Arquitectura de un nodo virtual de procesos

Virtual node process architecture

Susana Gonce Fernández, Mailen Escobar Pompa, Leydis Ortiz Azahares

Universidad de las Ciencias Infomáticas

Autor para la correspondencia:sgonce@uci.cu

Resumen

Un nodo virtual es una aplicación que permite correr diferentes instancias de un mismo software en un único nodo (nodo físico) y cada instancia del software trabaja en un entorno de ejecución independiente (nodo virtual). Partiendo de este concepto se establece el significado de nodo virtual de procesos: software que permite implementar modelos de distintos procesos tanto para su simulación como para la prueba de aplicaciones en tiempo real. El objetivo de este trabajo es el diseño de una arquitectura robusta que permita la implementación de un nodo virtual para la simulación concurrente de procesos industriales en la especialidad de Ingeniería Automática. Después de realizar un estudio de las tecnologías disponibles y teniendo en cuenta los requisitos planteados por el cliente se obtuvo como resultado la definición de los patrones de arquitectura, el estilo arquitectónico, el lenguaje de programación y el Entorno de Desarrollo Integrado (IDE) por sus siglas en ingles, a utilizar en el desarrollo del sistema. La arquitectura definida permitirá que el sistema sea flexible, de fácil mantenimiento y adaptable

Palabras clave: Arquitectura, nodo virtual, simulación de procesos.

Abstract

A virtual node is an application that makes it different instances of the same software on a single node (physical node), and each instance of the software works in a separate execution environment (virtual node). Building on this concept provides the meaning of virtual node process: software that allows to implement models of different processes for its simulation to test applications in real time. The objective of this work is to design a robust architecture that enables the implementation of a virtual node for the concurrent simulation of industrial processes in the specialty of Automatic Control Engineering. After conducting a survey of available technology and taking into account the requirements raised by the client was gained as a result the definition of the patterns of architecture, architectural style, the programming language and Integrated Development Environment (IDE) for their acronyms in English, to use in the development of the system. The defined architecture will allow the system to be flexible, adaptable and easy to maintain

Key words: *Architecture, simulation process, virtual node*

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 será 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.

Lo que se resume en que 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.

Desarrollo

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.

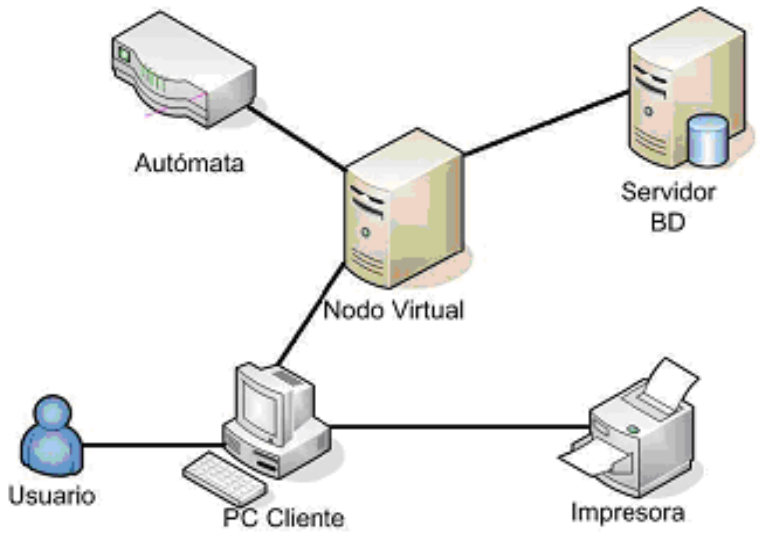

Fig 1. Propuesta del sistema

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]

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.

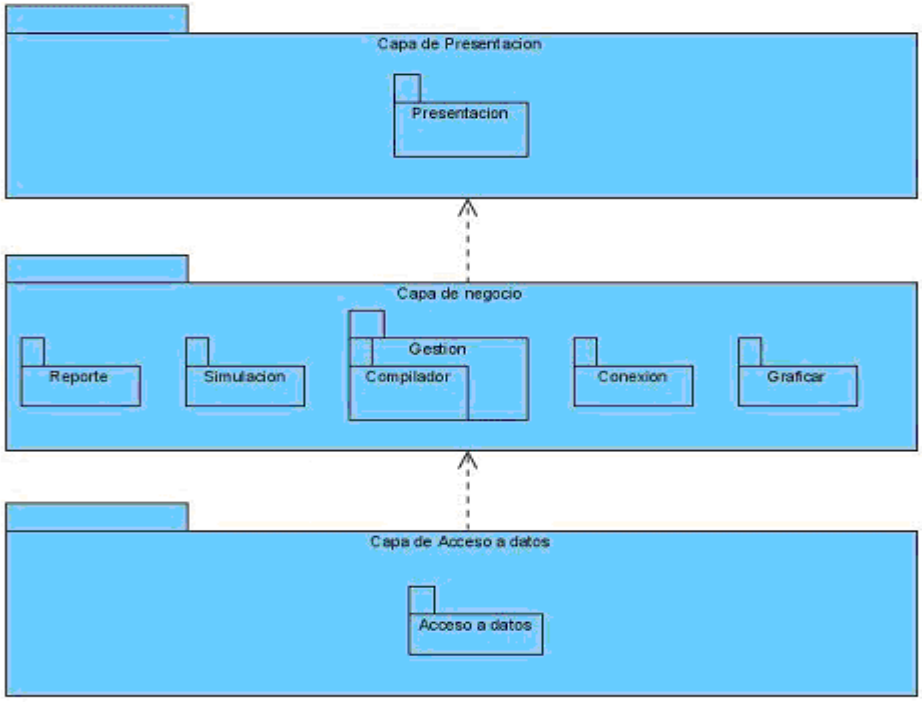

Fig 2. Estilo en capas.

