Universidad de las Ciencias Informáticas
Facultad 15

Título: *Definición de actividades para el rol de Diseñador de Aplicaciones en el proyecto SUA.*

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

Autor: Yanlay Gómez Bandera

Tutor (es): Ing. Raymond Weeden Gamboa

Ciudad Habana, junio del 2010

Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio, conscientes de recibir el premio de la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte.

Ché

Declaración de Autoría

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

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

_____

Firma del Autor

_____

Firma del Tutor

Agradecimientos

A mi mamá por ayudarme a ser quien soy, por apoyarme en los momentos difíciles que me ha tocado vivir, por darme una vida llena de amor y comprensión, por ser tan luchadora y por ser la mejor de las madres.

A mi hermanita la chungui, que la quiero muchísimo.

A mis abuelas, por estar siempre a mi lado y cocinando como a mi me gusta.

A mi papa, por tenerme en su Corazón.

A toda mi familia en general por creer en mí.

A mi novio Yasiel, por darme siempre ánimo y amor.

A mi suegrita Nercy, por preocuparse tanto por mí.

A mi tutor que siempre nos apoyó, gracias por su dedicación y entrega.

A todas mis amistades que a lo largo de estos años hemos compartido alegrías y momentos difíciles, pero siempre nos hemos mantenido juntos y como dice el buen cubano tirando pa'lante y apoyándome cuando lo he necesitado. Lianet, liyelis, Loreta, Mariesly, Ismary, Yobalis, Yadira, Jeanne, el primo, Eliza, Misael,

Gracias a todas las personas que de una forma u otra contribuyeron a mi formación durante todos estos años y a la realización de mi tesis.

Dedicatoria

Quiero dedicarle este logro tan importante para mí.

A mi mamá, esta victoria es tuya también, a quien más he de dedicarle, si yo he sido el fruto de tus manos.

A mi papá, mi hermana la chungui, mi novio y a toda mi familia

A todas aquellas personas que me han ayudado a salir adelante, y han hecho posible la realización de este sueño.

Resumen

La Aduana General de la República es un órgano de Administración Central del Estado subordinado al consejo de Estado y de Ministro, cuya misión es garantizar el cumplimiento de la política estatal para el tráfico internacional de medios de transporte, mercancías y viajeros. El proyecto Sistema Único de Aduana (SUA) del Centro CEIGE es un proyecto en conjunto a la Aduana General de la República de Cuba y la UCI.

Durante varios años de desarrollo de aplicaciones para automatizar sistemas aduanales, la dirección del proyecto ha detectado dificultades en la formación de estudiantes en el rol de Diseñador de Aplicaciones. Es por esto la necesidad de crear una propuesta que tiene como objetivo definir las actividades a realizar por el diseñador de aplicaciones.

El presente es un trabajo que consiste en la realización de una guía metodológica orientada al mejoramiento del diseño en este proyecto, además incluye un importante análisis relacionado con el correcto empleo de patrones, la evaluación de las métricas del software, la estructura que debe tener un equipo de diseño, como también la propuesta de roles de trabajos, además de los artefactos generados en el proceso de diseño y una conceptualización enfocada en dimensiones de diseño del software desde el punto de vista del marco tecnológico.

También en este trabajo se muestra como se realiza el diseño en las diferentes metodologías y en diferentes proyectos de la universidad, además de mostrar procesos de diseño con altos niveles de calidad como lo es la reutilización del diseño, los mecanismos de persistencia, robustez y fiabilidad.

ÍNDICE

Introducción

La actividad de diseñar en la vida del hombre tiene una gran importancia por ser un proceso previo de configuración mental. A través de este, se proyectan ideas reflejadas en elementos gráficos sobre diferentes tipos de soporte como el digital, el analógico o el virtual y se orienta dándole solución a problemas que existen en la realidad.

El diseño en la industria del software, puede verse de diferentes tipos y a través de diferentes métodos como el diseño arquitectónico, diseño de la interfaz de usuario, diseño a nivel de componentes y el diseño orientado a objetos. Sin embargo, podrían citarse mucho más métodos o categorías de diseño dentro del propio proceso de desarrollo de software. Una vez que se analicen y especifiquen los requisitos del software, el diseño es la primera de las tres actividades técnicas (diseño, generación de código y pruebas) que necesitan para hacer y verificar el software. Cada actividad transforma la información de manera, que al finalizar de lugar a un software de computación validado.

Existen muchos otros problemas en el proceso de diseñar un software de computadora, cuya principal causa está determinada por el poco o mal uso de patrones de diseño que en muchos casos no son tomados en cuenta para estructurar el diseño del software, inventando aspectos previamente definidos en patrones.

Existen dos tipos de metodologías, las cuales están clasificadas en dos grandes grupos: las ágiles o livianas y las pesadas o tradicionales, teniendo esta una gran diferencia, entre otras muchas cosas, reflejada en el modo de concebir el diseño...

La mayoría de los proyectos de software que se realizan en todo el mundo, se guían por alguna metodología en particular. Y el proyecto Sistema Único de Aduana de la Universidad de la Ciencias e Informática no está exento de utilizar alguna metodología, sin embargo, no cuenta con una guía metodológica de buenas prácticas de diseño que permita elevar la calidad.

Por eso es que Durante varios años de desarrollo de aplicaciones para la automatización de los procesos aduanales, la dirección del polo ha detectado dificultades en la formación de estudiantes en el rol de diseñador de aplicaciones, a pesar de ser uno de los roles más importantes la formación se ha centrado en la tecnología sin tener en cuenta las actividades que deberían llevar a cabo en la práctica.

Partiendo de esto el presente trabajo de diploma se encargara de dar solución al siguiente **Problema:** ¿Que actividades debe desarrollar el diseñador de aplicaciones en el proyecto Sistema Único de Aduana?

El **Objetivos General** que se persigue en este trabajo es aumentar los resultados en la formación del rol de diseñador de aplicaciones en el proyecto SUA a partir de la identificación de las actividades a realizar por el mismo.

El **Objeto de Estudio** está enmarcado en el proceso de diseño de aplicaciones en el proyecto SUA y como **Campo de Acción** se tiene las actividades a realizar en el diseño de aplicaciones en el proyecto SUA.

Objetivos Específicos

- Describir las principales actividades a realizar por el diseñador de aplicaciones.
- Elaborar y desarrollar propuestas para realizar un buen diseño.
- Definir un procedimiento para realizar un buen diseño.
- Elaborar un inventario de funciones y responsabilidades que debe realizar el diseñador de aplicaciones.

Las principales **tareas** que se realizan en este trabajo son**:**

1. Estudiar procedimientos de diseños en diferentes metodologías.
2. Estudiar métodos de documentación de diseño.
3. Establecer las actividades a realizar por el diseñador de aplicaciones en el proyecto SUA.
4. Definir las funciones y responsabilidades a cumplir por el diseñador de aplicaciones en el proyecto SUA.

Entre los **Posibles resultados** se encuentran**:**

1. Modelo de procesos para el rol de diseñador de aplicaciones.
2. Inventario de funciones.
3. Inventario de responsabilidades.

Métodos Científicos de Investigación:

Se llevaron a cabo diferentes métodos científicos para realizar la investigación. Como método teórico se utilizó el **Analítico-Sintético** pues uno de los primeros pasos para llevar a cabo la presente tesis es estudiar y analizar documentos y extraer los elementos más importantes relacionados con el objeto de estudio. Otro método teórico que se aborda en la investigación es el método **Histórico-Lógico** para analizar a nivel internacional y nacional como se encuentra actualmente el proceso de diseño de software, así como investigaciones realizadas anteriormente sobre el tema.

Dentro de los métodos **Empíricos** se usa la Entrevista, para de esta forma recoger toda la información referente a las necesidades actuales y perspectivas que se deben cumplir.

Capítulo 1: Es la introducción al trabajo y donde se realiza un estudio de las mejores prácticas empleadas en las diferentes metodologías.

Capítulo 2: Este capítulo está dedicado a establecer las actividades a realizar por el diseñador de aplicaciones en el proyecto SUA y también mostraremos como realizan el diseño en los diferentes proyectos de la Universidad de la Ciencia e Informática. Y por ultimo y no menos importante es la propuesta de las actividades a realizar por el Diseñador de Aplicaciones en el proyecto SUA.

Capitulo 3: Este capítulo estará dedicado a la validación de la propuesta de las actividades.

Capítulo 1

1 Capítulo 1"Fundamentación Teórica"

Introducción

En este capítulo se muestra el estado en que se encuentra actualmente el proceso de diseño de software en el mundo y un estudio sobre las mejores prácticas en los procedimientos de diseño en las diferentes metodologías.

Se realiza además un análisis y evaluación de las principales problemáticas que hoy afectan el diseño de software en el proyecto Sistema Único de Aduana (SUA) .Y Un análisis del empleo de patrones, incluyendo el impacto del mal uso de estos en el proceso de desarrollo de software.

1.1 ¿Qué es el diseño de software?

El diseño de software es una representación ingenieril significativa de algo que se puede construir, en este caso: un software. Se puede hacer el seguimiento de este proceso basándose en los requerimientos del cliente y al mismo tiempo, evaluar la calidad del mismo partiendo de los criterios predefinidos con el objetivo de alcanzar un buen diseño. Estando este ubicado en el mismo núcleo del proceso de desarrollo de software y constituye la primera actividad técnica de las tres que se realizan en el proceso de ingeniería de software (**PRESSMAN, 2001**).

Las etapas en las cuales el diseño se desarrolla están estrechamente ligadas a los pasos generales o niveles de detalles que rigen el proceso de diseño: la estructuración de los elementos correspondientes al nivel de datos; la arquitectura del sistema; la representación del diseño de la interfaz y finalmente, los detalles a nivel de componentes. En cada uno de estos niveles se aplica los conceptos y principios básicos, así como patrones de diseño, que llevan a la obtención de un diseño con una alta calidad.

El diseño de software en un proceso que tributa al proceso de desarrollo de sistemas para transformar los requisitos de los usuarios o clientes en un producto de software finalizado. Este en un proceso iterativo mediante el cual los requisitos se llevan a un plano, el que estará guiando el proceso de implementación del software.

En realidad, cualquier cosa asociada a la estructura e implementación de un programa esta relacionada directamente con el diseño .Esto incluye la estructura del programa, la naturaleza del lenguaje que esté usando, el uso de los símbolos de separación en el código fuente, etc. Si bien es cierto que todos estos elementos tienen relación estrecha con el diseño, este se desarrolla en mayor o menor grado de detalle en dependencia del contexto en que se vea (**HUMPHREY, 2001**).

La importancia del diseño radica en la necesidad de hacer un software más estable, con mucha más calidad. Precisamente es en el diseño donde se fomentara la calidad del producto de software final, y es donde se crearan las representaciones del software que se podrán evaluar en cuanto a calidad se refiere.

En principios, el diseño en el ámbito de la ingeniería software como disciplina, encierra cuatro etapas fundamentales: datos, arquitectura, interfaces y componentes.

Diseño de Datos

En este apartado se deberá realizar una clara definición de la información y de la relación lógica entre los elementos individuales de los datos. La estructura de datos establece las alternativas de organización, métodos de acceso, capacidad de asociación y procesamiento de la información. El diseño de datos es una de las actividades más importantes, datos bien diseñados conducen a una mejor estructura y modularidad del programa, y a una menor complejidad procedimental.

Diseño Arquitectónico

Es la estructura de programas que observará el sistema, resultado de una serie de refinamientos del modelo funcional basados en el análisis del flujo de transformación y transacción de datos. Es una descripción de la estructura modular del sistema y sus interfaces. El objetivo del diseño arquitectónico es desarrollar una estructura de programa modular y representar las relaciones de intercambio de información entre los módulos. El diseño

arquitectónico deberá combinar la estructura del programa con las estructuras de datos, definiendo interfaces que faciliten el flujo de datos a través del programa.

Diseño de Interfaz

La interfaz debe reflejar tres aspectos: el flujo de datos entre los módulos, el flujo de datos a entidades externas y al diseño de interfaz con el usuario.

El diseño de interfaz se refiere a la comunicación de datos de entrada y salida que realiza el sistema o programa en tres aspectos que se detallan en los puntos que integran este apartado. Una base guía para el diseño de interfaz interna y externa es el modelo funcional, ya que en él se plasma como van transformándose los datos entre las funciones del sistema.

Diseño Procedimental

En este apartado se deberá definir la especificación detallada del procesamiento y/o funcionamiento de cada programa. En el diseño procedimental se deberá expresar de una manera general el funcionamiento de cada programa, sin entrar en detalles de la manera en que se implementará en una herramienta de desarrollo particular en la fase de construcción de componentes. Con este propósito se podrán utilizar los tres tipos de construcciones lógicas con las que puede formarse cualquier programa:

- Secuencia. Instrucciones de procesamiento esenciales en un algoritmo.
- Condición. Selección del orden de procesamiento basándose en una condición lógica.
- Repetición. Ejecución de bloques de instrucciones en ciclos.

1.2 Estado Actual del diseño de software

En la actualidad el concepto de diseño de software está orientado a diferentes enfoques dentro del propio proceso de desarrollo del sistema informático, sin embargo, existe una convergencia en cuanto a que el diseño es el que dirige y lleva a cabo transformación de los requisitos del cliente en una implementación de software.

Lo real del asunto es que no existe un método o receta que dirija este proceso de manera infalible. **Grady Booch** plantea*:"el diseño es el proceso de determinar una implantación efectiva y eficiente que realice las funciones y tenga la información del análisis de dominio",* mientras que Pressman presentaba al diseño como la primera de las tres actividades técnicas del proceso de desarrollo de software como lo es el diseño, la generalización de código y

pruebas. Otra idea a la que se refiere **Pressman** es que la importancia del diseño del software se puede describir como una sola palabra: calidad (**Pressman, 2001**).

El diseño es el lugar en donde se fomentan la calidad en la ingeniería del software. Este proporciona las representaciones del software que se pueden evaluar en cuanto a calidad, sirve como funcionamiento para todos los pasos siguientes del soporte del software y de la ingeniería del software, ya que sin no se hace el diseño se está corriendo el riesgo de hacer un producto inestable, un sistema que cuando se lleven a cabo los cambios falle, un sistema que sea muy difícil de comprobar.

