Universidad de las Ciencias Informáticas
Facultad 3

Título: Arquitectura de un Nodo Virtual de Procesos

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

Autor(es): Susana Gonce Fernández

Tutor(es): Yalice Gámez Batista

Co-tutor: Carlos Yasmany Hidalgo García

Junio 16, 2008

DECLARACIÓN DE AUTORÍA

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

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

Susana Gonce Fernández	Yalice Gámez Batista
_____	_____
Firma del autor	Firma del tutor

Agradecimientos

A May por aguantarme en lo malos momentos

A Ary por no dejarme trabajar.

A Leandro y Yankiel por sus aclaraciones todas horas y desde cualquier lugar.

A Sergito por la ayuda incondicional.

A Rene Lazo, por sacar el tiempo de donde no había para aclarar mis dudas.

A mi tutora por ayudarme en el desarrollo de la tesis.

A mis amigos.

Dedicatoria

A mis padres por estar a mi lado, porque todo se lo debo a ellos.

A mi hermano por apoyarme y escucharme en todas las situaciones.

A Luis Jiménez.

Resumen

Es indudable la importancia adquirida por las nuevas tecnologías de la información y comunicaciones (TIC) en todos los órdenes de la sociedad y, en particular, en el ámbito de la Educación Superior donde han introducido nuevas formas de trabajo, relación e, incluso, cambios en los métodos pedagógicos con los que se superan los métodos tradicionales de difusión de la documentación por parte del profesor (Area Moreira, 2000).

En nuestro país, existen condiciones a nivel de instituciones escolares que propician el uso de las TIC en la educación. Cuba ha hecho una gran inversión en equipamiento para lograr que todos los centro educacionales dispongan de los recursos necesarios para la enseñanza de la informática en todos lo niveles.

En el Instituto Politécnico Julio Antonio Echeverría (CUJAE), Facultad de Ingeniería Eléctrica, se estudia la especialidad de ingeniería en Ingeniería Automática. En esta carrera 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 proveen a los estudiantes de las herramientas prácticas para enfrentarse a una situación real.

Todo esto hace necesario el desarrollo de una herramienta que de manera remota o local, les permita interactuar con diferentes procesos, configurados por ellos o por su profesor, y con el que puedan probar sus estrategias de control.

PALABRAS CLAVE

Simulación, Arquitectura del Software, Nodos Virtuales, Procesos Industriales

Tabla de Contenido

Introducción 1

Capítulo 1: Fundamentación Teórica 5

Introducción 5
1.1 Nodo Virtual de Simulación de Procesos. 5
1.2 Análisis de software similares 6
1.3 Autómatas 8
1.4 Arquitectura de software 9
1. 4. 1 Rol de arquitecto 10
1.5 Protocolos de comunicación a usar 12
1.6 Estilos y patrones de arquitectura 12
1. 6. 1 Estilos 12
1.6.2 Patrones de arquitectura 18
1.7 Fundamentación de los lenguajes de programación, entornos integrados de desarrollo. 20
1.7.1 Lenguajes de programación 20
1.7.2 Entornos de desarrollo integrado 23
1.8 Métrica de diseño arquitectónico 28
Conclusiones 29

Capítulo 2: Características del Sistema 30

Introducción 30
2.1 Requisitos del Nodo Virtual de Procesos. 30
2.2 Propuesta del sistema 31
2.3 Acepciones generales 32
2.4. Requerimientos del sistema 33
2.5 Casos de uso arquitectónicamente significativos 36
2.6 Estilo arquitectónico 38
2.7 Patrones de arquitectura 40
2.8 Lenguaje de programación. Justificación de su uso 42
2.9 Eclipse. Plugins a usar 44
2.10 Gestión de threads 45
2.11 Descripción de la arquitectura 45
2.11.1 Vista del modelo de CU 46
2.11.2 Vista del modelo de análisis 46
2.11.3 Vista del modelo de diseño 51
2.11.4 Vista del modelo de implementación 60
2.11.5 Vista del modelo de despliegue 63
2.12 Validación de la solución propuesta 64
2.12.1 Aplicación de la métrica de Card y Glass 64
2.12.2 Métrica de Hery y Kafura 66
2.12.3 Evaluación de los atributos de calidad 67
Conclusiones 68

Conclusiones 70

Recomendaciones 71

Bibliografía 72

Glosario de términos 77

Anexos 79

Introducción

Es difícil encontrar en nuestros días alguna esfera de la vida en la que no estén presentes las tecnologías de la información y las comunicaciones (TIC), cuyas aplicaciones pueden apreciarse en la economía, negocios, cultura, salud, educación. Un factor que sin dudas influyó mucho en la proliferación de su uso 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 (MSC) es uno de ellos.

La MSC se ha venido desarrollando desde la década de los 40 del siglo pasado (por ejemplo, el MSC utilizado para la investigación y construcción de las primeras bombas atómicas), pero adquieren auge e inusitado impulso a partir de los años 80. Ellas han sido empleadas de forma más intensa en unos campos que en otros, cuestión perfectamente lógica, ya que el MSC frecuentemente es el único método de investigación y análisis disponible para fenómenos y procesos, que no pueden ser reproducidos o experimentados en la realidad, o cuyo costo o peligrosidad hacen la experimentación real impracticable, conjuntamente con la connotación de obtener alguna solución plausible [Capote Abreu, Jorge; Alvear Portilla, Daniel; Abreu, Orlando; Urrutia, Mariano Lázaro; Espinas Santos, Pablo. 2006].

En la actualidad existen herramientas informáticas que permiten realizar simulaciones de procesos, pero todas ellas tienen limitaciones, algunas en cuanto a la cantidad de recursos que necesitan para su implementación, otras en cuanto al número de procesos simultáneos que se pueden simular. Para solucionar estas restricciones se han creado aplicaciones que usan nodos virtuales en los cuales se simulan los procesos.

Estado Unidos, Inglaterra y Alemania son los países que mayor progreso han logrado en el desarrollo de este tipo de software. Por ejemplo: la Universidad de Stuttgart en Alemania implementó una aplicación llamada Network Emulation Testbed [Arsene , Corneliu. 23] con el objetivo de simular redes, en Inglaterra la Universidad de Nottingham Trent creó una aplicación para la simulación de sistema 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 que se han desarrollado Sistemas de Control Industrial (ICS) en nuestro país. Por ejemplo, el Sistema de Control Industrial EROS desarrollado en el año 1991 por ingenieros de la Unión del Níquel en la provincia de Holguín. En la actualidad se está desarrollando en la Universidad de las Ciencias Informáticas (UCI) un sistema SCADA (Supervisory Control and Data Adquisition) Control Supervisor y Adquisición de Datos en español, para el control de los procesos que se desarrollan en las industrias. Su despliegue sera en instituciones venezolanas. También se conoce de la existencia en dicha universidad del Proyecto SIMPRO, que se dedica al desarrollo de simuladores. Sin embargo, estas investigaciones no han hecho uso de nodos virtuales para la simulación de procesos.

Por otra parte, en el Instituto Politécnico José Antonio Echeverría (CUJAE), en la Facultad de Ingeniería Eléctrica, se estudia la especialidad en Ingeniería Automática. En ella se imparten asignaturas de peso 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.

Actualmente para realizar la parte práctica de estas asignaturas se utilizan dos herramientas de simulación LabVIEW y MATLAB, y para correr aplicaciones en tiempo real la herramienta SCADA.

Todo esto hace necesario el desarrollo de una herramienta que reúna en ella características que eliminen las limitaciones que se encuentran en los 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.

Situación problémica

En la actualidad los estudiantes de la carrera de Ingeniería Automática no cuentan con una herramienta que de manera remota o local, les permita interactuar con

diferentes procesos, configurados por ellos o por su profesor, y con el que puedan probar sus estrategias de control. Lo que dificulta la puesta en práctica de los conocimientos adquiridos en clase.

.

Problema

¿Cómo realizar una arquitectura que permita la implementación de un nodo virtual de procesos que facilite y mejore la calidad del aprendizaje de los estudiantes de la carrera de Ingeniería Automática?

Objetivo

Diseñar una arquitectura robusta que permita la implementación de un Nodo virtual para la simulación de procesos industriales en la especialidad en Ingeniería Automática

Para lograr este objetivo se proponen los siguientes **objetivos específicos**:

- Caracterizar los protocolos de comunicación existentes y escoger el más adecuado.
- Caracterizar los estilos arquitectónicos y patrones de arquitectura. Escoger el más adecuado.
- Seleccionar herramientas de desarrollo.
- Definir la arquitectura inicial del sistema.

El **objeto de estudio** es, por tanto, el proceso de desarrollo del software, lo que conlleva a que nuestro **campo de acción** sea el diseño arquitectónico para la implementación de un nodo virtual de proceso

Hipótesis

Si se hace un estudio adecuado de las arquitecturas existentes y se tiene un buen dominio de la tecnología a usar, se obtendrá una arquitectura robusta, flexible y reusable que cumpla con las funcionalidades requeridas en el sistema.

Métodos

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.

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.

Capítulo 1: Fundamentación Teórica

Introducción

En este capítulo se realiza un análisis más profundo de los aspectos técnicos concernientes a los nodos virtuales de simulación de procesos, introduciendo conceptos básicos y ejemplos para una mayor comprensión. Se hace una breve reseña sobre lo que es la arquitectura del software y la importancia del rol del arquitecto dentro del proceso de desarrollo. Se lleva a cabo un estudio de los estilos y patrones arquitectónicos de posible aplicación en la solución, y se fundamenta la utilización de las herramientas en el desarrollo de la misma.

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 [SIG-UPV].

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 [Maier, Herrscher, 2005]. 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 para el proceso en ejecución.
- Cada entorno de ejecución introducido por la virtualización del nodo debe ser tan transparente como sea posible para los restantes. Esto es importante para que la medición de la implementación no sufra modificaciones en comparación con la real. [Maier, Herrscher, 2005].

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 software similares

Las herramientas informáticas que en la actualidad permiten realizar simulaciones de procesos (en tiempo real o no) tienen limitaciones. Algunas necesitan de muchos recursos para ser implementadas, otras 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. [Maier, Steffen; Herrscher, Daniel; Rothermel, Kurt. 2004]

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. [Hosseinzaman, A.; Bargiela, A**.** 1992]

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. [Wu, Y. 1999]

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. [Molino, Neil; Bao, Zhaosheng; Fedkiw, Ron. 2004]

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 el tratamiento de imágenes y para la simulación de procesos que por su complejidad no pueden ser realizadas por las herramientas tradicionales. Por estas razones es necesario que el Nodo Virtual de Procesos incorpore muchas características que no la poseen dichas aplicaciones, como es le uso de los autómatas.

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. [Castillo, J.M.C. 2001]

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 [Rodríguez, Minerva; Arteaga, Arantza. 2007]:

- 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 [startMedia. 27] 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 Arquitectura de software

Cuando se va a construir una edificación primero se crea la estructura: cimientos, columnas, vigas, plantas, después se ensamblan las distintas partes: paredes, suelos, puertas y ventanas, instalaciones eléctricas. La primera sirve de soporte a las segundas, y estas a su vez aportan la mayoría de las funcionalidades básicas del inmueble.

Pero antes de este proceso se lleva a cabo otro paso: la definición de la arquitectura del edificio. Esta tarea es realizada por el arquitecto, de quien depende el éxito de la construcción del predio. Si buscamos en el Diccionario de la Real Academia Española podemos ver que la arquitectura es: "el *arte de proyectar y construir edificios*". [Real Academia Española. 5]

Después de la llamada revolución informática y con el desarrollo de la industria del software, apareció otra acepción de arquitectura: arquitectura del software (AS). Según el Diccionario de la Real Academia Española la AS es la "e*structura lógica y física de los componentes de un computador*" [Real Academia Española. 5] la cual está más bien centrada en toda la estructura y composición de la parte dura de los ordenadores (hardware).

En la actualidad podemos encontrar disímiles conceptos del término "arquitectura del software". Por ejemplo, Paul Clements plantea que para él la AS es, a grandes

rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se percibe desde el resto del sistema y la forma en que los componentes interactúan y se coordinan para alcanzar la misión del sistema.

Por su parte David Garlan establece, de modo muy general, que la AS constituye un puente entre el requerimiento y el código, ocupando el lugar que en los gráficos antiguos se reservaba para el diseño. Posteriormente se acordó que la definición oficial de AS sea la que brinda el documento de IEEE Std 1471-2000. Esta definición fue adoptada también por Microsoft y plantea lo siguiente: La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. [Billy Reynoso, Carlos. 2]

Según la definición de Bass la arquitectura de software de un sistema de programa o computación es la estructura de las estructuras del sistema, la cual comprende los componentes visibles externamente, y las relaciones entre ellos [Pressman, Roger S. 2005].

Bass, Clements y Kazman [Bass, Len, Clements, Paul y Kazman, Rick. 2003] argumentaron en tres razones el porqué de la importancia de la AS:

- Las representaciones de la arquitectura de software facilitan la comunicación entre todas las partes interesadas en el desarrollo de un sistema basado en computadora. La arquitectura destaca decisiones tempranas de diseño que tendrán un profundo impacto en todo el trabajo de ingeniería del software que sigue, y es tan importante en el éxito final del sistema como una entidad operacional.
- La arquitectura constituye un modelo relativamente pequeño e intelectualmente comprensible de cómo está estructurado el sistema y de cómo trabajan juntos sus componentes.

1. 4. 1 Rol de arquitecto

En el proceso de desarrollo de software la definición de la arquitectura es uno de los pasos más importantes debido a que esta define la estructura del sistema. Por

esa razón el arquitecto debe ser capaz de [Suarez López, Juan Carlos; Corrales Valdés, Yurisnel. 2007]:

- Descomponer la aplicación en capas, al menos, lógicas. La descomposición en capas lógicas debe ser tal que pueda acomodarse a más de una descomposición en capas físicas, en la medida en que la aplicación -por razones de escalabilidad y/o rendimiento- deba re desplegarse.
- Descomponer cada capa lógica en componentes. Esto es, asignar claramente las responsabilidades de ejecución, ya sea en clases concretas o abstractas, o quizás también en interfaces que posean contratos completamente definidos, de manera tal que los desarrolladores de lógica de aplicación completen su implementación.
- Seleccionar tecnologías y/o frameworks de implementación. El arquitecto debe ser un experto en manipular la complejidad. En la tarea de seleccionar las tecnologías a utilizar el arquitecto debe ser capaz de adaptarlas de manera que se aprovechen la potencia y las funcionalidades que estas ofrecen pero tratando de evitar los mecanismos complejos o tratándolos a un nivel superior de la arquitectura de forma que no se tenga que obligar a todo el equipo a interactuar con las partes más complejas.
- Realizar una prueba de concepto de la arquitectura. La prueba de concepto es una etapa necesaria para validar la arquitectura, para decidir si seguir adelante o someterla a ajustes o revisiones ulteriores. Pero fundamentalmente, es un elemento que al arquitecto le debe proveer confianza en sus decisiones, y a la vez inspirar confianza en el resto de los interesados en la aplicación.
- Brindar algunos casos de uso de referencia. Tiene como finalidad explicar como juegan, en la práctica, la arquitectura y los componentes a implementar por los desarrolladores. Cuanto más representativos y emblemáticos sean los casos de uso, tanto mejor. Esto le va a dar no sólo retroalimentación al arquitecto respecto de la arquitectura que definió, sino que además le va a permitir ganar confianza respecto de cuán asimilable es su arquitectura.

El arquitecto de software es el principal responsable de las decisiones que se tomen respecto a la manera en que será construida la aplicación. Es el mayor apoyo del líder del proyecto y de su trabajo depende alcanzar o no el éxito del

proyecto optimizando el uso de la tecnología para desarrollar la solución correcta que proporcionará valor real a sus usuarios y al negocio al que le dará soporte

1.5 Protocolos de comunicación a usar

Los protocolos son una serie de estándares que definen las condiciones de la comunicación entre los sistemas de cómputo, por lo que están presentes en todas las fases de conexión de los mismos. O sea, establecen una descripción formal de los formatos que deberán presentar los mensajes para poder ser intercambiados por las computadoras y definen las reglas que ellos deben seguir para lograrlo. [Pingarron, Raúl.]

Para la comunicación entre los equipos que formaran la red donde se ejecutara nuestra aplicación se usaran los protocolos:

TCP/IP

El TCP/IP es la base de Internet, y sirve para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo PC, minicomputadoras y computadoras centrales sobre redes de área local (LAN) y área extensa (WAN). TCP/IP fue desarrollado y demostrado por primera vez en 1972 por el departamento de defensa de los EEUU ejecutándolo en ARPANET, una red de área extensa del departamento de defensa. Estos protocolos, TCP e IP, fueron los que primero se definieron y por sus características son los más usados.

- IP: El protocolo IP es parte integral del TCP/IP. Las tareas principales del IP son el direccionamiento de los datagramas de información y la administración del proceso de fragmentación de dichos datagramas.

- TCP: El TCP (Transmition Control Protocol) o protocolo de control de trasmisión es la otra parte integral del protocolo TCP/IP. Sus características son: Servicio Orientado a Conexión, Conexión de Circuito Virtual, Flujo no estructurado, Conexión Full Dúplex.

1.6 Estilos y patrones de arquitectura

1. 6. 1 Estilos

La aplicación de estilos arquitectónicos es primordial en el desarrollo de cualquier software. Su principal característica es prevenir que durante el desarrollo del

sistema este no se desvíe de su diseño original. Además, permiten sintetizar estructuras de soluciones y evaluar arquitecturas alternativas con ventajas y desventajas conocidas ante diferentes conjuntos de requerimientos no funcionales. Constituyen el preámbulo para la elección de los patrones de arquitectura a utilizar.

