Universidad de las Ciencias Informáticas
Facultad 3

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Título: Análisis y Diseño del Módulo Generador de Reportes del Proyecto ONE

Autor

Yoanki Pascual Moreno Tamayo

Tutor

Ing. Disnier Alberto Camejo Domínguez

Ciudad de la Habana

24 de junio 2008

Se el cambio que quieres ver en el mundo…

Mahatma Gandhi

DECLARACIÓN DE AUTORÍA

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yoanki Pascual Moreno Tamayo

_____

Disnier Alberto Camejo Dominguez

_____

Dedico este trabajo

A la persona que más quiero en la vida, a mi madre Leticia por educarme con su ejemplo y dedicación, por ser el sostén de mis días y guiarme por este infinito camino, te quiero mucho mamá…

A mi bisabuelo Santiago y mi tío Pipo, por haber sido la figura paterna en mi vida, no hubiera querido mejores padres, estarán siempre en mi…

A mis queridos hermanos Yusimi, Yosnier y Yaniri, por regalarme su alegría, los quiero mucho hermanitos…

A mi abuela Puchita, por estar conmigo en los momentos bueno y malos, te adoro mima…

A mi abuelo Santiaguito, por cuidarme desde el cielo e iluminarme con su amor, te extraño abuelo…

A mis tías Nena y Ángela, por haber sido madres para mi, las tengo en mi corazón…

A mi sobrinita Tere y mi primita Yudith, por ser personitas maravillosas, las quiero mucho…

A mis entrañables amigos Damián, Eddy, Indira y Karelia, por brindarme su amistad incondicionalmente…

A todas las personas que me quieren…

Agradezco

A Dios por brindarme la posibilidad hoy de amar y agradecer a mis seres queridos…

A mi madre Leticia por haberme traído al mundo y confiar ciegamente en mí, regalándome su amor y devoción día a día, gracias por tu presencia mamá…

A mis tíos Julito y Juan por quererme como si fuera su hijo, gracias mis tíos queridos…

A mis primos Julito, Manuel, Aida, Haydee y Santiaguito, por estar siempre conmigo y ayudarme tanto, muchas gracias a todos…

A mi Bisabuela Virgen, por regalarme sus consejos cuando los necesitaba, gracias mami…

A mis tías Yusimí y Yamilé por ayudarme en momentos tan difíciles, gracias tías las quiero mucho…

A todos los amigos de la ***uci*** *que siempre me apoyaron, y confiaron a mí hasta el final, muy especialmente a Dixson, Félix, Fidel, Yamira, El Guaro y mi pequeña hijita (Leyanis), a todos muchas gracias…*

A todas las personas en mi barrio que me quieren, especialmente a La China, Kiki, Ariel, Orlando y Encarna, muchas gracias por su apoyo…

A mi tutor Disnie y a Jorge Ernesto por ayudarme tanto con el trabajo en el proyecto…

A todo el que hizo posible este trabajo…

Resumen

La Oficina Nacional de Estadísticas (ONE) es la entidad en Cuba cuya misión es garantizar la producción de estadísticas de calidad a través del Sistema Estadístico Nacional y controlar la difusión de dichas estadísticas en dependencias de las necesidades del país. La ONE necesita cambiar el software que utiliza en la actualidad para la gestión estadística, es por ello que en convenio con la Universidad de las Ciencias Informáticas (UCI) propone desarrollar el Sistema Integrado de Gestión Estadísticas (SIGE) para construir una herramienta que sustituya de forma eficiente a la que se utiliza en la oficina. Dentro del (SIGE) se encuentra el Módulo Generador de Reportes, el cual es de singular importancia pues las informaciones que la ONE les proporciona a las instituciones que demandan de estas son enviadas en forma de reportes de ahí que se necesite que la herramienta para la generación de los reportes estadísticos posea buena potencia y dinamismo.

Partiendo de la premisa de que la realización de un buen análisis y diseño ayudaría a la futura implementación del sistema informático propuesto por el Módulo Generador de Reportes, el presente trabajo recoge los resultados del análisis y diseño de las actividades básicas desarrolladas por el módulo, los principales artefactos que se generaron así como los resultados de investigaciones realizadas en cuanto a metodologías de desarrollo de software y herramientas generadoras de reportes. Por otra parte se ofrece una comparación entre el software utilizado actualmente en la ONE y el propuesto por el módulo así como la correspondiente valoración critica de dicha comparación, se realiza además un análisis de la factibilidad con el objetivo de validar las tareas realizadas.

Palabras Claves: Ingeniería de Software, Ingeniería de Requisitos, Generador de Reportes,

TABLA DE CONTENIDOS

Introducción 1
Situación Problémica 1
Métodos y técnicas de investigación a utilizar. 3
Resultados Esperados 3
Capítulo 1: Fundamentación Teórica 5
1.1 Introducción 5
1.2 Trayectoria de la creación de reportes estadísticos en Cuba 5
1.3 Estado del Arte de los Generadores de Reportes 6
1.4 Generadores de Reportes 6
1.4.1 Aspectos importantes en los reportes: 6
1.4.2 Generadores de Reportes utilizados en la actualidad 7
1.5 Análisis de las Metodologías para el Desarrollo de Software 10
1.5.1 Concepto de Metodología 10
1.5.2 Tipos de Metodologías 10
1.6 Metodologías de Software 11
1.6.1 Extreme Programing (XP) 11
1.6.2 Metodología Scrum 12
1.6.3 Microsoft Solution Framework (MSF). 13
1.6.4 Técnica de Modelado en Objeto (OMT) 15
1.7 Análisis de la metodología utilizada para desarrollar el módulo 16
1.8 Lenguajes de Modelado seleccionado 18
1.8.1 Historia del UML 18
1.9 Herramientas Case 19
1.9.1 Visual Paradigm 19
1.9.2 Rational Rose Enterprise Edition 20
1.10 Herramientas seleccionadas para el desarrollo 20
1.11 Plataforma seleccionada para el desarrollo 22
1.11.1 Net Framework 23
1.12 Visual Studio.Net 24
1.13 Lenguaje de Programación C# 25
1.14. El análisis 25
1.14.1 Principios del análisis 26
1.15 Gestión de riesgos 27
1.15.1 Riesgo 27
1.15.2 Fases de la Gestión de Riesgos 27
1.16 Ingeniería de requerimientos 27
1.16.1 Actividades 28
1.16.2 Técnicas principales 29
1.16.3 Especificación de requisitos del software 31
1.16.4 Identificación de las personas involucradas 31
1.16.5 Problemas 31

1.17 Diseño del software 32
1.17.1 Principios del diseño 32
1.18 Patrones 34
1.18.1 Categorías de patrones 35
1.18.2 Patrones de Diseño 35
1.18.3 Patrones de Arquitectura 38
1.19 Métricas de Software 39
1.19.1 Características de las métricas de software 39
1.19.2 Métricas del análisis. 40
1.19.3 Métricas del Diseño 40
1.20 Conclusiones 41
Capítulo 2: Descripción del Sistema 42
2.1 Introducción 42
2.2 Objetivos del Módulo Generador de Reportes 42
2.2.1 Objetivos Estratégicos del Módulo Generador de Reportes 42
2.3 Flujo actual de los procesos involucrados en el campo de acción 42
2.3.1 Gestionar Reportes 42
2.4 Objeto de Automatización 43
2.5 Modelo de Dominio 43
2.5.1 Conceptos identificados 43
2.6 Propuesta del sistema 44
2.6.1 Cómo debe trabajar el sistema 44
2.6.2 Análisis comparativo entre la solución existente y la propuesta 45
2.6.3 Aspectos novedosos que propone la herramienta del (SIGE-MGR) 46
2.6.4 Beneficios para la ONE el uso del (SIGE-MGR) 47
2.7 Especificación de requisitos del Sistema 48
2.7.1 Requisitos Funcionales 48
2.7.2 Requisitos No Funcionales 48
2.8 Modelo del Sistema 50
2.8.1 Actores del Sistema 51
2.8.2 Diagrama de Paquetes 51
2.9 Conclusiones 57
Capítulo 3: Análisis y Diseño 58
3.1 Introducción 58
3.2 Modelo de Análisis 58
3.2.1 Clases del Análisis 59
3.3 Diagramas de Interacción 60
3.3.1 Tipos de diagramas de interacción 60
3.4 Modelo de Diseño. 61
3.5 Diagrama de Clases del diseño por paquetes 62
3.5.1 Paquete Diseño 63
3.5.2 Paquete MGR 63
3.5.3 Paquete Consulta 64

3.6 Herramienta utilizada para la capa de acceso a datos 64
3.7 Patrones utilizados 65
3.8 Conclusiones 66
Capítulo 4: Análisis de la Factibilidad 67
4.1 Introducción 67
4.2 Estimación del esfuerzo basado en Casos de Uso 67
4.3 Métricas del Diseño 73
4.4 Métricas Orientadas a Clases 75
4.4.1 Serie de Métricas CK 75
4.5 Conclusiones 76
Conclusiones Generales 77
Recomendaciones 78
Referencias Bibliográfica. 79
Bibliografía 80
Glosario de Términos 81
Anexos 82

Yoanki Moreno Tamayo

Introducción

En el mundo actual, para desarrollar software se llevan a cabo procesos con un alto nivel de complejidad, esto obedece sin lugar a duda al desarrollo vertiginoso que ha experimentando en los últimos tiempos la rama de la informática y las comunicaciones.

La Oficina Nacional de Estadísticas (ONE) es la entidad Cubana encargada de gestionar todas las estadísticas inherentes al país ya sean económicas o de otro tipo, además dicha oficina tiene la tarea de controlar la difusión de estas estadísticas de acuerdo con las diferentes necesidades que presente el país, tales actividades son llevadas a cabo a través del Sistema Estadístico (SEN). La difusión de los datos estadísticos se realiza en forma de reportes, para lo cual el país posee una infraestructura de información, la cual está organizada de la siguiente forma.

El primer eslabón en la cadena, son los Centros Informantes (CI), estos no son mas que empresas o entidades a nivel municipal que brindan informaciones a las Oficinas Municipales de Estadísticas (OME) que son oficinas de estadísticas rectoras en cada municipio del país. Las OME además de su trabajo de gestión tiene la tarea brindar sus informaciones a las Oficinas Territoriales de Estadísticas (OTE), las cuales son responsable de controlar las informaciones a nivel provincial y de brindarle sus reportes a la ONE que es la máxima entidad en este entorno, el país cuenta con una OTE en cada provincia.

En estos momentos la Oficina Nacional de Estadísticas posee un sistema informático de control de información que no es capaz de satisfacer de forma plena todas las necesidades con que ella cuenta. Las herramientas utilizadas por La Oficina Nacional de Estadísticas para elaborar sus reportes en el área de estadísticas continuas, no cumplen con los requerimientos necesarios para realizar un trabajo adecuado, pues crean sus reportes accediendo a los bancos de datos en sus estructuras de carpetas y en ocasiones, cuando el cúmulo de información es muy grande, el proceso se torna lento y complejo, en la ONE también se almacenan reportes en formato duro, estos son fundamentalmente los que contienen información muy antigua que son de difícil digitación.

Situación Problémica

En la actualidad la Oficina Nacional de Estadísticas utiliza el software MicroSet para la gestión de los reportes estadísticos, este software cuya creación data de la década del 70 posee numerosas incompatibilidades con los sistema operativos que se usan en la actualidad, pues no funciona

correctamente bajo los sistemas operativos de la familia NT, por otra parte la interacción del usuario con este sistema es un tanto difícil debido a que su interfaz no es amigable, de la misma forma a la hora de crear los reportes el usuario tiene que usar comando de textos para llenar todo el diseño que llevan los mismos haciendo sumamente engorrosa esta tarea cuando el reporte es un poco extenso, igualmente este software no permite exportar los reportes a otros formatos, ni realizar conexiones dinámicas con bases de datos u otros servidores. Todos estos inconvenientes que presenta el MicroSet, afecta la calidad del reporte así como el tiempo de entrega del mismo lo cuál es fundamental en este trabajo de gestión estadística.

Para dar solución a estas dificultades se crea el SIGE, el cual cuenta con seis módulos de trabajo entre ellos el Módulo Generador de Reporte (MGR), que se encargara de desarrollar una herramienta que sustituya al MicroSet y ayude a estandarizar el trabajo con los reportes en la ONE, por lo que resulta de medular importancia detallar de forma correcta el análisis y diseño del módulo para con ello establecer puntos en común entre los clientes y desarrolladores que ayuden a traducir al lenguaje de estos últimos las necesidades de los primeros y potenciar una efectiva implementación de módulo, lo que conduce al planteamiento del siguiente **problema científico**: *¿Cómo realizar el análisis y diseño del Módulo Generador de Reportes para el sistema que automatiza los procesos de gestión de reportes de la Oficina Nacional de Estadísticas, de forma más flexible que posibilite la implementación del sistema sin ambigüedades*?, teniendo como **objeto de estudio**: *El proceso de desarrollo de software para sistemas generadores de reportes*, y **campo de acción**: *El análisis y el diseño del Módulo Generador de Reportes para automatizar procesos de gestión de reportes en la Oficina Nacional de Estadísticas*.

El **objetivo general** consiste en: *Desarrollar el análisis y el diseño del Módulo Generador de Reporte del Proyecto ONE*, conduciendo al planteamiento de la siguiente **hipótesis** *si se logra desarrollar el análisis y diseño del Módulo Generador de Reportes para la automatización de procesos de gestión de reportes en la Oficina Nacional de Estadísticas, entonces se facilitará una adecuada implementación de dicho Módulo*. Para dar cumplimiento al objetivo planteado se definen las siguientes **tareas de la investigación**:

- Estudiar el estado del arte de los generadores de reportes utilizados en la actualidad.
- Seleccionar las herramientas idóneas para llevar a cabo el trabajo en el módulo.
- Estudiar los principios del análisis para la validación de los requisitos.
- Estudiar las principales tendencias del diseño.
- Realizar un estudio sobre las métricas para la validación del análisis y diseño.

- Sostener de forma frecuente reuniones y entrevistas con el personal de la ONE que se encarga de realizar los reportes.

Métodos y técnicas de investigación a utilizar.

Durante el desarrollo de la investigación se utilizó la estrategia exploratoria para tener conocimiento acerca de la generación de los reportes, se consultaron diferentes fuentes para la investigación, algunas especializadas en el tema otras con experiencias en el asunto, también se consultaron documentos existentes sobre la temática.

Se utilizaron métodos de investigación, entre los que están el método Sintético a través del cual se llevó a cabo una investigación previa para conocer la forma en que se generaban los reportes en la Oficina Nacional de Estadísticas (ONE) y lograr unificar un cúmulo de ideas importantes, para materializar el trabajo en el módulo, el método Histórico Lógico permitió constatar la evolución de los generadores de reportes, las metodologías de desarrollo de software y las herramientas utilizadas durante el trabajo, se utilizó el método de la modelación el cual permitió la creación de modelos que muestran una reproducción simplificada de la actualidad que proporcionó nuevas cualidades del objeto de estudio. Dentro de los métodos empíricos utilizados se tiene la entrevista, la cuál fue utilizada fundamentalmente para coleccionar la mayor cantidad de información posible sobre la realización del proceso de gestión de reportes en la ONE.

Resultados Esperados

- Especificación de requisitos.
- Especificación de Casos de Uso.
- Modelo de Análisis
  - Diagrama de clases del análisis
  - Diagramas de interacción
- Modelo de Diseño
  - Diagramas de clases de diseño
- Resultados de las métricas y métodos aplicados

El presente trabajo ha sido estructurado de la siguiente forma:

Capítulo 1. Fundamentación Teórica: En este capitulo se analizan algunos conceptos importantes para la comprensión del trabajo, se realiza un estudio del estado del arte de las herramientas generadoras de reportes así como de las metodologías de desarrollo de software, además se escogen las herramientas para el desarrollo del módulo, se profundiza en la descripción de la ingeniería de requisitos y del análisis y diseño.

Capítulo 2. Descripción del Sistema: Se muestran características importantes del sistema, los conceptos del dominio y el correspondiente modelo de dominio, se exponen los requisitos con que debe cumplir el sistema, se realiza una comparación entre la herramienta existente y la nueva propuesta, se obtiene como artefacto más importante el diagrama de Casos de Uso del Sistema.

Capítulo 3. Análisis y Diseño: Contiene los artefactos necesarios que se obtuvieron durante el flujo de trabajo análisis y diseño. Se exponen los patrones utilizados en el trabajo del módulo.

Capítulo 4. Análisis de La Factibilidad: Se aplican métricas y métodos que ayudarán a estimar el esfuerzo realizado en el módulo y que proporcionará mayor seguridad sobre la calidad del sistema, se muestran los resultados de las métricas aplicadas.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

El presente capítulo muestra una exposición de los conceptos y aspectos teóricos asociados al dominio del problema, que permitirá una mejor comprensión del tema tratado, también se exponen las características de varias herramientas utilizadas a nivel mundial para la generación de reportes estadísticos y de las metodologías de desarrollo de software más conocidas en la actualidad. Se analizan plataformas de desarrollo utilizadas y características relevantes del análisis y diseño.

1.2 Trayectoria de la creación de reportes estadísticos en Cuba

En materia de estadísticas los archivos más antiguos que se conocen en Cuba datan desde los tiempos de la colonia y el primer anuario cubano tiene su origen por los años 1920. Debido a la dinámica que supone el trabajo estadístico era de medular importancia la introducción de máquinas y medios de procesamiento que favorecieran el desarrollo de este trabajo.

En los inicios los reportes estadísticos eran realizados por analistas y especialistas haciendo uso del papel, este era el principal medio de creación de los reportes así como de los modelos estadísticos, más tarde con la aparición de máquinas de dibujos y el diseño de croquis por los diseñadores aumentó un poco la calidad de los reportes aunque seguía usándose el papel para su creación. Posteriormente fueron apareciendo algunos sistemas que hacían más fácil el manejo de las estadística el más notable fue el sistema Flow.

El Sistema Unificado para Reportes Estadísticos (SUPRE) fue un paso grande en aras de utilizar un sistema integral de gestión estadística, pues este sistema permitía realizar un trabajo más integral con los reportes pero todavía presentaba grandes problemas de eficiencia, SUPRE se puede considerar como un ancestro de SetaMot (Sistema Estadísticos para Tablas de los Modelos) el cual brindaba la posibilidad de realizar tareas estadísticas al gusto de los especialistas, los cuales podían crear un reporte con opciones más configurables y personalizadas.

El software más completo usado en Cuba para la gestión estadística es el MicroSet cuyo objetivo desde su primera edición en la década del 70 era proporcionar un sistema general e integral para el procesamiento de datos y las ediciones de tablas para con ello facilitar y mejorar el proceso de gestión de información sobre las estadísticas económicas almacenadas en la ONE, MicroSet permite crear

reportes con características personalizadas, posibilita listar, guardar y eliminar reportes, permite el trabajo en red de forma eficiente, el uso de las impresoras desde varias estaciones de trabajo entre otras actividades inherentes a la estadística.

1.3 Estado del Arte de los Generadores de Reportes

En el mundo contemporáneo donde las tecnologías alcanzan niveles ilimitados las empresas son cada ves más dependientes de estas, no se concibe una entidad en la cual no este inmerso un programa de software para mejorar su funcionamiento o para gestionar sus intereses, el avance tecnológico experimentado ha sido capaz de sustituir cientos de manos obreras por un programa de software. La Gestión Estadística no ha estado exenta de este insoslayable proceso y en la actualidad cuentan con una buena cantidad de programas bien estructurados capaces de proporcionar magníficos dividendos para sus usuarios. Una de las acciones más relevantes que se lleva a cabo en el proceso de gestión estadística es la de generar reportes con la cual se demuestra la eficiencia y la calidad de las aplicaciones realizadas.

1.4 Generadores de Reportes

Los generadores de reportes son herramientas complementarias de los sistemas de información. Utilizan una especie de lenguaje transparente para el usuario por medio del cual éste realiza consultas a la base de datos y obtiene información de ella en forma de reportes. **(6)**

1.4.1 Aspectos importantes en los reportes:

Definición del reporte: Momento en que el autor del reporte define los datos y la manera de presentación de estos. En esta etapa normalmente hay que definir conexiones a los distintos orígenes de datos para ver de donde obtener los resultados que debe reflejar el reporte. **(6)**

Administración del reporte: Está referido al hecho de que las organizaciones actuales tienen distintas categorías de usuarios por ejemplo, los gerentes, los usuarios de servicio al cliente, etc. Por lo tanto, es importante definir quienes serán los usuarios del reporte, para ello hay que publicar los reportes. **(6)**

Entrega del reportes: Es muy común en las organizaciones que muchos reportes sean requeridos de manera periódica, por ejemplo, el reporte de ventas diarias debe estar en la oficina del gerente de ventas todas las tardes a las 5 pm, o un reporte de inventario todos los fines de semana, se podría entonces aprovechar distintos servicios como el de mensajería para que estos reportes lleguen a los usuarios requeridos. **(6)**

Las tres acciones, mencionadas anteriormente, conforman lo que se denomina **"El Ciclo de Vida de un Reporte"**

1.4.2 Generadores de Reportes utilizados en la actualidad

- **El Rpv Printing System**

Herramienta que posibilita generar reportes sin discriminar lenguajes de programación y de la misma forma permite ver/imprimir bajo Windows reportes generados desde un servidor que puede estar corriendo en Windows, NT, Unix, Linux etc. Rpv es una excelente herramienta orientada hacia aquellos desarrolladores que necesitan producir reportes de calidad gráfica Windows, desde un lenguaje de programación basado en Windows, basado en DOS o bien desde cualquier lenguaje de programación. Lo que Rpv hace es leer simples archivos de texto ó ASCII y transformarlos en atractivos reportes con calidad gráfica Windows. Si eres un desarrollador y puedes escribir un archivo de texto con tu lenguaje de programación favorito, entonces Rpv puede servirte a ti también. Rpv es capaz de pre visualizar los reportes utilizando el visualizador o bien enviar los mismos directamente a la impresora en aquellos casos en los que los reportes no deben ser pres visualizados. **(4)**

- **Crystal Reports**

Esta potentísima herramienta que posee diferentes versiones es la herramienta de elaboración de informes estándar para Visual Studio .NET, la cuál permite crear contenido interactivo con una excelente calidad de presentación, con Crystal Reports puede almacenarse informes en aplicaciones de escritorio y web, además se puede publicar informes de Crystal como servicios Web de informes en un servidor Web.

Puede crear una aplicación Web que permita a los usuarios profundizar en un gráfico y filtrar la información en función de sus necesidades. Realmente, el gráfico es un informe de Crystal que interactúa con otros controles de la aplicación. Cristal es un producto creado en su esencia orientado al usuario final, es decir, que un ejecutivo puede crear sus propios informes sin necesidad de asistencia de un desarrollador. **[3]**

- **FastReportStudio**

Es un ambiente potente, compacto y flexible para darle a tu aplicación la capacidad de generar informes rápida y eficientemente. Todas las herramientas que necesitas para desarrollar informes rápidos y profesionales incluyendo motor de informes, diseñador de informes, visor de vista previa,

diseñador de ventana de diálogo, y cuatro macro intérpretes-estilo Basic (VB), estilo C++, estilo JS y estilo Pascal. **(6)**

- **moReport**

moReport es un generador de reportes con soporte para PostgreSQL (está pensado añadirle soporte para más Servidores de Base de Datos) con salida en formato Postscript (también se añadirá soporte para otros formatos de salida). El formato de entrada para moReport se basa en el standar XML. **(**8**)**

¿Que hace moReports?

moReport tiene las estructuras básicas para el diseño de un Informe:

- Encabezado de Página.
- Pie de Página.
- Encabezado de Reporte.
- Pie de Reporte.
- Sección Detalles.
- Admite la creación de más de un grupo. En el cuál se puede definir distintos componentes**:**
- Campos relacionados con una consulta SQL a una Base de Datos.
- Etiquetas de Texto.
- Inserción de Líneas.

- **El Reporting Services**

Es una plataforma de reportes basada en servidores, que puede ser empleada para crear y administrar reportes tabulares, de matrices, gráficos y de libre formato, la información de estos reportes pueden provenir de diferentes orígenes de datos. Los reportes que se definen pueden ser administrados a través de una conexión basada en Web. Reporting Services provee servicios, herramientas e interfaces de programación (API), aunque no es necesario ser desarrollador para usarlo. El Reporting Services tienes varias ventajas entre las que se pueden mencionar:

- Cuenta con una interface Web para la administración de los reportes, desde esta interface se puede determinar en que formato debe llegar el reporte, posibilita decidir que el reporte llegue a una de las gerencias en formato PDF y para el departamento de consolidación y validación de datos podría enviarse la información en formato XML.
- Con SQL Server Reporting Services, se puede conectar a cualquier repositorio de datos, a través de un .NET Data Provider, un proveedor OLE DB provider o uno de tipo ODBC.

- Para la distribución, los usuarios pueden acceder a los reportes en base a la infraestructura existente es decir estos acceden a los reportes a través de una barra de herramientas en el browser.
- Otra de las grandes características de Reporting Services, es que puede distribuir el reporte en distintos formatos, como hojas de Excel, documentos pdf, texto, XML, etc.
- La arquitectura de Reporting Services, permite a los desarrolladores preparar aplicaciones personalizadas que accedan a los reportes a través de una API que esta expuesta como un Web service. **(6)**

1.4.2.1 Generador de Reporte Seleccionado

Después de realizar un estudio del arte de varios generadores de reportes se llegó a la conclusion que el utilizado será el ActiveReports para .NET.

- **ActiveReports. NET**

Está escrito y completamente gestionado en Visual C # lo que proporciona una completa integración con el IDE Visual Studio. NET. De este modo, los desarrolladores pueden trabajar con lenguajes de programación como Visual C# o Visual Basic. ActiveReports. NET se licencia por cada desarrollador, y su distribución es gratuita. ActiveReports. NET viene con un asistente de acceso de información que posibilita que la importación de datos sea muy sencilla. Una de las características más importante que ActiveReports. NET presenta es que incluye filtros para exportar a formatos populares como Adobe PDF, Microsoft Excel, RTF, HTML, Texto y TIFF, tanto en aplicaciones de escritorio como Web. **(1)**
ActiveReports incluye un cuadro de control, que soporta gráficos en 2D y 3D con características muy avanzadas que permiten exportar a diversos formatos de imagen. El Visor de control de Active posibilita una vista previa de múltiples pestañas, en las que se pueden observar hipervínculos, tabla de contenidos, miniaturas, búsquedas de textos, anotaciones entre otras y la barra de herramientas es personalizada.
Otras características de ActiveReports

- Microsoft Visual Studio .NET 2003, Visual Studio 2005, or Visual Studio 2008
- Windows NT 4.0, Windows 2000, Windows XP, Windows Vista or Windows 2003 Server
- Microsoft .NET Framework v1.1, v2.0, v3.0, or v3.5
- Windows 98, Windows NT 4.0, Windows 2000, Windows XP, Windows Vista or Windows 2003 Server.
- Microsoft .NET Framework v1.1, v2.0, v3.0, or v3.5

- Visor de Anotaciones
- Código-detrás de diseños de informes
- Diseño de datos en tiempo fuente de apoyo
- Familiar interfaz de usuario
- Proporciona al usuario final la capacidad de edición de informe
- Presenta ventana que permite tirar consultas para acceder a la base de datos

1.5 Análisis de las Metodologías para el Desarrollo de Software

En la actualidad cuando la mayoría de los países del orbe así como las más trascendentales compañías apuestan por el desarrollo en el mundo informático, controlar las producciones es una de las tareas más sustanciales y complicadas que existen pues los procesos aumentan su complejidad conforme su importancia y resulta verdaderamente difícil tener un dominio total de cada uno de ellos, es por eso que en varias ocasiones han tenido que retirarse del mundo informático varias compañías dejando solamente la huella de un intento frustrado de incluirse en el competitivo mundo de la producción de software.

La necesidad de establecer métodos ordenados y exactos para evaluar las producciones de software se hace cada vez más latente con el objetivo de proporcionar soluciones a problemas habituales en la producción de software como pueden ser, la entrega atrasada de productos a los clientes, el exceso en el costo del producto dejando pérdida para los productores, o el no cumplimiento del total de las funcionalidades que el cliente señaló.

1.5.1 Concepto de Metodología

Metodología: Se refiere a los métodos de investigación que se siguen para alcanzar una gama de objetivos en una ciencia, es decir son el conjunto de métodos que se rigen en una investigación científica o en una exposición doctrinal. **[7]**

1.5.2 Tipos de Metodologías.

Metodologías Agiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el Proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es	Existe un contrato prefijado

bastante flexible	
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Tabla1.1. Comparación entre metodologías

"Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se lleva una metodología de por medio, lo que se obtiene son clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos**" (5)**

1.6 Metodologías de Software

1.6.1 Extreme Programing (XP)

- **XP**

Es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizada para proyectos de corto plazo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

Características de XP

La metodología **XP** se basa en:

- **Pruebas unitarias**

Se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se pueden hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.

- **Re fabricación**

Se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

- **Programación en pares**

Una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es algo parecido al chofer y el copiloto, mientras uno conduce, el otro consulta el mapa.

¿Qué es lo que propone XP?

