Universidad de las Ciencias Informáticas
Facultad #3

Arquitectura de los Módulos Entrada de Datos y Generador de Modelos, del Sistema Integrado de Gestión Estadística.

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

Autora: Yusmary Galbán Izquierdo.
Tutor: Ridosbey Milian Iglesias.

Universidad de las Ciencias Informáticas 2007.
Año 49 de la Revolución.

Declaración de autoría

Declaro que soy la única autora de este trabajo y autorizo a la facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los _16_ días del mes de _junio_ del año _2007.

Yusmary Galbán Izquierdo

Ridosbey Milian Iglesias

Agradecimientos

Agradezco a mi tutor por ayudarme a realizar la tesis y por estar dispuesto a brindarnos tanto tiempo.

A mis compañeros por estar a mi lado y ayudarme en lo que pudieron.

A los integrantes del proyecto ONE por ayudarme.

A la Universidad de las Ciencias Informáticas y a Fidel Castro por darme la oportunidad de haber pasado cinco años en esta carrera.

A toda mi familia.

A las personas que de una forma u otra contribuyeron a la realización de este trabajo.

Dedicatoria

Dedico esta tesis a:

Mi mamá por cuidarme, apoyarme y estar junto a mí toda la vida.

A mi papá por ayudarme y quererme.

A mi hermanito por estar conmigo en los buenos y malos momentos de mi vida.

A mi novio por soportarme y darme todo su amor.

A mis suegros por ayudarme y ser tan comprensivos.

A mi abuela y a mi tía por darme su cariño y ayudarme.

Y a todas las personas que me quieren y que les importo un poquito.

Resumen

Las estadísticas en cualquier país del mundo son de gran importancia, debido a que con estas se mantiene un control real de cómo funciona el país. En Cuba la Oficina Nacional de Estadísticas constituye el órgano rector de las estadísticas, actualmente se necesita cambiar el software utilizado para la gestión de toda esta información, llamado MicroSet NT, debido a que no cumple con todas las funcionalidades necesarias para este proceso. El software mencionado anteriormente será sustituido por el Sistema Integrado de Gestión Estadística, que se está desarrollando en la Universidad de las Ciencias Informáticas.

En esta tesis se presenta la arquitectura de los Módulos Entrada de Datos y Generador de Modelos, pertenecientes al Sistema Integrado de Gestión Estadística. En este documento se analizan varios estilos arquitectónicos usados hoy en día para el desarrollo de numerosos proyectos de software a nivel mundial. Además se realiza una explicación de herramientas y plataformas de desarrollo las cuales serán usadas para la implementación de los módulos antes mencionados.

Con la aplicación del estilo arquitectónico en capas, basado en componentes y orientado a objetos, la arquitectura de los módulos antes mencionados proporciona flexibilidad, porque permite realizar cambios en la misma, fomenta la reutilización, debido a que los componentes utilizados en la misma son reutilizables y proporciona la organización del desarrollo de los módulos antes mencionados, de forma que cada uno de los integrantes del equipo de trabajo realiza una parte del sistema.

Índice de Contenido

Indice de Figuras

Introducción

En la actualidad con el avance de las tecnologías en el campo de la informática, los procesos de desarrollo de software se hacen cada vez más complicados, un componente dentro de este proceso que constituye una solución a muchos de los problemas es el desarrollo de arquitecturas de software, que brinda una visión global del sistema y la relación entre las partes del mismo, además la arquitectura ayuda a la descomposición de la complejidad asociada a los proyectos informáticos. Del término arquitectura de software se habla desde hace mucho tiempo y han surgido muchos estilos arquitectónicos y patrones que han ido avanzando en desarrollo y en cualidades, los mismos responden a problemas cada vez más complejos en la sociedad actual. Hoy en día existen varias variantes en cuanto a estilos de arquitecturas, cada uno de los cuales defiende un criterio específico, existen arquitecturas basadas en componentes, arquitecturas basadas en servicios, arquitectura en capas, arquitecturas cliente servidor, arquitectura orientada a objetos, arquitectura orientada a recursos, entre otras.

La arquitectura de software es muy importante y cuando la estudiamos nos damos cuenta que con la representación de la misma podemos lograr una mejor y adecuada integración de las partes que intervienen en el desarrollo del sistema. Es muy importante tener en cuenta la arquitectura desde los momentos más tempranos del desarrollo del sistema, en el trabajo de la ingeniería del software, porque nos ayudará a tomar decisiones más tarde en cuanto el diseño y la implementación del mismo.

Toda la información estadística de nuestro país es procesada en varias oficinas de estadísticas, que se encuentran distribuidas entre las provincias y los municipios, esta información después que pasa por un proceso y varios niveles llega a formar parte en muchos casos de los reportes que se trasmiten al gobierno cubano, para que el país pueda trazarse sus proyecciones y tomar decisiones de gran importancia al respecto. El sistema Nacional de estadísticas cuenta con una Oficina Nacional de Estadísticas (ONE), que se encuentra en Ciudad de la Habana, y constituye el órgano rector en el país, a esta se subordinan un conjunto de Oficinas Territoriales de Estadísticas (OTE) que se encuentran situadas en todas las provincias del país, y a su vez a estas se le subordinan otro conjunto de Oficinas Municipales de Estadística (OME), estas últimas obtienen toda la información estadística de las empresas del país, que tienen la obligación de entregar esta información, a estas empresas se le conoce como centros informantes (CI). La información se recoge a través de modelos estadísticos que contienen un conjunto de indicadores que expresan el comportamiento de la empresa a lo largo de un período de tiempo.

Actualmente para gestionar toda esta información estadística existe un software llamado MicroSet NT que fue desarrollado noviembre de 1984, con el objetivo fundamental de lograr un sistema general e integral para el procesamiento de datos y las ediciones de tablas, y con ello facilitar el procesamiento de la información en la Oficina Nacional de Estadísticas. Este sistema tiene algunas deficiencias, por lo cual la Oficina Nacional de Estadística se ve en la necesidad de desarrollar un software que elimine estas deficiencias y además se le adicionen otras funcionalidades necesarias para la gestión de información estadística en el país.

Uno de los factores que más influyó en la situación en que se encuentra actualmente la gestión de información estadística en las oficinas del país es el no haber realizado un análisis tan fuerte de la arquitectura de software del MicroSet NT. No se hizo un estudio de las tecnologías de almacenamiento de datos adecuadas, porque a pesar de existir técnicas para el desarrollo de las bases de datos, la información fue almacenada en ficheros. No se hizo un estudio de técnicas arquitectónicas. No se tuvo en cuenta como estaba estructurada la infraestructura de las oficinas, porque desarrollaron el software con una arquitectura muy vertical, además de utilizar un paradigma de programación estructurado.

Problema científico.

La no aplicación de una arquitectura adecuada para el desarrollo de los Módulos de Entrada de Datos y Generador de Modelos, afecta el correcto desempeño del proyecto, su flexibilidad, reusabilidad y compromete el futuro mantenimiento del producto una vez implantado.

Objeto de estudio

La arquitectura en el ciclo de desarrollo de un software.

Campo de Acción

La arquitectura en el Sistema Integrado de Gestión Estadística.

Objetivo general

Definir la arquitectura para la implementación de los Módulos de Entrada de Datos y Generador de Modelos.

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

- Definir los estilos de arquitectura a utilizar para el desarrollo de los Módulos Entrada de Datos y Generador de Modelos.
- Definir la línea base de la arquitectura para los Módulos Entrada de Datos y Generador de Modelos.
- Definir la descripción de la arquitectura para los Módulos Entrada de Datos y Generador de Modelos.
- Definir las herramientas y plataformas de desarrollo a utilizar para la implementación de los Módulos Entrada de Datos y Generador de Modelos.
- Definir la estrategia de prueba para la arquitectura de los Módulos Entrada de Datos y Generador de Modelos.

Hipótesis

Si se establece una correcta arquitectura para el desarrollo de los Módulos de Entrada de Datos y Generador de Modelos, entonces se pueden lograr una mayor reusabilidad y flexibilidad en el producto final y se garantiza una mejor mantenibilidad del sistema.

Tareas

1. Analizar varias herramientas y plataformas para el desarrollo de los Módulos Entrada de Datos y Generador de Modelos.
2. Realizar un estudio del estado del arte de varios estilos y patrones de arquitectura de software.
3. Analizar varias estrategias de prueba para la arquitectura de software.

Variable independiente

Arquitectura

Variable dependiente:

Reusabilidad

Flexibilidad

Mantenibilidad

Estrategia de investigación

En las investigaciones se hace necesario tener una estrategia de investigación para poder resolver el problema a que nos enfrentamos, en este caso la estrategia utilizada es la exploratoria, porque nos enfrentamos a una problemática que afecta la sociedad y además no se posee un suficiente conocimiento acerca del asunto que se debe resolver. Por lo que para comenzar a realizar esta estrategia se busca información, datos y teoría sobre el tema, que permitan conocer más la situación a la cual nos enfrentamos.

Métodos científicos de la investigación utilizados

Para el desarrollo de este trabajo se hace necesario utilizar varios métodos y técnicas en la investigación del mismo. Dentro de los métodos a utilizar se encuentra el **Método Científico de Investigación**, ya que en el proyecto es necesario hacer un estudio de la arquitectura del sistema, específicamente la parte de los tipos de arquitectura, las herramientas y metodología a utilizar, que necesitan todo un proceso de investigación previa, para lograr llegar a seleccionar lo más conveniente, para que el producto final quede lo más eficiente posible.

Dentro de los **Métodos Teóricos** a utilizar están el **Método Analítico – Sintético**, el cual se puede emplear a la hora de analizar el historial de versiones del Software MicroSet, utilizado por la ONE actualmente, para llegar a sintetizar a partir de un análisis previo algunas de las deficiencias cometidas en el desarrollo de este software desde el punto de vista de la arquitectura y no cometer los mismos errores en la implementación de los módulos: Generador de Modelos y Entrada de datos.

Otro de los métodos es la **Modelación** que se puede llevar a cabo en el momento de desarrollar el modelo de arquitectura en tres capas, que nos propicia una representación visual y más detallada de las distintas capas que va a poseer la aplicación. Además este método se utiliza cuando se desarrollan los modelos de despliegue y componente en la fase de Análisis y Diseño del proyecto, y en la representación de cada estilo arquitectónico estudiado.

Estructuración del trabajo

Este trabajo está formado por tres capítulos, en el primer capítulo se realiza un estudio acerca de las tendencias de la arquitectura y los estilos arquitectónicos que son utilizados actualmente. Se realiza una explicación de lo que es la arquitectura de software y el estado del arte de versiones anteriores al Sistema Integrado de Gestión Estadística.

En el capítulo 2 se plantea cual es el estilo de arquitectura utilizado en los Módulos Entrada de Datos y Generador de Modelos, a partir del estudio realizado. Además se exponen algunos aspectos relacionados con la línea base de la arquitectura del proyecto y se realiza una combinación de algunos resultados relacionados con el desarrollo de los módulos mencionados.

En el capítulo 3 se realiza un análisis de los resultados obtenidos a partir de la utilización del estilo de arquitectura escogido y el avance del proyecto. También se realizan recomendaciones para próximas iteraciones del producto. Se evalúan los módulos a partir de métricas y pruebas realizadas a la arquitectura de los mismos.

Capítulo 1 Análisis de principales tendencias, estilos y antecedentes del trabajo.

En este capítulo se presentan una serie de aspectos relacionados con la arquitectura de software, se realiza una introducción al tema, donde se explica que es la arquitectura, para que se utiliza y las principales responsabilidades de los arquitectos de software. Se explican las modalidades y tendencias de la arquitectura de software y se selecciona cual es la que más se ajusta a nuestro sistema. Se realiza un análisis de los estilos arquitectónicos existentes y se específica el estilo arquitectónico que se va a utilizar para los Módulos Entrada de Datos y Generador de Modelos. Por último se realiza el estado del arte de versiones anteriores a nuestro producto.

1.1 Introducción a la arquitectura de software y al rol de arquitecto.

Cuando se comenzó a hablar de términos relacionados con la creación de software, en los inicios de la informática, todos se centraban en aspectos dentro de este proceso como la programación, el análisis y se hablaba además de una arquitectura muy precaria. La arquitectura de software nos da una estructura exacta del producto, y es fundamental en el proceso de vida del software ya que consiste en un conjunto de patrones, estilos, vistas y abstracciones coherentes, entre otros términos que son los que en realidad guían la construcción del sistema. La arquitectura se basa específicamente en una serie de objetivos y restricciones funcionales y no funcionales que le dan al sistema toda la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas. La arquitectura tiene además mucho que ver con la organización del sistema y este aspecto fue planteado desde su propio inicio. La arquitectura de software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y con su entorno. Muchas personas han hablado de este término y todas han coincidido en que es una premisa fundamental para el desarrollo de cualquier sistema informático.

La metodología RUP dentro de sus características esenciales plantea que es centrada en la arquitectura porque la misma está presente desde el modelado del negocio hasta la culminación del producto y a medida que pasa el tiempo esta se va fortaleciendo y refinándose, hasta obtener una arquitectura robusta que soporte cualquier cambio que se pueda necesitar en el software. Podemos plasmar en este documento las opiniones de varias personas acerca de la arquitectura, Roger S. Pressman plantea y cito:

"La arquitectura no es el software operacional, más bien es la representación que capacita al ingeniero del software para: analizar la efectividad del diseño para la consecuencia de los requisitos fijados, considerar las alternativas arquitectónicas en una etapa en la cual hacer cambios en el diseño es relativamente fácil y reducir los riesgos asociados a la construcción del software" (Pressman, 2005).

Bass y sus colegas plantearon y cito:

"La arquitectura del software de un sistema de programa o computación es la estructura de las estructuras del sistema, la cual comprende los elementos del software, las propiedades de esos elementos del software visibles externamente, y las relaciones entre ellos" (Bass, Clements y Kazman, 2003).

La arquitectura de software se encuentra influenciada por todas las personas que tienen que ver con el proyecto de una forma u otra, como por ejemplo: los clientes, los desarrolladores, los usuarios finales, el líder del proyecto, entre otros. Esta se relaciona además con la organización a la cual se le va a

desarrollar el sistema, porque la arquitectura va a resolver los problemas de dicha organización, expresados a través de los requisitos.

Para un desarrollo correcto de la arquitectura de software es muy importante tener en cuenta el ambiente técnico, el cual va a estar dado por la experiencia que tenga el arquitecto en las tecnologías y en otros proyectos realizados por él con anterioridad. La arquitectura es un diseño de alto nivel del software, esta se relaciona con los usos y la conexión de los elementos que forman el sistema.

El rol de arquitecto de software tiene una serie de responsabilidades dentro de su trabajo en un proyecto, así como actividades. El arquitecto es el encargado de seleccionar la arquitectura más adecuada para el sistema que responda a las necesidades del usuario, a los requisitos funcionales y no funcionales y además debe ser capaz de lograr los resultados esperados bajo las restricciones dadas.

El arquitecto va a estar presente en todo el ciclo de vida del software y debe mantener el mando en los flujos de trabajo de requerimiento, análisis y diseño e implementación de cada iteración que se realice en el proyecto. Además realiza un modelo entendible y manejable de la arquitectura del software, resumiendo la complejidad del producto y resaltando los detalles más importantes del mismo.

El arquitecto debe tomar decisiones críticas y cruciales para el sistema que deciden muchas veces la dirección que se va a tomar en el proyecto con respecto al mantenimiento y funcionamiento de la aplicación. Para tomar estas decisiones el arquitecto debe apoyarse en el equipo de trabajo y en las restricciones más importantes y cada uno de los cambios que se realicen dejarlos bien explicados y documentados para que las demás personas puedan entender lo que se hizo.

El arquitecto debe trazarse objetivos que estén relacionados con la calidad del sistema. También crea vistas arquitectónicas del sistema para detallar a los clientes o a los interesados, la arquitectura del software en diferentes momentos del desarrollo del sistema. Realiza el diseño arquitectónico de la aplicación a partir de los lenguajes de descripción arquitectónica, estos no son más que herramientas de modelado que soportan desarrollos basados en arquitecturas, describen una estructura de alto nivel, sin detalles de implementación. Posee metodologías arquitectónicas para la elección de estilos, estos deben ser los más adecuados para el sistema que se esté desarrollando.

RUP plantea varias responsabilidades para un arquitecto de software que quisiera mencionar, estas son:

- El arquitecto posee la responsabilidad técnica del sistema y debe ser capaz de desarrollar e implementar todas las funcionalidades de la aplicación económicamente.

- El arquitecto debe trabajar en conjunto con todos los implicados en el proyecto para obtener experiencia de cada uno.
- Debe ser flexible en caso de existir algún cambio en la aplicación que influya en la arquitectura.
- El arquitecto obtiene una arquitectura sólida después de haber pasado por varias iteraciones del producto, ya en la fase de elaboración se debe tener la arquitectura base y estable, porque este es precisamente el hito de esta fase de RUP.
- En caso de tener un sistema complejo esta metodología aconseja que exista un equipo de arquitectos para dividir las tareas porque la arquitectura en este caso puede llegar a ser demasiado compleja.
- El arquitecto debe tener conocimiento y experiencia en el desarrollo de sistemas, porque él es quien va a explicarle la arquitectura a cada uno de los miembros del equipo de trabajo.
- El arquitecto debe tener presente además que la arquitectura está dirigida por los casos de uso.
- Para el arquitecto la línea base de la arquitectura es el documento más importante en su trabajo.
- El arquitecto es responsable de seleccionar los estándares de codificación para el desarrollo del sistema (Jacobo, Booch y Rumbaugh, 2004).

1.2 Análisis de modelos y tendencias actuales de la arquitectura.

En la actualidad existen seis tendencias de la Arquitectura de Software, que se han estructurado de esta forma debido a las manera en que los arquitectos trabajan y hacia que criterios se perfilan más, evidentemente todos tienen su forma de hacer los productos, por lo que existen varios modelos que dividen los estudios de la arquitectura del software a nivel mundial, estos son:

- Arquitectura como etapa de ingeniería y diseño orientada a objetos, este modelo fue planteado por James Rumbaugh, Ivar Jacobson, Grady Booch, Craig Larman y otros, esta fuertemente relacionado con el UML y el Proceso Unificado del Rational. Ellos plantean que la Arquitectura de Software está implícita en la Ingeniería del software. Esta variante dice que la arquitectura tiene que ser establecida desde las fases más tempranas del desarrollo del software para que después la misma no se vea impactada con cambios mayores en la construcción y el mantenimiento del sistema, es decir desde el Flujo de trabajo del negocio ya se debe hacer una definición preliminar de la misma, con el objetivo de tener una idea, la cual se va a ir fortaleciendo con el paso del

tiempo. En este modelo se involucran términos como la abstracción y el ocultamiento de información pero más bien se centran en aspectos relacionados con el encapsulamiento de las clases y los objetos, a los cuales le conciernen una gran importancia. Además se hace necesario la representación de la misma a través de modelos y diagramas, utilizando para ello el UML como lenguaje de representación visual para sistemas que utilizan metodología Orientada a Objetos y el Rational Rose como herramienta que se utiliza para la construcción de los diagramas. Estos diagramas permiten una mayor visión y detalle del modelación del sistema, por eso es que se hacen tan abundantes en esta modalidad. La arquitectura en este modelo se centra en la organización del sistema y de los elementos estructurales del mismo, así como en el estilo arquitectónico utilizado en el desarrollo del software, combinándolo siempre con los patrones de diseño y arquitectura que son muy importantes para que el estilo se aplique con la mayor efectividad posible (Reynoso, 2004).

- Arquitectura estructural, basada en un modelo estático de estilos, ADLs y vistas: esta vertiente es planteada por Mary Shaw, Paul Clements, David Garlan, Robert Allen, Gregory Abowd, John Ockerbloom, académicos de la Universidad de Carnegie Mellon. Esta variante es muy utilizada en la academia, pero en la industria no es muy conocida. Esta posee varias divisiones en su interior, algunas utilizan descripciones verbales y otros lenguajes de representación visual (ADLs). En esta corriente el diseño arquitectónico es de muy alto nivel de abstracción, y no enfatizan mucho en modelos de datos, ni diagrama de flujo. No se habla de clases, ni objetos, porque no se tiene en cuenta el lenguaje de programación (Reynoso, 2004).
- Estructuralismo arquitectónico radical: esta variante se deriva de la anterior, y asume una actitud radical frente al Lenguaje de representación visual UML, en ella existen dos vertientes principales, una que plantea que el modelado orientado a objetos no es muy significativo para la Arquitectura del Software y otra que trata de buscar soluciones a los problemas que tiene el UML para la representación de la arquitectura, porque este lenguaje no se considera que sea un ADL (Lenguaje de descripción de la arquitectura), proponiendo nuevos estereotipos y otros elementos necesarios para ello (Reynoso, 2004).
- Arquitectura basada en patrones, esta reconoce que la tendencia de la arquitectura como etapa de la ingeniería del software y orientada a objetos es importante, aunque no enfatiza mucho en el UML, ni en las metodologías. Esta modalidad hace énfasis en los patrones de diseño planteados

en algunas literaturas muy conocidas como los de GOF entre otros. Además aquí se plantea que el diseño consiste en definir algunos patrones que ya existen para el sistema y estilos arquitectónicos relacionados con estos patrones y por su puesto que sean compatibles, para así conformar la solución más adecuada para el software (Reynoso, 2004).

- Arquitectura procesual es otra de las modalidades que se definen de la arquitectura de software, esta fue realizada en el SEI y con la participación de personalidades como Rick Kazman, Len Bass, Paul Clements, entre otros. Ellos tratan de precisar cuales son los métodos y técnicas que se relacionan con la práctica arquitectónica, desde el punto de vista de la justificación económica de la arquitectura de software, de que forma se seleccionan los atributos de la calidad, como se evalúa que una arquitectura de software propuesta satisface o no los requerimientos funcionales y no funcionales, como se evalúa el costo tanto material como intelectual que puede tener adoptar determinado estilo arquitectónico y no otro para la realización del software (Reynoso, 2004).
- Arquitectura basada en escenarios, esta fue desarrollada en Holanda y enfatiza en las funcionalidades del sistema y los requerimientos. Esta modalidad coincide con aspectos de la arquitectura procesual y define escenarios en los casos de usos, estos son representados a través de diagramas de interacción, con el UML.

Se han tratado de definir otras tendencias de la arquitectura de software en conferencias y debates relacionados con el tema, pero estos son los más generales y los que tienen más materiales que los sustentas. No se puede dejar de decir que hay otros escritos acerca de este tema porque los criterios son variados y diferentes (Reynoso, 2004).

