequeño
17	Tipo_Proceso	3	3	Pequeño
18	Metodo_Numerico	2	2	Pequeño
19	CC_Gestionar_Automata	0	3	Pequeño

20	IFachada_Compilador	0	2	Pequeño
21	CC_Scanner	0	1	Pequeño
22	CC_Symtable	1	4	Pequeño
23	Simbolo	2	2	Pequeño
24	Token	4	5	Pequeño
25	Token_Type	8	0	Pequeño
26	CC_Parser	0	1	Pequeño
27	CC_Control_Errores	0	3	Pequeño
28	Error	3	3	Pequeño
29	Valuable	0	1	Pequeño
30	Valor_Negativo	1	2	Pequeño
31	Constante_Numerica	1	2	Pequeño
32	Trigonometria	2	3	Pequeño
33	Algebra	2	3	Pequeño

Cuando existe un TC grande se afectan los parámetros de calidad definidos por esta métrica. Se reduce la reutilización de las clases, la implementación se hace más compleja, las pruebas son difíciles de realizar y aumenta la responsabilidad de las clases.

La mayoría de las clases que conforman el sistema están dentro de la categoría de pequeñas, lo que demuestra que el sistema no es complejo. Los resultados obtenidos son positivos según esta métrica, como se puede ver en las siguientes tablas.

Tabla 11 Cantidad de clases por clasificación

Clasificación	**Cantidad Clases**
Pequeño	32
Medio	1
Grande	0

Tabla 12 Resultados de la Métrica TC

Cantidad Clases	**Promedio Atributos**	**Promedio Operaciones**
33	2.24	3.03

4.3.2 Árbol de profundidad de herencia (APH)

Esta métrica está definida por la máxima longitud que exista entre el nodo y la raíz del árbol. Donde el nodo es una clase hija que hereda de una clase, y así respectivamente hasta llegar a la raíz. A medida que esa longitud va creciendo, entonces se van heredando más operaciones y atributos por las clases hijas. Se hace difícil predecir el comportamiento de las clases que se encuentran en los niveles más bajos del árbol. Esta tiene sus ventajas y desventajas. Si los valores de APH son grandes, entonces se garantiza que se reutilice gran cantidad de código; pero al mismo tiempo hace que el diseño sea más complejo. Esto provoca un mayor acoplamiento entre las clases.
En el Nodo Virtual no se hizo necesario hacer demasiado uso de la herencia. Aplicando esta métrica al diseño propuesto se obtienen resultados que demuestran su poca complejidad, el árbol de profundidad de herencia toma valor 2, por lo que existe bajo acoplamiento y es de fácil reparación.

4.3.3 Relaciones entre Clases (RC)

Esta métrica está dada por la cantidad de relaciones de uso que existe entre las distintas clases que forman el diseño propuesto. Se le aplica a las mismas clases que le fue aplicada la métrica TC. Los aspectos de calidad que se miden son: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas.

Tabla 13 Cantidad de relaciones de uso entre las clases

No	Nombre	Relaciones de uso
1	CC_Gestionar_Usuario	1
2	Usuario	0
3	CC_Gestionar_Autenticacion	1
4	Administrador	0
5	CC_Gestionar_Proceso	4
6	Proceso	0
7	Observer	0
8	Subject	1
9	CC_Gestionar_Mod_Proc	1
10	Modelo_Proceso	0
11	CC_Gestionar_Limite	2
12	Limite	0
13	CC_Gestionar_Controlador	2

14	Controlador	0
15	Automata	0
16	CC_Gestionar_Tipo_Proc	1
17	Tipo_Proceso	0
18	Metodo_Numerico	0
19	CC_Gestionar_Automata	1
20	IFachada_Compilador	0
21	CC_Scanner	1
22	CC_Symtable	1
23	Simbolo	0
24	Token	1
25	Token_Type	0
26	CC_Parser	3
27	CC_Control_Errores	1
28	Error	0
29	Valuable	0
30	Valor_Negativo	0
31	Constante_Numerica	0
32	Trigonometria	0
33	Algebra	0

Para medir el acoplamiento según los resultados de esta métrica, algunos especialistas plantean los siguientes valores.

Tabla 14 Acoplamiento

Categoría	**Relaciones de uso**	**Cantidad de Clases**
Ninguno	0	19
Bajo	1	10
Medio	2	2
Alto	>2	2

Los demás parámetros de calidad que mide esta métrica dependen del valor promedio de las dependencias de uso de todas las clases, en este caso ese promedio es de 0.63.

