Universidad de las Ciencias Informáticas

Facultad #15

Análisis y Diseño del módulo Inspección a Locales de Detención y Centros Penitenciarios

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

Autores: Dayana Ruiz Martinez

Luis Morejón Cruz

Tutor: Ing. Osmany Becerra González

Ciudad de La Habana, del 2010
"Año 52 de la Revolución"

Lo fundamental es que seamos capaces de hacer cada día algo que perfeccione lo que hicimos el día anterior.

DECLARACIÓN DE AUTORÍA

Declaramos que Dayana Ruiz Martinez y Luis Morejón Cruz somos los únicos autores del trabajo de diploma Análisis y Diseño del módulo Inspección a Locales de Detención y Centros Penitenciarios y otorgamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo, para que hagan el uso que estimen pertinente con el mismo.

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

Dayana Ruiz Martinez

_____

Firma del autor

Luis Morejón Cruz

_____

Firma del autor

Ing. Osmany Becerra González

_____

Firma del tutor

Agradecimientos

Agradecer Especialmente:

A Osmany Becerra y a todos los profesores y miembros del proyecto SGF que de una forma u otra nos ayudaron y apoyaron para realizar el presente trabajo de diploma.

La Universidad por ser nuestra casa durante estos cinco años de múltiples vivencias.

Luis

A mi madre por ser lo más grande que tengo en mi vida.

A mi padre por ser mi amigo, mi hermano.

A mi tía y mis primos hermano por estar siempre presentes.

A mi novia Mercy por estar presente cada día de mi vida.

A todas las personas, familiares o no, que hayan hecho posible que me encuentre a esta altura de la vida.

A mi grandes amigos Emilio, Eric, Hansel y todos los que han estado a mi lado durante estos años de estudio.

Dayana

A mis padres por ser el motor impulsor de mi vida, por el amor y el cariño que me han profesado siempre, porque no existe nada que me regocije más que su compañía y por el apoyo incondicional que me brindan diariamente. A mi mamita particularmente por su delicadeza y cariño, y a mi papito por ser todo el cimiento de la familia y por sus innumerables consejos de constancia y VOLUNTAD, que me han dado fuerza para seguir adelante en mis momentos más vulnerables.

A Dali y a Yeni por estar siempre cuando más las necesito, por ser las hermanitas que no tuve, por su comprensión y compañía, por haber compartido conmigo los mejores y peores momentos de mi vida.

A Noel porque me ha llenado este año de alegrías y amor, por su serenidad y paciencia ante mis ataques de estrés, y porque me ha hecho sentir la mujer más feliz del mundo.

A mis amigos Lianni, Yurieski, Yunet, Imilsy, Gustavo, Maylin, Pavel, Street, Isabel y Yinet por brindarme su amistad desinteresada a lo largo de todos estos años.

A Oigres porque sin él no habría llegado tan lejos.

Especialmente A:

Todas las personas que de una forma u otra me han ayudado durante el transcurso de la carrera.

Mis padres por haberme ayudado tanto en mi vida, por haber sido tan dedicados y entregados incondicionalmente. Los llevaré en mi corazón toda mi vida.

Mis familiares, novia y amigos.

Luis Morejón

Mis padres que siempre han sido mi inspiración y toda mi fuerza, el ejemplo a seguir en mi vida…

Mi hermano Marlon por ser el mejor regalo que me han dado mis padres, por ser la luz de mis días y noches.

A Oigres porque es la mejor persona que he conocido, porque me ha enseñado a levantarme y siempre está para sostenerme cuando he flaqueado por algún motivo.

Dayana Ruiz

Resumen

La Fiscalía General de la República de Cuba, como parte del profundo y novedoso desarrollo tecnológico que se está produciendo en el país, se encuentra inmersa en la informatización de sus procesos. Para ello mantiene con la Universidad de las Ciencias Informáticas una relación contractual a través del proyecto Sistema de Gestión Fiscal, que desarrolla un sistema de gestión de información que pretende aumentar la eficacia en el trabajo de los fiscales, apresurar el proceso de toma de decisiones, y elevar los niveles de conformidad por parte de los ciudadanos cubanos. El presente trabajo de diploma tiene como propósito diseñar el módulo Inspección a Locales de Detención y Centros Penitenciarios, que se encarga de la planificación de la ejecución de las inspecciones a establecimientos penitenciarios y locales de detención, la confección de los principales documentos que se derivan de las inspecciones realizadas y el registro de la información estadística.

Para darle cumplimiento a las expectativas propuestas se realiza un estudio profundo de los procesos relacionados con las inspecciones que se llevan a cabo en el departamento CLEP (Control de la Legalidad en Establecimientos Penitenciarios) de la Fiscalía General de la República, a partir de los cuales se identifican las funcionalidades que el sistema propuesto debe tener. Una vez descritas detalladamente cada una de estas funcionalidades, se realizó el diseño del sistema y la evaluación de la calidad de los resultados obtenidos. Todas estas actividades se llevan a cabo para facilitar el trabajo de los implementadores, poniendo a su disposición una guía para que el producto sea confiable y eficiente. Con la implementación de esta propuesta se espera que los clientes tengan a su disposición un sistema alejado de fallas y vulnerabilidades, que cumpla con sus necesidades y les permita el manejo de la información utilizada para su trabajo.

PALABRAS CLAVE

Sistema de Gestión de Información, funcionalidades, diseño, patrones.

Tabla de Contenidos

Tabla de contenidos

Introducción

El desarrollo constante de las tecnologías de la información y las comunicaciones (TIC) ha provocado sistemáticos cambios en el orden social y económico. Los niveles alcanzados en el procesamiento, almacenamiento y comunicación de la información, hacen que hoy en día se encuentre disponible en tiempo real. Los usuarios con acceso a estas tecnologías aumentan sus niveles de productividad, al disponer de las facilidades de cómputo, transmisión, gestión y análisis de información con que trabajan. De ahí que sea necesario para las sociedades actuales el acceso a las nuevas tecnologías existentes en este campo y el desarrollo de sistemas que informaticen sus procesos elementales.

En los lineamientos del V Congreso del Partido Comunista de Cuba se planteó el desarrollo de las TIC encaminado a lograr la informatización de nuestra sociedad, con el objetivo de facilitar la toma de decisiones en la gestión de dirección y aumentar la eficacia y la productividad en la producción de bienes y en los servicios. La Fiscalía General de la República (FGR), Órgano del Estado al cual corresponde como objetivos fundamentales el control y la preservación de la legalidad en Cuba, cuenta con un sistema de gestión de información que no satisface las necesidades funcionales de la misma, por lo que se encuentra inmersa en la informatización de sus procesos, a través de una relación contractual sostenida con la Universidad de las Ciencias Informáticas (UCI), mediante el proyecto Sistema de Gestión Fiscal (SGF).

El proyecto SGF ha asumido, como política para la informatización del producto, dividir su desarrollo según los procesos fundamentales que se ejercen en cada departamento de la fiscalía. Como fruto de esta división surgen ocho subsistemas de desarrollo y formando parte de estos se encuentra el Control de la Legalidad en Establecimientos Penitenciarios (CLEP). Este tiene como propósito fundamental la gestión de la información referente al cumplimiento de la legalidad y derechos de ciudadanos detenidos o sancionados en cualquier centro de reclusión, internamiento o de detención. El subsistema CLEP está dividido en 3 módulos, dentro de estos se encuentra insertado Inspección a Locales de Detención (LD) y Centros Penitenciarios (CP) que se encarga, entre otros aspectos, de la planificación de la ejecución de las inspecciones a CP y LD, la confección de los principales documentos que se derivan de éstas y el registro y control de la información estadística.

Introducción

El fiscal del departamento CLEP para comprobar el cumplimiento de la legalidad y garantizar el respeto a los derechos individuales de los internos y detenidos realiza de forma directa inspecciones a los establecimientos penitenciarios y locales de detención. En el supuesto caso de detectar irregularidades durante una inspección emite una resolución encaminada a restablecer la legalidad quebrantada, en caso contrario formula un Acta o algún Documento de otra modalidad de reacción fiscal.

Para este y otros procesos derivados de la inspección, los fiscales autorizados deben registrar y consultar un considerable cúmulo de informaciones legales. Actualmente toda la información generada en las fiscalías se gestiona de forma manual y queda almacenada en formato duro como un rollo. Esto trae consigo gastos en material de oficina y que los datos almacenados aumenten el riesgo de contener errores, extraviarse y estar duplicados. Los trabajadores de dichas instituciones presentan problemas para consultar la información que necesitan, y en ocasiones pueden incurrir en violaciones por no dar respuesta a un determinado caso en el tiempo establecido. Es por ello la necesidad que existe de dotar a las fiscalías de una aplicación que informatice sus procesos fundamentales.

En el mundo, el desarrollo de software es un proceso que sigue enfrentándose a importantes retos. Una gran cantidad de proyectos no alcanzan a cumplir las necesidades del cliente porque no satisfacen las expectativas que motivaron su origen y finalmente corren el riesgo de ser rechazados por los usuarios. Para el correcto desarrollo del sistema y el módulo Inspección a LD y CP en particular, se hace necesario hacer un análisis previo donde se identifiquen las necesidades del cliente y se traduzcan al lenguaje de los desarrolladores, debido a que estos últimos no tienen absoluta claridad de lo que el cliente quiere y cómo lo quiere.

La situación anterior da lugar a que el ***Problema Científico*** a solucionar sea: ¿Cómo transformar las necesidades del cliente en un lenguaje comprensible al equipo de desarrollo, que facilite la implementación del Módulo Inspección a Locales de Detención y Centros Penitenciarios del Proyecto Sistema de Gestión Fiscal?

Donde se identificó como ***Objeto de Estudio:*** El Proceso de Desarrollo de Software y como ***Campo de Acción:*** El Análisis y Diseño del Módulo Inspección a Locales de Detención y Centros Penitenciarios.

Para darle solución a este problema se plantea como ***Objetivo General:*** Realizar el Análisis y el Diseño del módulo Inspección a Locales de Detención y Centros Penitenciarios del proyecto Sistema de Gestión Fiscal.

Inicialmente se parte de la siguiente ***Idea a defender***: Si se realiza el Análisis y el Diseño de Software del Módulo Inspección a Locales de Detención y Centros Penitenciarios, que posibiliten la transformación de las necesidades del cliente en un lenguaje comprensible al equipo de desarrollo, entonces se facilitará la implementación de este módulo.

Para dar cumplimiento al objetivo propuesto se trazaron las siguientes ***tareas de la investigación:***

- ✓ Realización del Marco Teórico.
- ✓ Identificación de los procesos fundamentales que se desarrollan en el módulo Locales de Detención y Centros Penitenciarios del Sistema de Gestión Fiscal.
- ✓ Realización del modelado del sistema definiendo correctamente los requisitos del software.
- ✓ Realización del modelado del diseño.
- ✓ Validación de los resultados obtenidos.

Para dar cumplimiento a las tareas propuestas anteriormente se emplean diferentes ***Métodos Científicos*** de la investigación. Como método Teórico se utiliza el Analítico-Sintético, pues uno de los primeros pasos para llevar a cabo la presente tesis es buscar y analizar documentos e información sobre el desarrollo de los procesos de inspección a los Establecimientos Penitenciarios por los fiscales y extraer los elementos más importantes relacionados con el objeto de estudio. Otro método teórico que se aborda en la investigación es el Histórico-Lógico para analizar a nivel internacional y nacional el empleo de sistemas informáticos de gestión jurídica, así como investigaciones realizadas anteriormente sobre el tema. El método de Modelación se usa para la creación de abstracciones que explican la realidad mediante todos los modelos y diagramas a presentar. El Inductivo-Deductivo se utilizó para, a través de un razonamiento, arribar a un grupo de conocimientos particulares y generales. Dentro de los métodos Empíricos se usan las Entrevistas que se les realiza a los especialistas del departamento CLEP puestos a disposición del equipo de desarrollo, para de esta forma recoger toda la información referente al funcionamiento del proceso de inspección.

La presente investigación está estructurada por 4 capítulos cuya descripción será expuesta brevemente a continuación.

Capítulo 1: **"Fundamentación Teórica".** Se describen los conceptos fundamentales asociados al dominio del problema, se muestra el estado del arte de los sistemas de Gestión Jurídica tanto a nivel internacional

como nacional donde se presenta un análisis crítico de las soluciones ya existentes. Se caracterizan las tecnologías, metodologías y herramientas y se realiza la justificación de las seleccionadas en las que se apoya la solución al problema. Se abordan temas clave relacionados con la captura de requisitos y diseño del sistema como son: técnicas, métricas y principales patrones.

Capítulo 2: **"Análisis del sistema".** En este capítulo se realiza una caracterización del negocio, describiendo las principales reglas del mismo, así como los participantes por parte de la fiscalía y de los establecimientos penitenciarios o locales de detención. Se especificó detalladamente los diferentes procesos del negocio relacionados con las inspecciones.

En el presente capítulo se presentan los requisitos que debe cumplir la aplicación, se realiza una descripción de los casos de uso del sistema y se exponen los artefactos resultantes del análisis de los procesos fundamentales del módulo propuesto.

Capítulo 3: **"Diseño del sistema".** En este capítulo se realiza el modelado del diseño, enfocando el sistema desde una perspectiva interna en función de clases y paquetes que permiten darle cumplimiento a los requisitos establecidos en el capítulo 2. Se exponen los artefactos que integran el diseño y se hace referencia a la arquitectura y patrones utilizados para la construcción de la propuesta.

Capítulo 4: **"Validación de los resultados".** En este capítulo se realizará la validación de los resultados obtenidos a través de métricas que garanticen la calidad de la solución propuesta.

1. Capítulo 1. Fundamentación teórica

1.1. Introducción

En este capítulo se definen los conceptos principales que fueron necesarios estudiar para dar solución al problema planteado. Se muestra un estudio del estado del arte sobre diferentes aspectos como la Informática Jurídica y los principales sistemas de gestión de esta disciplina. Además de caracterizar las posibles tecnologías, metodologías y herramientas, se realiza la selección y justificación de las más idóneas a emplear para el desarrollo del trabajo de investigación. Se presenta una breve explicación de los patrones (casos de uso y diseño) y métricas que serán de gran utilidad para el desarrollo del trabajo de diploma.

1.2. La informática y la justicia

La informática, como uno de los adelantos científico-técnicos más significativo en los últimos tiempos, está íntimamente relacionada con casi todas las áreas del conocimiento humano, dentro de las cuales las ciencias jurídicas no están exentas, esto da lugar a que se establezcan relaciones entre estas y las ciencias informáticas, desarrollándose la Informática Jurídica, disciplina netamente instrumental donde se coloca la informática al servicio del Derecho.

En términos conceptuales la Informática Jurídica es la aplicación de los sistemas informáticos a las distintas esferas del Derecho, que debe alcanzar el estudio, análisis y aprovechamiento de los recursos que ofrece la informática al quehacer jurídico.(Fernández, 2002)

El término "Jurimetría" se utilizó por primera vez en 1949, en el artículo titulado “El próximo paso” escrito por el juez norteamericano Lee Loeverger, quien apuntaba a la inserción de la información jurídica en las aplicaciones “cibernéticas” y tenía como objeto de estudio la racionalización del Derecho a través de la aplicación de la automatización.

Luego de lo planteado en el “El próximo paso” desde finales de la década del 50 se continuó con la investigación sobre sistemas de recuperación de información legal. Es así como en 1960 surge la primera base de datos en materia jurídica y años más tarde se crean los primeros sistemas comerciales destinados a este ámbito.

Poco a poco esta ciencia fue evolucionando dando lugar a nuevas clasificaciones y definiciones de la misma, como son la informática jurídica documental, decisional y de gestión, brevemente explicadas en el siguiente concepto.

Se define a la informática jurídica como la tecnología aplicada a la sistematización que estudia el tratamiento automatizado de: las fuentes del conocimiento jurídico a través de los sistemas de documentación legislativa, jurisprudencial y doctrinal que sería lo mismo que la informática jurídica documental; las fuentes de producción jurídica, a través de la elaboración informática de los factores lógico-formales que concurren en el proceso legislativo y en la decisión judicial, es decir, informática jurídica decisional; y los procesos de organización de la infraestructura o medios instrumentales con los que se gestiona el Derecho, que no es más que la informática jurídica de gestión, la cual se utiliza para automatizar sistemas de gestiones jurídicas que facilitan el trabajo de las diferentes entidades que se encargan de consultar y/o prestar servicios jurídicos.(Silva, 2008)

1.2.1. Informática Jurídica de gestión

Al hablar acerca de Informática Jurídica de gestión, se hace referencia a la automatización de procedimientos, técnicas y medios utilizados en los procesos administrativos de las oficinas jurídicas, basado en los métodos informáticos y de comunicación, así como en el conjunto de elementos tecnológicos que permiten gestionar la información. Entre las principales ventajas de los sistemas de gestión jurídica se pueden citar que, posibilitan confeccionar escritos de forma automatizada, llevar un control de asuntos del personal jurídico, proteger la información con niveles de usuario y crear reportes de la información estadística.