Arquitectura basada en eventos.

Las arquitecturas basadas en eventos, también son conocidas como arquitecturas de invocación implícita [Shaw, Mary; Garlan, David. 1996], integración reactiva o difusión selectiva. En la actualidad existen estrategias de programación basada en eventos, sobre todo referida a interfaces de usuario, y hay además eventos en los modelos de objetos y componentes, pero no es a eso a lo que se refiere principalmente el estilo, aunque esas consideraciones no están del todo excluidas.

El patrón de diseño que más estrechamente está relacionado con este estilo es el que se conoce con el nombre de Observer, este término se hizo popular a principios de los ochenta en Smalltalk y que en el mundo de Java se le conoce como Modelo de delegación de eventos [Larman, Craig. 1999].

Este tipo de arquitecturas históricamente han estado vinculadas con sistemas basados en actores, daemons y redes de conmutación de paquetes (publicación-suscripción). Los conectores de estos sistemas incluyen procedimientos de llamada tradicionales y vínculos entre anuncios de eventos e invocación de procedimientos. La idea dominante en la invocación implícita es que, en lugar de invocar un procedimiento en forma directa (como se haría en un estilo orientado a objetos) un componente puede anunciar mediante difusión uno o más eventos. Un componente de un sistema puede anunciar su interés en un evento determinado asociando un procedimiento con la manifestación de dicho evento [Billy Reynoso, Carlos y Kiccillof, *Nicolás. 2004*]

Muchos son los sistemas que utilizan esta arquitectura. Este estilo es utilizado en ambientes de integración de herramientas, en sistemas de gestión de base de datos para asegurar las restricciones de consistencia, en interfaces de usuario para separar la presentación de los datos de los procedimientos que gestionan datos y en editores sintácticamente orientados para proporcionar verificación semántica incremental. Las reglas de Microsoft para el uso del modelo de eventos

señalan que este estilo puede utilizarse cuando [Billy Reynoso, Carlos y Kiccillof, *Nicolás. 2004*]:

- Se desean manejar independientemente y de forma aislada diversas implementaciones de una "función" específica.
- Las respuestas de una implementación no afectan la forma en que trabajan otras implementaciones.
- Todas las implementaciones son de escritura solamente o de dispararse-y-olvidar, tal que la salida del proceso de negocios no está definida por ninguna implementación, o es definida sólo por una implementación de negocios específica.

Entre las principales ventajas de este estilo arquitectónico se encuentran:

- Se optimiza el mantenimiento haciendo que procesos de negocios que no están relacionados sean independientes.
- Se alienta el desarrollo en paralelo, lo que puede resultar en mejoras de performance.
- Es fácil de empaquetar en una transacción atómica.
- Se puede agregar un componente registrándolo para los eventos del sistema; se pueden reemplazar componentes.

Y como desventajas están que:

- El estilo no permite construir respuestas complejas a funciones de negocios.
- Un componente no puede utilizar los datos o el estado de otro componente para efectuar su tarea.
- Cuando un componente anuncia un evento, no tiene idea sobre qué otros componentes están interesados en él, ni el orden en que serán invocados, ni el momento en que finalizan lo que tienen que hacer. Pueden surgir problemas de performance global y de manejo de recursos cuando se comparte un repositorio común para coordinar la interacción.

Arquitecturas en Capas

Este estilo se define [Garlan, David; Shaw, Mary. 1994] como una jerarquía de capas, tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. En algunas implementaciones las capas internas están ocultas a todas las demás, menos

para las capas externas adyacentes, y excepto para funciones puntuales de exportación. En la práctica, las capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas

Las restricciones topológicas del estilo pueden incluir una limitación, más o menos rigurosa, que exige a cada capa operar sólo con capas adyacentes, y a los elementos de una capa entenderse sólo con otros elementos de la misma. Esto permite el reemplazo de las capas sin que las otras se vean afectadas. Dadas las características de este estilo las ventajas son obvias [Billy Reynoso, Carlos y Kiccillof, *Nicolás. 2004*]:

- Soporta un diseño basado en niveles de abstracción crecientes.
- Permite a los programadores la partición de un problema complejo en una secuencia de pasos incrementales.
- Admite optimizaciones y refinamientos.
- Proporciona amplia reutilización.
- Provee la posibilidad de definir interfaces de capa estándar.

Es valido señalar que la aplicación de este estilo puede traer complicaciones pues muchos problemas no admiten un buen mapeo en una estructura jerárquica, a veces es también extremadamente difícil encontrar el nivel de abstracción correcto y puede complicar, a veces de manera no razonable, el desarrollo de aplicaciones simples. [Layered application. 9]

Estilos orientados a servicios

Recientemente ha sido que estas arquitecturas llamadas SOA han recibido un tratamiento intensivo en el campo de exploración de los estilos, Alrededor de los Web services, que dominan el campo de un estilo SOA, se han generado las discusiones que usualmente acompañan a toda idea exitosa, al principio hasta resultaba difícil encontrar una definición aceptable y consensuada. El grupo de tareas de W3C, por ejemplo, demoró un año y medio en ofrecer la primera definición canónica apta para consumo arquitectónico, y que se cita a continuación:

Un Web Service es un sistema de software diseñado para soportar interacción máquina-a-máquina sobre una red. Posee una interfaz descripta en un formato procesable por máquina. Otros sistemas interactúan con el Web Service de una

manera prescripta por su descripción utilizando mensajes SOAP, típicamente transportados usando Hypertext Transfer Protocol (HTTP1) con una serialización en e**X**tensible **M**arkup **L**anguage (XML2) en conjunción con otros estándares relacionados a la Web [Billy Reynoso, Carlos; Kiccillof, *Nicolás. 2004*].

En la literatura clásica referida a estilos, las arquitecturas basadas en servicios se ajustan con lo que Garlan & Shaw definen como el estilo de procesos distribuidos. Otros autores hablan de Arquitecturas de Componentes Independientes que se comunican a través de mensajes. Según esta perspectiva, existen dos variantes de este estilo:

- Participantes especificados (named): Estilo de proceso de comunicación. El ejemplar más conocido sería el modelo cliente-servidor. Si el servidor trabaja sincrónicamente, retorna control al cliente junto con los datos; si lo hace asincrónicamente, sólo retorna los datos al cliente, el cual mantiene su propio hilo de control.
- Participantes no especificados (unnamed): Paradigma publish/subscribe, o estilo de eventos.

Es fundamental destacar la forma en la cual este estilo redefine los viejos modelos de ORPC propios de las arquitecturas orientadas a objetos y componentes.

La descripción, publicación, descubrimiento, localización e invocación de los Web Services se puede hacer en tiempo de ejecución, de modo que los servicios que interactúan pueden figurarse la forma de operar de sus contrapartes, sin haber sido diseñados específicamente caso por caso. Desde el punto de vista arquitectónico, se puede hacer ahora una breve caracterización de este estilo:

- Un servicio es una entidad de software que encapsula funcionalidad de negocios y proporciona dicha funcionalidad a otras entidades a través de interfaces públicas bien definidas.
- Los componentes del estilo (o sea los servicios) están débilmente acoplados. El servicio puede recibir requerimientos de cualquier origen. La funcionalidad del servicio se puede ampliar o modificar sin rendir cuentas a quienes lo requieran.
- Los componentes que requieran un servicio pueden descubrirlo y utilizarlo dinámicamente mediante UDDI y sus estándares sucesores. En general

(aunque hay alternativas) no se mantiene persistencia de estado y tampoco se pretende que un servicio recuerde nada entre un requerimiento y el siguiente.

Un servicio puede incluir muchas interfaces y poseer propiedades tales como descripción de protocolos, puntos de entrada y características del servicio. Algunas de estas notaciones son provistas por lenguajes declarativos basados en XML, como WSDL (Web Service Description Language).

En el futuro próximo, el modelo de programación de Longhorn permitirá agregar a la ya extensa funcionalidad de los servicios una nueva concepción relacional del sistema de archivos, soporte extensivo de shell y prácticamente toda la riqueza del API de Win32 en términos de código manejado [Billy Reynoso, Carlos; Kiccillof, Nicolás. 2004].

Arquitectura basada en componentes

Los sistemas de software basados en componentes se basan en principios definidos por una ingeniería de software específica [Brown, Alan; Wallnau, Kurt. 1998], en un principio, hacia 1994, se planteaba como una modalidad que extendía o superaba la tecnología de objetos.

Dentro de las tantas definiciones que se han dado de componentes se encuentra la de Clemens Alden Szyperski; que proporciona una que es bastante operativa: Un componente de software, es una unidad de composición con interfaces especificadas contractualmente y dependencias del contexto explícitas [Clemens Alden Szyperski, 2002]. El hecho que sea una unidad de composición y no de construcción quiere decir que no es preciso confeccionarla: se puede comprar hecha, o se puede producir para que otras aplicaciones la utilicen en sus propias composiciones. También se puede definir un componente como un artefacto diseñado y desarrollado de acuerdo ya sea con CORBA Component Model (CCM), JavaBeans y Enterprise JavaBeans en J2EE. En un estilo como este:

- Los componentes son las unidades de modelado, diseño e implementación.
- Las interfaces están separadas de las implementaciones, y las interfaces y sus interacciones son el centro de incumbencias en el diseño arquitectónico.

En cuanto a las restricciones de este estilo, puede admitirse que una interfaz sea implementada por múltiples componentes. Usualmente, los estados de un

componente no son accesibles desde el exterior [Alden Szyperski, Clemens. 2002]. Que los componentes sean locales o distribuidos es transparente en la tecnología actual. La evaluación dominante del estilo de componentes subraya su mayor versatilidad respecto del modelo de objetos, pero también su menor adaptabilidad comparado con el estilo orientado a servicios.

1.6.2 Patrones de arquitectura

Los patrones son una serie de soluciones a problemas comunes que se relacionan con diferentes aspectos del proceso de desarrollo de software. En dependencia del problema que resuelven estos patrones se agrupan en diferentes clasificaciones.

La investigación se centrara en los patrones de arquitectura. Estos patrones ofrecen esquemas de organización general de un sistema que especifican una serie de subsistemas y sus responsabilidades e incluyen reglas para organizar las relaciones entre ellos [Canal, Carlos. 2005].

1. Table Data Gateway: Para la aplicación de este patrón se crea un objeto que actúa como “gateway” a una tabla de la base de datos, por tanto, debe crearse uno por cada tabla que se tenga definida. Contienen una interface que permite actualizar, buscar, borrar e insertar en la tabla y pueden retornar un registro, un grupo de registro, y hasta un objeto del dominio. [López, Ángel. 2005]

2. Row Data Gateway: Este patrón es similar al Table Data Gateway, pero cada objeto representa una fila de la tabla, no una tabla completa. Tiene propiedades que reflejan las columnas de la tabla, y métodos de actualización en la base. [López, Ángel. 2005]

3. Data Access Object: Este patrón encapsula la forma de acceder a la fuente de datos, resolviendo el problema de contar con diversas fuentes de datos como son las bases de datos, los archivos, los servicios externos. Su uso se extiende al problema de ocultar la forma de acceder a los datos. Se trata de que el software cliente se centre en los datos que necesita y se

olvide de cómo se realiza el acceso a los datos o de cual es la fuente de almacenamiento [Lago, Ramiro.2007].

4. Active Record: Al igual que el Row Data Gateway el objeto representa una fila de una tabla. Contiene propiedades de actualización. [López, Ángel. 2005]

5. Data Mapper: Como su nombre lo indica mapea un registro en la base a un objeto del domino, permitiéndonos separar un objeto del dominio de sus correspondientes datos en la base. [López, Ángel. 2005]

6. Unit of Work: Este patrón separa los objetos que van a ser afectados por una transacción, haciendo más fácil coordinar la actualización de los cambios y la resolución de problemas de concurrencia. [López, Ángel. 2005]

7. Identity Map: Permite la recuperación de objetos. Estos se cargan en memoria, aumentando el rendimiento y el consumo de memoria. Complica la actualización concurrente. [López, Ángel. 2005]

8. Service Layer: Este patrón se utiliza cuando se necesita definir la frontera de una aplicación, lo que se logra implementando una capa de servicios que otras aplicaciones pueden consumir. Expone las operaciones, y coordina su ejecución y respuesta. Otra de sus características se comunica con el Domain Model o Table Module. [López, Ángel. 2005]

9. Domain Model: Es un modelo que abarca los objetos del negocio en el que se representa la conducta y los datos en cada uno de ellos. Este modelo es el más cercano a la programación orientada a objetos. Por lo general los datos son mapeados a través del Active Record o el Data Mapper. [López, Ángel. 2005]

10. Template View: Se usa en aplicaciones web, pues produce una presentación HTML de los datos. Las distintas tecnologías de Server Pages son implementaciones de este patrón. [López, Ángel. 2005]

11. Model View Controller: Sugiere la separación en tres roles, que interactúan entre si, de la capa de presentación. El modelo representa los datos del dominio, la vista permite mostrarlo en la presentación y el controlador reacciona y atiende los gestos del usuario. [López, Ángel. 2005]

12. Page Controller: Plantea que debe crearse un objeto que maneje los pedidos para una página, y decide qué modelo y vista producir. [López, Ángel. 2005]

13. Front Controller: Es un controlador que maneja todos los pedidos de un sitio web. En general, tiene un "handler" que dado el pedido, decide qué operación ejecutar. [López, Ángel. 2005]

Dada las características del sistema los patrones que se aplicaran son: Data Access Object, Domain Model y Modelo-Vista-Controlador. En el epígrafe 2.7 del Capitulo 2 se argumentan las razones de su elección.

1.7 Fundamentación de los lenguajes de programación, entornos integrados de desarrollo.

1.7.1 Lenguajes de programación

Los lenguajes de programación constituyen el medio a través del cual podemos comunicarnos con los ordenadores. En otras palabras, un lenguaje de programación es una notación para escribir programas, a través de los cuales podemos comunicarnos con el hardware y dar así las ordenes adecuadas para la realización de un determinado proceso. Su evolución comienza con la creación en el año 1900 del lenguaje binario, pero es en realidad a partir del 1946 que estos alcanzan su verdadero auge.

Lenguaje de programación C++

El C++ es un lenguaje de programación que fue diseñado a mediados de la década de los 80 como extensión del lenguaje de programación C. El nombre de C++ fue idea de *Rick Masciatti,* que haciendo uso del operador post-incremento del lenguaje (++) sugería que era el lenguaje que le seguía a C. A medida que este compilador fue creciendo se le incorporaron características de otros

lenguajes, aunque siempre mantuvo compatibilidad con C. Por ejemplo: incorpora clases y funciones virtuales basándose en SIMULA67 tipos genéricos y expresiones de ADA, la posibilidad de declarar variables en cualquier lugar de ALGOL68, así como otras características que no existían antes y de las cuales se hablaran mas adelante. También se puede decir que es un lenguaje multiparadigma, pues abarca tres paradigmas de la programación:

- Programación estructurada: El compilador de C++ admite sentencias estructuradas debido a que mantiene características de su antecesor, el lenguaje C, que es un lenguaje estructurado de nivel medio.
- Programación genérica: La programación genérica es un tipo de programación que está mucho más centrada en los algoritmos que en los datos. La idea de esta forma de programar pretende generalizar las funciones utilizadas para que puedan usarse en más de una ocasión.
- Programación orientada a objetos: C++ permite el desarrollo de aplicaciones usando objetos y sus interacciones para representar la solución de un problema.

En la actualidad existen varias implementaciones de dicho compilador tanto en plataformas libres como privadas. Por ejemplo: GCC (Gnu Compiler Collection), Microsoft Visual C++, Borland C++ Builder, Dev-C++, C-Free. Dentro de las nuevas características que le fueron implementadas al C++ están:

- Facilidades que proporciona para la programación orientada a objetos como la herencia múltiple y la gestión de excepciones.
- Facilidades para el uso de plantillas o programación genérica (templates).
- Brinda la posibilidad de redefinir los operadores (sobrecarga de operadores)
- Hace la identificación de tipos en tiempo de ejecución (RTTI), lo que le da más velocidad frente a otros lenguajes.
- La sintaxis de clases y objetos permite manipular convenientemente diversas estructuras de datos y operaciones.
- Permite la creación de estructuras de datos muy poderosas y fuertemente integradas

Actualmente se le considera un lenguaje de alto nivel, pues una sola de sus instrucciones equivale a millares de líneas de código maquina. Estas y otras características hacen que, hoy por hoy, C++ sea uno de los lenguajes de programación más robustos y utilizados.

Entre las principales desventajas que presenta dicho lenguaje es que hasta el momento no cuenta con un entorno de desarrollo libre que sea puramente de el. Existen IDE que tienen una versión de su compilador, lo que no nos garantiza un desarrollo estable y con calidad.

Lenguaje Java

Sus inicios datan de 1991 cuando James Gosling (en Sun Microsystems) encabezó un proyecto cuyo objetivo original era implementar un lenguaje de programación (Java) y una máquina virtual (JVM) ampliamente portable. Dicho lenguaje adopta una sintaxis muy similar a la del lenguaje C++, aunque eliminando algunas de sus características más oscuras, entre las que podemos mencionar:

Además de las bibliotecas de C, cuenta con una biblioteca estándar de plantillas (STL) que proporciona una serie de clases parametrizadas que permite efectuar operaciones sobre el almacenado de datos, procesado y flujos de entrada/salida.