El diseño del software, al igual que los enfoques de diseño de ingeniería en otras disciplinas, va cambiando continuamente a medida que se desarrollan métodos nuevos, análisis mejores y se amplía el conocimiento. Las metodologías de diseño del software carecen de la profundidad, flexibilidad y naturaleza cuantitativa que se asocian normalmente a las disciplinas de diseño de ingeniería más clásica. Sin embargo, si existen métodos para el diseño del software; también se dispone de calidad de diseño y se pueden aplicar notaciones de diseño.

El proceso de diseño aparece bien explicado en todas las metodologías de desarrollo de software en alguna medida.

1.3 El Proceso de Diseño Orientado a Objetos (DOO)

El término diseño tiene disímiles aceptaciones de acuerdo con el contexto en que se pretenda definir. Hace referencia a la actividad de representar gráficamente una indicación de sentido o dirección, ya sea soporte analógico, digital o virtual, y en dos o más dimensiones. Es el proceso previo de configuración (pre-figuración) mental en la búsqueda de una solución en cualquier campo. Sin embargo, diseñar, en el ámbito de la ingeniaría de software es más que hacer representaciones gráficas sobre cualquiera de los ya antes mencionados soportes.

La actividad de diseñar, en cualquiera de los casos, debe partir de la acción de las siguientes funciones que a continuación se listan:

1. Primero parte de la **observación y análisis** del medio en el cual el ser humano se desenvuelve con el objetivo de descubrir alguna necesidad de acuerdo con sus funciones en dicho entorno.

2. Una vez detectada la necesidad, se procede a la **plantación y/o proyección** en el que se propone un modo de solucionar la necesidad, mediante, modelos o maquetas que recreen de manera visual las posibles soluciones para viabilizar el análisis de los que sería óptimo hacer.

El diseño orientado a objeto (DOO) cumple una serie de requisitos que se concretan en la necesidad de la existencia de una arquitectura de software, en la que se especifiquen subsistemas que realicen funciones y provean soporte de infraestructura de objetos (clases), los que constituyen los bloques de construcción del sistema; así como una descripción de los mecanismos de comunicación que permitan que los datos fluyan.

La principal virtud que tiene el diseño orientado a objeto radica en su capacidad de definir y desarrollar cuatro conceptos que hacen atractivo y más robusto a los sistemas orientados a objetos. Estos conceptos, entre otros, son la abstracción, ocultamiento de la información, modularidad e independencia funcional. Ellos son alcanzados únicamente usando un diseño orientado a objetos, siendo prohibido por este de un modo sencillo y sin compromiso. Muy a pesar de que todos los métodos tratan de exhibir estas características, solo el diseño orientado a objeto brinda el mecanismo capaz de desarrollarlas al máximo.

Pressman define cuatros capas del proceso de diseño orientado a objetos, reflejadas en lo que el nombro la pirámide del diseño orientado a objetos (**PRESSMAN, 2001**) como se muestra en la siguiente **figura 1.2.6.1**.

Figura 1.2.6.1 La Pirámide del Diseño Orientado a Objeto (DOO) **(PRESSMAN, 2001).**

La capa subsistema. Contiene una representación de cada uno de los subsistemas, para permitir a un software conseguir sus requisitos definidos por el cliente e implementar la infraestructura que soporte los requerimientos del cliente.

La capa de clases y objetos. Contiene la jerarquía de clases, que permiten al sistema ser creado usando generalizaciones y cada vez especializaciones más acertadas. Esta capa también contiene representaciones.

La capa de mensajes. Contiene detalles de diseño, que permite a cada objeto comunicarse con sus colaboradores. Esta capa establece interfaces externas e internas para el sistema.

La capa de responsabilidades. Contiene estructuras de datos y diseños, algorítmicos, para todos los atributos y operaciones de cada objeto.

En resumen, el diseño orientado a objeto es un proceso que se aplica únicamente a sistemas orientados a objetos, brindando un mecanismo sencillo para desarrollo de conceptos como abstracción, el ocultamiento de la información, modularidad e independencia funcional. Este a su vez, se centra en definir los procedimientos que dan "vida" a la arquitectura del software en construcción.

1.4 Procedimientos de diseño en las diferentes Metodología

El diseño surge en uno de los principales procesos dentro del ciclo de vida del desarrollo de un software, es cierto que el proceso de diseñar en el desarrollo del software no es más que hacer un sistema con los requisitos puestos por el cliente, este proceso en si no está desligado de otros como lo es el proceso de análisis o el de implementación.

Pressman define el modelo de diseño partiendo de la conversión de los aspectos del modelo de análisis y estructura el flujo de información de este de la siguiente manera (**PRESSMAN, 2001**):

- Diseño de Datos
- Diseño Arquitectónico
- Diseño de interfaz
- Diseño a nivel de componentes

No existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software. Toda metodología debe ser adaptada al contexto del proyecto (recursos técnicos y humanos, tiempo de desarrollo, tipo de sistema, etc. Históricamente, las metodologías tradicionales han intentado abordar la mayor cantidad de situaciones de contexto del proyecto, exigiendo un esfuerzo considerable para ser adaptadas, sobre todo en proyectos pequeños y con requisitos muy cambiantes. Las metodologías ágiles ofrecen una solución casi a medida para una gran cantidad de proyectos que tienen estas características.

Una de las cualidades más destacables en una metodología ágil es su sencillez, tanto en su aprendizaje como en su aplicación, reduciéndose así los costos de implantación en un equipo de desarrollo. Esto ha llevado hacia un interés creciente en las metodologías ágiles.

Sin embargo, hay que tener presente una serie de inconvenientes y restricciones para su aplicación, tales como:

- Están dirigidas a equipos pequeños o medianos (Beck sugiere que el tamaño de los equipos se limite de 3 a 20 como máximo otros dicen no más de 10 participantes).
- El entorno físico debe ser un ambiente que permita la comunicación y colaboración entre todos los miembros del equipo durante todo el tiempo.
- Cualquier resistencia del cliente o del equipo de desarrollo hacia las prácticas y principios puede llevar al proceso al fracaso (el clima de trabajo, la colaboración y la relación contractual son claves), el uso de tecnologías que no tengan un ciclo rápido de realimentación o que no soporten fácilmente el cambio, etc.

1.4.1 El diseño según Rational Unified Process (RUP)

RUP plantea un conjunto de flujos de trabajo dentro del proceso de desarrollo de software, en el que el diseño queda enmarcado de manera acentuada en el flujo de trabajo **Análisis** y **Diseño** ver **figura 1.3.2.1.1.**

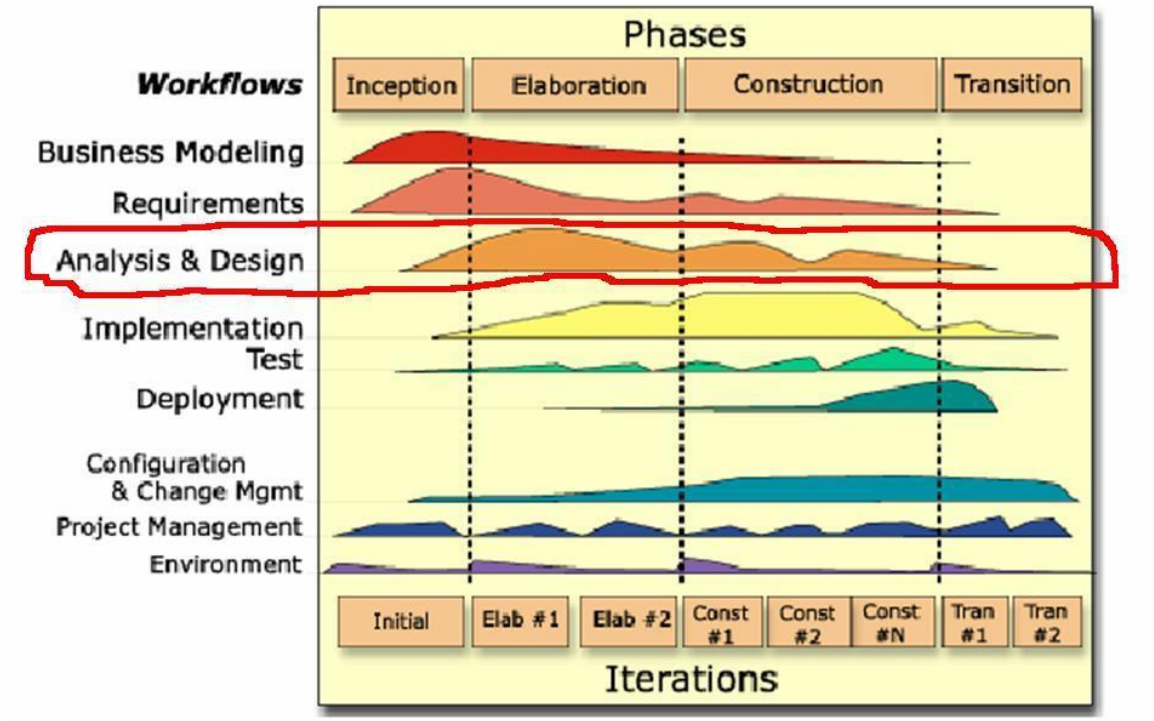


Fig. 1.3.2.1.1 Flujos de trabajos y fases de desarrollo de RUP.

De acuerdo con los principios de RUP como metodología el diseño de software tiene diferentes funciones esenciales (**Jacobson, y otros, 2000**):

1. Tener una comprensión con profundidad de los requisitos funcionales y no funcionales obtenidos de fases anteriores al diseño, así como de las diferentes tecnologías propuestas, las restricciones de los lenguajes de programación etc.
2. Definir elementos que sirvan para la actividad de implementación subsiguiente.
3. Descomponer la implementación en partes menos complejas como subsistemas, módulos, componentes de clases, etc.
4. Capturar las interfaces antes del ciclo de vida del software.
5. Crear una abstracción de la implementación del sistema sin caer en especificidades, en el sentido de que la implementación es el refinamiento del diseño que se encarga de rellenar lo que ya está diseñado sin cambiar la estructura.

1.4.2 El diseño según SCRUM

Scrum, más que una metodología de desarrollo software, es una forma de auto-gestión de los equipos de programadores. Un grupo de programadores deciden cómo hacer sus tareas y

cuánto van a tardar en ello. Scrum ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro. Además, permite además seguir de forma clara el avance de las tareas a realizar, de forma que los "jefes" puedan ver día a día cómo progresa el trabajo.

Scrum es una metodología ágil de desarrollo de software con un alto grado de solapamiento entre las actividades provocando que no existan fases de desarrollo si no tareas que se asignan y se ejecutan cuando sea necesario (**Palacio Bañares, 2007**).

Esta metodología incita a la participación colectiva de todo el equipo en el diseño como manera de aportar más conocimientos, y también estimula la práctica del intercambio de conocimiento entre equipos auto organizado. Es una metodología que esta orientada a la gestión de proyectos de software, en el cual el diseño se construye de manera evolutiva debido a la inestabilidad del entorno y de los requisitos.

Scrum es una metodología de desarrollo muy simple, que requiere trabajo duro, porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto.

Las prácticas empleadas por Scrum para mantener un control ágil en el proyecto son:

- Revisión de las iteraciones
- Desarrollo incremental
- Desarrollo evolutivo
- Auto-organización del equipo
- Colaboración

Los valores que hacen posible a las prácticas de Scrum crear "campos de Scrum" son:

- Autonomía del equipo
- Respeto en el equipo
- Responsabilidad y auto-disciplina
- Foco en la tarea
- Información transparencia y visibilidad

1.4.3 El diseño según XP (Extreme Programmnig)

XP presenta dentro de sus prácticas la definición de un diseño simple capaz de ser implementado en un momento determinado del proyecto. Lo curioso es que de acuerdo con la propuesta original de Beck para la especificación de los roles que participan en el proceso de desarrollo según XP, no existe un diseñador de software como tal, por lo que se asigna esta responsabilidad al propio rol del programador, el que además debe escribir las pruebas unitarias y producir el código del sistema. Para este caso surge la programación en pareja, la actividad de diseñar lo que se implementa, recae sobre la pareja de programadores que desarrollen dicha implementación (Canos, y otros 2003).

Las parejas de programadores se intercambian con frecuencia, de forma que todos acaban trabajando con todos. El trabajo por parejas haciendo intercambios tiene las siguientes ventajas:

- Cuatro ojos ven más que dos. Al trabajar de dos en dos, el código será de mayor calidad desde el mismo momento de crearlo y tendrá menos fallos.
- Los programadores novatos aprenderán de los expertos al emparejarse con ellos.
- Si una pareja realiza un trozo de código susceptible de ser reutilizado en el proyecto, hay dos programadores que lo saben y que lo reutilizarán cuando puedan (ya que saben cómo funciona), enseñándolo a sus nuevos compañeros. De esta manera, el conocimiento del código ya hecho se propaga de forma natural entre todos los programadores del equipo.
- El estilo de programación tiende a unificarse.

1.4.4 El diseño según Crystal

La familia de metodologías Crystal ofrece diferentes métodos para seleccionar el más apropiado para cada proyecto. Crystal identifica con colores diferentes cada método, y su elección debe ser consecuencia del tamaño y criticidad del proyecto, de forma que los de mayor tamaño, o aquellos en los que la presencia de errores o desbordamiento de agendas implique consecuencias graves, deben adoptar metodologías más pesadas. Los métodos Crystal no prescriben prácticas concretas, y se pueden combinar con técnicas como XP.

El tamaño del proyecto indica el método a utilizar, estableciéndose una clasificación por colores, por ejemplo Cristal Limpio (3 a 8 personas), seguido por Amarillo (10 a 20 personas),

Crystal Anaranjado (25 a 50 personas), y así sucesivamente hasta el marrón ver **figura 1.3.2.4.1**, mientras que la importancia indica la dureza con que se debe aplicar, esta va de cuarzo hasta diamante. A diferencia de otras metodologías que se basan en los procesos, arquitectura o incluso en herramientas, podríamos decir que Crystal es una típica metodología "basada en las personas".