Para la realización del presente trabajo se realizó una investigación de los siguientes sistemas de gestión jurídica existente en el mercado:

Infolex es un Software de Gestión Integral para Despachos Jurídicos, fue desarrollado en España por Jurisoft, empresa dedicada a la Informática Jurídica. El mismo permite realizar la gestión de expedientes, para ello ofrece múltiples sistemas de localización de expedientes que facilita al usuario búsquedas ágiles y sencillas. Está dotado de una gran flexibilidad, debido a que se adapta a las necesidades de cada profesional y da cobertura a todas sus especialidades: Despachos Judiciales o Extrajudiciales, Letrados

de Entidades Bancarias, Compañías de Seguros o Despachos Mercantilistas y otras. Cuenta además de un amplio Registro de poderes de clientes para llevar su control de forma eficaz y segura.
Infolex comprende la emisión de un gran número de listados que permiten extraer la información que le interese al cliente en cada caso: Listados e Informes sobre Clientes, Expedientes y seguimiento Procesal y Extraprocesal. Incluye además un exhaustivo control de la producción de todos y de cada uno de los profesionales del Despacho.

LurisExplorer es un software desarrollado en Argentina que posibilita organizar y consultar rápidamente la información manejada por el usuario. Permite gestionar los trabajos usuales de todo el Estudio Jurídico como administrar expedientes, redactar escritos, intercambiar documentación, llevar un orden de las causas y ordenar sus cosas particulares.
Su diseño facilita satisfacer y administrar las más complejas exigencias del trabajo de un abogado, tanto en su desempeño individual como con tareas que involucren altos volúmenes, se adapta a las necesidades de los profesionales. LurisExplorer no sólo puede realizar modelos de Escritos, sino también modelos de expedientes con la estructura necesaria y el contenido predeterminado indicado.

Lex-Doctor es un conjunto de sistemas jurídicos desarrollado en Argentina por Sistemas Jurídicos SRL, empresa que desarrolla y comercializa sistemas para ser aplicados al ámbito jurídico. Permiten realizar la gestión de expedientes, gestión documental, reportes y estadísticas. Su tecnología de administración de datos, posibilita manejar grandes volúmenes de información, y organizar grupos de trabajo operando en redes de gran cantidad de terminales.
Brinda poderosas herramientas de búsqueda, clasificación, y análisis de datos, que permiten la creación de reportes que involucran cualquier dato cargado en cada expediente, y reportes que involucren solamente expedientes que poseen determinadas características.

Gedex es un software jurídico con amplia experiencia en el sector. Uno de los más utilizados en España y Latinoamérica; sus inicios se remontan al año 1996 y actualmente cuenta con un gran número de licencias vendidas a diferentes despachos jurídicos. Permite realizar el seguimiento completo de los expedientes jurídicos de un despacho, bufete o departamento.
Este producto es aplicable a disímiles modos de gestión, gracias a su adaptabilidad, integrándose en su despacho sin cambiar el tratamiento de la información al que se encuentre habituado. Permite localizar y vincular rápidamente expedientes y contactos. Gedex ofrece un avanzado sistema de contraseñas, con el

que puede limitar el acceso a la información, ocultar expedientes y contactos a ciertos pasantes o empleados.

Softlex es un sistema nacido en Cuba que brinda servicios de tratamiento documental, almacenamiento y consulta de legislaciones agrupadas en listas organizadas por: emisión de la gaceta, institución que genera la norma, número de la normativa, tipo de normativa y página de la gaceta. Brinda además la posibilidad de contar con un índice referativo que recoge las referencias de la actividad jurídica en un período de tiempo determinado.

1.2.2. Justificación de la necesidad del Sistema de Gestión Fiscal

Con la inserción del sistema jurídico español en Cuba, la isla se incorpora al Sistema Jurídico Romano Francés, régimen por el cual se orientan también España, Argentina y otros países de América. Al triunfo revolucionario cubano se decidió permanecer bajo este sistema pero ajustado a principios de carácter socialista. A pesar de que estos tres países se guían por los mecanismos provenientes del sistema romano, sus realidades judiciales difieren parcialmente. Cuba mantiene sus aspectos funcionales pero a disposición de una sociedad diferente, es por ello que los sistemas de gestión jurídica, desarrollados en otros países, son inadaptables al sistema socialista cubano.

La situación económica por la que atraviesa el país dificulta la adquisición de estos sistemas, debido a sus elevados precios para la adquisición y el mantenimiento de sus licencias de uso. Esto se debe a que están desarrollados con tecnologías propietarias, otra de las razones por la que es casi imposible su utilización.

El sistema de gestión que se utiliza en las fiscalías cubanas, es una aplicación desktop que brinda algunos reportes de gran utilidad para los fiscales, pero no abarca temas como la toma de decisiones y la gestión de la información de algunas áreas de las fiscalías, como es el caso del Departamento CLEP, o sea, no satisface totalmente las necesidades delos fiscales.

Tomando en cuenta los elementos determinados en los sistemas de gestión jurídica anteriormente mencionados, se hace necesario el desarrollo del Sistema de Gestión Fiscal, con el que se pretende responder objetivamente a los problemas existentes en las fiscalías cubanas y reducir el tiempo de ejecución de las actividades que se llevan a cabo en dichas instituciones.

1.3. Ingeniería de requisitos

La ingeniería de requisitos es el uso sistemático de procedimientos, técnicas, lenguajes y herramientas para obtener con un coste reducido el análisis, documentación, evolución continua de las necesidades del usuario y la especificación del comportamiento externo de un sistema que satisfaga esas necesidades. (S.Pressman, 2005)

El desarrollo de software vive una continua crisis a escala mundial, esto se debe a que en la Industria de Software hay tendencia al crecimiento del volumen y complejidad de los productos, y en muchas ocasiones por falta de un personal calificado y comprometido los sistemas se entregan al cliente fuera del tiempo convenido, el cual queda insatisfecho con el mismo, rechazando el producto en su totalidad porque no satisface sus necesidades. Además, se alcanzan altos costos de mantenimiento del sistema por errores cometidos durante su desarrollo.

Una de las razones más importantes de esta problemática recae sobre la elicitación de los requisitos, actividad que debe estar considerada como una de las más críticas dentro del proceso de desarrollo, porque no sólo es necesario identificar y especificar correctamente esos requisitos, sino que además es preciso realizar un seguimiento de su aplicación durante todo el ciclo de vida del proyecto. La ineficiente captura de requisitos conlleva a la generación de errores costosos y difíciles de resolver que afectan la concepción de la aplicación en su conjunto, implicando más gasto de recursos en la fase de desarrollo.

1.3.1. Actividades de la Ingeniería de requisitos

Elicitación: Captura o identificación de requisitos

El proceso identificación de requisitos tiene lugar una vez que se ha hecho un estudio de viabilidad del sistema. Este proceso tiene como principal objetivo guiar el desarrollo de software hacia el sistema correcto, definiendo objetivos generales concretos de manera tal que tanto el negocio como sus actores se beneficien.

Durante el proceso de obtención de los requisitos desempeña un papel esencial el cliente, que se convierte en un miembro más del equipo de proyecto por lo que el resultado de la captura de requisitos debe ser escrito en su lenguaje.

Análisis y negociación de requisitos

En esta etapa los requisitos se agrupan por categorías y se organizan en subconjuntos, se estudia cada requisito en relación con el resto, se examinan los requisitos en su consistencia, completitud, y ambigüedad, y se clasifican y ordenan en base a las necesidades y prioridades de los clientes.
Los riesgos asociados con cada requisito serán identificados y analizados. Se efectuarán estimaciones del esfuerzo de desarrollo que se utilizan para valorar el impacto de cada requisito en el costo del proyecto y en el plazo de entrega. Utilizando un procedimiento iterativo, se irán eliminando, combinando y modificando requisitos, y así conseguir satisfacer los objetivos propuestos.

Especificación o documentación de requisitos

El objetivo en esta etapa es obtener una especificación de los requisitos que ya han sido analizados y negociados con los clientes, la cual puede desarrollarse en un documento escrito. Se propone que debe desarrollarse una plantilla estándar, para que los requisitos sean presentados de una forma más consistente y más comprensible.

Validación de requisitos

Permite demostrar que los requisitos definidos son los que realmente quiere el cliente, “examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencia, sin omisiones, que los errores detectados hayan sido corregidos y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso, el proyecto y el producto”.(S.Pressman, 2005)

Administración o gestión de los requisitos

Con esta actividad se pretende llevar un control sobre los cambios que pueden sufrir los requisitos debido a que no se hayan hecho las preguntas correctas a los usuarios, haya cambiado el problema que se estaba resolviendo, o simplemente cambiaron las expectativas de los clientes. Para gestionar los requisitos se llevan a cabo “un conjunto de actividades que ayudan al equipo de trabajo a identificar, controlar y seguir los requisitos y los cambios en cualquier momento”.(S.Pressman, 2005)

1.3.2. Técnicas para la captura de requisitos

La captura de requisitos es la actividad mediante la cual se extraen las necesidades que debe cumplir el sistema para satisfacer al cliente. El proceso de elicitación de requisitos puede ser muy complejo, y para evitar dicha complejidad se han desarrollado técnicas que permitan hacerlo de una forma más eficiente y precisa. A continuación se explican en qué consisten algunas de las técnicas empleadas para realizar dicha actividad:

- ✓ Entrevistas: resultan una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural.

  Básicamente, la estructura de la entrevista abarca tres pasos: identificación de los entrevistados, preparación de la entrevista, realización de la entrevista y documentación de los resultados (protocolo de la entrevista).(Escalona, et al., 2002)

  El analista debe tener la capacidad para elegir bien a los entrevistados y obtener toda la información posible en un período de tiempo siempre limitado.

- ✓ Tormenta de ideas. Esto es una técnica de reuniones en la que las personas que participan generan un conjunto de ideas de manera espontánea sin temor a críticas o al juicio de los otros participantes. En estas reuniones todo está permitido, hasta las ideas más irracionales e ilógicas que van surgiendo libremente en el transcurso de la reunión. El tamaño de este grupo generalmente oscila entre 4 y 10 personas, uno de los cuales es el moderador o jefe de sesión. Esta técnica cuenta con cuatro fases que son preparación, generación, consolidación y documentación.

- ✓ Juegos de Roles. En su forma más simple, consiste en que el desarrollador, el analista y cada uno de los miembros del equipo de desarrollo del software toman el lugar del interesado y ejecutan la actividad de trabajo que este desempeña. Ellos experimentan las inexactitudes y problemas ligados con el sistema que se está especificando. Se busca suministrarle al analista una perspectiva nueva del problema que le permita la obtención de los requisitos del sistema por construir.(Zapata, et al., 2007)

- ✓ Introspección. Esta técnica recomienda que sea el analista quien se ponga en el lugar del cliente y trate de imaginar la forma en que él desearía el sistema. Una vez obtenida una serie de suposiciones

acera de los requisitos funcionales, el ingeniero de requisitos le recomienda al cliente las funcionalidades que debería presentar la aplicación.

- ✓ Prototipado. Esta técnica es muy parecida a la anterior. Mediante la misma, el ingeniero de requisitos desarrolla la solución informática con una funcionalidad restringida para que sea el cliente quien la valide.

- ✓ Casos de uso o escenarios. Son descripciones que incluyen actores, eventos, operaciones y objetivos de esas operaciones, generalmente ligados con el funcionamiento de una solución informática. Los escenarios como técnica de obtención poseen dos limitaciones: exigen una alta participación del interesado en su elaboración y necesitan que él realice una concepción completa de la solución informática, que sólo sería posible al final del proceso de obtención de requisitos.(Zapata, et al., 2007)

- ✓ Arqueología de Documentos. Esta técnica trata de determinar posibles requisitos sobre la base de inspeccionar la documentación utilizada por la empresa; por ejemplo, manuales de procedimientos, reglamentos, boletas, facturas etc. Esta técnica sirve como complemento de las demás técnicas, a través de ella se consigue información que de otra manera seria muy difícil de conseguir.

1.3.3. Patrones de Casos de Uso

Los patrones son soluciones simples. Están formados por una pareja problema/solución que se fundamenta en la experiencia para problemas específicos y comunes, y que se ha demostrado que funcionan y pueden emplearse en diferentes contextos. Algunos de estos son:

- ✓ **Concordancia.** Este patrón toma una subsecuencia de acciones que estén en diferentes partes del flujo de casos de uso y la define por separado. (Overgaard, et al., 2004)
  *Concordancia: Re-uso.*El patrón Re-uso es un patrón de estructura que consiste en tres casos de uso. El primero, llamado “Sub-secuencia Común”, modela la secuencia de acciones que aparecen en múltiples casos de uso del modelo. Los otros dos casos de uso comparten de esta sub-secuencia común de acciones (dos es la menor cantidad que puede existir). La sub-secuencia tiene que estar en un fragmento, es decir, todo lo que requiere estar incluido tiene que estar en un único fragmento completo. Además, no se puede hacer referencia desde la sub-secuencia a donde esta es utilizada,

porque el caso de uso incluido tiene que ser independiente del caso de uso base. (Overgaard, et al., 2004)

Concordancia: Adición. En el caso de este patrón alternativo, la subsecuencia común de casos de uso, extiende los casos de uso compartiendo la subsecuencia de acciones. Los otros casos de uso modelan el flujo que será expandido con la subsecuencia. Este patrón es preferible usarlo cuando otros casos de uso se encuentran propiamente completos, o sea, que no requieren de una subsecuencia común de acciones para modelar los usos completos del sistema. (Overgaard, et al., 2004)

- ✓ **Extensión Concreta.** Este es un patrón de estructura que consiste en dos casos de uso y una relación de extensión. Es aplicable cuando un flujo puede extender el flujo de otro caso de uso, lo que significa que puede ocurrir el del caso de uso base con su caso de uso extendido o simplemente el caso de uso base.

- ✓ **Inclusión Concreta.** Es un patrón de estructura que consiste en dos casos de uso y una relación de inclusión entre el caso de uso base y el caso de uso incluido. El caso de uso base puede ser concreto o abstracto. Se utiliza este patrón cuando un flujo de datos puede ser incluido en el flujo de otro caso de uso y también puede ejecutarse por sí solo.

- ✓ **CRUD (Creating, Reading, Updating, Deleting).**Es un patrón de estructura que se basa en la fusión de casos de uso simples para formar una unidad conceptual.

  CRUD: Completo. Es un caso de uso llamado Información de CRUD o Administrar Información, que modela las diferentes operaciones que pueden realizarse en un pedazo de información de un cierto tipo, como crear, leer, actualizar, y eliminar. Este patrón debe usarse cuando todos los flujos contribuyen al mismo valor del negocio y estos son cortos y simples.

  CRUD: Parcial. Es un patrón alternativo, que modela una alternativa del caso de uso como un caso de uso separado. Se usa preferiblemente cuando dicha alternativa es más significativa o compleja que las otras.

- ✓ **Múltiples actores.** Captura la concordancia entre actores manteniendo roles separados.

Roles diferentes. Consiste en un caso de uso y por lo menos dos actores. Este patrón se usa cuando los dos actores juegan papeles diferentes hacia el caso de uso, es decir, ellos interactúan de forma diferentemente con el caso del uso.

Roles comunes. Este patrón se usa cuando dos actores jueguen el mismo rol sobre un caso de uso. El rol es representado por otro actor, heredado por los actores que lo comparten.

1.4. Diseño del sistema

El Diseño de Sistemas se ocupa de desarrollar las directrices propuestas durante el análisis en función de aquella configuración que tenga más posibilidades de satisfacer los objetivos planteados tanto desde el punto de vista funcional como del no funcional. (Deadalus, 2010)

El diseño es una representación significativa de ingeniería de algo que se va a construir. Se puede hacer el seguimiento basándose en los requisitos del cliente, y al mismo tiempo la calidad se puede evaluar y cotejar con el conjunto de criterios predefinidos para obtener un diseño “bueno”. En el contexto de la ingeniería del software, el diseño se centra en cuatro áreas importantes de interés: datos, arquitectura, interfaces y componentes. (S.Pressman, 2005)

El diseño del sistema se basa fundamentalmente en adquirir una comprensión profunda de los requisitos funcionales y no funcionales identificados durante el análisis. Su propósito es crear una entrada para la implementación del sistema, permitiendo descomponer esta actividad en partes más manejables. Se centra en objetivos específicos, como la reutilización y la capacidad de adaptación al cambio.

El diseño es uno de los pilares fundamentales en la ingeniería del software. Es una de las actividades técnicas necesarias para la elaboración del software. Entre las tareas fundamentales del diseño están: producir el diseño de datos, diseño arquitectónico, diseño de interfaz y diseño de componentes. Para su correcto modelamiento es necesario garantizar buenas técnicas y usar soluciones que hayan sido probadas en casos anteriores y hayan arrojado resultados satisfactorios.