- Punteros: El inadecuado uso de los punteros provoca la mayoría de los errores de colisión de memoria, errores muy difíciles de detectar. Además, existen virus que aprovechan la capacidad de un programa para acceder a la memoria volátil (RAM) utilizando punteros. En java no existen punteros
- Variables globales: Pueden provocar efectos laterales en cualquier función, e incluso se pueden producir fallos catastróficos cuando algún otro método cambia el estado de la variable global necesaria para la realización de otros procesos. En java lo único global son las clases.
- Asignación de memoria: Tanto C, con las funciones malloc y free, como C++, con las funciones new y delete, dejan en manos del programador la responsabilidad de liberar el espacio en memoria. Java usa el operador new, pero cuenta con un garbage collector que se encarga de la liberación de la memoria que no se esta utilizando.
- Conversión de tipos insegura: Los moldeados de tipo (type casting) permiten en C/C++ cambiar el tipo de un puntero. Esto requiere extremada

precaución puesto que no hay nada previsto para detectar si la conversión es correcta en tiempo de ejecución. En Java se puede hacer una comprobación en tiempo de ejecución de la compatibilidad de tipos y emitir una excepción cuando falla.

Se debe tener en cuenta también que Java proporciona tipos de datos primitivos y mediante su librería de clases estándar proporciona todas las estructuras contenedoras "clásicas". Tiene gran portabilidad ya que la máquina virtual (Java Virtual Machine JVM) es la encargada de interpretar el código del programa cuando se compila, convirtiéndolo en un conjunto de bytecodes independientes. Por eso el programa puede luego ser ejecutado en cualquier lugar siempre que esté instalada la JVM. Otra característica importante, y una de las más conocidas, es que pone al alcance de cualquiera la utilización de aplicaciones que se pueden incluir directamente en páginas Web. Estas aplicaciones se denominan *applets.*

1.7.2 Entornos de desarrollo integrado

Un entorno de desarrollo integrado, (IDE Integrated Development Environment) es un entorno de programación empaquetado como un programa, que brinda una serie de herramientas y facilidades a los programadores y pueden centrarse en uno o a varios lenguajes [Wikipedia]. A continuación se describen los IDE posibles a usar en el desarrollo del Nodo Virtual de Procesos.

KDevelop

El Proyecto KDevelop surgió en 1998 con el fin de desarrollar un IDE (Entorno de desarrollo integrado) fácil de usar para KDE(**K D**esktop **E**nvironment). Desde entonces está públicamente disponible bajo licencia GPL y soporta lenguajes de programación como: C, C++, Java, Ada, SQL, Python, Perl y Pascal. Solo corre en sistemas Linux y otros sistemas Unix. Su ultima versión es la 3.5 y salio el 16 de octubre del 2007. Entre las características que podemos mencionar de este IDE están:

- Editor de código fuente con destacado de sintaxis e indentado automático (Kate).
- Navegador entre clases de la aplicación.
- Front-end para GCC (GNU Compiler Collections)

- Front-end para el depurador de GNU.
- Asistentes para generar y actualizar las definiciones de las clases y el framework de la aplicación.
- Completado automático del código en C y C++.
- Compatibilidad nativa con Doxygen (Doxygen es un acrónimo de dox(document) gen(generator), generador de documentación para código fuente.)
- Permite control de versiones.

En la primera de las características se menciona el uso del editor de texto Kate. Kate significa **K**DE **A**dvanced **T**ext **E**ditor, es decir Editor de textos avanzado para KDE. Las características que más nos interesan de este editor son:

- Destacado de sintaxis.
- Búsqueda y reemplazo de texto usando expresiones regulares
- Seguimiento de código para C++, C y otros.
- Soporte de sesiones
- Manejador de archivos
- Emulador de terminal basado en Konsole: Konsole es un emulador de terminal desarrollado para el proyecto KDE. Soporta operaciones de edición (copiado, pegado, arrastres de texto) e impresiones.

Debemos mencionar también que KDE, es un entorno de escritorio gráfico e infraestructura de desarrollo para sistemas Unix y, en particular, Linux. Según definen en su sitio oficial: KDE es un entorno gráfico contemporáneo para estaciones de trabajo Unix. KDE llena la necesidad de un escritorio amigable para estaciones de trabajo Unix, similar a los escritorios de MacOS o Windows».

NetBeans

NetBeans es el nombre de una plataforma para el desarrollo de aplicaciones de escritorio usando Java y de un Entorno integrado de desarrollo (IDE) desarrollado usando dicha plataforma. Sun MicroSystems fundó el proyecto de código abierto en junio 2000, aunque fue en el año 1996 que surgieron las primeras ideas para su desarrollo en la Republica Checa. , y continúa siendo el patrocinador principal.

Las aplicaciones creadas en la plataforma son desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un modulo es un archivo que contiene clases java, que interactúan con las APIs de NetBeans, y un archivo, manifest file, que lo identifica como un módulo. Esta forma de implementar las aplicaciones da la ventaja de que pueden ser extendidas añadiéndole módulos nuevos. Los módulos pueden ser desarrollados de forma independiente.

✓ IDE NetBeans

Esta escrito completamente en Java, pero puede servir para otro lenguaje de programación. Es un sistema de proyecto basado en Ant, control de versiones y refactoring. La versión 5.5 fue lanzada en mayo del 2007. Entre las características que posee se pueden mencionar las siguientes:

- Constituye el primer y único IDE disponible de forma gratuita que ofrece soporte completo a Java EE 5.
- Garantiza una apariencia y funcionamiento común de las aplicaciones una vez desplegadas sobre diversos entornos, como Solaris, GNU/Linux, Microsoft Windows y MacOS.
- Web Application Development: Permite crear una página funcional completa JSF para manipular los datos de una base de datos. Viene con un editor visual para descriptores de despliegue y un seguimiento de los errores de HTTP Web.
- servicios orientados a arquitectura porfessionales. Escribir, probar y depurar aplicaciones utilizando XML, BPEL o los servicios web de java
- C/C++: El C/C++ Pack añade todas las funcionalidades al IDE necesarias para el desarrollo de aplicaciones nativas de C/C++.
- UML: Con el modelado UML que ofrece NetBeans los analistas pueden diseñar al aplicación usando UML y luego generar el código desde el diagrama UML. Esto puede hacerse en ambos sentidos, pues se puede actualizar el modelo si se hacen cambios en el código fuente. Realiza el control de versiones: El control de versiones es perfectamente integrado en el flujo de trabajo del IDE, este usa la versión que tiene o se actualiza directamente del centro de actualizaciones.

- Tiene un avanzado editor de código fuente: Analiza el código mientras se va escribiendo, las palabras claves, paréntesis, marca los errores. Puede ser totalmente personalizado.
- Servicios Orientados a arquitectura (Service-Oriented Architecture): El Enterprise Pack añade todas las funcionalidades necesaria al IDE para servicios orientados a arquitectura profesionales.

✓ Plataforma Netbeans

Es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Ofrece servicios comunes a dichas aplicaciones, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Ofrece administración de interfaz de usuario (menús y barras de herramientas), de configuraciones de usuario, de almacenamiento (guardando y cargando cualquier tipo de dato), de ventanas.
- Ofrece un framework basado en asistentes (diálogos paso a paso)
- Es consistente: Las aplicaciones basadas en NetBeans una vez hechas corren en cualquier lugar. Se puede precompilar componentes y resolver problemas comunes con la reutilización. Usa los módulos que se desarrollan como base para aplicaciones que comparten la misma lógica.
- Modularidad: Las aplicaciones basadas en esta plataforma pueden instalar módulos dinámicamente, de modo que no hay necesidad de descargar toda la aplicación para obtener una actualización o una nueva versión. Incluso se pueden montar módulos que ya están hechos por otros, beneficiándose así el programador de las características del código abierto, pues hay un gran número módulos implementados por la comunidad NetBeans que están listos para ser incorporados.

Eclipse

Eclipse es un IDE multiplataforma desarrollado por IBM. En la actualidad lo mantiene la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos, capacidades y servicios complementarios.

Pese a que Eclipse esté escrito en su mayor parte en Java (salvo el núcleo), se ejecute sobre máquina virtual de ésta y su uso más popular sea como un IDE para Java, Eclipse es neutral y adaptable a cualquier tipo de lenguaje, por ejemplo C/C++, Cobol, C#, XML, etc.

Inicialmente presentaba varios componentes como: la Plataforma Principal (inicio y ejecución de plug-ins), es Standar Widget Toolkit (portables widgets) y el Workbench (vistas, editores, perspectivas y asistentes). Actualmente emplea módulos (plug-ins) para adicionar funcionalidades según las necesite el desarrollador, a diferencia de otros entornos monolíticos que las tienen todas incluidas, sean necesarias o no. Gracias a estos plug-ins se ha extendido el soporte de Eclipse hasta lenguajes de procesado de texto como LaTeX, aplicaciones de red como Telnet, Sistemas de Gestión de Bases de Datos (DBMS), además de los lenguajes de programación antes mencionados. Independientemente de lo antes expuesto el uso de plug-ins constituye también una desventaja, pues la configuración de algunos de ellos puede volverse engorrosa y muy difícil. Para la gestión de la configuración y el control de versiones tiene soporte para CVS y Subversion, incluye plug-ins para realizar pruebas de unidad (Junit, CxxTest).

La detección de errores sintácticos está muy completa, porque no sólo detecta la mayoría de los errores, sino que da alternativas.

Eclipse esta formado por:

- Núcleo: Su tarea es determinar cuales son los plug-ins disponibles en el directorio de plug-ins de Eclipse. Cada plugin tiene un fichero XML manifest que lista los elementos que necesita de otros plug-ins así como los puntos de extensión que ofrece.
- Entorno de trabajo (Workspace): Maneja los recursos del usuario, organizados en uno o más proyectos. Cada proyecto corresponde a un directorio en el directorio de trabajo de Eclipse, y contienen archivos y carpetas.
- Interfaz de usuario (Workbench): Muestra los menús y herramientas, y se organiza en perspectivas que configuran los editores de código y las vistas.

- Ayuda al grupo (Team support): Este plug-in facilita el uso de un sistema de control de versiones para manejar los recursos en un proyecto del usuario y define el proceso necesario para guardar y recuperar de un repositorio. Eclipse incluye un cliente para CVS.
- Documentación (Help): Al igual que el propio Eclipse, el componente de ayuda es un sistema de documentación extensible.

En todo lo antes mencionado se habla de los plug-ins. Un plug-in es la mínima unidad de la plataforma que puede ser desarrollado por separado y que la aporta una nueva funcionalidad. Se instalan descomprimiendo el zip del plug-in en el directorio plugins de Eclipse. Ante plug-ins con mismo nombre, Eclipse selecciona la última versión.

Aunque su ejecución requiere el uso de una gran cantidad de recursos, se puede llegar a la conclusión de que Eclipse constituye una de las mejores herramientas para el desarrollo de aplicaciones libres, por lo que será el IDE que usaremos.

1.8 Métrica de diseño arquitectónico

1.8.1 Métrica de Card y Glass

Esta métrica, propuesta en el año 1990, define tres medidas de complejidad de diseño del software [Pressman, Roger. 2005]

1. Complejidad estructural (S(i)): Se define de la siguiente manera:

$$S(i) = f_{out}^2(i)$$

   Donde f out (i) es la expansión del módulo i, que no es mas que la cantidad de módulos inmediatamente subordinados al módulo i.

2. Complejidad de datos (D(i)): Proporciona una indicación de la complejidad en la interfaz interna de un módulo i y se define como:

$$D(i) = \frac{v(i)}{[f_{out}(i)+1]}$$

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

3. Complejidad del sistema (C(i)): Se define como la suma de las complejidades estructural y de datos, y se define como:

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

La complejidad arquitectónica del sistema aumenta a media que los valores de complejidad hallados van creciendo.

1.8.2 Métrica de Henry y Kafura

Henry y Kafura definen una métrica de complejidad que emplea la expansión y la concentración y se expresa con la siguiente fórmula [Pressman, Roger. 2005]
:

$$MHK = longitud(i) + [fin\ (i) + fout(i)]^2$$

Donde la longitud(i) es el número de sentencias en lenguaje de programación para el módulo i y fin(i) la concentración del módulo i. La concentración y la expansión no sólo incluyen el número de conexiones de control del módulo (llamadas al módulo), sino que también incluyen el número de módulos del que el módulo i recoge (concentración) o actualiza (expansión) datos.

Conclusiones

Después hacer un análisis de las posibles herramientas a usar en el desarrollo del nodo virtual, de hacer un estudio de algunos software existentes relacionados, se concluye que el mismo será desarrollado usando el entorno de desarrollo Eclipse con C++ como lenguaje. Por las ventajas que ofrece se utilizará la arquitectura en capas aplicando el patrón Data Access Objetc en la capa de acceso a datos, el patrón Domain Model en la capa de negocio y el patrón Modelo-Vista-Controlador en la capa de presentación.

Capítulo 2: Características del Sistema

Introducción

En este capitulo se realizara una propuesta del sistema y de solución al problema planteado. Haciendo uso de la estructura estándar del documento de arquitectura de software utilizado en nuestra universidad se detalla la línea base de la arquitectura a través de la justificación del uso de los patrones, estilo y lenguaje de programación propuestos y de la descripción de las vistas arquitectónicas.

En el presente capitulo se hará la validación a través de métricas de la solución propuesta. Se emplearan las métricas que se describieron en el Capitulo 1 epígrafe 1.8 y una evaluación a los atributos de calidad.

2.1 Requisitos del Nodo Virtual de Procesos.

Esta aplicación permitirá interactuar con los modelos de diferentes procesos, ya sea a través de simulaciones como probando aplicaciones en tiempo real. Estos modelos se agruparán por tipos y serán accedidos por aplicaciones clientes que se conectarán al nodo mediante un protocolo establecido. El nodo virtual permitirá la conexión en red tanto desde una computadora personal como desde un autómata.

El número de procesos activos simultáneamente es limitado, así como el número de clientes por proceso. Existen además niveles de prioridades para la cantidad de procesos por tipo que se pueden activar.

Los clientes tendrán diferentes niveles de privilegio: cliente común, cliente maestro y cliente administrador. Estos niveles se escogen al iniciar el trabajo desde una aplicación remota. Cada proceso activo tendrá un cliente maestro y varios clientes comunes. Los clientes comunes son aquellos que trabajan con un proceso ya activado y el número de ellos por proceso será definido por el sistema. Los parámetros del proceso serán establecidos por el cliente maestro, y los clientes comunes trabajarán con dichos parámetros. Estos clientes pueden trabajar como espectadores del proceso o podrán interactuar con una instancia del proceso al que se hayan suscrito.

El cliente administrador no actúa sobre los modelos sino que tiene acceso a otro tipo de información y ejecuta otras tareas. Esta categoría implica que el cliente administrador debe poseer un nombre de usuario y contraseña para poder registrarse como tal. De esta manera se podrá lograr la medición de los efectos de las acciones de control sobre los procesos.

2.2 Propuesta del sistema

Se propone realizar la arquitectura 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.
- 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.

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.

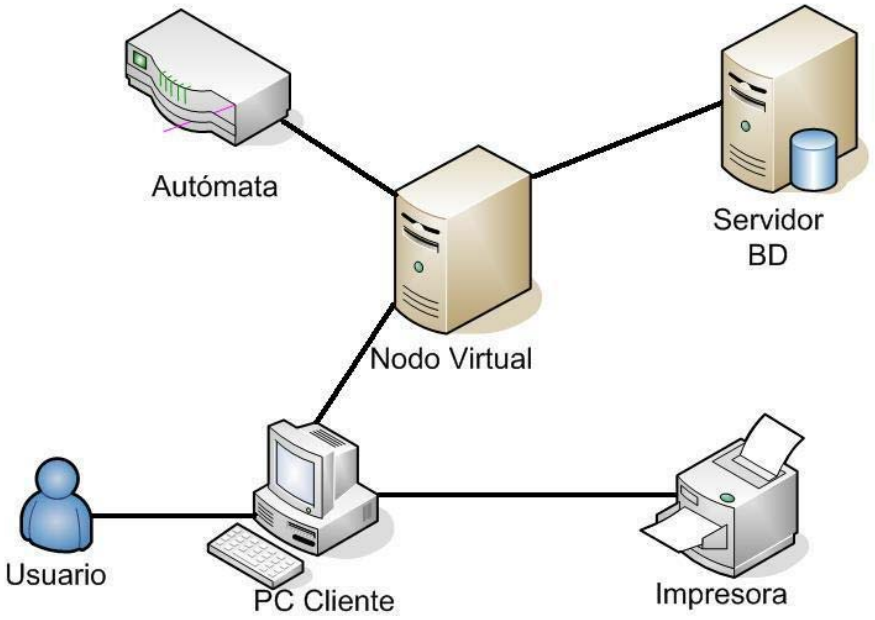

FIGURA 1

2.3 Acepciones generales

Para el desarrollo del Nodo virtual se utilizara la metodología de desarrollo de software Proceso Unificado de Desarrollo (Rational Unified Process), definido como 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. Utiliza UML (Unified Modeling Languaje) o Lenguaje de Modelación Unificado para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software, por lo que este será el lenguaje de modelado que usaremos. UML, como lo indica su nombre, 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).

En la actualidad existen varias herramientas CASE (Computer Aided Software Engineering) orientadas a UML. De todas ellas usaremos el Visual Paradigm. Es importante destacar que el Visual Paradigm como herramienta de modelado posee licencia gratuita. Dada la necesidad de la persistencia de los datos utilizaremos las facilidades que brindan los SGBD. En nuestro caso haremos uso del MySQL.