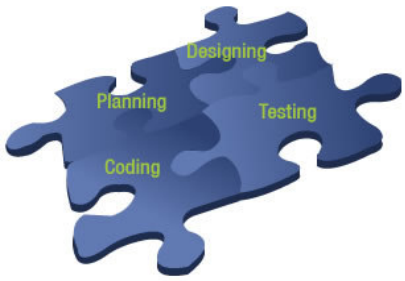


Figura1.1. Extreme Programing

- Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- El manejo del cambio se convierte en parte sustantiva del proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierten en miembro del equipo. **(5)**

1.6.2 Metodología Scrum

- **Scrum**

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle, define un marco para la Gestión de Proyectos, esta metodología se ha utilizado con éxito en los últimos 10 años. Está especialmente indicada para proyectos con rápido cambio de requisitos, sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones denominadas sprints, con una duración de 30 días, el resultado de cada sprints es un incremento ejecutable que se muestra al cliente, al final de cada sprint se realiza un Sprint Review para evaluar los artefactos construidos y comentar el plan para el próximo sprint, la segunda característica importante son las reuniones a lo largo del proyecto (Scrum Daily Meeting), convocada por el Scrum Master que será el líder del proyecto la cuál tiene como objetivo proporcionar una realimentación sobre las tareas de los recursos y los obstáculos que se presentan, esta reunión no excederá los 15 minutos de duración.

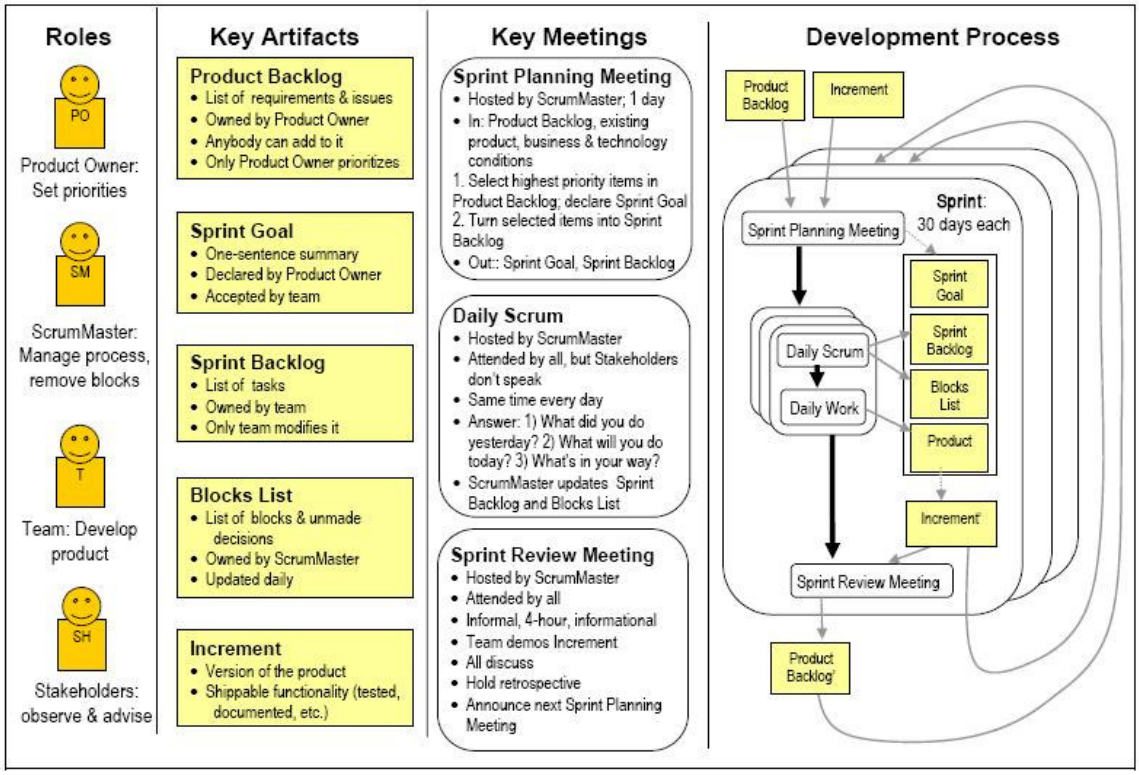


Figura1.2.Procesos de desarrollo de Scrum

1.6.3 Microsoft Solution Framework (MSF).

- **MSF**

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

Figura1.3. MSF

MSF tiene las siguientes características:

- **Adaptable**

Es parecido a un compás, usado en cualquier parte como un mapa, del cuál su uso es limitado a un específico lugar.

- **Escalable**

Puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas o más.

- **Flexible**

Es utilizada en el ambiente de desarrollo de cualquier cliente.

- **Tecnología agnóstica**

Porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

- **Modelo de Arquitectura del Proyecto**

Diseñado para acortar la planificación del ciclo de vida. Este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones.

- **Modelo de Equipo**

Este modelo ha sido diseñado para mejorar el rendimiento del equipo de desarrollo. Proporciona una estructura flexible para organizar los equipos de un proyecto. Puede ser escalado dependiendo del tamaño del proyecto y del equipo de personas disponibles.

- **Modelo de Proceso**

Diseñado para mejorar el control del proyecto minimizando el riesgo y aumentar la calidad, acortando el tiempo de entrega. Proporciona una estructura de pautas a seguir en el ciclo de vida del proyecto, describiendo las fases, las actividades, la liberación de versiones y explicando su relación con el Modelo de equipo.

- **Modelo de Gestión del Riesgo**

Diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir. Este modelo proporciona un entorno estructurado para la toma de decisiones y acciones valorando los riesgos que puedan provocar.

- **Modelo de Diseño del Proceso**

Diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario. Proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo. Las fases de diseño conceptual, lógico y físico proveen tres perspectivas diferentes para los tres tipos de roles: los usuarios, el equipo y los desarrolladores.

- **Modelo de Aplicación**

Diseñado para mejorar el desarrollo, el mantenimiento y el soporte, proporciona un modelo de tres niveles para diseñar y desarrollar aplicaciones de software. Los servicios utilizados en este modelo son escalables y pueden ser usados en un solo ordenador o incluso en varios servidores.

1.6.4 Técnica de Modelado en Objeto (OMT)

- **OMT**

Es una metodología de diseño clásico que ha servido como base para UML. Es orientada a objetos y fue desarrollada por James Rambaugh y Michael Blaha en 1991. Se hace cargo de todo el ciclo de vida del software, está divida en cuatro fases consecutivas centrándose en la primera que es la fase de análisis de objetos y de la cual depende el buen desarrollo de las siguientes, tiene dos fases de diseño no muy complejas y una última fase de implementación donde se codifica lo ya diseñado.

- Análisis de objetos.
- Diseño del sistema.
- Diseño de objetos.
- Implementación.

La fase de análisis se inicia con la descripción del problema a resolver en el cuál se elabora una lista de requisitos a cumplir y conceptos principales definidos para el entorno del problema a solucionar. A partir de la descripción del problema se elaboran tres modelos fundamentales: Modelo de Objetos, Modelo Dinámico y Modelo Funcional. Concluida esta fase se prosigue a la fase de diseño en la cuál se realiza el Diseño del sistema donde se define la arquitectura que va a tener el mismo, luego está la fase de Diseño de Objetos donde se crea el plan de implementación, definiendo las clases de los objetos y poniendo una fuerte atención a la persistencia de datos. Por último está la fase de implementación donde se implementa todo el sistema en correspondencia directa con el diseño.

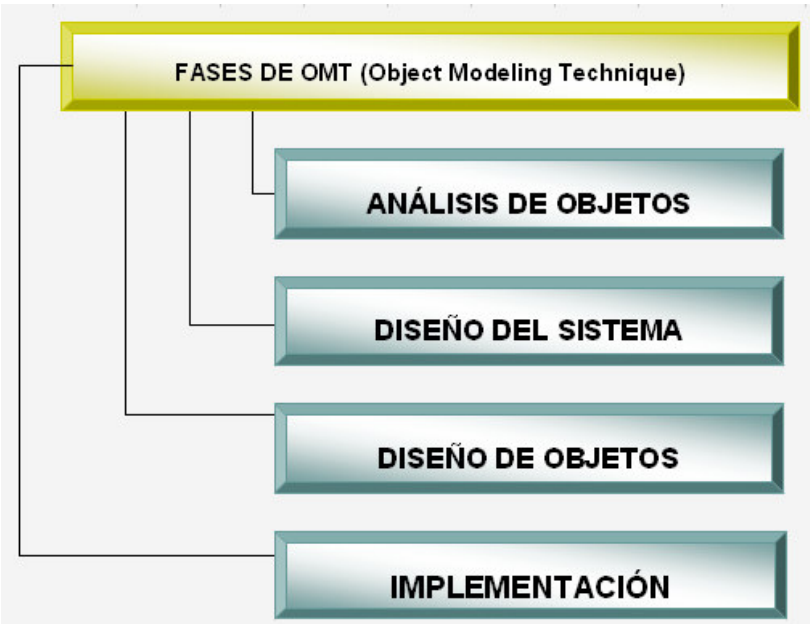


Figura1.4. Fases de Técnica de Modelado en Objeto

1.7 Análisis de la metodología utilizada para desarrollar el módulo

Se usará la metodología **RUP** (Rational Unified Process), la misma fue escogida por la dirección del proyecto por su posibilidad de adaptación y configuración a proyectos con un período de duración considerable y por ser una metodología robusta y bien definida probada anteriormente en otros proyectos con características similares al de la ONE. **RUP** no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. Es un proceso de desarrollo de software, o sea, un conjunto de actividades para transformar los requisitos de un sistema de software.

La metodología en cuestión tiene entre sus principales características que esta dirigida por casos de uso, es centrada en la arquitectura y además iterativo e incremental. Incluye artefactos, que son los

productos tangibles del proceso, y roles, que no es mas que el papel que desempeña una persona en un determinado momento.

- **Dirigido por Casos de Uso**

Se define como caso de uso al conjunto de acciones que puede realizar un sistema para dar un resultado de valor a un determinado usuario. Se obtiene cuando se modela el proceso del negocio y se representa a través de los requisitos.

- **Centrado en la Arquitectura**

La arquitectura muestra la visión común del sistema completo, describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo.

- **Iterativo e Incremental**

Es una secuencia de actividades con un plan establecido y un criterio de evaluación que hace referencia a pasos en el flujo de trabajo y los incrementos al crecimiento del producto.

Para preparar todos los esquemas de un sistema de software la metodología de **RUP** utiliza el Lenguaje Unificado de Modelado (**UML**) que es una parte esencial del Proceso Unificado.

La metodología **RUP**, llamada así por sus siglas en inglés Rational Unified Process, divide en 4 fases el desarrollo del software:

Inicio: El Objetivo en esta etapa es determinar la visión del proyecto.

Elaboración: En esta etapa el objetivo es determinar la arquitectura óptima.

Construcción: En esta etapa el objetivo es obtener la capacidad operacional inicial.

Transición: El objetivo es llegar a obtener el release del proyecto.

Los elementos del RUP son:

Actividades: Son los procesos que se llegan a determinar en cada iteración.

Trabajadores: Son las personas o entes involucrados en cada proceso.

Artefactos: Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

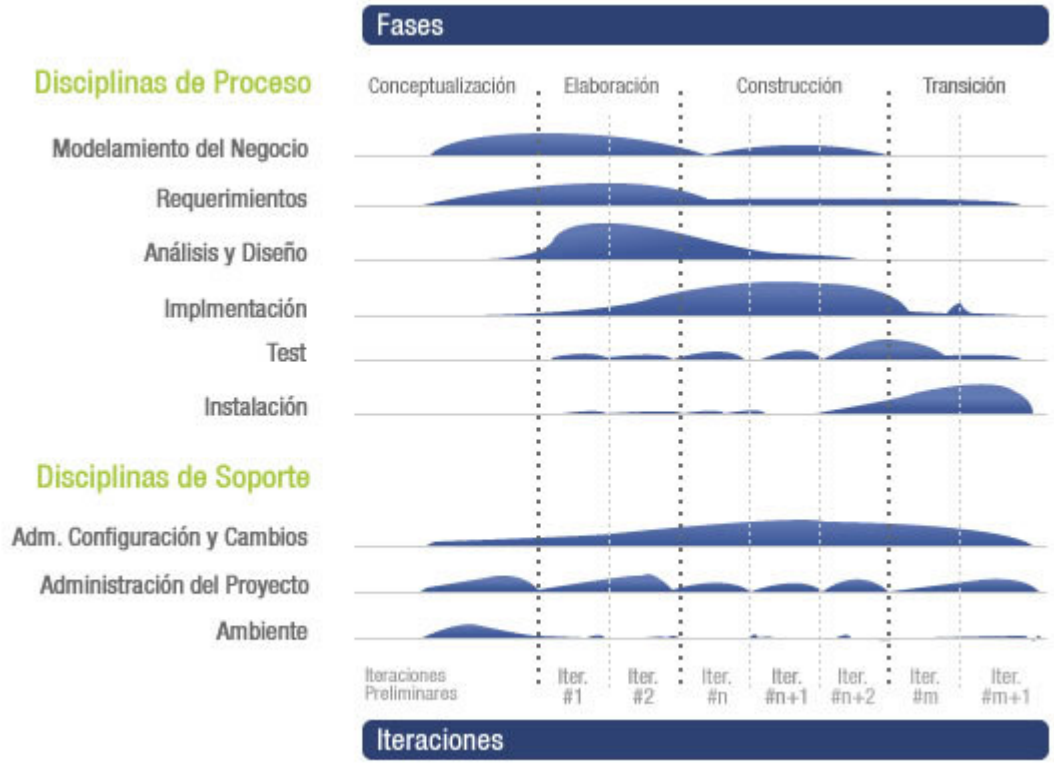


Figura1.5. Fases e iteraciones de la metodología Rup

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. **(5)**

1.8 Lenguajes de Modelado seleccionado

El lenguaje de modelado seleccionado para trabajar fue el Lenguaje Unificado de Modelado (**UML**), sus siglas en ingles Unified Modeling Language.

1.8.1 Historia del UML

Durante los ochenta y principios de los noventa Grady Booch, James Rumbaugh, e Ivar Jacobson trabajaban por separado en desarrollo de notaciones para el análisis y diseño de sistemas orientados a objetos. Los tres llegaron por separado a obtener bastante reconocimiento.

Booch había escrito "**Object-Oriented Analysis and Design with Applications**" un libro de referencia en el análisis y diseño orientado a objetos desarrollando su propia notación.

Por su parte James Rumbaugh había desarrollado su propia notación de diseño orientado a objetos llamada OMT (**Object Modeling Technique**) en su libro "**Object-Oriented Modeling and Design**".

Por otro lado Jacobson se había revelado como un visionario del análisis (padre de los casos de uso) y sobre todo del diseño orientado a objetos, sorprendiendo a todo el mundo en "**Object-Oriented Software Engineering: A Use Case Driven Approach**". A mediados de los noventa empezaron a intercambiar documentos y trabajar en conjunto produciendo grandes avances en el modelado de sistemas orientados a objetos. En 1994 Rational contrató a Rumbaugh en donde ya trabajaba Booch, un año después Jacobson se unía a ellos en Rational. En 1997 salió a la luz la versión 1.0 de UML. **[2]**

UML es un conjunto de herramientas, que permite modelar analizar y diseñar sistemas orientados a objetos

Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Ofrece un estándar para describir un panorama del sistema modelo, incluyendo aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables y aspectos conceptuales como los procesos de negocios y funciones del sistema.

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. Es importante recalcar que UML es un "lenguaje" para especificar y no un método o un proceso.

UML se divide fundamentalmente en dos partes: vistas y diagramas. Las vistas son una abstracción que muestra un aspecto particular del sistema. Ejemplo, vista de casos de uso, de diseño, de implementación, de procesos, entre otras. Los diagramas son una representación gráfica de un conjunto de elementos que visualizan un sistema desde diferentes perspectivas y se agrupan en tres grupos fundamentales: de estructura estática, de comportamiento y de implementación.

1.9 Herramientas Case

1.9.1 Visual Paradigm

Visual Paradigm es una herramienta muy potente y fácil de utilizar para crear los artefactos necesarios en la confección de un Software. Está diseñado para varios tipos de usuarios, incluyendo Ingenieros de Software, Analistas de Sistemas, Analistas de Negocio y Arquitectos de Sistemas. Es una suite completa de herramientas CASE que da soporte al modelado visual con UML 2.0 ofreciendo distintas perspectivas del sistema. Independiente de la plataforma y dotada de una buena cantidad de productos o módulos para facilitar el trabajo durante la confección de un software así como garantizar la calidad del producto final.

Es posible generar código desde Visual Paradigm para plataformas como .NET, Java y PHP, así como obtener diagramas a partir del código, esto es de gran utilidad pues ahorra tiempo a los desarrolladores y reduce las posibilidades de cometer errores.

Brinda la posibilidad de obtener una Base de Datos relacional y el código necesario para acceder a esta a partir del diagrama de un Diagrama Entidad Relación, además se conecta fácilmente a varios servidores de base de datos. Se integra con varios ambientes de desarrollo integrados (IDE) lo cual permite pasar del código al modelado y viceversa. Establece interoperabilidad con otras aplicaciones como el Visio y el Rational Rose y documentar todo el trabajo y especificaciones de CU sin necesidad de utilizar herramientas externas, por ejemplo editores de texto, utilizando plantillas que se encuentran o que pueden ser creadas por los usuarios. Disponible en múltiples lenguajes y plataformas: Microsoft Windows (98, 2000, XP, o Vista) Linux, Mac OS X, Solaris o Java. **[12]**

1.9.2 Rational Rose Enterprise Edition

Esta herramienta CASE creada por los ingenieros (Booch, Rumbaugh y Jacobson) es de fácil utilización. Permite una modelación absoluta de los procesos del negocio y del sistema, además resulta de gran utilidad para los desarrolladores de proyectos, pues ella cubre todo el ciclo de vida del proyecto desde su fase inicial hasta su culminación, esta herramienta posibilita establecer una trazabilidad entre los modelos (análisis y diseño) y el código ejecutable, también permite el desarrollo de software en equipo basado en metodología como RUP, cada rol tiene su propia vista de arquitectura (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y funcionalidad del sistema en construcción. Cada analista, desarrollador o diseñador puede usar Rational Rose para definir y comunicar el negocio, el diseño y la arquitectura de la aplicación que se este desarrollando. Es una completa solución para mostrar de forma gráfica el análisis de los procesos del negocio y los requerimientos del sistema. **(3)**

1.10 Herramientas seleccionadas para el desarrollo

La herramienta utilizada para la modelación fue el **Enterprise Architect (EA).**

EA es una herramienta progresiva que cubre todos los aspectos del ciclo de desarrollo, proporcionando una trazabilidad completa desde la fase inicial del diseño a través del despliegue y mantenimiento. También provee soporte para pruebas, mantenimiento y control de cambio. Algunas de las características claves de Enterprise Architect son:

- **Altamente eficaz**

Enterprise Architect es una herramienta comprensible de diseño y análisis UML, que cubre el desarrollo de software desde la captura de requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. EA es una herramienta de multi-usuarios, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener. Además, ofrece salida de documentación flexible y de alta calidad.

- **Velocidad, estabilidad y rendimiento**

El Lenguaje Unificado de Modelado provee beneficios significativos para ayudar a construir modelos de sistemas de software rigurosos y donde es posible mantener la trazabilidad de manera consistente. Enterprise Architect soporta este proceso en un entorno fácil de usar, rápido y flexible.

- **Trazabilidad de extremo a extremo**

Enterprise Architect provee trazabilidad completa desde el análisis de requerimientos y los artefactos de diseño, a través de la implementación y el despliegue. Combinados con la ubicación de recursos y tareas incorporados, los equipos de administradores de proyectos y calidad están equipados con la información que ellos necesitan para ayudarles a entregar los proyectos en tiempo.

- **Administración de complejidad**

EA ayuda a **administrar la complejidad** con herramientas para rastrear las dependencias, soporte para modelos muy grandes, control de versiones con proveedores CVS o SCC, líneas bases por cada punto en el tiempo, la capacidad de comparar diff para seguir los cambios del modelo, interfaz intuitiva y de alto rendimiento con una vista del proyecto como un "explorador".

- **Generación de documentos**

EA provee **una generación poderosa de documentos** y herramientas de reporte con un editor de plantilla completo *WYSIWYG*. Genera reportes detallados y complejos de EA con la información que usted necesita en el formato que su compañía o cliente demanda.

- **Ingeniería inversa**

EA soporta la **generación e ingeniería inversa de código fuente** para muchos lenguajes populares, incluyendo C++, C#, Java, Delphi, VB.Net, Visual Basic, ActionScript y PHP. También hay disponibles Add-ins gratis para CORBA y Python. Para aquellos que trabajan en Eclipse o Visual Studio.Net, Sparx Systems también existen puentes livianos para estas IDE's, permitiéndole modelar en EA y saltar directamente al código fuente en su editor preferido.

- **Visualización de aplicaciones**

EA le ayuda a **visualizar sus aplicaciones** soportando la ingeniería inversa de un amplio rango de lenguajes de desarrollo de software y esquemas de repositorios de base de datos.

- **Soporte de arquitectura**

EA soporta transformaciones **de Arquitectura avanzada dirigida por modelos** (MDA) usando plantillas de transformaciones fáciles de editar y desarrollar. Con las transformaciones incorporadas para DDL, C#, Java, EJB y XSD, puede desarrollar rápidamente soluciones complejas desde los "modelos independientes de plataforma" (MIP) simples que son el objetivo en los "modelos específicos de plataforma" (MEP). Un MIP se puede usar para generar y sincronizar múltiples MIP's - proporcionando un aumento de productividad significativo. **((**5**)**

EA posee soporte para 13 diagramas UML e incorpora dos más extendidos		
Diagramas Estructurales	**Diagramas de Comportamiento**	**Extendidos**
Clase	Casos de Uso	Análisis (actividad simple)
Objeto	Comunicación	Personalizado (para requisitos)
Compuesto	Interacción	
Paquete	Actividades	
Componente	Estado	
Despliegue	Tiempo	

Tabla1.2. Diagramas soportados por EA

1.11 Plataforma seleccionada para el desarrollo

La plataforma de desarrollo seleccionada fue la plataforma .Net ya que reúne en una misma plataforma un conjunto interesante de características, como independencia de plataforma, independencia de lenguaje, soporte de bases de datos, soporte para XML, servicios Web y aplicaciones Web, entre otras.

.NET es la plataforma de Microsoft para servicios Web XML, se dice que es la siguiente generación de software que conecta nuestro mundo de información, dispositivos y personas de una manera unificada y personalizada. **[9]**

1.11.1 .Net Framework

Admite la creación y la ejecución de aplicaciones para la plataforma .NET, tanto servicios Web como aplicaciones de consola, ventanas, servicios de Windows NT, etc. Posee un conjunto de clases Framework Class Library (FCL). Estás librerías ofrecen un gran número de posibilidades y funcionalidades en el desarrollo de aplicaciones modernas. Para ello dispone de librerías orientadas al manejo de colecciones, XML, comunicaciones, IO, multihilo, bases de datos. El Framework posee ventajas como: código administrado, el CLR realiza un control automático del código para que este sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente. El código puede ser escrito en cualquier lenguaje compatible con .Net ya que siempre se compila en código intermedio (MSIL). Otra característica es la compilación just-in-time, el compilador JIT incluido en el Framework compila el código intermedio (MSIL) generando el código máquina propio de la plataforma. Se aumenta así el rendimiento de la aplicación al ser específico para cada plataforma. El CLR es el encargado de gestionar la ejecución de las aplicaciones, proveyendo de servicios como compilación JIT, gestión de memoria, gestión de excepciones, depuración, seguridad o gestión de permisos.

Ventajas más importantes que proporciona .Net Framework:

- **Código administrado**

El CLR realiza un control automático del código para que este sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente.

- **Interoperabilidad multilenguaje**

El código puede ser escrito en cualquier lenguaje compatible con .Net ya que siempre se compila en código intermedio (MSIL).

- **Compilación just-in-time**

El compilador JIT incluido en el Framework compila el código intermedio (MSIL) generando el código máquina propio de la plataforma. Se aumenta así el rendimiento de la aplicación al ser específico para cada plataforma.

- **Garbage collector**

El CLR proporciona un sistema automático de administración de memoria denominado recolector de basura (garbage collector). El CLR detecta cuándo el programa deja de utilizar la memoria y la libera automáticamente. De esta forma el programador no tiene por que liberar la memoria de forma explícita aunque también sea posible hacerlo manualmente (mediante el método disponse() se libera el objeto para que el recolector de basura lo elimine de memoria).

- **Seguridad de acceso al código**

Se puede especificar que una pieza de código tenga permisos de lectura de archivos pero no de escritura. Es posible aplicar distintos niveles de seguridad al código, de forma que se puede ejecutar código procedente del Web sin tener que preocuparse si esto va a estropear el sistema.

- **Despliegue**

Por medio de los ensamblados resulta mucho más fácil el desarrollo de aplicaciones distribuidas y el mantenimiento de las mismas. El Framework realiza esta tarea de forma automática mejorando el rendimiento y asegurando el funcionamiento correcto de todas las aplicaciones. **(10)**

1.12 Visual Studio.Net

El sistema de desarrollo Microsoft Visual Studio es un conjunto de herramientas de desarrollo diseñadas para ayudar a los desarrolladores de software (tanto si son principiantes como profesionales con experiencia) a enfrentarse a los desafíos complejos y crear soluciones innovadoras. La función de Visual Studio es mejorar el proceso de desarrollo y facilitar el trabajo necesario para lograr grandes avances y hacerlo con mayor satisfacción.

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros. Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión 6). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles. El .NET Framework y VS.NET proporcionan una completa

herramienta, eficaz y sofisticada, para diseñar, desarrollar, depurar e implementar aplicaciones seguras. **(**4**)**

1.13 Lenguaje de Programación C#

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO. **(**9**)**

Visual C # ofrece una serie de ventajas basadas fundamentalmente en aspectos tales como su sencillez, pues elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Es moderno porque incorpora en el propio lenguaje elementos de otros, por ejemplo: un tipo básico decimal, una instrucción foreach, un tipo básico string y la distinción de un tipo bool específico para representar valores lógicos. Orientado a objetos pues no admite ni funciones ni variables globales, todo el código y datos han de definirse dentro de definiciones de tipos de datos, añade un modificador llamado internal, sólo admite herencia simple de clases. Permite definir propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) y atributos (información sobre un tipo o sus miembros). El lenguaje permite la gestión automática de memoria.

Todo lenguaje de .NET tiene a su disposición el recolector de basura del CLR. Sin embargo, dado que la destrucción de los objetos a través del recolector de basura es indeterminista y sólo se realiza cuando éste se active, C# proporciona un mecanismo de liberación de recursos determinista a través de la instrucción using. C# es muy eficiente, todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo permite marcar regiones de código como inseguras (modificador unsafe) y podrán usarse en ellas punteros de forma similar a cómo se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad procesamiento muy grandes. Permite incluir directamente en código escrito en C# fragmentos de código escrito en C, C++ o Java. Está concebido para trabajar en un entorno conectado en red. Cuenta con una amplia biblioteca de clases para comunicarse mediante TCP/IP: HTTP, FTP, etc.

1.14. El análisis

El análisis de requisitos es la primera fase técnica del proceso de ingeniería del software. En este punto se refina la declaración general del ámbito del software en una especificación concreta que se convierte en el fundamento de todas las actividades siguientes de la ingeniería del software. El análisis

debe enfocarse en los dominios de la información, funcional y de comportamiento del problema. Para entender mejor lo que se requiere, se crean modelos, los problemas sufren una partición y se desarrollan representaciones que muestran la esencia de los requisitos y posteriormente los detalles de la implementación. En muchos casos, no es posible especificar completamente un problema en una etapa tan temprana. La creación de prototipos ofrece un enfoque alternativo que produce un modelo ejecutable del software en el que se pueden refinar los requisitos. Se necesitan herramientas especiales para poder realizar adecuadamente la creación de prototipos. Como resultado del análisis, se desarrolla la especificación de requisitos del software. La revisión es esencial para asegurarse que el cliente y el desarrollador tienen el mismo concepto del sistema. Desgraciadamente, incluso con los mejores métodos, el problema sigue cambiando. **(**13**)**

1.14.1 Principios del análisis

En las últimas dos décadas, se han desarrollado un gran número de métodos de modelado. Los investigadores han identificado los problemas del análisis y sus causas y han desarrollado varias notaciones de modelado y sus correspondientes conjuntos de heurísticas para solucionarlos. Cada método de análisis tiene su punto de vista, sin embargo, todos los métodos de análisis se relacionan por un conjunto de principios operativos:

- Debe representarse y entenderse el dominio de información de un problema.
- Deben definirse las funciones que debe realizar el software.
- Debe representarse el comportamiento del software (como consecuencia de acontecimientos externos).
- Deben dividirse los modelos que representan información, función y comportamiento de manera que se descubran los detalles por capas (o jerárquicamente).
- El proceso de análisis debería ir desde la información esencial hasta el detalle de la implementación. **(13)**

¿Por qué es importante?

Para validar los requisitos del software se necesita examinarlos desde diferentes puntos de vista. El análisis representa los requisitos en tres dimensiones, por esa razón, se incrementa la probabilidad de encontrar errores, descubrir inconsistencia y detectar omisiones.

¿Cuáles son los pasos?

Los requisitos de datos, funciones y comportamientos son modelados utilizando diferentes diagramas. El modelado de datos define objetos de datos, atributos y relaciones. El modelado de funciones indica como los datos son transformados dentro del sistema. El modelado del comportamiento representa el

impacto de los sucesos. Se crean unos modelos preliminares que son analizados y refinados para valorar su claridad, completitud y consistencia. Una especificación incorporada en el modelo es creada y luego validada, tanto por el ingeniero del software, como por los clientes/usuarios.

¿Cuál es el producto obtenido?