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 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.

- 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 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. 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.

Descripción de la arquitectura

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.

En este artículo solo mostraremos las vistas que por su características muestran de manera general como se desarrollara el sistema.

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. 2000]

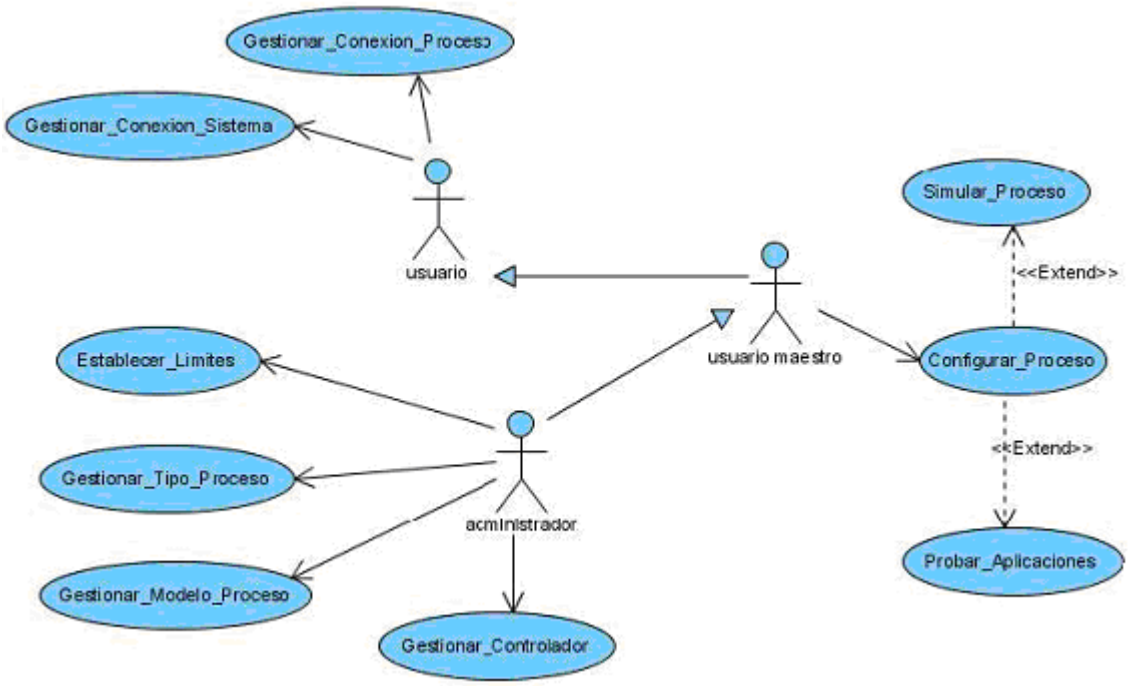

Fig 3. Vista de casos de uso.

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. 2000]:

- 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

Módulo Simulación

Caso de uso Simular.