1.3 Análisis de los estilos arquitectónicos utilizados en los sistemas de gestión.

Es muy importante dentro de la arquitectura de software el estudio de los estilos arquitectónicos, los estilos son previos a la elección de las herramientas y plataforma de desarrollo del sistema, además cuando vamos a establecer una arquitectura para nuestro sistema, es bueno tener en cuenta los estilos que existen, actualmente hay aproximadamente 20 estilos abstractos, para así poder buscar el que más se ajusta a nuestro problema y no tener que inventar uno nuevo. Los estilos se reconocen como las 4C de la arquitectura de software, componente, conectores, configuraciones y restricciones, ellos se representa a un alto nivel de abstracción. A continuación se muestra el concepto de estilo arquitectónico definido por Pressman y cito:

"Cada estilo describe una categoría del sistema que contiene: un conjunto de componentes por ejemplo, una base de datos, módulos computacionales que realizan una función requerida por el sistema; un conjunto de conectores que posibilitan la comunicación, la coordinación y la cooperación entre los componentes; restricciones que definen como se pueden integrar los componentes que forman el sistema; y modelos semánticos que permiten al diseñador entender las propiedades globales de un sistema para analizar las propiedades conocidas de sus partes constituyentes" (Pressman, 2005).

Los estilos arquitectónicos no son más que las posibles formas en que nuestro sistema puede estar estructurado, y son de gran importancia y fundamentales en el desarrollo del mismo. Cada uno de ellos tiene características propias que los distinguen de los demás. Están compuestos por una serie de elementos y restricciones arquitectónicas. Dentro de cada estilo se pueden aplicar gran cantidad de patrones y las relaciones entre ellos. Un sistema puede tener varios estilos arquitectónicos combinados que le dan solución al mismo, teniendo siempre en cuenta los requisitos no funcionales. En este epígrafe se da una breve descripción de algunos de los estilos arquitectónicos que existen actualmente.

En los comienzos de la arquitectura solamente contábamos con arquitecturas verticales, orientadas específicamente a un sistema que se iba a utilizar en un ordenador específico, y que este no tuviera ninguna interacción con los demás, sería solamente el usuario y su máquina. Con el desarrollo de las TIC, el incremento de las redes y la necesidad de comunicación surge la arquitectura Cliente-Servidor, esta arquitectura está compuesta por dos elementos principales, uno de ellos es el cliente que puede ser una o varias computadoras conectadas en red, que van a solicitar un servicio, y el servidor que va a ser otra computadora conectada en esa misma red, y le va a dar respuesta al pedido del cliente. La representación de esta arquitectura es la siguiente:

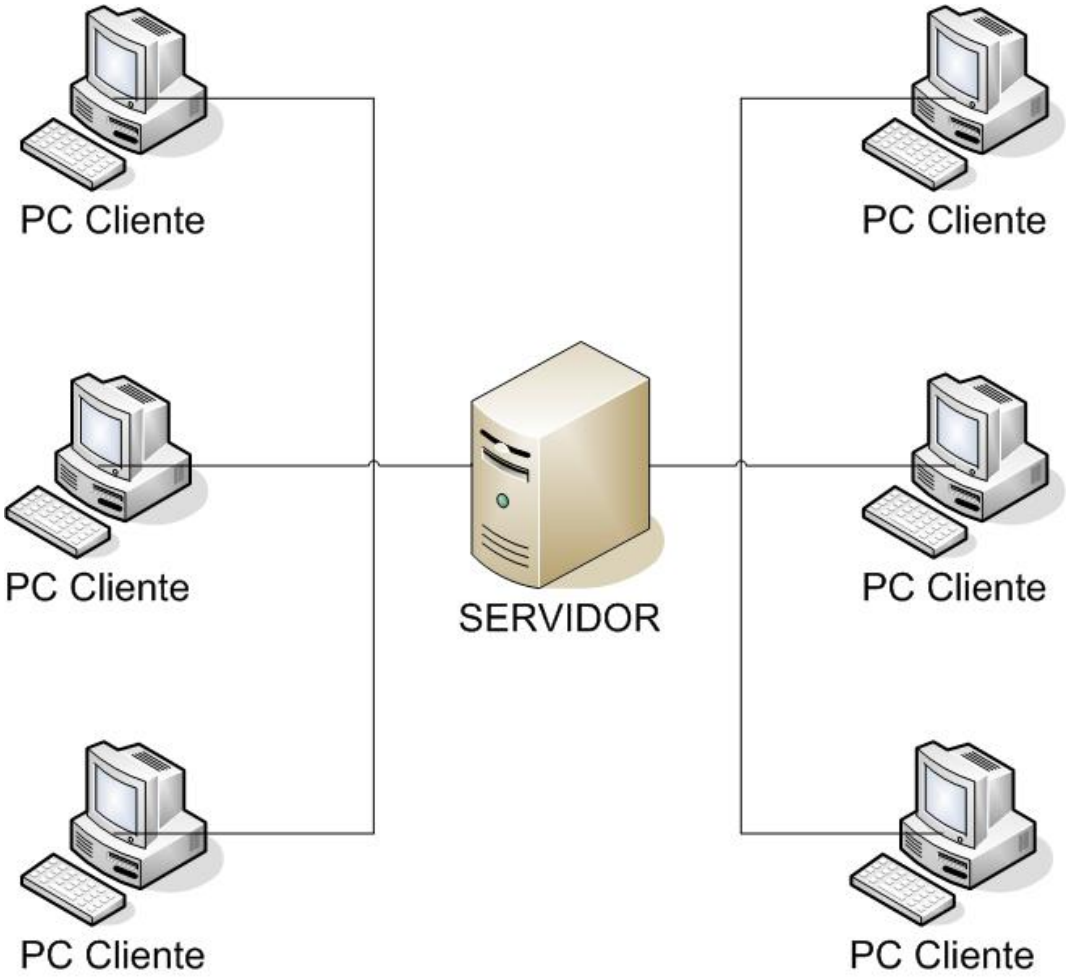


Figura 1 Representación de la arquitectura Cliente _ Servidor.

Este estilo tiene como principal objetivo que los usuarios tengan acceso a los mismos datos, con solamente establecer una conexión con el servidor, cuando sea necesario, obtener la información que necesitan y cerrar la conexión para que otros usuarios se puedan conectar. Además brinda al usuario final el acceso transparente a las aplicaciones, los datos o cualquier otro recurso que este necesite y tiene como principal ventaja que el servidor es el encargado de controlar el acceso a la información y los recursos, por lo tanto si hay algún cliente que no tenga permisos para acceder a estos, entonces el servidor se encargara de rechazar la solicitud.

Además existen arquitecturas basadas en datos, (Pressman, 2005), estas arquitecturas presentan una fuente de datos en el centro, a la cual los clientes pueden acceder para realizar operaciones en esos

datos, pueden crear nuevos datos, eliminarlos o modificar algún elemento en los mismos. En este estilo arquitectónico el software cliente accede a los datos y no tiene conocimiento de cualquier cambio en los mismos o de la acción de otro cliente sobre la fuente de datos. Este estilo es otra forma de la arquitectura Cliente-Servidor.

Otro estilo es el de tubería y filtros (Pressman, 2005), el cual se encuentra entre las arquitecturas de flujo de datos. Este estilo es característico de los compiladores. Se dice que este estilo fue una de los primeros en definirse. El mismo consiste en la conexión de varios componentes, llamados filtros a través de tuberías que trasmiten datos de un componente al siguiente, los filtros están preparados para recibir un grupo de datos de entrada y devolver una salida al siguiente filtro. Los filtros pueden descomponerse, pero los tubos no. Los tubos conectan exactamente dos puertos. Dentro de la arquitectura de flujo de datos también existe el patrón Secuencial por lotes (Pressman, 2005), este aplica una serie de componentes (filtros) que se encuentran en línea recta para ser transformados.

Otra de las arquitecturas es la orientada a objetos, la cual aparece explicada en (Pressman, 2005) en esta los componentes son orientados a objetos, basados en principios de encapsulamiento, herencia y polimorfismo, tanto los datos como las operaciones que se realizan con los mismos, se encuentran encapsulados en los componentes. La comunicación entre componentes se consigue a través del paso de mensajes, esta comunicación es implícita, se le habla a una instancia de objeto creada previamente. En este estilo las interfaces están separadas de las implementaciones, en general la distribución de objetos es transparente. El mejor ejemplo de esta arquitectura para sistemas distribuidos es el Common Object Request Broker, este media las interacciones entre objetos clientes y objetos servidores. Esta arquitectura forma parte de la familia de estilos Llamada y Retorno. En este estilo tanto el refinamiento como la reutilización son sencillos.

Otro de los estilos es el Peer-to-Peer, del cual se hace referencia en, (Reynoso y Kicillof, 2004), en esta familia los componentes se encuentran separados entre si, se comunican a través de entidades que se envían mensajes entre ellas para relacionarse, este estilo se basa en la modificación por separado de las partes que intervienen en el sistema. Las entidades se pueden enviar mensajes entre ellas, pero no se controlan directamente unas a las otras.

Otra de las arquitecturas es la orientada a servicios (Ciudad y Soto, 2007), esta es una arquitectura en la que los recursos están interconectados en una red y no son más que servicios a los cuales se puede acceder por terceros a través de una interfaz estándar, en este modelo existen tres actores principales, el

proveedor del servicio, el registro del servicio y el solicitante del servicio, un componente dentro de esta arquitectura podría considerarse como un servicio, el solicitante accede a los servicios a través de webservices generalmente, y los formatos de intercambio de los mismos se basan en XML. La característica más importante dentro de este estilo es que los componentes tienen muy bajo nivel de acoplamiento entre ellos, las funcionalidades de los servicios se pueden modificar sin tener que rendir cuentas a los que lo necesitan, así como una mayor flexibilidad ante posibles cambios que sean necesarios en algún momento. Este estilo arquitectónico es miembro de la familia Peer-to-Peer, explicada anteriormente y constituye una idea nueva y exitosa en nuestros días aunque no deja de tener desventajas como todo estilo.

En el diagrama siguiente se representa este tipo de arquitectura.

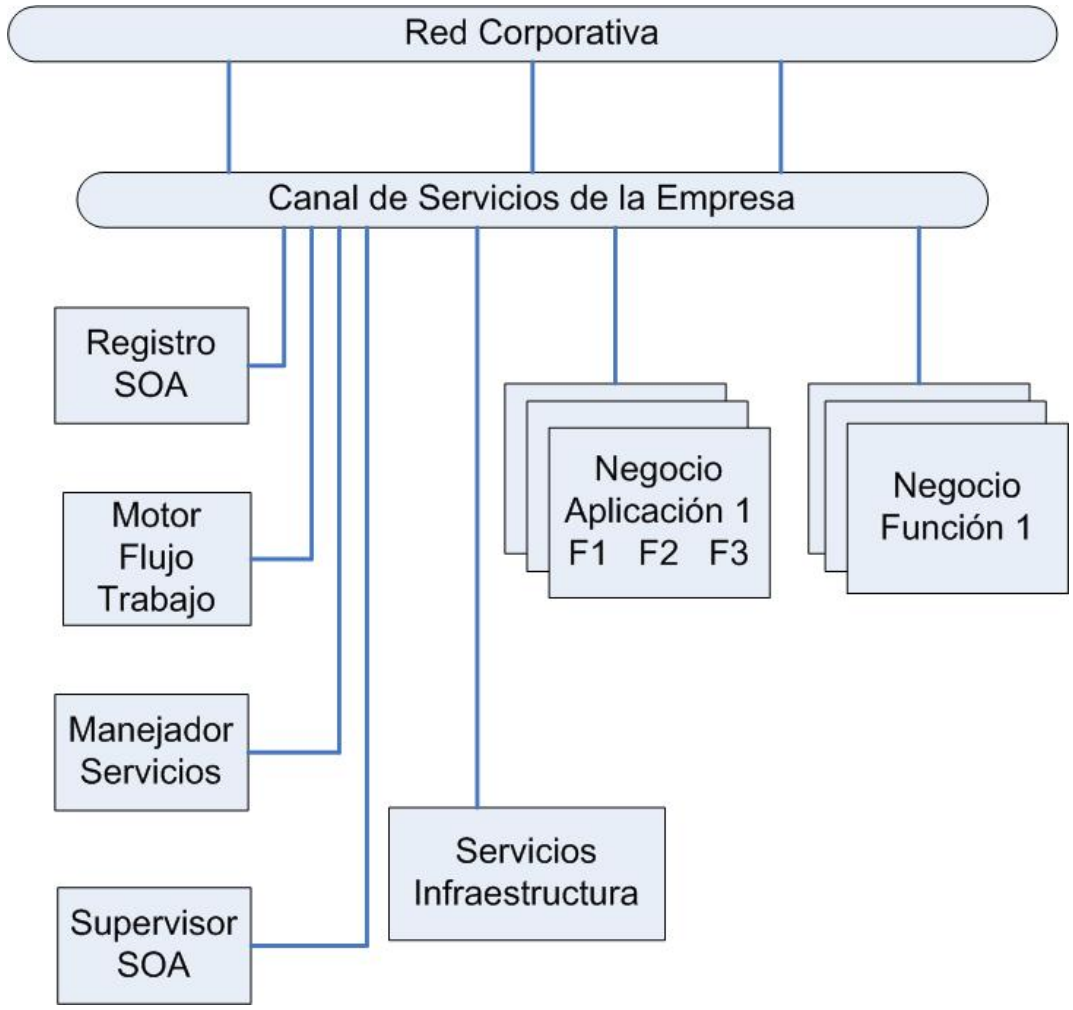


Figura 2 Arquitectura orientada a Servicios.

En este diagrama se ilustran componentes importantes dentro de esta arquitectura como son:

- Canal de Servicios de la empresa
- Registro SOA
- Motor de Flujo de Trabajo
- Manejador de Servicios
- Supervisor de SOA

Además tenemos la arquitectura basada en componentes (Pressman II, 2005), la cual consiste en que uno o varios componentes en la aplicación, utilizan otros componentes que ya están definidos, se relacionan entre si, y así proporcionan los servicios que se necesitan para que el sistema funcione. En

este estilo las interfaces son el elemento básico de conexión entre los componentes, se debe conocer las interfaces que puede ofrecer cada componente, así como las que necesita para su operatividad, porque estas son el centro de la atención en el diseño arquitectónico. Cuando se desarrollan sistemas en los cuales la arquitectura sea de este tipo se trata de introducir todos los requerimientos funcionales en los componentes que forman parte del sistema. La conexión y coordinación entre los componentes se define mediante la arquitectura y estos deben corresponder con las necesidades de la arquitectura y del sistema. Un componente en este tipo de arquitectura representa una parte del software que puede ser reemplazada, por lo que debe poseer una baja dependencia con los demás componentes del sistema. Además los componentes deben ser actualizados cada vez que se cambien los requisitos del sistema. En un sistema de software se pueden utilizar componentes que ya se hayan desarrollado con anterioridad y que se hayan utilizado en otros proyectos, o se pueden diseñar nuevos componentes para el sistema, los componentes nuevos deben ser comprobados y documentados. Esta arquitectura proporciona grandes ventajas para la calidad del software, ya que los gastos en el desarrollo del sistema no son tan elevados con este tipo de arquitectura. A continuación se representa esta arquitectura.

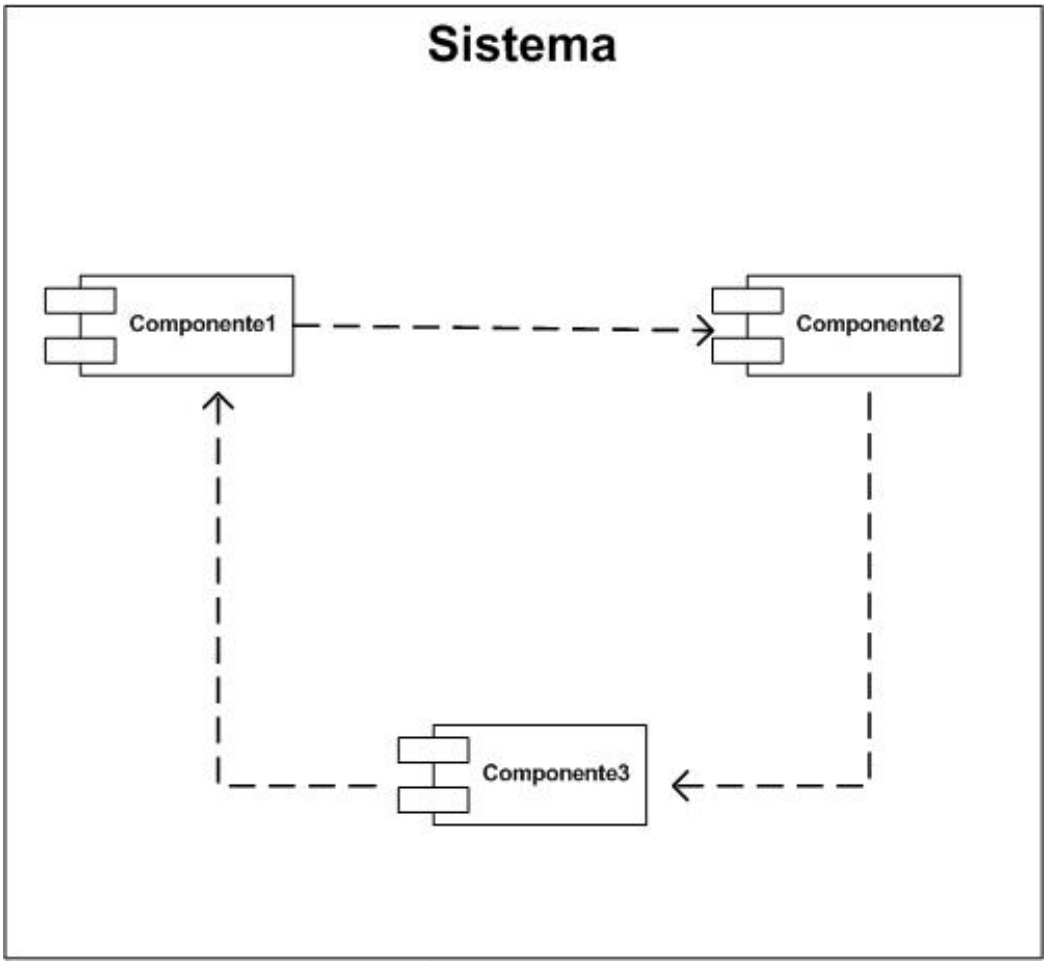


Figura 3 Representación de la Arquitectura basada en componentes.

Otro de los estilos que actualmente se utiliza mucho, es la arquitectura en capas (King, 2005), este estilo está conformado por una serie de capas o niveles, independientes una de otras, uno de los ejemplos de este estilo arquitectónico es la arquitectura en tres capas, la cual obedece dos reglas específicas, estas son:

- Una capa depende solamente de la capa inferior a ella.
- Cada capa es inconsciente de cualquier otra capa, a excepción de la capa que se encuentra debajo de ella.

La arquitectura en 3 capas va a estar compuesta por una capa de presentación, esta va a ser la capa superior en esta arquitectura, representa la interfaz de usuario y es la encargada de interactuar con este.

Esta capa es la encargada de la codificación y el control de las páginas y de las formas de navegación del usuario por las pantallas. Otra capa es la del Negocio, a esta se le conoce como capa intermedia y es donde se localiza la lógica del negocio, esta es responsable de implementar las reglas del negocio, las validaciones y los cálculos, esta capa recibe la petición del usuario a través de la capa de presentación y se encarga de darle respuesta mediante los repositorios de información. En algunos sistemas esta capa tiene su propia representación interna de las entidades del dominio del negocio, en otros utiliza el modelo definido por la capa de persistencia. La capa de persistencia no es más que un grupo de clases y de componentes responsables del almacenamiento de los datos, esta incluye necesariamente un modelo de las entidades del dominio del negocio. La capa de datos, es la representación real de los datos, y representa además la persistencia del estado del sistema, en esta capa se van a encontrar los datos, o sea esta capa representa la base de datos que puede estar desarrollada en SQL Server, o en cualquier otro sistema gestor de base de datos. Aquí se representan los procedimientos almacenados y el esquema de los datos. Además cada sistema tiene una infraestructura de clases de ayuda o utilidad, que son usadas por cada capa de la aplicación, estos elementos infraestructurales no forman una capa puesto que no obedecen las reglas para la dependencia de la capa intermediaria de una arquitectura en capas. El siguiente diagrama representa esta arquitectura:

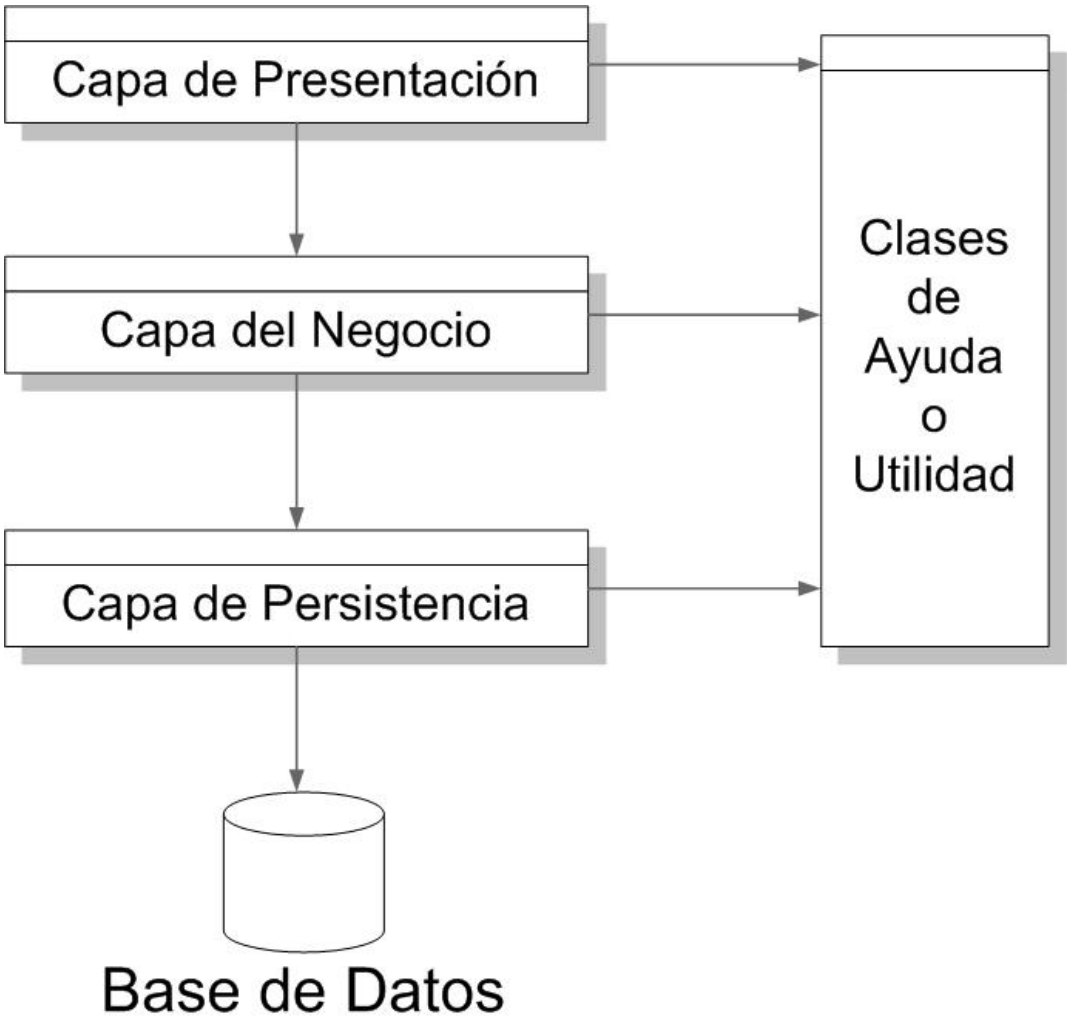


Figura 4 Representación de la arquitectura en tres capas.