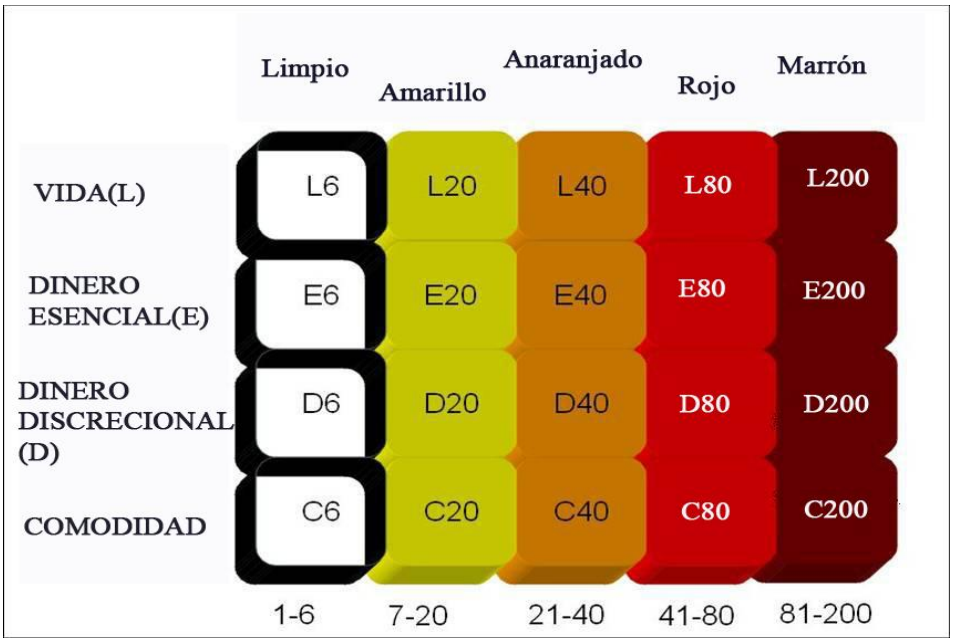


Fig. 1.3.2.4.1 Familia de Métodos de Cristal

Las metodologías de Crystal se basan en el principio de que tipos diferentes de proyectos requieren tipos diferentes de metodologías. La metodología escogida debe depender de dos factores:

- El número de personas en el proyecto, y
- Las consecuencias de los errores.

Existen muchos software basados en metodologías cristal las cuales Integran estrechamente capacidades de diseño, modificación y visualización en aplicaciones .NET, Java o COM. También Permitir a los usuarios finales acceder e interactuar con los reportes a través de portales Web, dispositivos móviles y documentos de Microsoft Office. De esta forma, podemos darnos cuenta que la aplicación de estas metodologías son extremadamente recomendables en el buen desarrollo de proyectos de software.

1.5 Principales causas de problemas de diseño de software

Los problemas que a menudo se cometen en el proceso de desarrollo de diseño de software, están asociados a los atributos de calidad del propio proceso en cuestión. Sin embargo, son precisamente los problemas del diseño los que tipifican los atributos de calidad del mismo y no estos los que determinan los problemas de diseño; es decir, dan una medida de que es lo que se debe medir y cuáles son los niveles aceptables de la solución al problema.

La poca flexibilidad y extensibilidad, el alto acoplamiento y la baja cohesión son algunos de estos, pero no los únicos. La tendencia que existe hacia el rediseño como solución a las consecuencias de cambios en los requisitos, así como la rapidez, la fragilidad, la inmovilidad y la viscosidad del diseño, son algunos de otros problemas importantes, siendo estos últimos síntomas de que se ha realizado un mal diseño.

Los rediseños generalmente no funcionan. A pesar de que el diseño no pierde su filosofía, el sistema va evolucionando con el tiempo y nuevamente vuelven a cometerse nuevos errores y el diseño nunca termina (**Gracia, 2004**).

Existen varias causas que determinan estos problemas.

1. La poca o nula comprensión de lo que debe hacer el sistema, como por ejemplo alguna función matemática no comprendida o él ignora alguna condición del sistema, provocaría un diseño erróneo (**HUMPHREY, 2001**).
2. Conociendo lo que se debe hacer, se comete un simple error de descuido en cuanto a la completitud de la funcionalidad de mejorar; ejemplo la no incorporación de todos los casos a evaluar en alguna sentencia de casos, provocaría un diseño incorrecto.
3. Tergiversar lo que se quiere hacer, tanto si se ha interpretado mal el diseño de alto nivel o si no se ha entendido los requisitos del sistema, provocando un diseño incorrecto aunque para los efectos el diseño estaría acorde a lo que se entendió o se creyó entender.
4. Concebir de manera correcta el diseño pero realizar una mala representación del mismo, es decir, el diseño adecuadamente concebido pero pobremente representado.

A lo largo de todo el proceso del diseño, la calidad de la evolución del diseño se evalúa con una serie de revisiones técnicas;

Tres características que sirven como guía para la evaluación de un buen diseño:

- El diseño deberá implementar todos los requisitos explícitos del modelo de análisis, y deberán ajustarse a todos los requisitos implícitos que desea el cliente.
- El diseño deberá ser una guía legible y comprensible para aquellos que generan código y para aquellos que comprueban y consecuentemente, dan soporte al software.
- El diseño deberá proporcionar una imagen completa del software, enfrentándose a los dominios de comportamiento, funcionales y de datos desde una perspectiva de implementación.

Con el fin de evaluar la calidad de una representación de diseño, deberán establecerse los criterios técnicos para un buen diseño.

1. Un diseño deberá presentar una estructura arquitectónica que se haya creado mediante patrones de diseño reconocibles, que esté formada por componentes que exhiban características de buen diseño y que se puedan implementar de manera evolutiva, facilitando así la implementación y la comprobación.
2. Un diseño deberá ser modular; esto es, el software deberá dividirse lógicamente en elementos que realicen funciones y subfunciones específicas.
3. Un diseño deberá contener distintas representaciones de datos, arquitectura, interfaces y componentes (módulos).
4. Un diseño deberá conducir a estructuras de datos adecuadas para los objetos que se van a implementar y que procedan de patrones de datos reconocibles.
5. Un diseño deberá conducir a componentes que presenten características funcionales independientes.
6. Un diseño deberá conducir a interfaces que reduzcan la complejidad de las conexiones entre los módulos y con el entorno externo.
7. Un diseño deberá derivarse mediante un método repetitivo y controlado por la información obtenida durante el análisis de los requisitos del software.

Estos criterios no se consiguen por casualidad. El proceso de diseño del software fomenta el buen diseño a través de la aplicación de principios de diseño fundamentales, de metodología sistemática y de una revisión cuidadosa.

1.5.1 Problemas más Importantes de la Calidad del Diseño de Software

La calidad del diseño se realiza durante todo el ciclo de vida del proceso y mediante un conjunto de tareas que se concretan en la realización de un enfoque de gestión de la calidad; el empleo de tecnologías de ingeniería de software efectivas usando métodos y herramientas para ellos; revisiones técnicas formales que se aplican durante todo el proceso; una estrategia de prueba multiescalada; el control de la documentación del diseño, que incluye además la de los cambios del diseño, que incluye además la de los cambios realizados; un procedimiento que asegure un ajuste y buen uso de los estándares internacionales y mecanismos de medición y generación de informes.

Uno de los principales problemas en la calidad del diseño es que hay una mala interpretación de cómo obtener una mejor calidad. Primero que existe una tendencia a enfocarse en la reducción de tiempo y costes de desarrollo cuando en realidad debiera ser en sentido contrario, centrarse en la calidad para poder alcanzar una verdadera reducción a largo plazo de costes y tiempo de desarrollo. Precisamente este enfoque o mejor dicho, este mal enfoque se le aplican a muchos proyectos, provocando una incidencia negativa en los atributos de calidad del diseño.

Por otra parte, están las revisiones técnicas formales. Estás constituyen un mecanismo esencial para la detección de errores, cuyo principal objetivo es la no propagación de estos errores a una etapa siguiente en el desarrollo del software, evitando de esta manera que se conviertan en defectos luego de entregado el producto. Estudios realizados por la Nippon Eléctrica y Mitre Corp. Entre otros, plantean que de un 50 a un 65 por ciento de errores (y en última instancia, los defectos), son introducidos por la actividad de diseñar. Por esta razón es preciso la realización de revisiones formales y más aun cuando su efectividad oscila entre un 70 y un 100 por ciento en la detección de errores.

Aunque en un momento posterior se tratara el tema del uso de los patrones de diseño, es preciso mencionar que otro de los problemas en la calidad de un diseño, es el uso inadecuado y muchos casos nulos, de estos patrones.

Por último, la violación del principio de correspondencia con los requerimientos del sistema hace que el diseño este expuesto constantemente a cambios, en principios innecesarios,

cayendo en lo que se conoce como rediseño. Evaluar dicha correspondencia desde inicios del proceso, por cualquiera de los mecanismos antes mencionados, evita que se caiga en tan ineficiencia práctica.

1.5.2 Atributos del Proceso de Diseño

El proceso de diseño tiene implícito un grupo de atributos que hacen más atractivo y/o eficiente el diseño, tal y como se ha venido mencionando anteriormente. Estos atributos deben ser medidos empleando métricas de software (que serán vistas en momentos posteriores) y pueden especificarse, entre otros más, los que a continuación se muestran:

- Modularidad.
- Cohesión.
- Acoplamiento.
- Adaptabilidad del diseño a diferentes escenarios y bajo disímiles condiciones.
- Mantenibilidad (Facilidad de mantenimiento).
- Reutilización.
- Abstracción (Nivel de abstracción en el diseño de software).
- Flexibilidad del diseño de software.

1.5.2.1 Modularidad

Se dice que la modularidad es el único atributo de software que permite gestionar un programa en forma de intelectual (**PRESSMAN, 2001**), en tanto Booch la define como “la propiedad que tiene un sistema que ha sido descompuesto en un conjunto de módulos cohesivos y débilmente acoplados” (**BOOCH, 1998**).

En cuanto al diseño se refiere, esta tiene que ver con la capacidad del mismo de separar en elementos (ya sean subsistemas, paquetes, componentes, etc.) tratados y nombrados por separados y que establecen algún mecanismo de integración para satisfacer los requerimientos del problema.

1.5.2.2 Cohesión

La cohesión es una extensión natural del término ocultamiento de la información. En cuanto al diseño de software, se puede decir que un módulo es cohesivo, si ha sido diseñado (idealmente) para realizar una sola función dentro de un procedimiento de software y que

requiere de poca interacción con otros procedimientos que se llevan a cabo dentro del mismo sistema pero en otros módulos (**PRESSMAN, 2001**).
Lo real es que un diseñador de software no deberá preocuparse por la categorización de la cohesión en un módulo en específico, sino que deberá entender el concepto en sentido general del diseño.

La escala de medición del nivel de cohesión en un diseño no es ideal. Lo que quiere decir que un nivel de bajo de cohesión no será mejor que un nivel intermedio y este a su vez, no mejor que un alto nivel, muy a pesar de que el rango medio es tan "aceptable" como la parte más alta de la escala

Un diseño fácilmente adaptable, reduce el tiempo de planificación, los costes de desarrollo, la usabilidad o reutilización del diseño, entre otros aspectos.

1.5.2.3 Facilidad de mantenimiento

La facilidad de mantenimiento es una cualidad que debe tener todo buen diseño, si embargo esta no se obtiene porque sí, sino como consecuencia de un buen uso de mecanismos que favorezcan los conceptos del diseño como la modularidad, un correcto uso de la abstracción, una alta cohesión y un bajo acoplamiento, hacen del diseño un modelo cada vez más entendible y por tanto, fácil de mantener.

El mantenimiento del software cuenta con más esfuerzo que cualquier otra actividad de ingeniería del software. La facilidad de mantenimiento es la habilidad con la que se puede corregir un programa si se encuentra un error, se puede adaptar si su entorno cambia u optimizar si el cliente desea un cambio de requisitos. No hay forma de medir directamente la facilidad de mantenimiento; por consiguiente, se deben utilizar medidas indirectas.

Una métrica orientada al tiempo simple es el tiempo medio de cambio (TMC), es decir, el tiempo que se tarda en analizar la petición de cambio, en diseñar una modificación apropiada, en efectuar el cambio, en probarlo y en distribuir el cambio a todos los usuarios. En promedio, los programas que son más fáciles de mantener tendrán un TMC más bajo (para tipos equivalentes de cambios) que los programas que son más difíciles de mantener.

Hitachi ha empleado una métrica orientada al costo (precio) para la capacidad de mantenimiento, llamada “desperdicios”. El costo estará en corregir defectos hallados después de haber distribuido el software a sus usuarios finales. Cuando la proporción de desperdicios en el costo global del proyecto se simboliza como una función del tiempo, es aquí donde el administrador logra determinar si la facilidad de mantenimiento del software producido por una organización de desarrollo está mejorando y así mismo se pueden emprender acciones a partir de las conclusiones obtenidas de esa información.

1.5.2.4 Reutilización

Es evidente que lograr que un determinado diseño de software que se ajuste a diferentes contextos y bajo diferentes condiciones, es una labor de marcado esfuerzo. De manera que la reusabilidad (o reutilización, como en otros casos es conocido) es una característica que debe cumplir todo buen diseño.

La reutilización en el diseño puede verse desde cualquiera de los niveles definidos por este, ya sea desde un diseño de alto nivel (arquitectura), como la reutilización de algún framework; a un nivel de objetos y clases; a nivel de componentes o desde la propia interfaz de usuario.

La reutilización de componentes constituye una muy buena práctica debido a que el futuro del software depende en gran medida de la capacidad de reutilizar módulos, estructuras de datos, unidades funcionales o abstracciones específicas.

1.5.2.4 Nivel de Abstracción

La abstracción es un mecanismo muy asociado al de generalización y que permite la descripción de un problema empleando conceptos que son familiares en el entorno de dicho problema y si tener en consideración detalles irrelevantes de bajo nivel.