2.4. Requerimientos del sistema

Existen diversos conceptos que definen a los requerimientos del sistema. Por ejemplo la IEEE Standard Glossary of Software Engineering Terminology define un requerimiento como “condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo”. Estos requerimientos incluyen el sistema operativo, el lenguaje de programación, la configuración del hardware, el ancho de banda, la velocidad de procesamiento. ”

Estos requerimientos se dividen en dos clasificaciones: Funcionales, No funcionales

2.4.1 Funcionales

R1. Conexión Sistema

R1.1. Conectarse al sistema.

R1.2. Registrarse

R1.3. Autenticarse

R1.4. Desconectarse del sistema.

R2. Conexión Proceso

R2.1. Conectarse a un proceso.

R2.2. Desconectarse de un proceso.

R2.3. Asignar roles.

R3. Desconectar a un usuario de un proceso.

R4. Modificar contraseña

R5. Gestionar el 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 el resultado final de la simulación.

R10.5. 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 el resultado final de la simulación.

R11.7. Terminar un proceso.

R12. Representar simulación de forma grafica

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.

2.4.2 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.

- 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).
5. Se deben borrar los usuarios de la base de datos cuando se termine la simulación.

2.5 Casos de uso arquitectónicamente significativos

Los casos de uso importantes arquitectónicamente son los que condicionan la arquitectura del sistema. Su correcta definición nos da una visión del tipo de arquitectura que se debe utilizar. En su definición RUP plantea que los casos de uso críticos son aquellos que reflejan una funcionalidad o tarea esencial para los clientes.

Conexión al sistema (Conetar_Sistema).

La función de este caso de uso es permitir la conexión de un usuario al sistema. En caso de que haya disponibilidad para el acceso el usuario debe registrarse poniendo un nombre de usuario antes definido por su profesor, el cual se guardara en la base de datos mientras la simulación del proceso se efectué. Si es un administrador debe autenticarse. En caso de no haber disponibilidad de conexión, el sistema muestra un mensaje informándole al usuario que no hay capacidad. (Ver Anexo 1)

Conexión a un proceso (Conectar_Proceso).

Tiene la responsabilidad de conectar a los usuarios a los diferentes procesos que pueden simularse. Al solicitar la conexión el sistema verifica que el proceso este activo. Si lo esta verifica disponibilidad de acceso y lo asigna al proceso deseado. En caso de que el proceso no este activo, el sistema lo activa y le asigna el rol de cliente maestro. (Ver Anexo 2)

Establecer limites (Establecer_Limites)

Este caso de uso tiene la responsabilidad de establecer los límites necesarios para el buen funcionamiento de la aplicación. Consta de tres escenarios. (Ver Anexo 3)

1. Limitar el número de usuarios conectados al sistema: Cuando el administrador de la aplicación selecciona la opción de limitar la cantidad de usuarios conectados el sistema solicita los datos necesarios y realiza la acción.
2. Limitar el número de usuarios conectados por procesos: Cuando el administrador de la aplicación selecciona la opción de limitar la cantidad de usuarios conectados a un proceso el sistema solicita los datos necesarios y realiza la acción.
3. Limitar el número de procesos simultáneos: Limitar el total de procesos simultáneos es una acción que realiza el cliente administrador. Cuando este solicita la acción el sistema pide los datos necesarios y la realiza.

Gestionar Tipos de Procesos (Gestionar_Tipo_Proceso).

Da la posibilidad al administrador del sistema de añadir, modificar o eliminar un tipo proceso. El sistema para cualquiera de estas acciones solicita la información necesaria para realizar la acción. (Ver Anexo 4)

Gestionar Modelo de Procesos (Gestionar_Modelo_Proceso).

Da la posibilidad al administrador del sistema de añadir, modificar o eliminar un modelo de proceso. El sistema para cualquiera de estas acciones solicita la información necesaria para realizar la acción. (Ver Anexo 5)

Configurar Proceso (Configurar_Proceso)

Cuando un cliente quiere simular un proceso el sistema le pide los datos necesarios para la misma. Con esta información el sistema configura el proceso y puede iniciar la simulación. (Ver Anexo 6)

Simular Proceso (Simular_Proceso).

El CUS se inicia cuando el Cliente Maestro de un proceso selecciona la opción de Simular Proceso después que este ha sido configurado. Durante el tiempo que dure la simulación el cliente puede reconfigurar el proceso, o sea, el sistema le permite cambiar los parámetros de un proceso. Esta acción solo puede ser realizada por el cliente maestro de un proceso. El sistema tomara los nuevos valores y realizara la simulación, actualizando los datos del proceso. (Ver Anexo 7)

Probar aplicaciones (Probar_Aplicaciones)

Este caso de uso permite al usuario escoger la opción de controlar el proceso de simulación a través de un autómata. (Ver Anexo 8)

Gestionar Controlador (Gestionar_Controlador)

Da la posibilidad al administrador del sistema de añadir y eliminar un controlador. El sistema para cualquiera de estas acciones solicita la información necesaria para realizar la acción. (Ver Anexo 9)

2.6 Estilo arquitectónico

Con el objetivo de aislar el negocio de la interfaz del usuario y del manejo de la persistencia aplicaremos en la implementación del sistema 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. [García de la Puente, Sergio Jesús; Plasencia Crespo, Mileisys. 2007]

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. [García de la Puente, Sergio Jesús; Plasencia Crespo, Mileisys. 2007]

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. [García de la Puente, Sergio Jesús; Plasencia Crespo, Mileisys. 2007]

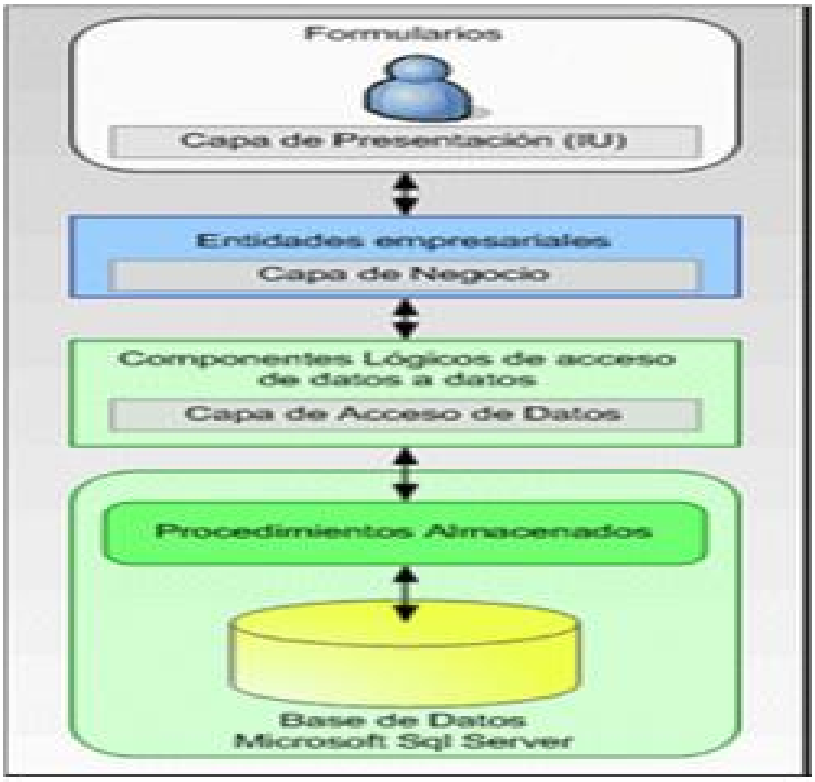

FIGURA 2

En el caso específico del Nodo Virtual la capa de presentación estaría compuesta por todos los elementos de la interfaz: representaciones graficas 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. La capa de negocio abarca los módulos de Simulación, 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 datos. La distribución de los módulos por capa se refleja en el siguiente grafico.

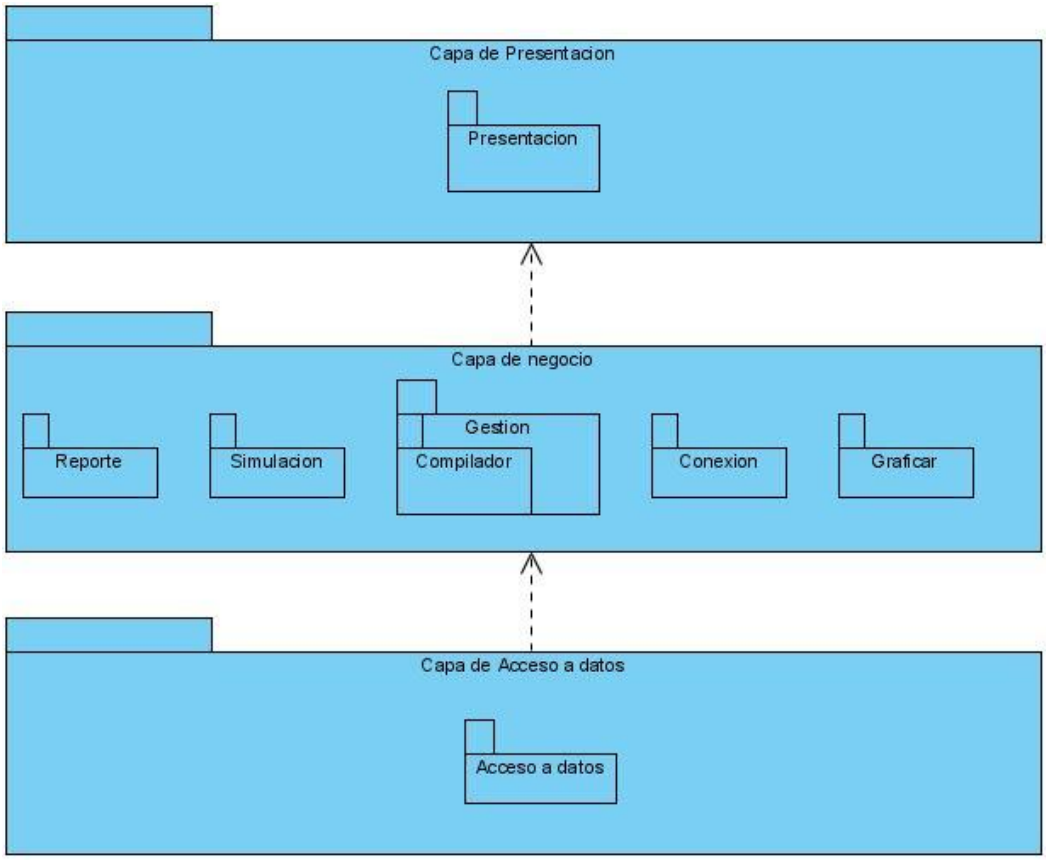

FIGURA 3

2.7 Patrones de arquitectura

Un patrón es una solución probada que se puede ser aplicada con éxito a un determinado tipo de problemas que aparecen repetidamente en algún campo.

Por tanto, un patrón codifica conocimientos específicos acumulados por la experiencia, describiendo un problema que ocurre una y otra vez en nuestro ambiente y luego el núcleo de la solución a ese problema [Perera Morales, José Raúl. 2007].

Durante la implementación del Nodo virtual se aplicaran 3 patrones de arquitectura, los que se mencionaran según la capa en la que puedan aplicarse.

Capa de acceso a datos

- Nombre: Data Access Object (DAO)
- Problema: Este patrón encapsula la forma de acceder a la fuente de datos, resolviendo el problema de contar con diversas fuentes de datos como son

las bases de datos, los archivos, los servicios externos. Su uso se extiende al problema de ocultar la forma de acceder a los datos. [Lago, Ramiro. 2007].

- Solución: La creación de los Objetos de Acceso a Datos (DAO) permite acceder a los datos a través de estos objetos y no tener que cambiar los objetos de negocio.

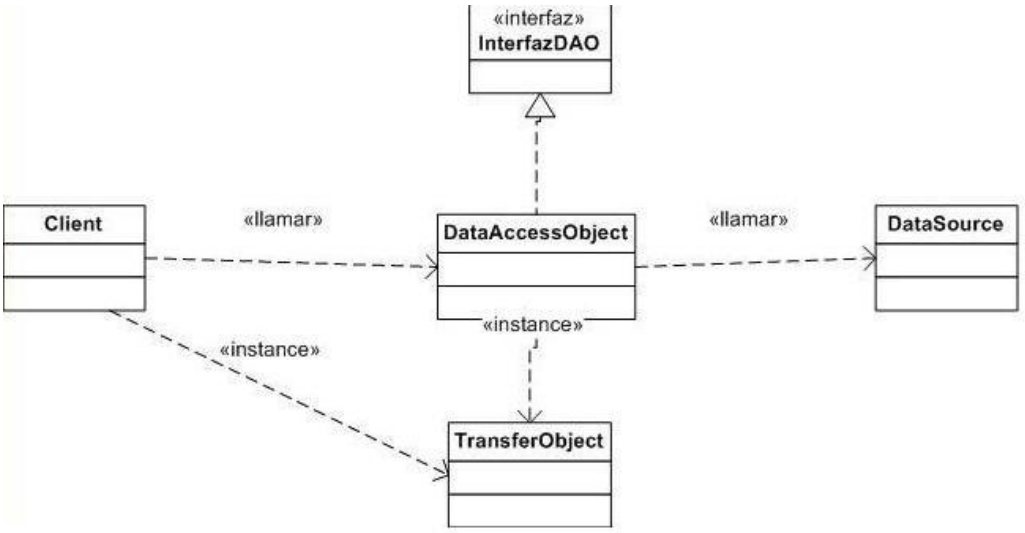

FIGURA 4

Se aplica porque el sistema debe ser capaz de guardar y cargar las configuraciones de las simulaciones lo mismo desde una base de datos como desde un archivo.

Capa de negocio

- Nombre: Domain Model

- Problema: La lógica del dominio de un sistema es compleja lo que hace difícil la representación de los conceptos y reglas del negocio.

- Solución: Permite expresar en objetos todos los componentes y abstracciones, dando todos los beneficios de la programación Orientada a objetos.

Capa de presentación

- Nombre: Modelo - Vista – Controlador

- Problema: La lógica de un interfaz de usuario cambia con más frecuencia que los almacenes de datos y la lógica de negocio. Si realizamos un diseño complicado, es decir, una mezcla de los componentes de interfaz y de negocio, entonces la consecuencia será que, cuando se necesite cambiar el interfaz, se tendrá que modificar trabajosamente los componentes de negocio. Mayor trabajo y más riesgo de error. [Welicki, 2007]

- Solución: Sugiere la separación en tres roles, que interactúan entre si, de la capa de presentación. El modelo representa los datos del dominio, la vista permite mostrarlo en la presentación y el controlador reacciona y atiende los gestos del usuario.

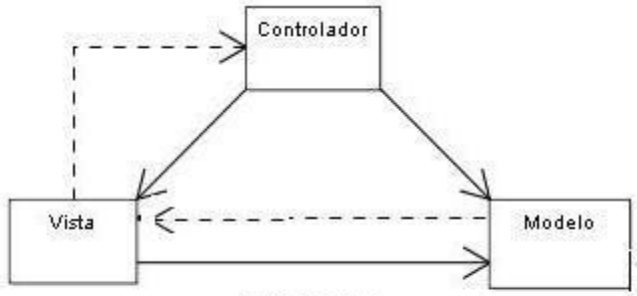

FIGURA 5

Se considero la aplicación de este patrón porque nos ofrece ventajas como la adaptación a posibles cambios y mostrar los mismos datos en múltiples vistas de manera simultanea.

2.8 Lenguaje de programación. Justificación de su uso

Para el desarrollo del nodo virtual se compararon en el capitulo anterior dos lenguajes de programación: C++ y Java. Por sus características generales se escogió el lenguaje C++. Para una mayor compresión de la elección, teniendo en cuenta los requerimientos del software, se establecieron una serie de criterios: portabilidad, capacidades 2D/3D, matemáticas de precisión compleja, gestión de memoria, velocidad de ejecución, licencia, eficiencia y modularidad.

Portabilidad.

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

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

- Características dependientes de la implementación: Lo que permite realizar fuertes optimizaciones en distintas arquitecturas, resulta con frecuencia una pesadilla para la portabilidad. Muchos detalles importantes son dejados a criterio de quien escribe el compilador, tales como los tamaños de diversos tipos de datos, juegos de caracteres, comportamiento ante ciertos errores, etc.
- Acceso a librerías del sistema operativo: Las interfaces y librerías principales no han seguido un proceso de estandarización tan riguroso como el lenguaje, lo que ha traído como consecuencia diversas soluciones incompatibles para los mismos problemas. Estrictamente este no es un problema del lenguaje, sino más bien de la plataforma utilizada (por ejemplo, las variantes de Unix.)

Capacidades 2D/3D.

C++ depende en este apartado por completo en bibliotecas externas al propio lenguaje, esto supone si se quiere portabilidad usar capas intermedias como vxWindows para 2D. [Lucas Rodríguez, Fernando. 2005].

Matemática de precisión compleja.

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

Le sigue a cierta distancia C++, mucho más estructurado y “moderno” pero menos especializado en este ámbito. Es muy frecuente usar bibliotecas que internamente hacen uso de código FORTRAN. [Lucas Rodríguez, Fernando. 2005].

Gestión de memoria.

En este aspecto C++ es lo más eficiente, ya que se tiene control absoluto de la memoria. Se pueden usar todas las técnicas de manejos de punteros, conteo de referencias. [Lucas Rodríguez, Fernando. 2005].

Velocidad de ejecución.

C++ es bastante eficiente, sobre todo si se trata de un compilador capaz realizar optimizaciones sobre el código. [Lucas Rodríguez, Fernando. 2005].

Licencia.

Existen variantes tanto comerciales como abiertas.

Eficiencia.