Fig 4. Caso de uso Simular

Caso de uso Configurar

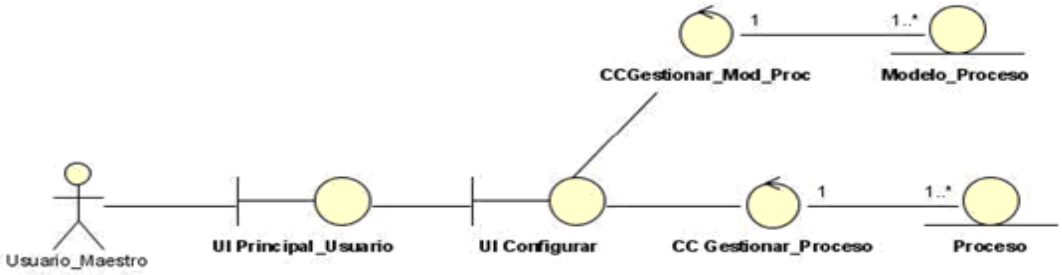

Fig 5. Caso de uso Configurar

Caso de uso Probar_Apliaciones

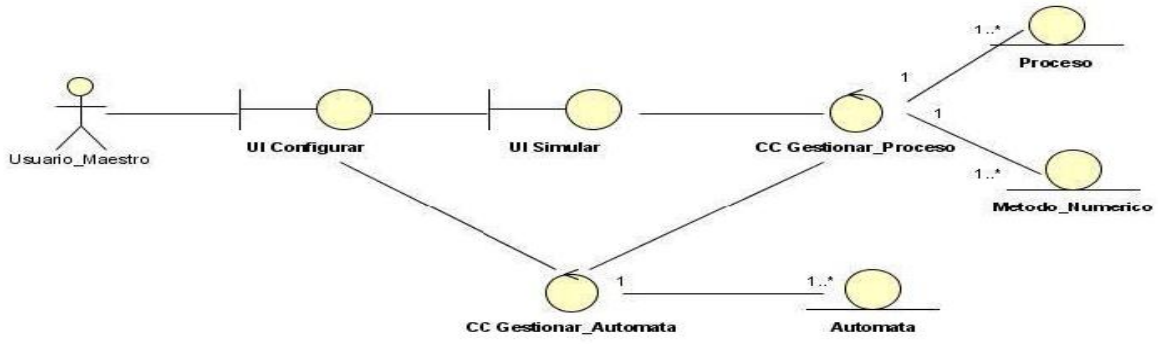

Fig 6. Caso de uso Probar _ aplicaciones

. Módulo Conexión

Caso de uso Conexión_Sistema

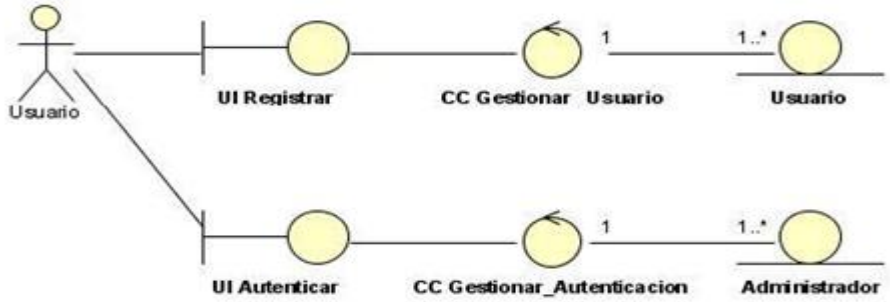

Fig 7. Caso de uso Conexión _ sistemas

Caso de uso Conexión_Proceso.

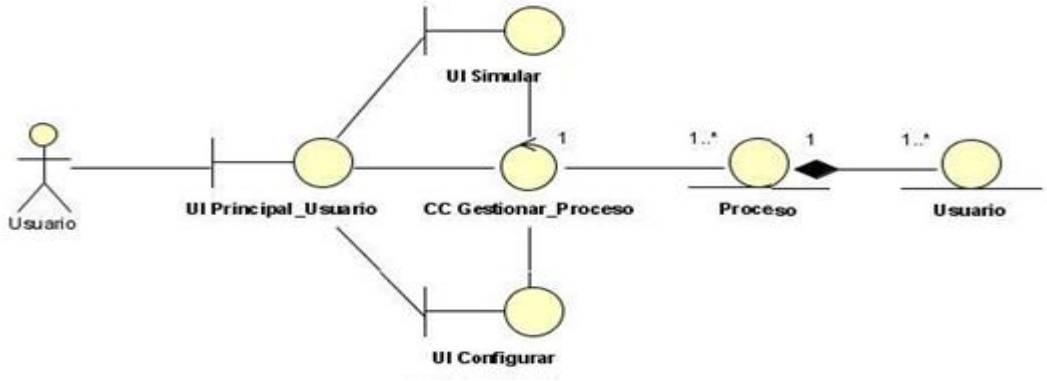

Fig 8. Caso de uso Conexión _ procesos

Módulo Gestión

Caso de uso Establecer_Limite

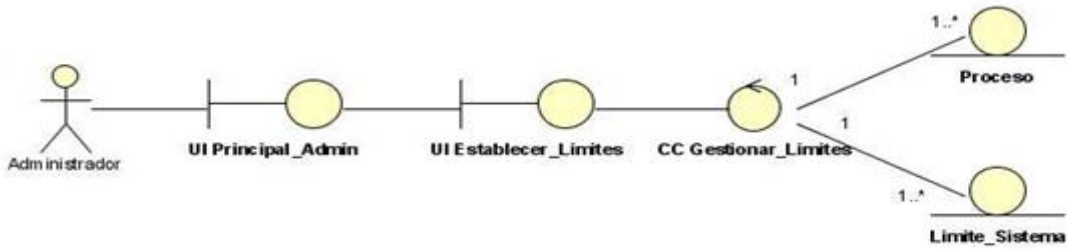

Fig 9. Caso de uso Establecer _ limite

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.