La IEEE define la abstracción como “el proceso de ocultamiento de la información que permite que elementos diferentes sean tratados de la misma manera”. En tanto contextualizaba este mismo concepto en el diseño de software definiéndole dos últimas categorías de abstracción e identificaba tres niveles de abstracción para el proceso de diseño del software como:

- Nivel de abstracción de Datos.
- Nivel de abstracción de Procedimiento.
- Nivel de abstracción de Control.

La abstracción puede ser vista desde dos niveles de generalización, partiendo de un análisis del proceso de diseño propiamente dicho. Se podría definir un nivel de abstracción asociado al macro diseño –diseño de alto nivel (arquitectura) y un nivel de abstracción asociado al micro diseño.

A raíz de la existencia de diferentes niveles de abstracción, es valido aclarar que no deberían mezclarse elementos de diferentes niveles de abstracción debido a que podría generar confusión en el diseño.

1.5.2.5 Flexibilidad

La flexibilidad en un término que comparte cierta analogía con la facilidad de mantenimiento y la adaptabilidad, y como estos, es una consecuencia de un correcto uso de conceptos como el acoplamiento, la cohesión, la modularidad y la abstracción, además de un correcto empleo de mecanismos como los patrones de diseño.

La flexibilidad en el diseño facilita la mantenibilidad en el sentido de la no propagación de cambios, pero como es lógico aumenta la complejidad del diseño, los costes de implementación y los costes de mantenimiento en dependencia de la complejidad del cambio, pero estos últimos aspectos son un efecto normal de introducir más flexibilidad en un diseño a cualquier nivel.

Está asociada a la capacidad del diseño de cambiar sin provocar grandes "traumas" o transformaciones en el resto del modelo. La flexibilidad tributa a la facilidad de mantenimiento en cierta medida, sin embargo, respecto a esto solo un inconveniente, a mayor flexibilidad menor es la entendibilidad y la analizabilidad del diseño, por lo que para este caso el coste de mantenibilidad es un poco mayor.

1.5.3 Empleo de Patrones en el proceso de Diseño de Software

Utilizar algún patrón de diseño en la construcción de un software sin analizar su efectividad y consecuencias, no garantiza que se obtenga un producto con una alta calidad. No es usar patrones de diseño, sino usar aquellos que sean los más adecuados. De manera que es importante señalar que los patrones que vayan a utilizarse en la definición del diseño del sistema, deberán ser analizados para evitar incorrectas selecciones.

Debido a la misma signatura con que se define un patrón, es posible conocer si este puede o no responder a la solución de determinado problema en específico ya que el patrón mismo le describe, por lo que se puede inferir, que nunca se debería seleccionar un patrón con el fin de resolver un problema para el cual no ha sido creado. Además de esto, se impone verificar que las consecuencias para el uso de determinado patrón no constituye una afectación significativa en la construcción del sistema. O sea, que el impacto negativo que cause el patrón aplicando a determinado atributo de calidad no sea significativo o no conlleve a la no correspondencia con los requisitos del sistema en construcción.

Los patrones no constituyen una teoría y mucho menos un lenguaje de programación, sencillamente son la experiencia obtenida en el proceso de desarrollo de software, probada y que funciona realmente. Usar patrones de diseño no es una acción trivial dentro de todo el proceso de desarrollo de software. Se requiere de un profundo análisis para poder identificar el patrón correcto.

Los patrones de diseño tienen una fuerte relación con los atributos de calidad. De hecho, puede afirmarse que estos en muchos casos son tipificados por los propios atributos de calidad del diseño.

Sobre el empleo de patrones existe otro elemento de modular importancia: el coste. Evidentemente, el empleo de patrones imprime un mayor nivel de flexibilidad al diseño. Esto puede verse reflejado en la introducción de clases de servicios en un diagrama de clases que resuelve determinado problema, que permite hacer de este diseño un modelo más extensible y flexible.
Partiendo de lo anterior planteado, se puede llegar a la conclusión de que aplicar patrones podría sacrificar en cierta medida el análisis del diseño. Es además significativo señalar que el coste de implementación inicial aumenta según se complica el diseño.

1.8 Documentación de Diseño

La Especificación del diseño aborda diferentes aspectos del modelo de diseño y se completa a medida que el diseñador refina su propia representación del software.
En primer lugar, se describe el ámbito global del esfuerzo realizado en el diseño. La mayor parte de la información que se presenta aquí se deriva de la especificación del sistema y del modelo de análisis (Especificación de los requisitos del software).

El diseño arquitectónico indica como se ha derivado la arquitectura del programa del modelo de análisis. Además, para representar la jerarquía del módulo se utilizan gráficos de estructuras.

Se representa el diseño de interfaces internas y externas de programas y se describe un diseño detallado de la interfaz hombre-maquina.

Los componentes de software que se pueden tratar por separado tales como subrutinas, funciones o procedimientos se describen inicialmente con una narrativa de procesamiento en cualquier idioma (castellano, Ingles). La narrativa de procesamiento explica la función procedimental de un componente (módulo).

La especificación del diseño contiene una referencia cruzada de requisitos. El propósito de esta referencia cruzada (normalmente representada como una matriz simple) es: establecer que todos los requisitos se satisfagan mediante el diseño del software, e indicar cuáles son los componentes críticos para la implementación de requisitos específicos.

El primer paso en el desarrollo de la documentación de pruebas también se encuentra dentro del documento del diseño. Una vez que se han establecido las interfaces y la estructura de programa podremos desarrollar las líneas generales para comprobar los módulos individuales y la integración de todo el paquete. En algunos casos, esta sección se podrá borrar de la especificación del diseño.
En el capítulo 3 se describe como están diseñados los artefactos relacionados al procedimiento de diseño que estamos realizando.

¿Que es un Caso de Uso (CU)?
Los Casos de Uso se han convertido en la técnica más utilizada a nivel mundial para el levantamiento y la comunicación clara y eficiente de los requisitos (mejor conocidos como “requerimientos”) para el desarrollo de sistemas.

1.9 Métricas del Proceso de Diseño

La medición es fundamental para cualquier disciplina de ingeniería, y la ingeniería del software no es una excepción. La medición nos permite tener una visión más profunda proporcionando un mecanismo para la evaluación objetiva.

Las métricas del software se refieren a un amplio elenco de mediciones para el software de computadora. La medición se puede aplicar al proceso del software con el intento de mejorarlo sobre una base continua. Se puede utilizar en el proyecto del software para ayudar en la estimación, el control de calidad, la evaluación de productividad y el control de proyectos.

Las métricas constituyen un mecanismo esencial para la toma de decisiones dentro de alguna organización, en función del mejoramiento de los procesos que se llevan a cabo dentro de esta, asociado a estimación de tiempos, planificación y demás: o dirigido al mejoramiento de la calidad de los productos que se obtienen dentro de dicha organización.

Las métricas del proceso se recopilan de todos los proyectos y durante un largo período de tiempo. Su intento es proporcionar indicadores que lleven a mejoras de los procesos de software a largo plazo. Un *indicador* es una métrica o una combinación de métricas que proporcionan una visión profunda del proceso del software, del proyecto de software o del producto en sí.

La medición del proceso implica las mediciones de las actividades relacionadas con el software siendo algunos de sus atributos típicos el esfuerzo, el coste y los defectos encontrados. Las métricas permiten tener una visión profunda del proceso de software que ayudará a tomar decisiones más fundamentadas, ayudan a analizar el trabajo desarrollado, conocer si se ha mejorado o no con respecto a proyectos anteriores, ayudan a detectar áreas con problemas para poder remediarlos a tiempo y a realizar mejores estimaciones.

Para mejorar un proceso se deben medir los atributos del mismo, desarrollar métricas de acuerdo con estos atributos y utilizarlas para proporcionar indicadores que conduzcan la mejora del proceso. Los errores detectados antes de la entrega del software, la productividad, recursos y tiempo consumido y ajuste con la planificación son algunos de los resultados que pueden medirse en el proceso, así como las tareas específicas de la ingeniería del software.

Las métricas del proceso se caracterizan por:

- El control y ejecución del proyecto.
- Medición de tiempos del análisis, diseño, implementación, implantación y postimplantación.

- Medición de las pruebas (errores, cubrimiento, resultado en número de defectos y número de éxito).
- Medición de la transformación o evolución del producto.

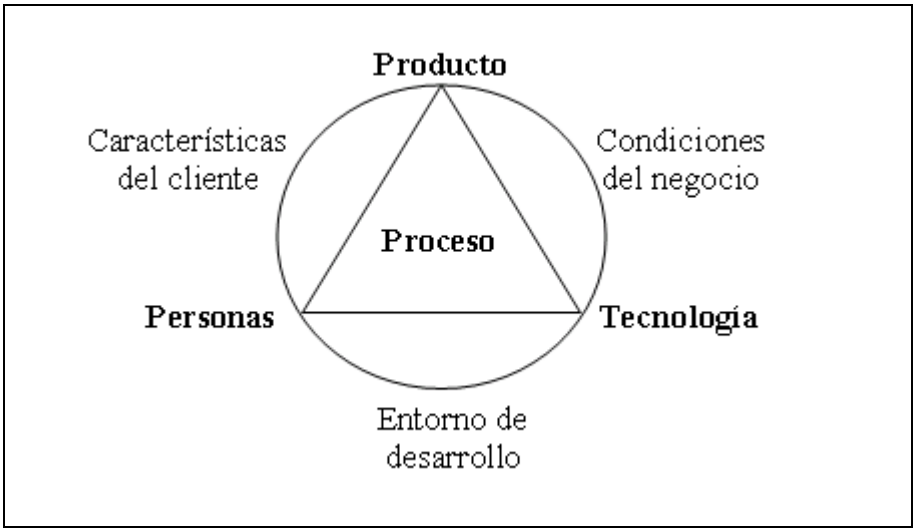


Figura 1.9.1 Determinantes de la calidad del software y de la efectividad de la organización.

Conclusiones

Hasta este entonces, se ha realizado un estudio sobre lo que es el diseño de software, su evolución y la manera en que es concebido por diferentes metodologías de desarrollo de software no solo en Cuba sino también en el mundo. Se llevó a cabo igualmente un análisis de estas metodologías de desarrollo de software que son utilizadas en la actualidad por los desarrolladores de sistemas para lograr productos de alta calidad e incluso otros autores de autorizado criterio en el mundo de la industria de sistemas informáticos.

Partiendo de esto, se definió en el presente trabajo una contextualización del proceso de diseño, debido a la carencia en otros conceptos sobre la especificación de donde se enmarca dicho proceso. De la misma manera en que se define un conjunto de perspectivas de diseño, con el objetivo de facilitar el análisis y desarrollo de esta importante actividad dentro del ciclo de vida.

Capítulo 2

2 Capítulo 2 "Solución Propuesta"

Introducción

Hasta este momento se han establecido las bases teóricas y los diferentes análisis sobre el proceso de diseño de software en las diferentes metodologías. Primero se estará haciendo referencia especialmente al procedimiento de diseño que utiliza actualmente el proyecto Sistema único de Aduana y en otros proyectos que se realizan en la Universidad de las Ciencias e Informática.

Este es el capítulo más importante ya que en él es donde se propone las actividades a realizar por el diseñador de aplicaciones en el proyecto SUA, junto con esto los documentos generados por cada actividad, los roles que realizan esas actividades y el procedimiento que deben de llevar a cabo para la realización de un buen diseño.

2.1 Las Dimensiones del Diseño de Software

Observar el proceso de diseño en dimensiones, permite ampliar el abanico temático sobre los elementos a tener en cuenta para desarrollar un buen diseño. Es decir, es preciso dejar de ver al diseño solo como la simple actividad de modelar lo que se quiere implementar y comenzar a verlo como lo que es, un proceso con todo y el significado semántico de la palabra.

Sobre el proceso de diseño inciden aspectos que tipifican estas dimensiones anteriormente definidas en el marco histórico, marco tecnológico, interrelación de elementos de software, seguridad y rendimiento e interfaz tales como los paradigmas y lenguajes más usados en el momento en que se enmarca el proceso, la selección de framework de trabajo, la automatización del código, la selección y uso de patrones, entre otros.

A medida que los sistemas de información son más avanzados, los clientes requieren mayor variedad de servicios lo que conduce a su vez a mayores y más complejos sistemas .Para poder responder a estas demandas con rapidez que el momento amerita, el software debe

construirse de acuerdo con esquemas de representación y diseño que proporcionen una potente capacidad expresiva y resulten fáciles de comprender, extender y reutilizar.

De ahí la necesidad de pensar en el diseño como un proceso industrializados, en el que los diseñadores al igual que los desarrolladores en sentido general fungirían como obreros de una fábrica, pero en este caso de software, produciendo a partir de modelos, estándares y normas. Sin embargo, esto nunca deberá limitar la capacidad creativa de los equipos de desarrollo. Producir software escala industrial, no es como producir autos, no es montar una línea de producción de ensamblaje secuencial y mucho menos es ejecutar sin pensar solo por el hecho de que esta normado. Es sumamente importante, que está industrialización conciba flexibilidad en su definición.

Industrializar el proceso de diseño, primero que todo abarata los costes y reduce el ciclo de desarrollo, y en segundo lugar, además de ser una consecuencia de esto mismo, es un enfoque de producción que responde a las demandas de un mercado tan volátil como es el de software y se apoya enormemente en una de las bondades del diseño orientado a objeto, la reutilización.

2.1.1 Marco Tecnológico

La complementariedad es una característica de este enfoque en perspectivas o dimensiones del diseño. Esto hace que los aspectos tecnológicos sean alcanzados con más detalles en esta otra perspectiva.

Una vez definida la tecnología a emplear en la construcción del mismo es necesario realizar un análisis sobre la incidencia de esta en la construcción de un modelo de diseño detallado. Este análisis debe estar esencialmente dirigido al empleo de patrones de diseño partiendo del principio de que muchos patrones no pueden ser aplicados a determinado lenguaje de programación, provocando que existan patrones específicos de un lenguaje o de alguna tecnología de desarrollo, incluso hasta de un paradigma en especifico.

El marco tecnológico del diseño brinda un entorno de análisis de dos aspectos esenciales relacionados con la tecnología, bien sea lenguajes, framework y paradigmas a emplear y el impacto en el diseño, como de la propia tecnología encargada de automatizar el proceso de diseño y la generalización de código fuente.