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

Por lo tanto, todos los lenguajes de programación deben ser "traducidos" en algún momento a "lenguaje máquina" para que los programas sean ejecutados; simplificando, a este proceso se le suele denominar "compilación" y tanto el lenguaje C como el lenguaje C++ siguen este esquema de ser "compilados" al "lenguaje máquina" del procesador en el que se van a utilizar. [AmericaTI EIRL, 2006].

Modularidad.

La modularidad [Tucker, 1992] es un referido a la posibilidad de desarrollar componentes de manera independiente, los que eventualmente interactuarían. Es ese sentido, los lenguajes analizados permiten desarrollar funciones, clases, y paquetes de modo independiente, cada cual con sus convenciones particulares.

2.9 Eclipse. Plugins a usar

Como antes se había mencionado el IDE en el que se desarrollara el nodo virtual es Eclipse. El plugins necesario para programar en C/C++ en este IDE es el CDT, C/C++ Development Tooling, que provee todas las funcionalidades necesarias para el trabajo con dichos lenguajes. Para dar formato a los archivos .c, .h y .cpp es necesario instalar el plugin Simple Ident. La última versión del CDT disponible, desde el 26 de febrero de este año, es la 4.0.3. [Eclipse] Para la compilación del programa se pueden instalar los compiladores gcc o g++ en Linux y en el caso de Windows MinGW o Cygwin.

2.10 Gestión de threads

Un nodo virtual es una aplicación que nos permite correr diferentes instancias de un mismo software en un único nodo físico y cada instancia del software trabaja en un entorno de ejecución independiente. Partiendo de este concepto es que se establece la gestión de los hilos en el Nodo Virtual. Se debe tener en cuenta que un hilo debe ser totalmente independiente de los demás para evitar conflictos e intromisiones de un hilo en otro, asegurando la ejecución independiente de cada uno sin que exista interrupción o ruptura. Para lograr que los hilos no colapsen entre si se implementara un gestor para administrar los hilos, la concurrencia, y las notificaciones.

2.11 Descripción de la arquitectura

Con este documento se pretende proporcionar un mayor entendimiento de la arquitectura global del sistema a través de las vistas arquitectónicas y de la exposición de los elementos más importantes que definen la misma.

Vistas Arquitectónicas

La arquitectura de un software tiene entre sus componentes varias vistas que describen diferentes dimensiones o perspectivas del mismo. Ninguna constituye por si sola la arquitectura del sistema, sino que en su conjunto muestran los elementos más importantes que se obtienen en cada fase.

2.11.1 Vista del modelo de CU

Esta vista es una representación de los casos de uso que describen alguna funcionalidad importante y crítica, a la que debe dársele primacía a la hora de implementar el sistema. Esta vista se utiliza como entrada cuando se priorizan los casos de uso para su desarrollo en cada iteración. [Jacobson, Ibar; Booch, Grady; Rumbaugh, James. 200]

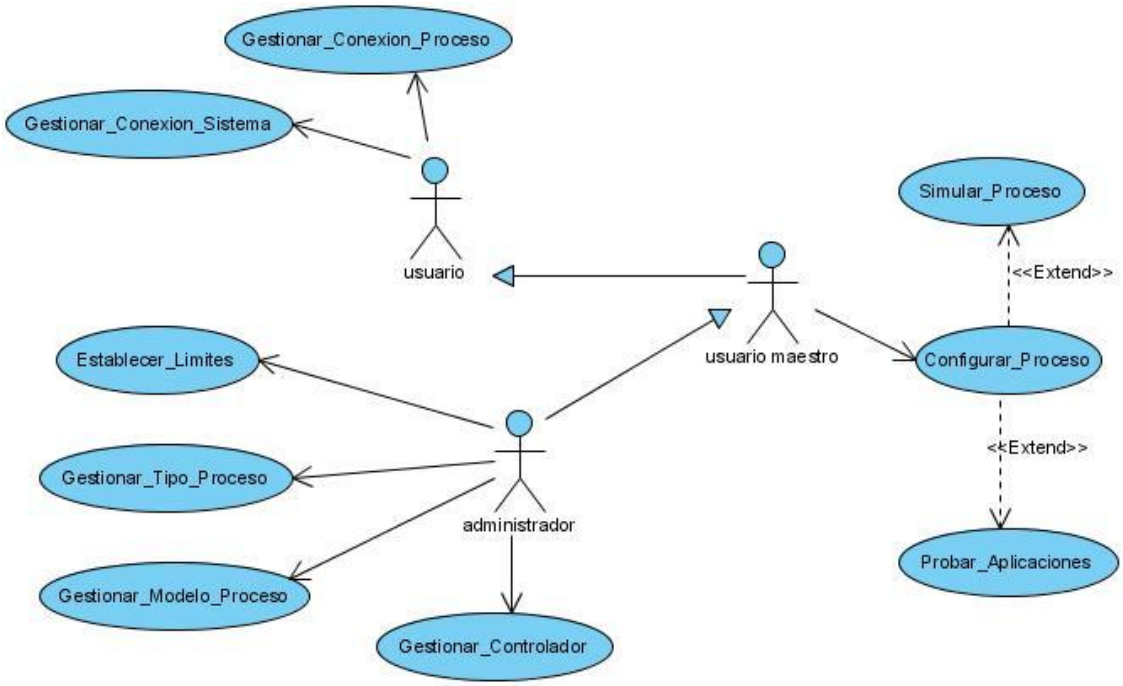

FIGURA 6

2.11.2 Vista del modelo de análisis

Esta vista muestra los artefactos del modelo de análisis que son significativos para la arquitectura. Por lo general los artefactos significativos son [Jacobson, Ibar; Booch, Grady; Rumbaugh, James. 200]:

- Clases de análisis que son generales, centrales y que tienen muchas relaciones con otras clases.
- Clases entidad que encapsulan un fenómeno importante del dominio del problema.
- Clases interfaz que encapsulan interfaces de comunicación importantes y mecanismos de interfaz de usuario.

- Clases de control que encapsulan importantes secuencias con un amplia cobertura, o sea, que coordinan realizaciones de casos de uso significativos.

En el análisis las clases más importantes para la arquitectura son las representadas en los diagramas siguientes que se presentaran según el módulo al que pertenecen.

Módulo Simulación

Caso de uso Simular.

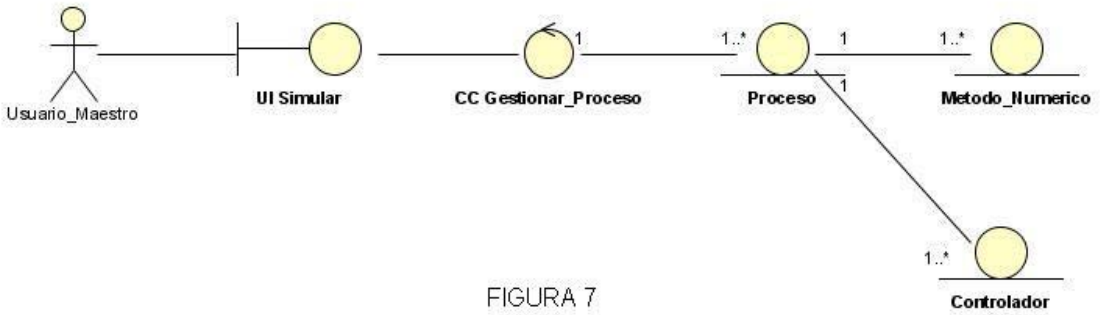

FIGURA 7

Justificación:

- UI Simular: Interfaz que permite al usuario ver el proceso de simulación.
- UI Configurar: Interfaz que permite la configuración del proceso a simular.
- CC Gestionar_Proceso: Es la encargada de gestionar todo lo concerniente a la simulación de un proceso determinado, o sea, procesa la información que entra el usuario y lleva a cabo la simulación.
- Proceso: Clase que representa a un proceso.
- Metodo_Numerico: Clase que representa a un método numérico.
- Controlador: Clase que representa a un controlador

Caso de uso Configurar

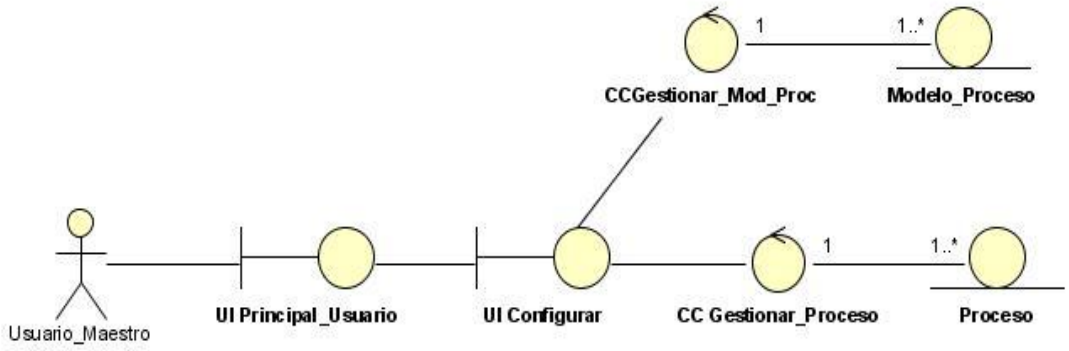

FIGURA 8

Justificación:

- UI Principal_Usuario: Interfaz que permite a los usuarios comunes y maestros acceder a todas las funcionalidades del sistema definidas para su nivel de privilegio.
- UI Configurar: Interfaz que permite la configuración del proceso a simular.
- CC Gestionar_Proceso: Es la encargada de gestionar todo lo concerniente a la simulación de un proceso determinado, o sea, procesa la información que entra el usuario y lleva a cabo la simulación.
- CC Gestionar_Mod_Proc: Es la encargada de gestionar la información de los modelos de los procesos.
- Proceso: Clase que representa a un proceso.
- Modelo_Proceso: Clase que representa a un modelo de proceso.

Caso de uso Probar_Apliaciones

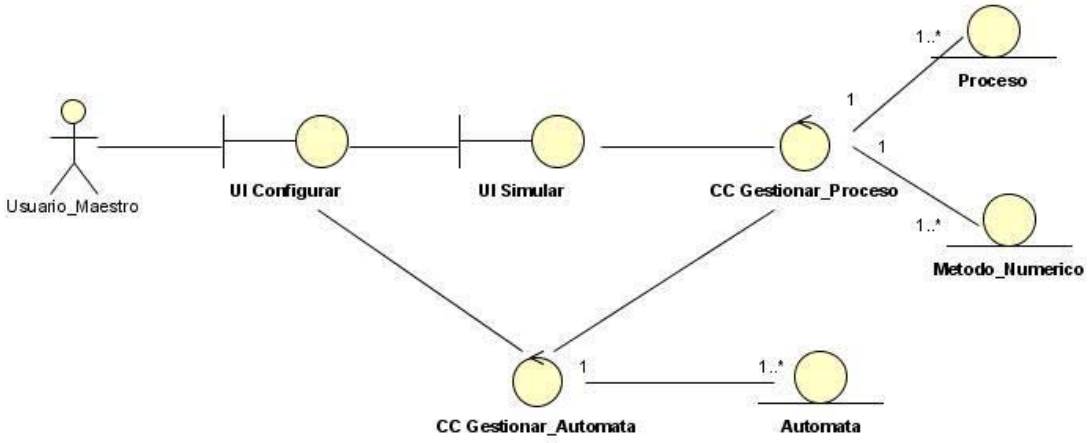

FIGURA 9

Justificación:

- UI Simular: Interfaz que permite al usuario ver el proceso de simulación.
- UI Configurar: Interfaz que permite la configuración del proceso a simular.
- CC Gestionar_Proceso: Es la encargada de gestionar todo lo concerniente a la simulación de un proceso determinado, o sea, procesa la información que entra el usuario y lleva a cabo la simulación.
- CC Gestionar_Automata: Se encarga de gestionar la conexión con el autómata y la información que este necesita para funcionar como controlador.
- Proceso: Clase que representa a un proceso.
- Autómata: Clase que representa al autómata
- Metodo_Numerico: Clase que representa a un método numérico.

Módulo Conexión

Caso de uso Conexión_Sistema

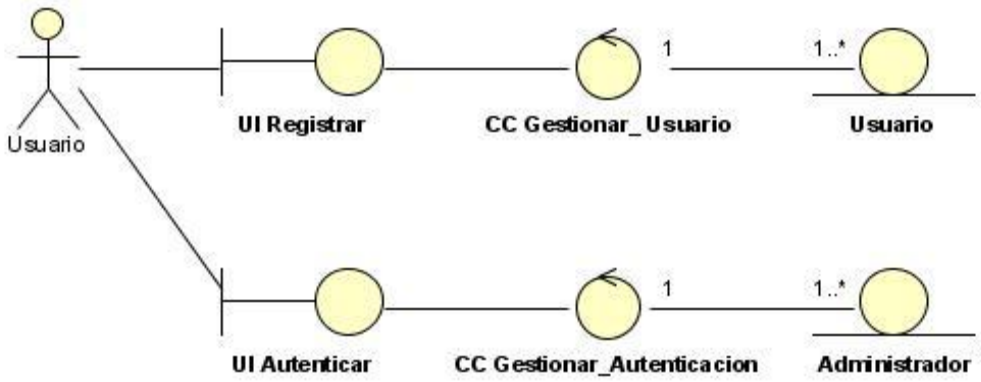

FIGURA 10

Justificación

- UI Registrar: Clase interfaz que tiene como función de esta el control de acceso de los usuarios al sistema.
- UI Autenticar: Clase interfaz que permite a los administradores entrar sus datos para autenticarse.
- CC Gestionar_Usuario: Clase de control que se encarga de gestionar todos los usuarios conectados al sistema.
- CC Gestionar_Autenticacion: Su función principal es verificar los datos del administrador para dar acceso al modulo de administrador.

- Usuario: Clase entidad que representa a un usuario.
- Administrador: Su función principal es guardar los datos de los administradores.

Caso de uso Conexión_Proceso.

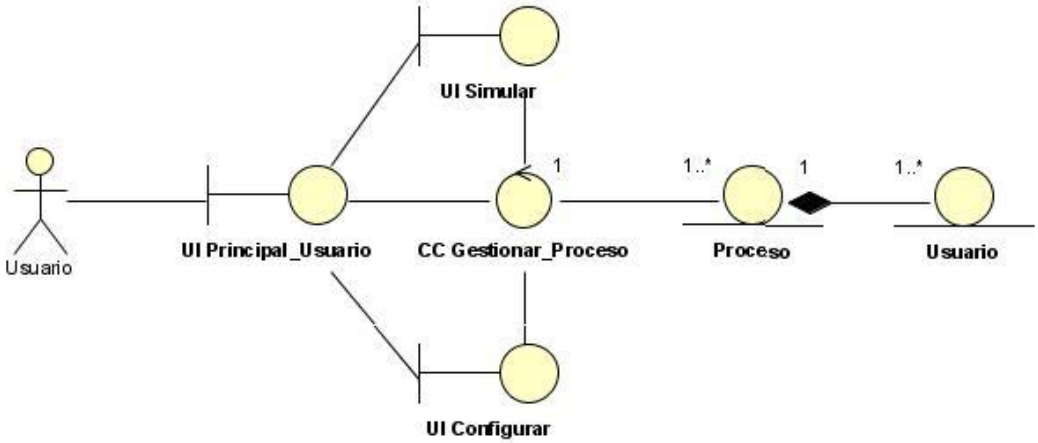

FIGURA 11

- UI Principal_Usuario: Interfaz que permite a los usuarios comunes y maestros acceder a todas las funcionalidades del sistema definidas para su nivel de privilegio.
- UI Simular: Interfaz que permite al usuario ver el proceso de simulación.
- CC Gestionar_Proceso: Es la encargada de gestionar todo lo concerniente a la simulación de un proceso determinado, o sea, procesa la información que entra el usuario y lleva a cabo la simulación.
- UI Configurar: Interfaz que permite al usuario entrar toda la información necesaria para realizar la simulación de los procesos.
- Proceso: Clase que representa a un proceso.
- Usuario: Clase que representa a un usuario.

Modulo Gestión

Caso de uso Establecer_Limite

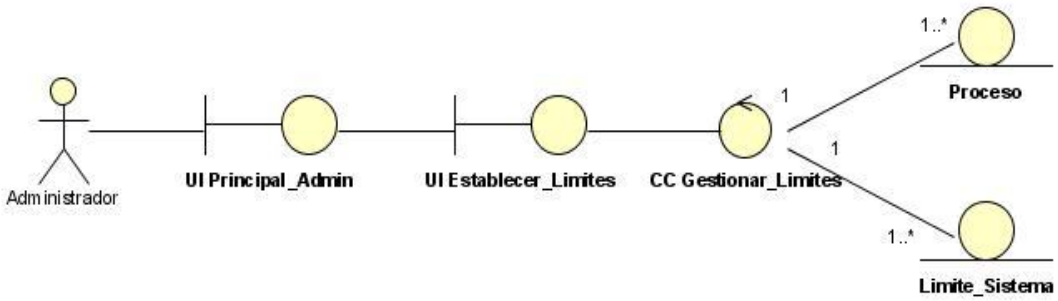

FIGURA 12

Justificación:

- UI Principal_Admin: Interfaz que permite a los usuarios comunes y maestros acceder a todas las funcionalidades del sistema definidas para su nivel de privilegio.
- UI Establecer_Limite: Interfaz que permite al administrador entrar los datos necesarios para establecer los limites en cuanto a la cantidad de usuarios conectados al sistema, la cantidad de procesos activos y la cantidad de usuarios conectados a un proceso.
- CC_Gestionar_Limite: Es la encarga de gestionar los limites que introduce el administrador para el buen funcionamiento del sistema**.**
- Limite_Sistema: Clase entidad que guarda los limites establecidos
- Proceso: Clase que representa a un proceso.