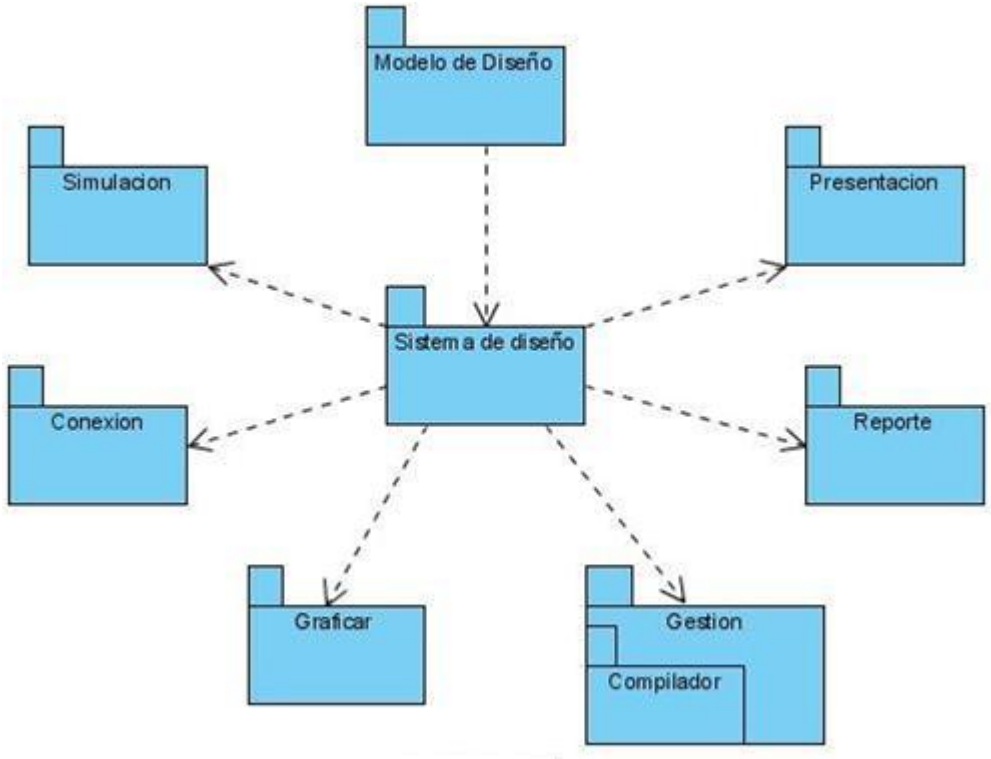

Fig 10. Sistema de diseño

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

Subsistema Presentación. Diagrama de diseño.

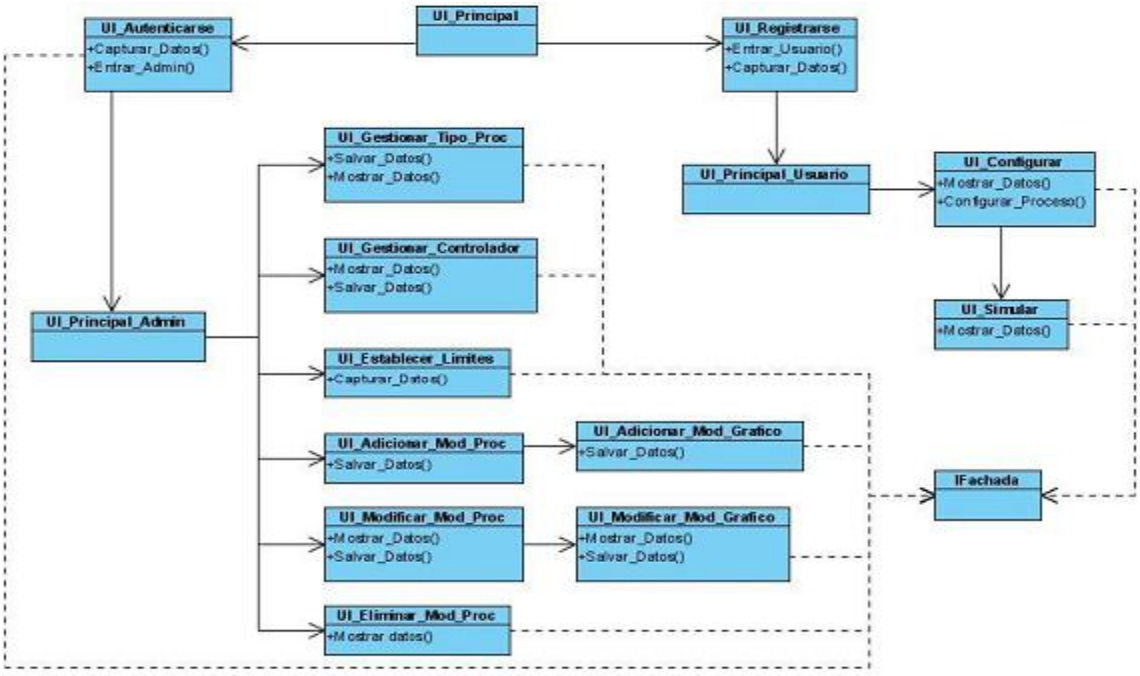

Fig 11. Presentación. Diagrama de diseño.

Subsistema Simulación. Diagrama de diseño.

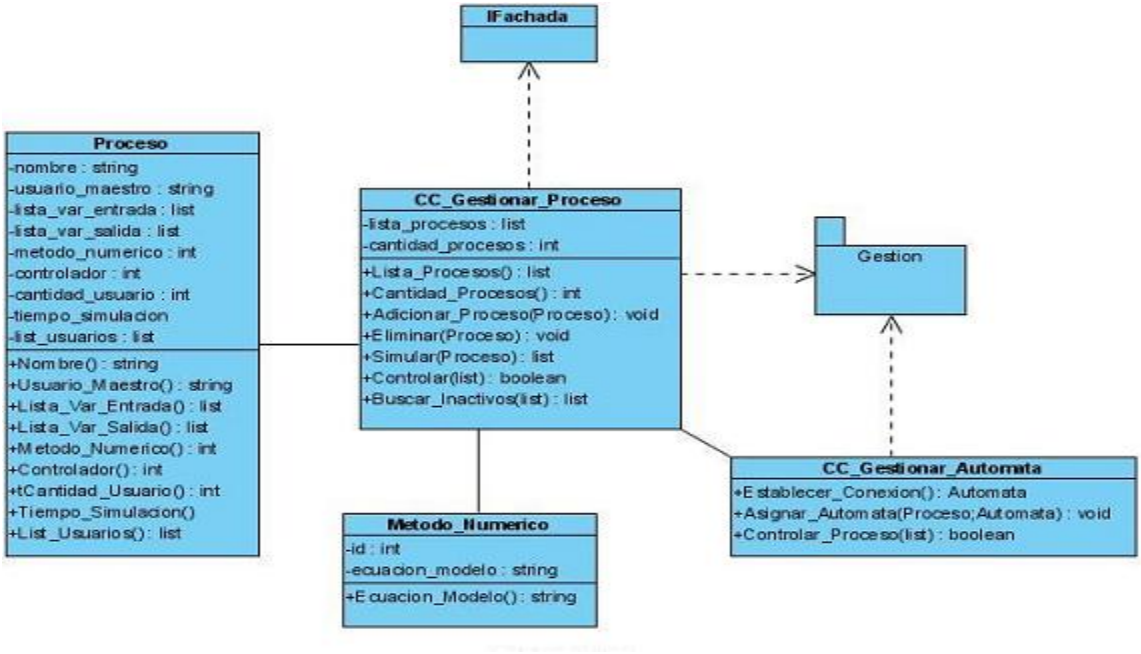

Fig 11. Simulación. Diagrama de diseño.