Las descripciones de los objetos de datos, los diagramas entidad–relación, los diagramas de flujo de datos, los diagramas de transición de estados, las especificaciones del proceso y las especificaciones de control son creadas como resultados de las actividades del análisis. **(13)**

1.15 Gestión de riesgos

1.15.1 Riesgo

Riesgo es el daño potencial que puede surgir por un proceso presente o suceso futuro, (y esto se puede dar en cualquier ámbito laboral) Diariamente en ocasiones se le utiliza como sinónimo de probabilidad, pero en el asesoramiento profesional de riesgo, el riesgo combina la probabilidad de que ocurra un evento negativo con cuanto daño dicho evento causaría. Es decir, en palabras claras, el riesgo es la posibilidad de que un peligro pueda llegar a materializarse. **(13)**

1.15.2 Fases de la Gestión de Riesgos

Entre las fases de la gestión de riesgos se pueden identificar:

- **Fase Identificación**
  - Establecer y mantener alcances y estrategias de gestión de riesgo.
  - Identificar, documentar y clasificar riesgos.
- **Fase de Evaluación**
  - Definir métricas de riesgo.
  - Analizar y priorizar riesgos.
- **Definición del Plan de Acción e Implementación**
  - Desarrollar estrategias de mitigación.
- **Seguimiento de Riesgos**
  - Monitorear y controlar riesgos.

1.16 Ingeniería de requerimientos

En la ingeniería de sistemas y la ingeniería de software la Ingeniería de requerimientos comprende todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un software nuevo o modificado, tomando en cuenta los diversos requerimientos de los

inversores, pueden entrar en conflicto entre ellos. Puede ser conocida también como "Análisis de requerimientos" o “especificación de requerimientos". El propósito de la ingeniería de requerimientos es hacer que los mismos alcancen un estado óptimo antes de seguir adelante con el proyecto. Los buenos requerimientos deben ser medibles, comprobables, sin ambigüedades o contradicciones.

1.16.1 Actividades

Las actividades avaluadas desde un punto de vista conceptual se dividen fundamentalmente en 5 clases.

- **Obtener requerimientos**

A través de entrevistas o comunicación con clientes o usuarios, para saber cuáles son sus deseos.

- **Analizar requerimientos**

Detectar y corregir las falencias comunicativas, transformando los requerimientos obtenidos de entrevistas y otras formas, en condiciones apropiadas para ser tratados por el diseño; los requerimientos deben estar debidamente documentados.

- **Verificar los requerimientos**

Una vez que la especificación de requisitos ha sido desarrollada, los requisitos son verificados. La verificación de requisitos es un proceso para asegurar que la especificación de requisito del producto es una representación exacta de las necesidades del cliente. Este proceso también asegura que los requisitos sean trazados y verificados a través de varias fases del ciclo de vida; particularmente en el diseño, implementación y pruebas. Los requisitos deben ser trazados desde fuentes externas, tales como los clientes, para derivar requisitos del nivel del sistema, para especificar requisitos del producto hardware/software. Además, todos estos requerimientos deben ser trazados al diseño, implementación y pruebas para asegurarse que los requerimientos han sido satisfechos.

- **Validar los requerimientos**

Comprobar que los requerimientos implementados se corresponden con lo que inicialmente se pretendía.

- **Gestión de cambios**

Gestión de cambios es un proceso formal para identificar, evaluar, trazar y reportar cambios propuestos y aprobados a la especificación del producto. Como el proyecto va evolucionando, los requerimientos pueden cambiar o expandirse para ajustar algunas modificaciones en el alcance o diseño del proyecto. Un proceso de gestión de cambios proporciona un rastreo completo y preciso de todos los cambios que son pertinentes al proyecto.

1.16.2 Técnicas principales

La ingeniería de requisitos puede ser un proceso largo y arduo para el que se requiere de habilidades psicológicas. Los nuevos sistemas cambian el entorno y las relaciones entre la gente, así que es importante identificar a todas las personas implicadas, considerar sus necesidades y asegurar que entienden las implicaciones de los nuevos sistemas. Los analistas pueden emplear varias técnicas para obtener los requisitos del cliente, históricamente, esto ha incluido técnicas tales como las entrevistas, o talleres con grupos para crear listas de requisitos, técnicas más modernas incluyen los prototipos, y utilizan casos de uso. Cuando sea necesario, el analista empleará una combinación de estos métodos para establecer los requisitos exactos de las personas implicadas, para producir un sistema que resuelva las necesidades del negocio.

1.16.2.1 Entrevistas

Las entrevistas son un método común, por lo general no se entrevista a toda la gente que se relacionará con el sistema, sino a una selección de personas que represente a todos los sectores críticos de la organización, con el énfasis puesto en los sectores más afectados o que harán un uso más frecuente del nuevo sistema. Los requerimientos que surgen de las entrevistas a menudo se contradicen unos a otros o se formulan desde la ignorancia de los detalles del funcionamiento del sistema, sus potencialidades, interdependencias o limitaciones; por lo que se debe trabajar con los mismos para corregir sus fallas. Las entrevistas pueden ser personales o grupales.

1.16.2.2Talleres

Los requisitos tienen a menudo implicaciones cruzadas desconocidas para las personas implicadas y que a menudo no se descubren en las entrevistas o quedan incompletamente definidas durante la misma. Estas implicaciones cruzadas pueden descubrirse realizando en un ambiente controlado, talleres facilitados por un analista del negocio, en donde las personas implicadas participan en discusiones para descubrir requisitos, analizan sus detalles y las implicaciones cruzadas. A menudo es útil la selección de un secretario dedicado a la documentación de la discusión, liberando al analista del negocio para centrarse en el proceso de la definición de los requisitos y para dirigir la discusión.

1.16.2.3 Forma de contrato

En lugar de una entrevista, se pueden llenar formularios o contratos indicando los requerimientos. En sistemas muy complejos éstos pueden tener centenares de páginas.

1.16.2.4 Objetivos mesurables

Los requerimientos formulados por los usuarios se toman como objetivos generales, a largo plazo, y en cambio se los debe analizar una y otra vez desde el punto de vista del sistema hasta determinar los objetivos críticos del funcionamiento interno que luego darán forma a los comportamientos apreciables por el usuario, luego, se establecen formas de medir el progreso en la construcción, para evaluar en cualquier momento qué tan avanzado se encuentra el proyecto.

1.16.2.5 Prototipos

Un prototipo es una pequeña muestra, de funcionalidad limitada, de cómo sería el producto final una vez terminado. Ayudan a conocer la opinión de los usuarios y rectificar algunos aspectos antes de llegar al producto terminado.

1.16.2.6 Casos de uso

Un caso de uso es una técnica para documentar posibles requerimientos, graficando la relación del sistema con los usuarios u otros sistemas. Dado que el propio sistema aparece como una caja negra, y sólo se representa su interacción con entidades externas, permite omitir dichos aspectos y determinar los que realmente corresponden a las entidades externas. El objetivo de esta práctica es mejorar la comunicación entre los usuarios y los desarrolladores, mediante la prueba temprana de prototipos para minimizar cambios hacia el final del proyecto y reducir los costos finales. Esta técnica se enfrenta a los siguientes peligros potenciales.

- A los directivos, una vez que ven un prototipo, les cuesta comprender que queda mucho trabajo por hacer para completar el diseño final.
- Los diseñadores tienden a reutilizar el código de los prototipos por temor a “perder el tiempo” al recomenzar otra vez.
- Los prototipos ayudan principalmente a las decisiones del diseño y del interfaz de usuario. Sin embargo, no proporcionan explícitamente cuáles son los requisitos.
- Los diseñadores y los usuarios finales pueden centrarse demasiado en diseño del interfaz de usuario y demasiado poco en producir un sistema que sirva el proceso del negocio.

Los prototipos pueden ser, por ejemplo, diagramas, aplicaciones operativas con funcionalidades sintetizadas. Los diagramas en los casos donde se espera que el software final tenga diseño gráfico, se realiza en una variedad de documentos de diseño gráficos y a menudo elimina todo el color del

diseño del software (es decir utilizar una gama de grises). Esto ayuda a prevenir la confusión sobre la apariencia final de la aplicación.

1.16.3 Especificación de requisitos del software

Una especificación de requisitos de software es una descripción completa del comportamiento del sistema a desarrollar, incluye un conjunto de casos de uso que describen todas las interacciones que se prevén que los usuarios tendrán con el software. También contiene requisitos no funcionales. Los requisitos no funcionales son los requisitos que imponen restricciones al diseño o funcionamiento del sistema (tal como requisitos de funcionamiento, estándares de calidad, o requisitos del diseño). Las estrategias recomendadas para la especificación de los requisitos del software están descritas por IEEE 830-1998. Este estándar describe las estructuras posibles, contenido deseable, y calidades de una especificación de requisitos del software.

1.16.4 Identificación de las personas involucradas

Debido a que los cambios que introduce un sistema nuevo tienden a afectar a más de un tipo de usuario, los analistas de requisitos han de tomar en consideración a todos los implicados para que se obtengan y depuren sus requerimientos de la forma más fidedigna posible. Entre las personas implicadas hay que considerar:

- Organizaciones que integran la organización del analista que está diseñando el sistema.
- Organizaciones o sistemas de respaldo.
- Dirección.
- Usuarios.

1.16.5 Problemas

1.16.5.1 Relacionados con las personas involucradas

Pueden existir algunos problemas que impidan una buena determinación de los requisitos, entre las que se destacan:

- Los usuarios no tiene claro lo que desean.
- Los usuarios no se involucran en la elaboración de requisitos escritos.
- Los usuarios insisten en nuevos requisitos después de que el coste y la programación se hayan fijado.
- La comunicación con los usuarios es lenta.

- Los usuarios no participan en revisiones o son incapaces de hacerlo.
- Los usuarios no comprenden los problemas técnicos.
- Los usuarios no entienden el proceso del desarrollo

Todo esto puede provocar cambios de gran envergadura inclusive cuando ya se ha trabajado bastante sobre el software.

1.16.5.2 Relacionados con los analistas

La correcta redacción de las especificaciones de requisitos software es imprescindible para el correcto desarrollo del proyecto. Por ello, en su redacción hay que evitar:

- Uso de terminología ambigua en la redacción de los documentos de requisitos.
- Escritura poco legible, voz pasiva, abuso de negaciones.
- Uso de verbos en condicional, expresiones subjetivas.
- Ausencia de términos y verbos del dominio de la aplicación.

1.16.5.3 Relacionados con los desarrolladores

Los problemas posibles causados por los desarrolladores durante análisis de requisitos son:

- El personal técnico y los usuarios finales pueden tener diversos vocabularios y pueden llegar a creer incorrectamente que están de acuerdo, no dándose cuenta del desacuerdo hasta que se provee el producto final.
- Los desarrolladores pueden intentar encajar el sistema en un modelo existente, en vez de desarrollar un sistema adaptado a las necesidades del cliente.
- El análisis de requisitos se puede realizar a menudo por los ingenieros o programadores, en vez de personal con el dominio de las habilidades de relación con la gente y el conocimiento para entender las necesidades de un cliente correctamente.

1.17 Diseño del software

1.17.1 Principios del diseño

El diseño de software es una representación significativa de ingeniería de algo que se va a construir, se puede hacer el seguimiento basándose en los requisitos del cliente, y al mismo tiempo la calidad se puede evaluar y cotejar con el conjunto de criterios predefinidos para obtener un diseño adecuado En el contexto de la ingeniería del software, el diseño se centra en cuatro áreas importantes, datos,

arquitectura, interfaces y componentes. En estas cuatro áreas se aplican los principios que se abordan a continuación.

- En el proceso de diseño no deberá utilizarse “orejeras”. Un buen diseñador deberá tener en cuenta enfoques alternativos, juzgando todos los que se basan en los requisitos del problema, los recursos disponibles para realizar el trabajo y los conceptos de diseño.
- El diseño deberá poderse rastrear hasta el modelo de análisis, dado que un solo elemento del modelo de diseño suele hacer un seguimiento de los múltiples requisitos, es necesario tener un medio para rastrear la forma en que los requisitos son apoyados por el modelo de diseño.
- El diseño no deberá inventar nada que ya esté inventado, los sistemas se construyen utilizando un conjunto de patrones de diseño, muchos de los cuales probablemente ya se han encontrado antes. Estos patrones deberán elegirse siempre como una alternativa para reinventar, cuando hay poco tiempo y los recursos son limitados, el tiempo de diseño se deberá invertir en la representación verdadera de ideas nuevas y en la integración de esos patrones que ya existen.
- El diseño deberá minimizar la distancia intelectual entre el software y el problema como si de la misma vida real se tratara, es decir, la estructura del diseño del software (siempre que sea posible) imita la estructura del dominio del problema.
- El diseño deberá presentar uniformidad e integración, un diseño es uniforme si parece que fue una persona la que lo desarrolló por completo. Las reglas de estilo y de formato deberán definirse para un equipo de diseño antes de comenzar el trabajo sobre el diseño. Un diseño se integra si se tiene cuidado a la hora de definir interfaces entre los componentes del diseño.
- El diseño deberá estructurarse para admitir cambios, los conceptos de diseño estudiados hacen posible un diseño que logra este principio.
- El diseño no es escribir código y escribir código no es diseñar, incluso cuando se crean diseños procedimentales para componentes de programas, el nivel de abstracción del modelo de diseño es mayor que el código fuente. Las únicas decisiones de diseño realizadas a nivel de codificación se enfrentan con pequeños datos de implementación que posibilitan codificar el diseño procedimental.
- El diseño deberá evaluarse en función de la calidad mientras se va creando, no después de terminarlo. Para ayudar al diseñador en la evaluación de la calidad se dispone de conceptos de diseño y de medidas de diseño.

- El diseño deberá revisarse para minimizar los errores conceptuales, a veces existe la tendencia de centrarse en minucias cuando se revisa el diseño, olvidándose del bosque por culpa de los árboles. Un equipo de diseñadores deberá asegurarse de haber afrontado los elementos conceptuales principales antes de preocuparse por la sintaxis del modelo del diseño. **(**13**)**

¿Quién lo hace?

El ingeniero del software es quién diseña los sistemas basados en computadoras, pero los conocimientos que se requieren en cada nivel de diseño funcionan de diferentes maneras. En el nivel de datos y de arquitectura, el diseño se centra en los patrones de la misma manera a como se aplican en la aplicación que se va a construir, en el nivel de la interfaz, es la ergonómica humana la que dicta nuestro enfoque de diseño, y en el nivel de componentes, un "enfoque de programación" conduce a diseños de datos y procedimentales eficaces.

¿Por qué es importante?

Si se construye una casa ¿se hace sin un plano?, el riego es muy grande, se cometerían errores. El software de computadora es considerablemente más complejo que una casa, de aquí que se necesita un plano "el diseño"

¿Cuáles son los pasos?

El diseño comienza con el modelo de los requisitos, se trabaja por transformar este modelo y obtener cuatro niveles de detalles de diseño: la estructura de datos, la arquitectura del sistema, la representación de la interfaz y los detalles a nivel de componentes. Durante cada una de las actividades del diseño, se aplican los conceptos y principios básicos que llevan a obtener una alta calidad.

¿Cuál es el producto obtenido?

Por último se produce una especificación del diseño, la especificación se compone de los modelo del diseño que describen los datos, arquitectura, interfaces y componentes, cada una de estas partes es la que forma el producto obtenido del proceso de diseño.

¿Cómo puedo estar seguro de que lo he hecho correctamente?

En cada etapa de revisión los productos del diseño del software se verifican en cuanto a claridad, corrección, finalización y consistencia, y se comparan con los requisitos. **(13)**

1.18 Patrones

¿Que son los patrones?

Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos.

Son soluciones basadas en la experiencia y que se ha demostrado que funcionan. **(2)**

1.18.1 Categorías de patrones

Según la escala o nivel de abstracción:

Patrones de arquitectura: Aquéllos que expresan un esquema organizativo estructural fundamental para sistemas de software.

Patrones de diseño: Aquéllos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que se pueden construir sistemas software.

Patrones de Idiomas: Patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto, además, también es importante reseñar el concepto de Anti patrón de Diseño, que con forma semejante a la de un patrón, intenta prevenir contra errores comunes de diseño en el software. La idea de los anti patrones es dar a conocer los problemas que acarrean ciertos diseños muy frecuentes, para intentar evitar que diferentes sistemas acaben una y otra vez en el mismo callejón sin salida por haber cometido los mismos errores.

1.18.2 Patrones de Diseño

Un patrón de diseño es una abstracción de una solución en un nivel alto. Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo; desde el análisis hasta el diseño y desde la arquitectura hasta la implementación. **(8)**

Un patrón de diseño es una solución a un problema de diseño, para que una solución sea considerada un patrón debe poseer ciertas características una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores, otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. **(9)**

1.18.2.1 Patrones GoF

Fue por los años 1994, que apareció el libro "Design Patterns: Elements of Reusable Object Oriented Sofware" escrito por los ahora famosos **Gang of Four** (GoF, que en español es la pandilla de los cuatro) formada por **Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides**. El grupo de GoF clasificaron los patrones en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento. **(8)**

Patrones creacionales: Engloban aquellos patrones de software que se centran en la forma de crear las clases y sus instancias, así como el uso que recibirán una vez creadas. **(7)**

- **Abstract Factory**

Proporciona una interfaz para crear familias de objetos o que dependen entre sí, sin especificar sus clases concretas.

- **Builder**

Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.

- **Factory Method**

Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.

- **Prototype**

Especifica los tipos de objetos a crear por medio de una instancia prototípica, y crear nuevos objetos copiando este prototipo.

- **Singleton**

Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella.

Patrones estructurales: Engloban aquellos patrones de software que se centran en la composición y estructura de las clases y en la herencia entre ellas. **(7)**

- **Adapter**

Convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permiten que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.

- **Bridge**

Desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.

- **Composite**

Combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos.

- **Decorator**

Añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.

- **Facade**

Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema se más fácil de usar.

- **Flyweight**

Usa el compartimiento para permitir un gran número de objetos de grano fino de forma eficiente.

- **Proxy**

Proporciona un sustituto o representante de otro objeto para controlar el acceso a éste.

Patrones de comportamiento: Engloban aquellos patrones de software que se centran en la comunicación entre las distintas clases y objetos. **(**7**)**

- **Chain of Responsibility**

Evita acoplar el emisor de una petición a su receptor, al dar a más de un objeto la posibilidad de responder a la petición. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada por algún objeto.

- **Command**

Encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer la operaciones.

- **Interpreter**

Dado un lenguaje, define una representación de su gramática junto con un intérprete que usa dicha representación para interpretar las sentencias del lenguaje.

- **Iterator**

Proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna.

- **Mediator**

Define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.

- **Memento**

Representa y externaliza el estado interno de un objeto sin violar la encapsulación, de forma que éste puede volver a dicho estado más tarde.

- **Observer**

Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.

- **State**

Permite que un objeto modifique su comportamiento cada vez que cambia su estado interno. Parecerá que cambia la clase del objeto.

- **Strategy**

Define una familia de algoritmos, encapsula uno de ellos y los hace intercambiables, permite que un algoritmo varíe independientemente de los clientes que lo usan.

- **Template Method**

Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura.

- **Visitor**

Representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera.

1.18.3 Patrones de Arquitectura

Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. **(10)**

Los patrones de arquitectura expresan el esquema fundamental de organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos; especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos. **(10)**

1.18.3.1 Estilos Arquitectónicos

El tópico más urgente y exitoso en arquitectura de software en los últimos cuatro o cinco años es, sin duda, el de los patrones, tanto en lo que concierne a los patrones de diseño como a los de arquitectura. Inmediatamente después, en una relación a veces de complementariedad, otras de oposición, se encuentra la sistematización de los llamados estilos arquitectónicos. Cada vez que alguien celebra la mayoría de edad de la arquitectura de software, y aunque señale otros logros como las técnicas de refinamiento, esos dos temas se destacan más que cualesquiera otros. Sin embargo, sólo en contadas ocasiones la literatura técnica existente se ocupa de analizar el vínculo entre estilos y

patrones, se los yuxtapone cada vez que se enumeran las ideas y herramientas disponibles, se señala con frecuencia su aire de familia, pero no se articula formal y sistemáticamente su relación.

Habrá que admitir que ambos asuntos preocupan y tienen como destinatarios a distintas clases de profesionales, o diferentes stakeholders, como ahora se recomienda llamar, quienes trabajan con estilos favorecen un tratamiento estructural que concierne más bien a la teoría, la investigación académica y la arquitectura en el nivel de abstracción más elevado, mientras que quienes se ocupan de patrones se centran en cuestiones que están más cerca del diseño, la práctica, la implementación, el proceso, el refinamiento, el código.

Los patrones coronan una práctica de diseño que se origina antes que la arquitectura de software se distinguiera como discurso en perpetuo estado de formación y proclamara su independencia de la ingeniería en general y el modelado en particular. Los estilos, en cambio, expresan la arquitectura en el sentido más formal y teórico, constituyendo un tópico esencial de lo que Goguen ha llamado el campo "seco" de la disciplina.

Un estilo describe entonces una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas, esas piezas se encuentran repetidamente en la práctica, trasuntando la existencia de decisiones estructurales coherentes. Una vez que se han identificado los estilos, es lógico y natural pensar en re-utilizarlos en situaciones semejantes que se presenten en el futuro.

1.19 Métricas de Software

El concepto de métrica es el término que describe muchos y muy variados casos de medición, siendo una métrica una medida estadística que se aplica a todos los aspectos de calidad de software, los cuáles deben ser medidos desde diferentes puntos de vista como el análisis, construcción, funcionalidad, documentación, métodos, proceso, usuario, entre otros.

1.19.1 Características de las métricas de software

Se han propuesto cientos de métricas para el software, pero no todas proporcionan un soporte práctico para el desarrollador de software. Algunas demandan mediciones que son demasiado complejas, otras son tan esotéricas que pocos profesionales tienen la esperanza de entenderlas y otras violan las nociones básicas intuitivas de lo que realmente es el software de alta calidad.

Algunos autores definen un conjunto de atributos que deberían acompañar a las métricas efectivas de software, entre las que están:

- **Simples y fáciles de usar**

Debería ser relativamente fácil aprender a obtener una métrica y su cálculo, no deberían demandar un esfuerzo o cantidad de tiempo inusuales.

- **Empírica e intuitivamente persuasivas**

La métrica debería satisfacer las nociones intuitivas del ingeniero sobre el atributo del producto en cuestión (por ejemplo, una métrica que mide la cohesión de un módulo debería aumentar su valor a medida que crece el nivel de cohesión).

- **Consistentes y objetivas**

La métrica debería siempre producir resultados sin ambigüedad. Un tercer equipo debería ser capaz de obtener el mismo valor de métrica usando la misma información del software.

- **Independientes del lenguaje de programación**

Las métricas deberían basarse en el modelo de análisis, modelo de diseño o en la propia estructura del programa. No deberían depender de los caprichos de la sintaxis o semánticas del lenguaje de programación.

- **Un eficaz mecanismo para la realimentación de calidad**

La métrica debería proporcionar, al desarrollador de software, información que le lleve a un producto final de mayor calidad.

1.19.2 Métricas del análisis.

El trabajo técnico en la ingeniería del software empieza con la creación del modelo de análisis. En esta fase se obtienen los requisitos y se establece el fundamento para el diseño. Por tanto, son deseables las métricas técnicas que proporcionan una visión interna a la calidad del modelo de análisis.

- **La métrica de punto de función (PF)**

1.19.3 Métricas del Diseño

También existe algunas métricas que miden la calidad del diseño, en este epígrafe se hace alusión a algunas de las métricas para el diseño que se han definido a nivel mundial. Muchos expertos han asegurado que la opción de tener un diseño sin medición no es aceptable sin embargo en la actualidad se sigue debatiendo sobre la eficacia de estas métricas para el diseño de software. **[1]**

- **Métricas del diseño arquitectónico**

Especialistas en el tema han definido tres medidas de complejidad del diseño del software

- **Complejidad estructural** $S(i)$ se obtiene aplicando la ecuación $S(i) = f_{out}^{2}(i)$
- **Complejidad de datos** $D(i)$ se obtiene mediante la ecuación $D(i) = V(i) / \lfloor f_{out}(i) + 1 \rfloor$
- **Complejidad del sistema** $C(i)$ se obtiene mediante la ecuación $C(i) = S(i) + D(i)$

1.20 Conclusiones

En este capítulo se describió la problemática existente y se definió el objeto de estudio, se hizo un análisis de cuestiones teóricas importantes que ayudaron a la comprensión del problema. Se expusieron las principales funcionalidades de diferentes generadores de reportes, de metodologías de desarrollo de software y de herramientas para desarrollo. Después de llevar a cabo un estudio exhaustivo se llegaron a las siguientes conclusiones:

Se utilizará el generador de reportes Active Reports para elaborar la herramienta, se usará el lenguaje de modelado UML y la herramienta CASE Enterprise Architect 7.0 para crear los diagramas del diseño del sistema, la plataforma .NET con C# como lenguaje de programación y .NET Framework como soporte de trabajo. La metodología utilizada será RUP. Se aplicarán métricas para el análisis y el diseño que ayudará a evaluar la calidad del sistema.

Capítulo 2: Descripción del Sistema

2.1 Introducción

En este capítulo se describen las principales características del sistema, se muestra el modelo de dominio y los conceptos que este abarca, se describen los aspectos novedosos del sistema y los beneficios que el mismo traería para la ONE.

2.2 Objetivos del Módulo Generador de Reportes

El Objetivo principal de este Módulo del SIGE facilitar al máximo todo el proceso de elaboración, administración y acceso a los reportes generados por las diferentes dependencias de la ONE.

2.2.1 Objetivos Estratégicos del Módulo Generador de Reportes

Debe permitir gestionar las distintas actividades para reportar la información en la ONE (incluyendo los Centros Informantes que dispongan de la capacidad tecnológica) mediante una interfaz gráfica sencilla y amigable. Además de proporcionar una conectividad en toda la estructura de la ONE de manera que permita un intercambio rápido y seguro de la información, eliminando las deficiencias que presenta el sistema actual.

2.3 Flujo actual de los procesos involucrados en el campo de acción

En el campo de acción el proceso que se involucra es gestionar reportes.

2.3.1 Gestionar Reportes

Este proceso se divide en 5 subprocesos fundamentales, crear reportes, editar reportes, eliminar reportes, consultar reportes y listar reportes.

Crear reportes: Para realizar este subproceso el administrador de reportes se encarga de llevar el orden de las acciones solo tienen que introducir los datos que van a contener los reportes.

Editar reportes: En este subproceso primero se busca el reporte deseado y luego se modifican los datos del mismo.

Eliminar reportes: Se busca el reporte que se quiere eliminar y se procede a su eliminación.

Consultar reportes: Este subproceso se realiza buscando el reporte que se quiere consultar y realizar la revisión del mismo.

Listar reportes: Para realizar este subproceso primeramente se buscan los reportes que se quieren listar y luego se procede a visualizar el listado con los mismos.

2.4 Objeto de Automatización

Se desea automatizar todos los procesos que le permitan a la Oficina Nacional de Estadísticas realizar una generación de reportes de forma dinámica. El principal proceso a automatizar es gestionar reportes.

2.5 Modelo de Dominio

El Modelo de Dominio (Modelo Conceptual) es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema. Representa conceptos del mundo real, no de los componentes de software. **(13)**

El modelo de dominio se representa en UML con un Diagrama de Clases en los que se muestra:

- Conceptos u objetos del dominio del problema: clases conceptuales.
- Asociaciones entre las clases conceptuales.
- Atributos de la clase conceptuales.

En el Modelo de Dominio no se muestra comportamiento. Las clases conceptuales pueden tener atributos pero no métodos.

Se decidió realizar el modelo de dominio debido a que el módulo no automatiza ningún proceso de petición de información u otra solicitud, no se encuentran claramente definido procesos de este tipo ni actores que inicialicen procesos de negocio, solo se encarga de proponer un software capaz de realizar las actividad que realiza MicroSet y que incluya nuevas y ventajosas funcionalidades que estandaricen el trabajo en la ONE.

2.5.1 Conceptos identificados

Usuario: Es el actor que se encarga de interactuar con el sistema, en dependencia del rol que tenga asignado podrá realizar determinadas acciones.

Administrador de Reporte: Persona que se encarga de administrar los reportes y los usuarios.

Diseñador de Reporte: Persona encargada de hacer el diseño de la consulta y del modelo de reporte.

Reporte: Archivo que contiene información.

SIGE-MGR: Software mediante el cual se realizará las gestiones con los reportes

Información: Datos que llenarán el cuerpo del reporte y son extraído mediante una consulta hecha a la base de datos.

Visor: Ventana que facilita al usuario la pre visualización del reporte seleccionado, permitiéndole revisarlo antes de guardarlo o exportarlo a otros formatos.

Modelo de Reportes: Modelo que contiene toda la información.