2.11.3 Vista del modelo de diseño

Esta vista describe los artefactos del modelo de diseño que son significativos para la arquitectura. Según plantean Jacobson, Booch, Rumbaugh en su libro “El Proceso Unificado de Desarrollo”, los artefactos significativos son:

- La descomposición del modelo de diseño en subsistemas, sus interfaces y las dependencias entre ellos.
- Clases de diseño que posean una traza con clases de análisis significativas.
- Clases de diseño generales y centrales que representen mecanismos de diseño genérico o que tengan muchas relaciones con otras clases de diseño.

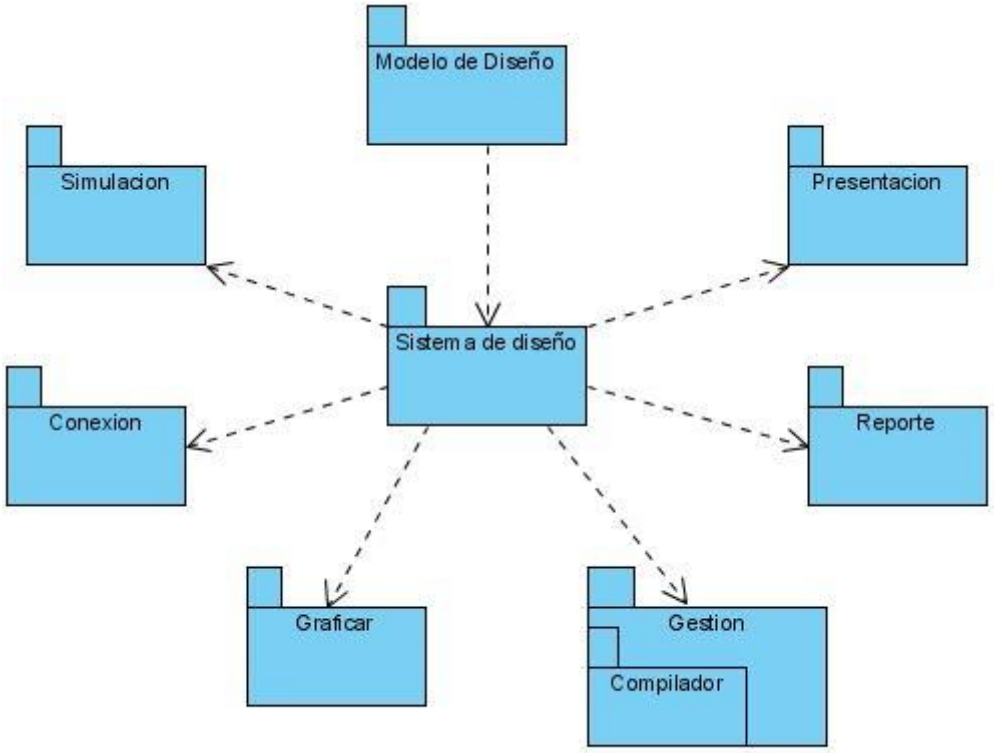

FIGURA 13

En el diseño del Nodo Virtual se definieron 6 subsistemas:

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.

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 tres clases, dos controladoras y una entidad. Su funcionamiento va a depender del subsistema Simulación debido a que la clase CC_Gestionar_Conexion_Proc hace uso de clases que se encuentran dentro de él.

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 seis clases, dos clases controladoras y cuatro clases entidades. La relación entre CC_Gestionar_Proceso y Gestion se debe a la necesidad de clases que existe en CC_Gestionar_Proceso para realizar 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.

Graficar: Este subsistema se encuentra en la capa de negocio. Su función es gestionar toda la representación gráfica de los procesos que están siendo simulados. Para ello se va a contar con una librería grafica donde se va a encontrar todos los iconos que se pueden utilizar para hacer la representación grafica de un modelo. Por cada icono se va a tener dos clases, una con la responsabilidad de llevar toda la lógica de la figura y otra con la responsabilidad de mostrar gráficamente los cambios que van a ir ocurriendo en el icono durante la simulación.

Reporte: En este subsistema se definirán las clases que le permitirán a la aplicación mostrar la información, tanto importante para los usuarios: tipos de procesos que se encuentran en el sistema y los procesos que se encuentran dentro de ellos, como importante para el administrador: 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.

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 y establecer límites necesarios para el buen funcionamiento del sistema. Va a estar integrado por siete clases, cuatro clases entidades y tres clases controladoras. Su funcionamiento depende de otros dos subsistemas, debido a que la clase CC_Gestionar_Limite necesita datos de las clases Proceso y Usuario, pertenecientes a Simulación y Conexión respectivamente.

Dentro de él 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 Fachada_Compilador encargada mediante la implementación del patrón fachada que solo se tenga acceso al subsistema mediante ella. También encontraremos las clases Scanner, Parser, Simbolo, Symtable y Token, cada una de ellas tiene una responsabilidad en específico para verificar que las cadenas string de las ecuaciones matemáticas son correctas.

Los más significativos para la arquitectura son: Presentación, Simulación, Conexión y Gestión.

Presentación. Diagrama de diseño.

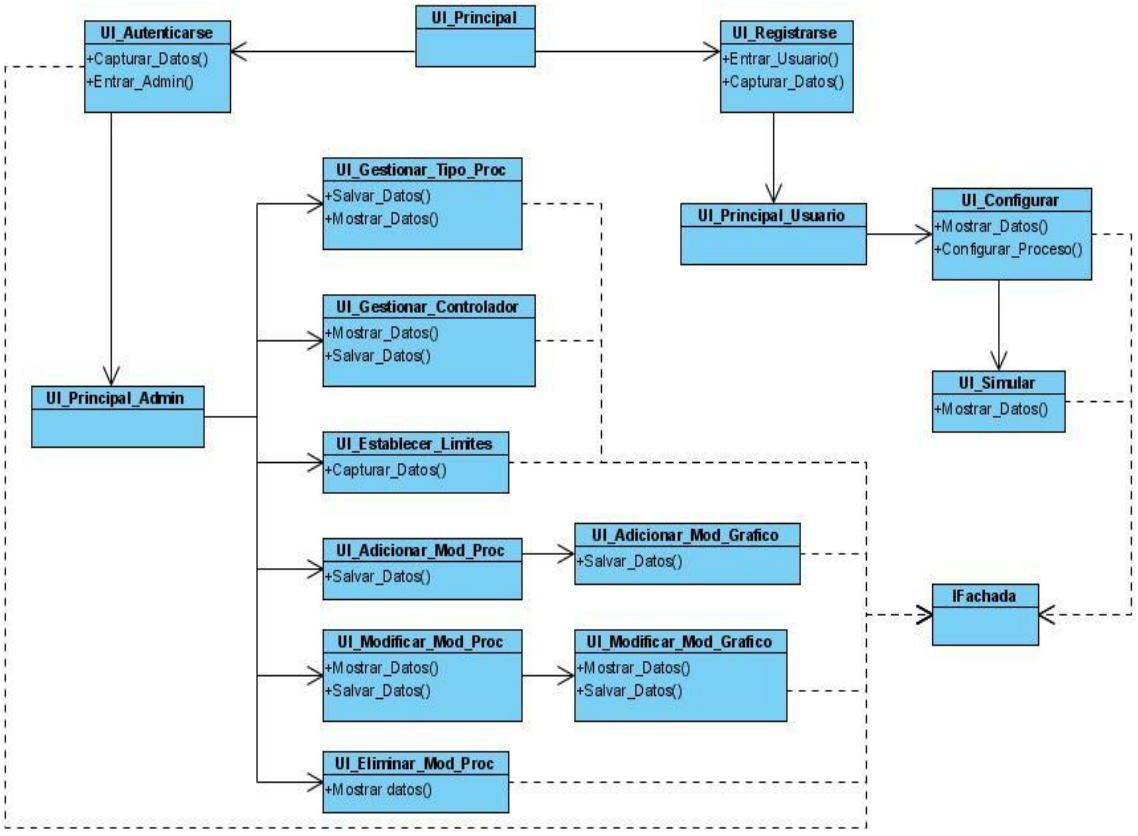

FIGURA 14

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

- 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.

Simulación. Diagrama de diseño.

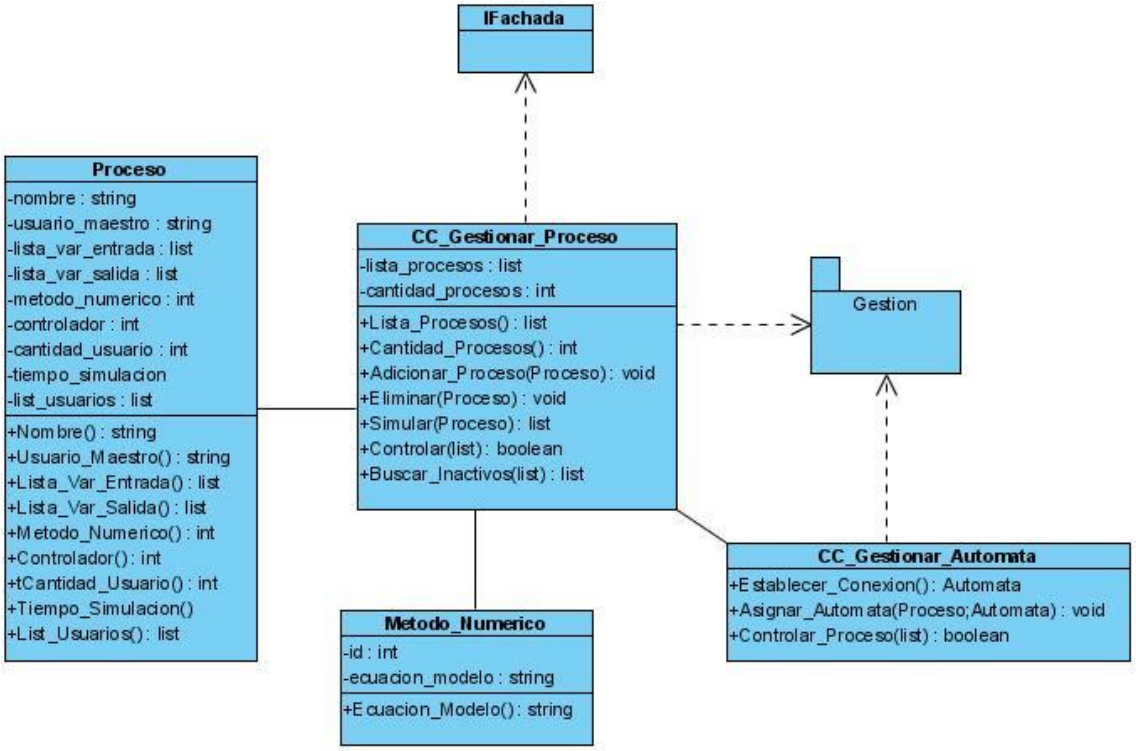

FIGURA 15

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.

- 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.

- CC_Gestionar_Automata: Se encarga de gestionar la conexión con los autómatas. Como atributo va a tener un listado con los autómatas del sistema. Entre sus funcionalidades se encuentran el asignar un autómata a un proceso que lo requiera verificando la disponibilidad de estos, y llevar el flujo de datos generados por el autómata en el control del mismo.

Diagrama de diseño. Conexión

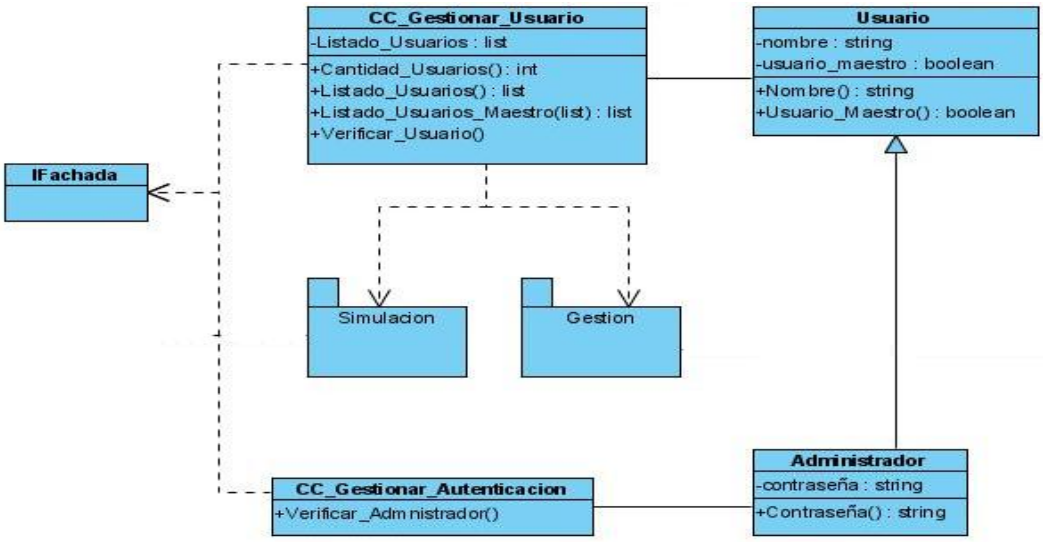

FIGURA 16

- 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.

Diagrama de diseño. Gestión

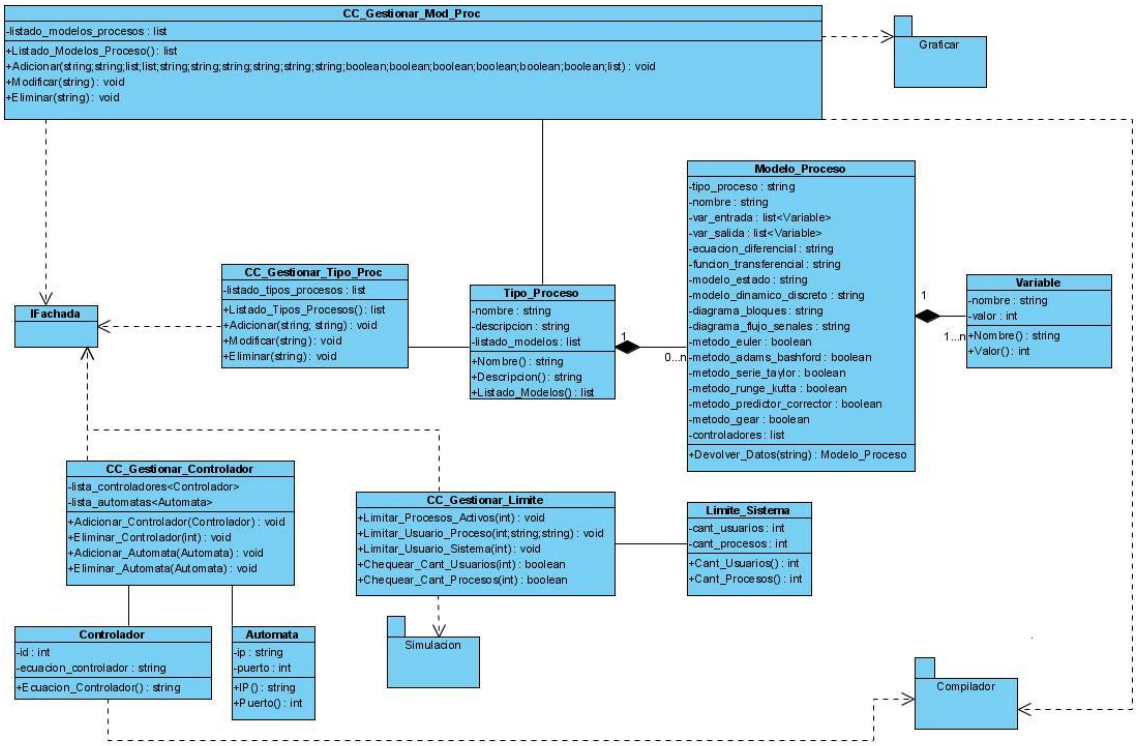

FIGURA 17

- 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 toda la información referente a los controladores.

- Controlador: Es la encargada de guardar los datos de los controladores

- Autómata: Es la encargada de guardar los datos de los autómatas que se encuentran en la red

2.11.4 Vista del modelo de implementación

Representa los artefactos arquitectónicamente importantes contenidos en el modelo de implementación. Estos artefactos son [Jacobson, Ibar; Booch, Grady; Rumbaugh, James. 200]:

- Componentes que siguen la traza de las clases de diseño significativas arquitectónicamente.
- Componentes ejecutables
- Componentes que son generales, centrales, que implementan mecanismos de diseño genéricos de los que dependen muchos otros componentes.

A continuación se presenta la vista general de implementación a través de las relaciones existentes entre los subsistemas de implementación.

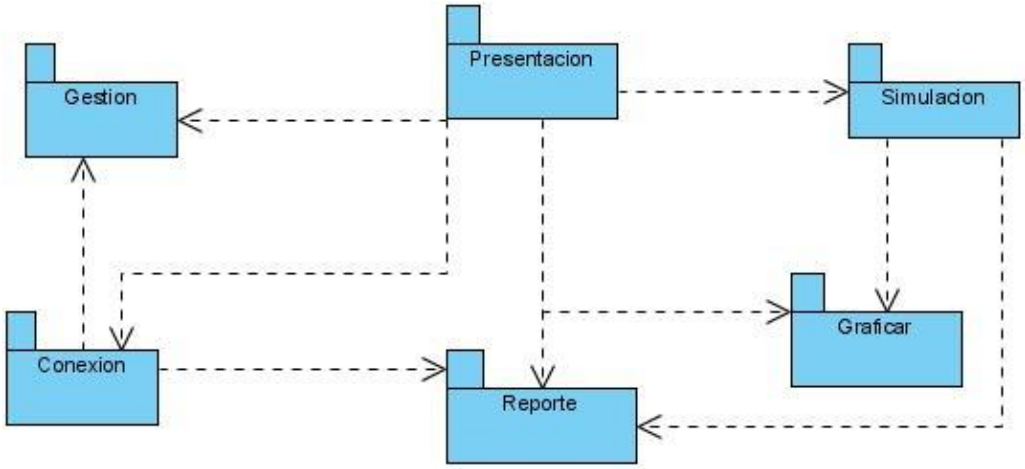

FIGURA 18

De los subsistemas definidos los más importantes para la arquitectura son Presentación, Simular, Conexión y Gestión.