Subsistema Conexión. Diagrama de diseño.

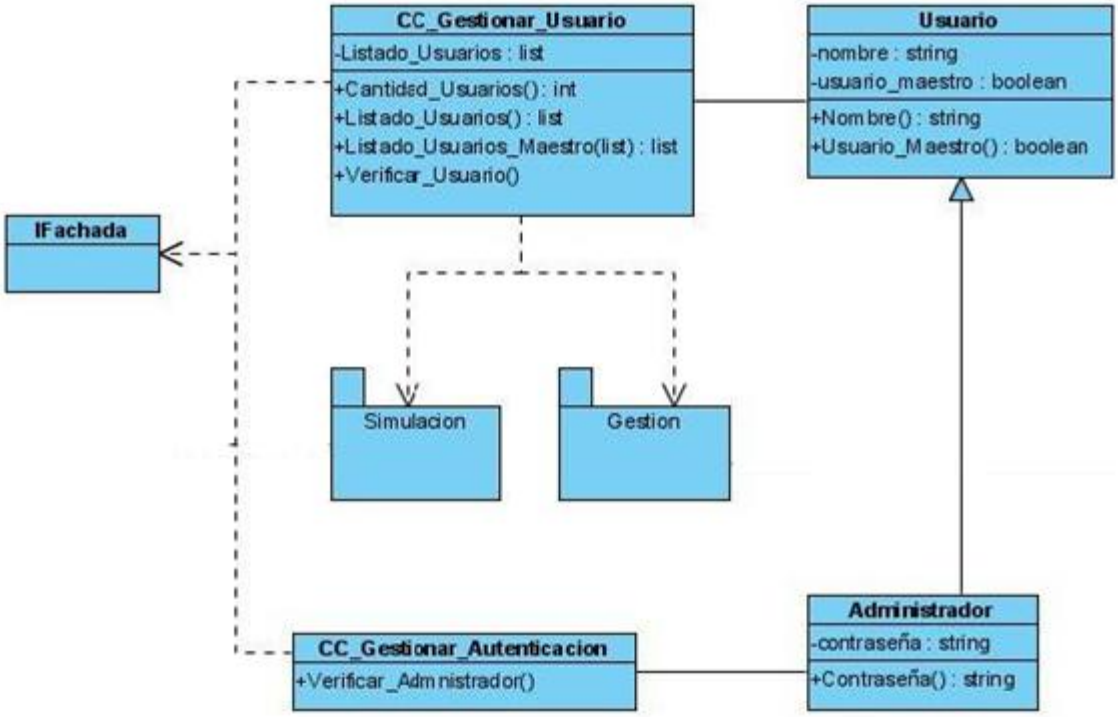

Fig 12. Conexión. Diagrama de diseño.

Subsistema Gestión. Diagrama de diseño.