1.4.1. Patrones de diseño

Los patrones de diseño son directrices y principios estructurados que describen un problema común y entregan una buena solución, ya probada, a la que le dan un nombre. El uso de patrones permite mejorar la flexibilidad, modularidad y extensibilidad, factores relacionados con la calidad percibida por el usuario y facilita la realización del diseño con un mayor nivel de abstracción.

Christopher Alexander comentó al respecto: “Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esta solución pueda ser usada más de un millón de veces sin hacerla siquiera dos veces de la misma forma.”

Los patrones de diseño contribuyen a reutilizar diseño, identificando aspectos clave de la estructura de un diseño que puede ser aplicado en una gran cantidad de situaciones. La importancia de la reutilización del diseño no es despreciable, debido a que nos provee de numerosas ventajas: reduce los esfuerzos de desarrollo y mantenimiento, mejora la seguridad, eficiencia y consistencia de nuestros diseños, y nos proporciona un considerable ahorro en la inversión.

Patrones GRASP

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable.(Gutierrez, 2007)

Principales patrones GRASP:

Experto: Consiste en asignarle una responsabilidad al experto en información. El problema que resuelve este patrón es el de determinar cuál es la base que debe asumir una responsabilidad a partir de la información que posee cada una.La solución que ofrece es asignar una responsabilidad al experto en información (clase que cuenta con la información necesaria para cumplir la responsabilidad). Este patrón es el más utilizado para asignar responsabilidades.

Creador: Asigna a la clase A la responsabilidad de crear una instancia de clase B. El problema que resuelve es ¿Quién es el responsable de crear alguna nueva instancia de alguna clase? La solución que propone es asignarle a la Clase B la responsabilidad de crear las instancias de la clase A en uno de los siguientes casos:

B agrega los objetos de A.

B contiene los objetos de A.

B registra las instancias de los objetos A.

B utiliza específicamente los objetos A.
B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así B es un Experto respecto a la creación de A).

Bajo Acoplamiento: Asigna responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible. El problema que resuelve es ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización? La solución que propone es asignar una responsabilidad para mantener bajo acoplamiento. Este patrón se deber tener presente siempre durante las decisiones del diseño.

Alta Cohesión: Asigna responsabilidades de forma tal que trabajen sobre una misma área de la aplicación y no tengan mucha complejidad. El problema que resuelve es ¿Cómo manejar las responsabilidades dentro límites manejables? La solución que propone es de asignar responsabilidades que la cohesión siempre siga siendo alta. Este patrón se debe tener presente en todas las decisiones de diseño, es un patrón evaluativo que el desarrollador aplica para valorar sus decisiones de diseño.

Controlador: Asigna la responsabilidad del manejo de mensajes de los eventos del sistema a una clase. El problema que resuelve es ¿Quién debería encargarse de atender un evento del sistema? La solución que propone es de asignar la responsabilidad del manejo del mensaje de los eventos del sistema, a una clase que represente una de las siguientes opciones:
De fachada: el sistema global, la empresa u organización.
De tarea: algo en el mundo real que es activo y que controla la realización de una tarea.
De casos de uso: manejador artificial de todos los eventos del sistema de un caso de uso.

Principales patrones GoF

Los patrones GoF nacen a raíz del trabajo de un grupo de autores conocido como el Gang of Four (GoF). Ellos recopilaron y documentaron 23 patrones de diseño aplicados usualmente por diseñadores de software orientado a objetos. El grupo GoF agrupó los patrones en tres grandes categorías de acuerdo a su propósito. Este criterio describe la función que el patrón cumple. Los patrones de diseño pueden tener propósito creacional, estructural o de comportamiento:

Patrones Creacionales: Se encargan de las formas de crear instancias de objetos. El objetivo de estos patrones es abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.

Patrones Estructurales: Describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.

Patrones Comportamiento: Facilitan definición de la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

Otro criterio de división que se aplica para los patrones es el alcance, que especifica si el patrón es aplicable primariamente a clases o a objetos. Los patrones de clase, tratan con relaciones entre clases y subclases. Los patrones de objeto, en cambio, tratan con relaciones entre objetos.

Creación de la clase: usan la herencia como un mecanismo para lograr la instanciación de la clase. Como ejemplo de estos está:

Factory Method (Método de fabricación): centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que crea.

Creación del objeto: son más escalables y dinámicos comparados con los patrones Creacionales de clases. Como ejemplos de estos están:

AbstractFactory(Fábrica abstracta): permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.

Builder (Constructor virtual): abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto.

Prototype(Prototipo): crea nuevos objetos clonándolos de una instancia ya existente.

Singleton (Instancia única): garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.(Eckel's, 2003)

Estructural de la clase: usan la herencia para proporcionar interfaces más útiles combinando la funcionalidad de múltiples clases. Como ejemplo de estos está:

Adapter(Adaptador): Este patrón es estructural de la clase, usan la herencia para proporcionar interfaces más útiles combinando la funcionalidad de múltiples clases, adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla.(Eckel's, 2003)

Estructural del objeto: crean objetos complejos agregando objetos individuales para construir grandes estructuras. La composición de un patrón estructural del objeto puede ser cambiada en tiempo de ejecución, la cual nos da flexibilidad adicional sobre los patrones estructurales de clases. Como ejemplos de estos están:

Decorator (Envoltorio): añade funcionalidad a una clase dinámicamente.

Facade (Fachada): provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.

Flyweight(Peso ligero): reduce la redundancia cuando gran cantidad de objetos poseen idéntica información.

Proxy: mantiene un representante de un objeto.(Larman, 2002)

Comportamiento de la clase: usan la herencia para distribuir el comportamiento entre clases. Como ejemplos de estos están:

Interpreter (Intérprete): define una gramática para un lenguaje dado, así como las herramientas necesarias para interpretarlo.

TemplateMethod (Método plantilla): define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Esto permite que las subclases redefinan ciertos pasos de un algoritmo sin cambiar su estructura.(Eckel's, 2003)

Comportamiento del objeto: nos permite analizar los patrones de comunicación entre objetos interconectados, como objetos incluidos en un objeto complejo. Como ejemplos de estos están:

Chain of Responsibility (Cadena de responsabilidad): permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada.(Eckel's, 2003)

Command (Orden): encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.(Eckel's, 2003)

Iterator(Iterador): permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos.

Mediator (Mediador): define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto.

Memento (Recuerdo): permite volver a estados anteriores del sistema.

Observer (Observador): define una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambie de estado, se notifiquen y actualicen automáticamente todos los objetos que dependen de él. (Ocaña, 2003)

State (Estado): permite que un objeto modifique su comportamiento cada vez que cambie su estado interno. (Eckel's, 2003)

Strategy (Estrategia): permite disponer de varios métodos para resolver un problema y elegir cuál utilizar en tiempo de ejecución.(Eckel's, 2003)

Visitor(Visitante): permite definir nuevas operaciones sobre una jerarquía de clases sin modificar las clases sobre las que opera.(Ocaña, 2003)

1.5. Métricas para evaluar la calidad del análisis y diseño

Las cuatro razones para medir los procesos del software, los productos y los recursos son caracterizar, evaluar, predecir, y mejorar.(Jimenez, 2009)

Se ha intentado hacer una métrica específica que resulte ser una medida completa de la calidad del software, pero no se ha logrado desarrollar ninguna debido a la complejidad que presenta obtener un único valor al medirlo. Debido a esto se han propuesto muchísimas métricas con un punto de vista diferente. El uso de las mismas se ha hecho frecuente para garantizar la obtención de un producto con alta calidad.

El uso de las métricas tiene como objetivos:

- Comprender mejor la calidad del producto.
- Estimar la efectividad del producto.
- Mejorar la calidad del trabajo realizado en el nivel del proyecto.

1.5.1. Métricas de la calidad de la especificación

Existe un conjunto de características que se utilizan para valorar la calidad del análisis y la especificación de requisitos, los mismos son: especificidad (ausencia de ambigüedad), compleción, corrección, comprensión, capacidad de verificación, consistencia interna y externa, capacidad de logro, concisión, trazabilidad, capacidad de modificación, exactitud y capacidad de reutilización.

Para determinar la especificidad (ausencia de ambigüedad) de los requisitos se sugiere la siguiente métrica basada en la consistencia de la interpretación de los revisores para cada requisito:

$\mathbf{Q_i} = {}^{\mathbf{n_{ui}}}/_{\mathbf{n_r}}$ donde $\boldsymbol{n_{ui}}$es el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas y $\boldsymbol{n_r}$cantidad de requisitos en una especificación, tal como: $\boldsymbol{n_r} = \boldsymbol{n_f} + \boldsymbol{n_{nf}}$ siendo$\boldsymbol{n_f}$ el número de requisitos funcionales y $\boldsymbol{n_{nf}}$ el número de requisitos no funcionales. Cuanto más cerca esté de 1 el valor de Q, menor será la ambigüedad de la especificación.

1.5.2. Métricas de casos de uso

Métrica NOAS /NOS. Esta heurística se basa en la idea de que un caso de uso sirve para expresar una interacción actor–sistema. Por ello, el número de pasos de actor y el de pasos del sistema deben estar en torno al 50%, considerando también la posibilidad de que existan pasos de inclusión o extensión en los que se realice otro caso de uso. El rango habitual de esta métrica es [30%,60%]. Un valor alto de la misma puede estar condicionado por el hecho de que el caso de uso no incluye todo lo que debe hacer el sistema para alcanzar el objetivo de este o demasiado desglose de los pasos del actor. Un valor bajo de la misma puede estar condicionado por el hecho de que no incluye todo lo que debe hacer el actor para alcanzar el objetivo del caso de uso, demasiado desglose de los pasos del sistema.(Bernádez, 2004)

1.5.3. Métricas para evaluar diseño

Métricas propuestas por Lorenz y Kidd. Estos autores proponen métricas basadas en las clases. Las separan en cuatro categorías: tamaño, herencia, valores internos y valores externos. Además, presentan otras métricas orientadas a la complejidad de la clase y al tamaño, estas últimas se centran en contar los atributos y operaciones de cada clase.

Métricas propuestas por Chidamber y Kemerer (CK). Es uno de los conjuntos de métricas más difundidos y conocidas como las CK, también se les llama MOOSE. Adoptan tres criterios a la hora de definirlas: capacidad de satisfacer propiedades analíticas, aspecto intuitivo a los profesionales y facilidad para su recogida automática.

Métricas propuestas por Li y Henry. Se consideran como una extensión del MOOSE con otras métricas de tamaño, acoplamiento y diseño, también proponen un sistema de predicción de reutilización y mantenibilidad. Ellos modificaron y ampliaron las CK, además de añadir algunas nuevas. Tienen un objetivo claro; la medición y mejora de la mantenibilidad del software orientado a objeto.

1.6. Tecnologías, metodologías y herramientas de desarrollo

1.6.1. Metodologías de desarrollo

Las metodologías de desarrollo surgen por la necesidad de evitar problemas que se presentan en la producción de un software por no seguir normas específicas. Cubren todo el ciclo de desarrollo del producto, estableciendo etapas y controles a aplicar en cada momento. Estas recopilan un conjunto de técnicas y procedimientos en cada una de las fases que las componen.

Metodologías robustas

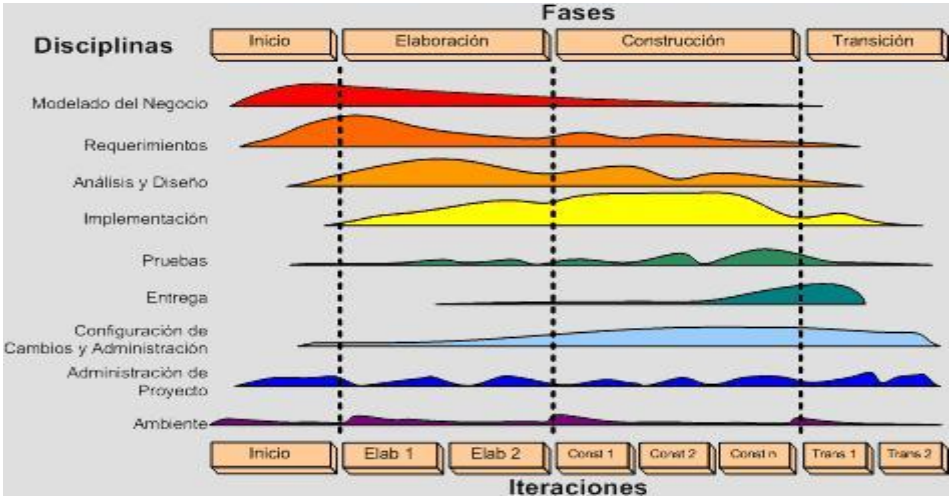
1.6.1.1. Proceso Unificado de Desarrollo (RUP)

EL Proceso Unificado de Desarrollo es una metodología tradicional para el desarrollo de productos de software que permite la realización del análisis y el diseño orientado a objetos. Esta además de ser un proceso, es un marco de trabajo extensible, porque puede ser adaptado a proyectos y organizaciones específicas. A través de una serie de actividades transforma los requisitos de los usuarios de forma tal que se obtiene un sistema de software que satisfaga las necesidades de los clientes.

Características principales de RUP

- ✓ RUP está dirigido y guiado por casos de uso. Los casos de uso además de ser una representación de los requisitos funcionales del sistema, son utilizados por los desarrolladores e ingenieros de prueba para la realización de su trabajo, debido a que los mismos orientan su diseño, implementación y prueba.
- ✓ El proceso de desarrollo avanza a través de una serie de flujos que parten de los casos de uso.(Jacobson, et al., 2000)
- ✓ Centrado en la arquitectura. La arquitectura es una vista de diseño completo con las características más importantes resaltadas.(Jacobson, et al., 2000)
- ✓ Iterativo e incremental. En el proceso unificado cada iteración se refiere a pasos en el flujo de trabajo y el incremento al crecimiento del producto.(Jacobson, et al., 2000)

El ciclo de vida de RUP cuenta con 4 fases fundamentales: inicio, elaboración, construcción y transición.



Fases y Flujos de trabajo de RUP.

Inicio

La fase de inicio tiene como objetivo definir los límites y la visión del proyecto, así como realizar la estimación riesgos potenciales y establecer la justificación económica. En esta fase se planifica el proyecto.

Elaboración

En la fase de elaboración se define la arquitectura del sistema, es donde se realiza la especificación detallada de cada caso de uso.

Construcción

Durante la fase de construcción, fundamentándose en la línea base de la arquitectura establecida, se fabrica el producto y se define si está preparado para el despliegue.

Transición

En la fase de transición se garantiza la disponibilidad del software para los usuarios, además se prueba el producto para detectar y corregir errores. En la misma se le brinda asistencia ayuda al cliente para su formación.

1.6.1.2. Microsoft solutions framework (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.

Las principales características que presenta la metodología MSF son:

- ✓ Adaptable: es parecido a un compás, usado en cualquier parte como un mapa, del cual 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 a 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.

Esta metodología está compuesta por modelos, principios y disciplinas. Los modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto son: 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.
La principal desventaja de MSF es que su desarrollo se basa en tecnología Microsoft la cual es cara y restringe mucho las herramientas de desarrollo.

Metodologías ágiles

1.6.1.3. Crystal methodologies

Es un conjunto de metodologías que tienen como objetivo reducir al máximo el número de artefactos producidos y se centra en las personas que componen el equipo, quienes deben invertir esfuerzo en mejorar sus habilidades porque son considerados el factor clave en dicha metodología. Cada equipo está regido por un conjunto de políticas que dependerán del tamaño del mismo. El desarrollo de software se considera un juego cooperativo de invención y comunicación.

Crystal Methodologies presenta algunos impedimentos para ser aplicada al proyecto SGF. Esta metodología está basada en iteraciones cortas, va dirigida a equipos pequeños o medianos y que el entorno físico debe ser un ambiente que permita la comunicación y colaboración entre todos los miembros del equipo durante todo el tiempo que dure el desarrollo del proyecto.

1.6.1.4. Scrum

Esta metodología define la gestión de proyectos. Es dirigida a proyectos con un constante cambio de requisitos. Sus características principales son que el desarrollo de software se realiza mediante iteraciones (representan un incremento ejecutable que se muestra al cliente); y que a lo largo del proceso de desarrollo se realizan constantes reuniones. Una de ellas y la que más se destaca por su importancia es la que se efectúa diariamente para realizar coordinación e integración. Esta metodología contiene algunas ventajas como son: que entrega al cliente un producto funcional en periodo de 30 días, muestra una visualización diaria del proyecto y está dotada de un equipo integrado y comprometido, capaz de auto-administrarse, una vez que han definido el alcance del proyecto.