Diagrama de implementación. Subsistema Presentación

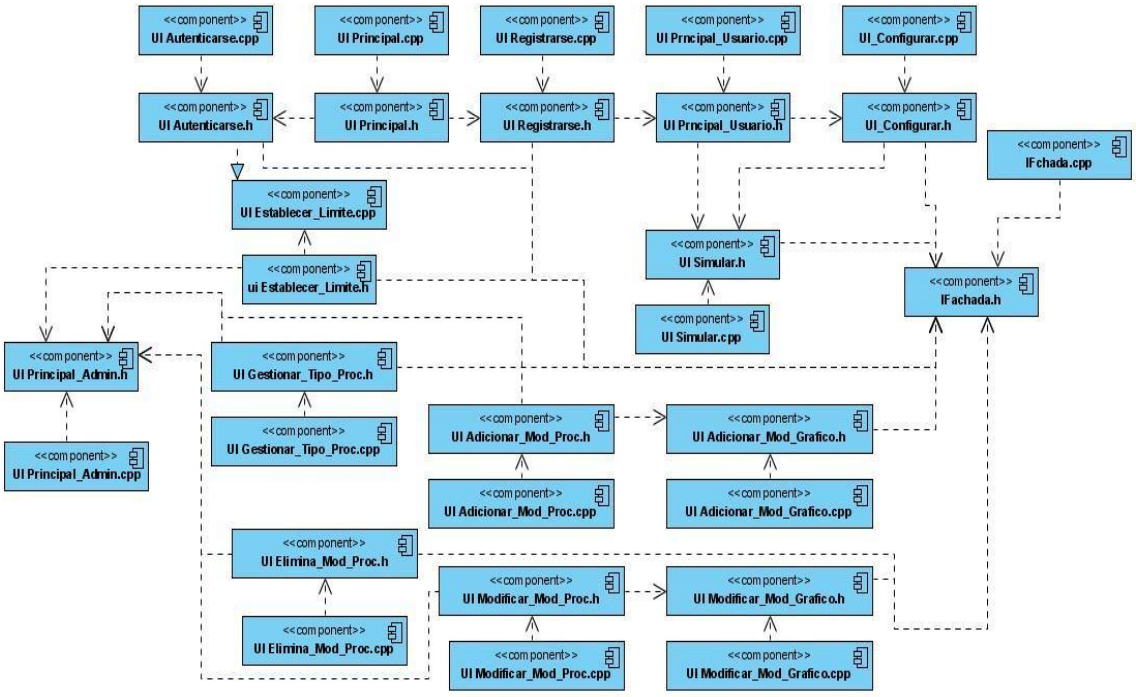

FIGURA 19

Diagrama de implementación. Subsistema Simulación

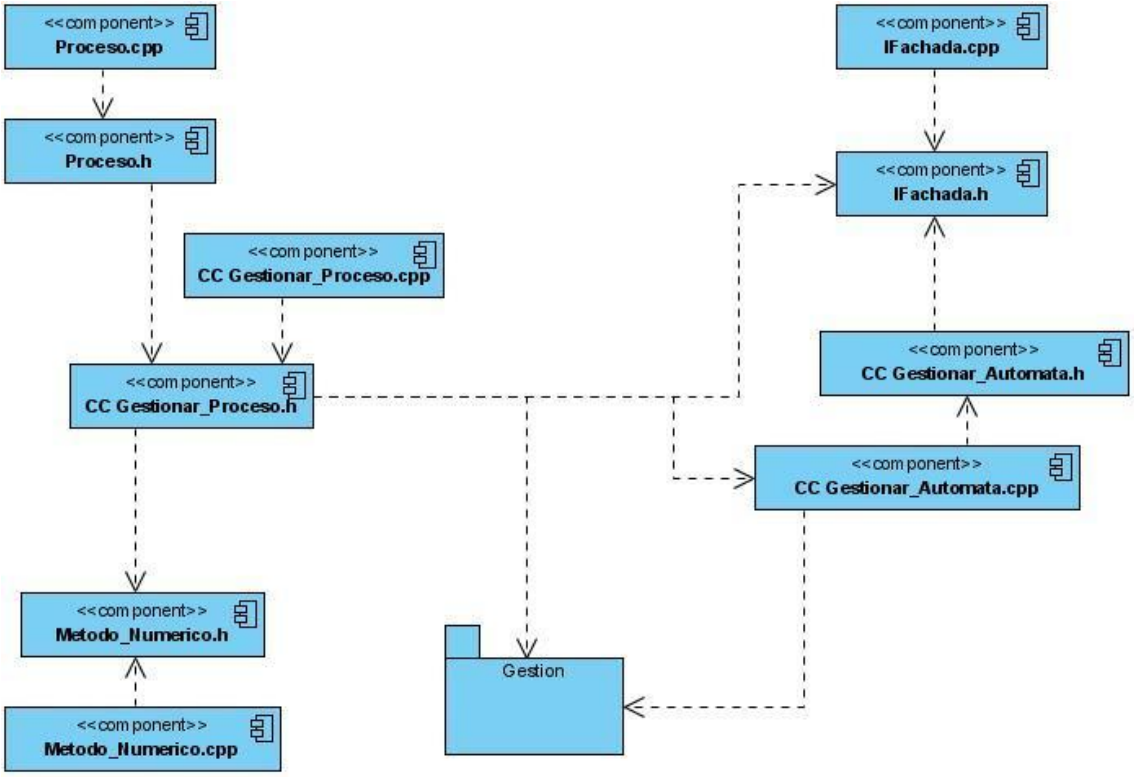

FIGURA 20

Diagrama de implementación. Subsistema Conexión

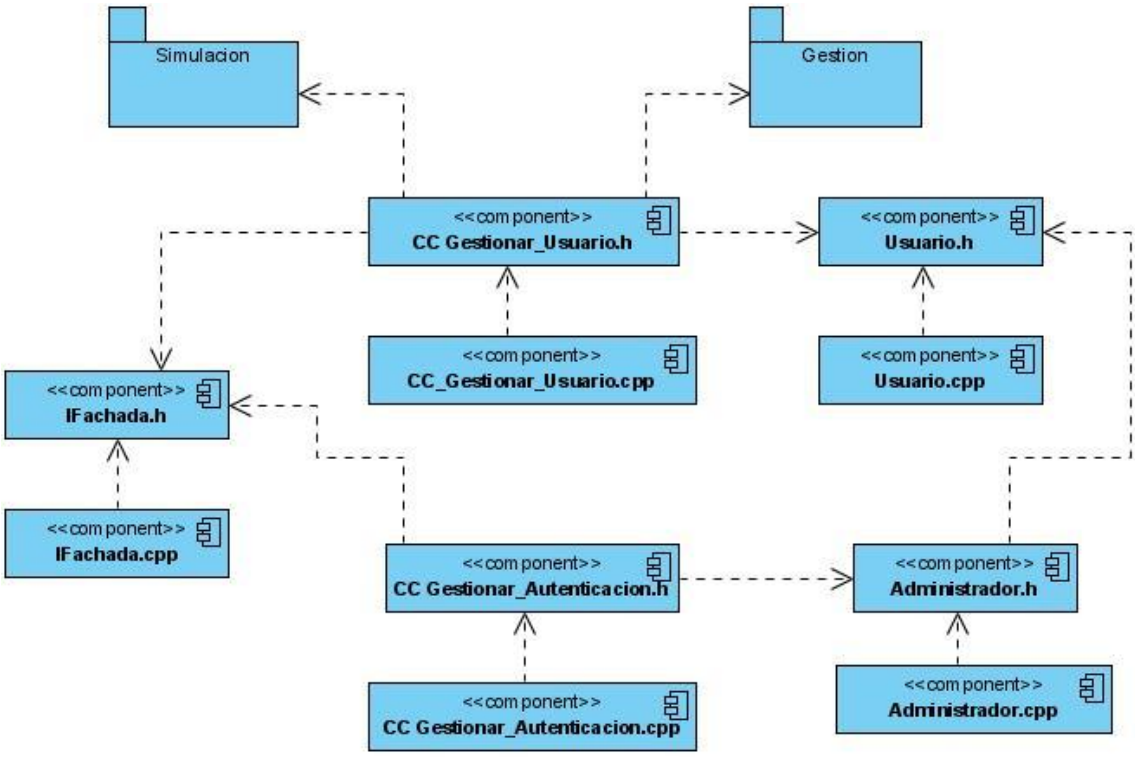

FIGURA 21

Diagrama de implementación. Subsistema Gestión

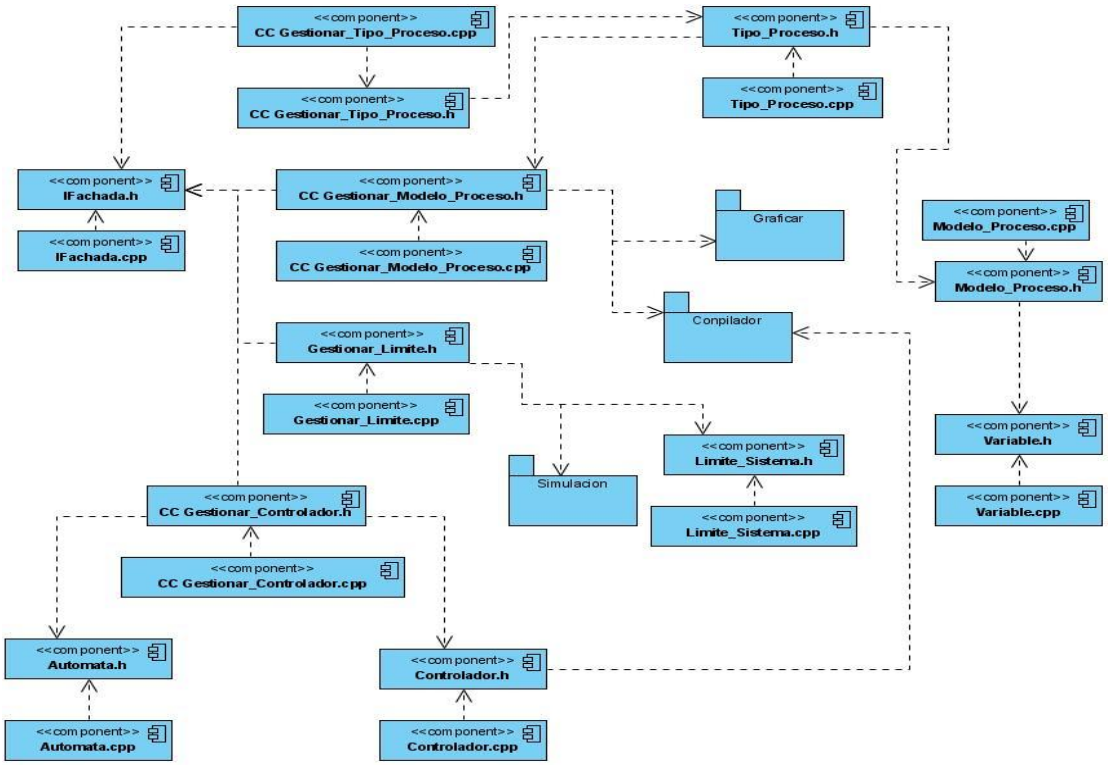

FIGURA 22

2.11.5 Vista del modelo de despliegue

Esta vista representa los artefactos del modelo de despliegue que son significativos para la arquitectura. En el libro "El Proceso Unificado de Desarrollo", Jacobson, Booch, Rumbaugh plantean que deberían mostrarse todos los aspectos del modelo de despliegue, pues todos son importantes.

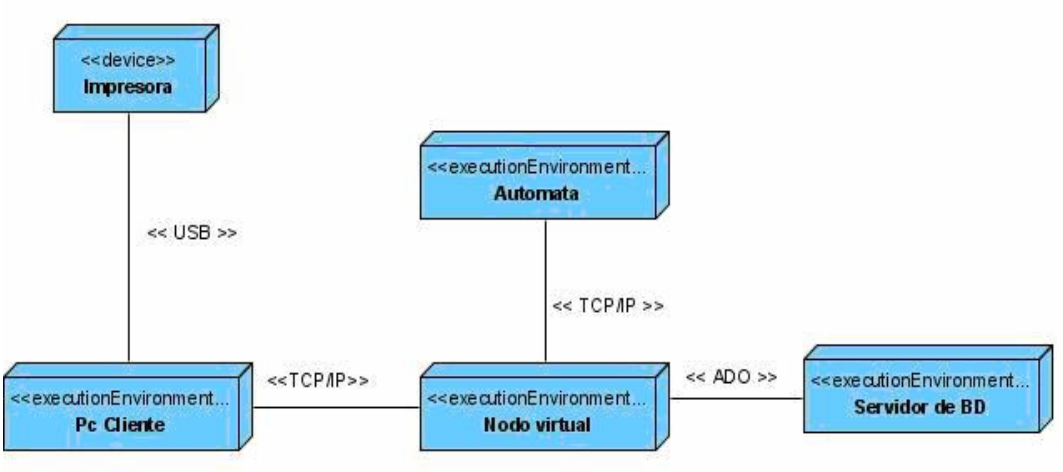

FIGURA 23

Descripción de los nodos

- PC cliente: Este nodo representa a todas las computadoras desde las que se conectaran los usuarios al Nodo virtual. Estas deben ser Pentium IV o superior con una RAM de al menos 256 Mb.
- Servidor de aplicaciones (Nodo virtual): Este nodo representa el servidor de aplicación donde se encuentra el ejecutable del Nodo virtual.
- Servidor de base de datos: En este nodo es donde correrá el servidor de base de datos. A este se conectara el Nodo virtual para obtener y salvar la información necesaria para las simulaciones.
- Autómata: Representa el autómata que se usara opcionalmente como controlador de la simulación. Este tendrá conexión con el servidor de aplicaciones pues de el recibirá los datos que entre el usuario para la simulación y a este enviara la alerta en caso de que el proceso simulado llegue a un punto critico.
- Impresora: Es definida para que el usuario, si lo desea, imprima los reportes que puede generar la aplicación.

Descripción de los protocolos

- TCP/IP: Comunica la pc cliente con el servidor de aplicaciones y a este ultimo con el autómata, permitiendo la transmisión segura de la información.
- ADO: Comunica al servidor de aplicación con el servidor de base de datos, permitiendo la transportación de los datos gestionados en el Nodo Virtual.
- USB: Permitirá la comunicación entre la pc cliente y la impresora. Por esta se envía la solicitud empaquetada de la información a imprimir

2.12 Validación de la solución propuesta

2.12.1 Aplicación de la métrica de Card y Glass

a) Complejidad estructural

$$S(i) = f_{out}^{2}(i)$$

Donde fout(i) es la expansión del modulo a analizar. La expansión de un modulo es la cantidad de módulos directamente subordinados a i.

b) Complejidad de datos

$$D(i) = \frac{v(i)}{[fout(i) + 1]}$$

Donde v(i) es la cantidad de variables de entrada y salida que entran y salen del modulo 1

c) Complejidad del sistema

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

A continuación se calculan las complejidades de los diferentes módulos.

Modulo Reporte

a) S(i) = 0 fout(i) = 0
b) D(i) = 2 v(i) = 2
c) C(i) = 2

Modulo Graficar

a) S(i) = 0 fout(i) = 0
b) D(i) = 23 v(i) = 23
c) C(i) = 23

Modulo Conexión

a) S(i) = 9 fout(i) = 3
b) D(i) = 0.75 v(i) = 3
c) C(i) = 9.75

Modulo Simulación

a) S(i) = 9 fout(i) = 3
b) D(i) = 12 v(i) = 48
c) C(i) = 21

Modulo Gestión

a) S(i) = 4 fout(i) = 2
b) D(i) = 36 v(i) = 48
c) C(i) = 40

Análisis de los resultados:

Para analizar los resultados obtenidos se tendrán en cuenta los niveles de complejidad según los intervalos de valores siguientes de C(i).

	No complejo	Complejo	Muy complejo
C (i)	x< 32	32 <= x < 50	X >= 50

Según los valores obtenidos se puede constatar que el módulo de mayor complejidad estructural es el modulo de Gestión. El resto de los módulos no son considerablemente complejos.

2.12.2 Métrica de Hery y Kafura

Esta métrica plantea el calculo de la complejidad a través de la formula

$$MHK = longitud(i) + [fin\ (i) + fout(i)]^2$$

Donde longitud(i) es la cantidad estimada de sentencias para el modulo, fin(i) es la concentración del modulo, la concentración no es mas que la cantidad de módulos de los que i recoge datos.

Modulo Reporte
longitud(i) = 257
fin(i) = 3
fout(i) = 0
MHK = 266

Modulo Graficar
longitud(i) = 676
fin(i) = 2
fout(i) = 0
MHK = 680

Modulo Conexión
longitud(i)= 318
fin(i) = 0
fout(i) = 3
MHK = 327

Modulo Simulación
longitud(i) = 548

fin(i) = 1
fout(i) = 3
MHK = 564

Modulo Gestion
longitud(i) = 1387
fin(i) = 2
fout(i) = 2
MHK = 1403

Según los resultado obtenidos de la aplicación de esta métrica se puede observar que el modulo mas complejo es el modulo de gestión.