2.2 Procedimiento de diseño en los diferentes proyecto

A continuación sé vera los procedimientos de diseño que se realizan en los diferentes proyectos realizados en la Universidad de las Ciencias e Informática.

2.2.1 Proyecto Sistema Único de Aduana (SUA)

En el proyecto Sistema Único de Aduana lo primero que ellos realizan en el proceso de diseño es la **Elaboración de la Descripción Diseño** dentro de este se realizan:

Diagramas de clases: donde se diagraman las clases utilizadas para modelar el negocio, se describen cada una de las clases previamente definidas, se describen cada uno de los métodos a realizar así como su funcionamiento, se diagrama la interacción de (los) paquete(s) del módulo, con los demás módulos y aplicaciones así como con el núcleo de la aplicación, cumpliendo con la arquitectura, se realiza el diagrama de clases del diseño de cada uno de los escenarios de los casos de uso.

Además, identifican las acciones donde detallan los diagramas de bloques de las acciones, descripción de las acciones y modo de funcionamiento, en su interacción con ExtJs (Componentes Visuales).También identifican los eventos de los componentes visuales detallando diagrama de bloques de los componentes visuales y describiendo los componentes visuales y modo de funcionamiento, en su interacción con las acciones.

Realizan la identificación de servicios a otros módulos, realizando el diagrama de bloques del funcionamiento de cada uno de los servicios a ofertar a cada uno de los módulos.

Y finalmente se reúnen todos los diseñadores de cada uno de los módulos, donde se hace una presentación del diseño de cada modulo, después identifican nuevos requerimientos no funcionales y definen los servicios para otros módulos y por último realizan una verificación del cumplimiento de la Arquitectura.

2.2.2 Proyecto Banco

Antes que todo ellos dependen del analista que les dice las clases que van a utilizar por cada modulo y junto con el analista estudian los casos de usos, después de esto ellos se reúnen con el arquitecto para definir la arquitectura y acoplar el diseño a esa arquitectura describiendo casos de usos y modelos de casos de usos.

Realizan el diseño del modelo de datos, el diseño de Paquetes y el diseño del dominio. Diseñan tablas de bases de datos de módulos y esto lo realizan con el analista. Describen los métodos fundamentales y diagraman las interfaces, atributos y métodos.

2.2.3 Proyecto de Prisiones

Primeramente ellos para realizar su diseño lo primero que hacen es un Análisis inicial entre todos los diseñadores del proyecto para ver que es lo que van hacer, intercambiando conocimientos etc. Definen la estructura para la implementación lógica del negocio y después de esto ellos definen que es lo que realmente hace falta para poder integrar el trabajo de los programadores del equipo.

Realizan la implementación, definen las tareas y realizan la documentación donde se encuentran los diagramas de clase de diseño por cada caso de uso, los diagramas de secuencia más complicados.

En general los proyectos de la universidad tienen su forma de realizar el diseño, pero al final todos proceden a llevar a cabo los mismos pasos. A pesar de ello aun les faltan algunos elementos importantes para optimizar el trabajo en el mismo, como son los mecanismos de respaldo a tener en cuenta para que el sistema pueda recuperarse de los errores detectado; el mantenimiento del diseño, el cual no es realizado con la consecutividad que debería ser y que los cambios realizados en el mantenimiento no se les informa al grupo de proyecto para que los mismos trabajen sobre las actualizaciones; la documentación necesaria para cada uno de los artefactos generados, así como desarrollar y aplicar desde inicios algún mecanismo de pruebas conceptuales y estructurales al modelo de diseño, con el objetivo de garantizar la calidad del proceso.

Además de que muchos definen los estándares a utilizar, pero la gran parte de los diseñadores no se rigen por estos estándares definidos y es por eso que surgen los parches, para poder unir un módulo con otro. Tampoco utilizan los patrones de diseño, ya que podría sacrificar en cierta medida el análisis del diseño.

2.3 Proceso de Diseño con Altos Niveles de Calidad

Para desarrollar un proceso de diseño con altos niveles de calidad, es preciso:

- Tener un equipo de diseñadores de software experimentados, bien organizados y conocedores del dominio del problema sobre el cual se estará haciendo la modelación del sistema.
- Desarrollar un proceso de diseño jerárquico hacia abajo en todo el momento, manteniendo la trazabilidad para poder realizar *backtracking* cuando se ha generado algún problema en alguno de los niveles de detalles.
- Análogamente, evitar la jerarquización del diseño de abajo hacia arriba debido a que este provoca en la mayoría de los casos, el rediseño del sistema.
- El diseño deberá ser abierto en todo momento y bien documentado, haciendo posibles la familiarización con este de todo el equipo de trabajo.
- Desarrollar y aplicar desde inicios algún mecanismo de pruebas conceptuales y estructurales al modelo de diseño, con el objetivo de garantizar la calidad del proceso y del producto.
- Aplicar patrones de diseño y documentar el uso de estos en el modelo de diseño.
- El diseño deberá conducir a interfaces cada vez más sencillas y generadas con el objetivo de que reduzca la complejidad de conexión entre módulos y el exterior.
- Definir y aplicar mecanismos de detección temprana de errores para reducir los costes de corrección.
- Implementar un mecanismo de revisiones sistemáticas del diseño por pares.
- Centrar la atención en desarrollar un diseño en principios sencillos y claro, y abstenerse a garantizar cualquier tipo de optimización hasta que el modelo cumpla con determinada completitud y pueda ser analizada la posibilidad de mejorar algún aspecto, ya sea de rendimiento, la nitidez de las estructuras, entre otros elementos.

Estos son algunos elementos, técnicas y principios que permitirán desarrollar un proceso de diseño con calidad.

2.3.1 La Reutilización del Diseño

Como se había dicho en ocasiones anteriores, reutilizar ya sean elementos de diseño o código disminuye el tiempo de desarrollo y abarata los costos de producción, a lo que se le podría añadir que permite concentrarse en el dominio del problema y no en problemas que ya se encuentra evaluados, analizados e incluso solucionados por otras personas.

Un elemento importante para la reutilización y su correspondiente institucionalización, es la adopción de algún modelo que permita definir los procesos a seguir para alcanzar una mejor efectividad en esta actividad. Este modelo ya fue creado en la universidad Valladolid con el nombre de Modelo de Reutilización Giro (MGR).

Otro aspecto muy importante en el modelo MRG, se encuentra la de comenzar a tener en cuenta la reutilización desde etapas tan tempranas como lo son requisitos funcionales del sistema a partir de la analogía de escenarios, debido a que estos (requisitos funcionales) dan a medida de si un elemento reutilizable se puede emplear en un contexto dado.

Resumiendo, deberán tenerse en cuenta tres aspectos esenciales para lograr un adecuado proceso de reutilización en el diseño.

- Definir un equipo responsable de realizar esta actividad.
- Seleccionar un modelo de reutilización el cual bien podría ser el propio modelo MRG.
- Crear un repositorio o biblioteca de elementos reutilizables para próximas situaciones.

2.3.3 Robustez y fiabilidad del diseño

Es preferible entender el término fiabilidad en el contexto del diseño, como la correspondencia y cumplimiento de este con los requerimientos del sistema, dado que la fiabilidad, como se mencionaba en el capítulo anterior, es un atributo que se mide en etapas posteriores y está relacionado más específicamente con la fiabilidad del sistema. Evidentemente, de etapas de diseño puede comenzarse a garantizar dicha fiabilidad.

Es evidente que un diseño robusto hará más fiable el sistema. Por un lado, la robustez del diseño radica en el grado de capacidad que presenta para funcionar correctamente frente a entradas de información erróneas. Es un término muy asociado a los niveles de tolerancias de errores y fallos y más usado en el contexto de la implementación. Sobre la fiabilidad del diseño se puede decir que es la probabilidad de que este funcione como se espera que lo haga bajo condiciones fijas y durante un tiempo determinado.

2.3.3.1 Mecanismo de recuperación de errores

Recordemos que un error es esa parte del estado del sistema que es distinto de los valores esperados y que pueden conducir a la falla de un sistema, la recuperación de una falla es un

proceso que involucra la recuperación de estados erróneos a un estado libre de error. Hay dos enfoques para la recuperación de un estado de error a un estado libre de error.

Todos los sistemas deben implementar algún mecanismo de recuperación de errores, bien sea hacia adelante o hacia atrás, que permite corregir el funcionamiento del mismo o llevarlo a un estado anterior de funcionalidad respectivamente.

El mecanismo de recuperación de errores debe tener incorporado algún mecanismo de depuración para que después de cada avería, se depure el programa y se ejecute de forma iterativa hasta tener una fiabilidad aceptable.

Para que la cobertura de detección de errores en un mecanismo de recuperación aumente, podría definirse algún esquema de tolerancia de fallos.

Los fallos no se pueden observar de forma directa, sino que deben ser deducidos a través de la presencia de errores. Dado que un error es entendido como un estado del sistema, pueden realizarse comprobaciones para ver si existe o no el error. Idealmente, el mecanismo de detección de errores debe detectar valga la redundancia cualquier posible error causado por aquellos fallos que el esquema de tolerancia a fallos intenta tratar. Sin embargo, esto no es totalmente posible. Para poder ganar en efectividad a la hora de realizar algún diseño de esquema de tolerancia a fallos, es preciso se tengan en cuenta dos aspectos importantes:

- Que al esquema o sistema de comprobación de tolerancia a fallos no debe afectarle los fallos del sistema que se pretende comprobar.
- Y en segundo lugar, el esquema debe ser completo, de manera que pueda tener en cuenta todos o la mayor cantidad posible de errores y que nunca informe sobre un error inexistente, respectivamente.

Existen dos maneras de recuperación de un sistema dada la ocurrencia y detección de algún error. Por un lado está la técnica de recuperación hacia adelante (**ford ward recovery**), la cual se encarga de realizar un análisis tanto del alcance de los daños como de las razones de los mismos para poder eliminar el error del sistema, llevándolo a un estado de normalidad. Esta técnica realiza un importante empleo de mecanismo de tratamiento de excepciones como vías

para realizar la recuperación del sistema hacia delante, haciendo correcciones desde un estado erróneo.

Por otra parte, está la técnica de recuperación hacia atrás (**backward recovery**), la cual se encargara de llevar el sistema a un estado anterior, el que se supone sea correcto. Esta técnica conlleva a un considerable consumo de recursos debido a que debe almacenarse periódicamente el estado del sistema, lo que la hace en ocasiones ser inapropiada para sistemas donde el rendimiento sea un requisito indispensable para el correcto funcionamiento. Sin embargo, su dependencia a la naturaleza de la falla la hace apropiada para fallos transitorios.

Se ha estado hablando de dos conceptos fundamentales: la prevención y la tolerancia a fallos. La primera se realiza en cuatro fases fundamentales:

1. Evitación
2. Comprobación
3. Detección
4. Eliminación

La evitación de fallos trata de impedir que se introduzcan fallos durante la construcción del sistema para que no se produzcan errores posteriores. Este proceso se realiza desde dos perspectivas diferentes pero estrechamente relacionadas. Desde una perspectiva de hardware, mediante la utilización de componentes fiables, técnicas rigurosas de montajes de sub-sistemas, apantallamiento de hardware, entre otros aspectos. Y desde un enfoque de software, en la especificación rigurosa de requisitos, métodos de diseño comprobados, utilización de lenguajes de abstracción y modularidad.

Evidentemente, tratar de evitar los fallos no garantiza que estos no sucedan; por este motivo, la etapa de comprobación de fallos es crucial para lograr una correcta efectividad en el proceso. Las revisiones de diseño o revisiones técnicas (**PRESSMAN, 2001**) es una de las técnicas de comprobación más usadas y efectivas, además de la verificación de programas y la inspección de código. Otra técnica es la de realizar pruebas del sistema.

La eliminación de fallos es la última etapa y una de las más importantes. La efectividad de esta depende de la eficiencia con que se hayan desarrollado las otras etapas anteriores. Para el

caso de la tolerancia a fallos, esta trata de que el sistema continúe funcionando aunque se produzca algún fallo. Existen diferentes grados de tolerancia a fallos en dependencia del tipo y objetivos de la aplicación.

Para aumentar la fiabilidad del sistema pueden usarse los dos mecanismos para crear uno solo, un mecanismo de prevención y tolerancia a fallos.

2.3.3.2 Políticas de tratamientos de excepciones

La gestión de excepciones forma parte del mecanismo de gestión de errores de un sistema. La diferencia radica básicamente en que la primera "conecta" la gestión de errores directamente con el lenguaje de programación y a veces hasta con el sistema operativo. Una excepción es un objeto que sé "lanza" desde el lugar del error y puede ser "capturado" por un manejador de excepciones apropiadamente diseñado para manipular este tipo en particular de error.

La esencia de una correcta política de manejo de excepciones en la construcción de un sistema es desarrollar diseños simples, modulares y extensibles. El mayor enemigo de la fiabilidad de un sistema es la complejidad. Una codificación elegante y legible así como la claridad y simplicidad de las construcciones, ayuda a que se eleve la fiabilidad del software. Tanto el diseño como el código deben estar "escritos" para ser "leídos".

Para enriquecer el mecanismo de tratamiento de excepciones de un sistema, pudiera modelarse el diseño del mismo desde una perspectiva de teoría de aserciones que incluya todo lo relacionado con la declaración formal de lenguajes y diseño desde el punto de vista de las precondiciones de cada contrato de abstracciones, post-condiciones e invariantes, que viene siendo los posibles valores a asumir por parte del dominio de datos que se modela.

Sea cual sea el mecanismo que se vaya a emplear, debiera partirse del principio de que el tratamiento de excepciones no forma parte de la lógica del funcionamiento del sistema, sino de la vía para hacer que este funcione correctamente y de manera fiable y robusta.

En resumen, los mecanismos de tratamiento de excepciones son más que necesarios, imprescindibles, en todo sistema, pero embeberlos en el diseño y el código de la lógica de funcionamiento básica del sistema, que debe responder al cumplimiento de requisitos funcionales del mismo, sencillamente es una mala práctica que trae consigo consecuencias en

el mantenimiento del modelo de diseño y el código producto a la poca legibilidad; además del comprometimiento de la fiabilidad de los mecanismos de detección de errores debido a que al producirse una falla sería muy difícil localizar la región en donde se produjo y hacer más difícil conocer la causa.