Las capas pueden estar formadas por paquetes y subsistemas, el uso de la arquitectura en capas es muy frecuente en nuestros días, porque permite una mayor organización del sistema y para este estilo hay una gran cantidad de patrones que se recomiendan en varias bibliografías. Este estilo se usa mucho en sistemas informáticos complejos porque simplifica la comprensión de los mismos. Los elementos de una capa pueden interactuar con los demás elementos de esa misma capa, para mantener la flexibilidad, y el estilo nos permita eliminar una capa completa si es necesario en algún momento.

Otro tipo de arquitectura es la basada en recursos (Reynoso y Kicillof, 2004), esta define un conjunto de recursos y métodos que permiten acceder a estos recursos y manipular el estado de los

mismos. Esta arquitectura se centra más bien en la Web. Un ejemplo de esta arquitectura es el modelo REST que combina estilos como el cliente-servidor, máquinas virtuales y arquitectura basada en eventos, esta es una especie de arquitectura orientada a servicios sin el concepto de webservices, es una arquitectura formulada en términos de recursos, los recursos serían URIs y representaciones basadas en XMLs. Esta arquitectura es capaz de comprender cualquier clase en la Web, por ser una combinación de varios estilos. Además la transferencia entre las entidades es a través de representaciones usando conectores que utilizan formato XML, para entenderse entre sí.

Además tenemos la arquitectura basada en eventos (Reynoso y Kicillof, 2004), primeramente empezar diciendo que existen formas de programación que se basan en eventos, este estilo posee componentes que pueden anunciar uno o varios eventos. Los conectores de este estilo incluyen procedimientos de llamada tradicionales y vínculos entre anuncios de eventos e invocación de procedimientos. Un componente puede anunciar su interés en un evento determinado asociando un procedimiento a dicho evento. El anuncio de un evento provoca la invocación de determinados procedimientos asociados a este evento. Este modelo a veces suele vincularse con el patrón Observador, esta arquitectura es muy fácil de programar. Existen dos modelos de arquitectura e implementación de este estilo los cuales son:

- Eventos fuertemente acoplados.
- Eventos débilmente acoplados.

De forma general existen muchos estilos arquitectónicos que cada uno resuelve un problema específico y están orientados más bien a las distintas formas en que se puede estructurar el sistema que se está desarrollando. Durante el diseño y la programación estos estilos serán refinados mediante técnicas de patrones, refactorización, etc. Existen herramientas en .Net y otras para la programación de algunos de estos estilos arquitectónicos. Los estilos arquitectónicos que más se usan actualmente en el desarrollo de software a nivel mundial son: arquitectura orientada a servicios, la arquitectura basada en componente, la arquitectura orientada a objetos y la arquitectura en capas, además de otros tipos de arquitecturas que son derivadas de los estilos que ya existen, es decir son la combinación de varios estilos unidos, por ejemplo la C2, GenVoca y REST, estas también son muy utilizadas.

1.4 Definición de la modalidad y los estilos arquitectónicos correspondientes a los Módulos Entrada de Datos y Generador de Modelos.

A través del estudio realizado de las diferentes tendencias propuestas y explicadas anteriormente, la arquitectura como etapa de ingeniería y diseño orientada a objetos es la que más se asemeja a la línea por la cual se ha trabajado para la elaboración de los Módulos Entrada de Datos y Generador de Modelos. Primeramente la arquitectura definida para el sistema se realiza a través de las prácticas del Proceso Unificado de Desarrollo, el cual entre sus características principales plantea que es centrado en la arquitectura, porque esta nos permite ver el sistema como un todo y tener una visión común entre los implicados en el mismo, usuarios, desarrolladores y otros. La arquitectura de los módulos antes mencionados, se ve afectada durante el proceso de desarrollo por aspectos relacionados con el sistema operativo, plataforma de desarrollo, protocolos de comunicación, gestor de base de datos, que se engloban en los requerimientos no funcionales del sistema, además de tener en cuenta aspectos de la calidad del sistema, rendimiento, evolución, por lo que esta es flexible. Se establece tempranamente una arquitectura que luego no se vea impactada con los cambios que se puedan presentar en las etapas de construcción y mantenimiento del mismo. La arquitectura permite el desarrollo de los casos de usos requeridos, y le da la forma al sistema, por lo que está muy asociada a los casos de uso. La arquitectura del sistema va evolucionando con el desarrollo de las fases, convirtiéndose al final en una arquitectura robusta. La arquitectura está implícita en la Ingeniería del software, la cual es una tecnología multicapa en la que se identifican los métodos, el proceso y las herramientas. RUP define una serie de actividades que corresponden al arquitecto, de las cuales se generan los artefactos correspondientes, que se desarrollan para conformar toda la documentación relacionada con la arquitectura del sistema. Además en los Módulos Entrada de Datos y Generador de Modelos se utiliza el UML como lenguaje de representación visual para representar el diseño y otros aspectos que se obtienen con los 6 flujos de trabajo de ingeniería que propone la metodología RUP. Se utiliza el Rational como herramienta de modelación visual para el proceso de modelación del negocio, análisis de requerimientos y diseño de arquitectura de componentes. Se tienen en cuenta en la arquitectura del sistema los elementos orientados a objetos, se definen encapsulamiento de clases. La arquitectura se describe a través de un alto nivel de abstracción, donde se involucran componentes, conectores, configuraciones y restricciones que describen el estilo arquitectónico que se utiliza, el cual es una combinación de varios estilos por la complejidad del sistema. De esta forma queda explicado que coincido con los aspectos definidos por James Rumbaugh, Ivar Jacobson y Grady Booch, acerca de la arquitectura de software.

A partir de los conceptos estudiados y de otras reflexiones realizadas con el tema de la arquitectura de software, en los módulos: Generador de modelos y Entrada de datos, se va a utilizar una arquitectura en capas basada en componentes y orientada a objetos, porque es la que abarca los requerimientos funcionales y no funcionales necesarios en estos módulos y además porque esta arquitectura a sido muy utilizada y ha coincidido con uno de las tendencias tecnológicos más importantes a fines del siglo XX, además es una de las arquitectura planteada por la metodología RUP y las métricas de esta metodología son las que vamos a seguir en el desarrollo de estos módulos. Esta arquitectura proporciona modularidad y componibilidad lo que constituye una evolución para los sistemas desarrollados hoy en día. Además el desarrollo de software basado en componentes es la evolución natural de la ingeniería software para mejorar la calidad, disminuir los tiempos de desarrollo y gestionar la creciente complejidad de los sistemas. Podemos decir también que la arquitectura en capas es muy factible porque simplifica la comprensión de estos dos módulos, reduciendo las dependencias y además nos ayuda a identificar que se puede reutilizar y que no y evita poner código indebido en la capa impropia. Otro de los motivos por lo que escogimos esta arquitectura es por la relación entre las capas, que es unidireccional, o sea solo las capas superiores pueden utilizar los servicios que brindan las inferiores. Este estilo admite muy naturalmente optimizaciones y refinamientos. Este estilo soporta un diseño basado en niveles o capas de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales. Esta arquitectura presenta robustez y seguridad que son imprescindibles para estos módulos, sobre todo la seguridad, muy necesaria para la preservación de la información estadística en el proyecto, porque esta información debe ser confidencial y no ser modificada por personal que no este autorizado a acceder a la misma. El término arquitectura de software esta sufriendo transformaciones aún y seguirá evolucionando, pero pienso que con la arquitectura antes propuesta para los Módulos Generador de Modelos y Entrada de Datos que se va a desarrollar y de acuerdo con el problema que se quieren resolver se va a dar soluciones a muchas deficiencias y el país podrá contar con una adecuada gestión de información estadística.

1.5 Estado del arte de versiones anteriores del Sistema Integrado de Gestión Estadística.

Desde 1984 con la introducción de las primeras computadoras a la Dirección de Estadística del país se pensó en la construcción de un sistema que resolviera las dificultades que existían con la gestión de información estadística en Cuba, para ello se desarrollo el software MicroSet, el cual a medida que pasó el tiempo fue adquiriendo varias versiones, por mejoras que se le fueron introduciendo, con la llegada de las redes y otras.

Las oficinas de estadística del país trabajan actualmente con el software MicroSet NT (Hernández y Piquera, 1997), este permite procesar modelos sin que haya que realizar una programación específica para cada uno, y tiene como objetivo principal el procesamiento de datos y la creación de tablas para así poder facilitar la gestión y procesamiento de la información estadística en nuestro país. Este sistema fue construido en el lenguaje de programación Borlan Pascal versión 7.0 y posee una arquitectura muy simple, se puede decir que prácticamente no presenta detalles arquitectónicos. Además las herramientas utilizadas en el mismo son muy atrasadas, por el momento en que se realizó. Es un sistema multiusuario y permite la carga, actualización y validación de un modelo de forma simultánea desde todas las estaciones de trabajo incluidas en la red.

En el MicroSet el usuario define sus directorios de trabajo, esto implica que se pueda configurar de diferentes maneras para el trabajo en las oficinas. Por ejemplo:

- Puede instalarse completamente en un servidor de red y cada usuario ejecutarlo en este.
- Puede instalarse solamente en una computadora y poder ser utilizado solamente por este usuario.
- Se pueden instalar algunos directorios en el servidor y otros en el puesto de trabajo de la persona que va a utilizarlo, en este caso es necesario que en la computadora de la persona estén presentes los ficheros siguientes, MSET.EXE, ASIGNA.EXE, RTM.EXE, DPMI16BI.OVL, ASIGNA, README.COM.

MicroSet NT a pesar de poseer funcionalidades que son necesarias en el procesamiento de información, no abarca todo lo que se necesita actualmente, por lo que fue necesaria la construcción del Sistema Integrado de Gestión Estadística y con ello la arquitectura de los Módulos Entrada de Datos y Generador de Modelos, para solucionar muchos de los problemas existentes con este software. Estos módulos serán desarrollados con tecnología avanzada, con un equipo de desarrollo en el que estará presente el arquitecto. Se tienen en cuenta todas las restricciones de hardware y software para lograr el funcionamiento adecuado de los módulos. Además cuentan con servidores de base de datos, para implementar una arquitectura Cliente – Servidor.

Capítulo 2 Diseño Arquitectónico de los Módulos Entrada de Datos y Generador de Modelos.

En este capítulo se define cual va a ser la arquitectura que se utiliza en los Módulos Entrada de Datos y Generador de Modelos y además se da una explicación de porque se utiliza esta. Además se darán a conocer detalles de la línea base de la arquitectura, que constituye un esqueleto del sistema, definiendo primeramente una arquitectura inicial a partir de los casos de uso, de manera económica e ir adicionándole aspectos importantes a la arquitectura, hasta obtener una maduración de la misma con el paso del tiempo y agregándole elementos de diseño e implementación, requisitos funcionales, requisitos no funcionales, entre otros. También se realiza la combinación de algunos resultados referentes a las herramientas y estrategias de desarrollo, ubicación de puestos de trabajo, entre otros. Se definen algunas de las pruebas que se le van a realizar a la arquitectura, para comprobar su robustez.

2.1 Definición de la línea base de la arquitectura.

La línea base de la arquitectura es uno de los artefactos más importantes que se define en la metodología RUP, como resultado del trabajo de los arquitectos de software y se obtiene en la fase de Elaboración. Los arquitectos de software deben ser capaces de desarrollar una línea base de la arquitectura estable, a partir de recursos que existen para ello, como los estilos arquitectónicos, y los patrones de diseño, además los arquitectos deben conocer las ventajas y desventajas de cada estilo para así poder saber cual es el que más se ajusta al sistema que se esté desarrollando y así poder obtener una descripción de la arquitectura lo más correcta posible, para que soporte los posibles cambios que surjan en el sistema en próximas iteraciones (Jacobo, Booch y Rumbaugh, 2004).

2.1.1 Aspectos importantes de la arquitectura de los sistemas.

Existen algunos objetivos y restricciones de los sistemas en cuanto a seguridad, portabilidad y reusabilidad que son significativas para la arquitectura. Ellos son:

- El tiempo de respuesta de la aplicación en el caso de los subprocesos: carga de modelos XML, y seguridad debe ser atendiendo a las condiciones de velocidad de los CPU de no menos de 400 MHZ.
- El servidor de la entidad debe de soportar varias conexiones a la vez por lo que debe de tener como mínimo 512 MB de RAM para poder responder a todas las solicitudes que se le puedan hacer al servidor montado. Esta capacidad satisface la necesidad de memoria para el servidor de Base de Datos (SQL 2000) que estará en esta misma máquina.
- La capacidad de disco duro en el servidor debe de ser no menos de 80 GB, en el caso de los clientes no menos de 30 GB.
- La red existente en las instalaciones debe de soportar la transacción de paquetes de información de al menos unas 10 máquinas recurrentes.
- Para hacer más fiable la aplicación debe de estar protegida contra fallos de corriente y de conectividad, para lo que se deberá parametrizar los tiempos para realizar copias de seguridad. Implementar las transacciones de paquetes en la red con el protocolo TCP/IP que permite la recuperación de los datos.
- Los Módulos de Entrada de Datos (MED) y Generador de Modelos (MGM) proporcionan un subsistema de seguridad, por lo que la interfaz del sistema para la integración con el Sistema

Integrado de Gestión Estadística (SIGE) debe permitir crear los distintos roles para cada usuario definido en la aplicación.

- Todas las funciones del sistema deben ser accesibles a partir de un sistema implementado de ventanas de navegación.
- En cuanto a los sistemas operativos en el caso de los clientes deben de tener instalado un sistema operativo de la familia Windows NT. En caso de los servidores deben tener sistema operativo Windows Server con Service Pack 1.
- Todas las estaciones de trabajo contarán con la instalación del Framework 2.0.
- Todos los componentes estarán encapsulados en sus capas respectivas lo que hará que sean todos reutilizables.

2.1.2 Requerimientos funcionales de los Módulos Entrada de Datos y Generador de Modelos.

Módulo Entrada de Datos.

Autenticar.

Este requerimiento es muy importante en el sistema porque en dependencia del rol que tenga el usuario, el sistema le da permisos para que trabaje con la parte de la aplicación que esté autorizado, un mismo usuario puede interpretar varios roles en el sistema. El administrador es el autorizado para dar los permisos a cada usuario en el sistema.

Adicionar Usuario.

Cuando se va a adicionar un nuevo usuario en el sistema, el administrador que es el encargado de realizar esta operación, debe especificar usuario, nombre, apellido y contraseña.

Eliminar Usuario.

Cuando el administrador va a eliminar un usuario en el sistema, debe seleccionarlo de una lista y confirmar si lo desea eliminar.

Asignar rol.

El administrador es el encargado en el sistema de asignar los roles a los usuarios, el cual da el nivel de acceso que tiene cada usuario para que pueda trabajar en la parte del sistema que le corresponde. La aplicación da los permisos a los usuarios en el sistema según el rol que este desempeñe.

Instalar Modelo.

Cuando se desea instalar un modelo para digitarlo, el sistema debe cargar el modelo desde el sistema de archivo. Esta operación es realizada por el digitador de la oficina de estadística.

Eliminar Modelo.

Cuando se desea eliminar un modelo estadística el sistema debe preguntar si realmente se desea realizar esa operación, para que el digitador sepa a que consecuencia se enfrenta cuando realiza esta acción.

Actualizar Modelo.

Para actualizar un modelo estadístico el sistema debe cargar el mismo del sistema de archivos, esta acción es realizada por el digitador de la oficina.

Administrar Configuración de Módulo.

El sistema debe permitir la personalización del módulo, en función del entorno donde corre el mismo de forma que este varíe en dependencia de que esté instalado en un CI o en alguna Oficina Estadística.

Digitar Datos Estadísticos.

El sistema debe de permitir cargar los modelos (entiéndase el formulario) desde un archivo y llenarlos según los datos, tanto desde una vista amigable diseñada para los CI, como desde una interfaz de formularios dedicada a las oficinas estadísticas y adecuada a las características del Digitador (similares a la de MicroSet). Este proceso será realizado por el Digitador.

Guardar los Datos digitados en archivo.

El sistema debe permitir al digitador guardar el modelo llenado en un archivo xml.

Validar Modelo.

El validador es el encargado de la validación de los datos estadísticos una vez que se haya realizado cada uno de los modelos. Este debe verificar que los datos entrados sean los correctos.

Auditar el Sistema.

El sistema debe brindar la posibilidad de realizar un reporte con la fecha, hora, evento, usuario y descripción de las modificaciones que se hicieron en el mismo, para de esta forma poseer un control detallado del manejo de la información en la aplicación.

Exportar datos del modelo al formato de MicroSet.

El sistema debe exportar datos del modelo al formato de MicroSet con el fin de lograr compatibilidad con este sistema, por lo que luego de realizado el llenado de modelos y guardados se exportarán a dicho formato, esto no se realizará en el caso de los CI.

Introducir datos del Modelo en la Base de Datos una vez que estos estén validados.

Cuando se culmina con las validaciones correspondientes el sistema debe permitir que se guarden los datos del modelo (la información estadística digitada) en la base de datos.

Módulo Generador de Modelos.

Crear modelo estadístico.

Para la creación de los modelos se deben introducir los datos siguientes en el sistema: Número, Subnúmero, Descripción, Unidad de Medida, Periodicidad, Día de Captación, los cuales la aplicación debe identificar como datos obligatorios. Los demás datos del modelo son variante, código de la variante y descripción de la variante.

Modificar modelo estadístico.

Cuando los datos de los modelos se modifiquen el sistema debe comprobar que los nuevos datos introducidos no tienen errores.

Eliminar modelo estadístico.

Cuando se elimina un modelo estadístico el sistema debe notificarlo al usuario.

Exportar modelo estadístico.

El sistema debe ser capaz de exportar los modelos estadísticos a formatos PDF, EXCEL o WORD.

Enviar modelo estadístico.

El sistema enviará los modelos estadísticos al servidor especificado.

Imprimir modelo estadístico.

El sistema debe imprimir el modelo especificado.

Crear cuadres del modelo estadístico.

Cuando se registra el código correspondiente al cuadre del modelo, el sistema debe compilarlos y dar la posibilidad de guardarlos.

Modificar cuadres del modelo estadístico.

Cuando se modifique el cuadre de un modelo, el sistema debe identificar los cambios y compilar los nuevos datos.

Eliminar cuadres del modelo estadístico.

Cuando se elimine un modelo el sistema debe notificarlo.

Imprimir cuadres.

El sistema debe imprimir el cuadre seleccionado.

Crear Nomenclatura

Para la creación de las nomenclaturas el sistema debe identificar los datos que tienen carácter obligatorio, los cuales son: Tipo de indicador, código del indicador, otros indicadores, unidad de medida.

Modificar Nomenclaturas

Cuando se modifiquen las nomenclaturas el sistema debe ser capaz de verificar que los datos modificados no contengan errores.

Eliminar Nomenclatura

Cuando se elimine una nomenclatura el sistema debe notificarlo.

2.1.3 Definición del estilo arquitectónico utilizado para los Módulos Entrada de Datos y Generador de Modelos.

Los estilos arquitectónicos están tomando mayor importancia en el desarrollo de los sistemas a medida que pasa el tiempo, uno de los estilos arquitectónicos que más se utilizan hoy en día es el estilo en capas el cual fue explicado en el capitulo 1. En esta arquitectura las interfaces entre las capas están bien definidas, además de ser una arquitectura muy favorable a la hora de sustituir algún componente. Las dependencias en esta arquitectura se limitan a la parte interna de las capas. Permite abstraerse completamente del origen de los datos, porque cada capa realiza sus funcionalidades, y no importa cual es el origen de los datos que se utilizan para que la aplicación funcione. Esta arquitectura brinda la posibilidad de la reutilización de código, porque los componentes desarrollados en la misma pueden ser utilizados en la misma aplicación o por otras aplicaciones futuras. Es segura porque los desarrolladores que se ocupan de la interfaz de usuario no podrán modificar nada en el servidor, ya que todo lo hacen a través de componentes. Representa el sistema con un alto nivel de abstracción, lo que permite tomar decisiones de diseño, que son muy importantes para la posterior evolución del sistema.

Los Módulos Generador de Modelos y Entrada de Datos poseen un estilo arquitectónico en capas, basado en componentes y orientado a objetos, este estilo se representa a continuación.