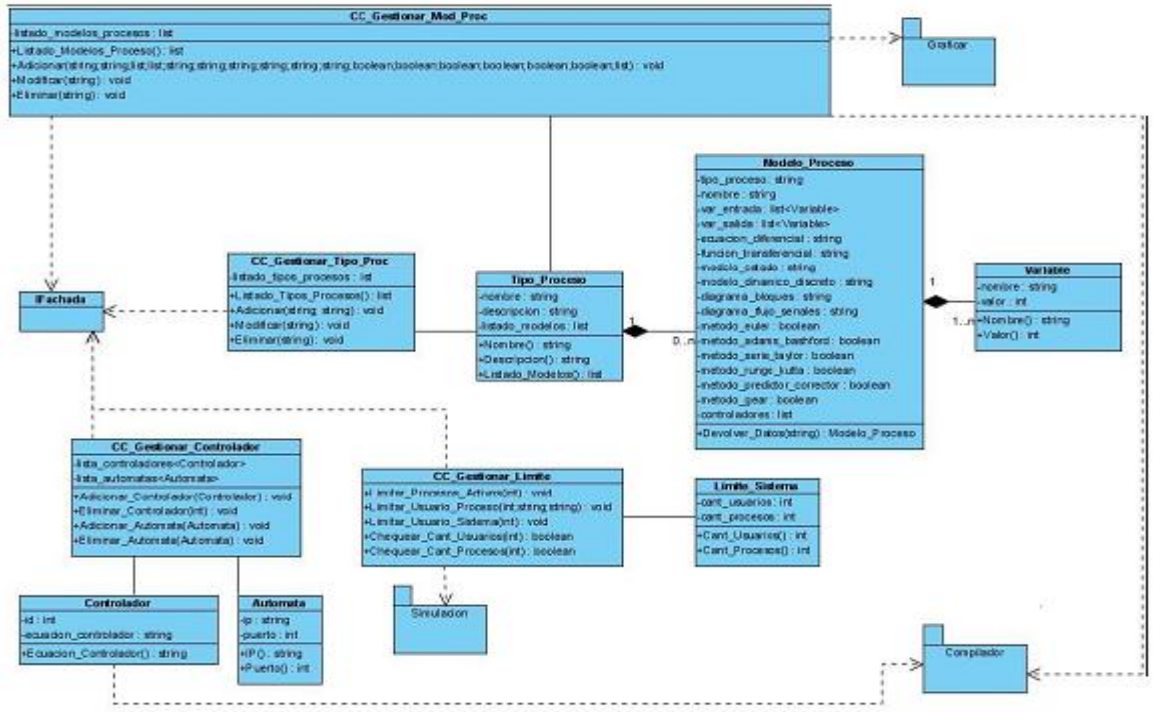

Fig 13. Gestión. Diagrama de diseño.

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. 2000]:
- 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.

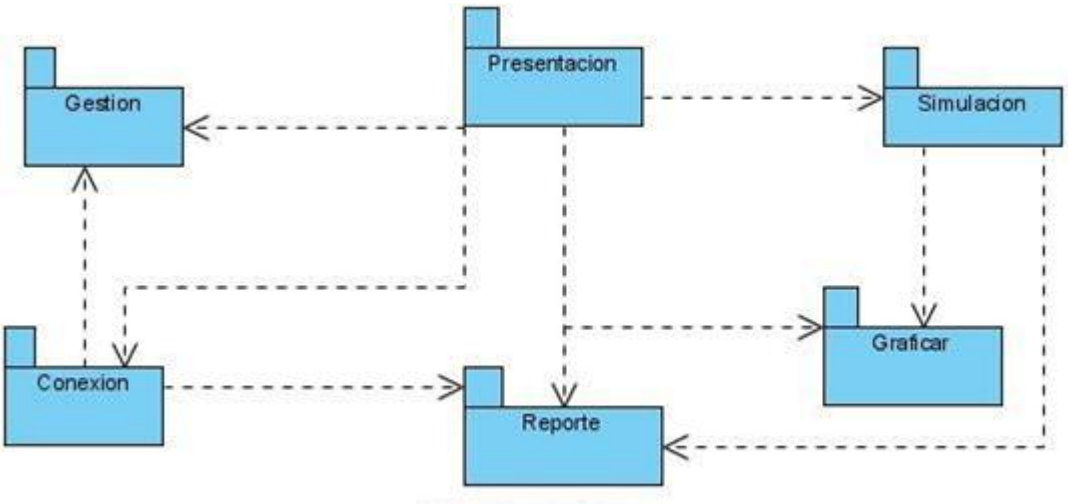

Fig 14. Vista general de implementación

Los diagramas de implementación de los subsistemas más importantes son:

Subsistema Presentación. Diagrama de implementación.

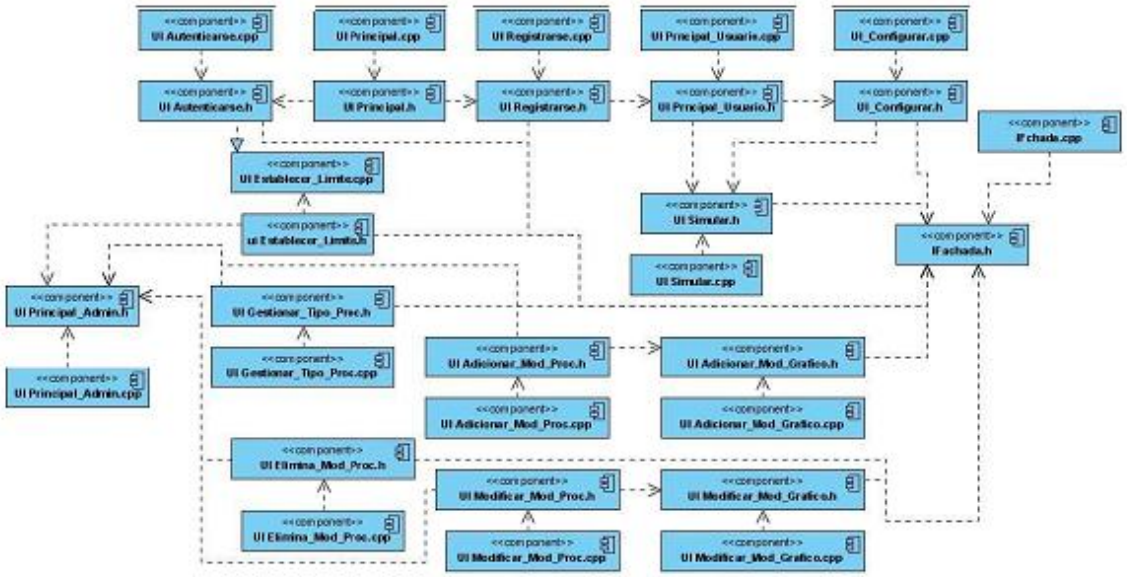

Fig 15. Subsistema Presentación. Diagrama de implementación.

Subsistema Simulación. Diagrama de implementación.