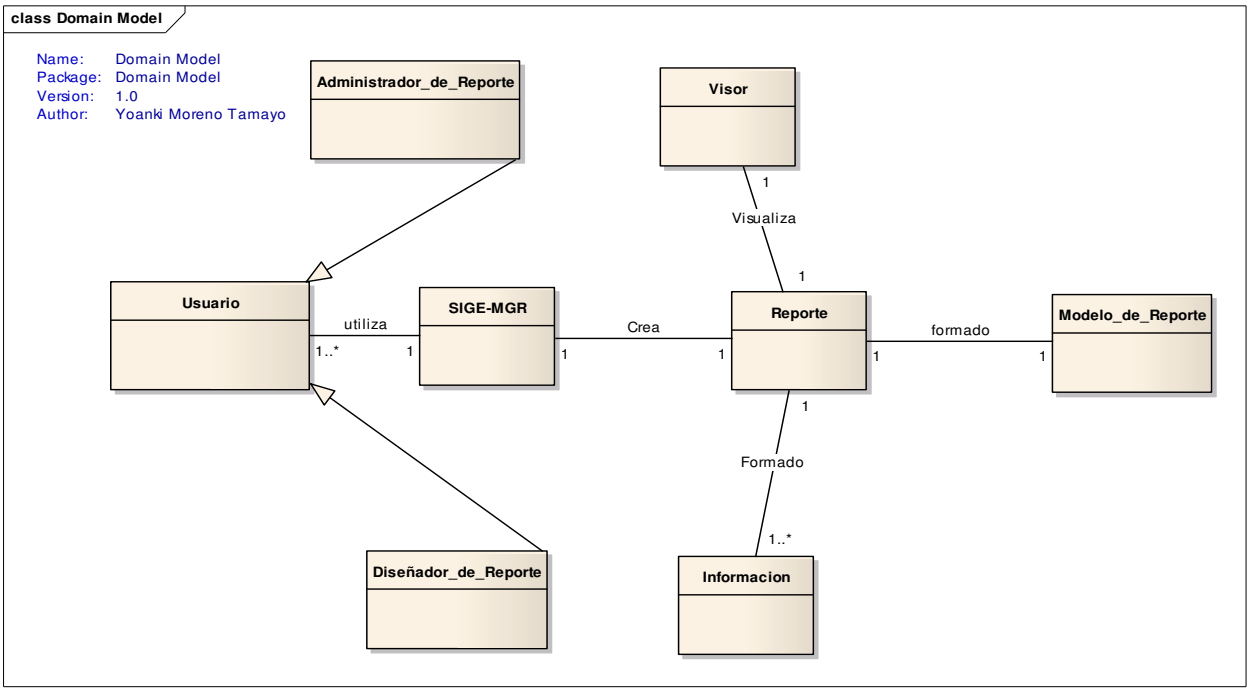


Figura2.1. Modelo de Dominio

2.6 Propuesta del sistema

El sistema que propone el módulo generador de reportes dentro del Sistema Integrado de Gestión Estadísticas (SIGE), debe poseer gran dinamismo, el mismo deberá posibilitar realizar una serie de actividades con el reporte que estandarizarán el trabajo en la Oficina Nacional de Estadísticas y le proporcionará a la misma una fiabilidad en relación a la información que debe suministrar.

2.6.1 Cómo debe trabajar el sistema

El sistema que propone el MGR debe permitir acceder a la información almacenada sobre un gestor de base de datos SQL server 2000 o sobre un Datawarehouse, debe ser capaz de brindar funcionalidades básicas como realizar consultas parametrizadas, es decir lograr establecer una conexión con la base de datos teniendo en cuenta los indicadores que se enmarcan. Permitir el diseño de un reporte asociado a la consulta anterior, con facilidades visuales para el usuario o sea diseñar en un modelo de reporte con todos los parámetros seleccionados previamente en el cuerpo del reporte. Como etapa

final la creación de un reporte configurable, un reporte vinculado a las dos acciones anteriormente mencionadas. Además de esto, el sistema debe permitir realizar otras actividades de importancia para la comodidad del usuario como migrar a otro formato el reporte, listar y guardar los reportes.

2.6.2 Análisis comparativo entre la solución existente y la propuesta.

La solución que existe hasta el momento es el programa MicroSet, el mismo, creado en la década del 70 ha visto nacer nuevas versiones de si mismo con algunos avances pero en la actualidad su funcionamiento no satisface plenamente las necesidades con que cuenta la ONE.

Esta comparación viene a mostrar un estado deseado de la herramienta del MGR, debe aventajar al MicroSet en varios aspectos de importancia en la generación de reportes.

Parámetros	MicroSet	Propuesta (SIGE-MGR)
Interfaz gráfica	Poco amigable	Interfaz amigable para el usuario, colores frescos y agradables.
Portabilidad	No es portable, no funciona correctamente sobre los sistemas operativos de la familia NT. Solo en MSDOS	Hasta el momento esta diseñado para que trabaje bajo Windows
Conexiones	No permite establecer conexiones con Bases de Datos, solo a sus bancos de carpetas.	Debe permitir conexiones con Bases de Datos con SQL server 2000 y con Datawarehouse
Forma de trabajo	Llena de forma manual mediante comandos de textos el cuerpo del reporte, lo que puede provocar demora.	La realización de los reportes será de forma dinámica utilizando mouse y teclado al unísono.
Migración de reportes	No permite la migración de los reportes a otros formatos	Con las facilidades que brinda Active Reports debe permitir migrar los reportes a formatos como: pdf, HTML, txt, Doc. etc.
Ampliación de espectro de trabajo	No permite ampliar los campos de clasificación para generar reportes de entidades actuales.	Esta diseñado para que posibilite adicionar nuevas entidades y campos que surgen en la actualidad.
Salida de información	No permite la salidas de tablas con información de varios modelos	Debe permitir la salida de información en cualquier tipo de tablas y en forma de gráfico.

Tabla2.1. Comparación entre MicroSet y propuesta SIGE-MGR

Después de analizar la tabla2.1 representada en la parte superior y mirando con sentido crítico ambas propuestas se arriba a la conclusión de que la propuesta realizada por el Módulo Generador de Reportes (MGR) debe ser superior al MicroSet, pues en todos los aspectos de la comparación esta propuesta propone mejores funcionalidades.

2.6.3 Aspectos novedosos que propone la herramienta del (SIGE-MGR)

El (SIGE) Sistema Integrado de Gestión Estadística por si solo es un aspecto novedoso, lo que hace que el trabajo realizado por el MGR también lo sea, esta herramienta le proporcionará a la ONE una serie de beneficios que darán al traste con la calidad de su trabajo. Dentro de los aspectos novedosos más significativos que tendrá la nueva herramienta se pueden mencionar.

- **Creación de reportes digital con todo tipo de información**

En la ONE algunos reportes se entregan en formato duro, ya sea por el tipo de información que contenía o por la complejidad del modelo del reporte, con esta herramienta todos los reportes que se entregan en formato duro ahora se entregarán de forma digital, podrán ser enviados por correos o en medios de almacenamiento digital, lo que eliminará la demora en la realización y entrega de estos reportes estadísticos.

- **Seguridad de la información**

El sistema de seguridad que se propone para la aplicación es muy confiable, pues otorga privilegios para realizar las tareas con los reportes según el usuario que acceda al sistema, de esta forma solo podrá interactuar con la información el personal autorizado y se minimiza el riego de que la información pueda ser contactada por personal ajeno.

- **El diseño de consultas**

El sistema cuenta con un asistente de consulta que posibilita diseñar una consulta con los parámetros deseados, esto es importante pues si algún cliente quiere una información específica, mediante ese asistente se podrá realizar una consulta a esa información solamente y de esta forma es más efectivo el trabajo, igualmente incluye un cuadro de consultas, en el cuál se puede declarar la consulta de forma manual por si no quiere utilizar el asistente. Las consultas quedarán almacenadas en la base de datos y podrán ser asignadas posteriormente a otras peticiones.

- **Migración de Reportes a otros formatos**

Esta opción facilitará el manejo del reporte, el sistema haciendo uso de las características del Active Reports permitirá migrar los reportes a diferentes archivos como PDF, Txt, HTML, Excel entre otros.

- **Creación de Reportes en forma de gráfico**

La posibilidad de realizar reportes en forma tabular y de forma gráfica es sin duda unos de los aspectos más novedosos que incluye esta herramienta pues los formatos gráficos pueden mostrar en ocasiones perspectivas que en una tabla no son tan fáciles de observar, este sistema brindará opciones para diseñar reportes en graficas como barras, pastel y otros que harán más dinámico el trabajo con los reportes, brindará la opción de escoger el gráfico deseado y de realizar la consulta pertinente para mostrar la información.

- **Uso del Datawarehouse**

Para realizar las peticiones a la base de datos, las consultas no solamente se pueden hacer usando Sql 2000 sino que el módulo tiene una segunda parte en la cual incluye la utilización del Online Analitical Processing (OLAP) para hacer peticiones mediante un sistema Datawerhouse implementado para el SIGE. El uso de este Datawarehouse propiciará que el tiempo de respuesta del sistema sea agilizado de forma tal que se pueda reportar con una velocidad 3 veces mayor que con Sql 2000.

2.6.4 Beneficios para la ONE el uso del (SIGE-MGR)

El Sistema Integrado de Gestión Estadísticas (SIGE) es un paquete integral que posee una serie de características que lo hacen único en el ámbito de la gestión estadística, el Módulo Generador de Reporte como parte de este paquete elabora una herramienta generadora de reportes con características muy particulares que proporcionará un gran beneficio para el trabajo de la ONE.

Primeramente el SIGE tiene características especiales, pues debido a la forma de controlar las estadísticas en Cuba es muy difícil adaptar otro sistema integral gestor de estadística a su entorno, la Oficina Nacional de Estadísticas almacena y gestiona información de todas las entidades del país lo que quiere decir que la información económica del país esta totalmente centralizada, en otros países esto no funciona así pues cada empresa posee una política de trabajo basada en sus estrategia que solo ellas conocen.

El sistema generador de reportes del SIGE le permitirá a la ONE la gestión de datos en cada núcleo de su estructura, lo cuál supondrá un acceso rápido y sencillo a los datos, gracias a interfaces gráficas sencillas y amigables y a las facilidades brindadas por la herramienta Active Reports para la elaboración de reportes. Además, los datos accedidos estarán siempre actualizados, lo que constituye un factor muy importante para poder mantener al día con la información periódica a todos los órganos de dirección a los diferentes niveles.

2.7 Especificación de requisitos del Sistema

2.7.1 Requisitos Funcionales

Un requerimiento funcional son capacidades o condiciones que el sistema debe cumplir definiendo el comportamiento interno del software así como los comportamientos del sistema.

Autenticar

El sistema permite saber quién accede en todo momento a él.

Crear Reportes

El sistema dará la posibilidad de crear reportes.

Editar Reportes

El sistema debe ser capaz de localizar un reporte deseado y si este es modificado actualizarlo.

Eliminar Reportes

El sistema debe permitir eliminar un reporte.

Graficar Reportes

El sistema dará la posibilidad de mostrar la información de forma tabular o en gráficas.

Guardar Reportes

El sistema permitirá guardar reportes en la base de datos a consideración del usuario.

Listar Reportes

El sistema permitirá mostrar los reportes almacenados en la base de datos, podrá hacerlo por fecha, por indicadores, etc.

Terminar Reportes

El proceso de terminar un reporte estará dado por el hecho de que cuando se quiera terminar un reporte se pueda imprimirlo o guardar y de esta forma se termina el reporte. El sistema permitirá realizar cada una de estas acciones.

Gestionar Información

El proceso de gestionar información permite tener un acceso a la información como es el caso de trabajar la información con fórmulas matemáticas, tales como promedio, por ciento y conteo, además se puede hacer consultas de donde extraer la información.

2.7.2 Requisitos No Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable,

rápido o confiable. En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Existen múltiples categorías para clasificar a los requerimientos no funcionales, siendo las siguientes, representativas de un conjunto de aspectos que se deben tener en cuenta, aunque no limitan a la definición de otros.

Usabilidad

El sistema tiene que ser fácil de manejar, algo interactivo y de fácil comprensión por las personas que lo vayan a manipular debido a la poca experiencia de estos con la informática.

Confiabilidad

El sistema mantendrá la información que maneja protegida de accesos no autorizados. Para esto cuenta con un grupo de usuarios a los cuales se les asignarán privilegios de acuerdo con el rol que desempeñan mediante un sistema de autentificación con usuario y contraseña. Cuenta además con un solo administrador de Base de Datos y un Administrador del Sistema. Todo tipo de contraseña en el sistema será debidamente encriptado.

Soporte

El equipo de desarrollo se mantendrá en contacto con el cliente hasta tanto no se logre el entendimiento total del producto por un período de un año. Esto puede ser por vía e-mail o personalmente.

Restricciones de diseño

El sistema funcionará sobre aplicaciones desarrolladas mediante la programación orientada a objetos (POO) y filosofía cliente servidor, usando como gestor de Base de Datos SQL Server. Se debe utilizar Visual Studio .NET, Enterprise Architect, Ado.Net, SQL Server 2000 además el color predominante en el sistema será el azul.

Software

En la computadora que haga función de servidor, independientemente del sistema operativo, se necesita el SGBD SQL Server 2000. En las computadoras de los usuarios y del grupo de soporte sólo se requiere de navegador para Internet o Intranet y la instalación del Framework .NET 2.0.

Hardware

Se requiere de un servidor de 256 MB de RAM como mínimo y 10 MB de espacio libre en disco duro, todas las computadoras implicadas, tanto para la administración como las de los usuarios, deben estar conectadas a una red y tener al menos 128 MB de RAM.

Requerimiento de ayuda y documentación

El sistema contará con un mapa de navegación donde orientará al usuario de la interacción entre las diferentes interfaces visuales.

Interfaz

El protocolo de comunicación que usará el sistema para comunicarse con otros sistemas que no se encuentren en la misma estación de trabajo será el TCP/ IP.

Además cuenta con las siguientes interfaces:

Interfaces de Usuarios

El sistema contará con una interfaz de usuario aunque con varios ambientes de trabajo en dependencia de la funcionalidad que se brinda.

Interfaces con otros Hardware

El sistema podrá mantener comunicación con otros hardwares con el fin de dar salida a los reportes realizados, para ello cuenta con esta interfaz.

Interfaces con otros Software

Con el fin de hacer más fácil y dinámico la realización de los reportes, estas interfaces permitirán la comunicación con software que lo posibilite.

2.8 Modelo del Sistema

En el epígrafe que se muestra a continuación se exponen los resultados obtenidos después de realizar la captura de requisitos para el MGR, además este epígrafe contiene las especificaciones de los Casos de Usos identificados para el sistema (CUS), los cuales se determinaron teniendo en cuenta las necesidades de los usuarios y las demandas del cliente. El modelo de sistema permite tener un entendimiento más detallado de cómo va a estar estructurado el sistema a partir de los CU identificados. Para ello se identifican los actores y se especifican los casos CUS.

Los **actores** de un sistema son agentes externos, es decir, aquellas personas o sistemas que interactúan con él.

Los **casos de usos** son un conjunto de secuencia de acciones que un sistema ejecuta y produce un resultado observable para un actor.

2.8.1 Actores del Sistema

Actor	Descripción
Usuario del sistema	Es la persona que accede al sistema, el que se autentifica y se le da por tanto un determinado nivel de acceso según su rol. Puede ser Administrador, Diseñador o un usuario con permisos suficientes para revisar los reportes.
Administrador	Es el usuario con mayor nivel de acceso, el encargado de administrar o controlar los usuarios que accedan al sistema y los roles que desempeñan. Se encarga además de controlar lo referente a los reportes, la inserción, eliminación y actualización de los mismos.
Diseñador.	Este usuario tiene la tarea de diseñar el reporte, es decir, organizar la forma en que la información de la base de datos será mostrada en este, además de salvarlo en la base de datos.

Tabla2.2.Actores del Sistema

2.8.2 Diagrama de Paquetes

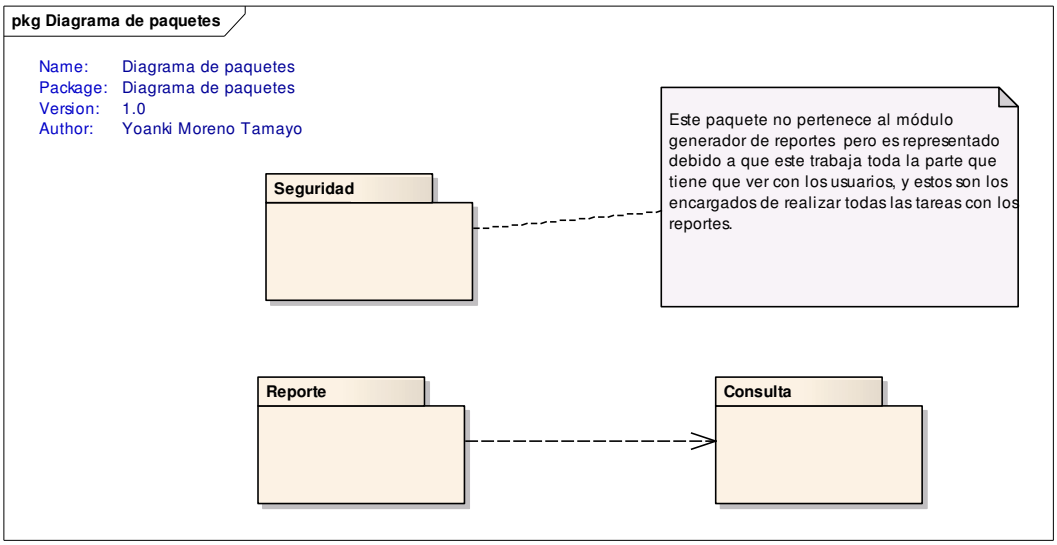


Figura2.2. Diagrama de Paquetes

El diagrama de paquetes presentado muestra la relación entre los paquetes existentes, el paquete de Reportes que agrupa los Casos de Usos que trabajan con los reportes plenamente estos los CU Administrar Reportes y Consultar Reportes, el otro paquete que se llama Consulta agrupa los Casos de Usos que se encargan del diseño que se realiza previo a la creación del reportes que incluye los Casos de Uso Diseñar Consultas y Diseñar Modelo de Reportes. El criterio usado para agrupar los paquetes fue la funcionalidad de los Casos de Uso.

2.8.2.1 Descripción de Casos de Uso del Sistema

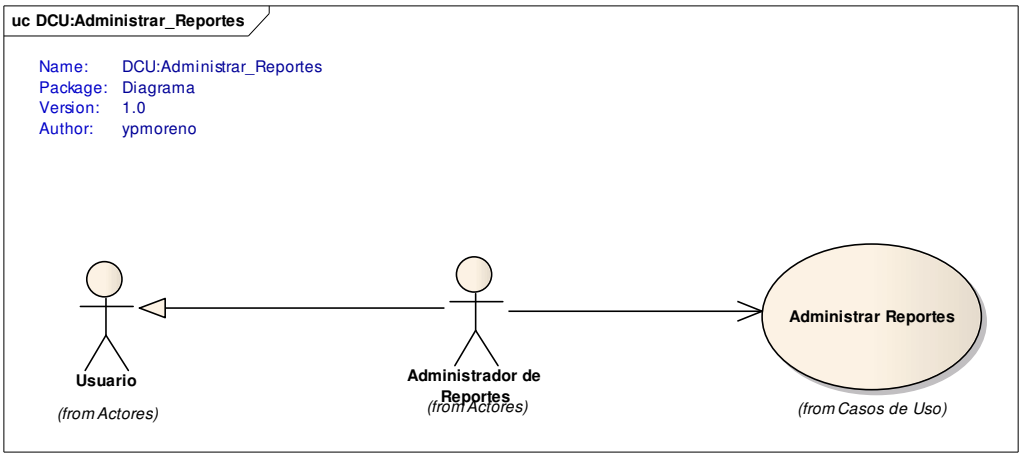


Figura2.3.DCU: Administrar Reportes

Caso de Uso:	Administrar Reportes
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el administrador accede al sistema, y lleva a cabo acciones referentes al control de los reportes. Insertar, eliminar y actualizar reportes.
Precondiciones:	Que el Administrador se haya autenticado
Prioridad	Crítica
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El administrador selecciona la opción que desea ejecutar.	1.1 a) Si desea modificar el reporte ir a Sección “Modificar Reporte” b) Si desea eliminar un determinado

	reporte ir a Sección “Eliminar Reporte” c)Si desea buscar un determinado reporte, ir a Sección “Buscar Reporte”
Sección “Actualizar Reporte”	
Acción del Actor	Respuesta del Sistema
2. El administrador indica el reporte a actualizar.	2.1 El sistema busca el reporte y lo muestra al usuario.
3. Actualizar la información.	3.1 Si lo encuentra el sistema actualiza el reporte en la Base de Datos.
	3.2 Muestra el reporte actualizado.
Sección “Eliminar Reporte”	
Acción del Actor	Respuesta del Sistema
4. El administrador indica el reporte que desea eliminar.	4.1 El sistema busca el reporte en la Base de Datos.
	4.2 Si lo encuentra lo elimina de la Base de Datos.
	4.3 Muestra el listado actualizado de los reportes.
Sección “Buscar Reporte”	
Acción del Actor	Respuesta del Sistema
5. El administrador indica los parámetros por los que desea buscar el reporte.	5.1 El sistema busca los reportes en la Base de Datos que coincidan con esos parámetros.
	5.2 Muestra una lista de esos reportes.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
	[4.1, 5.2]Si no lo encontró muestra un mensaje que indica que el reporte no se encuentra en la Base de Datos.
Poscondiciones	

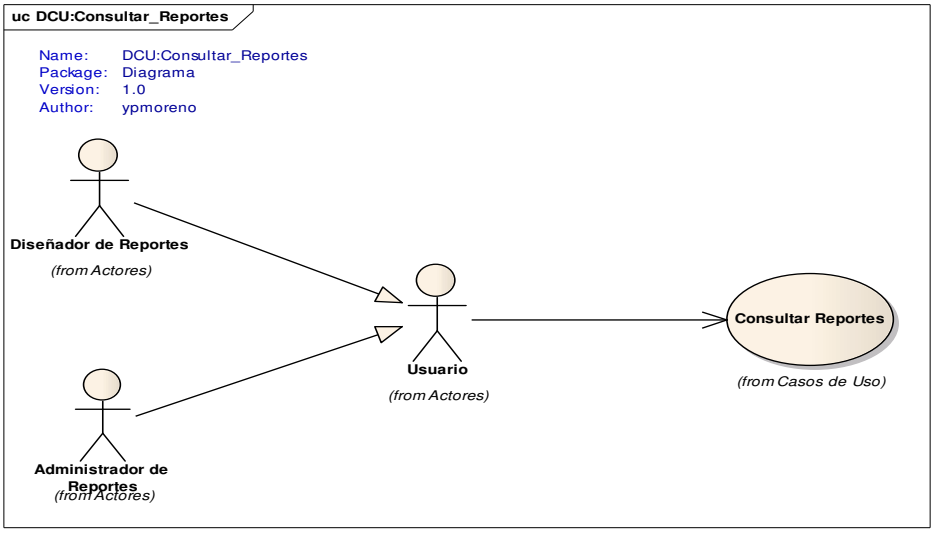


Figura2.4.DCU: Consultar Reportes

Caso de Uso:	Consultar Reportes
Actores:	Diseñador, Administrador, Usuario.
Resumen:	Este caso de uso se inicia cuando el usuario solicita que se le muestre un determinado reporte.
Precondiciones:	Que el usuario se haya autenticado.
Prioridad	Crítica

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1- El usuario solicita consultar un determinado reporte.	1.1 El sistema busca el modelo de reporte solicitado en la Base de Datos. 1.2 Muestra el visor de reportes que contiene el reporte solicitado con la información correspondiente de la Base de Datos.

Flujos Alternos	
Acción del Actor	Respuesta del Sistema
	1.1 Si no encuentra el modelo de reporte muestra una ventana indicando que no se encontró el reporte solicitado.

Poscondiciones	

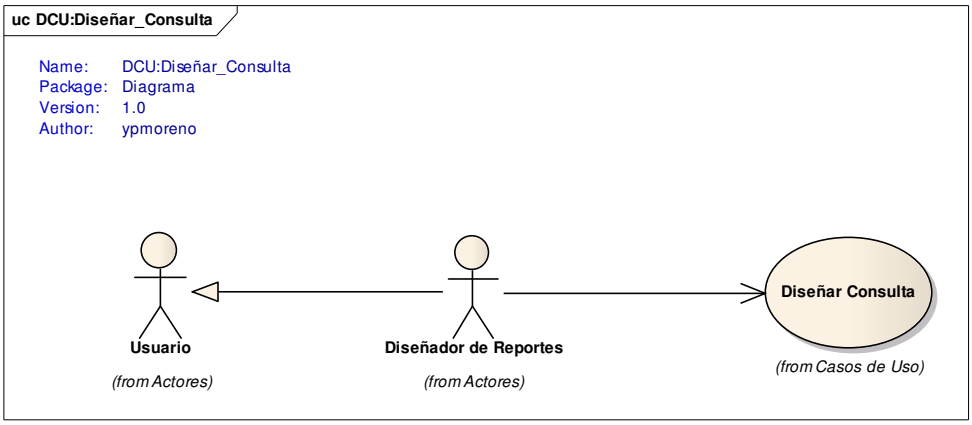


Figura2.5.DCU: Diseñar Consultas

Caso de Uso:	Diseñar Consulta
Actores:	Diseñador
Resumen:	Este caso de uso se inicia cuando el diseñador crea la consulta para obtener la información y organiza la forma en que esta información será mostrada.
Precondiciones:	Que el diseñador se haya autenticado
Prioridad	Critica

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1- El diseñador solicita crear la nueva consulta.	1.1 Muestra la ventana para generar la consulta, conteniendo información sobre las tablas y los campos disponibles.
2- El diseñador crea una consulta. (Aquí se incluye la generación de fórmulas para calcular totales, promedios, etc.)	2.1 Toma los parámetros seleccionados y construye un script con el código necesario para generar la consulta.

Flujos Alternos	
Acción del Actor	Respuesta del Sistema

Poscondiciones	

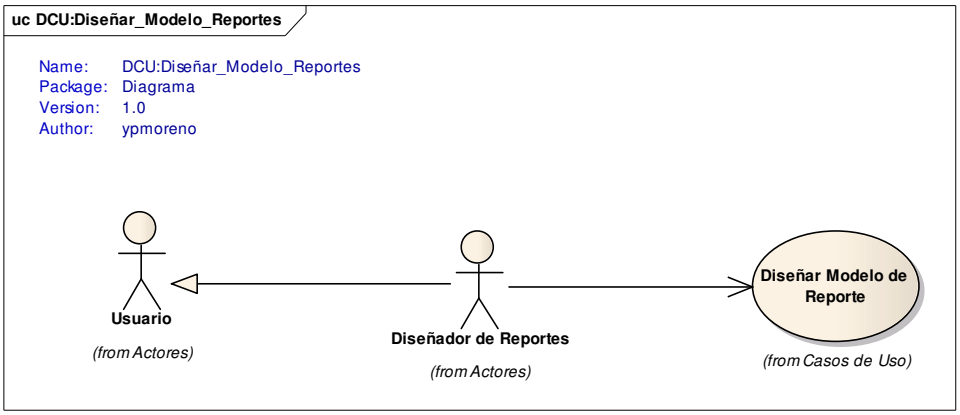


Figura2.6.DCU: Diseñar Modelo Reportes

Caso de Uso:	Diseñar Modelo de Reporte
Actores:	Diseñador
Resumen:	Este caso de uso se inicia cuando el diseñador crea un nuevo reporte y organiza la forma en que la información de la Base de Datos será mostrada.
Precondiciones:	Que el diseñador se haya autenticado
Prioridad	Crítica

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1- El diseñador crea el modelo de reporte a realizar.	1.1 Crea un nuevo modelo de reporte.
2- El diseñador diseña el modelo de reporte	2.1 Muestra la ventana para diseñar el modelo de reporte, con todos los componentes necesarios para realizar tal acción.

Flujos Alternos	
Acción del Actor	Respuesta del Sistema

Poscondiciones	

2.9 Conclusiones

La vida de un sistema informático, depende de su capacidad para brindar una respuesta a las necesidades de los usuarios que utilizarán el producto. De esta forma el cumplimiento de una serie de requisitos del sistema garantizará, finalmente, el éxito del mismo.

EL capítulo presentado muestra las principales características del sistema, se desarrolla la forma en que será concebida su realización, a través de un diagrama integrado por paquetes, agrupados por sus funcionalidades, en los cuales se abunda sobre las especificaciones de cada uno de los Casos de Uso, los cuales, junto a los actores, tratan de brindar de forma explicita, un diseño a seguir para la construcción final del sistema a desarrollar. Este capítulo también muestra los aspectos novedosos del sistema y los beneficios que traería su utilización.

Yoanki Moreno Tamayo

Capítulo 3: Análisis y Diseño

3.1 Introducción

El capítulo que se muestra a continuación expone el análisis y diseño que será usado en la implementación de la aplicación realizada por el Módulo Generador de Reportes, muestra los artefactos generados en el módulo así como la descripción de algunos de ellos.

3.2 Modelo de Análisis

En el modelo de análisis una de las principales actividades que se realiza es que se refinan los requisitos, en este modelo no se especifica cual lenguaje de programación se utilizará, la plataforma en la que se ejecutará la aplicación, los componentes prefabricados o reutilizables de otras aplicaciones, entre otras características que afectan al sistema, ya que el objetivo del análisis es comprender perfectamente los requisitos del software y no precisar cómo se implementará la solución. Durante la construcción del modelo del análisis se deben identificar cuales son las clases que describen las realizaciones de los casos de usos así como los atributos y relaciones que existen entre ellas.

Este modelo es usado para representar la estructura global del sistema, describe la realización de casos de uso y sirve como una abstracción del Modelo de Diseño

Este modelo de análisis no es un diagrama final que describe todos los posibles conceptos y sus relaciones, es un primer intento por definir los conceptos claves que describen el sistema. Este artefacto es opcional, pero también tiene a su vez la propiedad de ser temporal en el caso en que se planea su desarrollo. Su utilidad radica en que permite una apreciación global conceptual del sistema.

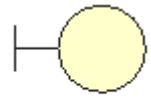
El Modelo de Análisis puede contener: las clases y paquetes de análisis, las realizaciones de los casos de uso, las relaciones y los diagramas.

Es opcional detallar aquí las realizaciones de los casos de uso ya que estas pueden estar en el modelo de diseño donde se recomienda que se encuentre. **(11)**

3.2.1 Clases del Análisis

El modelo de análisis contiene las clases del análisis y sus objetos organizados en paquetes que colaboran, las clases de análisis se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Tienen atributos y entre ellas se establecen relaciones de asociación, agregación / composición, generalización / especialización y tipos asociativos. RUP propone clasificar a las clases en:

Clases Interfaz:



nombre_interfaz

Al modelar la interacción entre el sistema y sus actores, pueden identificarse a partir de estos últimos:

- Al menos una clase que modele la interacción del actor usuario con el sistema, es decir, una clase para cada interacción actor-caso de uso sin preocuparse de que en la solución puede que se presente más de una pantalla dentro de un caso de uso para un mismo usuario.
- Una clase para cada sistema externo que será el responsable de la relación del sistema con cada uno de ellos.
- Una clase para cada actor que represente un dispositivo sobre el cual el sistema actúa o recibe información.