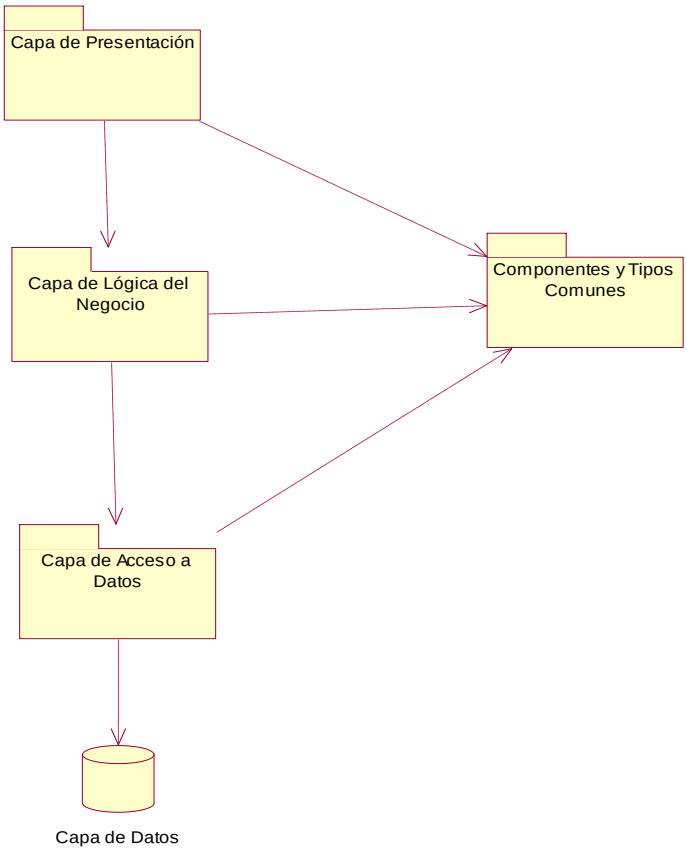


Figura 5 Estilo arquitectónico de los Módulos Entrada de Datos y Generador de Modelos.

La capa de Presentación es la capa que se encarga de la interacción con los usuarios que van a utilizar las aplicaciones, en este caso serian los estadísticos del país. En esta capa se va a encontrar todo lo relacionado con la Interfaz de usuario de ambos módulos. Ella recoge la información que solicita el usuario y la envía a la capa de Lógica del Negocio. Además es la encargada de darle una respuesta al usuario, después de haber recibido lo que necesita de la capa inferior para ello.

En la capa de Lógica del Negocio se implementan todas las reglas del negocio. Esta capa recibe información de la capa de Presentación, hace la petición a la capa de Acceso a Datos y da una respuesta de lo que le solicitaron con antelación.

La capa de Acceso a Datos, es la encargada de acceder a los datos que se encuentran en el repositorio de información, para la generación de esta capa se utiliza el TierDeveloper, la capa de persistencia que este genera está compuesta por tres clases principales:

- Entity
- Collection
- Factory

La clase factory se utiliza para llamar a las consultas y los procedimientos almacenados que se implementan en la base de datos.
La Entity se utiliza para llamar a las entidades del negocio.
La collection para llamar una colección de entidades.
Esta capa entre otras funciones tiene la responsabilidad de mantener los datos y de asegurar la integridad de los mismos.

La capa de Datos representa la base de datos como tal, y en ella se encuentran los datos reales con los que se va a trabajar en las aplicaciones, los datos acerca de los modelos, indicadores, aspectos, usuarios, entre otros.

Los componentes y tipos comunes, son un conjunto de funcionalidades que los sistemas necesitan para su trabajo y que pueden ser utilizados por las capas indistintamente. Aquí no hay más que clases y métodos de utilidad para los módulos.

2.2 Descripción de la arquitectura

La descripción de la arquitectura se obtiene a partir de los diferentes modelos que se desarrollan en el ciclo de desarrollo del software, los cuales son: modelos de casos de uso, de análisis, de diseño, de implementación y de despliegue. Esta descripción es un conjunto de vistas que contienen los elementos arquitectónicamente significativos de cada modelo, en ella se especifican casos de uso, componentes,

nodos, subsistemas y clases. La descripción de la arquitectura debe mantenerse actualizada con los cambios que se realicen en el sistema durante su ciclo de vida.

2.2.1 Casos de uso arquitectónicamente significativos.

Los casos de uso van guiando la arquitectura de software durante todo el ciclo de vida del proyecto, estos son los que responden a las funcionalidades del sistema por tanto a partir de ellos es que se sabe como va a ser la interacción del usuario con el sistema. Los casos de uso arquitectónicamente significativos se realizan en las primeras iteraciones del proyecto y luego se van desarrollando los demás, por lo que es muy importante seleccionar los casos de uso críticos adecuadamente para que a partir de ellos se puedan centrar los cimientos de la arquitectura de software. A partir de los casos de uso más importantes, también se puede definir cuales de estos, son los que necesitan mayor rendimiento y capacidad de hardware para su funcionamiento dentro del sistema. Existen varias vistas de la arquitectura planteadas por la metodología RUP, una de ellas es la vista de la arquitectura del modelo de casos de uso. Esta vista está compuesta por los casos de usos y los actores más significativos en el sistema.

2.2.1.1 Casos de Uso arquitectónicamente significativos para el Módulo Entrada de Datos.

En este módulo hay cuatro casos de uso arquitectónicamente significativos, estos son: Realizar modelo, Administrar modelo, Validar modelo y Autenticarse. Ver anexo 1 donde se muestra el diagrama de casos de uso del sistema completo.

- Realizar modelo: tiene la responsabilidad de realizar todas las acciones asociadas a la digitación de los modelos, cargando el modelo en formato XML y luego introduciendo los datos estadísticos que pertenecen al mismo, este caso de uso lo inicia el digitador. Para obtener más información del caso de uso ver anexo 3.
- Administrar modelos: en este se realizan todas las acciones referentes al control de los modelos como insertar y eliminar los mismos, el caso de uso es iniciado por el administrador. Para obtener más información del caso de uso ver anexo 4.
- Administrar usuarios: se llevan a cabo ciertas funcionalidades como el control de usuarios, la asignación de cierto nivel de acceso, así como privilegios, este caso de uso es iniciado por el

administrador, el que puede adicionar o eliminar usuarios y asignarle los roles. Para obtener más información del caso de uso ver anexo 5.

- Autenticarse: permite que el usuario acceda solamente a la parte del sistema que le corresponde, para ello debe introducir su usuario y contraseña, para verificar si tiene permisos en el sistema o no. Para obtener más información del caso de uso ver anexo 6.

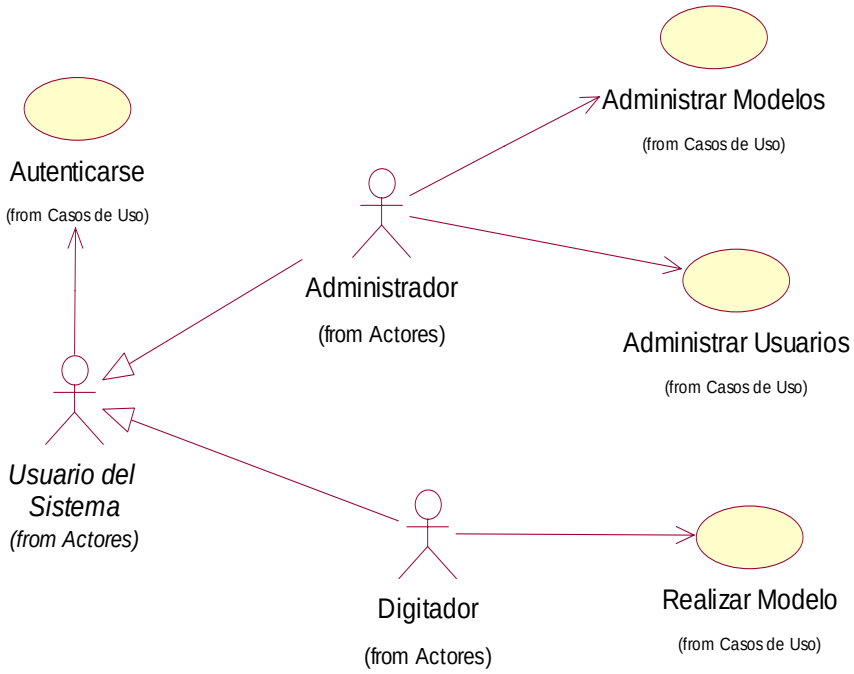


Figura 6 Diagrama de casos de uso arquitectónicamente significativos para el Módulo Entrada de Datos.

2.2.1.2 Casos de Uso arquitectónicamente significativos para el Módulo Generador de Modelos.

Para este módulo hay tres casos de uso significativos para la arquitectura, estos son:

- Gestionar modelos: el cual administra todos los procesos relacionados con la creación, modificación y eliminación de los modelos, este caso de uso es iniciado por el especialista de dirección estadística, para obtener más información del caso de uso ver anexo 7.
- Gestionar Nomenclatura: se controlan todos los procesos relacionados con la creación, modificación o eliminación de las nomenclaturas de los modelos, es iniciado por el especialista de dirección estadística, es un caso de uso extendido del caso de uso Gestionar modelos. Por que este se realiza si el usuario decide que su modelo tiene nomenclatura. Para obtener más información del caso de uso ver anexo 8.
- Gestionar Cuadres: administra todos los procesos relacionados con la creación, modificación o eliminación de los cuadres de los modelos, el caso de uso es iniciado por el especialista de dirección estadística y tiene una relación de inclusión con Gestionar modelos. Para obtener más información del caso de uso ver anexo 9.

Esta vista está representada por un diagrama de casos de usos que constituye un subconjunto del modelo de casos de uso. Para observar el diagrama de casos de uso del sistema completo ver anexo 2.

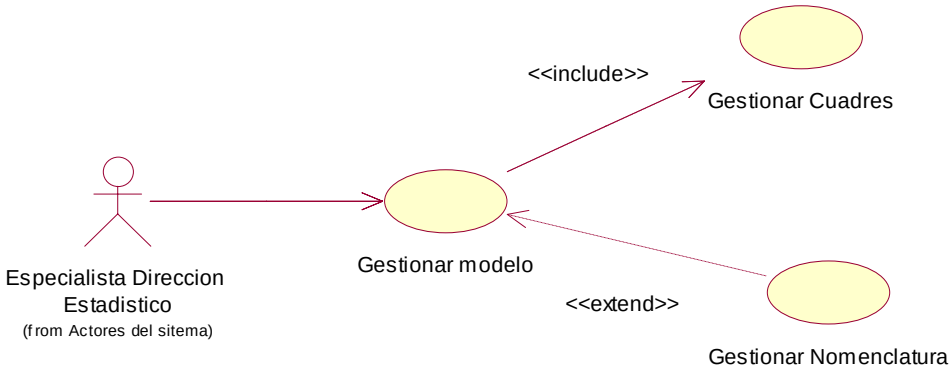


Figura 7 Diagrama de Casos de Uso arquitectónicamente significativos para el Módulo Generador de Modelos.

2.2.2 Vista de la arquitectura del modelo de análisis.

Uno de los modelos que se hace necesario plasmar en la descripción de la arquitectura es el modelo de análisis, el mismo queda representado mediante una vista que contiene las clases, paquetes y colaboraciones más importantes para la arquitectura. Las clases del análisis se dividen en tres tipos:

- Clases de Interfaz: la cual representa la interacción de los actores con el sistema, para esta comunicación utilizan ventanas y formularios, entre otros elementos.
- Clases de Control: representa la coordinación entre las clases interfaz y clases entidad, realizan todas las acciones y comportamiento de los casos de uso y desarrollan funciones complejas.
- Clases de Entidad: almacenan información que es persistente y el comportamiento asociado a la misma, se obtienen a partir de los objetos del negocio y del glosario de términos. (Jacobo, Booch y Rumbaugh, 2004).

2.2.2.1 Vista de la arquitectura del modelo de análisis para el Módulo Entrada de Datos.

Paquetes de análisis más importantes para la arquitectura del Módulo Entrada de Datos.

El Módulo Entrada de Datos se divide en dos paquetes del análisis y los dos son de gran importancia para la arquitectura, estos son: el paquete de las Funcionalidades Principales, el cual contiene dentro los casos de uso Autenticarse y Realizar Modelo, que son críticos para la arquitectura. Además esta el paquete de Administración, el cual contiene dentro el caso de uso Administrar Modelos y Administrar usuarios, que también son críticos. Cada uno de los casos de uso encierran dentro las clases del análisis correspondientes.

Figura 8 Paquetes del análisis del Módulo Entrada de Datos.

Clases de análisis más importantes para la arquitectura del Módulo Entrada de Datos.

Para el caso de uso Administrar Modelos, las clases del análisis son las siguientes.

IUAdministrarModelo: esta representa la interfaz con la que el usuario se va a comunicar con el sistema, para este caso de uso.

CGestorModelo: se encarga de implementar la adición, actualización y eliminación de los modelos.

CGestorOperaciones: clase controladora que realiza todas las operaciones relacionadas con las acciones que hacen los usuarios en el sistema, esta clase pertenece a la seguridad del sistema.

EModelo: contiene la información correspondiente a los modelos que están disponibles para la aplicación.

EOperacion: clase entidad que contiene todos los datos relacionados con una operación, entiéndase por esto que hizo, a que hora, la fecha y a que elementos afecto la operación entre otros elementos.

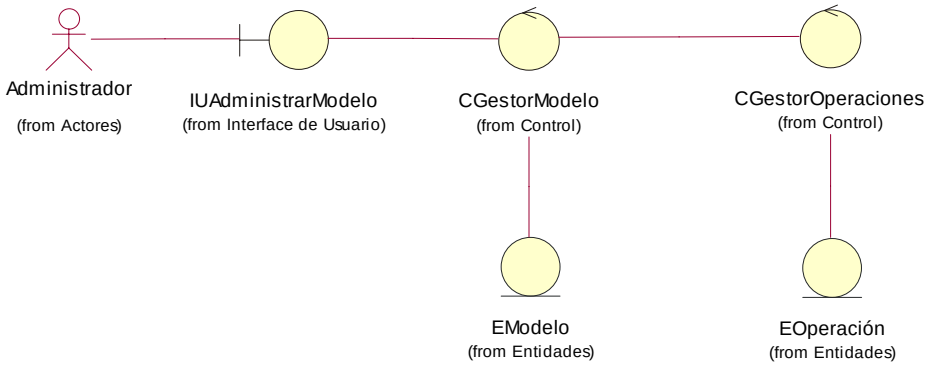


Figura 9 Diagrama de clases del análisis para el caso de uso Administrar Modelo.

Las clases del análisis que corresponden al caso de uso Autenticarse son las siguientes:

IUAutenticarse: representa la interfaz de usuario correspondiente a este caso de uso.

CGestorUsuario: implementa acciones relacionadas con el chequeo de los usuarios en el sistema.

CGestorOperaciones: clase controladora que realiza todas las operaciones relacionadas con las acciones que hacen los usuarios en el sistema, esta clase pertenece a la seguridad del sistema.

EUsuario: contiene información correspondiente a los usuarios, como nombre, contraseña y los roles que interpreta en el sistema.

EOperacion: clase entidad que contiene todos los datos relacionados con una operación, entiéndase por esto que hizo, a que hora, la fecha y a que elementos afecto la operación entre otros elementos.

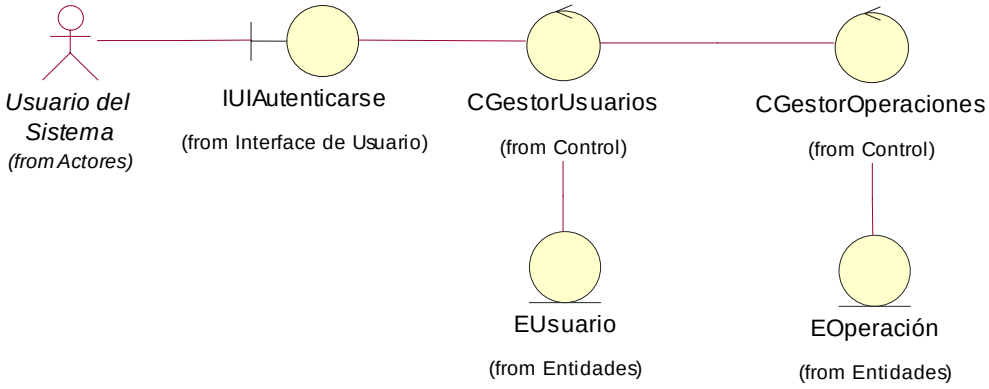


Figura 10 Diagrama de clases del análisis para el caso de uso Autenticarse.

Para el caso de uso Realizar Modelos se identifican las siguientes clases del análisis:

UIExploradorModelos: representa la interfaz que nos permite ver los diferentes modelos que han sido instalados en el sistema con su base metodológica correspondiente.

UIRealizarModelo: representa la interfaz a través de la cual se van a llenar los modelos.

CGestorModelo: se encarga de implementar la adición, actualización y eliminación de los modelos.

CGestorOperaciones: clase controladora que realiza todas las operaciones relacionadas con las acciones que hacen los usuarios en el sistema, esta clase pertenece a la seguridad del sistema.

EModelo: contiene la información correspondiente a los modelos que están disponibles para la aplicación.

EModeloDatos: contiene la información correspondiente a los modelos con los datos estadísticos introducidos, disponibles para la aplicación actualmente.

EOperacion: clase entidad que contiene todos los datos relacionados con una operación, entiéndase por esto que hizo, a que hora, la fecha y a que elementos afecto la operación entre otros elementos.

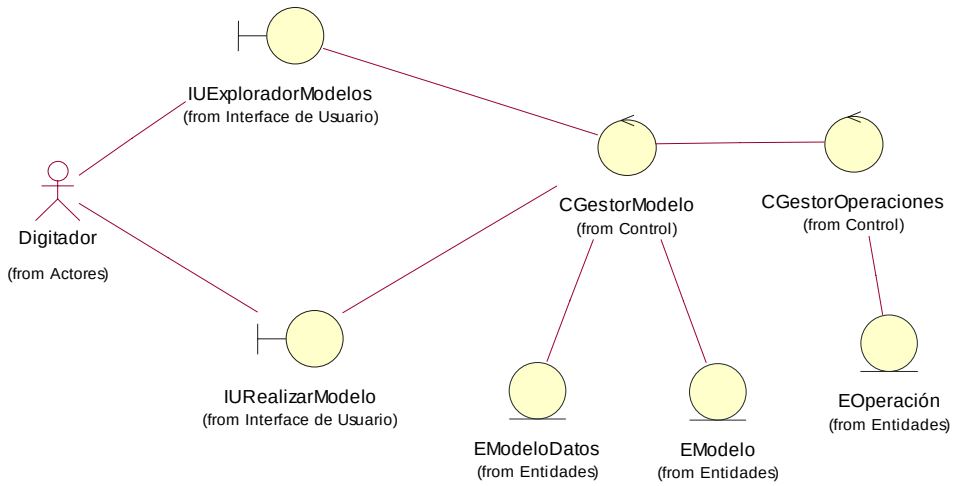


Figura 11 Diagrama de clases del análisis para el caso de uso Realizar Modelos.

Para el caso de uso Administrar Usuarios, las clases del análisis son las siguientes:

UIAdministrarUsuarios: mediante esta interfaz se pueden asignar roles e ingresar nuevos usuarios en el sistema.

CAdminUsuarios: se encarga de implementar acciones como adicionar, eliminar y editar datos de los usuarios, así como asignar roles a los mismos.

CGestorOperaciones: clase controladora que realiza todas las operaciones relacionadas con las acciones que hacen los usuarios en el sistema, esta clase pertenece a la seguridad del sistema.

ERol: contiene la información correspondiente a los roles, asociada a los permisos de los mismos.

EUsuario: contiene información correspondiente a los usuarios, como nombre, contraseña y los roles que interpreta en el sistema.

EOperacion: clase entidad que contiene todos los datos relacionados con una operación, entiéndase por esto que hizo, a que hora, la fecha y a que elementos afecto la operación entre otros elementos.

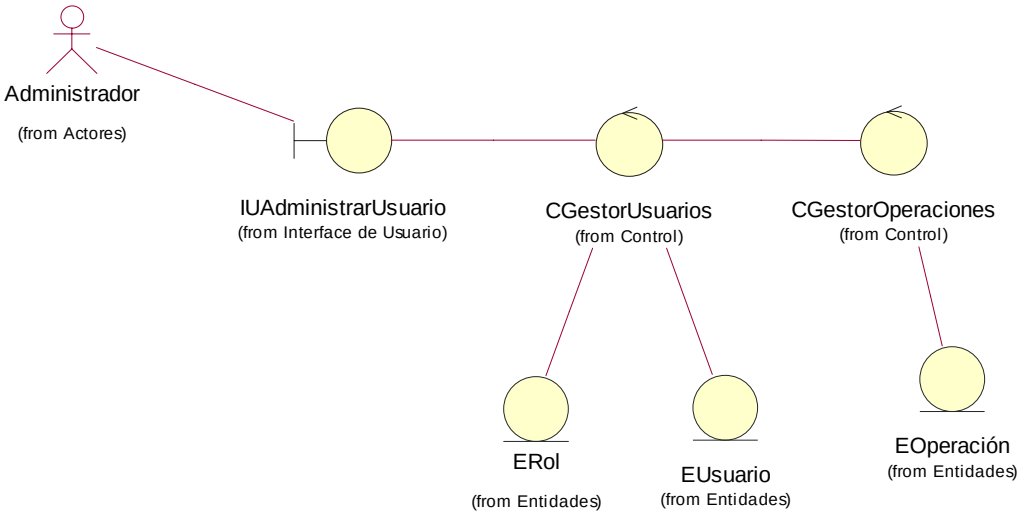


Figura 12 Diagrama de clases del análisis para el caso de uso Administrar Usuario.

2.2.2.2 Vista de la arquitectura del modelo de análisis para el Módulo Generador de Modelos.

Paquetes de análisis más importantes para la arquitectura del Módulo Generador de Modelos.

El Módulo Generador de Modelos está compuesto por varios paquetes del análisis, los más importantes para la arquitectura son los paquetes que encierran los casos de uso críticos, estos paquetes son:

- Paquete Gestionar Modelos, el cual tiene dentro el caso de uso Gestionar Modelos.
- Paquete Gestionar Nomenclatura, está compuesto por el caso de uso Gestionar Nomenclatura.
- Paquete Gestionar Cuadres, tiene dentro el caso de uso Gestionar Cuadres.

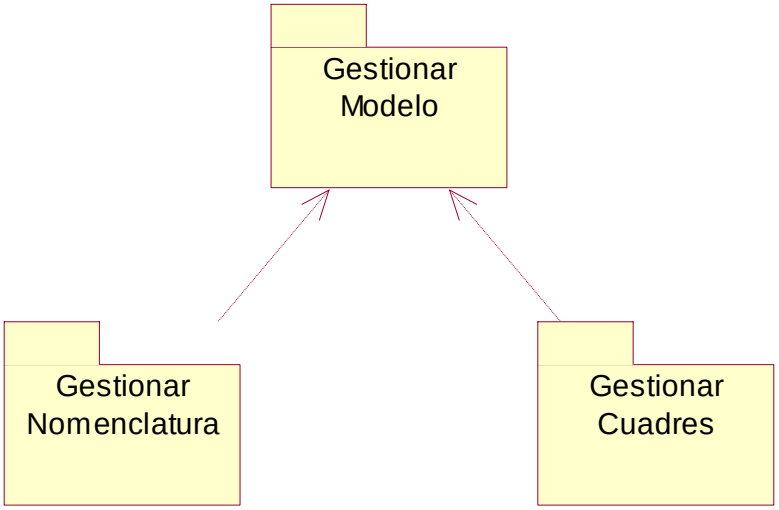


Figura 13 Paquetes del análisis del Módulo Generador de Modelos.