2.3.3.3 Estrategias de mantenimiento

Existe una incorrecta de denominar mantenimiento a lo que se le debiera llamar evolución. Este previsto o no, los diseñadores de sistemas, bien sean sistemas simples o complejos, tienden a evolucionar con el tiempo, situación que con frecuencia se etiqueta de manera errónea como darle mantenimiento al diseño. Evidentemente, este análisis es análogo para con los sistemas en sentido general.

La principal estrategia a seguir deberá ser la de realizar mantenimiento preventivo sistemático, con el objetivo de dar respuestas tempranas a señales de deterioro. El mantenimiento es una actividad muy ligada a otras acciones del diseño, como lo son la gestión de riesgos, mecanismos de pruebas, mecanismos de recuperación de errores y manejo de excepciones, revisiones técnicas del diseño, gestión de cambios, entre otros aspectos.
Es sumamente importante que el personal de mantenimiento del diseño este a tono con los estándares usados, las políticas de manejo de excepciones y la aplicación de los conceptos de fiabilidad durante la fase de diseño. Dicho personal bien puede ser parte del equipo que realizo la sección del diseño objeto de mantenimiento pero que hayan estado en contacto directo con el proceso de diseño del sistema. No es conveniente reasignar personal nuevo para que ejecute esta actividad, dado que aumentaría las probabilidades de introducir nuevos errores y aumentarían el tiempo de respuesta a cambios; dos factores que evidentemente afectarían la probabilidad de una empresa de software.

Otra buena práctica es la cuantificación de las decisiones de diseño en términos de costes de mantenimiento. Esto quiere decir que dado que detrás de cada decisión de diseño, se ponen en juego una gran cantidad de dinero y oportunidades de mercado, es necesario que se optimice desde el punto de vista del valor, el manteniendo implicado en cada decisión de diseño realizada. Existen innumerables esfuerzos por lograr este objetivo, sin embargo, ninguna de las heurísticas, patrones o principios hacen posible este hecho (**CABRERO, y otros 2008**).

Para hacer óptimo la mantenibilidad del diseño, debe partirse básicamente de un diseño simple. Eh aquí la importancia de la simplicidad del diseño. Comenzando por modelar la solución de manera simple e incrementando la complejidad solo en aquellas partes del diseño donde el sistema lo requiera, el diseñador tendrá que ser capaz de evaluar la factibilidad de cada decisión de diseño desde un enfoque orientado a la mantenibilidad.

Finalmente, se sugiere partir de algún modelo de reglas para la mantenibilidad del diseño, capaz de brindar un esquema que permita al diseñador guiarse por él para realizar el análisis de factibilidad de algún cambio en el diseño y conservar a la vez los costes de mantenibilidad bajo control.

De manera conclusiva, el mantenimiento no es algo trivial dentro del proceso de diseño, sino todo lo contrario. Constituye una labor cuya atención desde inicios del proceso y la aplicación de buenas prácticas, reglas, principios, un trabajo bien organizado, buena gestión y estimación de riesgos, entre otros aspectos relacionados con la gestión de proyectos, harán que su aplicación sea efectiva y el impacto en el sistema positivo.

2.3.4 Equipo de diseño

La experiencia y conocimiento del dominio del problema de los miembros de un equipo de diseño influye, casi de manera determinada, en la obtención de un buen diseño. Por una parte la experiencia de los diseñadores deberá estar basada en la experiencia que tengan estos sobre cómo realizar el proceso, el empleo de patrones, métricas, lenguajes de modelado, paradigmas, estándares, tecnología y mecanismos.

A esto deberá sumársele el conocimiento que tenga sobre el dominio del problema a modelar, los requerimientos del sistema, los estándares que deberán emplear para este caso, el modo de trabajo en equipo, entre otros aspectos relacionados con la organización.

Lo anterior impone que se defina algún método de diagnostico de formación del equipo de diseñadores. Esto ayudara a determinar cuáles son los cambios necesarios y si la formación será una actividad apropiada. En caso que así sea, deberán centrar los temas de preparación en los problemas detectados en el diagnóstico, el cual a su vez deberá estar orientado a las necesidades de la organización.

2.5 Modelo de Proceso para el rol de Diseñador de Aplicaciones

Aquí se muestra una vista de cómo es el procedimiento a seguir por el diseñador de aplicaciones en el Proyecto Sistema Único de Aduana.

1. Primeramente se realiza un análisis y estudio previo donde se reúnen todos los trabajadores para ponerse de acuerdo de cómo se deberá hacer el diseño. Después se comienza con la realización de diseño, donde se realizan todas las actividades vistas en **el epígrafe 2.4**. Al terminar con este paso procedimos a la revisión del diseño donde se documenta las principales observaciones hecha, si se encuentra algún error se le manda al equipo de realización del diseño el error encontrado, allí ellos lo analizan y lo arreglan; si no se pasa al siguiente paso. Entendimiento y compromiso es el siguiente paso donde se reúne el programador con el diseñador de la lógica funcional para establecer la fecha de entrega del proyecto.

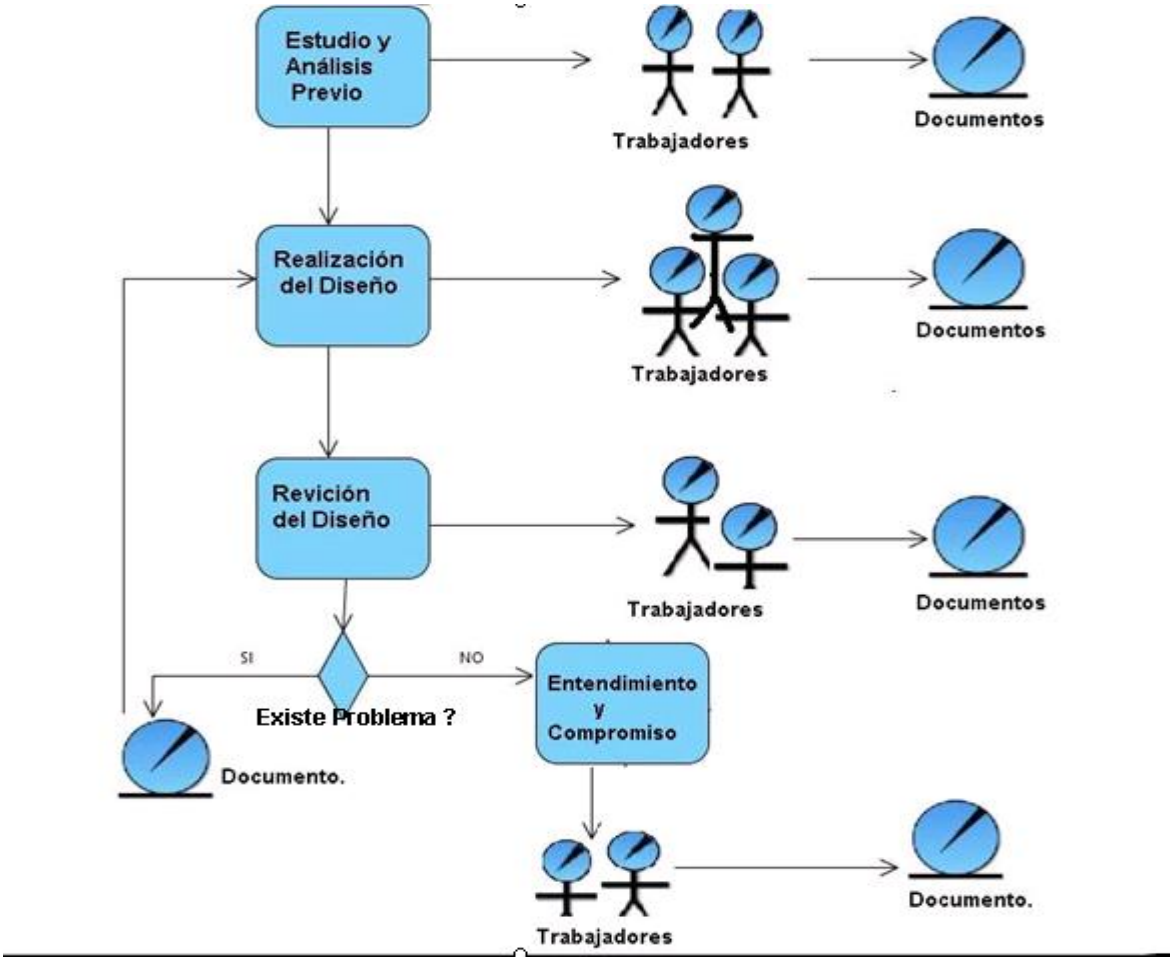


Figura 2.4.1 Procedimiento de Diseño.

2.4 Principales Actividades a Realizar por el Diseñador de Aplicaciones

Definiendo los roles de trabajo y su ubicación dentro de la propuesta de estructuras organizativas de un equipo de diseño, es conveniente que se pase a definir cuales son las actividades que deberá realizar cada rol.

Es muy importante que se tenga en cuenta una vez más que cuando se emplea alguna metodología de desarrollo de software, esta define en cierta medida que actividades se realizan dentro del proceso de diseño y por quienes son realizadas. Por esta razón la propuesta estará orientada en todo momento a apoyar y no a sustituir el proceso. El modelo es lo suficientemente flexible como para poder adaptarlo a las condiciones reales de la organización que lo implemente.

Cada actividad la ejecuta un rol determinado, en un momento determinado y manipula determinados artefactos de diseño. En este momento solo se hará referencia a quien realiza que actividad y los documentos generados por cada actividad. En el **epígrafe 2.4.1** se estará abordando sobre el expediente de diseño y los artefactos que lo conforman.

Las actividades propuestas a realizar por el equipo de diseñadores en el procedimiento serán las siguientes:

Realización del Diseño:

Diseñador principal:

Definición de los estándares de diseño: esta actividad tiene una importancia significativa, ya que de ella depende que todo el equipo de diseñadores “hablen el mismo idioma”. Dentro de sus principales aspectos estará la selección de algún lenguaje de modelado para la modelación gráfica del diseño (se propone UML). Esta actividad genera el ***documento de definición de estándares del proceso de diseño.***

Diseño Arquitectónico: Esta actividad está muy relacionada con el rol de arquitecto del sistema. En realidad pudiera interpretarse como tal. Se refiere a la modelación del sistema a nivel de componentes y subsistemas. El artefacto generado es el ***documento de diseño Arquitectónico.***

Conformación del expediente de diseño: este es el principal artefacto que es generado por el proceso de diseño. Este es constantemente actualizado y servirá como referencia principal de la organización para abordar temas relacionados con el proceso de diseño.

Reuniones de planificación y control: estas reuniones se realizaran con determinada sistematicidad definida por el equipo de diseño y respondiendo en todo momento a las necesidades de la organización (**ver Anexo 1**).

Diseñador de la lógica funcional

Diseño de sub-sistemas: Es una forma de organizar los artefactos del modelo de diseño en piezas más manejables, definiendo el diseño arquitectónico del sistema. Los detalles relacionados con la estructuración de cada subsistemas, corresponde a este rol crearlos. Se deberá definir el comportamiento específico de cada subsistema en términos de colaboración entre los elementos de diseño contenidos y los subsistemas externos a través de las interfaces. El ***Modelo de subsistemas*** será el artefacto generado en esta actividad.

Diseño de clases: corresponde a la modelación de las clases de diseño, en esta actividad:

- Se especifican utilizando la sintaxis del lenguaje de programación elegido.
- Se especifica la visibilidad de los atributos y operaciones ej. Public, Protected, Private.
- Se implementan las relaciones (referencias entre objetos) las que tienen un significado en la implementación.
- Se especifican los métodos. Tienen correspondencia directa con los métodos de la implementación.
- A menudo, aparece con un estereotipo que se corresponde con una construcción de lenguaje de programación.
- Puede realizar interfaces si tiene sentido en el lenguaje de programación.

En esta actividad se obtendrán los ***diagramas de clases del diseño*** que son una representación más concreta que el diagrama de clases del análisis (**ver figura 3.1**), representa la parte estática del sistema las clases y sus relaciones**.**

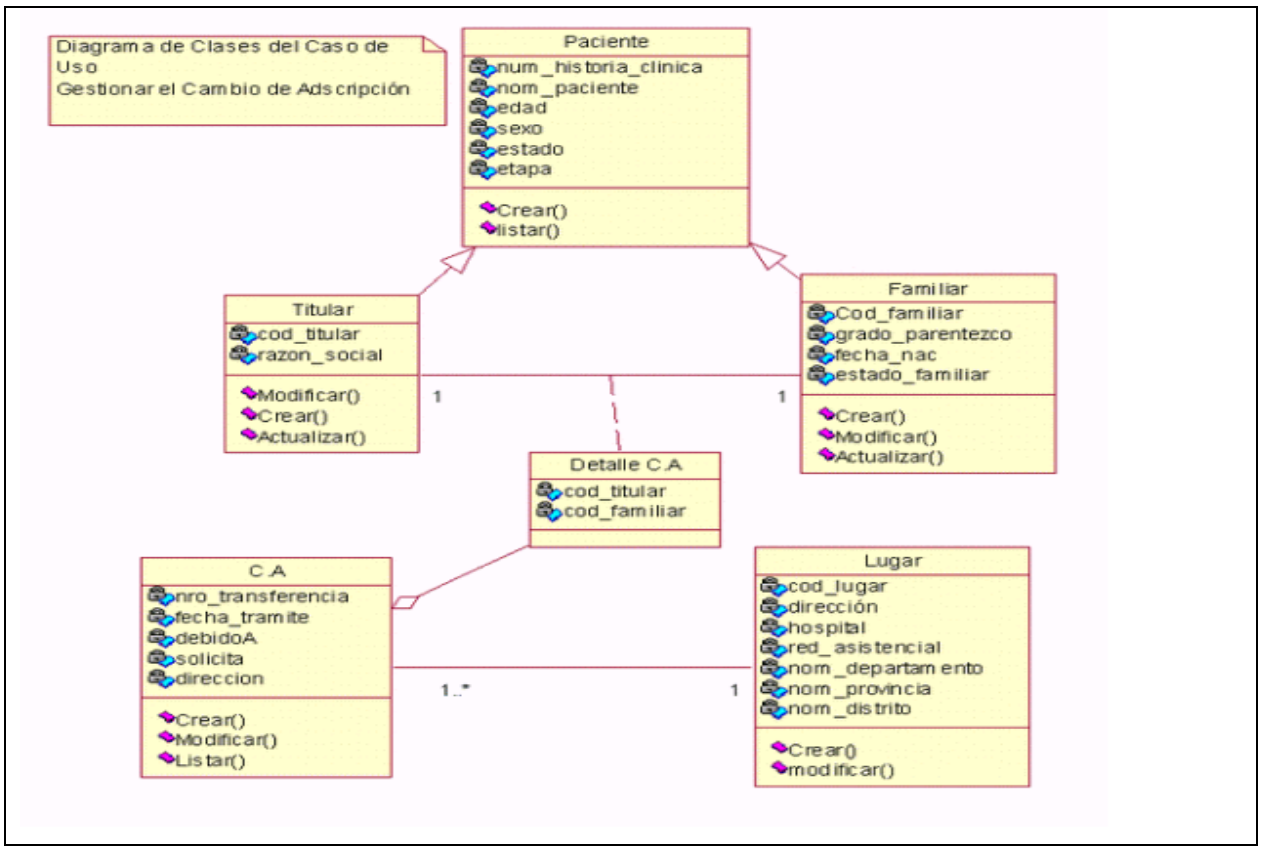


Figura 3.1 Diagramas de clases de análisis.