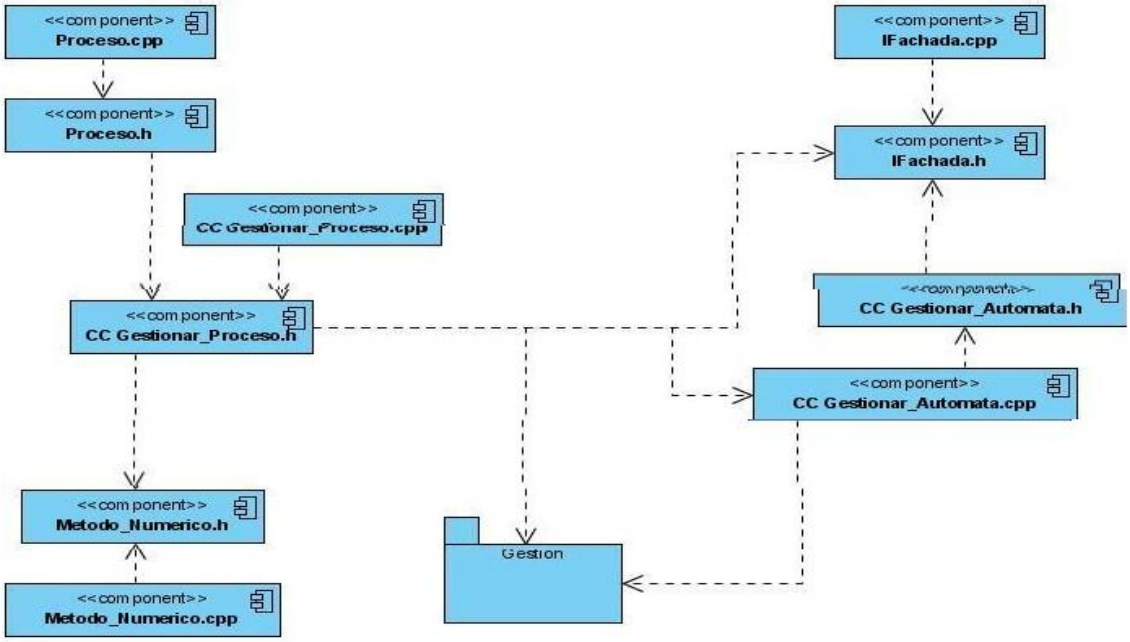

Fig 16. Subsistema Presentación. Diagrama de implementación

Subsistema Conexión. Diagrama de implementación.

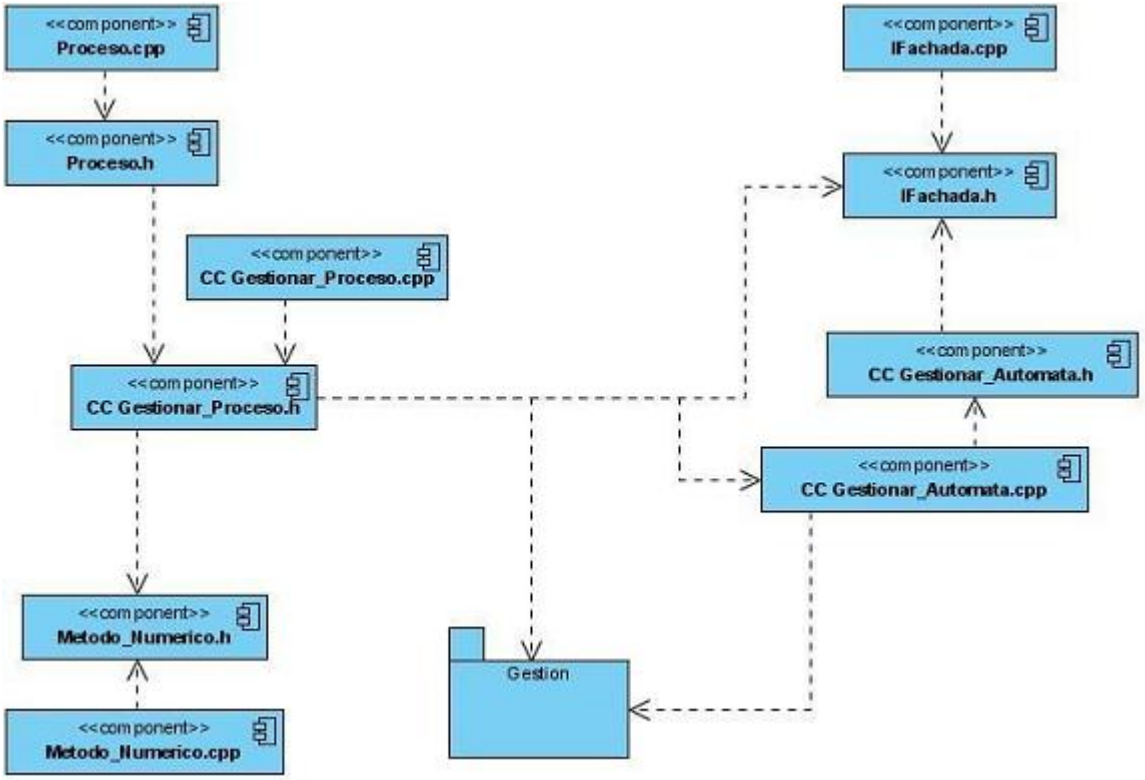

Fig 17. Subsistema Conexión. Diagrama de implementación.

Subsistema Gestión. Diagrama de implementación.