Clases de análisis más importantes para la arquitectura del Módulo Generador de Modelos.

Para el caso de uso Gestionar Modelos tenemos las siguientes clases del análisis:

UI_Crear_Modelos: representa la clase interfaz mediante la cual se introducen los datos para la creación de un nuevo modelo en el sistema.

UI_Modificar_Modelos: representa la clase interfaz mediante la cual se modifican y actualizan los datos de un modelo que se encuentre en el sistema.

CC_Gestor_Modelos: clase controladora que se encarga de realizar todos los procesos de búsqueda, muestra y actualización de los modelos.

CC_Crear_Modelo: clase controladora encargada de realizar las acciones de búsqueda, muestra y creación de los modelos.

CE_Modelos: clase entidad que contiene los datos relacionados con los modelos.

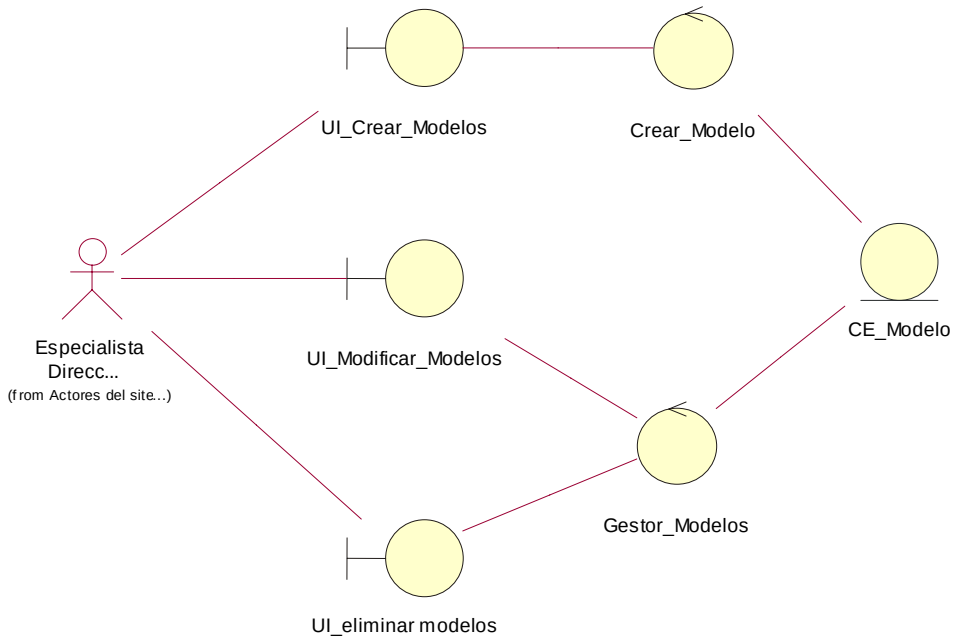


Figura 14 Diagrama de clases del análisis para el caso de uso Gestionar Modelos.

Para el caso de uso Gestionar Cuadres se tienen las siguientes clases del análisis.

UI_Crear_Cuadres: representa la interfaz mediante la cual se introducen los datos para la creación de un nuevo cuadre en el sistema.

UI_Modificar_Cuadres: representa la interfaz mediante la cual se pueden modificar los datos de los cuadres del sistema.

UI_Eliminar_Cuadres: representa la clase interfaz mediante la cual se puede eliminar un cuadre del sistema.

CC_Crear_Cuadres: clase controladora que realiza todas las acciones para la creación, búsqueda y muestra de los cuadres existentes en el sistema.

CC_Gestor_Cuadres: clase controladora que realiza todos los procesos de actualización, eliminación, búsqueda y muestra de los cuadres existentes en el sistema.

CE_ Cuadre: clase entidad que contiene los datos de un cuadre en el sistema.

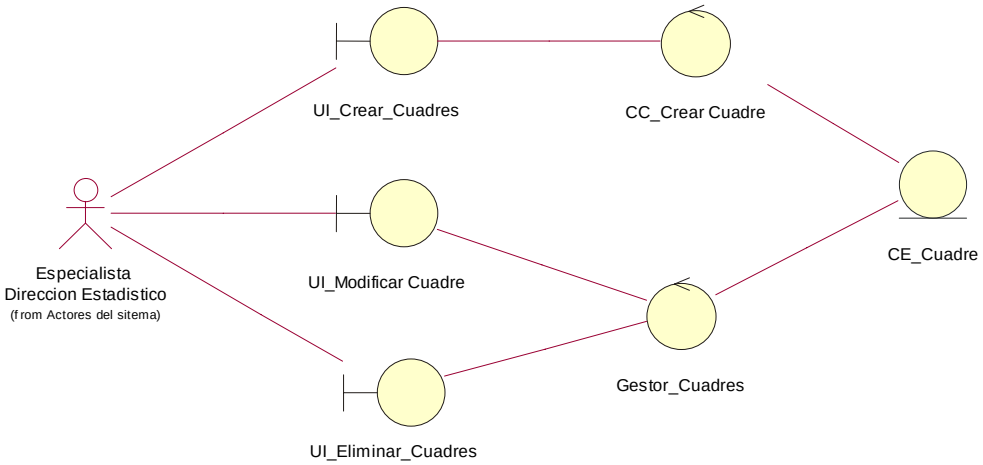


Figura 15 Diagrama de clases del análisis para el caso de uso Gestionar Cuadres.

En el caso de uso Gestionar Nomenclatura se encuentran las siguientes clases del análisis:

UI_Crear_Nomenclatura: clase interfaz a través de la cual se puede adicionar una nueva nomenclatura al sistema, con los datos necesarios para ello.

UI_Actualizar_Nomenclatura: clase interfaz mediante la cual se pueden modificar los datos de una nomenclatura que este en el sistema.

CC_Crear_Nomenclatura: clase controladora que realiza los procesos de creación, muestra y búsqueda de las nomenclaturas en el sistema.

CC_Gestor_Nomenclatura: clase controladora mediante la cual se realizan todos los procesos de actualización, búsqueda, muestra y eliminación de las nomenclaturas existentes en el sistema a través de las clases interfaces.

CE_ Nomenclatura: clase entidad que contiene los datos de una nomenclatura del sistema.

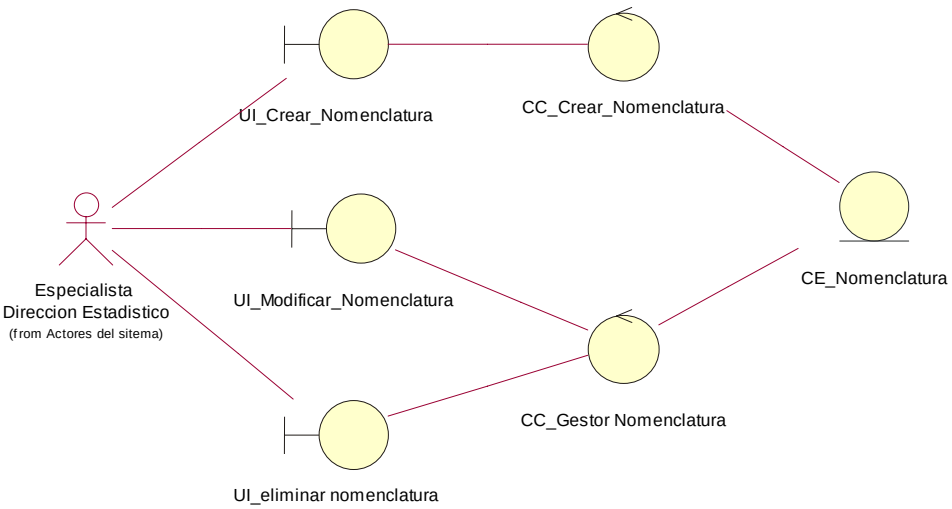


Figura 16 Diagrama de clases del análisis para el caso de uso Gestionar Nomenclatura.

2.2.3 Vista de la arquitectura del modelo de diseño.

Esta vista es muy importante para la arquitectura de los sistemas porque destaca los subsistemas, sus relaciones, las interfaces y las clases del diseño que se deben implementar primeramente, porque son más necesarias que las demás, su importancia es vital para que los sistemas funcionen, y muchas otras clases dependen de estas. Esta vista queda plasmada en el documento descripción de la arquitectura. Muchas de las clases que se desarrollan en el diseño tienen una traza con las del análisis y además están muy relacionadas con los casos de uso. Las clases del diseño deben aparecer ya con los atributos y los métodos que se van a implementar. En el diseño aparecen clases nuevas ya que se aplican restricciones del diseño, como por ejemplo se comienzan a establecer patrones de diseño, entre otras.

2.2.3.1 Vista de la arquitectura del modelo de diseño para el Módulo Entrada de Datos.

Subsistemas del diseño arquitectónicamente significativos para el Módulo Entrada de Datos.

El Módulo Entrada de Datos está dividido en dos subsistemas, los cuales son muy importantes en la arquitectura del sistema, estos son el Subsistema Principal, que contiene dentro todo lo relacionado con los casos de uso Realizar Modelo y Administrar Modelo y el Subsistema de Seguridad que se encarga de todo lo que tiene que ver con la Administración de permisos a los usuarios y la definición de los roles en los sistemas. Los elementos de cada uno de estos subsistemas se encuentran fuertemente relacionados.

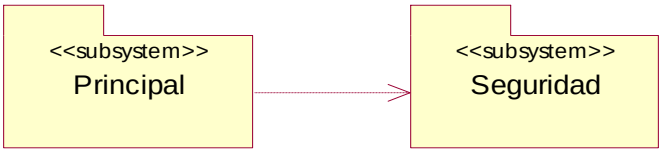


Figura 17 Subsistemas del diseño del Módulo Entrada de Datos.

Clases del diseño importantes para la arquitectura del sistema correspondiente al Módulo Entrada de Datos.

Para el subsistema Principal las clases del diseño más importantes son:

Modelo: clase contenedora de datos, se puede tracear hasta la entidad Modelo en el análisis. Esta clase contiene los siguientes atributos número del modelo, descripción, estado y clasificador de empresa.

EncabezadodeModelo (User Control): control visual, para introducir los datos del encabezado del Modelo.

EncabezadodeModeloMVC: clase controladora visual. Implementación del Patrón Modelo Vista Controlador. Responsable de atender la lógica visual del Encabezado y su relación con la clase controladora del negocio. Entre los principales métodos que se implementan en esta clase están ActualizarEncabezadodeModelo(), GuardarInformacionModelo(), VerificarCentroInformante() y MostrarClasificadores(), entre otros.

Digitador (User Control): control visual, para la digitación de los modelos. Está compuesto por un conjunto de controles Celda.

DigitadorMVC: clase controladora visual. Implementación del Patrón Modelo Vista Controlador. Responsable de atender la lógica visual del Digitador y su relación con la clase controladora del negocio.

Entre los principales métodos que se implementan en esta clase están: EliminarFilaActualdeModelo(), GuardarModeloenDigitacion(), GuardarModeloListoparaValidar() y LimpiarDigitador(), entre otros.

GestorModelo: realiza todas las operaciones relacionadas con los modelos, ya sea adicionar, modificar o eliminar los mismos. Entre las principales acciones que realiza esta clase están: DescargarModelo(), VerificarIntegridadModelo(), InstalarModeloPlantilla(), ActualizarModeloPlantilla() y ReportarModelosPlantillas, entre otras.

GestorModeloAccesoXML: gestiona todas las operaciones para garantizan la persistencia de modelos a nivel del sistema de archivos. En esta clase se implementan acciones como CargarModeloXML(), EliminarModeloEstadistica(), GuardarModelo(), entre otros.

GestorModeloAccesoBD: gestiona todas las operaciones que garantizan la persistencia de los archivos a nivel de base de datos. Dentro de los principales métodos que se implementan en esta clase están ActualizarModeloEstadistico(), CargarModeloEstadistico(), EliminarModelosEstadistico(), InstalarModeloEstadistico(), entre otros.

RegistroModelo: Es la entidad básica que compone el registro central de modelo, cada modelo tiene asociado un registro de modelo que se actualiza cada vez que este es modificado. Los atributos de esta clase son número de modelo, dirección y estado. Esta clase posee el método RegistroModelo() dentro.

RegistroCentralModelos: Es el catálogo que contiene toda la información de los archivos procesados por la aplicación.

Para el subsistema de seguridad las clases del diseño más importantes son:

AdministrarUsuario (User Control): control visual a través del cual se pueden adicionar nuevos usuarios al sistema, especificando el rol que juega el mismo.

Autenticar (User Control): control visual que se utiliza para la autenticación de los usuarios en el sistema.

GestordeUsuarios: realiza las operaciones relacionadas con los usuarios. En esta clase hay métodos como Autenticarse(), ExisteUsuario(), VerificarClave(), AdicionarUsuario(), EliminarUsuario(), CambiarClave(), entre otros.

GestordeOperaciones: clase que gestiona las operaciones realizadas por los usuarios. Esta clase tiene métodos como: RegistrarOperacion() y SolicitarOperacion(), entre otras.

PerfildeUsuario: esta clase guarda la información del usuario que está autenticado en ese momento y dice los permisos que tiene el mismo. Entre los principales métodos que se encuentran implementados en esta clase están: UsuarioActual(), EsRol(), TienePermiso() y UsuarioAutenticado(), entre otros.

Operación: clase que contiene los datos de las operaciones que realiza el usuario. Los principales atributos de esta clase son: fecha, evento, usuario y descripción.

Permiso: clase que contiene los datos de los permisos de los usuarios en el sistema. Los principales atributos son nombre del permiso, lista de dominios.

Rol: clase que contiene todo lo relacionado con los roles de los usuarios en el sistema. Los principales atributos de esta clase son nombre del rol, lista de permisos.

Usuario: clase que contiene los datos de los usuarios del sistema. Los atributos de esta clase son los siguientes: clave, nombre, nombre de usuarios, roles.

2.2.3.2 Vista de la arquitectura del modelo de diseño para el Módulo Generador de Modelos.

Subsistemas del diseño arquitectónicamente significativos para el Módulo Generador de Modelos.

El Módulo Generador de Modelos posee varios subsistemas del diseño, posee el subsistema Principal, que está compuesto por los subsistemas Gestionar Modelos, Enviar Modelos y Salidas del sistema, de todos estos subsistema el más significativo para la arquitectura es el de Gestionar Modelos.

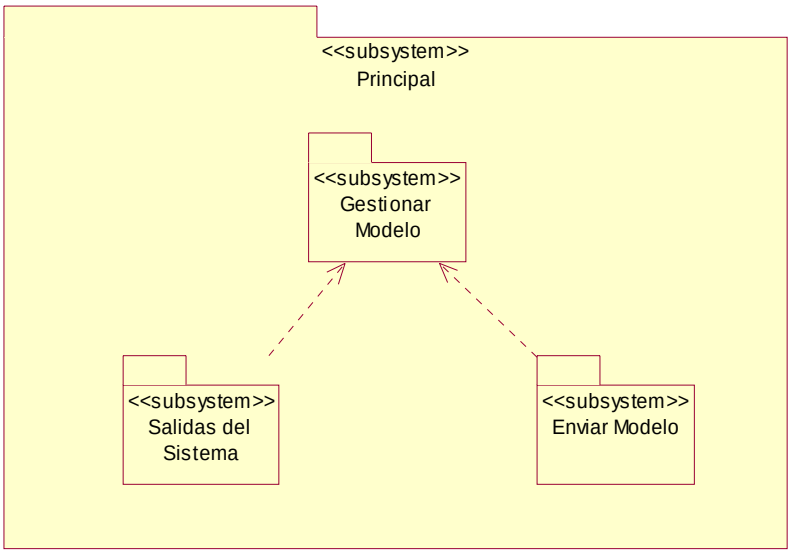


Figura 18 Subsistema del diseño del Módulo Generador de Modelos.

Clases del diseño importantes para la arquitectura del sistema correspondiente al Módulo Generador de Modelos.

Clases del diseño más importantes para el subsistema Principal.

frmPrincipal: formulario que se encarga de la gestión del Menú Principal y da la posibilidad de trabajar con varios modelos al mismo tiempo.

ControlCrearAspectos (User Control): control que permite toda la gestión de los aspectos.

ControlCrearIndicadores (User Control): control que permite toda la gestión de los indicadores.

ControlCrearModeloDatos (User Control): control que permite captar los datos generales del modelo, con su periodicidad y variante.

ColeccionDePeriodicidades: Es la clase que contiene la lista de periodicidades que puede tener un modelo.

ColecciondeModelos: Es la clase que contiene la lista de los modelos**.**

ColecciondeAspectos: Es la clase que contiene la lista de los Aspectos que tiene un modelo.

ColeccíondeIndicadores: es la clase que contiene la lista de los indicadores de un modelo.

GestordeModelo: esta clase es la que controla todo lo que tiene que ver con los modelo, algunos de los métodos que implementa son: GuardarModelo(), ActualizarModelo(), CargarModelo(), ConstruirParteDatosGenerales(), ConstruirParteIndicadores(), ConstruirParteAspectos y EliminarModelo(), entre otros.

GestordeTiposdeAspectos: clase que controla lo relacionado con los Aspectos de los modelos. Dentro de los principales métodos de esta clase están: BuscarPosiciondeTipodeAspecto(), NiveldeTipodeAspecto(), Nuevo() y EliminarA(), entre otros.

GestordeTiposdeIndicadores: clase que controla lo relacionado con los Indicadores de los modelos. Dentro de los principales métodos de esta clase están: Nuevo(), BuscarPosiciondelTipodeIndicador(), NiveldelIndicador(), EliminarIndicador() y ActualizarNumerodeFilaAsociada(), entre otros.

Modelo: Es la clase que posee los atributos generales necesarios para la creación de un modelo. Esta clase es abstracta, contiene dentro atributos como clasificación, descripción del modelo, numero del modelo y periodicidad. Entre las principales operaciones que realiza esta clase están Modelo(), Clasificación(), descripción(), Periodidad mínima(), entre otros.

ModeloDeEstadisticaContinua: Esta clase hereda de la clase Modelo y contiene los atributos específicos de los modelos de estadística continua. Esta clase es abstracta. Esta clase tiene atributos dentro como informe acumulado, suma de control, unidad de medida, variante 1 y variante 2. Esta clase tiene algunos métodos asociadas para su trabajo como ColeccionPaginas(), Informeacumulado(), SumaControl(), Variante1(), Variante2(), entre otros.

ModeloTalonAbierto: Clase que hereda de **ModeloDeEstadisticaContinua** y contiene las funcionalidades especificas de un Modelo de Talón Abierto.

ModeloTalonCerrado: Clase que hereda de **ModeloDeEstadisticaContinua** y contiene las funcionalidades especificas de un Modelo de Talón Cerrado.

TipodeAspecto: Clase que contiene las características de los tipos de aspectos. Entre los atributos de esta clase están: código, descripción, colección de aspectos.

Aspecto: Clase que contiene las características de los Aspectos. Entre los atributos de esta clase están Cantidad de decimales y numero de columna asociada. Dentro de las funcionalidades que realiza esta clase están: Aspecto(), Cantidaddecimales() y NumdeColumnaAsociadas.

TipoDeIndicador: Clase que contiene las características de los tipos de indicador. Entre los principales atributos de esta clase se encuentran: colección de indicadores, nombre y código.

Indicador: Clase que contiene las características de los indicadores como son clasificación, numero de fila asociada, periodicidad, tipo de indicador y unidad de medida. Entre los principales métodos están Indicador(), NumFilaAsociada(), Periodicidad(), TipoIndicador(), UnidadMedida().

Celda: Clase que almacena los datos estadísticos digitados en un modelo.

Periodicidad: clase que contiene las características de las periodicidades, en esta clase están los atributos descripción y fecha de partida. Los métodos de la clase son Periodicidad(), Descripción() y FechadePartida().

2.2.4 Componentes de los sistemas

La vista de la arquitectura del Modelo de Implementación, destaca los componentes del sistema que son relevantes para la arquitectura, esto influye en este flujo de trabajo porque esos componentes seleccionados son necesarios para iniciar así la implementación del sistema.

El Módulo Entrada de Datos está compuesto por dos subsistemas, el subsistema de Seguridad y el subsistema Principal, estos dos subsistemas son significativos para la arquitectura del módulo. El Módulo Generador de Modelos está compuesto por un subsistema Principal que tiene dentro varios subsistemas pero el más importante para la arquitectura en el flujo de trabajo de implementación es el de Gestionar Modelos.

Estos dos módulos se encuentran divididos por el estilo en capas presentado anteriormente en la figura 5, cada una de las capas tiene varios componentes dentro y estos se relacionan entre si, estos componentes tienen dentro la implementación de las clases y métodos necesarios para que los sistemas funcionen.