Relaciones entre las clases de diseño **(ver Anexo 2).**

Diseño de la lógica funcional: no es más que la modelación del diseño que responde a los requerimientos funcionales del cliente. Corresponde a la creación de ***Diagramas de Estados e interacción.***

Diseñador del modelo de datos.

Análisis de las clases de diseño: constituye el primer paso para la transformación del modelo de clases del diseño a un modelo de datos esto sería: ver **figura 3.2**

1. Convertir cada clase en una tabla
2. Especificar la llave primaria de cada tabla
3. Crear asociaciones Entidad Relación (ER) con distintas multiplicidades.

Diseño del modelo de datos: Es la construcción del **modelo de datos** en cuestión y este se divide en dos partes el modelo lógico y el modelo físico.

Modelo lógico: proveer de una vista de las entidades lógicas de datos y sus relaciones con independencia de la plataforma de base de datos a utilizar ver **figura 3.2**

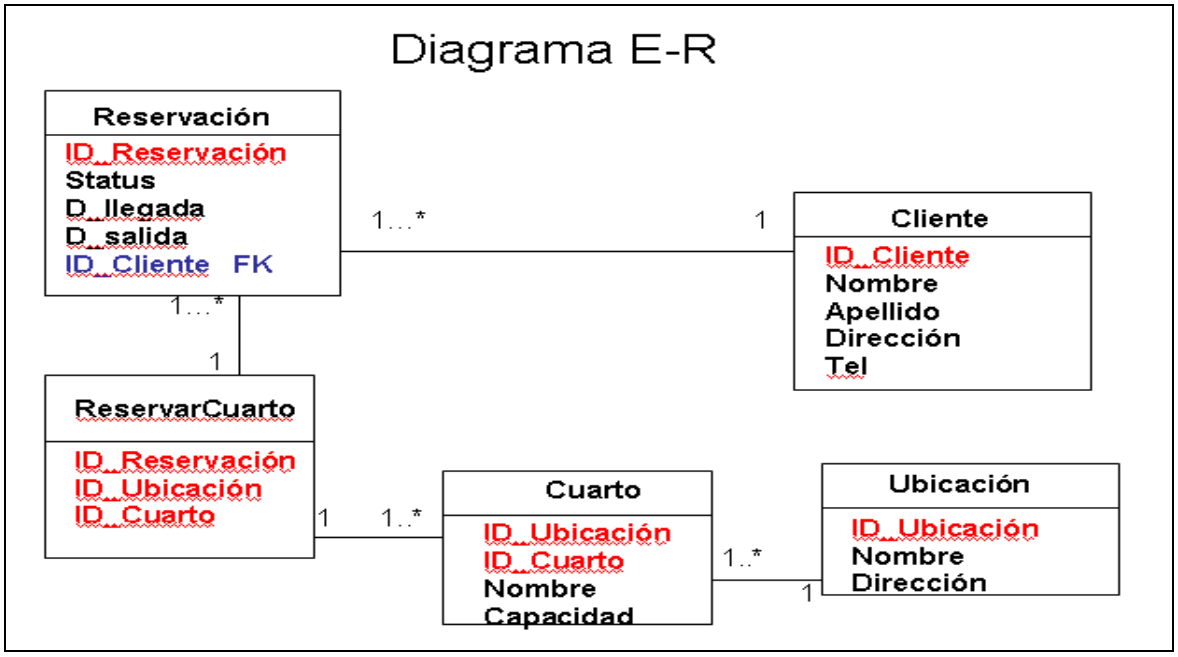


Figura 3.2 Diagrama entidades lógicas de datos y sus relaciones.

Modelo físico: Representan las funciones del sistema y como se van a llevar a cabo en una plataforma tecnológica específica de hardware y software.

Diseñador de mecanismo de respaldo

Diseño de mecanismos de recuperación de errores: Un importante momento en el funcionamiento del sistema, es la capacidad que tenga este de recuperarse de errores detectados. Estos mecanismos responderán a las necesidades del cliente en el sentido en que deberán ser hacia adelante o hacia atrás o partirán de alguna política de tolerancia a fallos. El artefacto asociado es el ***documento de Especificaciones de Modelo de Mecanismos de Recuperación de errores y Tolerancia a Fallos.***

Diseño de mecanismos de manipulación de Excepciones: se deberán definir el modelo de clases que deberá al tratamiento de excepciones. Para esta actividad se genera el ***Mecanismo de excepciones***, el cual incluirá además una vista de las clases que definen dicho mecanismo.

Responsable de integración.

Definición de estándares de integración: estos estándares estarán orientados a la manera de integración del modelo de diseño de mecanismos de respaldo con la lógica funcional del sistema. Para el caso del empleo de orientación a aspectos para el diseño de los mecanismos de respaldo, se basara básicamente en la definición de los puntos de enlaces donde se tejera el código correspondiente a la implementación de estos aspectos de respaldo del sistema.

Responsable de mantenimiento

Mantenimiento sistemático del diseño: Esta actividad genera un artefacto que se estará muy asociado a la gestión de cambio pero que se encargara de especificar el plan de mantenimiento, las razones de cambio, los cambios en sí y la descripción sobre la factibilidad de ejecutar el cambio: ***Especificaciones de Mantenimiento*.**

Analista de reutilización

Identificación de elementos de reutilización: Básicamente consiste en realizar un estudio del estado del arte sobre diseños asociados a sistemas similares, con el objetivo de detectar posibles elementos reutilizables para el actual diseño en construcción. Esta actividad genera precisamente el ***Documento de estudio del estado del arte de la reutilización.***

Análisis de uso de Patrones de diseño: Análogamente, para el caso de los patrones el analista de reutilización deberá realizar un estudio asociado a los patrones que pueden ser empleados en el desarrollo del proceso y compatibles con la tecnología y los paradigmas que se estén empleando en la organización.

Desde esta perspectiva sé estará garantizando en cierta medida (entre otros aspectos) que el diseño actual sea lo más reutilizable posible. Para este se realizara el ***Compendio de Patrones de Diseño*** donde el principal objetivo de este artefacto es realizar una breve descripción sobre la base de la factibilidad, de los patrones objetos de estudio.

Responsable de repositorio de reutilización

Administración del repositorio de reutilización: este rol y el anterior trabajan en conjunto. El presente se encarga de administrar el ***Repositorio de Reutilización***, precisamente

construido partiendo del análisis realizado por el analista de reutilización. En esencia el repositorio no es más que el nicho donde se podrán encontrar aquellos diseños o segmentos de diseño o cualquier elemento de diseño reutilizable, que podrán ser reutilizados en la construcción del actual diseño.

Responsables de calidad del diseño

Evaluación de Métricas de diseño: Dado a que el rol que deberá realizar esta actividad es miembro del equipo de calidad de la organización o proyecto, la misma por consiguiente formara parte de las actividades que desarrollara dicho equipo. Se seleccionaran las métricas de acuerdo con los atributos de diseño que se evaluaran y se generara un ***Documento de Evaluación de Métricas de Diseño*** en el cual aparecerán registradas las mediciones y se especificaran potenciales mejoras.

Evaluación del cumplimiento de Estándares de diseño: esta es una tarea complementaria cuyo objetivo es velar por el cumplimiento de los estándares de diseño definidos en el Documento de Definición de estándares de diseño, especificando por la Gerencia General de equipo.

Revisión del Diseño:

Revisor Técnico

Revisiones Técnicas Formales: Las revisiones devienen en importante mecanismo para la detección de errores en el diseño. Son empleadas en muchas metodologías y su efectividad es considerable partiendo de la idea de que un gran por ciento de los errores del sistema es introducido durante la fase de diseño. Cada vez que se realice una revisión del diseño se generara un ***Registro de Revisión Técnica***, documento en el cual se reflejaran las principales observaciones realizadas en función de potenciales mejores, los errores encontrados en caso de que existan y sus posibles soluciones.

Si este revisor encuentra algún error después de documentarlo lo envía al equipo que trabaja en la revisión del diseño, para que estos arreglen el problema encontrado.

Estas son las principales actividades que se proponen se realicen en el proceso de diseño.

Es responsabilidad de cada rol que realice su actividad con buena calidad y con tiempo para que la entrega del producto se haga como está planificado y que el mismo tenga altos niveles de calidad.

2.4.1 Expediente de diseño

El Expediente de Diseño es un artefacto por sí solo, de hecho, es el principal artefacto que genera el proceso de diseño. Este es un elemento que se propone sea empleado como principal mecanismo para presentar el avance y la forma de trabajo desde la perspectiva del diseño.

En sentido general podría decirse que el expediente de diseño no es más que un archivo que comprende la construcción de un documento denominado **Informe General del Proceso de diseño**, en el cual se recoge información relacionada con los elementos estructurales de conformación del equipo de diseño, el modo de trabajo, la definición de estándares y una visión general de los artefactos más importantes generados por los departamentos del proceso de diseño. La otra parte del expediente de diseño es la conformación integra de todos los artefactos del proceso sobre su formato original. Es decir, el archivo, como elemento aglutinador, define un conjunto de secciones análogas a la definición estructural del proceso y en las que se incluye un espacio donde quedaran almacenados todos los artefactos.

Para que se comprenda con más claridad, la estructura general que se propone del Expediente de Diseño es la siguiente:

Documentos Generales

*Informe General del proceso de Diseño

*Actas de Reuniones de control

Artefactos

- Documento de Definición de Estándares del Proceso de Diseño.
- Documento de Diseño Arquitectónico.
- Documento de Evaluación de Métricas de Diseño.
- Documento de Análisis de Factibilidad Tecnológica.
- Documento de Estudio del Estado del Arte de la Reutilización
- Compendio de Patrones de Diseño.

- Repositorio de Reutilización.
- Modelo de Subsistemas.
- Diagramas de clases del Diseño.
- Diagrama de Estado e Iteración.
  (a) Diagrama de secuencia.
  (b) Diagrama de Colaboración.
- Mecanismo de Excepciones.
- Especificaciones del modelo de Mecanismos de Recuperación de Errores y Tolerancias a Fallos.
- Especificaciones de Mantenimiento.
- Registro de Revisiones Técnicas.

El desarrollo o no de algunos de estos artefactos estará en dependencia de los objetivos de la organización y de su modo de trabajo. La propuesta realizada responde al cumplimiento del objetivo general que persigue el presente trabajo, de brindar una propuesta de un guía de actividades que sirva como soporte para el desarrollo procedimientos de diseño.

2.4.2 Artefactos

Cada uno de los artefactos que define la propuesta del presente trabajo tiene un objetivo en especifico y tributa y determina información al proceso para que este pueda desarrollarse lo más efectivamente posible y con calidad. Es preciso se aclare que el emplear estos artefactos no garantiza la calidad del diseño como proceso ni como producto sino que servirá como soporte para que el proyecto gane en control y organización, dos elementos que favorecen en gran medida la calidad.

Cada artefacto es el resultado de un sub-proceso del diseño, lo que indica que para un incremento de la calidad en el proceso deberá realizarse cada tarea de manera correcta, respetando los principios del diseño y el análisis sobre el empleo de la tecnología.

La descripción general de cada artefacto es la siguiente:

Documento de Definición de estándares del proceso de diseño

El objetivo principal del mismo es definir los estándares desde los que deberá partir el proceso (**ver Anexo 3**).

Dichos estándares recogerán aspectos relacionados con Métodos de trabajos, Estilos de diseño, Plantillas para el desarrollo de artefactos, modelos de calidad, métricas de diseño y principios organizativos

Una vez creado deberá ser presentado a la organización para su aprobación, y posteriormente gestionado por el departamento de calidad el cual deberá divulgarlo al resto del equipo de diseño para hacer cumplir lo establecido en el mismo.

Documento de diseño Arquitectónico

El objetivo de este documento es brindar una vista arquitectónica del sistema en construcción, desde la perspectiva de estructuración de los datos y los componentes del programa que se desea construir, en subsistemas, módulos y/o paquetes; así como las características de cada componente y la forma en que estos interactúan. (**Ver Anexo 4**)

Documento de Evaluación de Métricas de Diseño

El resultado de evaluar métricas de calidad orientadas al diseño de software será plasmado en el presente documento, cuyo objetivo principal es servir de fuente de consulta para el equipo de diseño, sobre los atributos de calidad del proceso y sus potenciales mejoras. El mismo se encargara de especificar la familia de métricas empleadas y el resultado de los cálculos realizados. (**Ver Anexo 5**)

Documento de Estudio del Estado del Arte de Reutilización

El objetivo de este artefacto es realizar un estudio sobre la posibilidad de reutilización de diseños externos a la organización o internos a ella. Esto se refiere a empleo de Frameworks, secciones de diseño de otros sistemas y utilización de patrones de diseño. Su alcance será hasta la posibilidad de reutilizar código para minimizar el tiempo y los costes de desarrollo.