Scrum presenta también algunos inconvenientes. No genera toda la evidencia o documentación de otras metodologías, no es aplicable a todos los proyectos y en ocasiones es necesario integrarlo con otros procesos como es el caso de XP.

1.6.2. Lenguajes de modelado

1.6.2.1. Métodos Integrados de Información (IDEF)

IDEF es una técnica de modelado de sistemas que comprende desde la modelación de la información de un sistema u organización hasta el análisis y diseño orientado a objetos.

La familia de técnicas de modelado IDEF fue desarrollada como un conjunto de notaciones formales para representar y modelar procesos y estructuras de datos en una forma integrada, es decir, proveer técnicas de modelado simples y formales que permitan describir, analizar y evaluar distintos puntos de vista de un sistema.

Un modelo IDEF 0, permite representar las actividades que conforman un sistema de forma jerárquica. Está compuesto por un conjunto de diagramas que facilitan la descripción de las funciones especificadas en el nivel superior. En las vistas superiores del modelo la interacción entre las actividades representadas permite visualizar los procesos fundamentales que sustentan la organización. Los elementos gráficos utilizados para la construcción de los diagramas IDEF 0 son cuadros y flechas.

IDEF1 es empleado como modelo de representación y estructuración de la información.

IDEF2 sirve para representar modelos que varían con el tiempo.

IDEF3 es una técnica de modelación para representar el flujo de trabajo de un proceso.

1.6.2.2. BPMN (Business Process Modeling Notation)

BPMN es un lenguaje de modelado para modelar conceptualmente procesos de negocio. Proporciona la capacidad de entender y definir dichos procesos, ya sean internos o externos, a través de un diagrama de procesos de negocio. Se diseñó con el objetivo de facilitar la comprensión por parte de todos los implicados (expertos TIC, analistas de negocio, directivos, etc.) que participan en el proceso.

El modelado de procesos de negocio suele empezar capturando actividades de alto nivel para luego ir bajando de nivel de detalle dentro de diferentes diagramas. Puede haber múltiples niveles de diagramas, dependiendo de la metodología usada para desarrollar los modelos. De todas formas, BPMN es independiente de cualquier metodología.

BPMN define un Business Process Diagram (BPD), que se basa en una técnica de grafos de flujo para crear modelos gráficos de operaciones de procesos de negocio. Un modelo de procesos de negocio, es una red de objetos gráficos, que son actividades y controles de flujo que definen su orden de rendimiento.

1.6.2.3. Lenguaje Unificado de Modelado (UML)

El UML es la unión de un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe lo que estos diagramas y símbolos significan. Una de las metas principales de UML es avanzar en el estado de la integración institucional proporcionando herramientas de interoperabilidad para el modelado visual de objetos.

Entre sus características más importantes está que introduce un ítem universal para agrupar un gran número de elementos permitiendo dividir el sistema en partes, también los paquetes pueden ser usados desde el nivel más alto hasta el nivel más bajo, que en la mayoría de los casos contienen casos de uso independientes, clases o componentes.

1.6.3. Herramientas de prototipado

1.6.3.1. Axure RP Pro

Es una herramienta para crear plantillas para aplicaciones web. La interfaz es sencilla y ofrece el cómodo sistema de arrastrar y soltar. Tiene una sola ventana desde donde operar. Le permite al usuario la

posibilidad de descargar más herramientas desde la web del usuario. Se ha convertido rápidamente en el favorito de los usuarios y las comunidades de arquitectura de la información. La aplicación facilita la construcción de modelos de alambre, documento de especificaciones funcionales, y la generación de prototipos, todas ellas con un sistema integrado de control de versiones. Es una herramienta rápida que permite una completa flexibilidad en el diseño de una interfaz.

1.6.3.2. Pencil

Pencil es un plugin para Firefox que permite construir wireframes y prototipos. Como herramienta de creación de prototipos es muy buena, permitiendo establecer rápidamente una maqueta con una alta fidelidad. Se basa en arrastrar y soltar elementos gráficos pre-hechos. Los elementos de alambre de salida que se incluyen con Pencil tienden a basarse en la apariencia de una aplicación de escritorio de Windows, lo que no es deseable para una aplicación web. Otra desventaja que presenta es que su funcionalidad de exportación proporciona sólo un formato de imagen, dando lugar a que Pencil no sea la herramienta interactiva de desarrollo de prototipos.

1.6.3.3. Visio

Visio comenzó como un complemento para MS Word, llenando la necesidad de un negocio y técnicas de diagramación de herramientas, luego pasa a la condición de producto separado. El poder de Visio permanece en el área de la documentación esquemática, como una herramienta de creación de prototipo. Sin embargo Visio, es particularmente adecuado para el contenido de generación de mapas a partir de archivos CSV. En la versión reciente, Visio 2007, la adición del plug-in de UML ha permitido la importación de elementos UML. Visio genera una serie de herramientas adicionales como lo es el Swipr[1] que es gratuita y es la más relevante, permitiendo la rapidez de las exportaciones de wireframes y flujos de pantalla. La mayor desventaja de Visio es que sólo está disponible en la plataforma Windows.

[1]Conjunto de herramientas para crear una estrategia integrada y prestación interactivos a partir de archivos estándar de Visio.

1.6.4. Herramientas CASE (Ingeniería de Software Asistida por Ordenador)

Hoy en día, muchas empresas se han extendido a la adquisición de herramientas CASE, con el fin de automatizar los aspectos clave de todo el proceso de desarrollo de un sistema. CASE proporciona un conjunto de herramientas automatizadas que están desarrollando una cultura de ingeniería. Uno de los objetivos, más importante a largo plazo, es conseguir la generación automática de programas desde una especificación a nivel de diseño.

Para contribuir a mejorar la calidad y la productividad en el desarrollo de sistemas de información, las herramientas CASE se plantean los siguientes objetivos:

- Permitir la aplicación práctica de metodologías estructuradas, las cuales al ser realizadas con una herramienta se consigue agilizar el trabajo.
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
- Mejorar y estandarizar la documentación.
- Aumentar la portabilidad de las aplicaciones.
- Facilitar la reutilización de componentes software.
- Permitir un desarrollo y un refinamiento visual de las aplicaciones, mediante la utilización de gráficos.

1.6.4.1. Rational Rose

Es una herramienta orientada a objetoUML, está destinada al modelado visual y construcción de componentes de aplicaciones empresariales de software a nivel. Dos características populares que presenta son su capacidad para proporcionar el desarrollo iterativo y la ingeniería de viaje redondo. Mediante la utilización de esta herramienta los desarrolladores pueden crear nuevas etapas de desarrollo, porque la salida de una iteración puede convertirse en la entrada de la siguiente.

Algunas características que ofrece(Rational, 2003):

- Soporte para análisis de patrones ANSI C++, Rose J y Visual C++.
- Característica de control por separado de componentes modelo que permite una administración más granular y el uso de modelos.

- Soporte de ingeniería directa y/o inversa para algunos de los conceptos más comunes de Java 1.5.
- La generación de código Ada, ANSI C++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo- código configurables.
- Capacidad de análisis de calidad de código.

1.6.4.2. Enterprise Architect (EA)

Enterprise Architect es una herramienta de modelado multiusuario diseñada para el desarrollo de sistemas robustos y duraderos. Cubre el ciclo de vida completo para el desarrollo de un proyecto, para lo que brinda capacidades de gestión de requisitos y permite la realización de especificaciones de alto nivel a modelos de análisis, diseño, implementación, pruebas y mantenimiento, usando UML y BPMN.

La arquitectura en Enterprise Architect, dirigida por modelos, facilita la obtención de elementos complejos del modelo a partir de los más simples. Esta herramienta genera código fuente en C++, Java, C#, VB.Net, Visual Basic, Delphi, PHP, Python y ActionScript. Cuenta con plug-ins para vincularse a Visual Studio, NET o Eclipse. La ingeniería inversa para muchos de los sistemas gestores de bases de datos más populares, como Oracle, SQL Server, My SQL, Access y Postgre SQL es otra de las habilidades que oferta.

1.6.4.3. Visual Paradigm

Visual Paradigm es una herramienta de diseño que soporta todos los diagramas UML y de entidad-relación. Ofrece amplias características de modelado de caso de uso incluyendo la función completa de UML, Diagrama de Casos de Uso, flujo de eventos y otras representaciones gráficas. Produce la documentación completa del sistema en formato PDF, HTML y MS Word. Los desarrolladores pueden diseñar la documentación del sistema y los analistas estimar las consecuencias de los cambios a través de los diagramas. Es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue.

Algunas funcionalidades que ofrece Visual Paradigm son:

- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa en su versión profesional, e inversa.

- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDE.
- Disponibilidad en las plataformas Windows y Linux

1.6.5. Lenguajes de programación

Un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo, son herramientas que permiten crear programas y software. Los lenguajes de programación web facilitan la interacción y personalización de la información web con el usuario. Estos son aplicados tanto del lado del servidor como del lado del cliente. El PERL, ASP, PHP son los principales lenguajes de programación web por el lado del servidor, estos son los encargados de desarrollar la lógica del negocio dentro del servidor así como acceder a la base de datos; por otra parte entre los principales lenguajes del lado del cliente se encuentran Java Script (Script) y el Visual Basic Script (VBScript) que se encargan de aportar dinamismo a la aplicación de los navegadores.

1.6.5.1. Lenguaje 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 (European Computer Manufacturers Association) e ISO (International Organization for Standardization). Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes, como es el caso de Delphi.

Ventajas:

Su código se puede tratar íntegramente como un objeto. Mediante su uso se logra ahorrar tiempo en la programación porque cuenta con una librería de clases muy completa y bien diseñada. En C# existe un rango más amplio y definido de tipos de datos que los que se encuentran en C, C++ o Java. Antes de que un método pueda ser redefinido en una clase base, debe declararse como virtual. El método redefinido en la subclase debe ser declarado con la palabra override. C# permite la declaración de propiedades dentro

de cualquier clase, debido a que las clases pueden ser utilizadas como objetos y los objetos tienen internamente propiedades.

Desventajas:

Las desventajas que se derivan del uso de este lenguaje de programación son que en primer lugar se debe contar con una versión reciente de Visual Studio .NET y algunos requisitos mínimos del sistema para poder trabajar adecuadamente, tales como,disponer con Windows NT 4 o superior y 4 gigas de espacio libre para su completa instalación.

1.6.5.2. Lenguaje Java

Java es un lenguaje de programación orientado a objeto desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995.

Ventajas:

Es un lenguaje multiplataforma, se puede descargar de forma gratuita. Disponible de una gran cantidad de librerías que añaden una potencia increíble al lenguaje.

Su integración con la red es asombrosa, puesto que existen librerías para acceder e interactuar con protocolos como http yftp, lo que permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales.

Desventajas:

El compilador es su principal desventaja pues sabe decir si un programa al compilarlo tiene errores pero no sabe decir con precisión el lugar donde está el error. La Máquina virtual Java consume muchos recursos, así que debemos tener un ordenador con muy buenas propiedades sino se notará mucho la recarga de memoria.

1.6.5.3. Lenguaje PHP

PHP (Hypertext Pre-processor) es un lenguaje de programación interpretado y diseñado originalmente para la creación de páginas web dinámicas. A diferencia de Java o JavaScript que se ejecutan en el navegador, PHP se ejecuta en el servidor por lo que posibilita acceder a los recursos que tenga el servidor, como es el caso de la base de datos, por esta razón no es necesario que su navegador lo soporte, es independiente del navegador, sin embargo, para que sus páginas PHP funcionen el servidor donde están alojadas debe soportar PHP. Debido a esta característica el resultado de su ejecución es enviado al navegador, este normalmente es una página HTML pero también podría ser una página WML (Wireless Markup Language).

Las principales ventajas de PHP son: su rapidez; su facilidad de aprendizaje; su soporte multiplataforma tanto de diversos Sistemas Operativos, como servidores HTTP y de bases de datos; y el hecho de que se distribuye de forma gratuita bajo una licencia abierta así como que permite técnicas de programación orientada a objeto y a nivel de código, no requiere de definición de variables y tiene manejo de excepciones.

Este lenguaje libre dispone de una gran cantidad de características que se mencionan a continuación:

- ✓ Soporte para una gran cantidad de bases de datos: MysSQL, PostgreSQL, Oracle, MS SQL Server, SybasemSQL, Informix, entre otras.
- ✓ Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader) hasta analizar código XML.
- ✓ Ofrece una solución simple y universal para las paginaciones dinámicas del Web de fácil programación.
- ✓ Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.
- ✓ Soportado por una gran comunidad de desarrolladores, como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente.
- ✓ El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP.

- ✓ Con PHP se puede hacer cualquier cosa que podemos realizar con un script CGI, como el procesamiento de información en formularios, foros de discusión, manipulación de cookies y páginas dinámicas.(Zubyc, 2009)

1.6.6. Framework de desarrollo

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.(Potencier, 2008)

1.6.6.1. php.MVC

El framework php.MVC implementa del patrón de diseño Modelo-Vista-Controlador (MVC), y alienta el diseño de aplicaciones basadas en el paradigma del modelo 2. Este modelo de diseño permite que la página web u otros contenidos puedan seren su mayoría, separados del código de la aplicación interna (Controller / Model), lo que hace que sea más fácil para los diseñadores y programadores centrarse en sus respectivas esferas de competencia. El framework proporciona un único punto de entrada al controlador. El Contralor es el responsable de asignar las peticiones HTTP al gestor de acción correspondiente basándose en las asignaciones de configuración. El modelo contiene la lógica de negocio para la aplicación. El controlador envía la solicitud al componente Vista apropiado, que suele ejecutarse mediante una combinación de etiquetas HTML con PHP en la forma de plantillas. El contenido resultante se devuelve al navegador del cliente, o a través de otro protocolo como SMTP (Simple Mail Transfer Protocol).

1.6.6.2. Kumbia

Kumbia es un framework libre escrito en PHP5. Se basa en las mejores prácticas de desarrollo web, usando un software comercial y educativo. Fomenta la velocidad y eficiencia en la creación y mantenimiento de aplicaciones web, reemplazando tareas de codificación repetitivas por poder, control y placer. Es idóneo para la creación de proyectos en PHP con características como: Sistema de Plantillas

sencillo, Administración de Cache, Scaffolding Avanzado, Programación Modular, Modelo de Objetos y Separación MVC, Soporte para AJAX, Componentes Gráficos y Seguridad.

1.6.6.3. Symfony

Symfony fue diseñado para que se ajustara a los siguientes requisitos:

- ✓ Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares[2]).
- ✓ Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- ✓ Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- ✓ Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- ✓ Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- ✓ Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- ✓ Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros. (Potencier, 2008)

1.7. Fundamentación de tecnologías, metodologías y herramientas de desarrollo seleccionadas

Después de haber realizado un análisis de las características fundamentales de las tecnologías, metodologías y herramientas de desarrollo para la realización de este trabajo, se consideró como elemento primordial el que sean tecnologías o herramientas libres o de código abierto, debido a las características que presentan, que los hacen viables para Cuba, por su desenvolvimiento económico.

Para un buen desarrollo del proyecto es crucial una correcta selección de la metodología a utilizar. Después de analizar las características fundamentales de las metodologías de desarrollo se concluyó que

[2]Abreviatura utilizada para referirse al conjunto de sistemas UNIX como Linux, Minix, Ultrix, y Xenix.

era necesario emplear RUP. Esta robusta metodología constituye un estándar probado a nivel mundial, hace énfasis en realizar una buena elicitación de requisitos y por ende un fructuoso diseño. Es aconsejable su uso para proyectos grandes y de larga duración como es el caso de SGF. RUP genera un sinnúmero de artefactos que permiten la documentación del módulo Inspección a LD y CP, facilitando la comunicación a los miembros del equipo.

Luego de estudiar los diferentes lenguajes de modelado se seleccionaron BPMN para modelar los procesos de negocio y UML para el resto de los procesos.

BPMN es elegido debido a la complejidad que presentan los procesos de negocio del módulo Inspección a LD y CP, dado que este lenguaje considera un único diagrama para representar los procesos de negocio, que está basado en la técnica de diagramas de flujo y adaptado para graficar las operaciones de los procesos de la organización; no utilizando UML para modelar estos procesos debido a que su enfoque orientado a objetos dificulta la comunicación con el cliente porque está pensado para un público predominantemente técnico, a diferencia de BPMN que está compuesto por un conjunto de elementos gráficos que facilitan que tanto los miembros del equipo de desarrollo como el cliente entiendan con claridad sus diagramas. UML por su parte está más orientado a diseñadores de software. Es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos. El mismo está definido como estándar en el análisis y diseño de sistemas. Los diseños realizados se pueden implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos). Establece conceptos y artefactos ejecutables. Además, es el lenguaje propuesto por RUP para llevar la documentación del sistema.