2.2.4.1 Componentes del sistema del Módulo Entrada de Datos.

Los componentes relacionados con el Módulo Entrada de Datos son los siguientes:

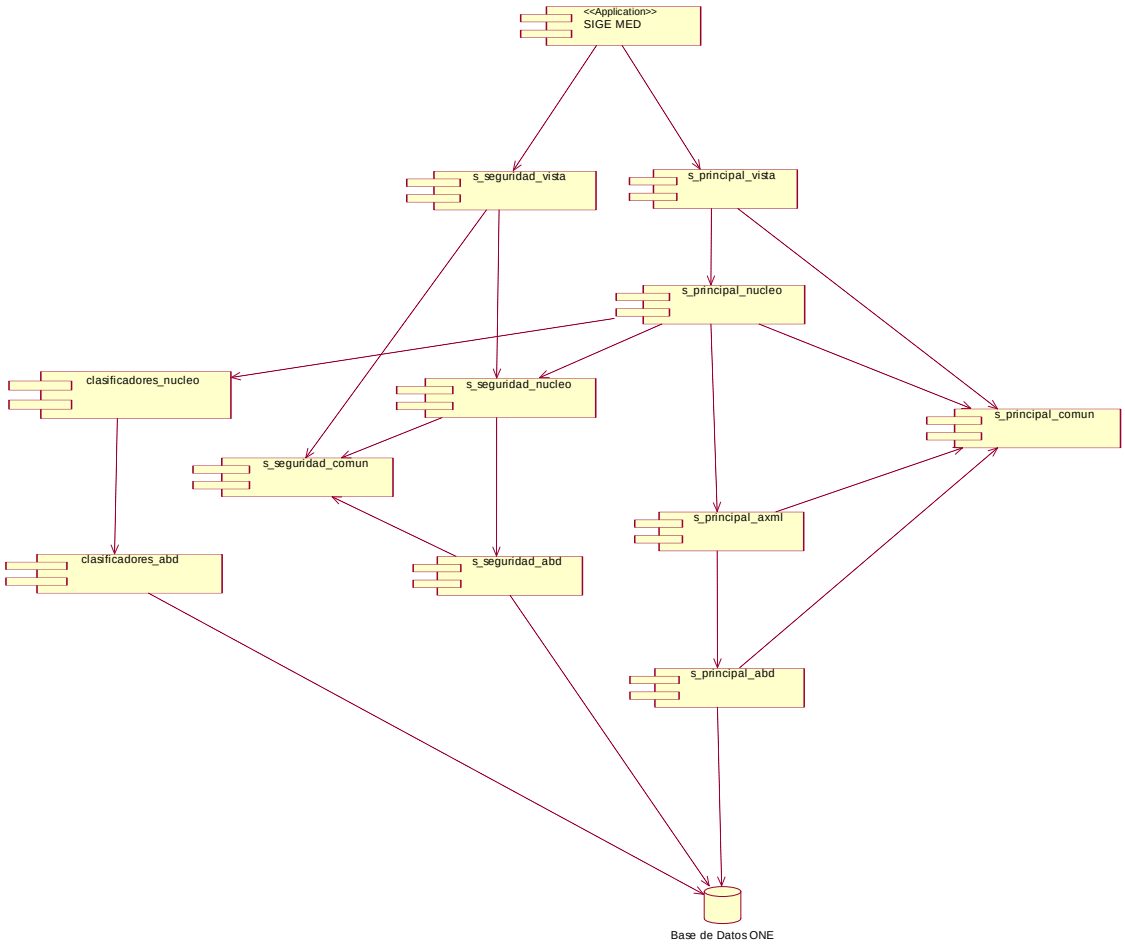
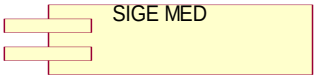
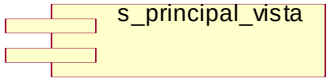


Figura 19 Diagrama de componentes del Módulo Entrada de Datos.

En la capa de Presentación están los siguientes componentes:

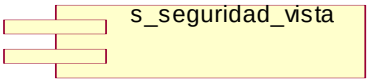


Este componente constituye el ejecutable de la aplicación, y está relacionado con los componentes s_principal_vista y s_seguridad_vista.



Este componente pertenece al subsistema Principal y se utiliza para establecer la conexión entre los usuarios y la aplicación, ya sea para realizar un modelo, para validar los

modelos u otras funcionalidades necesarias en este módulo. Este componente depende del componente ejecutable SIGE MED, que es a partir del cual la aplicación comienza a trabajar y se encuentra relacionado con los componentes s_principal_núcleo y s_principal_comun.



Este componente pertenece al subsistema de seguridad, es utilizado para la interacción del usuario con la aplicación a la hora de autenticarse, de establecer nuevos roles o cambiar la contraseña de alguno de los usuarios, entre otras funcionalidades que permite el sistema en cuanto a la administración de los usuarios y roles. Este componente al igual que el anterior depende del ejecutable SIGE MED y se relaciona con los componentes s_seguridad_comun y s_seguridad_nucleo.

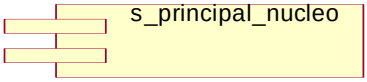
En esta capa se trabaja con algunos componentes que ya están implementados y se reutilizan en este sistema, estos son lo que se muestran a continuación:

- SandDock
- SharpLibrary

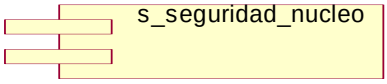
El componente SandDock, se utiliza para trabajar con el explorador de modelos, y agrupar herramientas que se utilizan para el trabajo en el módulo.

El componente SharpLibrary, librería de controles de usuario que posee la herramienta Microsoft Visual Studio .Net y se utiliza en el componente explorador de modelos perteneciente al componente s_principal_vista.

En la capa del negocio se encuentran los siguientes componentes:



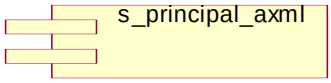
Este componente pertenece al subsistema principal y realiza las funcionalidades del sistema pertenecientes a la lógica del negocio, posee la clase gestora GestorModelos y la interfaz IGestorModelos. Este componente está relacionado con clasificadores_nucleo, con s_seguridad_nucleo, con s_principal_axml y con s_principal_comun, a su ves depende del componente s_principal_vista.



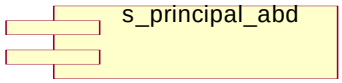
Este componente pertenece a el subsistema de seguridad y en el se encuentra implementada la lógica del negocio para este subsistema, por lo que se implementan las clases gestoras GestorOperaciones, GestorSeguridad y GestorUsuario, entre otras. Este componente depende

de los componentes s_seguridad_vista y de s_principal_nucleo y está relacionado con los componentes s_seguridad_comun y s_seguridad_abd.

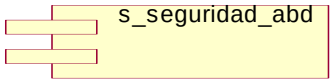
En la capa de acceso a datos se encuentran los siguientes componentes:



Este componente pertenece el subsistema Principal, garantiza la persistencia de los XML, mediante este componente es que el sistema permite entrar una colección de modelos que se encuentran guardados en la computadora. El depende de s_principal_nucleo y está relacionado con s_principal_abd y con s_principal_comun.



Este componente pertenece al subsistema Principal y a través del mismo podemos acceder a la base de datos para trabajar con los modelos, este componente depende del componente s_principal_axml y está relacionado con s_principal_comun y con Base de Datos ONE.

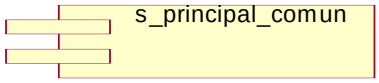


Este componente pertenece al subsistema de seguridad, y permite acceder a la base de datos para trabajar con tablas relacionadas con usuarios, roles y otras entidades que pertenecen a este subsistema. Este componente depende de s_seguridad_nucleo y está relacionado con los componentes s_seguridad_comun y Base de Datos ONE.

En la capa de los datos, está el siguiente componente:

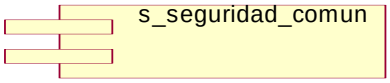
Este componente puede ser utilizado por los dos subsistemas y depende de los componentes clasificadores_abd, s_seguridad_abd y s_principal_abd. Este componente es la base de datos del proyecto donde se encuentra toda la información relacionada con los modelos, los clasificadores, los usuarios y otros elementos importantes para el proyecto.

En los componentes y tipos comunes está el siguiente componente:



Este componente pertenece al subsistema principal y tiene servicios comunes que son utilizados por los componentes s_principal_vista, s_principal_nucleo, s_principal_axml y s_principal_abd. También consta de un grupo de funcionalidades que pertenecen a los modelos y que

hacen falta para trabajar en este módulo, aquí se encuentran las clases relacionadas con los tipos de modelos, con los indicadores y aspectos que pertenecen a los modelos, entre otras. Hay además otros recursos que también están disponibles para ser utilizados por cualquier capa que lo necesite. Este componente utiliza un conjunto de clases que le hacen falta del Módulo Generador de Modelos, estas son: Aspecto, ColeccióndeModelos, Modelos, Indicador, ColeccióndeIndicador, entre otras.



Este componente pertenece al subsistema de seguridad, y posee las siguientes entidades dentro: dominio, operación, permiso, rol y usuario. En este componente hay un conjunto de clases y utilidades que son comunes para todas las capas, y que en cualquier momento se pueden utilizar sin ningún problema. Los componentes que pueden utilizarlo son los siguientes: s_seguridad_vista, s_seguridad_nucleo y s_seguridad_abd.

2.2.4.2 Componentes del sistema del Módulo Generador de Modelos.

El Módulo Generador de Modelos está compuesto por los siguientes componentes.

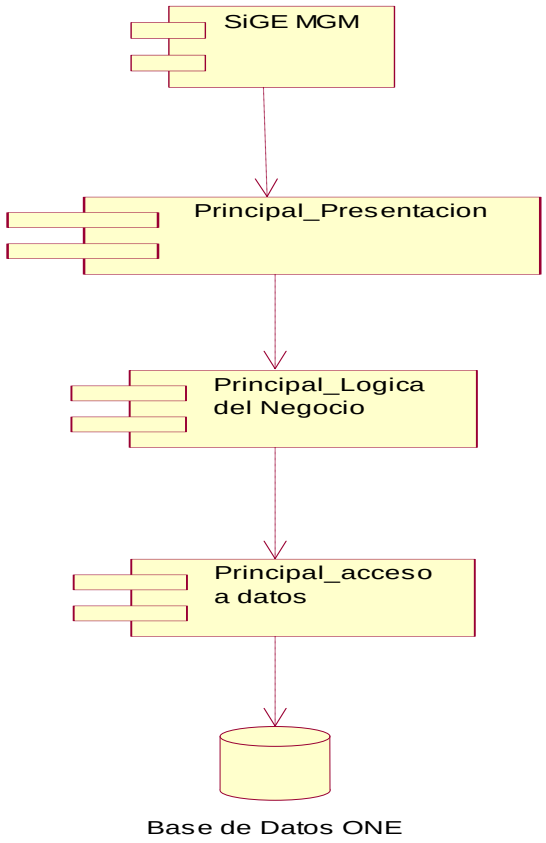
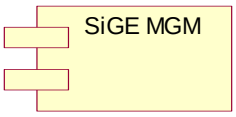
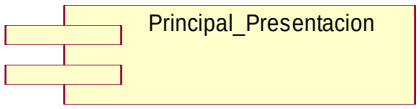


Figura 20 Diagrama de componentes del Módulo Generador de Modelos.

En la capa de presentación están los siguientes componentes:



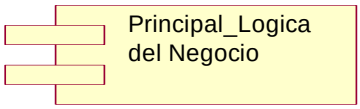
Este componente constituye el ejecutable de la aplicación, solamente posee dentro del un formulario, que va a permitir cargar los controles de usuario de la capa de Presentación. Esta relacionado con el componente Principal_Presentacion.



Este componente pertenece al subsistema Principal. Está compuesto por los controles de usuarios, que permiten la interacción del usuario con el sistema, a la hora de crear un nuevo modelo, de modificarlo o de eliminarlo. En este componente están los siguientes controles: ControlCrearAspecto, ControlCrearModelodeDatos y ControlCrearIndicadores, entre otros.

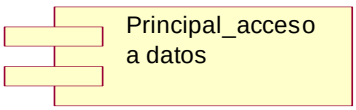
En esta capa también se utiliza el componente SandDock para controlar los documentos, porque en las oficinas de estadística se realizan varios modelos al mismo tiempo y este componente te muestra varias pestañas por las que te puedes mover en dependencia del modelo en que estés trabajando. Este componente está dentro de SIGE MGM también.

En la capa de Lógica del Negocio se encuentra el componente siguiente:



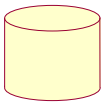
Este componente está compuesto por las clases ColecciondeAspectos, ColecciondeCuadres, ColecciondeIndicadores, ColecciondeModelos, y están las clases gestoras GestorModelos, GestordeCuadres, GestordeVariantel, GestordeVariantell, entre otras necesarias para la construcción de los modelos estadísticos. Este componente depende de Principal_ Presentación, y está relacionado con el componente Principal_acceso a datos.

En la capa de acceso a datos está el componente:



Este componente es el que permite que el sistema se pueda acceder a los datos, para trabajar con las tablas Indicador, Aspecto, para obtener la descripción real de los indicadores y de los aspectos que se introducen en el modelo cuando se está construyendo el mismo, entre otras tablas como modelo que también son necesarias para este módulo. Este componente depende de Principal _ lógica del negocio y está relacionado con la Base de Datos ONE.

En la capa de datos está el componente:



Base de Datos ONE

Este componente constituye la base de datos del proyecto ONE, aquí se encuentra toda la información necesaria para el trabajo en los módulos. Este depende del componente Principal_acceso a datos.

2.2.5 Despliegue de componentes de los sistemas.

Descripción de cómo se despliegan los componentes en las diferentes máquinas.

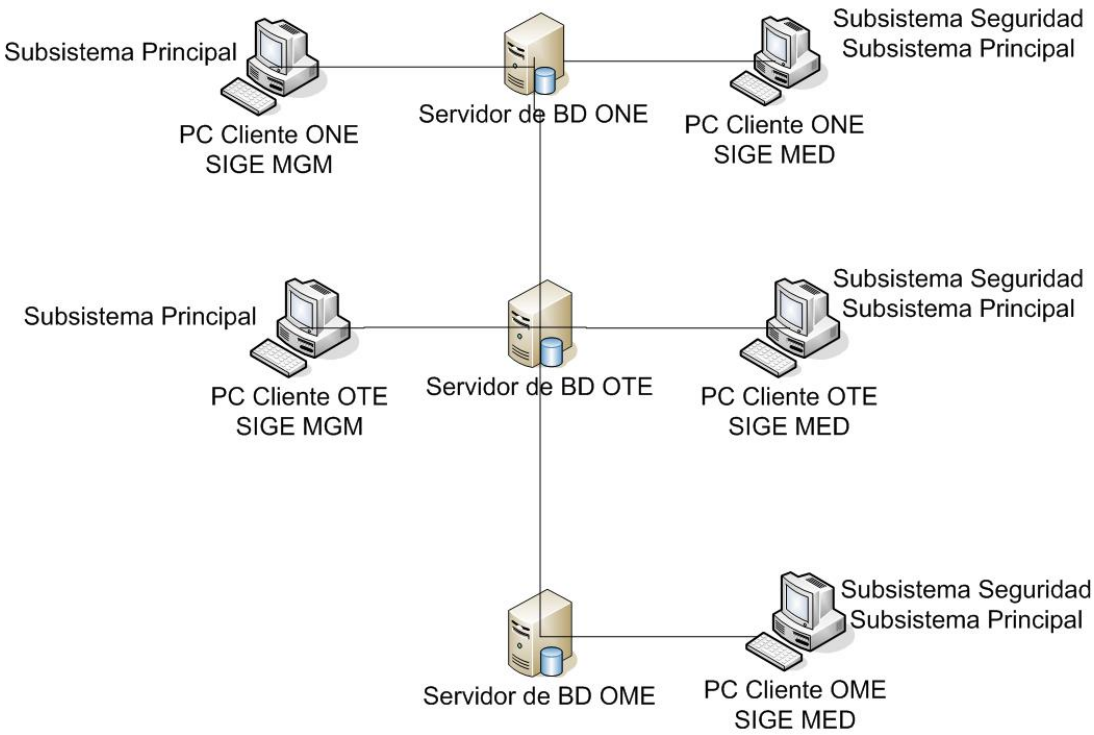


Figura 21 Despliegue de los sistemas en las distintas estaciones de trabajo.

El Módulo Entrada de Datos puede ser desplegado en la ONE, la OTE, la OME y en los centros informantes, aunque en estos últimos en dependencia de la tecnología que exista en los mismos es que se podrán instalar las funcionalidades que permite el sistema.

En la ONE van a estar instalados en las computadoras clientes los dos módulos, con todos los componentes que integran los subsistemas Principal y de Seguridad, además existe otra computadora que constituye el servidor de base de datos de la ONE, esta contiene el componente correspondiente a la base de datos de la ONE.

El Módulo Generador de Modelos genera un XML, que se envía al Módulo Entrada de Datos, y este lo utiliza para introducir los datos a los modelos ya creados con anterioridad y aprobados para ser llenados.

El servidor de base de datos de la ONE se conecta al servidor de base de datos de la OTE, y en este último está el componente Base de Datos.

En las máquinas clientes de la OTE estarán instalados los dos módulos también con los subsistemas Principal y de Seguridad. El servidor de base de datos de la OTE se conecta con el servidor de base de datos de la OME y en este se encuentra el componente base de datos.

En la OME se instala en las máquinas clientes solamente la aplicación del Módulo Entrada de Datos, aquí el Módulo Generador de Modelos no se instala porque en los municipios no se pueden crear modelos, aquí se llenan los modelos que se definieron en niveles más altos, como la ONE y la OTE.

2.2.5.1 Diagrama de despliegue para el Módulo Entrada de Datos.

En la vista de la arquitectura del modelo de despliegue se describen los principales nodos físicos que se necesitan para configurar la plataforma que puede soportar la implantación del sistema. Esta se obtiene en el flujo de trabajo de Análisis y Diseño y se perfecciona poco a poco, a medida que se vayan desarrollando las iteraciones. Esta vista se representa a través del siguiente diagrama de despliegue:

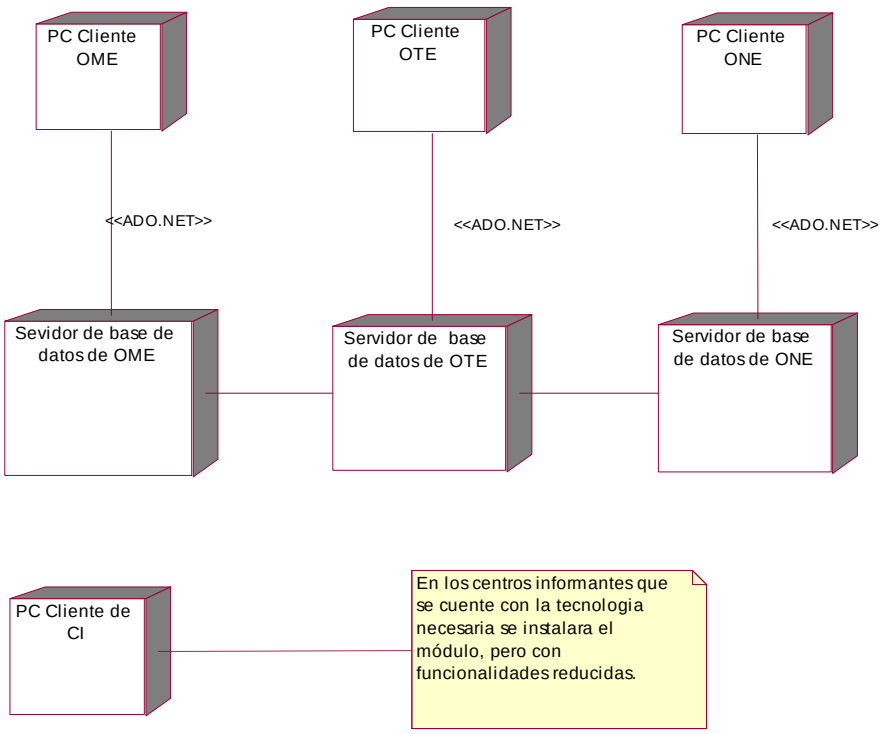


Figura 22 Diagrama de Despliegue del Módulo Entrada de Datos.

- Nodo PC Cliente de la OME: se encuentran los componentes necesarios para realizar todas las funcionalidades que el módulo requiere en el cliente de la OME.
- Nodo PC Cliente de la OTE: se encuentran los componentes necesarios para realizar todas las funcionalidades que el módulo requiere en el cliente de la OTE.
- Nodo PC Cliente de la ONE: se encuentran los componentes necesarios para realizar todas las funcionalidades que el módulo requiere en el cliente de la ONE.
- Nodo PC Cliente de los Centros Informantes: se encuentran los componentes necesarios para realizar las funciones del Módulo Entrada de Datos en los Centros Informantes.
- Nodo Servidor de Base de Datos de la OME: Contiene la Base de Datos donde se encuentran los datos correspondientes a los modelos de la OME.
- Nodo Servidor de Base de Datos de la OTE: Contiene la Base de Datos donde se encuentran los datos correspondientes a los modelos de la OTE.

- Nodo Servidor de Base de Datos de la ONE: Contiene la Base de Datos donde se encuentran los datos correspondientes a los modelos de la ONE.

2.2.5.2 Diagrama de despliegue para el Módulo Generador de Modelos.

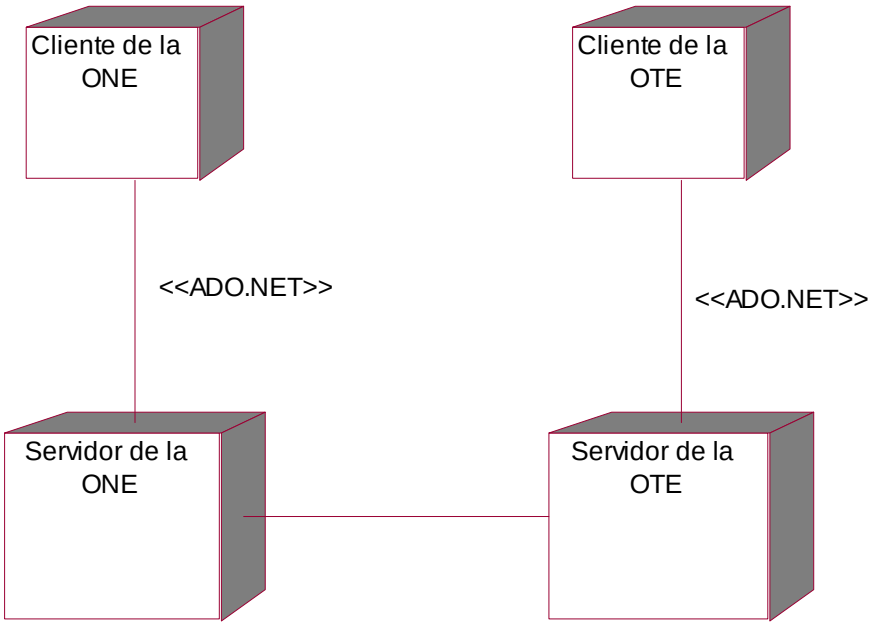


Figura 23 Diagrama de Despliegue del Módulo Generador de Modelos.

- Nodo Cliente de la ONE: Contiene los componentes necesarios para realizar todas las funcionalidades que el módulo requiere en el cliente de la ONE.
- Nodo Cliente de la OTE: Contiene los componentes necesarios para realizar todas las funcionalidades que el módulo requiere en el cliente de la OTE.
- Nodo Servidor de la ONE: Contiene la Base de Datos donde se encuentran los datos correspondientes a los modelos de la ONE.
- Nodo Servidor de la OTE: Contiene la Base de Datos donde se encuentran los datos correspondientes a los modelos de la OTE.

2.2.5.3 Recursos compartidos por las aplicaciones.

Estos módulos comparten el servidor de base de datos en la ONE y en la OTE, porque ambos se comunican con la base de datos para realizar el trabajo con los modelos y los demás elementos correspondientes a los mismos.