2.12.3 Evaluación de los atributos de calidad

Los atributos de calidad están enfocados al producto final, a las características externas del producto cuando esta en operación pero indiscutiblemente la arquitectura tiene una gran influencia en el cumplimiento de estos atributos. El análisis en este epígrafe estará dirigido a verificar si la arquitectura propuesta cumple o no con estas características.

Funcionabilidad

En primer lugar la arquitectura tiene que ser funcional, garantizar que el sistema cumpla con todos los requisitos del usuario. El desarrollo de la arquitectura a partir de la selección de casos de usos arquitectónicamente significativos posibilita que esta se desarrolle dirigida por esos casos de usos, es decir que se piense con lo que quiere el usuario en mente. Una vez que se tiene la arquitectura inicial, en las siguientes iteraciones va creciendo esta arquitectura con la implementación de nuevos casos de usos, garantizando que todo el tiempo se trabaje para cumplir con los requerimientos.

La adecuación se define como la capacidad del producto de software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados. En la arquitectura propuesta se dieron solución a las propuestas hecha por los usuarios y se tuvieron en cuenta todos los requisitos funcionales y no funcionales que tendría el software. La arquitectura desarrollada presenta una buena adecuación puesto que en ella esta soportado todo lo necesario para desarrollar simuladores con características semejantes.

Usabilidad

Muchas veces se piensa que la usabilidad es un atributo al que se le da solución en la capa de presentación y no tiene más influencia en el resto de la arquitectura. Aunque la mayoría de los elementos relacionados con la facilidad de uso del programa están encapsulados en la capa de presentación existen otros más generales que son tratados a otros niveles. Otro aspecto importante esta relacionado con darle información al usuario de la naturaleza de los errores que pueda presentar el programa, con este fin se definió una estrategia de tratamiento de errores que se integran a la ayuda del programa, cuando ocurre un error el usuario es notificado y se le da posibilidad de que consulte la ayuda relacionada con ese error.

Portabilidad

La portabilidad es un aspecto importante del software, y las decisiones arquitectónicas tienen una gran influencia en esta característica. Por ejemplo, la arquitectura en capas da posibilidad de que se pueda hacer con mayor facilidad una migración a Linux, el hecho de separar la interfaz de usuario de la capa de negocio hace posible que en caso necesario se sustituya completamente la capa de presentación sin que las capas inferiores sufran cambios.

Mantenebilidad

Este atributo constituye un buen medidor de la robustez y flexibilidad del producto, y son fundamentales las decisiones arquitectónicas que se tomen. La arquitectura que se analiza en este trabajo tiene como característica fundamental su extensibilidad. En la actualidad la persistencia se basa en el acceso a la base de datos, si en un futuro se quiere usar serializacion, y gracias la patrón que se aplica, solo hay que gestionar el acceso a el. Además, la aplicación del estilo en capas puro hace que sea más fácil el proceso de adicionar un nuevo modulo o funcionalidad.

Conclusiones

En este capitulo se trato todo lo referente al diseño arquitectónico del Nodo Virtual de procesos: se realizo de la descripción de la arquitectura mediante las diferentes vistas arquitectónicas de los modelos del sistema y se justificaron los diferentes patrones y estilos utilizados. Se pudo constatar, ademas la complejidad del

sistema a través de la aplicación de las métricas. Como resultado de las misma se llego a la conclusión de que el modulo de mayor complejidad es Gestión, por lo que se necesitará mayor esfuerzo por parte de los programadores para poder desarrollarlo. Además se comprobó que la arquitectura diseñada cumple con los atributos de calidad.

Conclusiones

Al terminar la realización de este trabajo se arribaron a las siguientes conclusiones:

- La descripción de la arquitectura mediante las 4 + 1 vistas definidas por RUP garantizaron que se tuviera una visión de todos los modelos del sistema y de aquellos elementos arquitectónicamente significativos durante el Proceso de Desarrollo.

- El estilo arquitectónico en Capas permitió que el sistema se estructurara de forma tal que permitiera la descomposición de la complejidad estructural y que el mantenimiento del sistema no sea complejo.

- Al ser diseñada y desarrollada en componentes separados, la arquitectura permite la agregación, eliminación e implementación de componentes sin que la arquitectura tenga que cambiar mucho su estructura, lo que denota su flexibilidad.

- Con el trabajo desarrollado se generaron los artefactos necesarios que contemplan y permiten la implementación del sistema a desarrollar.

Recomendaciones

- Refinar la propuesta de arquitectura durante todo el proceso de desarrollo del software.
- Realizar la validación de la arquitectura a través de la aplicación de otras métricas para la identificación de posibles debilidades.

Bibliografía

1. **Alden Szyperski, Clemens**. "*Component software: Beyond Object-Oriented programming."* 2002.

2. **AmericaTI EIRL**. "*Ventajas y desventajas: Comparación de los lenguajes C, C++ y Java"*. 2006. Disponible en: www.americati.com

3. **Arsene , Corneliu. "***Modelling and simulation of water system based on loop equations."* Disponible en: http://ducati.doc.ntu.ac.uk/uksim/journal/Vol-5/No-1&2/ARSENE.pdf

4. **Bass, Len, Clements, Paul y Kazman, Rick. "***Software Architecture in Practice".* Segunda edición. s.l.: Addison Wesley, 2003. 0-321-15495-9.

5. **Bernadi Millan, Xavier.** *"Que son los triggers y como usarlos en MySQL 5.0?".* Disponible en: http://www.webtaller.com/construccion/lenguajes/mysql/lecciones/que-son-los-triggers-como-usarlos-mysql-5.0.php

6. **Billy Reynoso, Carlos. "***Arquitectura de Software: Introducción a la Arquitectura de Software"*. Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.asp

7. **Billy Reynoso, Carlos; Kiccillof, *Nicolás***. *"Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* **"** [Documento PDF] Buenos Aires: s.n., 2004.

8. **Bosque, Iñigo**. *"Implementación de patrones de diseño con .Net."* Disponible en: http://download.microsoft.com/download/7/9/9/79981b45-cdb0-4958-9c0a-6db02ddbb976/I_Bosque_Msdn_SesionPatrones_es-es.pps

9. **Brown, Alan; Wallnau, Kurt**. "*The current state of CBSE*". IEEE Software, pp.37-46, Octubre de 1998.

10. **Burbeck, Steve**. *"Application programming in Smalltalk-80: How to use Model-View-Controller (MVC)"*. University of Illinois in Urbana-Champaign, Smalltalk Archive. Disponible en: http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html.

11. **Canal, Carlos**. "*Arquitectura, marcos de trabajo y patrones*." Marzo, 2005

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

13. **Capote Abreu, Jorge; Alvear Portilla, Daniel; Abreu, Orlando; Urrutia, Mariano Lázaro; Espinas Santos, Pablo.** *"Algunos conceptos y Definiciones del Modelado y Simulación Computacional de Incendios"*. Grupo GIDAI. Universidad de Cantabria. Marzo, 2006. Disponible en: http://www.duiops.net/manuales/faqinternet/faqinternet4.htm]. www.enpresadigitala.net

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

15. **Dueñas, Francisco Armando.** *"Ciclo de vida del SO Linux".* Disponible en: \\ucistore\Software\Documentation\Linux\Docs

16. **Eclipse.** *"Eclipse, C/C++ Development Tooling - CDT".* Disponible en: http://www.eclipse.org/cdt/

17. **García de la Puente, Sergio Jesús; Plasencia Crespo, Mileisys**. "*Sistema para la descarga y procesamiento automatizado de patentes: Predictor. Rol de arquitecto*". 2007

18. **Garlan, David; Shaw, Mary**. "*An introduction to software architecture*". CMU Software Engineering Institute Technical Report, CMU/SEI-94-TR-21, ESC-TR-94-21, 1994.

19. *Glosario de e-learning.* Disponible en: http://pulsar.ehu.es/pulsar/glosario/glosario_R

20. **Hosseinzaman, A.; Bargiela, A.** "*ADA's Virtual Node based Water System Simulator*". Department of Computing-Nottingham Trent University. 1992. Disponible en: http://citeseer.ist.psu.edu/rd/3162609%2C104974%2C1%2C0.25%2CDownload/http://citeseer.ist.psu.edu/cache/papers/cs/571/http:zSzzSzwww.doc.ntu.ac.ukzSzRTTSzSzPaperszSzrttg-publ10.pdf/ada-s-virtual-node.pdf

21. **Jacobson, Ibar; Booch, Grady; Rumbaugh, James.** *"El Proceso Unificado de Desarrollo".* 2000.

22. **Lago, Ramiro**. *"Patrón: Data Access Ojbect*". 2007. Disponible en: http://www.proactiva-calidad.com/java/**patrones**/DAO.html

23. **Larman, Craig**. *"UML y patrones: introducción al análisis y diseño orientado a objetos".* Mexico.1999.

24. **Microsoft Patterns & Practices.** *"Layered application. Version 1.0.1.",* 2004. Disponible en: http://msdn.microsoft.com/architecture/patterns/default.aspx?pull=/library/en-us/dnpatterns/html/ArcLayeredApplication.asp

25. **López, Ángel**. *"Patrones de arquitectura y diseño en aplicaciones .Net".* 2005. Disponible en: www.ajlopez.com/downloads/**Patron**esNet/ArqPuntoNet200501.ppt

26. **Lucas Rodríguez, Fernando.**"*Simulación de procesos de información y criptografía cuántica"*. 2005. Departamento Física Aplicada I, Sevilla. Disponible en: http://www.fernandolucas.es/data/qcs/memoria.pdf

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

28. **Miranda, Javier; Guerra, Francisco; Hernández, Luis.** *"Fundamentos de programación con ADA".* 2002

29. **Molino, Neil; Bao, Zhaosheng; Fedkiw, Ron.** *"Virtual Node Algorithm for Changing Mesh Topology During Simulation"*. Stanford University. 2004. Disponible en: http://physbam.stanford.edu/~fedkiw/papers/stanford2004-01.pdf

30. **Mollineda, Ramon.** "*Arquitectura de software: arte y oficio"*. Revista del Instituto Tecnológico de Informática. España. 2005

31. **Perera Morales, José Raúl.** *"Arquitectura de software para un sistema Gestión de Inventarios".* 2007.

32. **Pingarron, Raúl.** *"Arquitectura TCP/IP."* Disponible en: http://web.frm.utn.edu.ar/comunicaciones/tcp_ip.html

33. **Pressman, Roger S. "***Ingeniería del Software: Un enfoque práctico"* Parte I La Habana: Félix Varela, 2005.

34. **Real Academia Española. "***Diccionario de la Lengua Española*." Vigésima segunda edición. 24 de junio de 2004. Disponible en: http://buscon.rae.es/draeI/

35. **Rodríguez, Minerva; Arteaga, Arantza.** *"¿Qué es un autómata programable?" 2007.* Disponible en: http://www.pangea.org/iesoa/sp/article.php?id_article=536

36. **Salazar, Leonel; Pérez-Alejo Raúl**. **"***Módulo de Visualización de Gráficos Vectoriales para aplicaciones SCADA"*. 2006

37. **Serrano, Juan Manuel.** *"Arquitecturas software y familias de productos."* 2007. Disponible en: http://triana.escet.urjc.es/aspf/Tema4-1.pdf

38. **Shaw, Mary; Garlan, David**. "*Software Architecture: Perspectives on an emerging discipline"*. 1996.

39. *"Sistema Operativo Linux."* Disponible en: \\ucistore\Software\Documentation\Linux\Docs

40. ***SIG*-UPV.** *"Aplicación de motores de simulación interactivos continuos a la construccion de entornos virtuales corporativos e inteligentes."Disponible en: http://www.sig.upv.es/proyectos/simulacion/proyectos.html*

41. **SM Data, S.A.** *"Tecnología RAID". 2008* Disponible en: http://www.smdata.com/TECNRAID.htm

42. **startMedia**. "*Autómatas"* Disponible en: *http://html.rincondelvago.com/automata.html*

43. **Suarez Lopez, Juan Carlos; Corrales Valdes, Yurisnel**. *"Desarrollo de la arquitectura del Sistema para la Simulación de Procesos en la Industria Química Cubana."* 2007.

44. **Tucker, Jr**. *"Lenguajes de Programación".* 2da ed. McGraw-Hill México. 1992.

45. **UNED.** *".NET y J2EE valoración y comparación de los elementos de las dos plataformas".* Programa de enseñanzas no regladas de la Universidad Nacional de Educación a distancia UNED, España. 2007.

46. **University of Stuttgart**. Disponible en: http://net.informatik.uni-stuttgart.de/

47. *"Ventajas y desventajas: Comparación de los lenguajes C, C++ y Java."* Disponible en: http://americati.com/doc/ventajas_c.pdf

48. **Welicki, L**. *"Patrones y Antipatrones: una Introducción"*. 2007

49. **Wikipedia.** *"Entorno de desarrollo integrado". Disponible en: http://es.wikipedia.org/wiki/Ambiente_integrado_de_desarrollo*

50. **Wu Y**. "*A virtual node method for treatment of wells in modeling multiphase Flow in reservoirs".* 1999

Glosario de términos

ADA: Ada es un lenguaje de programación. El nombre fue puesto en recuerdo de Augusta Ada Byron, considerada la primera programadora de la historia. Fue desarrollado porque el departamento de defensa de Estados Unidos descubrió que ninguno de los lenguajes existentes era apropiado para el control de tiempo real de sistemas empotrados grandes. Fue diseñado por Jean Ichbiah de la compañía francesa Honeywell Bull. Entre sus principales características están que es estructurado, fuertemente tipado de forma estática, multipropósito, orientado a objetos y concurrente.

ALGOL 68: ALGOL es el acrónimo de **Algo**rithmic **L**anguage. Es un lenguaje de programación que aunque no tuvo mucha difusión comercial influyo en otros como Pascal, C y Ada. La versión conocida como ALGOL 68 fue presentada en el año 1965, pero su definición definitiva fue terminada en el año 1968

Datagrama: Unidad de transferencia que el protocolo IP utiliza, algunas veces identificada en forma más específica como datagrama Internet o datagrama IP.

FORTRAN: El FORTRAN (*FORmula TRANslator)* es un lenguaje de programación muy potente. Es el primer lenguaje científico, siendo desarrollado a últimos de la década de los 50, pero es aún ampliamente utilizado en aplicaciones científicas y de ingeniería. La inclusión en el lenguaje de la aritmética de números complejos lo ha convertido en una fuerte herramienta para el desarrollo de aplicaciones que realicen cálculos complejos.

Módulos: Parte de un programa, o sea, un aplicación debe realizar varias tareas, un modulo realizara una de estas tareas.

Plugins: Software que interactúa con otra aplicación para aportarle una función especifica**.**

RAID: Es acrónimo de Redundant Array of Independent Disks, la cual significa matriz redundante de discos independientes. RAID es un método de combinación de varios discos duros para formar una única unidad lógica en la que se almacenan los datos de forma redundante. Ofrece mayor tolerancia a fallos y más altos niveles de rendimiento

que un sólo disco duro o un grupo de discos duros independientes. [SM Data, S.A. 2008]

SCSI: Acrónimo inglés ***S****mall* ***C****omputer* ***S****ystem* ***I****nterface*, es una interfaz para la transferencia de datos entre distintos dispositivos del bus de la computadora. Se pronuncia *escousi.*

SIMULA 67: Simula es el primer lenguaje de programación orientado a objetos. Su versión SIMULA 67 fue lanzada en mayo del 1967 por Ole Johan Dahl y Kristen Nygaard, sus autores.

SOCKET: Es un método para la comunicación entre un programa del cliente y un programa del servidor en una red. Un socket se define como el punto final en una conexión. También una dirección de Internet, combinando una dirección IP y un número de puerto.

Subsistema de diseño: Forma de organizar los artefactos del modelo de diseño.

Tecnología multicast: Transmisión simultanea de paquetes en la red.

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

Anexo 1

Descripción del caso de uso Conectar_Sistema

Caso de Uso – 1	Conectar_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

Anexo 2
Descripción del caso de uso Conectar_Proceso

Caso de Uso – 2	Conectar_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.

Anexo 3
Descripción del caso de uso 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

Anexo 4
Descripción del caso de uso 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

Anexo 5
Descripción del caso de uso 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 proceso a modificar. 3. El Administrador realiza los cambios necesarios a los datos	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 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	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

eliminar el Modelo de Proceso.	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

Anexo 6
Descripción del caso de uso 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

Anexo 7
Descripción del caso de uso 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. 2. El Usuario Maestro confirma si quiere comenzar la simulación o no. 3. El Usuario Maestro puede o no cambiar los valores de las variables de entradas.	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. 2.1. 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. 3.1. 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

4. El Usuario Maestro selecciona la opción de terminar la simulación.	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

Anexo 8
Descripción del caso de uso Probar_Aplicaciones.

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. 2. El Usuario Maestro confirma si quiere comenzar a probar la aplicación o no.	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. 2.1. 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

Anexo 9
Descripción del caso de uso Gestionar _Controlador.

Caso de Uso – 13	Gestionar _Controlador
Actores	Administrador
Propósito	Permite al administrador adicionar los controladores y los autómatas conectados al sistema.
Resumen	El CUS se inicia cuando el administrador de la aplicación selecciona la opción de Gestionar Controlador, luego selecciona el tipo de controlador, 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 autómata adicionada 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 Adicionar Automata.
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: Adicionar autómata	
1. El Administrador selecciona la opción de Adicionar Autómata. 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 autómata. 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.
Referencia:	R12
Prioridad:	Crítico