Para el modelamiento del sistema se seleccionó Visual Paradigm, debido a las características y ventajas que posee. Esta herramienta CASE soporta UML y BPMN, que son los lenguajes de modelado seleccionados. La principal razón que sustenta esta elección es que la universidad cuenta con una licencia para el uso de la misma, además en el año 2009 se realizó un taller de arquitectura donde se decidió que los proyectos de la UCI, en su mayoría, la emplearan para modelar sus artefactos. Es multiplataforma, soporta el ciclo de vida completo del desarrollo de software, permite dibujar todos los tipos de diagramas de clases, genera documentación a partir de los mismos y es la que más se integra a Eclipse, que es el entorno de desarrollo integrado, utilizado por los implementadores.

Como herramienta para el prototipado de interfaces se seleccionó Axure RP Pro, dado que la misma permite observar el desarrollo de la aplicación a través de la simulación del sitio web y genera un

documento en formato Word con la imagen de cada una de las interfaces diseñadas. Fue la herramienta elegida por los directivos de proyecto para el desarrollo de prototipos, porque es utilizada por los arquitectos de información del proyecto SGF. Esto facilita la comunicación entre el equipo de desarrollo.

Teniendo en cuenta las consideraciones del uso de tecnologías libres o de código abierto se usará PHP como lenguaje del lado del servidor porque cuenta con una distribución gratuita bajo licencia abierta. Este es un lenguaje simple, elegante y fácil de aprender. Se ejecuta rápidamente ocupando poca memoria y corre sobre cualquier plataforma usando el mismo código fuente.

Symfony, desarrollado completamente en PHP, se escogió como framework, debido a que fue diseñado para optimizar el desarrollo de aplicaciones web. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Además, de poseer una extensa documentación, cuenta con un foro en español en el cual se pueden intercambiar ideas y evacuar dudas. Facilita la reducción considerable del tiempo de desarrollo del software debido a la característica que posee de generar clases y herramientas.

Conclusiones

Con el estudio de diferentes sistemas de gestión jurídicos se demostró la necesidad del desarrollo del proyecto SGF y por ende del módulo Inspección a Locales de Detención y Centros Penitenciarios, para alcanzar una mejor organización en los procesos que se desarrollan en el Departamento CLEP de la Fiscalía General de la República y en las fiscalías provinciales y municipales.

Para dar solución a dicha actividad se estudiaron las principales metodologías, tecnologías y herramientas. Se designó como metodología RUP porque era la que más se ajusta a las necesidades del proyecto. Para el modelado se utilizan BPMN y UML porque cuentan con características específicas para cada uno de los flujos de trabajo que modelaran. Con el objetivo de mejorar y estandarizar la documentación se escogió Visual Paradigm, herramienta que soporta todo el ciclo del desarrollo de un proyecto.

Debido a que se está luchando en la Universidad por alcanzar la independencia tecnológica se seleccionaron tecnologías potentes en su mayoría multiplataforma y de software libre. De ahí la elección de PHP como lenguaje de programación y Symfony como framework. Una vez seleccionada las tecnologías, metodologías y herramientas de desarrollo a utilizar, se da paso a la realización del análisis y diseño del módulo.

2. Capítulo 2. Análisis del sistema

2.1. Introducción

En este capítulo se comienza a tener la concepción práctica sobre lo que se pretende que haga el módulo Inspección a Locales de Detención y Centros Penitenciarios. Se tiene en consideración las leyes establecidas, las necesidades y características del Departamento CLEP de las fiscalías cubanas.

Se realiza una descripción de los procesos del negocio, además se definen los requisitos tanto funcionales como no funcionales, estableciendo lo que el sistema debe hacer y la forma en que lo hará. Es decir, se tratarán los artefactos determinados en la metodología de desarrollo RUP para el análisis de dicho módulo.

2.2. Modelo de negocio

Los sistemas generalmente suelen ser muy complicados, es por eso que se dividen en pequeñas piezas que se representan a través de modelos que permiten la abstracción de sus características esenciales. El modelado del negocio es una actividad crítica para la comprensión y reestructuración de las actividades que una organización establece para lograr los objetivos y metas de negocio. Con este modelo se refuerza la idea de que sea el propio negocio lo que determine los requisitos.

2.3. Modelo de negocio basado en proceso

Para conseguir sus objetivos, una empresa organiza su actividad por medio de un conjunto de procesos de negocio. Cada uno de ellos se caracteriza por una colección de datos que son producidos y manipulados mediante un conjunto de tareas, en las que ciertos agentes (por ejemplo, trabajadores o departamentos) participan de acuerdo a un flujo de trabajo determinado. Además, estos procesos se hallan sujetos a un conjunto de reglas de negocio, que determinan las políticas y la estructura de la información de la empresa. Por tanto, la finalidad del modelado del negocio es describir cada proceso, especificando sus datos, actividades (o tareas), roles (o agentes) y reglas de negocio. El primer paso del modelado del negocio consiste en capturar los procesos de negocio de la organización bajo estudio. La

definición del conjunto de procesos del negocio es una tarea crucial, porque define los límites del proceso de modelado posterior.(Molina, 2007)

2.3.1. Reglas del negocio

Las Reglas del Negocio (Business Rules) describen las políticas, normas, operaciones, definiciones y restricciones presentes en una organización y que son de vital importancia para alcanzar los objetivos misionales. Las organizaciones funcionan siguiendo múltiples reglas de negocio, explícitas o tácitas, que están embebidas en procesos, aplicaciones informáticas, documentos, etc. Pueden residir en la cabeza de algunas personas o en el código fuente de programas informáticos.(Deadalus, 2010)

A continuación se muestran las reglas identificadas de las inspecciones a LD y CP:

- ✓ Al inspeccionar un Local de Detención o Centro Penitenciario en caso de que no existan violaciones, o las que existan queden solucionadas durante el desarrollo de la misma, no sean reiterativas o no tengan gravedad, ni trascendencia se emite un Acta, en caso contrario se emite una Resolución.
- ✓ Si el Plan de Medidas satisface los requisitos se archiva con la resolución en la OCIC.
- ✓ Si el Plan de Medidas no satisface los requisitos se devuelve a la persona notificada.
- ✓ El Acta emitida como resultado de la inspección es siempre archivada.
- ✓ Cada municipio debe confeccionar su Plan Trimestral de Inspección a Centros Penitenciarios el último mes de cada trimestre y enviarlo a la Departamento Provincial CLEP.
- ✓ El Órgano Infractor tiene diez días naturales para impugnar la Resolución.
- ✓ El Fiscal Superior Jerárquico del que emite la Resolución tiene diez días naturales para tramitar la impugnación de la misma.
- ✓ Si una Resolución es impugnada, se recepciona el Escrito de Impugnación y se actualiza el Registro Primario.
- ✓ Si una Resolución no es impugnada el destinatario de la Resolución confecciona y remite al Fiscal el Plan de Medidas.
- ✓ El Superior Jerárquico tiene veinte días para hacer cumplir los pronunciamientos del Fiscal CLEP en caso de que el Infractor no envíe el Plan de Medidas.

- ✓ El Infractor tiene veinte días a partir de la notificación de la Resolución para enviarle el Plan de Medidas al Fiscal CLEP.
- ✓ Un Registro de notas contiene el resultado de la comprobación de los objetivos propuesto o de cualquier otra acción o actividad realizada por el Fiscal CLEP durante la ejecución de la inspección y el nombre o firma de la persona que lo confeccione.
- ✓ Toda entrada o salida de documentos requiere la actualización de los correspondientes registros o controles, realizados por la Secretaria o Funcionario de la OCIC según corresponda.
- ✓ Si como resultado de la inspección se detectan presuntos hechos delictivos en los que se encuentran involucrados militares, se da cuenta a la Fiscalía Militar.
- ✓ Todos los documentos generados en una inspección forman parte del Rollo.
- ✓ Siempre que se encuentren detenidos ilegales fuera de los casos previstos en la ley o detenidos ilegales por incumplimiento de los términos de detención, se considerará como una violación grave.

2.3.2. Participantes del negocio

Participantes de la Fiscalía	Descripción
Fiscal CLEP	Comprende el órgano municipal, el provincial y nacional. Propone los objetivos que serán comprobados o verificados durante la inspección a los establecimientos penitenciarios. Localiza las normativas necesarias y la información para comprobar los objetivos. Participa en la inspección y en caso necesario, confecciona y notifica el Auto disponiendo Libertad. Realiza la reunión de conclusión y hace entrega a la secretaria de los documentos que se generan en la inspección.
Secretaria	Actualiza los registros primarios. Revisa el

	Documento que se enviará a un destinatario. Archiva el Rollo final con los documentos de la inspección. Realiza el envío de la Resolución a la OCIC. Alerta sobre el cumplimiento de los términos. Remite el Plan Trimestral a los Municipios.
Fiscal Jefe	Puede realizar visitas a Locales de Detención y a Centros Penitenciarios y decidir los objetivos que se verificarán en la inspección. Toma cualquier otra decisión para la organización, planificación, control y ejecución de las visitas de inspección a los Locales de Detención y Centros Penitenciarios.

Descripción de trabajadores de la FGR.

Participantes del Establecimiento Penitenciario	Descripción
Jefe del Establecimiento Penitenciario	Recibe la inspección de la Fiscalía y acompaña a los fiscales todo el tiempo que dure el recorrido por la instalación. Envía el Plan de Medidas a la Fiscalía en caso de que sea víctima de una Resolución, además establece el recurso de impanación en el caso de no estar de acuerdo con las violaciones que se le imputan.

Descripción de trabajador del establecimiento penitenciario.

2.3.3. Procesos que tienen lugar en la inspección a CP y LD

- ✓ Inspeccionar Locales de Detención
- ✓ Inspeccionar Centros Penitenciarios
- ✓ Decidir Impugnación
- ✓ Receptar Plan de Medidas
- ✓ Tomar decisiones

A continuación se explica de forma concisa en qué consiste cada proceso. Una descripción más detallada con el diagrama de negocio correspondiente se puede encontrar en el documento Modelo de Negocio con BPMN que se adjunta a la investigación. ***(Ver Modelo de Negocio).***

Proceso de negocio: Inspeccionar Locales de Detención

El proceso se inicia cuando el Fiscal Jefe Municipal, Vice fiscal Jefe Municipal, Fiscal Jefe de Departamento CLEP o Fiscal Jefe de Dirección CLEP, según corresponde o se decida, confeccionan un Plan Trimestral de Inspección a Locales de Detención. Se seleccionan por el Fiscal CLEP y el Asistente del Fiscal los objetivos que serán comprobados, se localiza las normativas e información necesaria para su comprobación. Se procede a realizar la inspección al Local de Detención conjunto con el Jefe de la Unidad, comprobando los objetivos trazados y finalizando con una reunión de conclusión y la actualización del Registro Primario.

Proceso de negocio: Inspeccionar Centros Penitenciarios

Este proceso se diferencia del anterior en la forma en que se genera el Plan Trimestral de Inspección a Centros Penitenciarios y la definición de los objetivos que serán comprobados. Al igual que el antepuesto se procede a realizar la inspección al Centro Penitenciario, se comprueban los objetivos trazados y se finaliza con una reunión de conclusión y la actualización del Registro Primario de información.

Proceso de negocio: Tomar decisiones

El proceso se inicia cuando el Fiscal Jefe Municipal, o el Vice fiscal Municipal o el Fiscal CLEP (consultado con Fiscal Jefe Municipal, Fiscal Jefe de Departamento CLEP o Fiscal Jefe de Dirección), en caso que se decida, luego de analizar la documentación derivada de la inspección ejecutada, toma una decisión

acerca del documento que va a emitir, con respecto a la inspección realizada a un centro determinado y termina con el envío de un Documento o Escrito al destinatario correspondiente. Este proceso contiene dos subprocesos: Confeccionar Acta y Confeccionar Resolución.

El subproceso Confeccionar Acta comienza cuando el Fiscal CLEP luego de hacer el estudio del Registro de notas de la reunión de conclusión u otros Registros de notas y consultado con Fiscal Jefe Municipal, Fiscal Jefe de Departamento CLEP o Fiscal Jefe de Dirección, según proceda, o el propio Fiscal Jefe Municipal o Vice fiscal Jefe Municipal, en caso que se decida, confecciona el acta resultado de la inspección, finalizando el proceso con el archivo del Rollo.

El subproceso Confeccionar Resolución comienza cuando el Fiscal CLEP, o el propio Fiscal Jefe Municipal o Vice fiscal Jefe Municipal, en caso que se decida, hace un estudio del Registro de notas de la Reunión de conclusión u otros Registros de notas y luego de consultado con el Fiscal Jefe Municipal, Fiscal Jefe de Departamento CLEP o Fiscal Jefe de Dirección, según proceda, confecciona la Resolución de la inspección, finalizando el proceso con el archivo del Rollo.

Proceso de negocio: Receptar Plan de Medidas

El proceso comienza cuando la Secretaria archiva el Plan de Medidas y lo entrega al Fiscal CLEP, Fiscal Jefe Municipal, Fiscal Jefe de Departamento CLEP o Fiscal Jefe de Dirección CLEP, según corresponda, para que sea revisado.

El proceso de negocio concluye cuando el Responsable de la OCIC archiva el Plan de Medidas en la Resolución.

Proceso de negocio: Decidir Impugnación

El proceso comienza cuando el Funcionario o Jefe del Local inspeccionado archiva y revisa la Resolución que se le notifica y decide impugnar o no la misma. En caso de ser impugnada, en dependencia de la respuesta al Escrito de Impugnación, concluye el proceso con el envío o no del Plan de Medidas del Infractor.

2.4. Especificación de requisitos

El modelamiento del Negocio ofreció una visión general de los procesos de inspección realizados en el departamento de Control de la Legalidad de las fiscalías cubanas, permitió comprender a que se dedica el mismo, así como establecer una comunicación entre el cliente y el equipo de desarrollo. Una vez en este punto, se hace necesario comenzar a definir qué es lo que debe hacer el sistema, siendo necesario ir a la captura de los requisitos que este debe cumplir.

Un requisito es una representación documentada de una condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo, que puede ser alcanzada o poseída por un sistema o un componente para satisfacer un contrato, estándar, u otro documento impuesto formalmente. Permite definir el ámbito del sistema; detalla una interfaz de usuarios para el sistema, enfocada a las necesidades y metas del usuario; así como establece y mantiene un acuerdo entre clientes y otros involucrados sobre lo que el sistema debería hacer. Se pueden encontrar dos tipos de requisitos: funcionales y no funcionales. (S.Pressman, 2005)

Todas las ideas aportadas por el cliente, referente a lo que debe hacer el sistema, deben ser analizadas como posibles requisitos del software.Para la captura de requisitos se utilizaron las siguientes técnicas: Entrevistas, Prototipado, Tormenta de ideas, Introspección y Arqueología de Documentos.

2.4.1. Requisitos Funcionales

Los requisitos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Además, son independientes de las tecnologías usadas por el producto.

A partir de los procesos de negocio estudiados y las actividades a automatizar identificadas en el módulo Inspección a Locales de Detención y Centros Penitenciarios, se identificaron 70 requisitos funcionales.

En este apartado se registran el total de requisitos capturados, y se especifican las capacidades funcionales que el sistema deberá cumplir con el mayor detalle posible. Para consultar los requisitos funcionales. ***(Ver Especificación de Requisitos).***

2.4.2. Requisitos no Funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Definiendo a las propiedades como características que hacen al producto atractivo, usable, rápido o confiable. En muchos casos los requisitos no funcionales son fundamentales en el éxito del producto y normalmente están vinculados a requisitos funcionales, es decir, una vez se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. Los 34 requisitos no funcionales identificados del módulo se detallan en el documento Especificación de requisitos. ***(Ver Especificación de Requisitos).***

2.5. Modelo de caso de uso del sistema

Los requisitos funcionales del sistema fueron agrupados en casos de uso (CU), para lo cual se utilizaron los patrones de CU Concordancia Reusabilidad y CRUD parcial.

El modelo de casos de uso del sistema, constituido por casos de uso, actores y las relaciones que se establecen entre ellos; describe a través de las especificaciones de casos de uso las funcionalidades propuestas para el sistema que está siendo desarrollado. Un caso de uso representa la interacción entre el usuario y el sistema.