Clases Entidad:

nombre_entidad

Estas clases modelan información que posee una larga vida y que a menudo es persistente y fenómenos, conceptos y sucesos que ocurren en el mundo real. La fuente principal de obtención son las clases entidades del negocio y el glosario de términos que se ha ido elaborando. Algunos autores proponen un estudio del texto, a partir de las frases nominales, de manera que los sustantivos representan objetos y clases.

Clases de Control:

Las clases de control coordinan el trabajo de uno o unos pocos casos de uso, coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso, por lo que definen el flujo de control y las transacciones dentro de un caso de uso delegando el trabajo a otros objetos.

Se realiza un diagrama de clases del análisis por cada caso de uso ver **(Anexos 2).**

3.3 Diagramas de Interacción

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema, lo que conlleva modelar instancias concretas o prototípicas de clases interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento. En el contexto de las clases describen la forma en que grupos de objetos colaboran para proveer un comportamiento. **(6)**

3.3.1 Tipos de diagramas de interacción

- Diagramas de Secuencia (dimensión temporal).

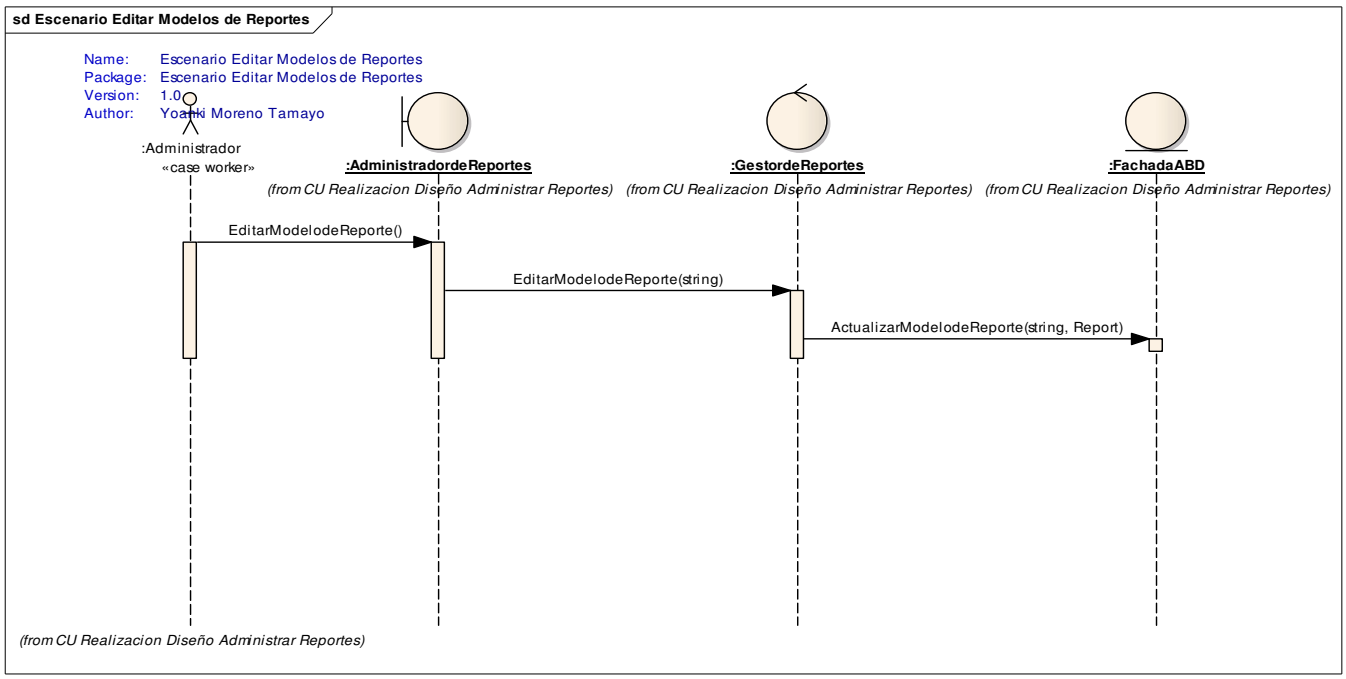


Figura3.1. DI del CU: Administrar Reporte

- Diagramas de Colaboración (dimensión estructural)

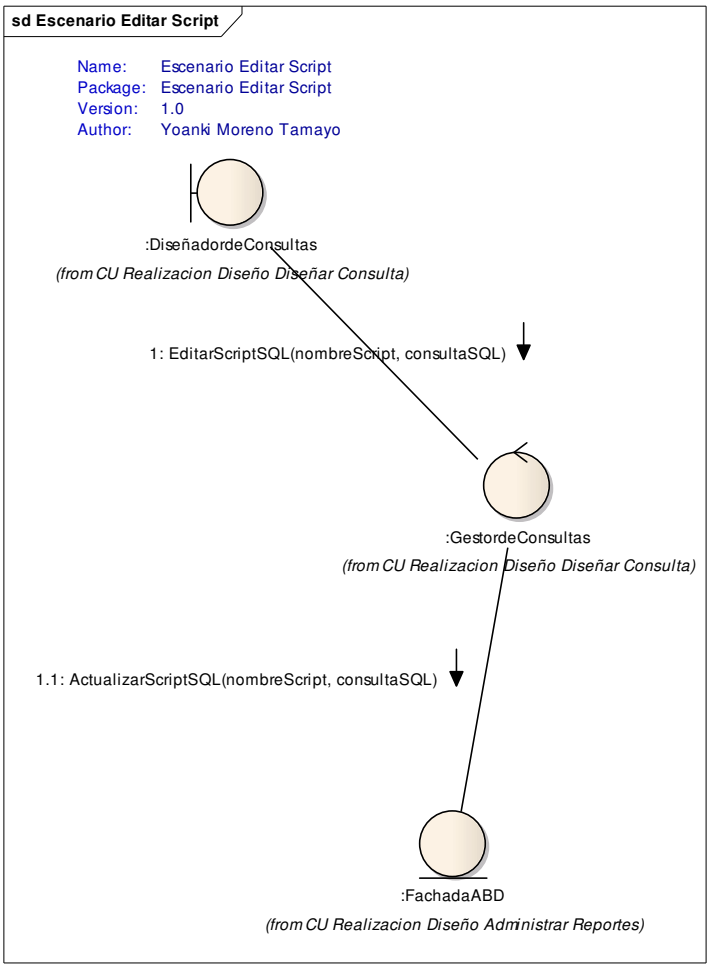


Figura3.2.DI del CU: Administrar Reporte

Se realiza un diagrama de interacción por cada escenario que tenga cada Caso de Uso, para observar los otros diagramas compruebe el **(Anexos 2)**

3.4 Modelo de Diseño.

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Sirve de abstracción de la implementación y es utilizada como entrada fundamental de las actividades de implementación.

El Modelo de Diseño puede contener: los diagramas, las clases, paquetes, subsistemas, cápsulas, protocolos, interfaces, relaciones, colaboraciones, atributos, las realizaciones de los casos de uso, entre otros que se puedan considerar para el sistema en desarrollo. **(11)**

3.5 Diagrama de Clases del diseño por paquetes

Los diagramas de clases muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras. Se dice que los diagramas de clases son diagramas "estáticos" porque muestran las clases, junto con sus métodos y atributos, así como las relaciones estáticas entre ellas: qué clases "conocen" a qué otras clases o qué clases "son parte" de otras clases, pero no muestran los métodos mediante los que se invocan entre ellas. **(12)**

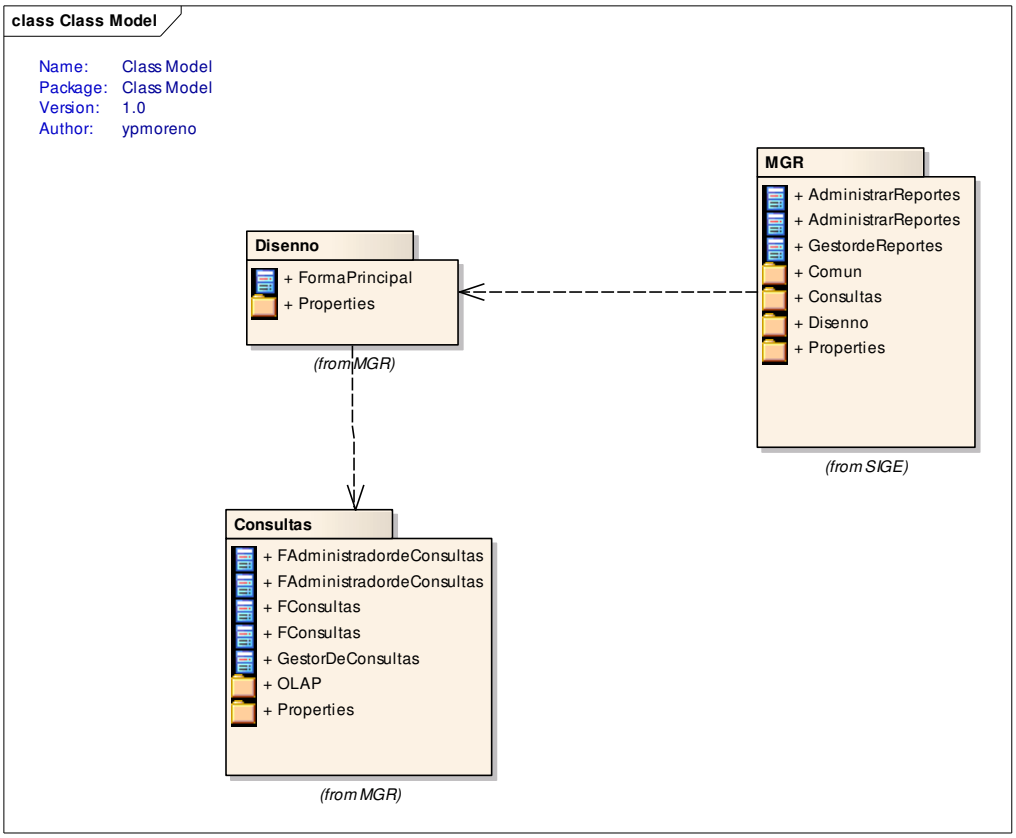


Figura3.3.Diagrama de clases del diseño por paquetes

3.5.1 Paquete Diseño

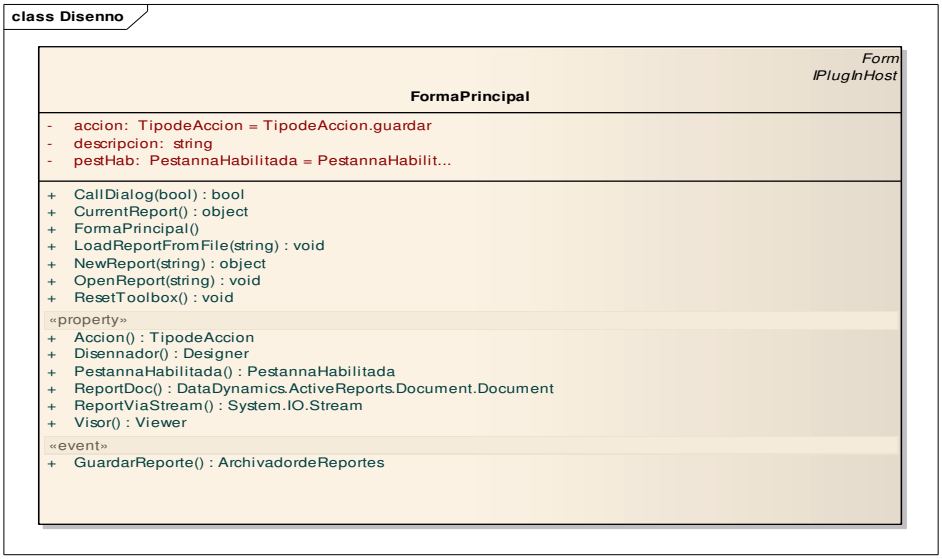


Figura3.4.Diagrama de Clase del diseño

3.5.2 Paquete MGR

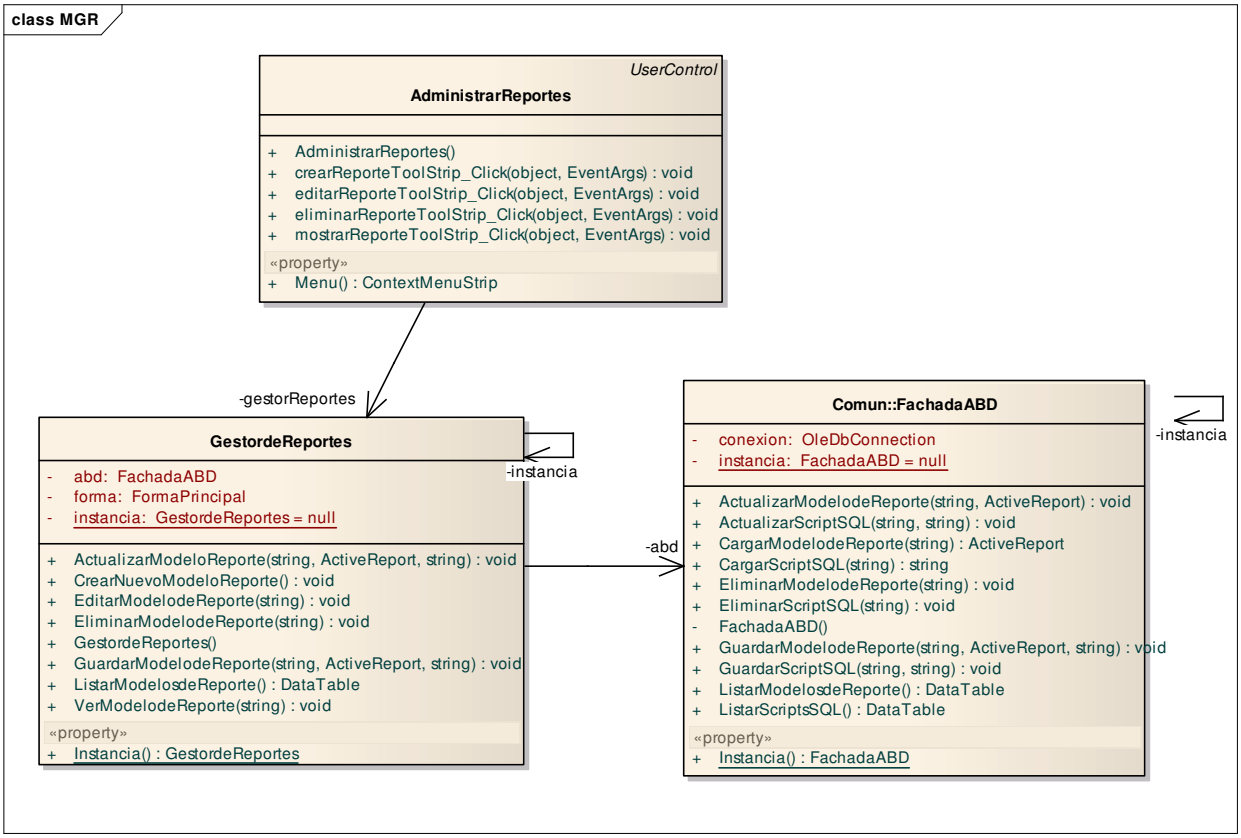


Figura3.5. Diagrama de clases del diseño

3.5.3 Paquete Consulta

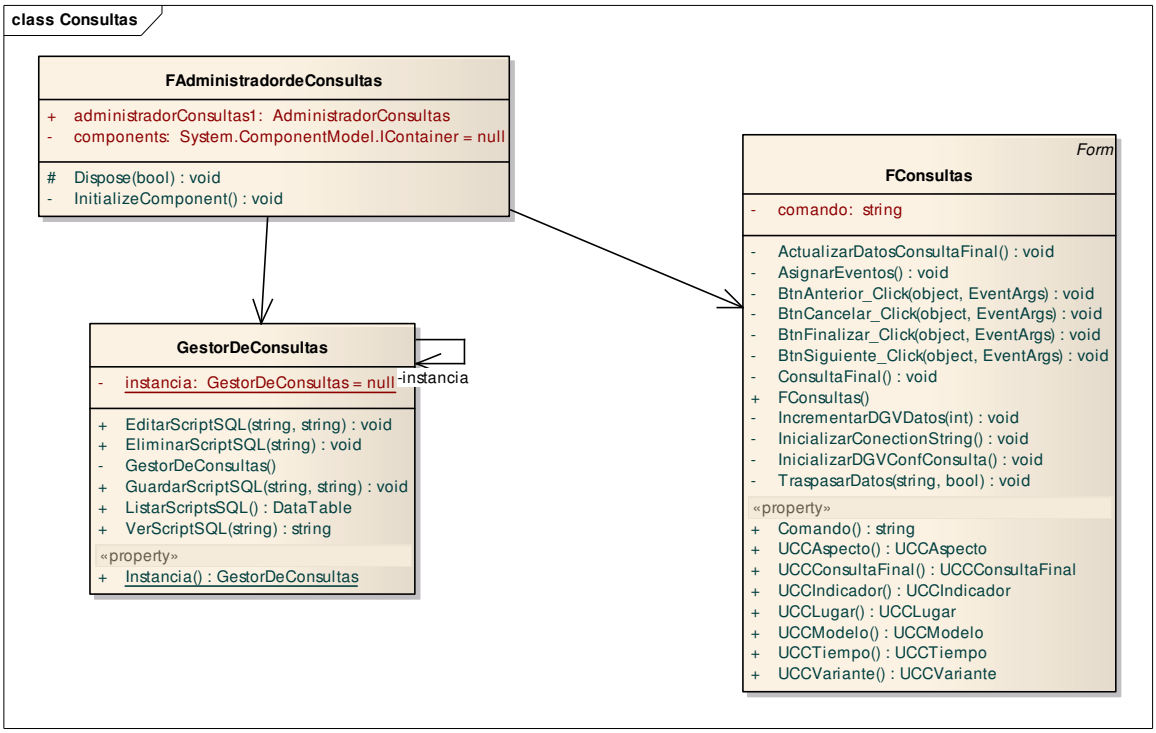


Figura3.6. Diagrama de clases del diseño

3.6 Herramienta utilizada para la capa de acceso a datos

La información que contienen los reportes es extraída de una base de datos de la cual no se comentará en este trabajo pues no fue tarea del módulo el diseño de la misma sin embargo si perteneció al módulo la capa de acceso a datos la cual fue hecha utilizando ADO.NET.

ADO.NET es un conjunto de clases que exponen servicios de acceso a datos para el programador de .NET. ADO.NET ofrece abundancia de componentes para la creación de aplicaciones de uso compartido de datos distribuidas. Constituye una parte integral de .NET Framework y proporciona acceso a datos relacionales, XML y de aplicaciones. ADO.NET satisface diversas necesidades de desarrollo, como la creación de clientes de base de datos de aplicaciones para usuario y objetos empresariales de nivel medio que utilizan aplicaciones, herramientas, lenguajes o exploradores de Internet. (11)

3.7 Patrones utilizados

Se utilizaron fundamentalmente dos patrones de diseño y uno de arquitectura.

- Patrón Singleton.
- Patrón Facade.
- Patrón tres N Capas.

Con el objetivo de lograr una instancia única en las clases se utilizó el patrón de diseño singleton patrón este que se incluye dentro de los llamados patrón de creación, el cuál además de garantizar que una clase solo tenga una instancia, proporciona un punto de acceso global a ella.

Figura3.7 Patrón Singleton

El patrón Facade, perteneciente a los patrones GoF está ubicado dentro del grupo de los patrones estructurales tiene como objetivo simplificar o hacer más amigable la complejidad asociada a una librería software, facade proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema se más fácil de usar.

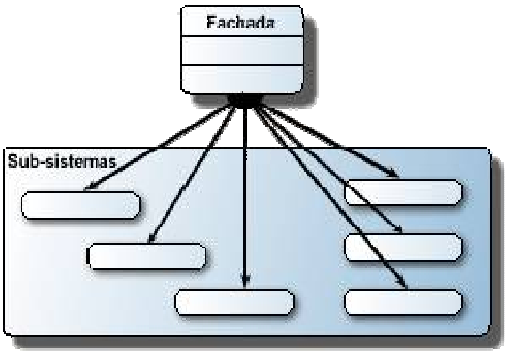


Figura3.8. Patrón Facade

Se utilizó el patrón de arquitectura N Capas este es uno de los patrones de arquitectura más potente que existe y posee numerosas ventajas entre las que se pueden mencionar: **(7)**

- Centralización de los aspectos de seguridad y transaccionalidad, que serían responsabilidad del modelo.
- No replicación de lógica de negocio en los clientes: esto permite que las modificaciones y mejoras sean automáticamente aprovechadas por el conjunto de los usuarios, reduciendo los costes de mantenimiento.
- Mayor sencillez de los clientes.
- Aísla la lógica de la aplicación y la convierte en una capa intermedia bien definida.
- En la capa de presentación se realiza relativamente poco procesamiento de la aplicación.
- Minimiza las relaciones de los elementos componentes de las diferentes capas (los cambios sobre una capa no influyen, o muy poco, sobre las demás).
- Facilita la reutilización.

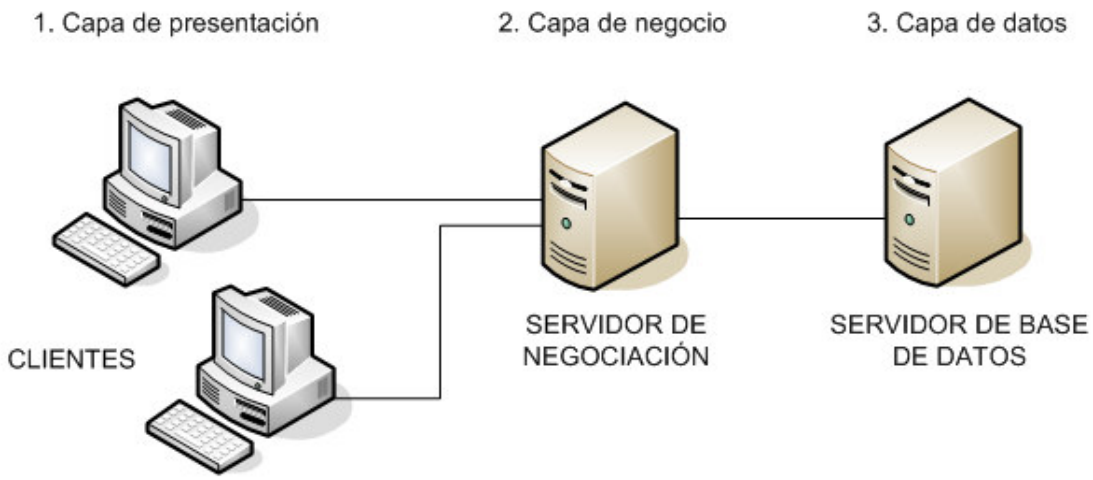


Figura3.9. Arquitectura en capas

3.8 Conclusiones

En este capítulo se realizó el modelo de análisis, presentando las clases del análisis participante, se construyó un modelo de diseño junto con todos los artefactos inherentes al mismo y se presentaron los patrones utilizados.

.

Yoanki Moreno Tamayo

Capítulo 4: Análisis de la Factibilidad

4.1 Introducción

En la ingeniería de software el proceso de medición es uno de los procesos más relevantes que se aplica pues este no solamente brinda una visión más entendible de los atributos que poseen cada uno de los modelos que se elaboran sino también que da la posibilidad de valorar la calidad del producto a construir. El flujo de trabajo análisis y diseño es un flujo en el cual se generan una gran cantidad de artefactos, es por ello que resulta de sumamente importante realizar planificaciones estimaciones y evaluaciones para con ello ayudar al cumplimiento del objetivo en el tiempo deseado y con la calidad requerida, además que la aplicación de métricas para el diseño proporciona una visión interna del sistema más palpable y ayuda en la validación del mismo.

4.2 Estimación del esfuerzo basado en Casos de Uso

El método utiliza los actores y casos de uso identificados para calcular el esfuerzo que costará desarrollarlos.

El primer paso para esta estimación es identificar los puntos de los Casos de Uso desajustados que obtienen según la ecuación **(#1).**

(1) UUCP= UAW + UUCW

Parámetros	Descripción
UUCP	Puntos de Casos de Uso sin ajustar
UAW	Factor de Peso de los Actores sin ajustar
UUCW	Factor de Peso de los Casos de Uso sin Ajustar

Tabla4.1. Ecuación (1) parámetros de la ecuación

Para calcular UAW (Factor de peso de los actores sin ajustar) lo primero que se realiza es identificar los actores que interactúan con el sistema así como el tipo de actores al que pertenecen, este último criterio se obtiene a través de la tabla 4.2

Tipo de Actor	Descripción	Factor de Peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Application Programming Interface).	1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3

Tabla4.2. Tipos de actores existentes

En el presente caso es una persona el actor encargado de interactuar con los casos de uso por lo que el tipo de actor es complejo con factor de peso 3 según la tabla 4.2. Luego para obtener el valor de **UAW** se aplica la ecuación **(#2)**:

(2) UAW= ∑ cant actores * peso

UAW= 3 ***** 3

UAW= 9

Tabla4.3. Valor de UAW

Para obtener UUCW (Factor de peso de los Casos de Uso sin ajustar) se lleva a cobo un proceso bastante parecido al anterior identificando el tipo de Caso de Uso al que pertenecen y el peso de los mismos en virtud de las transacciones que tengan cada uno, ver tabla 4.4.

Tipos de Caso de Uso	Descripción	Factor de Peso
Simple	El caso de uso que contiene de 1 a 3 transacciones.	5
Medio	El caso de uso que contiene de 4 a 7 transacciones.	10
Complejo	El caso de uso que contiene más de 8 transacciones.	15

Caso de Uso	Transacciones	Factor de peso
Administrar Usuario	5	10
Administrar Reportes	5	10
Autenticarse	3	5
Diseñar Modelo Reportes	3	5
Diseñar Consultas	3	5
Consultar Reportes	3	5

Tabla4.4. Tipos de CU y factores de peso

Los resultados de la tabla 4.4 muestra que existen 6 casos de los cuales 4 son del tipo simple y factor de peso 5, mientras que 2 casos de uso pertenecen al tipo medio y su factor de peso es 10, luego para obtener el resultado de **UUCW** se aplica la ecuación **(#3)**:

(3) UUCW= ∑ ((cant CU1 * peso CU1) +(cant CU2 * peso CU2))

UUCW= (4 * 5) + (2 * 10)

UUCW= 40

Tabla4.5. Valor de UUCW

Luego se procede a calcular UUCP pues se conocen los valores de UAW y UUCW. Aplicando la ecuación **(#1).**

UUCP= UAW + UUCW

UUCP= 9 + 40

UUCP= 49

Tabla4.6. Valor de UUCP

El segundo paso es obtener el valor de los puntos de Casos de Uso ajustados. Para ello se aplica la ecuación **(#4)**.

(4) UCP= UUCP * TCF * EF

Parámetros	Descripción
UCP	Casos de Uso ajustados
UUCP	Casos de Uso sin ajustar
TCF	Factor de complejidad técnica
EF	Factor de ambiente

Tabla4.7. Ecuación (3) y parámetros de la ecuación

Para obtener la complejidad técnica TCF de un sistema hay que tener en cuenta una serie de factores de importancia, los cuales se cuantifican con valores entre 0 y 5 en dependencia de su influencia vea tabla 4.8.

Valor	Descripción
0	No presente o sin influencia
1	Influencia incidental o presencia incidental
2	Influencia moderada o presencia moderada
3	Influencia media o presencia media
4	Influencia significativa o presencia significativa
5	Fuerte influencia o fuerte presencia

Tabla4.8. Descripción de valores de TFC

Factor	Descripción	Peso	Valor	Σ(peso x valor)
T1	Sistema distribuido.	2	0	0
T2	Objetivos de performance (funcionamiento) o tiempo de respuesta.	1	4	4
T3	Eficiencia del usuario final.	1	4	4
T4	Procesamiento interno complejo.	1	2	2
T5	El código debe ser reutilizable.	1	4	4
T6	Facilidad de instalación.	0.5	4	2
T7	Facilidad de uso.	0.5	4	2
T8	Portabilidad.	2	5	10
T9	Facilidad de cambio.	1	4	4
T10	Concurrencia.	1	4	4
T11	Incluye objetivos especiales de seguridad.	1	5	5
T12	Provee acceso directo a terceras partes.	1	0	0
T13	Se requieren facilidades especiales de entrenamiento a usuarios.	1	2	2

Tabla4.9. Valores asignado por influencia a cada factor

Después de analizar los datos de la tabla 4.9, se puede calcular TFC según la ecuación **(#5)**.

(5)TFC= 0.6 + 0.01 * ∑ (peso i * Valor i)

TFC= 0.6 + 0.01 * ∑ (0 + 4 + 4 + 2 + 4 + 2 + 2 + 10 + 4 + 4 + 5 + 0 + 2)

TFC= 0.6 + 0.01 * 43

TFC= 1.03

Tabla4.10. Valor de TFC

El factor ambiente EF se determina teniendo en cuenta cada una de las habilidades y experiencias con que cuentan las personas que van a trabajar en el proyecto, estos parámetros tienen un gran peso en la estimación de tiempo y se le otorgan valores entre 0 y 5.