2.2.5.4 Alternativas para la configuración del despliegue.

El despliegue está concebido como se presentó en la **Figura 21,** pero puede ocurrir que en las oficinas de estadística se instalen los dos módulos en la misma computadora con todos sus componentes, y haya otra computadora que se utilice para el servidor de la base de datos, debido a razones económicas o porque un mismo usuario desempeñe varios roles dentro de la oficina, y no haga falta utilizar dos computadoras para ello.

2.3 Combinación de resultados.

En este epígrafe se realiza la combinación de algunos resultados obtenidos en la arquitectura, como la definición de las herramientas y plataformas necesarias en los Módulos Entrada de Datos y Generador de Modelos para su desarrollo. También se define la estrategia de desarrollo utilizada en los dos módulos dichos anteriormente, así como la definición de la configuración de los puestos de trabajo según los roles y la ubicación de los servidores más importantes dentro del proyecto. Por último se definen las pruebas que se le van a realizar a la arquitectura establecida en estos módulos y algunas métricas para medir el diseño arquitectónico realizado.

2.3.1 Definición de las herramientas y plataformas a utilizar para el desarrollo de los Módulos Entrada de Datos y Generador de Modelos.

Las herramientas de desarrollo de software construyen y diseñan las aplicaciones, para el desarrollo de los módulos contamos con herramientas de Diseño y Construcción, de Gestión de Configuración, de Requisitos y Análisis entre otras. Es muy importante seleccionar las herramientas que sean más compatibles a la hora de desarrollar las aplicaciones, por lo que se debe realizar un estudio de las mismas. Para el desarrollo de ambos módulos se utilizaron las mismas herramientas**,** estas son:

- Para la implementación se utiliza la herramienta de desarrollo Microsoft Visual Studio 2005 y lenguaje de programación C#.
- Para el modelado el Rational Rose Enterprise Edition 2003.
- Como servidor de base de datos el Microsoft SQL Server 2000.
- Herramienta Case para el diseño de la base de datos el Erwin Studio 6.6.
- Como servidor de control de versiones el Subversión 1.4.2.
- La plataforma de desarrollo escogida para la construcción de las aplicaciones de ambos módulos es Microsoft .NET, con Framework 2.0.
- Para la generación de la capa de acceso a datos se utiliza el TierDeveloper 4.0.
- Para garantizar la integración entre el Visual Studio 2005 y el SVN se utiliza el AnkhSetup-1.0.1.2734**.**
- Para la planificación del proyecto se utiliza el Microsoft Project 2003.

2.3.2 Definición de la estrategia de desarrollo.

La estrategia de desarrollo de los Módulos Generador de Modelos y Entrada de Datos, está centrada en el Proceso Unificado del Rational (RUP), este proceso consta de 9 flujos de trabajo, 6 de ingeniería, los cuales son: Modelamiento del negocio, Requisitos, Análisis y Diseño, Implementación, Pruebas e Instalación y 3 de apoyo, estos son: Administración de Configuración y Cambios, Administración de Proyectos y Ambiente, además posee tres características esenciales, está dirigido por casos de uso, centrada en la arquitectura y es iterativo e incremental.

La estrategia tiene como principal objetivo transformar los requisitos del usuario en un sistema, a través de un conjunto de actividades que se realizan a lo largo del proceso de desarrollo. El proceso se basa en componentes bien conectados a través de interfaces y utiliza el UML, como lenguaje de representación visual, para la representación de los esquemas.

El estar dirigido por los casos de uso significa que cada fase en el camino hacia el producto final está relacionada con lo que los usuarios hacen realmente, porque los casos de uso reflejan las funcionalidades que el usuario necesita. Los casos de uso no son más que fragmentos de funcionalidades del sistema y constituyen los requerimientos funcionales del mismo. Lleva a los desarrolladores a garantizar que el sistema se ajuste a las necesidades reales del usuario.

El estar centrado en la arquitectura significa que el desarrollo del software se centra en obtener el estilo arquitectónico que dirigirá la construcción del sistema en las primeras fases, garantizando un proceso continuo no sólo para la versión en curso del producto sino para la vida entera del mismo. La arquitectura muestra una visión del sistema, con la cual el equipo de desarrollo del proyecto tiene que estar de acuerdo. La arquitectura está relacionada con las vistas, las cuales conforman el modelo de la arquitectura del sistema. La arquitectura trabaja con los casos de uso claves en el sistema, a medida que estos se especifican y evolucionan la arquitectura va madurando y se descubre más de la misma.

El proceso es iterativo porque el desarrollo del producto va a pasar por varios pasos en el flujo de trabajo, las iteraciones por las que pasa el software deben ser planificadas y tratar los riesgos del sistema. Cuando una iteración cumple con todos los objetivos trazados en la misma entonces se puede pasar a la siguiente. Es incremental porque el producto va creciendo a medida que pase el tiempo. RUP propone que cada una de las 4 fases, ya sea la de Inicio, Elaboración, Construcción o Transición, se realicen a través de iteraciones, en las cuales se pasa por todos los flujos de trabajo, realizando actividades de los mismos, en mayor o menor medida (Jacobo, Booch y Rumbaugh, 2004).

2.3.3 Definición de la configuración de los puestos de trabajo según los roles.

Los Módulos Generador de Modelos y Entrada de Datos pertenecientes al proyecto ONE de la facultad 3, radican en el docente 2, laboratorio 18, los equipo de desarrollo de ambos módulos están formado por varios estudiantes y cada uno de los integrantes del equipo desempeña un rol definido por RUP. Está el analista, el arquitecto y tres programadores. Además hay otros estudiantes que se encargan de la gestión de configuración, la calidad, la planificación y la administración de la base de datos de los módulos.

Figura 24 Distribución de los puestos de trabajo según los roles.

Herramientas utilizadas por los integrantes de los módulos según su rol.

Herramientas	**1**	**2**	**3**	**4**	**5**	**6**
Rational Rose Enterprise Edition 2003	X		X			
Microsoft Visual Studio 2005	X	X				
Microsoft SQL Server 2000	X	X				
TortoiseSVN-1.4.3.8645-win32	X					
AnkhSetup-1.0.1.2734	X					
TierDeveloper 4.0	X	X				
Microsoft Project 2003					X	
Subversión 1.4.2				X		
Microsoft SQL Server 2000	X					X

1. Arquitecto
2. Programador
3. Analista

4. Gestión de configuración
5. Planificador
6. Administrador de Base de Datos.

Las computadoras utilizadas por los integrantes de los módulos tienen las siguientes características de hardware:

- Pentium 4.
- Velocidad de procesamiento de 2.40 GHz.
- Memoria RAM de 256.
- Disco Duro: 80 GB.

La computadora que se utiliza como servidor de la Base de Datos posee las siguientes características:

- Pentium 4.
- Velocidad de procesamiento de 2.40 GHz.
- Memoria RAM de 512.
- Disco Duro: 80 GB.

2.4 Modelo de pruebas realizadas a la arquitectura de los Módulos Entrada de Datos y Generador de Modelos.

Las pruebas son de gran importancia para la calidad del sistema que se está construyendo, estas toman su mayor fuerza cuando ya el software está construido, con el objetivo de detectar la mayor cantidad de errores en el mismo, de validar y probar que el sistema cumpla con los requisitos y que trabaje como fue diseñado. Las pruebas se utilizan entre otras cosas para comprobar la lógica de los componentes del sistema y para detectar errores en el código fuente. Para la realización de las pruebas se cuenta con procedimientos y métodos de prueba. En las pruebas el sistema y los componentes que lo integran son ejecutados a través de condiciones específicas para detectar la mayor cantidad de errores en el mismo antes de ser entregado al usuario final. Existen pruebas encaminadas a chequear un conjunto de características relacionadas con la calidad de arquitecturas de software, estas son: funcionalidad, confiabilidad, eficiencia, mantenibilidad y portabilidad (Camacho, Cardeso y Núñez, 2004). Todas estas características relacionadas con la arquitectura de software se deben tener en cuenta a la hora de realizar

algún cambio en la misma, y se debe realizar un diseño arquitectónico teniendo en cuenta las funcionalidades que debe tener el sistema, para no incumplir con estos elementos de calidad relacionados con la arquitectura de software. Es bueno tener presente que casi siempre cuando se realiza un cambio en la arquitectura de software este está relacionado con alguna de estas características, antes de realizar este cambio es preciso saber cuáles son los elementos de la calidad que se perjudican con el cambio.

Existen pruebas relacionadas con la arquitectura cliente – servidor (Pressman, 2005), y estas son muy complejas, tanto que hoy en día constituyen un reto para los ingenieros de prueba de software, porque para realizar estas pruebas se deben tener en cuenta aspectos relacionados con la plataforma de desarrollo, con la red, con el almacenamiento de información centralizada y con el acceso de los clientes a esa información, entre otros.

Hay varios tipos de pruebas de software propuestas por la metodología RUP, estas son:

- Pruebas de unidad: correspondiente a la revisión de una parte del sistema, o sea del código fuente de los componentes y revisa además las entradas y salidas de la interfaz del sistema.
- Pruebas de integración: en estas se prueban los componentes del sistema combinados.
- Pruebas de Sistema: estas prueban el software funcionando como un todo.
- Pruebas de Aceptación: estas se realizan antes de desplegar el sistema y la realiza el usuario final.

De estas pruebas la que más interesa para la arquitectura de software es la prueba de integración, estas tienen como principal objetivo unir las partes que fueron probadas en las pruebas de unidad y realizar una prueba con la estructura del programa completo para ver si este cumple con los elementos de diseño propuestos.

Existen varios tipos de pruebas de integración, la prueba de integración descendente, la prueba de integración ascendente, la prueba de regresión y la prueba de humo, estas estrategias de pruebas surgieron para que no existan tantos errores en conjunto cuando se unen todas las partes del sistema y que se realice una integración poco a poco, porque cuando se tiene el sistema unido completo es demasiado complicado corregir los errores y entonces se entra en un ciclo sin fin de errores. Estas estrategias de prueba son realizadas indistintamente en dependencia de las características del sistema. Además para un mismo sistema se pueden efectuar varias estrategias de pruebas juntas.

A la arquitectura de los Módulos Entrada de Datos y Generador de Modelos se le aplican las pruebas de integración ascendente y de regresión. En esta prueba comenzamos por los módulos más bajos en la estructura del sistema, los cuales son la BD y la capa de acceso a datos.

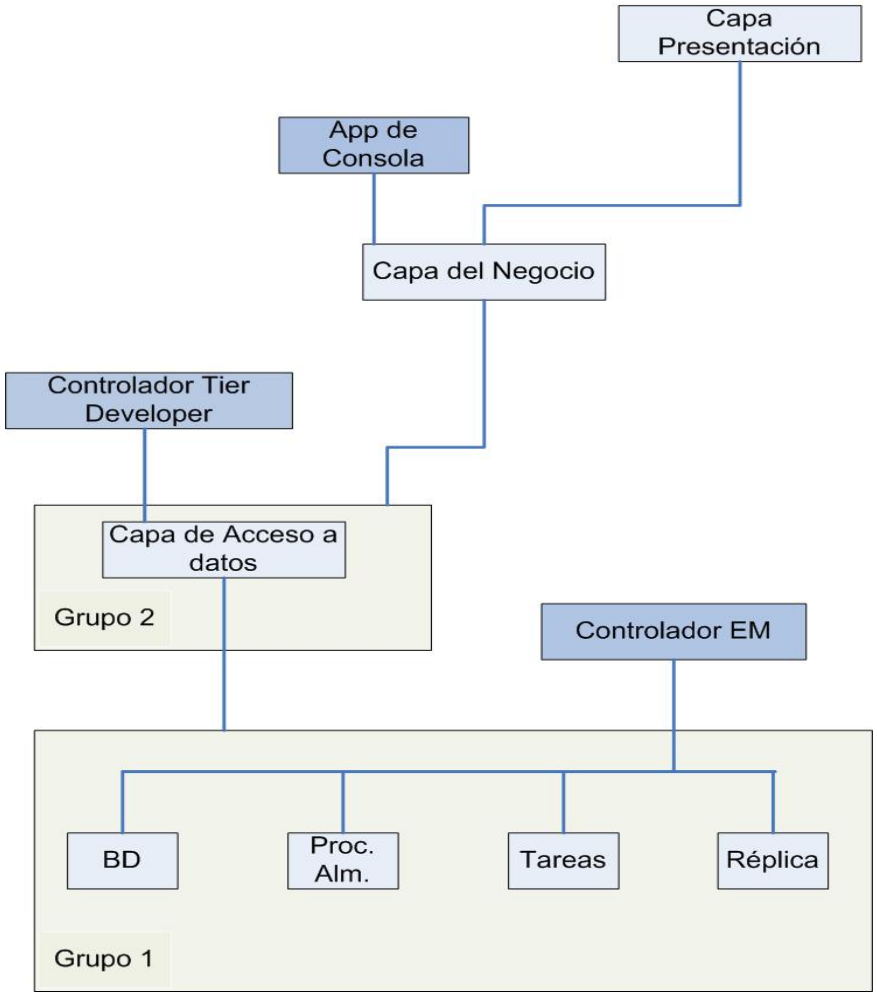


Figura 25 Pruebas de Integración Ascendente para la arquitectura aplicada a los módulos.

Para probar la capa de los datos, que sería donde se encuentra la Base de Datos ONE utilizamos como controlador el Enterprise Manager, este es un componente del Microsoft SQL Server 2000 que permite realizar pruebas en la Base de Datos. Este módulo constituye el primer grupo y se encuentra dividido por la Base de Datos, los Procedimientos Almacenados, las Tareas y la Réplica.

Para probar la capa de acceso a datos utilizamos como controlador el mismo programa que se utiliza para generar la misma, el TierDeveloper 4.0, y esta constituye el grupo 2.

Para probar la capa del Negocio se hizo una aplicación de consola.

Para probar la capa de Presentación se utiliza la interfaz de los sistemas entrando datos a los mismos y obteniendo los errores.

Por cada uno de los módulos o capas se debe obtener la cantidad de errores utilizando para ello los controladores.

Para realizar todas estas pruebas se escogen los casos de uso arquitectónicamente significativos por módulos, los casos de pruebas son definidos por la ingeniera de pruebas. Las pruebas están centradas a los siguientes casos de uso:

- Gestionar Modelos
- Gestionar Nomenclatura
- Administrar Modelos
- Administrar Usuarios.
- Autenticarse

Las pruebas realizadas a los distintos módulos son de caja negra porque no tienen que ver directamente con el código sino con la entrada y salida de datos a las distintas capas.

En estas pruebas se utilizan además la regresión por lo que los casos de uso probados en el módulo de Base de Datos son los mismos que se utilizan en el módulo de accesos a datos, aunque en este último se pueden adicionar nuevos casos de uso, pero se deben mantener por lo menos los mismos que se tenían anteriormente.

2.5 Métricas utilizadas para evaluar el diseño arquitectónico de los Módulos Entrada de Datos y Generador de Modelos.

Para los sistemas informáticos es necesaria la medición, porque esta da la medida de la calidad que posee el software que se está construyendo y de los modelos que desarrollamos para ello. Estas métricas dan resultados cuantitativos y con ellos los desarrolladores del sistema pueden prevenir problemas que si no se corrigen a tiempo puede ser que después no tengan solución. Existen métricas para evaluar el software de forma general, pero también existen otras que sirven para evaluarlo mientras se está desarrollando, en el análisis, el diseño y cuando se construye la estructura del mismo, entre otras.

Las métricas de diseño arquitectónico se centran en las características de la estructura que tenga el programa y son de caja negra porque no se necesita saber la parte interna del sistema, el código como tal, sino más bien tiene que ver con los módulos y como se relacionan estos entre si.

Pressman (2005) habla de métricas para medir la arquitectura planteadas por varios autores que conocen del tema como Card y Glass que plantean medidas para la complejidad como son:

- La complejidad estructural.
- La complejidad de los datos.
- La complejidad del sistema.

Además expresa las métricas planteadas por Henry y Kafura basadas en la complejidad también y las de Fenton que expresa varias métricas de morfología, tienen que ver con la forma del sistema.

Todas estas métricas se utilizan en la evaluación de la arquitectura de los Módulos Entrada de Datos y Generador de Modelos, por su gran importancia.

1. La complejidad estructural se calcula por módulos y la ecuación para realizar el cálculo es la siguiente:

$M(i) = h^2_{out}(i)$

Donde **$h_{out}(i)$** es el número de módulos subordinados o invocados por el módulo(i).

2. La ecuación para calcular la complejidad de los datos es la siguiente.

$N(i) = g(i) / [h_{out}(i)+1]$

Donde **g(i)** representa el número de variables de entrada y salida que entran y salen del módulo (i).

3. Para calcular la complejidad de sistema se utiliza la siguiente ecuación:

$S(i) = M(i) + N(i)$

4. La ecuación para calcular la métrica propuesta por Henry y Kafura es la siguiente:

$MHK = long(i) * [h_{in}(i) + h_{out}(i)]^2$

Donde **long(i)** representa el número de sentencias en lenguaje de programación en el módulo (i).

$h_{in}(i)$ representa la concentración del módulo (i) y la concentración esta dada por el número de llamadas al módulo más el número de estructuras de datos de que el módulo (i) recoge datos.

5. Para calcular las métricas de morfología simple se utiliza la siguiente ecuación:

Tamaño = n + a

Donde **n** representa el número de nodos (módulos)

a representa el número de arcos (líneas de control)

Capítulo 3 Análisis del resultado de la Arquitectura planteada.

En este capítulo se analizan los resultados obtenidos a partir de la aplicación del diseño arquitectónico establecido para los Módulos Entrada de Datos y Generador de Modelos. Además se obtienen resultados de las pruebas de integración realizadas al estilo arquitectónico utilizado para el desarrollo de los módulos y de las métricas de diseño arquitectónico establecidas para la medición de la calidad de la arquitectura. Se realiza también una valoración crítica de la arquitectura actual de los módulos. Se plasma la opinión de especialistas sobre la arquitectura establecida en los módulos y se da una pequeña valoración del cliente acerca de la arquitectura de los módulos.

3.1 Análisis de los resultados obtenidos a partir de la aplicación de las métricas de diseño arquitectónico y de las pruebas realizadas.

En cuanto a las métricas aplicadas se obtuvieron varios resultados, los cuales están en dependencia de la medición realizada. Hay resultados por capas y por módulos de cada una de las métricas aplicadas.

3.1.1 Módulo Entrada de Datos.

La Complejidad Estructural del módulo por capas es la siguiente:

La capa de Presentación tiene una complejidad estructural de 8.

La capa de Lógica del Negocio tiene una complejidad estructural de 20.

La capa de Acceso a Datos tiene una complejidad estructural de 9.

La complejidad estructural del módulo es de 37.

Como se puede observar la capa de Lógica del Negocio es la que más complejidad estructural posee en el Módulo de Entrada de Datos, porque esta es la que tiene el mayor peso en el sistema, es invocada por varios componentes y es una de las más necesarias dentro de la arquitectura del sistema, sin esta capa y los componentes que la integran, las demás capas no pudieran funcionar acorde con las necesidades de la aplicación. Por su parte las capas de Presentación y de Acceso a Datos poseen una complejidad estructural bastante similar, y un poco baja con respecto a la de Lógica del Negocio, aunque no por ello dejan de ser necesarias e importantes para la arquitectura del sistema. El sistema de forma general posee una complejidad estructural media.

La Complejidad de los datos del módulo por capas es la siguiente:

La capa de Presentación tiene una complejidad de los datos de 29.

La capa de Lógica del Negocio tiene una complejidad de los datos de 15.

La capa de Acceso a Datos tiene una complejidad de los datos de 31.

La complejidad de los datos del módulo es de 75.

En este caso la capa de Acceso a Datos es la que posee mayor complejidad de los datos para el Módulo de Entrada de Datos, esto da la medida de la complejidad interna de esta capa, aunque la capa de Presentación también posee una complejidad bastante similar a la de la capa de Acceso a Datos. La complejidad del módulo de forma general es media.

La Complejidad del Sistema por capas es la siguiente:

La capa de Presentación tiene una complejidad de 37.

La capa de Lógica del Negocio tiene una complejidad de 35.

La capa de Acceso a Datos tiene una complejidad de 40.

La complejidad del sistema es de 112.

En cuanto a la complejidad del sistema la capa que más complejidad posee es la de Acceso a Datos, este resultado es a partir de la suma de la complejidad estructural con la complejidad de los datos de esta capa, en este aspecto todas las capas son más o menos similares, los valores de las mismas están dentro de un rango bastante estrecho, si acaso se llevan una o dos unidades de diferencia, a medida que aumente la complejidad por capas se hace mayor la complejidad del sistema, pero en este caso la complejidad del sistema para este módulo es media, se comportó proporcional con la complejidad estructural y de los datos calculadas con anterioridad.

Las Métrica de Henry y Kafura aplicadas al módulo por capas tienen los siguientes valores:

La capa de Presentación tiene un valor de 4851.

La capa de Lógica del Negocio tiene un valor de 6368.

La capa de Acceso a Datos tiene un valor de 4157.

El módulo tiene un valor de 15376.

En este caso la capa que mayor complejidad posee es la de Lógica del Negocio, esta métrica da la medida de la concentración de la capa y de las sentencias de código en lenguaje de programación, en este caso C#, que se emplearon en la misma. Las capas de Presentación y Lógica del Negocio son similares en este caso. Pienso que el módulo de forma general tiene un valor medio y el esfuerzo en las pruebas del mismo va a estar centrado en la capa de Lógica del Negocio.

La métrica de morfología simple aplicada al módulo completo tiene un valor de 33.

Esta métrica nos da la medida de la conectividad entre los nodos que presenta nuestra arquitectura, además nos da la medida del acoplamiento de los componentes de la misma.

En esta métrica es importante destacar dos aspectos, que aparecen en (Pressman, 2005) y son necesarios conocer a la hora de medir la arquitectura, estos son:

Profundidad: está dada por el camino más largo desde el nodo raíz hasta la hoja, en este caso es de 6.

Anchura: está dada por la mayor cantidad de nodos que exista en cualquier capa de la arquitectura, en este caso es de 4.

3.1.2 Módulo Generador de Modelos.

La Complejidad Estructural del módulo por capas es la siguiente:

La capa de Presentación tiene una complejidad estructural de 1.

La capa de Lógica del Negocio tiene una complejidad estructural de 1.

La capa de Acceso a Datos tiene una complejidad estructural de 1.

La complejidad estructural del módulo es de 3.