2.5.1. Definición de actores

Teniendo en cuenta que en el modelo de casos de uso queda descrito lo que hace el sistema para cada tipo de usuario y cada uno de ellos se representa mediante uno o más actores, se puede decir que los actores simbolizan terceros fuera del sistema que colaboran con el mismo. A continuación se describen los actores del sistema propuesto:

Actores del Sistema	Descripción
Fiscal CLEP	Es el actor encargado de crear y actualizar los registros que se harán en la Fiscalía referentes a las inspecciones realizadas a un CP o LD. Cuenta con permisos para registrar y modificar

	documentos como: Acta, Resolución, Auto de Libertad, Documento de otra modalidad de reacción fiscal, Respuesta al Documento de otra modalidad de reacción fiscal, Plan de Medidas, Plan de objetivos, Diligencia de Notificación y Documento de Comunicación. Tiene permiso de lectura de los siguientes documentos: Plan de Inspección a LD, Plan de Inspección a CP y la Resolución resolviendo el recurso de impugnación.
Fiscal Jefe Municipal	Es el actor encargado de gestionar los asuntos más importantes que se llevan a cabo en la Fiscalía como es el caso de las resoluciones por tipo de decisión. Gestiona además Los planes de inspección a LD y CP. Tiene permisos para ver los reportes municipales.
Fiscal Jefe Provincial	Es el actor encargado de generar el Plan Trimestral Provincial de Inspección a CP.
Reloj del sistema	Es el encargado avisar al sistema que debe empezar la verificación correspondiente a la creación del Plan trimestral de Inspección a CP.

Descripción de actores del sistema.

2.5.2. Diagrama de Casos de Uso

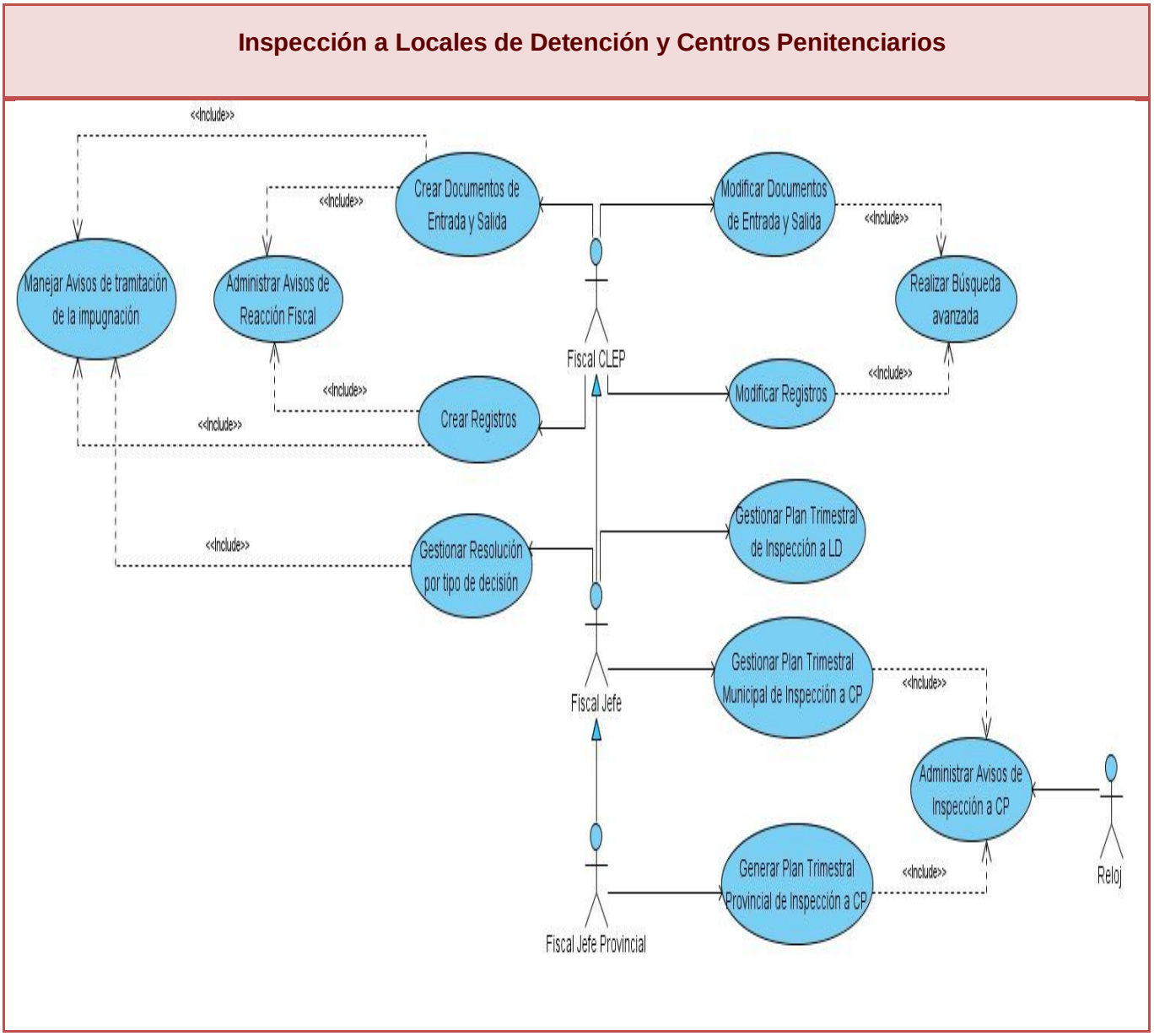


Diagrama de Casos de Uso del sistema.

2.5.3. Descripción Textual de los Casos de Uso del Sistema

Un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Este artefacto contiene actores, casos de uso y sus relaciones y sirve como entrada fundamental para el análisis, diseño y pruebas.

Una descripción detallada de los mismos se puede encontrar en el documento Modelo de Casos de uso del sistema que se adjunta a la investigación. ***(Ver Modelo de Casos de Uso del Sistema).***

Conclusiones

En este capítulo se precisaron las características del sistema a desarrollar, las cuales quedaron especificadas en términos de requisitos funcionales y no funcionales. Se identificaron y describieron los actores y las funcionalidades previstas más importantes y las relaciones establecidas entre ellos. Con el desarrollo de este capítulo quedaron sentadas las bases que darán paso al diseño del módulo Inspección a LD y CP.

3. Capítulo 3 Diseño del sistema

3.1. Introducción

En el presente capítulo se obtendrán los principales artefactos que posibilitaran la implementación del módulo Inspección a LD y CP. El diseño definido por la metodología de desarrollo escogida, viabilizará un mejor entendimiento del módulo por parte de los implementadores. Como parte de esta solución se explican las clases, los patrones utilizados, la arquitectura definida por los arquitectos del proyecto y se desarrollan los diagramas de clases del diseño así como también las descripciones de dichas clases.

3.2. Flujo de Análisis y Diseño

El artefacto modelo del análisis se basa en un modelo de objetos conceptuales. El mismo se genera con el objetivo de lograr un mejor entendimiento de los requisitos y describe sin entrar en detalles la realización de los requisitos funcionales, utilizando un lenguaje más técnico para el desarrollo de esta actividad. Se realiza cuando no se tiene concebida una idea clara y concisa de los procesos y requisitos del sistema.

A pesar de que el análisis ofrece una visión general del sistema, no precisa como se implementa la solución, es decir, no sirve para generar líneas de código porque no se basa en la resolución de los requisitos no funcionales, los cuales si se tienen presente en el desarrollo del diseño.

Después de un estudio perpetrado se decide no realizar el modelo de análisis. Dicha decisión está basada en los siguientes argumentos:

- ✓ El modelo de análisis es solo una aproximación al modelo de diseño.
- ✓ Los procesos y requisitos del sistema están bien definidos.
- ✓ El lenguaje y la plataforma de programación está bien definidos.

La mayoría de sistemas, incluso los sistemas más pequeños, se deben diseñar antes de su implementación a fin de evitar revisiones costosas debido a errores de diseño. Los modelos visuales simplifican la comunicación del diseño. Las herramientas para revertir y aplicar la ingeniería pueden garantizar la coherencia con el modelo de implementación, así como el ahorro de esfuerzo.(Rational, 2003)

El diseño de software OO requiere especificación de una arquitectura de software multicapa, la especificación de subsistemas que realizan funciones necesarias y proveen soporte de infraestructura, una descripción de objetos(clase) que son los bloques de construcción del sistema y una descripción de los mecanismos de que permiten que los datos fluyan entre las capas, subsistemas y objetos. El Diseño Orientado a Objeto cumple todos estos requisitos.(S.Pressman, 2005)

3.2.1. Modelo de Diseño

Un Modelo de Diseño es una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es usado como entrada esencial en las actividades relacionadas a implementación. Representa a los casos de uso en el dominio de la solución. Este Modelo puede contener: los diagramas, las clases, paquetes, subsistemas, relaciones, colaboraciones, atributos, las realizaciones de los casos de uso, entre otros que se puedan considerar para el sistema en desarrollo.(MeRinde, 2010)

3.2.1.1. Arquitectura definida para el Sistema

El flujo de trabajo de análisis y diseño, y la arquitectura están íntimamente relacionados, pues durante el desarrollo de esta actividad se generan tres de las cinco vistas arquitectónicas del sistema. El establecimiento de una buena arquitectura garantiza un mejor desempeño, reuso, robustez, portabilidad, flexibilidad, escalabilidad y mantenibilidad de las aplicaciones. Esta será la encargada de establecer los fundamentos para que el equipo de desarrollo trabaje en una línea común que permita alcanzar los objetivos del sistema, cubriendo todas las necesidades.

En el proyecto SGF se utiliza el estilo arquitectónico Modelo Vista Controlador. Dicho estilo tiene además de facilitar la estandarización y la utilización de los recursos, permite la reutilización y la independencia entre las capas, o sea, en caso de que existe algún cambio solo se verá afectado el nivel requerido.

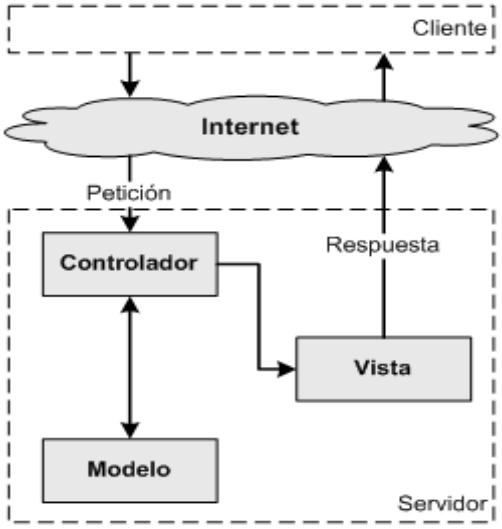
Patrón arquitectónico Modelo Vista Controlador (MVC)

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email).

El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes del tipo de gestor de bases de datos utilizado por la aplicación.

Para el desarrollo del sistema se escogió el framework Symfony, framework que está basado en un patrón clásico del diseño web conocido como arquitectura MVC, formado por 3 niveles:

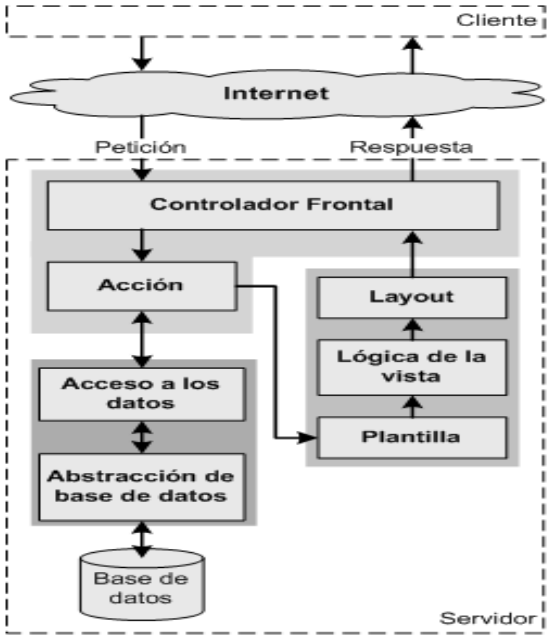
- ✓ El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- ✓ La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- ✓ El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.(Potencier, 2008)



Modelo-Vista-Controlador.

Symfony toma lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo. (Potencier, 2008)

Symfony genera un componente denominado Controlador Frontal que unido con el layout son comunes para todas las páginas de la aplicación. Las clases de la capa del modelo son generadas automáticamente en función de la estructura de datos de la aplicación por la librería Propel y la abstracción de la base de datos es completamente invisible al programador, la misma es realizada mediante un componente llamado Creole.



Flujo de trabajo de Symfony.

Aplicación de patrones en Symfony

Conocer la arquitectura del framework seleccionado, en este caso particular Symfony, permite un mejor entendimiento del funcionamiento y la estructura de las clases contenidas en él. Las clases del modelo y del controlador propuestas en el estilo arquitectónico MVC implementan diferentes patrones, a continuación se explican algunos ejemplos de los utilizados en dichas clases.

- **Patrones GRASP en Symfony**

Patrón Creador

Una de las formas de crear objetos en Symfony es a través de la clase Actions. Por ejemplo cuando se necesita obtener el identificador de un objeto determinado y ver los datos del mismo, debe existir un action donde se cree la instancia de dicho objeto que representará una entidad.

Patrón Experto

Symfony cuenta con la librería Propel, la cual utiliza para realizar su capa de abstracción en el modelo, para esto encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

Por cada tabla de la BD Symfony genera 4 clases: Clase, ClasePeer, BaseClase y BaseClasePeer. Las clases a las que el framework añade el sufijo Peer trabajan directamente con la base de datos y por lo tanto son las encargadas de la abstracción utilizando Propel, en ellas se encuentran los atributos necesarios para este proceso, de ahí la necesidad de que implementen la responsabilidad de efectuar las operaciones con la base de datos, aplicando de esta manera el patrón experto.

Alta Cohesión

En Symfony se definen clases con una alta cohesión, por ejemplo a través de la clase Action se definen acciones para las plantillas, se crean instancias a objetos, se accede a las properties, lo que evidencia que está constituida por varias funcionalidades que se encuentran relacionadas entre sí y proporcionan que el software sea manejable ante grandes cambios.

Patrón Bajo Acoplamiento

Las clases que implementan la lógica de negocio y de acceso a datos se encuentran en el modelo, estas clases no tienen asociaciones con las de la vista o el controlador por lo que la dependencia en este caso es baja, cumpliéndose así perfectamente el patrón**.**

- **Patrones GoF en Symfony**

Patrón Decorador

Symfony define la clase abstracta sfView, la cual contiene un método que tiene incluido un decorador para cada vista, que permite añadirles a estas funcionalidades de forma dinámica.

El archivo llamado layout.php contiene el layout de la página, este archivo se le nombra plantilla global y almacena el código HTML que es común a todas las páginas de la aplicación, el contenido de la plantilla se integra en el layout.

Patrón Singleton

La clase sfRouting se encuentran en la capa Controlador del framework, la misma es muy usada porque es la encargada de enrutar todas las peticiones que se hagan a la aplicación. El ejemplo tiene solamente un punto de creación, el cual es estático lo que permite entonces la aplicación perfecta del patrón.

Patrón Command

Se evidencia en la clase sfFrontWebController, esta clase es la encargada de determinar cuál módulo y acción deben responder a las solicitudes de los usuarios.

3.2.1.2. Descripción de las clases

Fr_Nombre (Formularios): Colección de elementos de entrada que son parte de una página cliente. Se relaciona directamente con la etiqueta de igual nombre del HTML.

Cp_Nombre (Página Cliente): Una instancia de Página Cliente es una página Web, con formato HTML. Mezcla de datos, presentación y lógica. Son interpretadas por el navegador. Sus atributos son las variables declaradas dentro del script que son accesibles para cualquier función dentro de la página.

Sp_Nombre (Página servidora): Representa la página Web que tiene código que se ejecuta en el servidor. Este código interactúa con recursos en el servidor. Las operaciones representan las funciones del código y los atributos las variables visibles dentro del alcance de la página.***(Ver Modelo de Diseño)***

3.2.1.3. Diagrama de secuencia

Los diagramas de secuencia son utilizados para ilustrar la realización de un CU. Presentan vital importancia para los diseñadores porque aclaran los roles jugados por los objetos en un flujo, lo cual le proporciona un gran valor para la determinación de las responsabilidades de las clases. Estos incluyen secuencia cronológica de los mensajes y no la relación entre los objetos, por lo que es mejor su utilización cuando el orden en el tiempo de los mensajes es de importancia. **(*Ver Modelo de Diseño*)**

Conclusiones

En el capítulo se trataron temas relacionados con la arquitectura seleccionada y el diseño del sistema que se presenta. El uso de patrones permitió obtener un diseño flexible y de mayor calidad. Esto facilitará una mejor organización del trabajo de los desarrolladores y un entendimiento más claro del funcionamiento del sistema. La propuesta de modelo de diseño constituirá un plano del modelo de implementación, permitirá además descomponer los trabajos de implementación del módulo en partes más manejables que puedan ser desarrolladas por diferentes equipos de desarrollo.

Las realizaciones de los casos de uso aportarán la información necesaria para el proceso de implementación. Una vez realizado el análisis y diseño del módulo se da paso a la validación de los resultados propuestos.