Factor	Descripción	Peso	Valor	Σ(peso x valor)
E1	Familiaridad con el modelo de proyecto utilizado	1.5	2	4.5
E2	Experiencia en la aplicación	0.5	1	1.5
E3	Experiencia en orientación a objetos	1	4	4
E4	Capacidad del analista líder	0.5	4	2
E5	Motivación	1	5	5
E6	Estabilidad de los requerimientos	2	4	8
E7	Personal a tiempo compartido	-1	2	-2
E8	Dificultad del lenguaje de programación	-1	3	-3

Tabla4.11. Muestra el valor y el peso de cada uno de éstos factores

- Para los factores E1 al E4, los valores asignados de 0 significan sin experiencia, 3 con experiencia media y 4 experiencia media-avanzada.
- Para el factor E5, 5 significa alta motivación.
- Para el factor E6, 0 significa requerimientos extremadamente inestables, 3 estabilidad media y 5 requerimientos estables sin posibilidad de cambios.
- Para el factor E7, 5 significa que todo el personal trabaja a tiempo compartido (nadie trabaja a tiempo completo).
- Para el factor E8, 3 significa que el lenguaje de programación es medianamente fácil de usar.

Apoyándose en los resultados arrojados por la tabla 4.11 Se puede calcular el factor ambiente EF según la ecuación **(#6)**:

(6) EF=1.4 – 0.03 * ∑ (Peso i * Valor i)

EF=1.4 – 0.03 * ∑ (4.5 + 1.5 + 4 + 2 + 5 + 8 – 2 - 3)

EF=1.4 – 0.03 * 20

EF= 0.80

Tabla4.12. Valor de EF

Después de haber obtenido todos los valores pertinentes se procede a calcular el valor de UCP. Aplicando la ecuación **(#4)**

UCP = UUCP * TCF * EF

UCP = 49 * 1.03 * 0.80

UCP= 40.376

Tabla4.13. Valor de UCP

Cálculo del esfuerzo en horas – hombres (E)

Cuando se habla de esfuerzo en este contexto se refiere a la relación entre la cantidad de hombres y el tiempo. Para hallar este valor depende del valor de los Puntos de Casos de Uso Ajustados y el Factor de Conversión, como lo muestra la ecuación **(#7)** en la tabla 4.14:

(7) E= UCP * CF

Parámetros	Descripción
E	Esfuerzo estimado en horas-hombres
UCP	Casos de Uso ajustados
CF	Factor de conversión

Tabla4.14. Ecuación (7) y parámetros de la ecuación

Conversión de los puntos de Casos de Uso ajustados a esfuerzo de desarrollo

Para la búsqueda del Factor de Conversión CF se siguen una serie de pasos que contribuirán a la toma de un criterio:

- Se contabilizan cuántos factores de los que afectan al factor de ambiente están por debajo del valor medio (3), para los factores desde E1 hasta E6.
- Se contabilizan cuántos factores de los que afectan al factor de ambiente están por encima del valor medio (3), para los factores E7 y E8.

En el presente caso existen dos factores desde E1 hasta E6 cuyo valor es menor que 3, mientras que de E7 a E8 no hay ninguno con valor mayor que 3.

Para este caso la cantidad de factores por debajo del valor medio es 1. Luego se analizan los parámetros:

CF= 20 horas – hombre (si Total $_{EF} \leq 2$)

CF= 28 horas – hombre (si Total $_{EF} = 3$ o Total EF = 4)

CF= Abandonar o cambiar proyecto (si Total ≥ 5)

Total $_{EF}$= cant EF < 3 entre (E1 y E6) + cant EF > 3 entre (E7 y E8)

Total EF= 2 + 0

Total EF= 2

CF = 20 horas – hombres (por que Total EF= 2)

Tabla4.15. Valor de CF

Finalmente se puede calcular el Esfuerzo estimado en horas – hombres E, aplicando la ecuación **(#7)**

E=UCP * CF

E = 40.376 * 20

E= 807.52 hora – hombres

Tabla4.16. Valor de E

La estimación por puntos de Casos de Uso brinda una estimación del esfuerzo en horas-hombres solamente tomando en cuenta el desarrollo de la funcionalidad especificada en los casos de usos es decir para la implementación del sistema pero si se quiere obtener la duración del proyecto entero, hay que agregarle a esta estimación obtenida las estimaciones del esfuerzo de las otras actividades involucradas en el desarrollo del software, se ha considerado estadísticamente aceptable una serie de criterios que brindan la distribución del esfuerzo por flujos de trabajos a las que se le han calculado el valor (horas – hombres) la tabla 4.17 Muestra la mencionada distribución.

Actividad	% esfuerzo	Valor esfuerzo
Análisis	10%	201.88 horas-hombre
Diseño	20%	403.73 horas-hombre
Implementación	40%	807.52 horas-hombre
Prueba	15%	302.82 horas-hombre
Sobrecarga	15%	302.82 horas-hombre
Total	100%	2018.8 horas-hombre

Tabla4.17. Estándar para el esfuerzo por flujos de trabajo

Si **ET = 2018.8 horas-hombre** y se estima que cada mes tiene 30 días de los cuales se trabaja aproximadamente 24 y se dedica al trabajo 4 horas diarias se trabajaría aproximadamente 96 horas al mes.

Si el esfuerzo total (ET) es de 2018.8 horas - hombres y 96 horas equivale a un mes de trabajo de un hombre entonces el (ET=21.02 mes-hombre). Lo que quiere decir que un hombre puede realizar el trabajo del módulo completo en 21 meses aproximadamente.

Si el grupo de trabajo consta de 3 personas, al mes completarían 288 horas de trabajo lo que quiere decir que se terminaría el trabajo en aproximadamente 7 meses.

4.3 Métricas del Diseño

Las métricas de diseño de alto nivel se concentran en las características de la arquitectura del programa, con especial énfasis en la estructura arquitectónica y en la eficiencia de los módulos. Estas métricas son de caja negra en el sentido que no requieren ningún conocimiento del trabajo interno de un módulo en particular del sistema.

El presente epígrafe muestra el cálculo de la complejidad del diseño en virtud de las ecuaciones definidas en el capítulo 1.

Nombre de la medida	Notación	Ecuación	Descripción
Complejidad Estructural	$S(i)$	$S(i) = f^{2}_{out}(i)$	$f_{out}(i)$: *relación del módulo*
Complejidad de Datos	$D(i)$	$D(i) = V(i) / \lfloor f_{out}(i) + 1 \rfloor$	$V(i)$: *número de variables de entrada y salidas que entran y salen del módulo i.*
Complejidad del Sistema	$C(i)$	$C(i) = S(i) + D(i)$	

Tabla4.18. Medidas de complejidad del diseño

A continuación se procede a determinar el tipo de complejidad que posee el módulo, para ello se tienen en cuenta los siguientes detalles:

complejidad	$S(i)$	$D(i)$	$C(i)$
No complejo	1, 4, 9, 16 , 25	<=7	<=23
Complejo	36, 49, 64	>7 <=12	>23 <=50
Muy complejo	81<	>12	>50

Tabla4.19. Umbrales de complejidad

Existe una relación directamente proporcional entre los valores de la complejidad y la complejidad arquitectónica global del sistema, lo que quiere decir que uno aumenta conforme el otro, este provoca que exista un aumento en las probabilidades de que el esfuerzo que se tenga que realizar para la integración y las pruebas sea mayor.

La puesta en prácticas de estas métricas brinda la posibilidad de medir indicadores de relevancia como son la integración entre módulos y la complejidad de las pruebas que se hacen en los mismos.

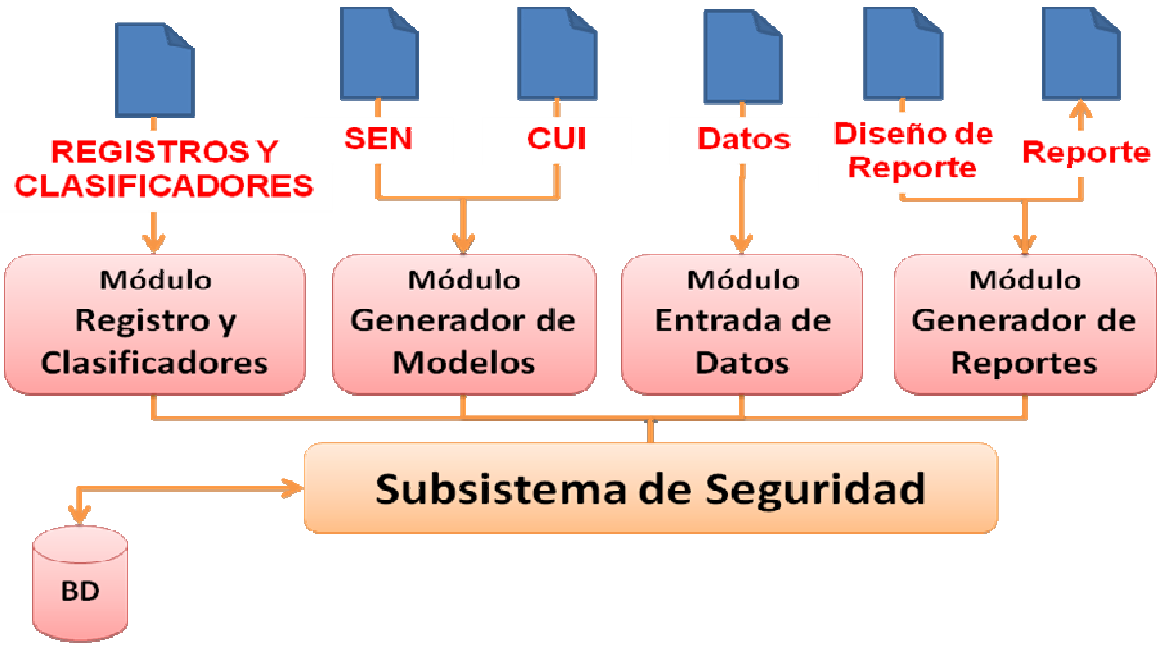


Figura4.1. Relaciones entre Módulos

Ecuaciones de la complejidad

$S(i)$ = 4² = 16

$D(i)$ = 57 / [3 + 1]= 14.25

$C(i)$ = 14.25 + 4= 18.25

Tabla4.20. Valor de la complejidad

Una ves obtenidos estos resultados se procede a evaluar los mismos teniendo en cuenta los parámetros de la tabla 4.19

Tipo de complejidad	Clasificación
Complejidad Estructural	No complejo
Complejidad de Datos	Muy Complejo
Complejidad del Sistema	No complejo

Tabla4.21. Clasificación de la complejidad

4.4 Métricas Orientadas a Clases

4.4.1 Serie de Métricas CK

Los ingenieros (Chidamber y Kemerer) han propuesto un conjunto de métricas orientadas a objetos que han sido ampliamente referenciadas nivel mundial, son identificadas como la serie de métricas CK en la cuales los autores han propuesto seis métricas para clases en sistemas orientados a objetos.

- Métodos ponderados por clase (**MPC**).
- Árbol de profundidad de herencia (**APH**).
- Número de descendiente (**NDD**).
- Acoplamiento entre clases objeto (**ACO**).
- Respuesta para una clase (**RPC**).
- Carencia de cohesión en los métodos (**CCM**).

4.4.1.1 Árbol de profundidad de herencia (APH)

Esta métrica se define como la máxima longitud del nodo a la raíz del árbol, a medida que el APH crece, es posible que clases de más bajos niveles hereden muchos métodos. Esto conlleva dificultades potenciales, cuando se intenta predecir el comportamiento de una clase. Una jerarquía de clases profunda (el APH es largo) también conduce a una complejidad de diseño mayor. Por el lado positivo, los valores APH grandes implican un gran número de métodos que se reutilizarán.

Algunos autores sugieren un umbral de 6 niveles como indicador de un abuso en la herencia en distintos lenguajes de programación.

Resultado de la métrica (APH)

En virtud del planteamiento de la métrica (APH) y obedeciendo a los datos obtenidos después de aplicar la métrica al módulo generador de reporte, se llega a la conclusión de que el diseño no es complejo, no existe un alto acoplamiento y no es de difícil mantenimiento ya que el nivel de herencia en el módulo es uno.

4.5 Conclusiones

En este capítulo se realizaron mediciones de varios tipos al sistema propuesto, para ello se utilizaron métricas que aportaron un nivel de seguridad al trabajo realizado pues en cada uno de los casos se llegaron a resultados convincentes y lógicos, primeramente se estimó el esfuerzo necesario para terminar el trabajo completo del módulo, luego se demostró que en sentido general el diseño no posee una gran complejidad lo cual posibilita que la integración entre los módulo se pueda llevar a cabo sin inconvenientes y que se puedan aplicar diferentes tipos de prueba para validar el funcionamiento, por último se aplico una métrica de clase para ver la complejidad de la herencia existente en el sistema y se determinó que no existe complejidad pues el nivel de herencia es ínfimo.

Conclusiones Generales

Construir un sistema gestor estadístico es un trabajo complejo, para ello tienen que conjugarse una serie de factores que posibiliten que tal actividad sea llevada a cabo con organización y responsabilidad si es que se quiere obtener resultados positivos, una ves terminado este trabajo y con la satisfacción de haber logrado cumplir el objetivo propuesto se arribó a las siguientes conclusiones.

- Se escogió la herramienta **Active Reports** para el trabajo en el módulo.
- Después de un análisis minucioso y crítico de herramientas **CASE** quedó definida para el trabajo la herramienta **Enterprise Architect**, la cual usa el lenguaje de modelado **UML** para la construcción de diagramas y modelos. Se usará la metodología **RUP**.
- Seleccionada la plataforma **.NET** para el desarrollo del módulo, usándose el **visual studio.net** como **IDE** y lenguaje de programación **C#**.
- Con la correspondiente modelación del negocio, el análisis de los requisitos y la especificación de los Casos de Usos del sistema se sentaron las bases para desarrollar todo el trabajo de manera coherente y poder dar cumplimiento a las expectativas del cliente.
- Se tomaron en cuenta los conceptos empleados en la fundamentación teórica, los principios del diseño así como el uso de patrones para elaborar un diseño poco complejo y flexible.
- Con el uso del **EA** y en virtud de la metodología seleccionada se obtuvieron todos los artefactos necesitados por el módulo, quedando estos bien definidos y explicados en el presente trabajo.
- Se demostró la valía que supone la implantación del sistema en la **ONE**, exponiendo sus principales características y aspectos novedosos.
- Se realizó un estudio de la factibilidad, aplicando **métodos** y **métricas** para la validación del **análisis y diseño** que arrojaron resultados positivos

Recomendaciones

Tras haber cumplidos los objetivos trazados en el presente trabajo se recomienda:

- Que se realice la implementación del sistema de forma eficiente.
- Que se le apliquen todas las pruebas pertinentes y luego se instale en la Oficina Nacional de Estadísticas.
- Que se valore la migración hacia software libre.
- Que se agregue al sistema la funcionalidad de interpretar código de barras, para facilitar la identificación de la entidad creadora del reporte.

Referencias Bibliográficas.

(1).**DataDynamics**. [Online] 1997. [Cited: mayo 11, 2008.] http://www.datadynamics.com/Products/ProductOverview.aspx?Product=ARNET3.

(2). **J Gracias.** Ingenieros de Software. [Online] 2003. [Cited: Abril 14, 2008.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno .

(3).**IBM Rational.** [Online] [Cited: Abril 8, 2008.] http://www.rational.com.

(4).**Microsoft.** MSDN. [Online] [Cited: Mayo 14, 2008.] from http://msdn.microsoft.com/es-es/vstudio/products/default.aspx.

(5). **Sparx.** [Online] [Cited: mayo 25, 2008.] http://www.sparxsystems.com.ar/products/ea.html.

(6).**ReallTech.** [Online] 2001. [Cited: Febrero 10, 2008.] http://www.sqlmax.com/reportin_services1.asp.

(7). **Viklund.** ALPHASITE. [Online] [Cited: Abril 19, 2008.]

(8). **CIBERAULA.** [Online] 1996. [Cited: Abril 18, 2008.]

(9). **msdn.** [Online] [Cited: Enero 20, 2008.] http://msdn.microsoft.com/en-us/vcsharp/default.aspx

(10). **Microsoft.** MSDN. [Online] [Cited: Mayo 16, 2008.] http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3317/default.aspx#M15.

(11). **msdn.** [Online] [Cited: Junio 4, 2008.] http://msdn.microsoft.com/es-es/library/e80y5yhx(VS.80).aspx.

(12). KDE Documentation [Online] [Cited: Junio 6, 2008.] http://docs.kde.org/kde3/es/kdesdk/umbrello/uml-elements.html

(13). PRESSMAN, R. (2002). *Ingeniería de Software "Un enfoque práctico"* (5ta edición ed.). Madrid: Graw Hill.

Bibliografía

1. **Pressman.** *Ingenieria un enfoque practico.* 1998.

2. **IngenieroSoftware.** [Online] 2003. [Cited: 6 1, 2008.] http://www.ingenierosoftware.com/analisisydiseno/uml.php.

3. **Marroquin.** Wily.net. [Online] Febrero 13, 2003. [Cited: Abril 11, 2008.] http://www.willydev.net.

4. **Red de Software Picks.** [Online] [Cited: Abril 11, 2008.] http://espanol.softpicks.net/software/Rpv-Printing-System_es-32462.htm.

5. **Reynox.** Reynox Servicios Informaticos. [Online] 2005. [Cited: 20 Enero, 2008.] http://reynox.com/publicaciones/.

6. **Gris nd.** [Online] [Cited: Abril 14, 2008.] http://www-gris.det.uvigo.es/~avilas/ISII..

7. **Sanchez.** ONESS. [Online] 2003. [Cited: Abril 18, 2008.] http://oness.sourceforge.net/proyecto/html/ch03s02.html.

8. **UNSA nd.** Unsa. [Online] [Cited: Mayo 12, 2008.] http://bo.unsa.edu.ar .

9. **Galeon.** Hispavista. [Online] 1996. [Cited: Mayo 13, 2008.] http://galeon.hispavista.com.

10. **Guiarte Multimedia.** Desarrollo Web. [Online] [Cited: Mayo 11, 2008.] http://www.desarrolloweb.com.

11. **Gobierno Bolivariano de Venezuela.** MeRinde. [Online] [Cited: Mayo 22, 2008.] http://merinde.rinde.gob.ve/index.php.

12. **V P Company**. Visual Parading. [Online] 2002[Cited: Mayo 4, 2008] http://www.visual-paradigm.com/news/vpsuite33/vpuml63.jsp

13. http://iie.fing.edu.uy/ense/asign/desasoft/practico/hoja8/ejemplos_clase2.pdf.

Yoanki Moreno Tamayo

Glosario de Términos

ONE: "Oficina Nacional de Estadísticas"

OTE: "Oficina Territorial de Estadísticas"

OME: "Oficina Municipal de Estadísticas"

CI: "Centros Informantes"

SEN: "Sistema Estadístico Nacional"

SIGE: "Sistema Integrado de Gestión Estadística"

MGR: "Módulo Generador de Reportes"

RUP: "Rational Unified Process" (Metodología de desarrollo de software)

UML: "Lenguaje Unificado de Modelado"

EA: "Enterprise Architect"

CASE: "Computer Aided Software Engineering"

CU: "Casos de Usos"

DCU: "Diagrama de Casos de Usos"

DI: "Diagrama de Interacción"

DCA: "Diagrama de clase del análisis"

CUS: "Casos de Uso del Sistema"

IDE: "Entorno de Desarrollo Integrado"

Anexos

1.2 Diagrama de Casos de uso del Sistema

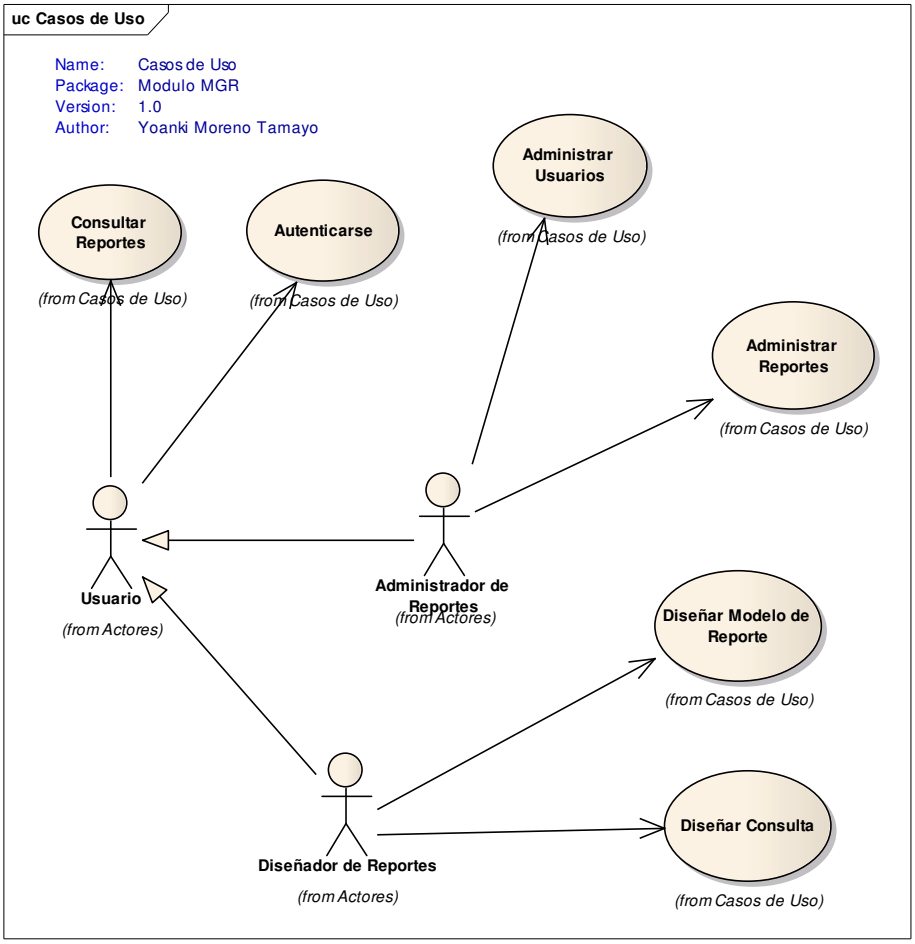


Diagrama de Casos de usos del Sistema

1.3 Especificación de los casos de Uso

1.3.1 Administrar Usuarios.

Caso de Uso:	Administrar Usuarios
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el administrador accede al sistema, y lleva a cabo ciertas funcionalidades como el control de usuarios, la asignación de cierto nivel

	de acceso, así como privilegios. El administrador adiciona o elimina usuarios, y le asigna los roles.
Precondiciones:	Que el Administrador se haya autenticado
Prioridad	Crítica

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El administrador selecciona la opción que desea ejecutar.	1.1 a) Si desea insertar un nuevo usuario al sistema ir a Sección "Insertar Usuario" b) Si desea modificar el rol de un usuario ir a Sección "Modificar Rol" c) Si desea eliminar un determinado usuario ir a Sección "Eliminar usuario"
Sección "Insertar Usuario"	
Acción del Actor	Respuesta del Sistema
2. El administrador introduce los datos del nuevo usuario.	2.1 El sistema registra los datos del usuario en la Base de Datos.
	2.2 El sistema muestra el listado de los usuarios previamente insertados más los del nuevo ingreso.
Sección "Modificar Rol"	
Acción del Actor	Respuesta del Sistema
3. El administrador indica el usuario a modificar su rol.	3.1 El sistema busca al usuario y muestra una interfaz al administrador con los datos del mismo.
4. Asigna el rol	4.1 Si lo encuentra el sistema modifica su rol en la Base de Datos.
	4.2 Muestra el listado de usuarios actualizado.
Sección "Eliminar Usuario"	
Acción del Actor	Respuesta del Sistema
5. El administrador indica el usuario que	5.1 El sistema busca el usuario en la Base de Datos.

desea eliminar.	5.2 Si lo encuentra lo elimina de la Base de Datos.
	5.3 Muestra el listado actualizado de usuarios.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
	[4.1, 5.2]Si no lo encontró muestra un mensaje que indica que el usuario o se encuentra en la Base de Datos.
Poscondiciones	

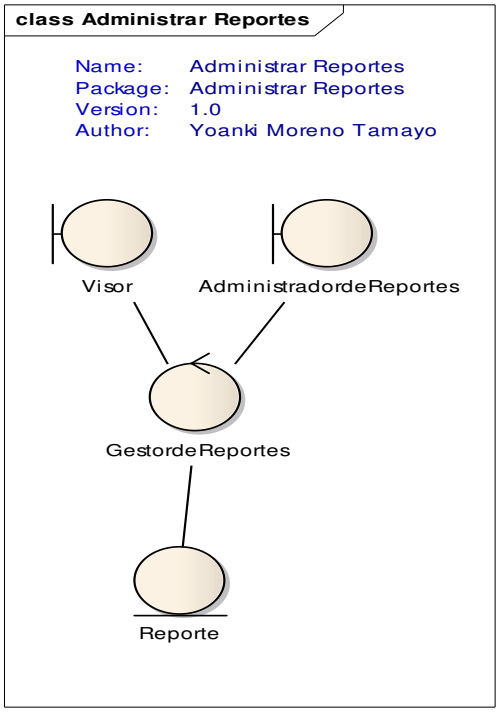
1.3.3 Autenticarse.

Caso de Uso:	Autenticarse
Actores:	Usuario del Sistema
Resumen:	El usuario introduce su nombre de usuario y contraseña. El sistema verifica si está, si es así le da el acceso a las funcionalidades especificas, configuradas para el rol.
Precondiciones:	
Prioridad	Crítico

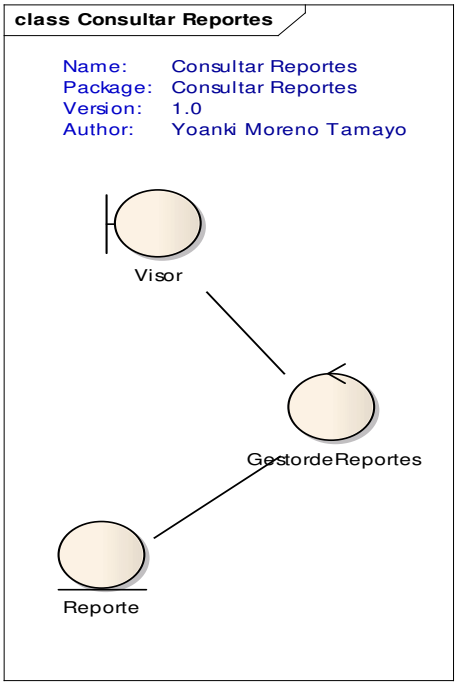
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El usuario introduce Usuario y Contraseña.	1.1 El sistema comprueba que el usuario exista en la base de datos. 1.2 Si lo encuentra verifica su contraseña es correcta. 1.3 Si es correcta, crea los datos del usuario y accede al sistema.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
	[1.1] El sistema muestra un mensaje ”El usuario no existe”
	[1.2] El sistema muestra un mensaje ”Su contraseña es incorrecta”
Poscondiciones	