Como se puede ver anteriormente las tres capas del Módulo Generador de Modelos poseen la misma complejidad estructural, este módulo de forma general tiene menos complejidad estructural que el Módulo Entrada de Datos, lo que significa que la invocación entre los componentes en este módulo es menor que en el Módulo Entrada de Datos. La complejidad estructural del Módulo Generador de Modelos es baja con respecto al Módulo Entrada de Datos.

La Complejidad de los datos del módulo por capas es la siguiente:

La capa de Presentación tiene una complejidad de los datos de 7.

La capa de Lógica del Negocio tiene una complejidad de los datos de 63.

La capa de Acceso a Datos tiene una complejidad de los datos de 26.

La complejidad de los datos del módulo es de 96.

La capa de Lógica de Negocio es la que posee mayor complejidad de los datos, los valores de las capas de Presentación y de Acceso a Datos son similares en este sentido. El Módulo Generador de Modelos tiene mayor complejidad de los datos que el Módulo de Entrada de Datos. Esto es debido a que en el Módulo Generador de Modelos es donde se crean los modelos con todas sus características dentro. Este tiene una complejidad de los datos alta con respecto al Módulo Entrada de Datos.

La Complejidad del Sistema por capas es la siguiente:

La capa de Presentación tiene una complejidad de 8.

La capa de Lógica del Negocio tiene una complejidad de 64.

La capa de Acceso a Datos tiene una complejidad de 27.

La complejidad del sistema es de 99.

En cuanto a la complejidad del sistema la capa que mayor valor presenta es la de Lógica del Negocio, este valor está dado por la suma de la complejidad estructural más la complejidad de los datos, esta capa es la que mayor valor a adquirido en todas las complejidades calculadas para la misma, por lo que se debe prestar una mayor atención a la misma en las pruebas. La complejidad del sistema para el Módulo

Generador de Modelos es menor que para el Módulo Entrada de Datos y por su puesto el esfuerzo en la realización del segundo es mayor que en el primero. La complejidad del sistema para el Módulo Generador de Modelos es baja con respecto al Módulo Entrada de Datos.

Las Métrica de Henry y Kafura aplicadas al módulo por capas tienen los siguientes valores:

La capa de Presentación tiene un valor de 2328.

La capa de Lógica del Negocio tiene un valor de 4995.

La capa de Acceso a Datos tiene un valor de 660.

El módulo tiene un valor de 7983.

En estas métricas la capa que mayor valor posee es la capa de Lógica del Negocio, este valor fue calculado mediante la cantidad de sentencias de código que posee la capa y mediante la concentración de la misma que está dada por la cantidad de solicitudes que le hacen a los componentes de la capa y que ella hace a los componentes de las demás capas. Esta capa es la que mayor complejidad ha tenido en todos los sentidos. El Módulo Generador de Modelos posee menor valor en esta métrica que el Módulo Entrada de Datos.

La métrica de morfología simple aplicada al módulo completo tiene un valor de 9.

Esta métrica da una visión de la forma del sistema, en ella se tienen en cuenta los nodos del sistema y los arcos que los relacionan. Como se puede ver el Módulo Generador de Modelos posee menor valor en esta métrica que el Módulo Entrada de Datos.

Para este módulo también se calcularon los valores de Anchura y Profundidad, el valor de la Anchura es de 1 y el de la Profundidad es de 4.

En este módulo los valores de anchura y profundidad son menores que en el Módulo Entrada de Datos.

3.1.3 Resultado de las pruebas realizadas a los Módulos Entrada de Datos y Generador de Modelos.

Se realizaron pruebas a cada una de las capas que componen la arquitectura, a estas capas se le llaman módulos. En el módulo de la Base de Datos se aplicaron pruebas específicamente a las tablas relacionadas con los modelos y a los usuarios que son los que tienen que ver con los sistemas Entrada de Datos y Generador de Modelos. La tabla modelo está relacionada con las tablas Indicador, Aspecto, Periodicidad y Variante, entre otras tablas que se forman por la relación de dependencia entre las tablas. Cuando se adiciona, elimina o modifica algún modelo, la base de datos está preparada para que se realice

en forma de cascada, cuando se adiciona un modelo, se adiciona su variante, los indicadores relacionados con el mismo, la periodicidad y los aspectos, esto sucede igual para cuando se modifica o se elimina algún modelo. En la base de datos no existe ningún problema a la hora de trabajar con los modelos.

Cuando se trabaja con las tablas relacionadas con los usuarios, tampoco se detecta ningún problema. Se puede adicionar perfectamente un nuevo usuario, o eliminarlo o modificar los datos del mismo. Además se puede gestionar todo lo relacionado con los roles y los permisos de los usuario en el sistema. Cuando se autentican los usuarios también mediante la base de datos se sabe si estos tienen permisos o no en el sistema y si su nombre de usuario y contraseña es correcto o no.

El Módulo Generador de Modelos está relacionado con los indicadores, cuando se crean los modelos en este sistema se chequea la descripción real de estos indicadores y esta descripción aparece en la base de datos, esto también fue probado y no se encontraron errores en este sentido.

El módulo de Base de Datos está bien implementado y no posee errores, por lo que las aplicaciones pueden conectarse perfectamente a la base de datos y realizar las tareas necesarias en la misma sin dificultades.

En la Capa de Acceso a Datos se aplicaron pruebas a las funcionalidades relacionadas con los modelos, aquí se midió la carga de los modelos de estadística continua, la eliminación de este mismo tipo de modelos, la actualización, como instalar los modelos y registrar información captada por los mismos. Estas funcionalidades para su trabajo en esta capa necesitan de los objetos Aspecto, Indicador, Información Acumulada, Modelo, Modelo_Indicador, Variante, Relacion_T, Variante 2 y Modelo_Aspecto_Indicador, todos estos objetos están en el TierDeveloper y representan las tablas de la base de datos con que necesitan trabajar los Módulos de Entrada de Datos y Generador de Modelos para su funcionamiento.

Esta capa cuenta además con un grupo de consultas que se muestran en el TierDeveloper y se relacionan con las que se encuentran en la base de datos, que son necesarias para el desarrollo de la misma como son: Listar_Ind_hijos, Listar_Modelos, Listar_Asp_hijos, Relación_T, Listar_Tipo_Asp, Listar_Modelos_llenados, que son utilizadas para obtener datos de los modelos en la base de datos y luego trasmitirlos a la capa del negocio.

Cuando se registra información de los modelos, se actualizan o se eliminan no existe ningún tipo de problema en esta capa, pero a la hora de Instalar los modelos y de cargarlos se detectaron algunos errores, como por ejemplo cuando se instaló el modelo 005, todo funcionó bien, pero cuando se va a

instalar el modelo 006 da error y no te permite instalar más modelos. A la hora de cargar un modelo determinado no carga todos los datos que debería del mismo aunque si carga los fundamentales. Esta capa de forma general está bien lo que se detectaron dos errores en la misma que pueden ser depurados.
La Capa del Negocio no presenta errores, todas las funcionalidades de gestión de modelos, de indicadores, de administración de modelos y de usuarios, entre otras se cumplen perfectamente.
En la Capa de Presentación no se encontraron grandes errores a la hora de realizar las pruebas en ambos módulos. En el Módulo Generador de Modelos, se pueden crear los modelos de estadística continua tanto de talón abierto como de talón cerrado, eliminarlos y modificar los datos de los indicadores, aspectos, número del modelo, periodicidad, día de captación entre otros. Solamente se encontró un error cuando se fue a crear un nuevo modelo, que no permitía eliminar un indicador específico. Estos modelos se guardan perfectamente en formato XML, y se pueden abrir para hacer modificaciones en los mismos. Además el sistema muestra mensajes para indicar que las operaciones se han realizado correcta o incorrectamente y cuando los datos entrados no son correctos también te lanza excepciones. El Módulo Entrada de Datos permite instalar los modelos, digitar los modelos, eliminar modelos, adicionar usuarios al sistema. Además la autenticación en esto módulo trabaja bien.

3.2 Valoración crítica de la arquitectura actual de los Módulos Entrada de Datos y Generador de Modelos.

La arquitectura de los Módulos Entrada de Datos y Generador de Modelos ha tenido buenos resultados de forma general. La capa de Presentación está bastante adelantada a pesar de que ahora en la próxima iteración del proyecto se le van a agregar algunos elementos, la capa del Negocio responde bien a las funcionalidades del sistema y está bastante completa, a pesar de que faltan algunas funcionalidades que serán desarrolladas en la próxima iteración.

La capa de acceso a datos trabaja bastante bien de forma general, le brinda a la capa del negocio las funcionalidades necesarias para trabajar con los modelos y con la gestión de los usuarios en el sistema. La única desventaja que presenta, está relacionada con la herramienta que se utiliza para la generación de la misma, el TierDeveloper que soporta 4 gestores de base de datos, el DataBase Enterprise Oracle, el Microsoft SQL Server, el Microsoft Access y el DB2, lo que elimina la posibilidad de emigrar algún día para software libre. A pesar de que el TierDeveloper presente este detalle, ha sido de

gran utilidad para la conexión de los módulos con la base de datos y hasta el momento ha resuelto los problemas y funcionalidades que necesitan las aplicaciones.

La capa de los datos está bien implementada y a partir de la misma se obtienen todos los datos necesarios para la interacción con los sistemas. A pesar de existir algunos detalles en la implementación de las capas, la arquitectura se ha mantenido bastante estable, por lo que va a seguir siendo utilizada por el proyecto en próximas iteraciones del producto.

Los dos módulos poseen su prototipo de interfaz, para ver la página principal del Módulo Entrada de Datos ir al anexo 10 y para ver la del Módulo Generador de Modelos ir al anexo 11. Ambos módulos cumplen con las funcionalidades propuestas en la primera iteración como la generación de los modelos de estadística continua de talón abierto y de talón cerrado. Además permiten guardar los modelos realizados, eliminarlos y actualizarlos. Permiten además llenar estos dos tipos de modelos y validarlos, así como realizar otro gran número de funcionalidades explicadas en el capítulo dos, en la descripción de la arquitectura. Por lo que de forma general creo que la arquitectura esta bastante bueno y se ha mantenido, a pesar de que existan algunos problemas que se deben depurar en próximas iteraciones del producto.

3.3 Opinión de especialistas en arquitectura software de la universidad.

De modo general la arquitectura planteada es ventajosa pues, presenta un modelo en tres capas que establece independencia entre las diferentes partes del sistema, es flexible a cambios y eficiente para el trabajo en equipo. Podemos decir además que con el uso de la plataforma .Net, se logra una gran productividad, aunque no se debe dejar de señalar que esta plataforma es propietaria y no está acorde con las políticas de migraciones a software libre de nuestro país, máxime para entidades como La Oficina Nacional de Estadísticas órgano rector de la estadística en Cuba.

Es importante destacar además el uso de herramientas de corta generación, para el desarrollo, por ejemplo TierDeveloper para la capa a de acceso a datos de las aplicaciones brindando esto las siguientes ventajas:

1. Es una herramienta que permite hacer el mapeo objeto relacional de manera muy intuitiva, lográndose una gran productividad en los equipos de desarrollo.
2. Genera código libre de errores de forma automática, lo que hace que se tenga una cierta seguridad de ante mano en las aplicaciones.

3. Permite sincronización del código cuando se realizan modificaciones a la base de datos, esto hace que sea relativamente fácil realizar cambios en la capa de acceso a datos.
4. Permite cambiar el código que se genera mediante templates.

No obstante también se pueden hacer algunos señalamientos que serían muy positivos tenerlos en cuenta:

1. Sólo soporta los gestores de base de datos SqlServer, Oracle, DB2, Access, esto sería negativo si se decide migrar las aplicaciones a otro gestor de bases de datos, como PostGres.
2. Es una herramienta propietaria.
3. Posee un framework del cual no se conoce el código y es necesario para que las capas generadas puedan ejecutarse correctamente. Este código no es posible modificarlo.
4. Además no se posee una licencia de software válida, lo que trae cierta inseguridad para el desarrollo con este sistema.

Proponemos que se estudie la alternativa del uso de un mapeador objeto relacional libre como NHibernate.

3.4 Valoración del cliente acerca de la arquitectura de los módulos.

En cuanto a la tecnología utilizada para soportar los sistemas pienso que es muy válida, que le da un salto bastante grande a lo que utilizábamos anteriormente, pienso que ustedes han hecho un estudio muy bueno de las mismas, por lo que esta debe durar varios años para el trabajo en las oficinas. En todo momento pienso que es un avance grande para nosotros porque en primera ya el MicroSet con el avance actual de las tecnologías y otros elementos está un poco atrasado, aunque no se puede dejar de decir que es un sistema muy bueno y nos ha servido durante muchos años para todo el trabajo estadístico en el país y también necesitábamos de nuevas funcionalidades para el trabajo en las oficinas, como por ejemplo una base de datos como la que ustedes tienen hecha que soporte toda la información con la que trabajamos, por ejemplo la réplica de información entre las oficinas, y otros elementos que son muy importantes y significativos para nosotros. Nunca el sistema que ustedes crearon va a constituir un freno, sino que vamos a tratar de adaptarnos al mismo y constituye un tremendo avance.

En cuanto a las principales ventajas y desventajas del sistema hasta el momento, puedo decir que indudablemente este sistema aumenta más funcionalidades que el sistemas con que contábamos anteriormente para el procesamiento de toda la información estadística en el país, porque como han

integrado mucha tecnología de punta a los mismos van a ser más avanzados y pienso además que nos duren varios años porque ustedes seleccionaron la tecnología apropiada para ello.

Considero que tiene una forma de estructuración muy buena ya que tiene una integralidad total que parte desde el momento en que se genera el modelo, hasta culminar el trabajo con los mismos.

Actualmente el principal problema aunque creo que no se debe considerar como una desventaja sino como una preocupación es que a pesar de que se realizó una aplicación con interfaz amigable, las personas que tenemos en provincia trabajando en las oficinas son de determinada edad y están adaptados a la aplicación anterior, nos preocupa mucho como van a asimilar la nueva misión de trabajo con este sistema nuevo completamente y con tecnología desconocida para ellos. Pero bueno pienso que en esto nos ayudarán los prototipos que ya ellos se llevaron para las provincias hace poco cuando se hizo el taller con ustedes aquí en la universidad, porque ya con esto se van a ir familiarizando con el sistema y todas las características novedosas que puede tener el mismos.

En las futuras iteraciones nos debemos enfocar a la base de datos, porque ya se están revisando los prototipos en las provincias como dije anteriormente. Ahora yo vine para darles un volumen grande de información con la que nosotros trabajamos para introducirlo en la base de datos que ustedes tienen aquí y esta sería una prueba de carga de información que se va a hacer a la Base de datos para ver si en realidad soporta esto o no, y que ustedes sepan lo que deben mejorar en la misma si algo sucediera. Después de que se prueben los prototipos en las provincias, veremos cuales serán las principales preocupaciones que tienen los compañeros y nos trazaremos a partir de ahí nuevos objetivos.

Creo que se han visto resultados significativos hasta el momento porque se ha avanzado muchísimo más de lo que esperábamos, generalmente una persona recién graduada necesita por lo menos un año para familiarizarse y adquirir todos los conocimientos estadísticos, sin embargo con ustedes comenzamos en julio del 2006 y mira ya como han avanzado y se ha logrado gran dominio en cuanto a la estadística, bueno tanto es así que ya contamos con una primera versión del producto y ya se realizó el taller con los demás compañeros de provincia que vinieron hace unos días, y cuando se fueron se llevaron el prototipo inicial del sistema para que trabajen con el allá y traten de encontrarle los errores a las aplicaciones.

El sistema cumple con los requisitos propuestos, por lo menos cumple con los requisitos propuestos por nosotros hasta el momento, y con lo que nos propusimos para la primera iteración, aunque como ya dije anteriormente se están probando todavía en las provincias y hay que esperar a ver que pasa.

Para obtener algo más acerca de la opinión del cliente sobre el proyecto en general y los sistemas pertenecientes a los Módulos Entrada de Datos y Generador de Modelos, ir a ver anexos desde el 12 al 17, donde hay una serie de avales realizados por los clientes de las provincias del país cuando vinieron al taller celebrado aquí en la universidad, en total los avales son 15, pero aquí en la tesis no aparecen todos.

Conclusiones

Con el desarrollo del presente trabajo basado en el establecimiento del diseño arquitectónico adecuado para los Módulo Entrada de Datos y Generador de Modelos, se han cumplido los objetivos propuestos al inicio del mismo como son:

A través de la utilización de las herramientas de desarrollo para la construcción de las aplicaciones, se logró reducir el tiempo de desarrollo para la implementación de las aplicaciones de los módulos y el cumplimiento de los requerimientos funcionales de los mismos.

El estudio de los principales estilos arquitectónico que existen a nivel mundial, y la aplicación de un estilo en capas, basado en componentes y orientado a objetos, permitió darle modularidad a las aplicaciones, bajo acoplamiento entre las capas y los componentes que integran las mismas y enfatizando en la reutilización de estos componentes que es un elemento muy importante en el desarrollo de sistemas informáticos hoy en día.

La definición de la línea base de la arquitectura para estos dos módulos, enfatizando en las principales restricciones con que debe cumplir la arquitectura de estos sistemas tanto desde el punto de vista del hardware como del software, permitió que el equipo de desarrollo del proyecto trabajara sobre una misma línea en la implementación de los módulos.

La descripción de cada una de las vistas de la arquitectura proporcionó un mejor entendimiento de los sistemas por parte de los desarrolladores y del equipo de arquitectos y así tener en cuenta elementos importantes en el diseño e implementación de las aplicaciones.

Con la evaluación de la arquitectura de los módulos a través de las pruebas de integración ascendentes y de regresión, se obtuvo una idea de cuan eficiente es el sistema y de los errores que tiene el mismo, para así poder enfatizar en los principales problemas y contribuir a la calidad del sistema, lo que constituye una de las características esenciales en la arquitectura de los sistemas informáticos actuales.

Además se realizaron mediciones a ambos módulos a través de métricas que calculan la complejidad de los sistemas lo que permitió saber los valores cuantitativos de las complejidades de los mismos y no basarnos solamente en los aspectos cualitativos del software que son necesarios pero no suficientes.

Recomendaciones

Con el desarrollo de este trabajo se han visto algunos aspectos que seria bueno recomendar para futuras iteraciones del producto.

Aplicar el estilo arquitectónico utilizado en el desarrollo de los Módulos Entrada de Datos y Generador de Modelos, a los demás módulos del proyecto como el Módulo de Registros y Clasificadores y el Generador de Reportes.

Extender el subsistema de seguridad a los demás módulos del Sistema Integrado de Gestión Estadística.

Que se automatice la validación de los modelos en el caso del Módulo Entrada de Datos.

Realizar una mejor gestión de excepciones en los sistemas, para que estos sean más robustos y tolerantes a fallos.

Bibliografía Citada

Bass L., Clements P. y Kazman R. (2003). *Software Architecture in Practice.* Second Edition*.* Carnegie Mellon University: Addison Wesley.

Pressman R. (2005). *Ingeniería del Software. Un enfoque práctico*. Parte 1. La Habana Cuba: Félix Varela.

Bibliografía Consultada

Ciudad R. F. y Soto L. N. (2007). Electronic Source: *Patrones de Arquitectura.* Ciudad de la Habana. <http://inter-nos/Teleclases/Teleclases.asp?id_as=12> [Consulta: 20 de enero del 2007].

Camacho E., Cardeso F. y Núñez G. (2004). Guía de estudio de la Arquitectura de Software.

Hernández G. T. y Piquera V. L A. (1997). *Manual de usuarios del MicroSet NT*. Camagüey Cuba.

Jacobo I., Booch G. y Rumbaugh J. (2004). *El Proceso Unificado de Desarrollo de Software.* Volumen 1. La Habana Cuba: Félix Varela.

King G. C. (2005). *Iberneig in Action*. Estados Unidos de América: Manning Publications.

Larman C. (2004). *UML y Patrones. Introducción al análisis de diseño orientado a objetos*. Parte 2. La Habana Cuba: Editorial Félix Varela.

Pressman R. (2005). *Ingeniería del Software. Un enfoque práctico*. Parte 1. La Habana Cuba: Félix Varela.

Pressman R. (2005). *Ingeniería del Software. Un enfoque práctico.* Parte 2. La Habana Cuba: Félix Varela.
Reynoso C. B. (2004). Electronic Source: *Introducción a la Arquitectura del Software.* Universidad de Buenos Aires. <http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.asp#12 > [Consulta: 2 de diciembre del 2006].

Reynoso C. y Kicillof N. (2004). Electronic Source: *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Universidad de Buenos Aires. <http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp#1 > [Consulta: 2 de diciembre del 2006].

Glosario de Términos

Modelo: especie de planilla compuesta por tablas, cuadres e indicadores.

Modelo de talón cerrado: modelo que contiene un conjunto de indicadores definidos, de forma tal que los Centros Informantes deben reportar exactamente lo que aparece en el modelo.

Modelo de talón abierto: modelo que no tiene los indicadores definidos, por tanto los Centros Informantes deben seleccionar sus indicadores y luego reportar el modelo.

Cuadre: Se dice de una forma de validación que se realiza a través de fórmulas o comparaciones matemáticas.

Variante: Un modelo contiene variantes que especifican si el mismo pertenece a una empresa estatal o privada y a que organización. Aunque en otros caso permite especificar si el modelo tiene frecuencia mensual, trimestral o anual.

Indicador: se dice de una variable que puede tomar un valor de una determinada unidad de medida y de un determinado tipo de datos (generalmente numérico). Los indicadores de la ONE están bien definidos y tienen un código que los identifica.

Periodicidad: Se refiere a la frecuencia con que se los centro informantes deben de entregar la información. Puede ser mensual, trimestral, anual.

Nomenclatura: Documento que posee el universo de indicadores a captar en un modelo estadístico determinado.

Aspecto: Definen los datos que van en las columnas de los modelos.

Centros Informantes (CI): Los Centros Informantes son las empresas u organismos que deben dar parte a las oficinas de estadísticas. En algunos casos existen establecimientos que pueden ser considerados Centros Informantes.

OME: Oficina Municipal de Estadística, se encarga de las estadísticas a nivel municipal.

OTE: Oficina Territorial de Estadística, se encarga de las estadísticas a nivel territorial. Las oficinas a nivel territorial son generalmente oficinas provinciales aunque en algunos casos existen OTE 's que realizan las estadísticas de un territorio que no se considera una provincia, como por ejemplo: La Isla de la Juventud.

ONE: Oficina Nacional de Estadística, se encarga de las estadísticas a nivel nacional.