4. Capítulo 4 Validación de los resultados

4.1. Introducción

En el presente capítulo se realizará la validación de los resultados obtenidos. Se hará uso de métricas para evaluar la especificidad de los requisitos, con el objetivo de probar su bajo nivel de ambigüedad. Se demostrará la calidad que presentan los casos de uso, así como la robustez del diseño del sistema debido a la aplicación de métricas que lo garanticen.

4.2. Validación de requisitos

La validación de requisitos se basa en demostrar que los mismos definen el sistema funcional que el cliente desea, y la importancia del cumplimiento de esta etapa se debe a que muchos de los errores que se pueden encontrar durante la misma podrían significar importantes costos en el desarrollo del proyecto. Dichos errores pueden llevar a repetir el trabajo durante su desarrollo o cuando este ya esté en uso.

De acuerdo con el estándar IEEE 830, se considera que una especificación (SRS) es de “calidad” cuando se puede decir de ella que es "correcta, no-ambigua, completa, consistente, ordenada por importancia y estabilidad, verificable, modificable y trazable."

Las métricas de calidad para la especificación son una técnica que permite medir la calidad de la especificación de requisitos, estableciendo magnitudes y valores que se podrán deducir de la especificación de requisitos y comparar con valores de referencia universal.

En el presente trabajo se realizaron dos revisiones para obtener requisitos fuertes y sin ambigüedades, a continuación se presentan los resultados de la aplicación de las métricas de calidad para la especificación de requisitos:

Revisión 1: En la primera revisión se encontraron dos requisitos que tenían el mismo nombre y descripciones diferentes. Además, los revisores encontraron un requisito que podía ser reestructurado en dos. Por lo que de 69 requisitos tuvieron ambigüedad 3 de ellos.

n_r:Número de requisitos.

n_f: Número de requisitos funcionales (69).

n_{nf}: Número de requisitos no funcionales (34).

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

$$n_r = 69 + 34$$

$$n_r = 103$$

Q_1: Consistencia de la interpretación por los revisores.

N_{u1}: Número de requisitos para la que todos los revisores tuvieron interpretaciones iguales.

$$Q_1 = \frac{N_{u1}}{n_r}$$

$$Q_1 = \frac{100}{103} = 0,97$$

El valor obtenido por Q_1fue de 0.97, el cual es muy cercano al valor 1 lo que comprueba que los requisitos contienen un alto grado de claridad ya que responden a una única interpretación siendo satisfactoria para el desarrollo de la aplicación.

Revisión 2: En esta revisión se solucionaron los problemas de ambigüedad y se reestructuraron los requisitos que fueron señalados, obteniéndose un nuevo requisito. Después de los cambios realizados se cuenta con iguales interpretaciones por parte de los revisores para los 104 requisitos obtenidos.

n_f: Número de requisitos funcionales (70).

n_{nf}: Número de requisitos no funcionales (34).

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

$$n_r = 70 + 34$$

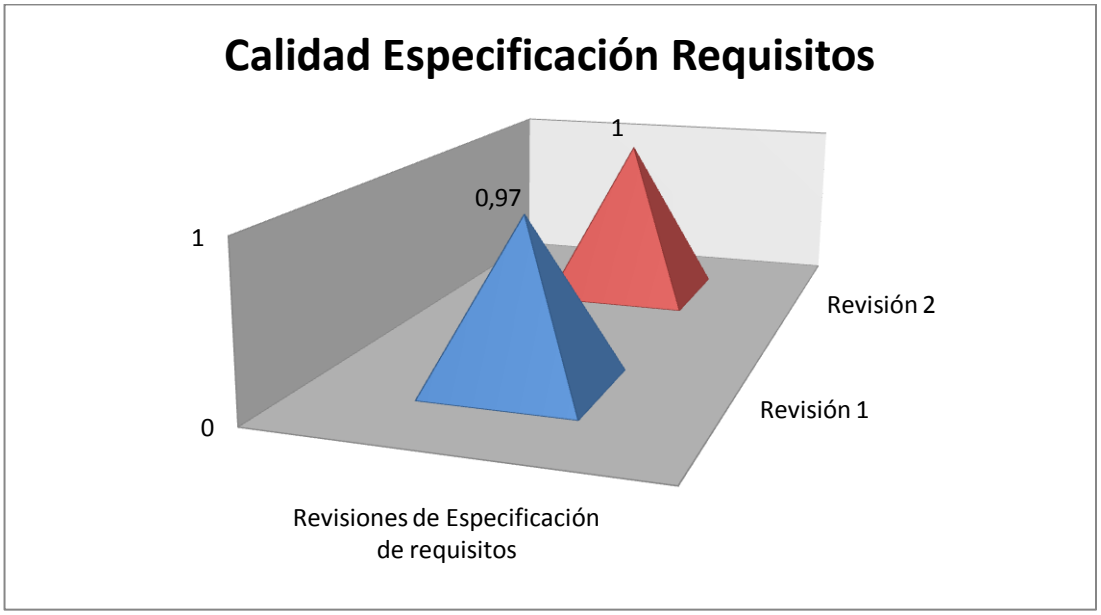
$$n_r = 104$$

Q_1: Consistencia de la interpretación por los revisores.

N_{u1}: Número de requisitos para la que todos los revisores tuvieron interpretaciones iguales.

$$Q_1 = \frac{N_{u1}}{n_r}$$

$$Q_1 = \frac{104}{104} = 1$$



Calidad Especificación de Requisitos.

Resultado

Después de haber analizado las anteriores revisiones se concluye que los requisitos no presentan ambigüedad, pues ya en la segunda revisión se solucionaron todos los problemas presentados en la primera. Los resultados obtenidos son satisfactorios.

Revisor:

Calidad del proyecto Sistema de Gestión Fiscal

Ing. Ginisleisy Morales Gómez

Equipo de Calidad del proyecto SGF

4.3. Validación de casos de uso del sistema

En este acápite se aplica un conjunto de métricas orientadas a objeto para evaluar las siguientes propiedades de calidad: consistencia, correctitud, completitud y complejidad. Cada uno de estos factores tendrá asociada una o más métricas, que establecen una medida cuantitativa del grado de calidad que presentan.

Completitud: Grado en que se ha logrado detallar todos los casos de uso relevantes.

Consistencia*:* Grado en que los casos de uso del sistema describen las interacciones adecuadas entre el usuario y el sistema.

Correctitud*:* Grado en que las interacciones actor / sistema soportan adecuadamente el proceso del negocio.

Complejidad*:* Grado de claridad en la presentación de los elementos que describen el contexto y la claridad del sistema.

Para conseguir una certera validación de los casos de uso del sistema se realizaron dos revisiones, a continuación se muestra la segunda revisión y se presentan los resultados obtenidos:

Atributo	Factor	Métrica asociada	Valor
Completitud	Factor 1. ¿Han sido definidos todos los roles relevantes de usuario encargados de generar, modificar o	Métrica 1: Número de roles relevantes omitidos.	Número de roles relevantes omitidos: 0 Se presenta un 100%.

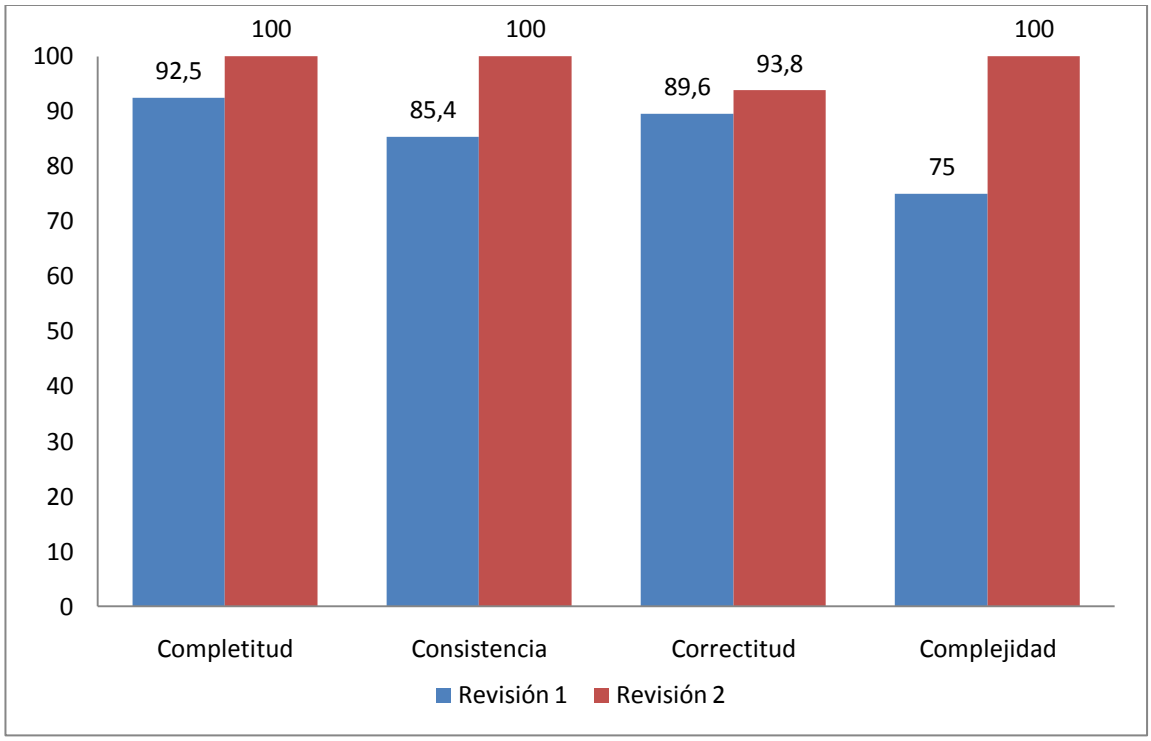
	consultar información?		
	Factor 2. ¿Se presenta una descripción resumida de todos los casos de uso?	Métrica 2. Número de casos de uso que no tiene descripción resumida.	Número de casos de uso que no tiene descripción resumida: 0 Se presenta un 100%.
	Factor 3. ¿Están definidos todos los requisitos que justifican la funcionalidad del caso de uso?	Métrica 3: Número de requisitos omitidos por caso de uso. Métrica 4: Número de casos de uso que tienen requisitos omitidos.	Número de requisitos omitidos por caso de uso: 0 Se presenta un 100%. Número de casos de uso que tienen requisitos omitidos: 0 Se presenta un 100%.
	Factor 4. ¿Todos los casos de uso han sido clasificados de acuerdo a su relevancia en (crítico, secundario, auxiliar, opcional)?	Métrica 5. Número casos de que no han sido clasificados.	Número casos de que no han sido clasificados: 0 Se presenta un 100%.
			Se presenta un: 100%
Consistencia	Factor 5. ¿El nombre dado a los casos de uso es una expresión verbal que describe alguna funcionalidad relevante en el contexto del usuario?	Métrica 6: Número de casos de uso que tienen un nombre incorrecto.	Número de casos de uso que tienen un nombre incorrecto: 0 Se presenta un 100%.

	Factor 6. ¿Está adecuadamente redactado (en el lenguaje del usuario) el flujo de eventos?	Métrica 7: Grado de adecuación de la descripción del flujo de eventos para un caso de uso	La descripción se define en el lenguaje del usuario. Se define el responsable de cada acción: 0 Se presenta un 100%.
	Factor 7. ¿La descripción del flujo de eventos se inicia con la descripción de una acción externa originada por un actor o por una condición interna del sistema claramente identificable?	Métrica 8: Número de casos de uso cuya descripción extendida no inicia con una acción externa o con una condición monitoreada por el sistema.	Número de casos de uso cuya descripción extendida no inicia con una acción externa o con una condición monitoreada por el sistema: 0 Se presenta un 100%.
	Factor 8. ¿Existe una adecuada separación entre el flujo básico de eventos y los flujos alternos y/o flujos subordinados?	Métrica 9: Número de casos de uso complejos que no tienen separación del flujo básico y de flujos alternos.	Número de casos de uso complejos que no tienen separación del flujo básico y de flujos alternos: 0 Se presenta un 100%.
			Se presenta un: 100%
Correctitud	Factor 9. ¿Representa el caso de uso requisitos comprensibles por el usuario?	Métrica 10: Número de casos de uso en que los requisitos representados no son comprensibles por el usuario.	Número de casos de uso en que los requisitos representados no son comprensibles por el usuario: 0 Se presenta un 100%.

	Factor 10. ¿Las interacciones definidas describen la funcionalidad requerida del sistema?	Métrica 11: Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema.	Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema: 0 Se presenta un 100%.
	Factor 11. ¿Se ajusta la representación del diagrama del caso de uso de acuerdo a lo normado en la metodología?	Métrica 12: Grado en que se ajusta el diagrama del caso de uso a la metodología	Grado en que se ajusta el diagrama del caso de uso a la metodología: 3 Se presenta un 75%.
	Factor 12. ¿Las interacciones definidas introducen mejoras al proceso actual?	Métrica 13: Número de casos de uso que deben ser modificados para mejorar el proceso actual.	Número de casos de uso que deben ser modificados para mejorar el proceso actual: 0 Se presenta un 100%.
			Se presenta un: 93.75%
Complejidad	Factor 13. ¿Los elementos dentro del diagrama están adecuadamente ubicados de manera que facilitan su interpretación?	Métrica 14: Número de elementos del diagrama que requieren reubicación.	Número de elementos del diagrama que requieren reubicación: 0 Se presenta un 100%.
			Se presenta un: 100%

Métricas para validar diagrama de caso de uso del sistema.

Gráfica de Factores de Métrica



Factores de Métrica.

Resultados:

Los valores mostrados por todas las métricas aplicadas demuestran que los casos de uso del sistema presentan alta calidad de su funcionalidad, para la demostración se tuvo en cuenta el umbral de cada una de las métricas.

Revisor:

Ing. Yelena Hernández Estrada (Analista Principal del proyecto Sistema de Gestión Fiscal)

4.4. Validación del diseño

Tamaño de clase (TC)

La métrica de tamaño de clase pertenece a la familia de métricas propuesta por Lorens y Kidd.

El tamaño general de una clase puede medirse determinando las siguientes medidas:

- ✓ Total de operaciones (tanto heredadas como privadas de la instancia), que se encapsulan dentro de la clase.
- ✓ El número de atributos (atributos tanto heredados como privados de la instancia), encapsulados por la clase.
- ✓ Promedio general de los dos anteriores para el sistema completo.

Un TC grande afecta los indicadores de calidad definidos para esta métrica por los especialistas:

- ✓ ***Reutilización:*** reduce la reutilización de la clase.
- ✓ ***Implementación:*** complica la implementación.
- ✓ ***Responsabilidad:*** la clase debe tener bastante responsabilidad.

Para la evaluación de las métricas son necesarios los valores de las medidas o umbrales, a continuación se muestran los tipos de umbrales existentes para dicha métrica, los que responderán a la clasificación de la clase.

Clasificación(TC)	Valores de los umbrales
Pequeño	<=20
Medio	>20 y <=30
Grande	>30

Tamaño de clase.

No	Nombre de la Clase	Cantidad de Atributos	Cantidad de Operaciones	Tamaño
1	CC_crear_DocumentosActions	0	21	Mediana

2	CC_crear_RegistrosActions	0	7	Pequeño
3	CC_búsquedaAvanzadaActions	0	14	Pequeño
4	CC_generar_Plan_trimestral_Procincial_InspecciónActions	0	4	Pequeño
5	CC_ gestionar_Plan_Trimestral_InspecciónCPActions	0	5	Pequeño
6	CC_ gestionar_Plan_Trimestral_InspecciónLDActions	0	5	Pequeño
7	CC_ gestionar_Resolución_Tipo_DecisiónActions	0	5	Pequeño
8	CC_ modificarRegistrosActions	0	8	Pequeño
9	CC_ modificarDocumentosActions	0	22	Mediana
10	CE_acta	16	0	Pequeño
11	CE_resolución	15	0	Pequeño
12	CE_documento_Comunicación	6	0	Pequeño
13	CE_documento_Otra_Modalidad	8	0	Pequeño
14	CE_diligencia_Notificación	11	0	Pequeño
15	CE_respuesta_Documento_Otra_Modalidad	4	0	Pequeño
16	CE_plan_Medidas	3	0	Pequeño
17	CE_auto_Libertad	7	0	Pequeño

18	CE_estudio_Estado_Legalidad	5	0	Pequeño
19	CE_plan_Objetivos	8	0	Pequeño
20	CE_registroLD	14	0	Pequeño
21	CE_registroCP	16	0	Pequeño
22	CE_registroResoluciones	11	0	Pequeño
23	CE_plan_Trimestra_ProvincialCP	3	0	Pequeño
24	CE_plan_Trimestral_InspecciónLD	10	0	Pequeño
25	CE_ plan_Trimestral_InspecciónLD	10	0	Pequeño
26	CE_resolución_Tipo_Decisión	26	0	Mediana

Cantidad Clases	Promedio Atributos	Promedio Operaciones
26	6.7	3.5

Resultados:

La mayoría de las clases que conforman la aplicación están dentro de la categoría de pequeñas, solamente tres tienen umbral medio lo que demuestra que el sistema no es complejo, quedando demostrada la calidad del diseño y que los indicadores de calidad reutilización, implementación y responsabilidad no se ven afectados.