Anexos 2 Análisis y Diseño

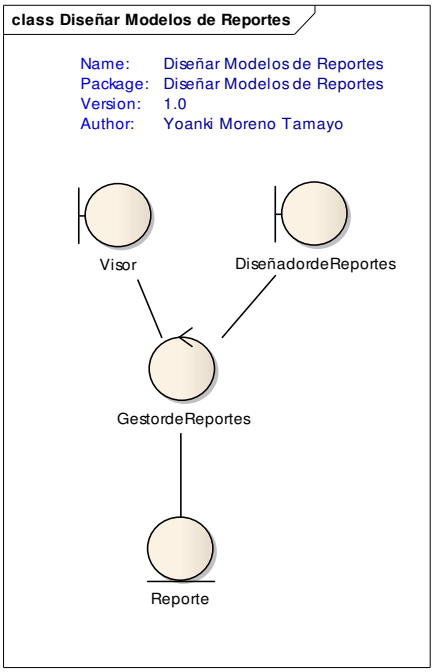
2.1 Diagramas de clase de Análisis



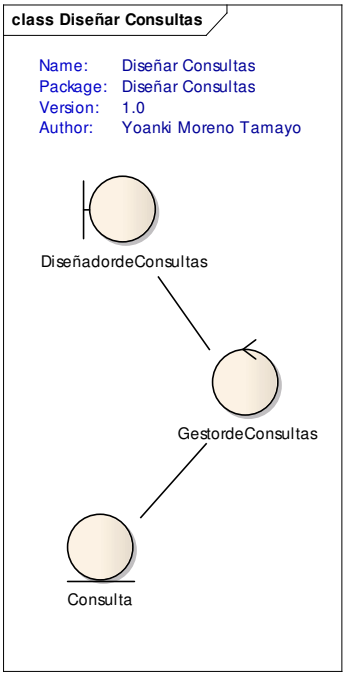
2.1.1 DCA_CU: Administrar Reporte



2.1.2 DCA_CU: Consultar Reporte

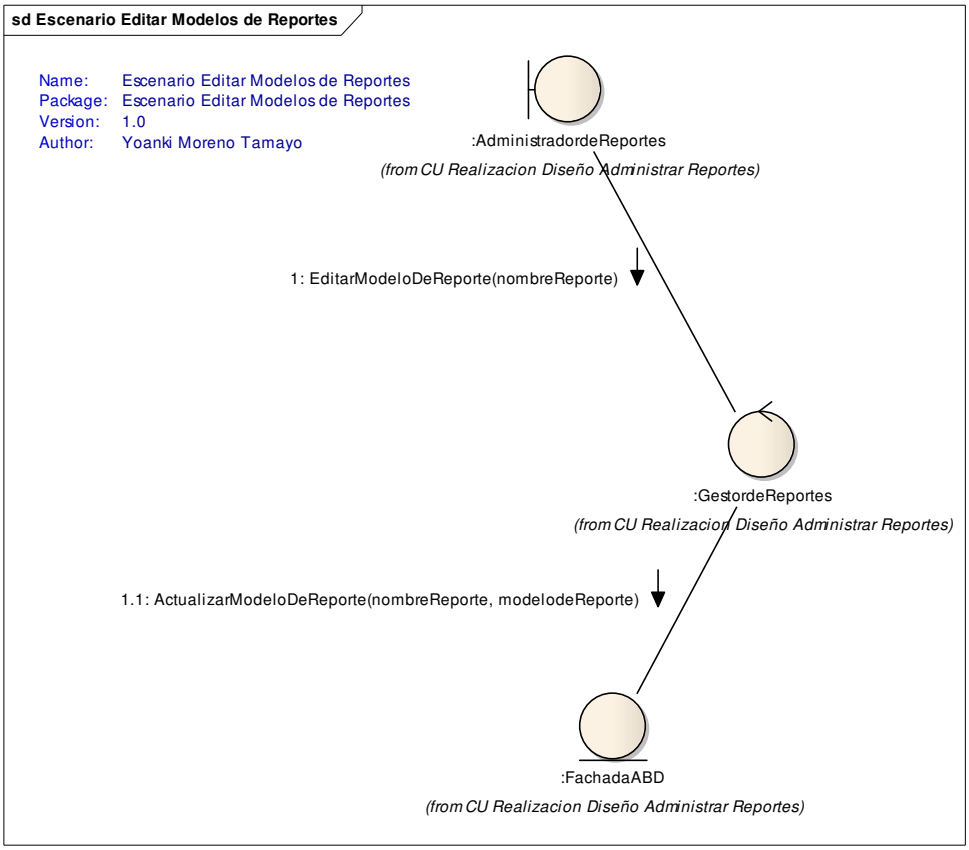


2.1.3 DCA_CU: Diseñar Modelo Reporte

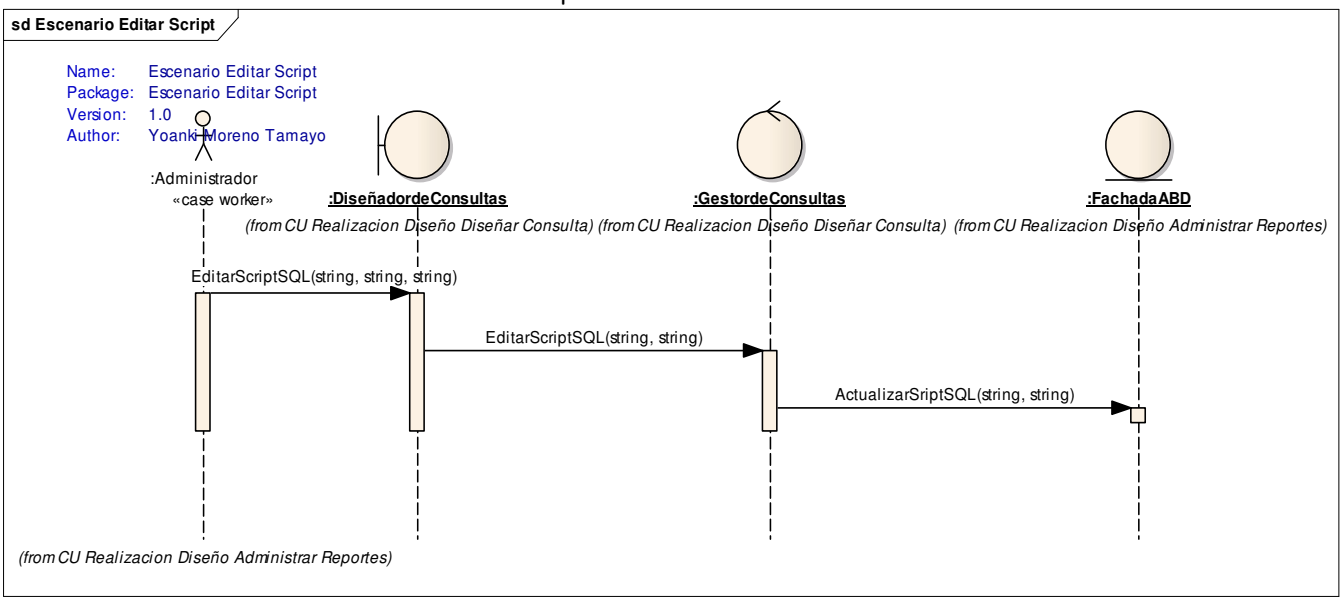


2.1.4 DCA_CU: Diseñar Consultas

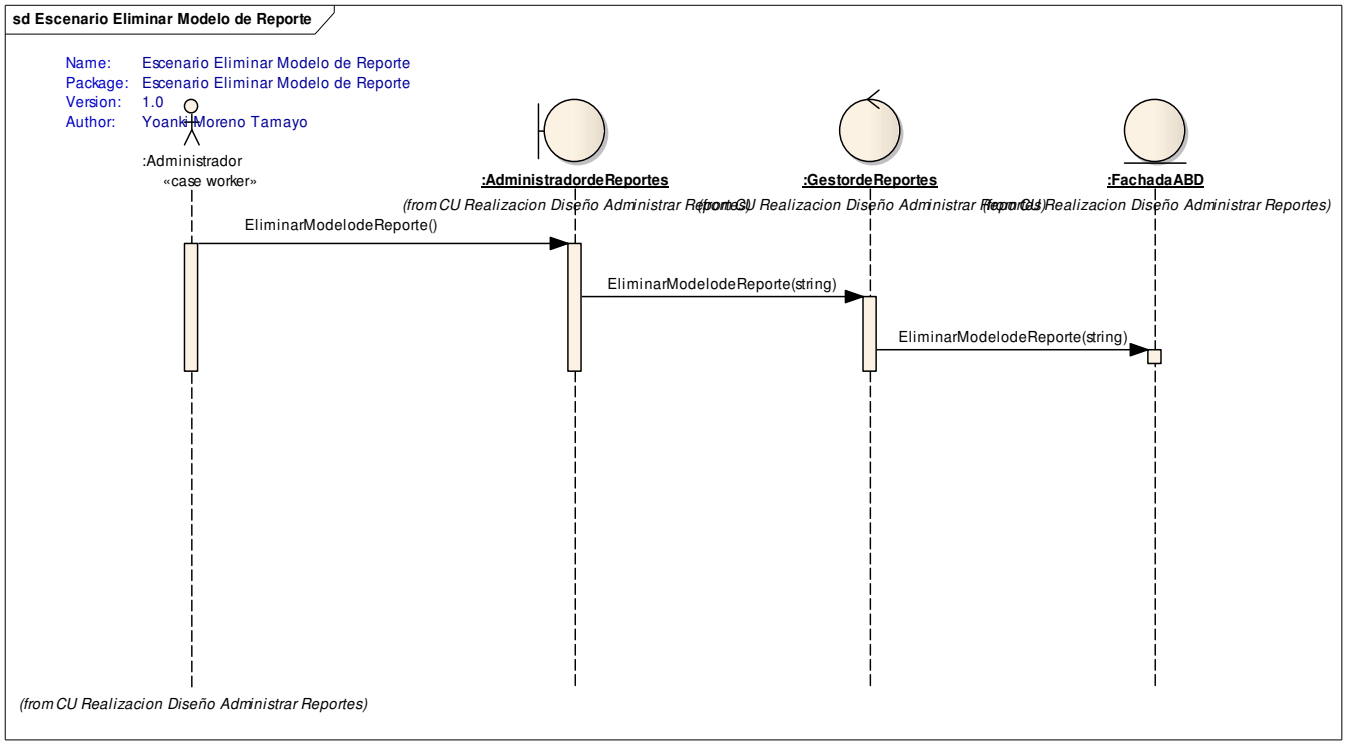
Anexos 2.2 Diagramas de Interacción



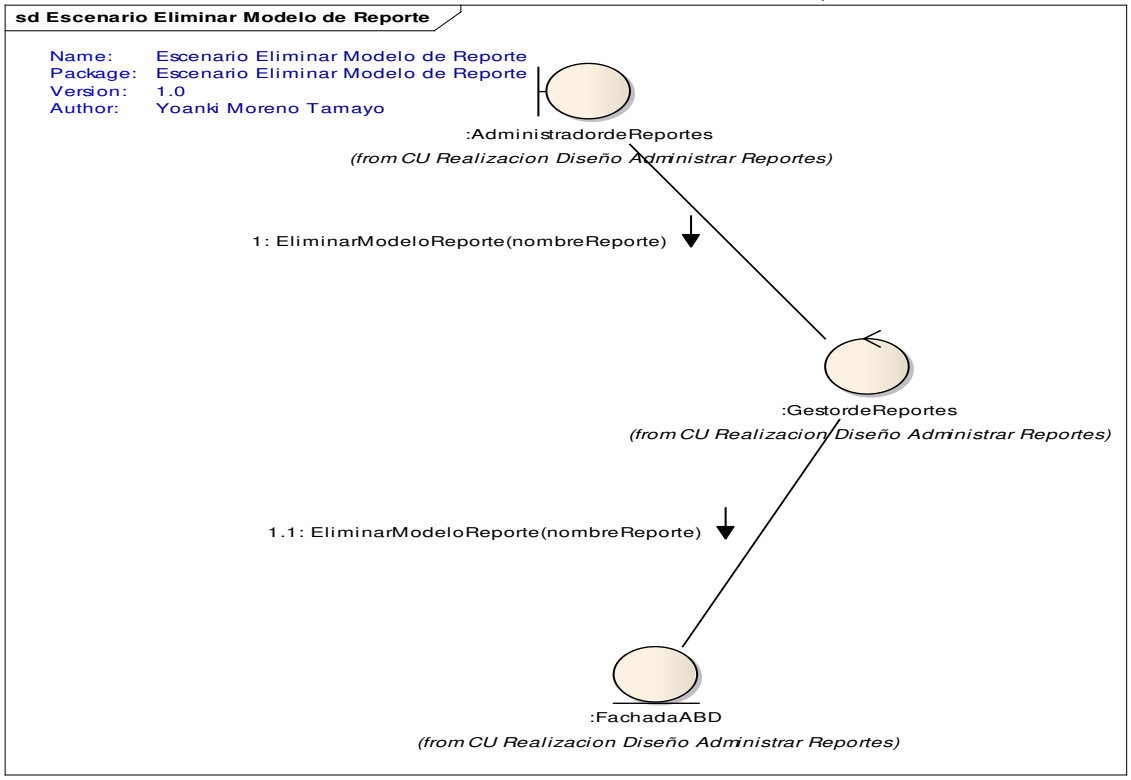
2.2.1 DI del CU: Administrar Reportes



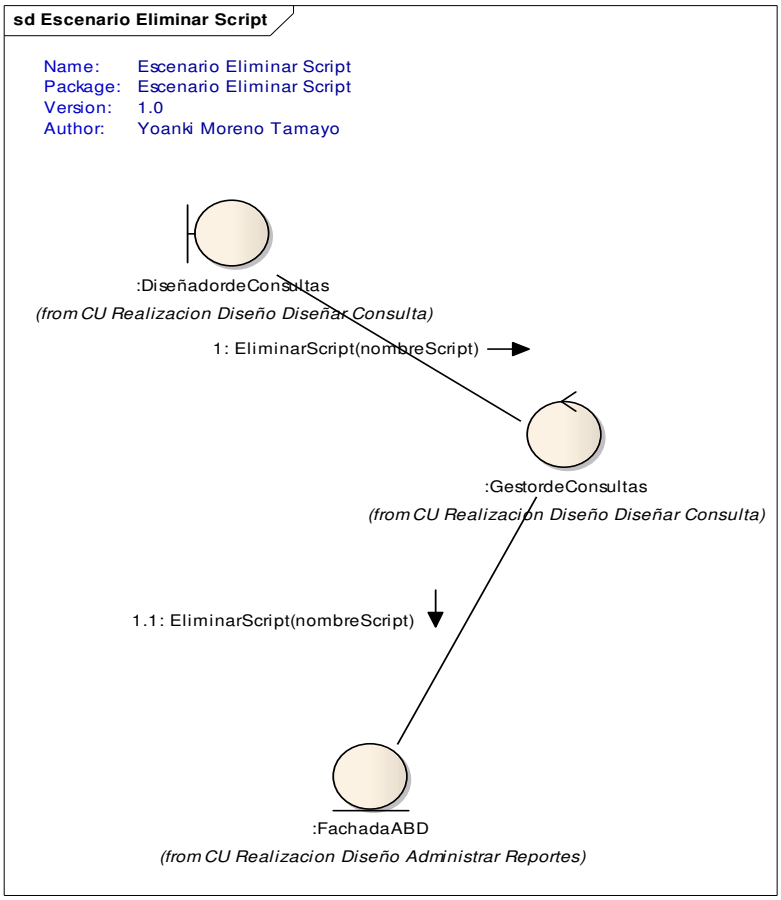
2.2.2 DI del CU: Administrar Reportes



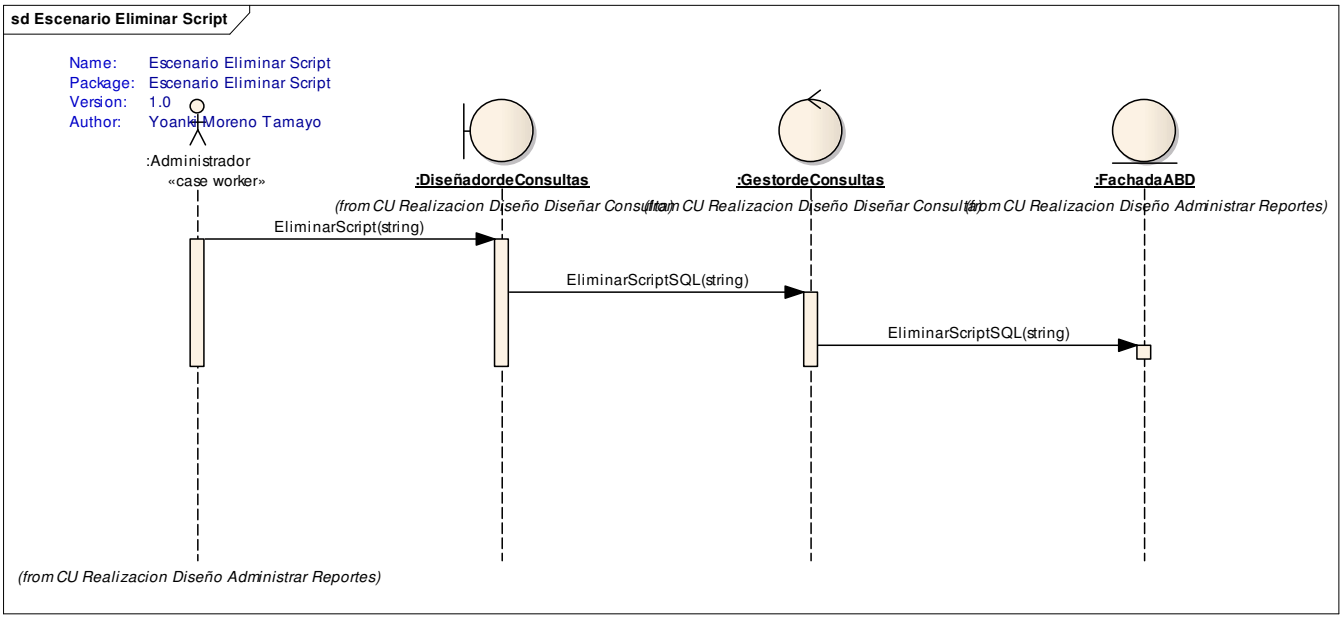
2.2.3 DI del CU: Administrar Reporte



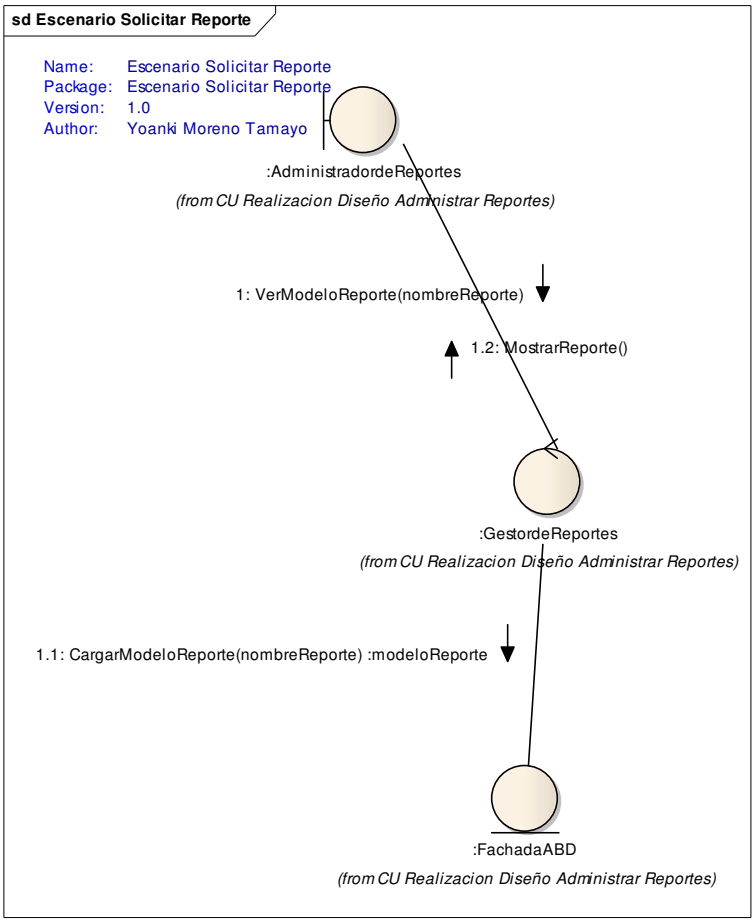
2.2.4 DI del CU: Administrar Reporte



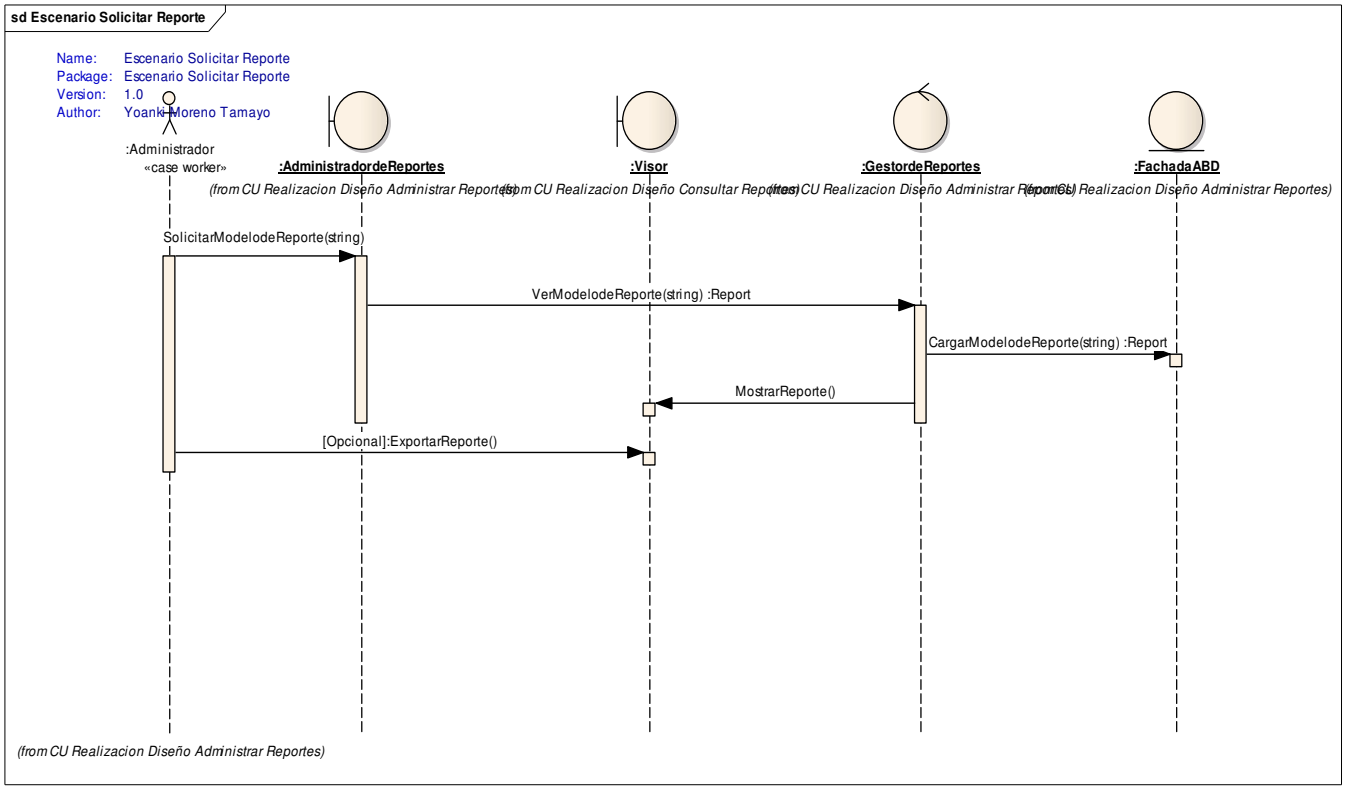
2.2.5 DI del CU: Administrar Reporte



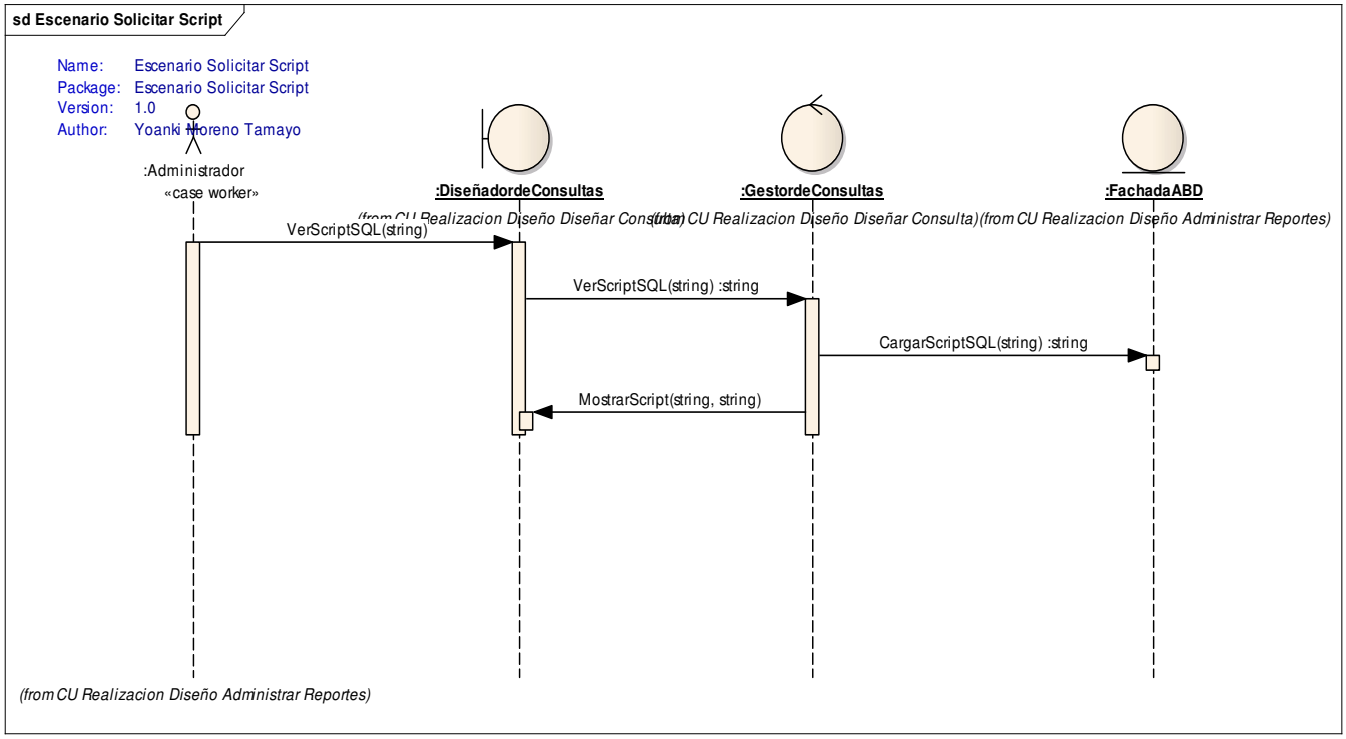
2.2.6 DI del CU: Administrar Reporte



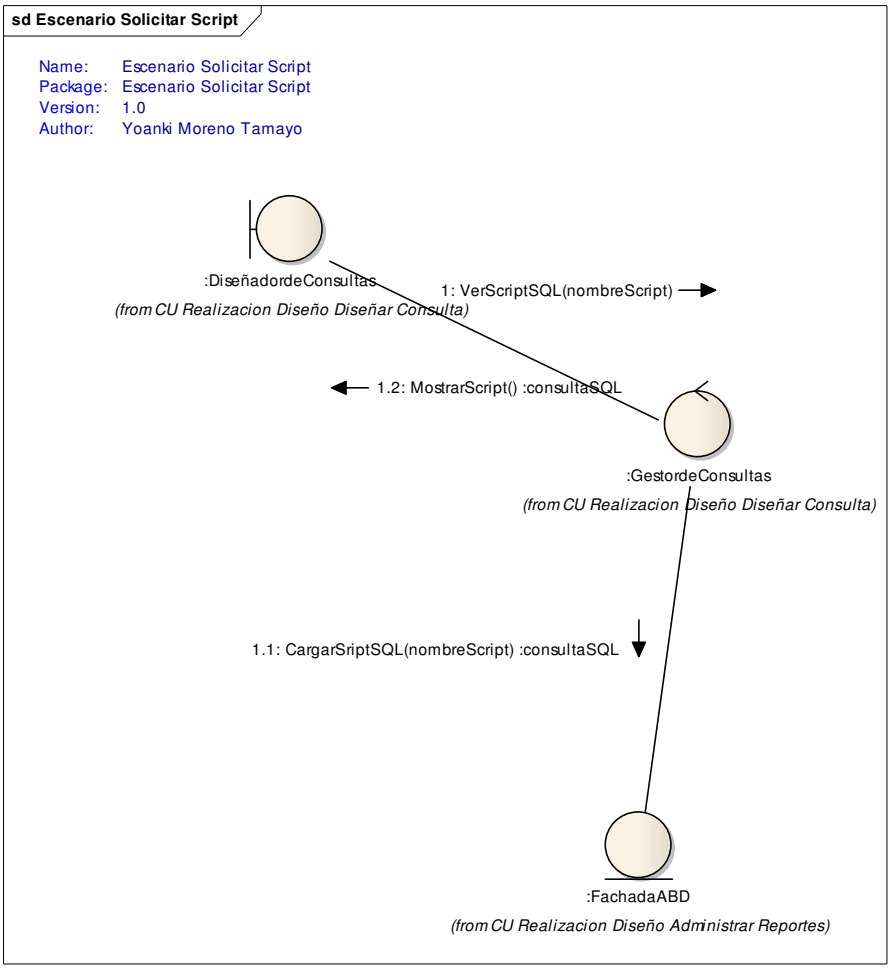
2.2.7 DI del CU: Administrar Reportes



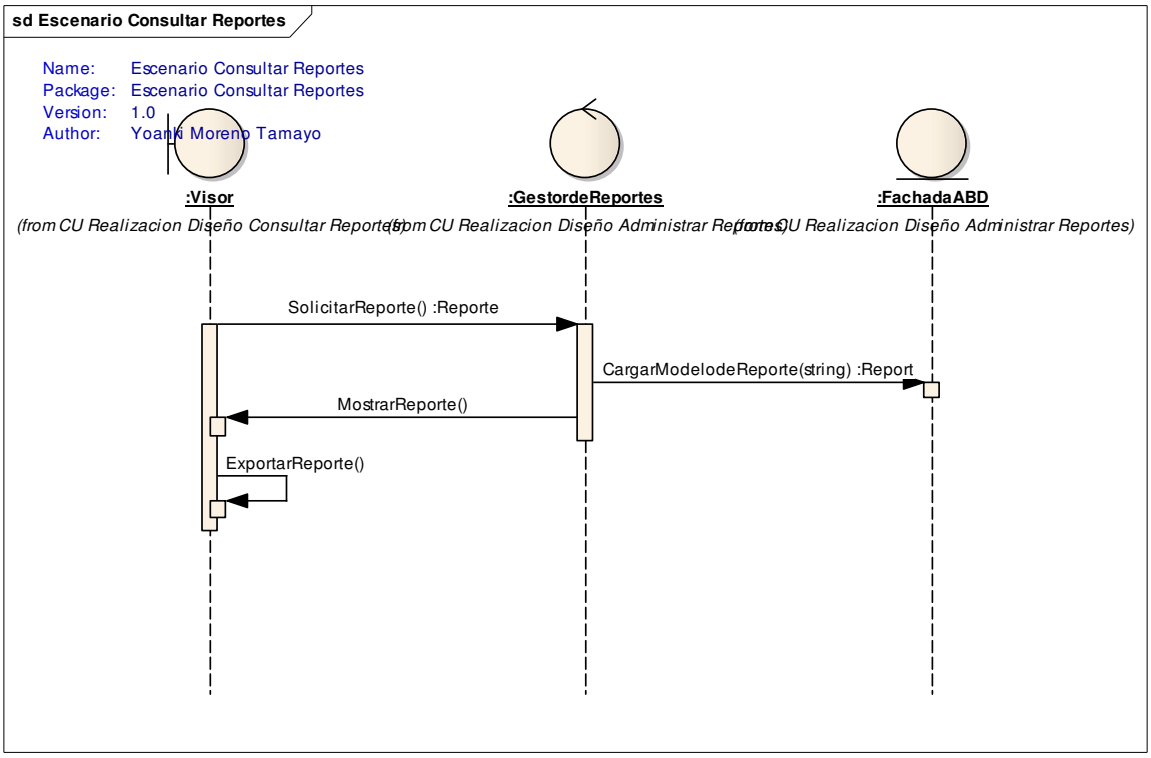
2.2.8 DI del CU: Administrar Reportes