Tabla 15 Cantidad de Pruebas y Complejidad de Mantenimiento

Categoría	**Criterio**	**Cantidad de Clases**
Baja	<=Prom.	19
Media	>Prom. Y <=2*Prom.	10
Alta	>2*Prom.	4

Tabla 16 Reutilización

Categoría	**Criterio**	**Cantidad de Clases**
Baja	>2*Prom.	4
Media	>Prom. Y <=2*Prom.	10
Alta	<=Prom.	19

De manera general los resultados de esta métrica son positivos. El acoplamiento existente entre las clases es bajo, el 58% no tiene ningún acoplamiento, el 30% es bajo, el 6% es medio, y solo el 6% tiene alto acoplamiento. El nivel de reutilización de las clases es bastante bueno, el 58% de las clases pueden ser reutilizadas. Pasa lo mismo con la cantidad de pruebas y la complejidad de mantenimiento, el 58% de las clases son fáciles de reparar y la cantidad de pruebas a realizar es relativamente corta.

Conclusiones

Al aplicar estas métricas al análisis y diseño propuesto se arribaron a las siguientes conclusiones:

- Después de las revisiones realizadas a los requisitos, el nivel de ambigüedad en ellos decreció, aumentando el nivel de calidad de los mismos.
- El módulo mas complejo es el de Gestión, lo que demuestra que se va a necesitar más esfuerzo para su construcción.
- El tamaño de la mayoría de las clases del diseño es pequeño. El 96.6% de las clases del sistema están consideradas pequeñas, son fáciles de probar, implementar y son reutilizables, lo que demuestra que el diseño propuesto tiene calidad.
- La relación de herencia que existe entre las clases demuestra que el diseño no es complejo y cumple con el patrón de Bajo Acoplamiento.

- De forma general las métricas aplicadas demuestran que la solución propuesta tiene calidad.

Conclusiones

- El estudio sobre los conceptos, principios y patrones del análisis y el diseño, sirvió de base para realizar el análisis y diseño del sistema.
- Se utilizaron técnicas de captura de requisitos, destacándose como más efectiva la Entrevista, esto proporcionó que se obtuvieran los requisitos que verdaderamente responden a las expectativas y necesidades de los clientes.
- Se generaron los artefactos necesarios en cada uno de los flujos de trabajo.
- Se utilizaron patrones de diseño que posibilitan un sistema robusto, flexible y poco complejo.
- Los valores tomados por los indicadores en cada una de las métricas aplicadas demuestran que el análisis y diseño tienen calidad.
- Se cumplió el objetivo por el cuál se llevó a cabo este trabajo. Realizar el análisis y diseño de los casos de uso críticos del sistema Nodo Virtual para la Simulación de Procesos Industriales.

Recomendaciones

- Hacer el análisis y diseño de los casos de uso del sistema que no son significativos para la arquitectura.
- Se propone que se le de seguimiento a este trabajo para lograr un producto de mayor calidad.

Bibliografía

[1] Cano Santamaría, A. "¿Que es un autómata programable?" [Disponible en: **http://www.juntadeandalucia.es/averroes/iespablopicasso/1999/articulos/articulo14.PDF**]

[2] Capote Abreu, J. Alvear Portilla, D. Abreu, Orlando, U. Mariano, L. Espinas Santos, P. "Algunos conceptos y Definiciones del Modelado y Simulación Computacional de Incendios". Grupo GIDAI. Universidad de Cantabria. Marzo, 2006.

[3] Castillo, J.M.C. "Iniciación a los autómatas programables." 2001[Disponible en: **http://olmo.pntic.mec.es/jmarti50/automatas/auto.htm**]

[4] Conferencia 5. Fase de Elaboración. Análisis y Diseño. [Disponible en **http://inter-nos.uci.cu/Teleclases/Teleclases.asp?id_as=12]**

[5] [Disponible en: **http://es.wikipedia.org/wiki/MySQL**].

[6] [Disponible en: **http://es.wikipedia.org/wiki/PostgreSQL**].

[7] [Disponible en: **http://griho.udl.es/mpiua/mpiua/queesprototipo.htm**]

[8] [Disponible en: **http://www.duiops.net/manuales/faqinternet/faqinternet4.htm**]**.**

[9] [Disponible en: **http://www.versioncero.com/noticia/210/visual-paradigm-for-uml**]

[10] Fernando, A. "El Modelamiento del Negocio usando RUP". 2006. [Disponible en: **http://daquilar.evolutionperu.com/?p=7**].