Árbol de profundidad de herencia (APH)

La métrica Árbol de Profundidad de Herencia fue propuesta por Chidamber y Kemerer (CK). Esta métrica se define como la máxima longitud del nodo a la raíz del árbol. Donde el nodo es una clase hija que

hereda de una clase, y así respectivamente hasta llegar a la raíz. A medida que esa longitud va creciendo, entonces se van heredando más operaciones y atributos por las clases hijas. Esto conlleva a dificultades potenciales cuando se intenta predecir el comportamiento de una clase. Si los valores de APH son grandes, entonces se garantiza que se reutilice gran cantidad de código, pero al mismo tiempo hace que el diseño sea más complejo. Esto provoca un mayor acoplamiento entre las clases.

Resultados:

En este sistema no se presenta un alto nivel de jerarquía de herencia, al aplicar la métrica APH se obtuvo que el nivel más alto de herencia entre las clases del diseño es de: 2 niveles de profundidad, lo que demuestra que el diseño no es complejo y existe bajo acoplamiento en el diseño y es de fácil reparación.

Conclusiones

En el presente capítulo se aplicaron métricas para evaluar la especificación de los requisitos, el modelo del sistema y el diseño realizado. Reflejando como resultados que los requisitos presentan un bajo nivel de ambigüedad, los casos de uso del sistema presentan alta calidad de su funcionalidad y el diseño presenta un bajo acoplamiento y una gran robustez, lo que garantiza una mayor confiabilidad al implementar el sistema.

Conclusiones Generales

- Después de haber profundizado en el estudio de los sistemas de gestión jurídica existentes en Cuba y el mundo se demostró la necesidad del desarrollo del módulo Inspección a LD y CP, para realizar la gestión documental de forma automática y de este modo agilizar el trabajo en las fiscalías.
- Se realizó un estudio del estado del arte de las herramientas, metodologías, lenguajes y framework así como los patrones y métricas, que permitió la preparación teórica que sustenta el desarrollo del presente trabajo de diploma. La selección de RUP, como metodología de desarrollo, proporcionó una guía para ordenar las diferentes actividades a realizar. Los lenguajes de modelado elegidos posibilitaron la creación de diagramas en la representación visual de los artefactos mediante la herramienta Visual Paradigm.
- Con el uso de diferentes técnicas para la captura de requisitos, se identificó un conjunto de requisitos donde se recogieron las necesidades del personal del departamento CLEP. Estas necesidades fueron traducidas a un conjunto de Casos de Uso que describen las principales funcionalidades del sistema.
- El uso de diferentes patrones en el modelado del sistema y el diseño, permitió generar artefactos haciendo uso de buenas prácticas para lograr un desarrollo robusto.
- Se evaluó la calidad de los principales artefactos generados haciendo uso de métricas, lo que permitió confirmar la realización de artefactos confiables y con calidad.

Recomendaciones

Se recomienda realizar la implementación del módulo Inspección a Locales de Detención y Centros Penitenciarios, dada la propuesta presentada en este trabajo de diploma, con el fin de obtener una versión del producto.

Referencias bibliográficas

Amoroso, Fernández Yarina, Sociedad de la Información: Contribución de la Informática Jurídica. Revista de Derecho Informático. Julio del 2002, No 0.48. [Fecha de consulta: 15 febrero 2010]. Disponible en: http://www.alfa-redi.org/rdi-articulo.shtml?x=1483. ISSN: 1681-5726.

Bernádez, Beatriz. 2004. *UNA PROPUESTA PARA LA VERIFICACIÓN DE REQUISITOS BASADA EN MÉTRICAS*. Sevilla : Asociación Española de Sistemas de Informáticos, 10 de 2004, Revista de Procesos y Métricas de las Tecnologías de la Información (RPM), Vol. 1, págs. 13-14. 1698-2029. Deadalus. 2010. Deadalus. [En línea] 2010. [Citado el: 2010 de 12 de 12.] http://www.daedalus.es/inteligencia-de-negocio/sistemas-complejos/ingenieria-de-sistemas/diseno-de-sistemas/.

Eckel's, Bruce. 2003.*Thinking in Patterns with Java.* 2da. 2003.

Gunnar Overgaard, Karin Palmkvist. 2004.*Use Cases: Patterns and Blueprints.* s.l. : Addison Wesley, 2004. pág. 464 . 0131451340.

Gutierrez, Jorge A. Saavedra. 2007. El Mundo Informático. *El Mundo Informático.* [En línea] 8 de 5 de 2007. http://jorgesaavedra.wordpress.com/category/patrones-grasp/.

Jacobson, Ivar, Booch, Grady, Rumbaugh, James. 2000.*El proceso unificado de desarrollo de software.* Madrid : Pearson Education, 2000.

Jimenez, Roman. 2009. Sistemas de Software 2. *Sistemas de Software 2.* [En línea] 25 de 2 de 2009. http://sitemasdesoftware2.blogspot.com/2009/02/unidad-2-proceso-del-software-y.html.

Larman, Craig. 2002.*UML y Patrones. Una introducción al análisis orientado a objetos y al proceso unificado.* 2da. Madrid : Pearson Education, 2002.

María José Escalona, Nora Koch. 2002.*Ingeniería de Requisitos en Aplicaciones para la Web –Un estudio comparativo.* Departamento de Lenguajes y Sistemas Informáticos, Escuela Técnica Superior de Ingeniería Informática. Universidad de Sevilla. Sevilla : s.n., 2002.

MeRinde. *MeRinde.* [En línea] 2010. http://merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=137&Itemid=192.

Molina, Jesús García. 2007. 04. De los Procesos del Negocio a los Casos de Uso, Buenos Aires: s.n., 2007, Vol. 06. ISSN 1666-1680.

Ocaña, Sanchez. 2003.*Diseño orientado a objetos y software estadísticos: patrones de diseño,UML y composición v/s herencia.* Barcelona : s.n., 2003.

Potencier, Fabien. 2008.*Symfony la guía definitiva.* 2008.

Rational, Software Corporation. 2003. Ayuda del Rational (Español). *Ayuda del Rational (Español).* [En línea] 2003. [Citado el: 10 de 2 de 2009.]

S.Pressman, Roger. 2005.*Ingeniería de Software. Un enfoque práctico.* 6ta. s.l. : Mcgraw-hill, 2005.

Silva, Víctor Manuel Talavera. 2008.*LA FIRMA DIGITAL.* Buenos Aires : s.n., 2008.

Zapata , Carlos Mario, Palacio, Carolina, Olaya, Natalí. 2007. *UNC-ANALISTA: HACIA LA CAPTURA DE UN CORPUS DE REQUISITOS*. Medellín (Colombia) : s.n., Junio de 2007, Revista EIA, págs. 25-40. ISSN 1794-1237.

Zubyc. 2009. POO en PHP5. *POO en PHP5.* [En línea] 10 de 10 de 2009. [Citado el: 20 de 02 de 2010.] http://www.php-hispano.net/novedades/1252616380-ayuda-para-haitiacute.html.

Bibliografía

Álvarez Romero, Eduardo y Pueyo. Daniel. *Integration Definition For Function Modeling* (IDEF0), 2004

Amoroso Fernandez, Yarina. 2002. *Sociedad de la Información: Contribución de la Informática Jurídica*, AR: Revista de Derecho Informático, Editorial Alfa-Redi, 2002.

Beck, Kent y Otros. *Agile Manifesto*. http: //agilemanifesto.org/, 2001. [En línea] 2009

BerlinchesCerezo, Andrés. 2004. *Calidad.* s.l. :Cengage Learning Editores, 2004. ISBN: 8497320832.

Boehm, Barry W. 1981. *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981

Castillo, Carlos. *PHP: Nociones Básicas.* 2005.

Cendrós G, Jesús; Durante, Carlos y José Fermín. *Factores estratégicos para desarrollar el gobierno electrónico en las Alcaldías de Venezuela*. Revista de Ciencias Humanas y Sociales v.20 n.45 Maracaibo dic. 2004

Deadalus. Deadalus. [En línea] 2010. [Citado el: 2010 de 12 de 12.] http://www.daedalus.es/inteligencia-de-negocio/sistemas-complejos/ingenieria-de-sistemas/diseno-de-sistemas/.

Diego Pérez, Juan. *Notaciones y lenguajes de procesos*. Una visión global.

Durán Toro, Amador y Bernárdez Jiménez, Beatriz. 2000. *Metodología para la Elicitación de Requisitos de Sistemas Software.* [En línea] Octubre de 2000.
[Citado el: 07 de Marzo de 2008.] www.lsi.us.es/~informes/lsi-2000-10.pdf. 92

EAFIT, 2007. *Modelo de métricas Orientado a Objetos (OO) de la universidad EAFIT de Medellín*. Medellín, Colombia.

Eckel, *Bruce.* *Thinking in C++.*Volumen 2.Mindview, Inc, 2003. [En línea] Enero de 2009. http://arco.esi.uclm.es/~david.villa/pensar_en_C++/products/vol2/index.html

Escalona, María José y Koch, Nora. 2002. *Ingeniería de Requisitos en Aplicaciones para la Web – Un estudio comparativo.* Departamento de Lenguajes y Sistemas Informáticos. [En línea] Diciembre de 2002. [Citado el: 17 de Marzo de 2008.] http://www.lsi.us.es/docs/informes/LSI-2002-4.pdf.

FGR, Portal de la Fiscalía General de la República. [En línea] 2008. [Citado el: 1 de diciembre de 2008.] http://www.fgr.cu/

Gamma, Erich, y otros. 1995. *Design Patterns. Elements of Reusable Object-Oriented Software.*s.l. : Addison Wesley, 1995. ISBN:0201633612.

Gause, D. C. y Weinberg, G. M. 1989. *Exploring Requirements: Quality Before Design. Dorset House,* 1989.

González Estrada, Joel. *Desarrollo web con PHP y MySQL.*

Grupo Soluciones Innova S.A. [En línea] 2009. [Citado el: 19 de enero de 2009.] http://www.rational.com.ar/herramientas/roseenterprise.html (19/01/09)

Gutierrez, Jorge A. Saavedra. El Mundo Informático. *El Mundo Informático.* [En línea] 8 de 5 de 2007. http://jorgesaavedra.wordpress.com/category/patrones-grasp/.

Hoffer, J; George, J. y Valacich, J. 1996. *Modern Systems Analysis and Design.*Benjamin/Cummings Publishing Company, inc. 1996. 93

IBM OOTC. 1997. *Developing Object Oriented Software.* IBM Object Oriented Technology Center. Prentice-Hall.

Institute of Electrical and Electronics Engineers.*IEEE Standard 610. Computer dictionary.Compilation of IEEE Standard Computer Glossaries*, IEEE, 1990.

IEEE. 1993. *Standards Collection: Software Engineering.* s.l. : IEEE Standard 610.12-1990, 1993.

IEEE.*Guide to the Software Engineering Body of Knowledge*. 2004 Version SWEBOK.

Jacobson, Ivar; Booch, Grady y Rumbaugh, James.2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000.

Jimenez, Roman. Sistemas de Software 2. *Sistemas de Software 2.* [En línea] 25 de 2 de 2009. http://sitemasdesoftware2.blogspot.com/2009/02/unidad-2-proceso-del-software-y.html.

Kendall, K. 1997. *Analisis y diseño de sistemas.* s.l. : Prentice Hall., 1997.

Larman, Craig.*UML y Patrones. Una introducción al análisis orientado a objetos y al proceso unificado.* 2da. Madrid : Pearson Education, 2002.

Larman, Craig. 2003. *Agile and Iterative Development: A Manager's Guide.* s.l. : Addison-Wesley, 2003. ISBN:0131111558.

Lowe, D. y Hall, W. 1999.*Hypermedia and the Web.*An Engineering approach.John Wiley & Son.

Mendoza Sánchez, María A. *Metodologías De Desarrollo De Software.* (Junio -2004) Menéndez, R. y B. Asensio. 2005. *Metodologías de desarrollo de software*, 2005. [En línea] Enero de 2009. http://www.um.es/docencia/barzana

Molina, Jesús García.*De los Procesos del Negocio a los Casos de Uso***.**04, Buenos Aires : s.n., 2007, Vol. 06. ISSN 1666-1680.

MeRinde. *MeRinde*. [Enlínea]2010.

http://merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=137&Itemid=192.

Ocaña, Sanchez.*Diseño orientado a objetos y software estadísticos: patrones de diseño,UML y composición v/s herencia.* Barcelona : s.n., 2003.

Olmedilla Arregui, Juan José. 2005. *Revisión Sistemática de Métricas de Diseño Orientado a Objetos.* Universidad Politécnica de Madrid, Facultad.deInformática : s.n., 2005.

OMG. *Business ProcessModelingNotationSpecification*. OMG Final AdoptedSpecification, 2006. Övergaard, Gunnar y Palmkvist, Karin. 2004. *Use Cases Patterns and Blueprints.* s.l.: Addison Wesley Professional, 2004.

Pan, D.; Zhu, D. y Johnson, K. *Requirements Engineering Techniques, Internal Report.*Department of Computer Science.University of Calgary. Canada.

Pérez, M. 1999. *Arquitectura para Ambientes CASE Integrados.* Tesis Doctoral. UCV. 1999.

PHP. En línea] 2009. [Citado el: 5 de marzo de 2009.] http://www.php.net

Pohl, Klaus; Haumer, Peter y Weidenhaupt, Klaus. 1998. *Requirements Elicitation and Validation with Real World Scenes*. IEEE PressPiscataway, NJ, USA, 1998. Potencier, Fabien y Zaninoto, Francois. 2007. *Symfony la guía definitiva.* 21 de octubre del 2007.
Potencier, Fabien.*Symfony la guía definitiva.* 2008.

Pressman, Roger S. 2005. Ingeniería de Software. Un enfoque Práctico, McGraw-Hill, 5ta edición, 2005.

Raghavan, S.; Zelesnik y Ford, G. 1994.*Lectures Notes of Requirements Elicitacitation*. EducationalMaterials.

Rational, Software Corporation. Ayuda del Rational (Español).*Ayuda del Rational (Español).* [En línea] 2003. [Citado el: 10 de 2 de 20]

Ramos González, Juan José.2004. *PML-A modeling Language for Physical Knowledge Representation.*Barcelona : Universidad Autonoma de Barcelona, 2004.

Rational, Software Corporation. 2003. *Ayuda del Rational UnifiedProcess.* 2003. 95

Silva, Víctor Manuel Talavera.*LA FIRMA DIGITAL.* Buenos Aires : s.n., 2008.

Sommerville, I y Sawyer, P. 1997. Requirements Engineering: A Good Practice Guide, John Wiley & Sons, Inc. New York, NY, USA, 1997

Sommerville, Ian. 2000. *Requirements Engineering An Overview.* 2000.

Sommerville, I. 2005. Ingenieríadel Software (7th Edition), Addison Wesley, 2005 Sparxsystems, Empresa. 2009. [En línea] 2009. [Citado el: 19 de enero de 2009.] http://www.sparxsystems.com.ar/products/ea_features.html.

Thayer, R y Dorfman, M. 1990. *Standards, Guidelines and Examples on System and Software Requirements Engineering.*IEEE ComputerSocietyPress, 1990.

UNC-ANALISTA: HACIA LA CAPTURA DE UN CORPUS DE REQUISITOS. Carlos Mario Zapata, Carolina Palacio,Natalí Olaya. 7, Medellín (Colombia) : s.n., Junio de 2007, Revista EIA, págs. 25-40. ISSN 1794-1237.

Zubyc. POO en PHP5. *POO en PHP5.* [En línea] 10 de 10 de 2009. [Citado el: 20 de 02 de 2010.] http://www.php-hispano.net/novedades/1252616380-ayuda-para-haitiacute.html.

Anexo

Acta de Liberación de Artefactos

UCI

Informáticas

Acta de Liberación de Artefactos, Grupo de Calidad Centro CEGEL de la Facultad 15 de la Universidad de las Ciencias Informáticas.

Jueves, 03 de junio de 2010.

Luego de haber efectuado 3 iteraciones de revisiones a los artefactos: Especificación de Requisitos Funcionales, Modelo de Negocio, Modelo de Sistema y Modelo de diseño del módulo Inspección a Locales de Detención y Centros Penitenciarios del proyecto Sistema de Gestión Fiscal del Centro CEGEL de la Facultad 15 y haberse detectado un promedio de 5 No Conformidades, se puede afirmar que se han corregido los defectos encontrados, por lo que quedan liberados los artefactos.

Firma del Asesor y Jefe del Grupo de Calidad Centro CEGEL

Ing. Raúl Velázquez Alvarez

Declarato Facultad