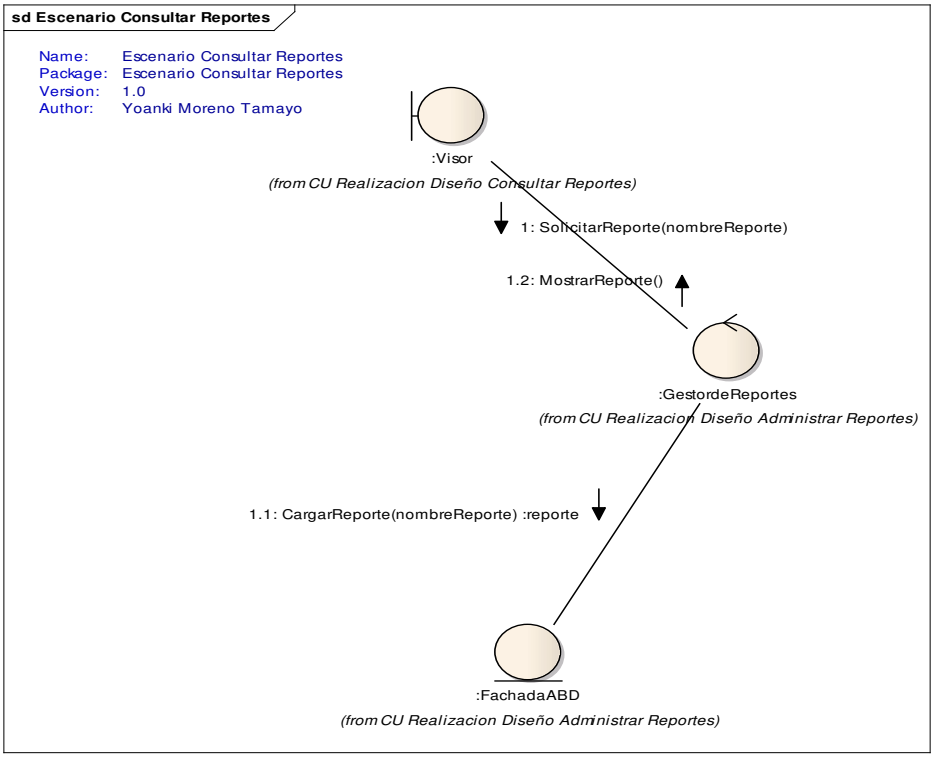
2.2.9 DI del CU: Administrar Reportes



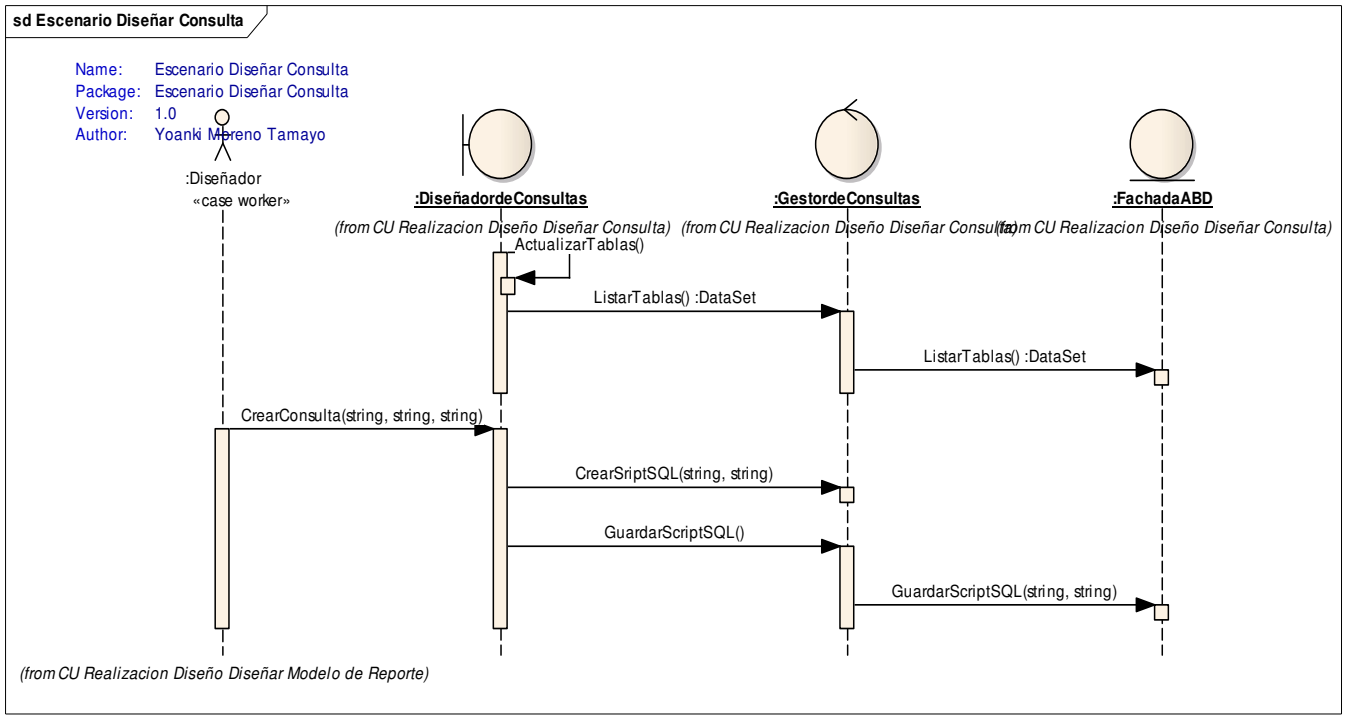
2.2.10 DI del CU: Administrar Reportes



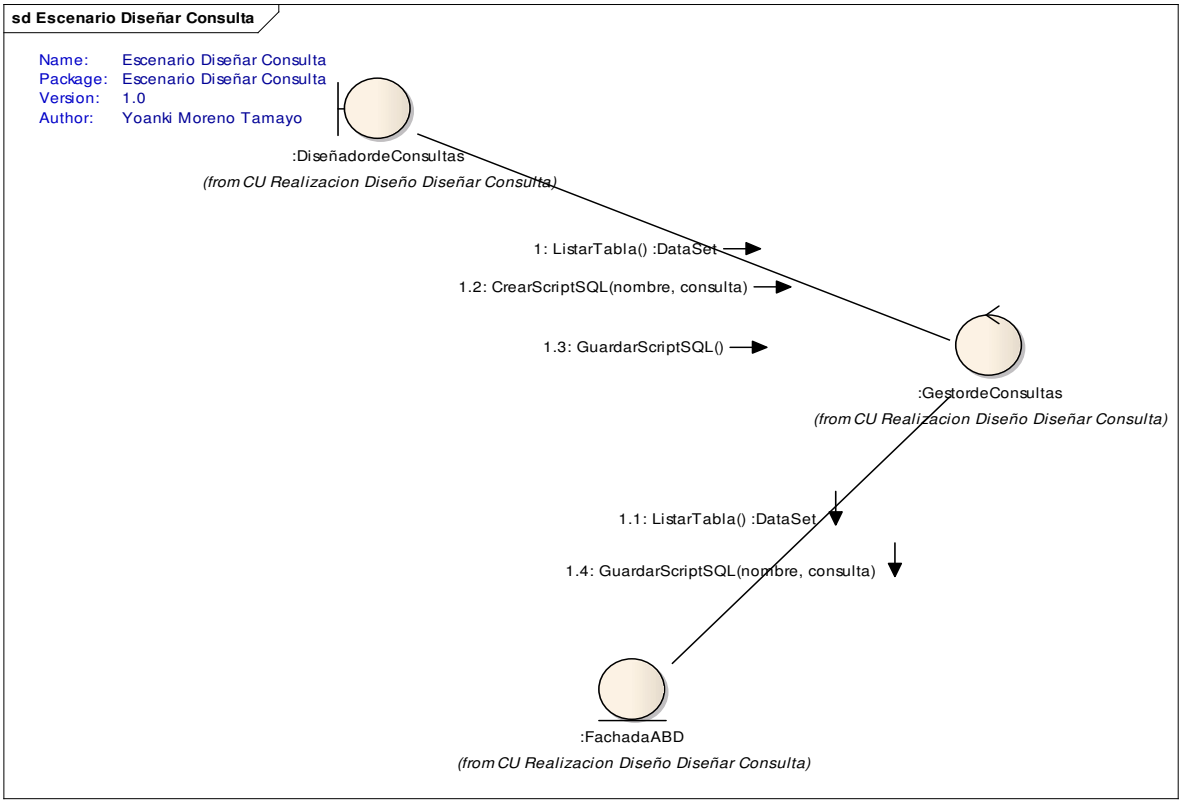
2.2.11 DI del CU: Consultar Reporte



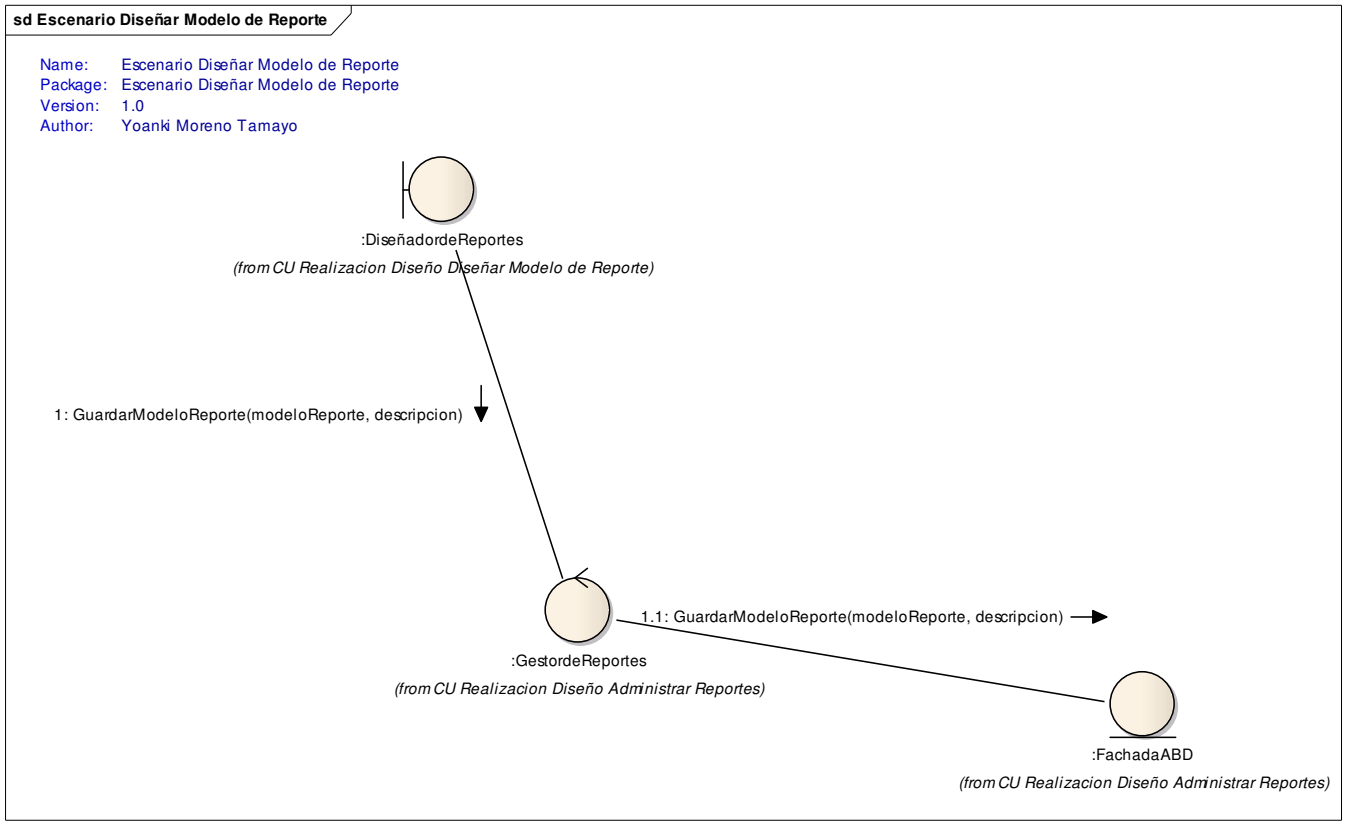
2.2.12 DI del CU: Consultar Reporte



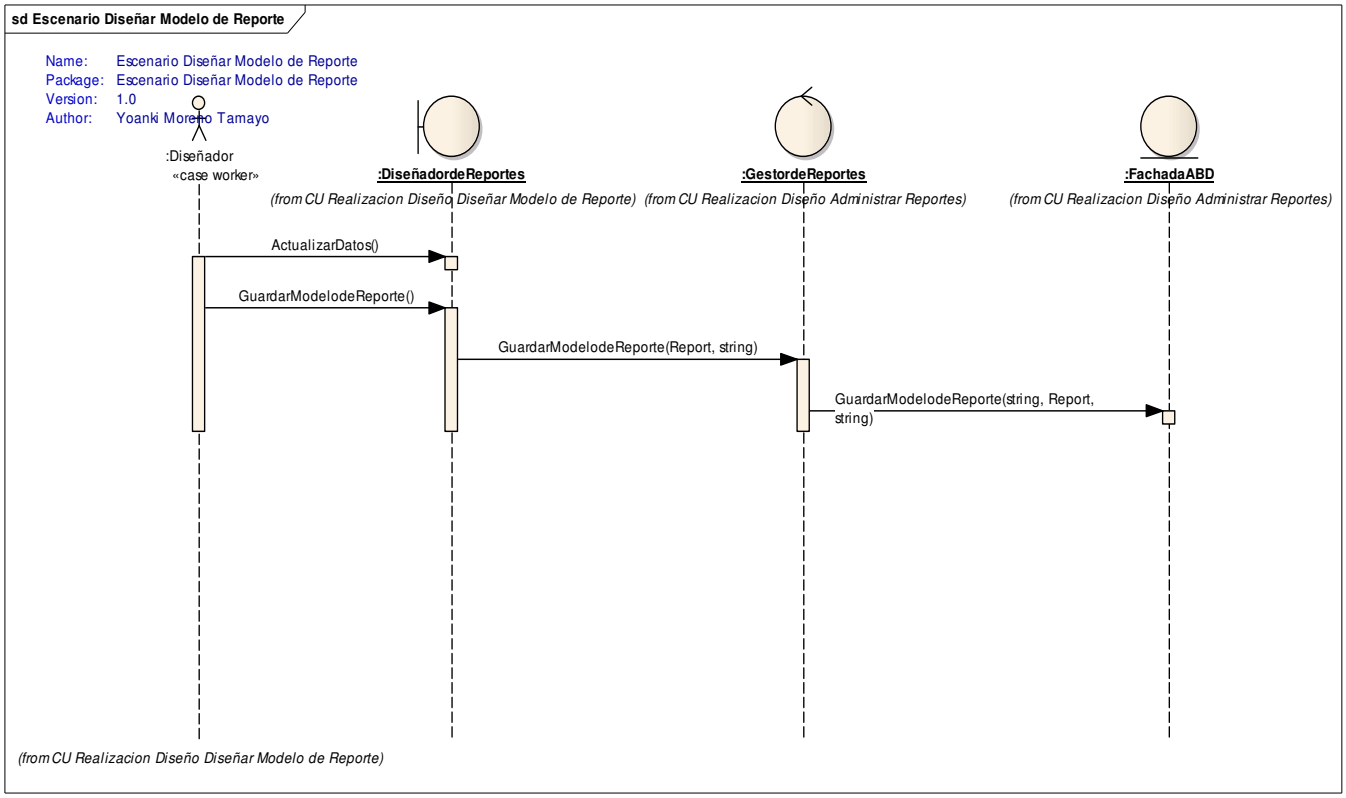
2.2.13 DI del CU: Diseñar Consulta



2.2.14 DI del CU: Diseñar Consultas



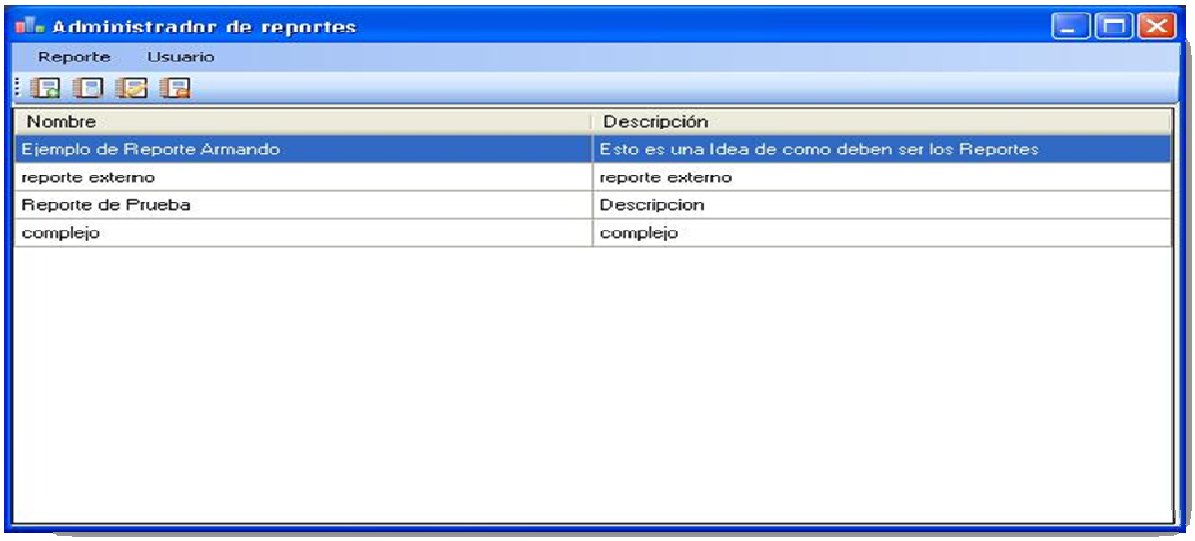
2.2.15 DI del CU: Diseñar Modelo de Reporte



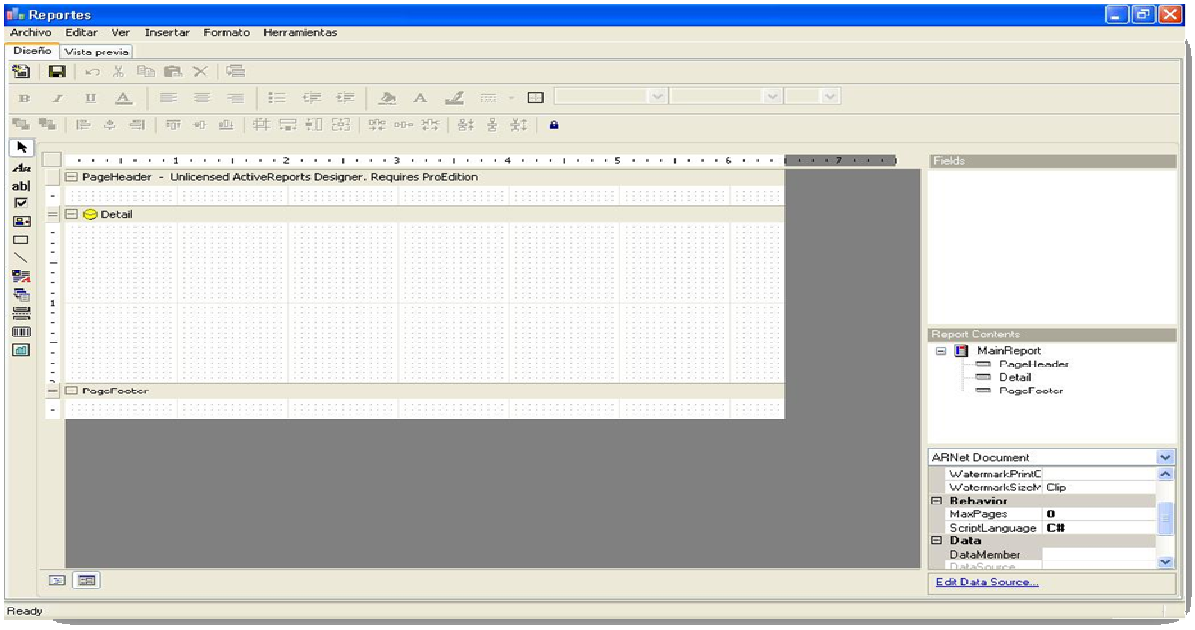
2.2.16 DI del CU: Diseñar Modelo de Reportes

Anexo 3

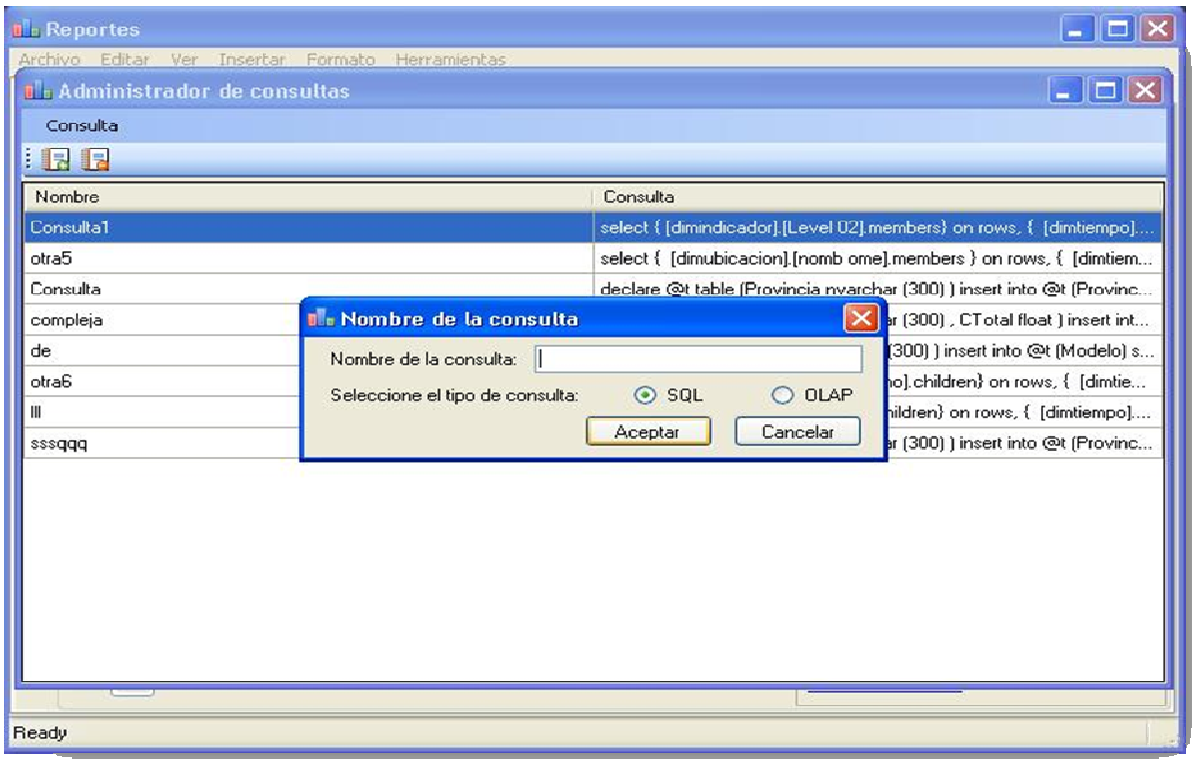
3.1 Interfaces principales del módulo



Interfaz de Administrador de Reportes



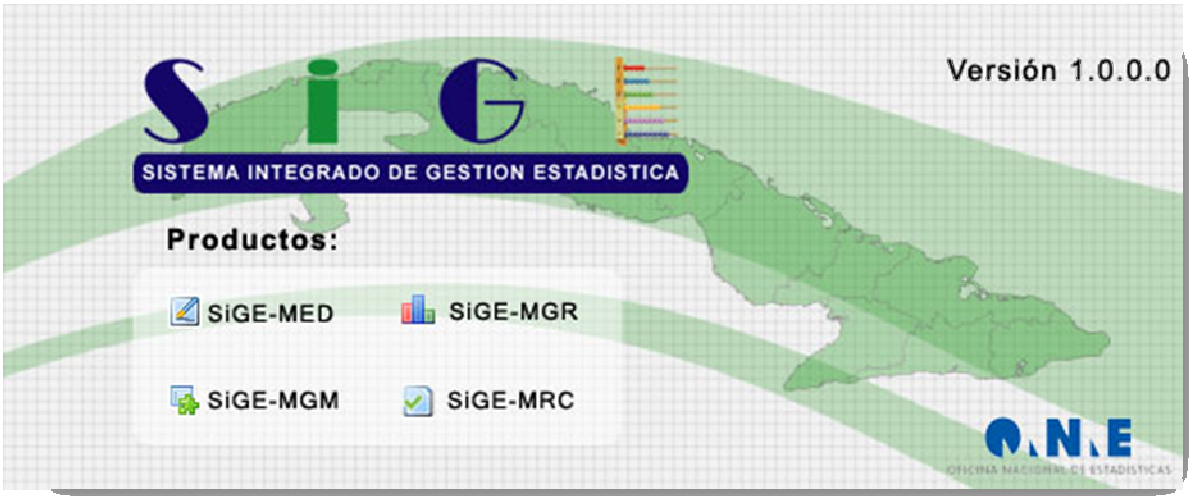
Interfaz del Reporte



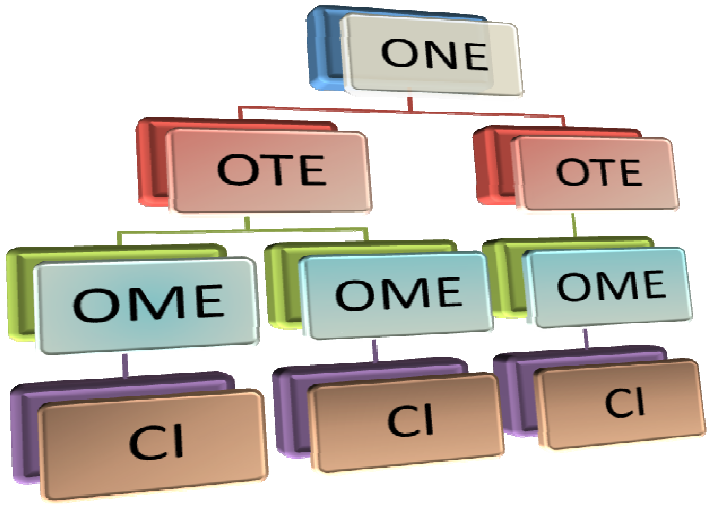
Interfaz del Diseñador de Consultas

Anexo 4
4.1 Imágenes Complementarias

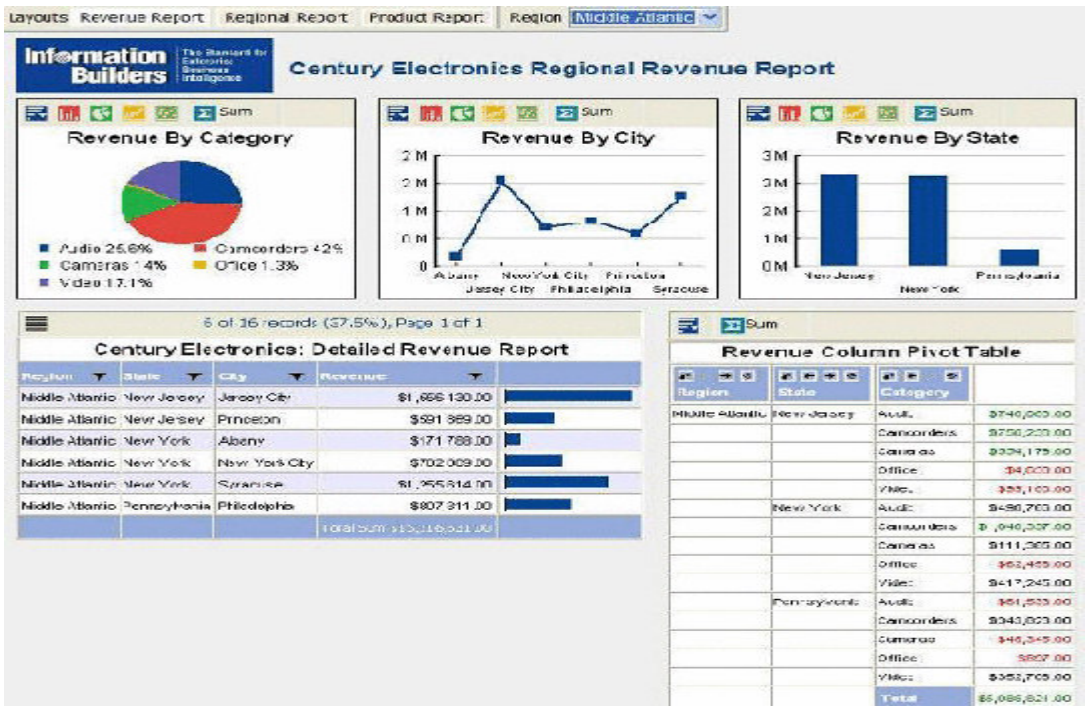
Presentación del MicroSet



Representación del SIGE



Infraestructura de Oficinas Estadísticas



Herramienta Active Reports

Cuba sobre Estadísticas