[11] Gamma, E. "Design Patterns. Elements of Reuseable Object-Oriented Software".

[12] Giraldo, Z. "Herramientas de Ingeniería de software".

[13] Gómez, J. "¿Metodología?... Sí, pero, ¿cuál? " 12 de enero del 2006. [Disponible en: **http://www.versioncero.com/articulo/469/metodologiasi-pero-cual**].

[14] Guillemi Martin, J. Rodríguez Villanueva, O. "Análisis y Modelado de la Solución Informática para el proceso de Inscripción en los Registros Mercantiles de la Republica Bolivariana de Venezuela". Caracas, Republica Bolivariana de Venezuela. 2007.

[15] Hosseinzaman, A. Bargiela, A. "ADA's Virtual Node based Water System Simulator". Department of Computing-Nottingham Trent University.

[16] Jacobson, I. Booch, G. Rumbaugh, J. "El Proceso Unificado de Desarrollo de Software" La Habana, Félix Varela, 2004.

[17] Jiménez Alonso, P. "Sistema de Información y Análisis Económico del Grupo de la Electrónica del Ministerio de Informática y las Comunicaciones (GE-MIC)". Universidad de las Ciencias Informáticas. 2007.

[18]: La nueva metodología. [Disponible en: **http://www.programacionextrema.org/articulos/newMethodology.es.html#tth_sEc5.5**]

[19] Larman, C. “UML y PATRONES. Introducción al análisis y diseño orientado a objetos”. La Habana, Félix Varela, 2004.

[20] Limia Navarra, A. “Predictor: Sistema de descarga y procesamiento automatizado de patentes. Diseño del sistema”. Universidad de las Ciencias Informáticas. 2007.

[21] López Barrio, C. “Metodología de Desarrollo: Programación Extrema”. 25 de noviembre del 2005.

[Disponible en: **http://www-lsi.die.upm.es/~carreras/ISSE/programacion_extrema_1.x2.pdf**].

[22] Maier, S; Herrscher, K. “On node virtualization for scalable Network Emulation. 2003. Stuttgart, Alemania.

[23] Macías Hernández, D. “Sistema Integrado de Gestión Estadística, Rol Analista de Sistema, Módulo Generador de Modelos.” Universidad de las Ciencias Informáticas. 2006.

[24]: Metodología ágil para la planificación y seguimiento de proyectos. [Disponible en: **http://oeguzman.googlepages.com/**]

[25] Neil Molino, Z. Bao, R. Fedkiw, A. “Virtual Node Algorithm for Changing Mesh Topology During Simulation”. Stanford University. [Disponible en: **Stanford2004-01.pdf**].

[26] Patrones del “Gang of Four”. Unidad Docente de Ingeniería Del Software. Facultad de Informática. Universidad Politécnica de Madrid.

[27] Pérez Mariñán, P. “Patrones de Diseño (Design Patterns)”.

[28] Pressman, R. “Ingeniería del Software. Un enfoque práctico”. La Habana. Félix Varela. 2005. 601.

[29] Rincón, J. “Cooperación del Personal Académico: Mecanismo para la Integración del Sistema Universitario”. 1998. Universidad Simón Rodríguez. San Fernando de Apure. Venezuela.

[30] Rodríguez, M. Arteaga, A. “¿Qué es un autómata programable?” 2007 [Disponible en: **http://www.pangea.org/iesoa/sp/article.php?id_article=536**].

[31] “Simulador de Pozos de Petróleo”. Laboratorio Nacional de Lawrence Berkeley en EUA.

[32] Sixto. “¿Patrones?”, 2003.

[Disponible en: **http://www.code4net.com/archives/000030.html**].

[33] Visual Paradigm para UML. [Disponible en: **http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(Iglesia_Anglicana)_para_Windows_14718_p/**]

[34] “10 Reasons to Choose Visual Paradigm”.

[Disponible en: **http://www.visual-paradigm.com/aboutus/10reasons.jsp**]

[35] **www.enpresadigitala.net**

Glosario de Términos

NV: Nodo Virtual

Object Management Group: Grupo de Gestión de Objetos.

Tarjetas CRC (Clase - Responsabilidad - Colaborador): Estas tarjetas se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores.

OO: Orientado a Objetos.

XML (Extensible Markup Language): Es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos.

Linux: Sistema operativo.

Windows: Sistema operativo.

PHP: Lenguaje de programación.

Triggers: Objetos relacionados con tablas y almacenados en la base de datos que se ejecutan o se muestran cuando sucede algún evento sobre sus tablas asociadas.

ANEXOS