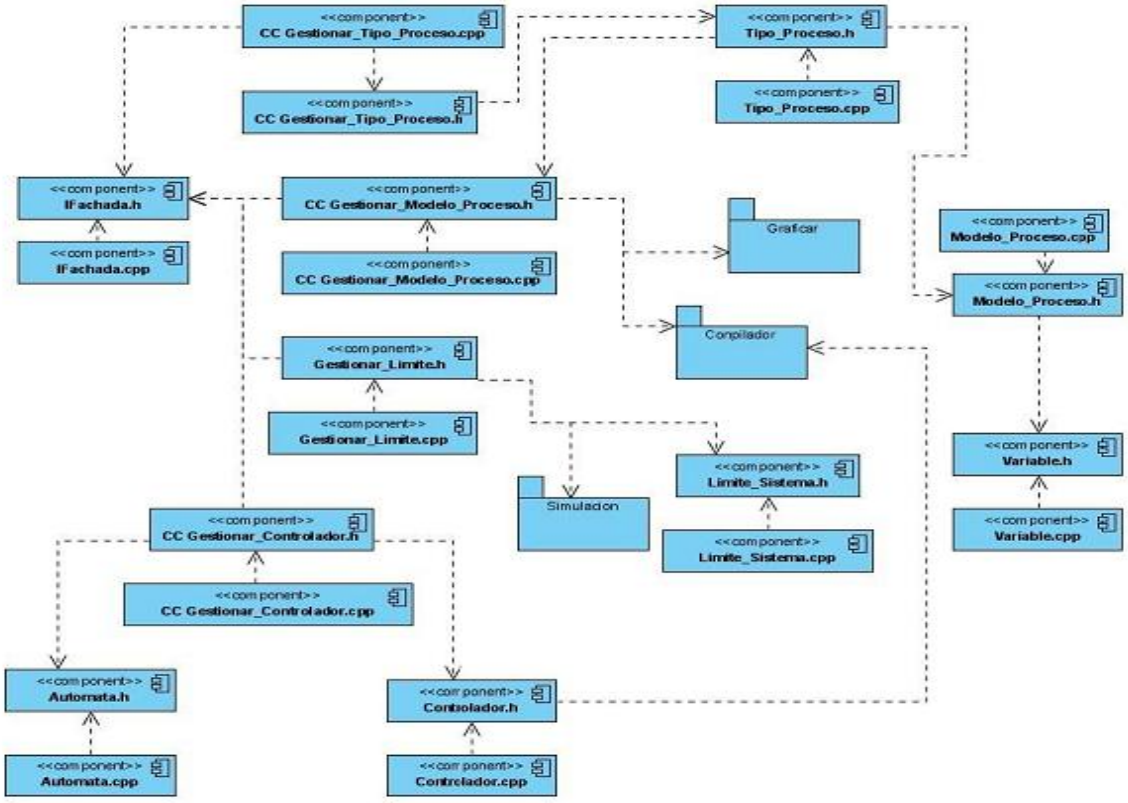

Fig 18. Subsistema Conexión. Diagrama de implementación.

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.

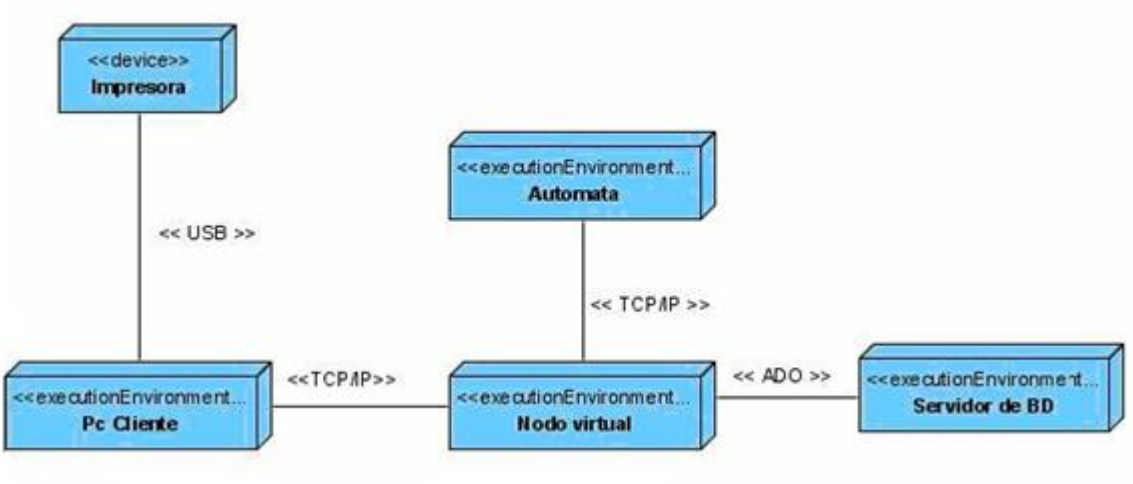

Fig 19. Vista de Despliegue

1. 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.

2. 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

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.

1. 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.

2. 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.

3. 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.

4. 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

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.

Referencias Bibliográficas

1. **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
2. **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.
3. **García de la Puente, Sergio Jesús; Plasencia Crespo, Mileisys**. "*Sistema para la descarga y procesamiento automatizado de patentes: Predictor*. *Rol de arquitecto*". 2007
4. **Jacobson, Ibar; Booch, Grady; Rumbaugh, James.** *"El Proceso Unificado de Desarrollo".* 2000.
5. **Perera Morales, José Raúl.** *"Arquitectura de software para un sistema Gestión de Inventarios".* 2007.
6. **Welicki, L**. *"Patrones y Antipatrones: una Introducción"*. 2007