Otro aspecto a tratar en el mismo será la evaluación de la compatibilidad de lo que se reutilice respecto a los principios y estándares definidos por el equipo de diseño y la organización en sentido general. Por esta razón, se incluirá el análisis sobre el ajuste de los elementos reutilizables a dichos principios y estándares. (**Ver Anexo 6**)

Compendio de Patrones de Diseño

El compendio de patrones es un documento que definirá el paquete de patrones de diseño que podrán emplearse en la construcción del diseño, sobre la base de la tecnología y los paradigmas que fungen como pilares para el desarrollo del sistema en sentido general. El análisis no se centrara en la factibilidad de uso de determinado patrón en específico sino en el paquete en sentido general (**ver Anexo 7**).

Repositorio de Reutilización

Este es un importante artefacto dado que constituye el nicho de reutilización más valioso para el proceso de diseño. Básicamente será un archivo físico donde podrá consultarse posibles elementos a reutilizar (**ver Anexo 8**).

Será administrado por el Responsable de Repositorio de Reutilización el cual a su vez se encargara de divulgar a todo el equipo de diseñadores los cambios y actualizaciones que en el se realicen así como también de los elementos que los compongan y que puedan ser reutilizados en determinado momento del proceso de diseño.

Expediente de Diseño

Como se mencionaba en ocasiones anteriores, este es el principal artefacto propuesto para ser generado por el proceso de diseño. Su principal objetivo es brindar a la organización toda la información necesaria sobre el proceso de diseño. Está dividido en dos partes:

1 La definición integra de todos los artefactos del proceso (**ver epígrafe 2.5.1)**

2 El informe General del proceso de Diseño.

El informe General del proceso de Diseño

Este es un documento que resume los principales elementos del proceso de diseño. En sentido general, sirve para poder navegar por todo el proceso de una manera rápida y precisa (**ver Anexo 9**). Es sistemáticamente actualizado y constituye una herramienta para poder mostrar al resto de la organización el avance del proceso de diseño.

Registro de Revisión Técnica

Es un documento de anotaciones generado como consecuencia de la realización de una revisión técnica al diseño. Su principal objetivo es realizar un registro de los resultados de haber realizado una revisión al diseño o a una sección de este. Es un artefacto que tributa

información al proceso de mantenimiento y sirve de soporte para llevar un control de registro de errores del diseño. (**Ver Anexo 10**).

Modelo de Subsistemas

Constituye una vista del sistema en subsistemas de diseño, mostrando la forma en que estos interactúan entre sí y otros sistemas externos **ver Anexo 11**).

Diagramas de clases del Diseño

Este es un de los más importantes diagramas estáticos. Su objetivo es mostrar como están diseñadas las clases del sistema y sus interacciones. Se parte del mismo principio de diagramas de clases de diseño que brinda UML y emplean otras muchas metodologías de desarrollo como ROP, XP, entre otras (**ver Anexo 12)**.

El empleo de una determinada herramienta para construir los diagramas del proceso de diseño, será un resultado del análisis de factibilidad tecnológica realizado en etapas anteriores.

Diagramas de Estado e Interacción

La definición de ese tipo de diagramas parte de la que brinda UML propiamente. Los mismos sirven para ilustrar el modo en el que los objetos interaccionan por medio de mensajes y se dividen en dos tipos de diagramas: los de secuencia y los de colaboración. (**Ver Anexo 13**).

- **Diagrama de Colaboración:** ilustra las iteraciones entre objetos en un formato de grafo o red, en el cual los objetos se pueden colocar en cualquier lugar del diagrama (**LARMAN, 2003**) y sirve para entender el orden en que se envían los mensajes.
- **Diagramas de Secuencia:** Un diagrama de Secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo. El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado.

Por otro lado, están los diagramas de Estados, los que constituyen una manera para caracterizar un cambio en un sistema, es decir, que los objetos que lo componen modificaron su estado como expuesta a los sucesos y al tiempo (**SCHMULLER, 2000**).

Mecanismo de Excepciones

En realidad los mecanismos de excepciones son más que un simple modelo de clase basado en aserciones u orientado a aspectos (**ver Anexo 14**). Para este artefacto se deberán tener diferentes consideraciones relacionadas con las políticas de tratamientos de excepciones, estándares de mensajes, además de la infraestructura de clases que soporta esa lógica de tratamiento.

Especificación de mantenimiento

Este es un documento en el cual se definirá la planificación del mantenimiento del diseño, se registraran los cambios realizados y las causas que los provocaron (**ver Anexo 15**). Es un artefacto que será manipulado por gran parte del equipo de diseño y estará constantemente bajo actualizaciones sistemáticas, determinadas por la frecuencia con que se realice el mantenimiento.

Especificaciones del Modelo de Mecanismos de Recuperación de Errores y Tolerancia a Fallos

Este es un documento en que se deberá plasmarse las políticas de tratamiento de errores y la tolerancia o no a fallas del sistema (**ver Anexo 16**). Si se decide realizar este artefacto, no será necesario construir el artefacto Mecanismos de Excepciones ya que se encuentra implícito en el presente.

Conclusiones

En este epígrafe se realizó un estudio sobre los Proceso de Diseño con Altos Niveles de Calidad y la importancia que tiene para la realización del diseño de software ejemplo de esto es La Reutilización del Diseño que disminuye el tiempo de desarrollo y abarata los costos de producción , los mecanismos de Persistencia permite que la existencia de los datos trascienda en tiempo y espacio en un sistema informático, además de técnicas y principios que permitirán desarrollar un proceso de diseño con calidad. Se llevó a cabo igualmente un análisis de como se realiza el diseño actualmente en algunos proyectos de la universidad de la Ciencias e Informática.

Capítulo 3

3 Capítulo 3 "Validación de la Propuesta"

Introducción

Se entiende por evaluación al estudio de factibilidad que pretende detectar y evaluar riesgos y problemas, para luego buscar recomendaciones para solucionarlos y así evitar que el proyecto fracase en posteriores etapas.

En este capítulo se hace una evaluación mediante el uso de los criterios de un panel de expertos y el empleo de técnicas propuestas por el método Delphi con el objetivo de obtener una garantía sobre la calidad de la propuesta realizada. Se hace necesario evaluar el trabajo para contar con opiniones especializadas en el tema antes de continuar con la implementación del mismo dentro del proyecto SUA.

3.1 Método Delphi

El método Delphi fue desarrollado por la Corporación Rand, es una metodología de investigación multidisciplinaria para la realización de pronósticos y predicciones a corto o largo plazo en varias ramas de las ciencias y la técnica, es considerada una de las técnicas subjetivas de pronosticación más confiable, para el desarrollo de este método se necesita la opinión de un grupo de Expertos los cuales son consultados individualmente mediante cuestionarios para que sus planteamientos no influyan en los ajenos.

3.2 Proceso de Selección de expertos.

El grupo de expertos se conformó con especialistas que poseen un amplio conocimiento y experiencia en el desarrollo de aplicaciones de Software, medimos algunos parámetros obligatorios a cumplir por los mismos, estos son:

- Graduado de nivel superior.
- Mantenga una vida activa en el desarrollo de productos informáticos.
- Conocimientos sobre el Diseño de Software.
- Un año de experiencia laboral como mínimo.

Escoger correctamente los expertos que conformarán el equipo, y debe ser una persona experimentada en el tema que se trata, capaz de ofrecer valoraciones objetivas y hacer recomendaciones al respecto.

3.2.1 Cantidad de Expertos seleccionados

No existe una norma que rija la cantidad de expertos que deben conformar un equipo para que este se considere válido, sin embargo, se plantea que a partir de 7 expertos el error disminuye exponencialmente y a partir de 30 la disminución del error continuo disminuyendo pero de una manera poco significativa por lo que no compensa el incremento de costo y esfuerzo, por tal motivo se considera que el rango óptimo sería entre 7 y 30 expertos. En este caso contaremos con las valoraciones de 7 expertos.

3.3 Objetivo a evaluar por los especialistas:

Valorar la propuesta de definición de actividades a realizar por el Diseñador de Aplicaciones en el proyecto Sistema Único de Aduana (SUA).

3.4 Guía para la validación de la propuesta.

Una vez conformado el equipo de Expertos se elabora el cuestionario con los parámetros a medir teniendo en cuenta las características y necesidades de la propuesta de diseño.

El cuestionario se elaboró de manera que las respuestas fueran categorizadas con una puntuación entre 1 y 5 puntos.

1. No adecuado
2. Poco Adecuado
3. Adecuado
4. Bastante Adecuado
5. Muy Adecuado

A continuación se muestran los criterios de evaluación que fueron utilizados en el desarrollo de la encuesta, los cuales se agruparon por trabajador (**Ver Anexo 17**)

En la **Tabla 1** se muestran los valores otorgados por cada experto (E) a cada actividad (A) respectivamente, una vez recogida las opiniones de los expertos se comienza el análisis de la información.

E/A	E1	E2	E3	E4	E5	E6	E7	EP
A1	5	5	5	5	5	5	5	5
A2	4	5	5	4	5	5	5	4.71
A3	5	5	5	5	4	5	5	4.85
A4	5	4	5	5	4	5	4	4.57
A5	5	5	5	5	5	5	5	5
A6	5	5	5	5	5	5	5	5
A7	4	5	4	3	4	5	3	4
A8	4	4	4	4	4	3	4	3.85
A9	5	5	5	5	5	4	5	4.85
A10	4	3	4	4	4	5	3	3.85
A11	3	4	4	4	3	4	5	3.85
A12	5	4	5	4	5	5	4	4.57
A13	3	4	4	4	5	4	3	4.57
A14	5	4	3	4	5	5	4	4.28
A15	4	5	4	5	4	4	5	4.42
A16	4	5	4	5	4	4	5	4.42
A17	4	5	4	4	5	4	5	4.42
A18	4	4	4	4	5	4	3	4
A19	5	5	5	5	4	5	4	4.71

Tabla 1 Valoración de los expertos

Para verificar la consistencia en el trabajo de los expertos utilizamos el coeficiente de concordancia de **Kendall y el estadígrafo Chi cuadrado (X2).**

Coeficiente de concordancia de Kendall y el estadígrafo Chi cuadrado (X2).

Sea **A** el número de actividades que van a evaluarse **(9)** y **E** el número de expertos que realizan la evaluación **(7).**

Para cada criterio se determina:

ΣE: Sumatoria del peso dado por cada experto.

Ep: Puntuación promedio del peso dado por cada experto.

MΣE: media de los **ΣE**.

ΔC: Diferencia entre **ΣE** y **MΣE**.

2. Se determina la desviación de la media, que posteriormente se eleva al cuadrado para obtener la **dispersión (S)** por la expresión:

S = Σ (ΣE - Σ (ΣE / A))2

3. Una vez conocida la dispersión se puede calcular el coeficiente de concordancia de **Kendall (W): W = S / E2 (A^3 – A) / 12**

4. El coeficiente de concordancia de Kendall permite calcular el **Chi cuadrado** real:

X^2 = E (A-1) W

E/A	E1	E2	E3	E4	E5	E6	E7	EP
A1	5	5	5	5	5	5	5	5
A2	4	5	5	4	5	5	5	4.71
A3	5	5	5	5	4	5	5	4.85
A4	5	4	5	5	4	5	4	4.57
A5	5	5	5	5	5	5	5	5
A6	5	5	5	5	5	5	5	5
A7	4	5	4	4	4	5	5	4.42
A8	4	4	4	4	4	4	4	4
A9	5	5	5	5	5	4	5	4.85
A10	4	4	4	4	4	5	3	4
A11	3	4	4	4	5	4	5	4.14
A12	5	4	5	4	5	5	4	4.57
A13	3	4	4	4	5	4	5	4.14
A14	5	4	3	4	5	5	4	4.28
A15	4	5	4	5	4	4	5	4.42
A16	4	5	4	5	4	4	5	4.42
A17	4	5	4	4	5	4	5	4.42
A18	4	4	4	4	5	4	3	4
A19	5	5	5	5	4	5	4	4.71
S	2142							
M ΣE	70							
W	0.53							
X^2	66.78							
X^2(α, ε-1)	25,9894							

Tabla 2 Valores Obtenidos.

Hipótesis fundamental

Determinado el coeficiente de Kendall, es necesario realizar la prueba de hipótesis de que los expertos no tienen comunidad de preferencia.

Con este criterio se intenta verificar la hipótesis fundamental:

H0: No existe concordancia entre los criterios emitidos por los expertos.

Contra la hipótesis alternativa

H1: Hay una concordancia no casual entre los criterios emitidos por los expertos

El cálculo del coeficiente de concordancia de Kendall resultó ser de 0.536, se presupone que si **W** > 0.5 existe convergencia de criterios entre los expertos, por tanto, como queda demostrado, 0.536 > 0.5, en resumen, los criterios alcanzados convergen, en caso de no existir convergencia se hace necesario repetir el trabajo de expertos. Para validar este resultado estadísticamente, se utilizó el estadígrafo **Chi cuadrado (X2).**

El Chi cuadrado calculado se compara con el obtenido de las tablas estadísticas y se considera que es válida la Hipótesis **H0** si se cumple:

X^2real < X^2 (α, c-1)

Por tanto, se rechaza la hipótesis **H0** demostrando que:

X^2real > X^2(α, c-1)

66.78 > 25,9894

Quedando demostrada la hipótesis **H1** que plantea la existencia de concordancia en el trabajo de expertos.

- Una vez demostrada la consistencia del trabajo realizado por los expertos se puede definir el peso relativo de cada criterio
- Conociendo el peso de cada criterio y la calificación dada por los evaluadores se puede construir la siguiente tabla donde se muestra la calificación de cada criterio para obtener el valor de **P × c**., donde **(c),** es el criterio promedio concebido por los expertos.

Actividades	Calificación Promedia (A)					P	P*C
	1	2	3	4	5		
A1					x	0.014	0.501
A2					x	0.012	0.500
A3					x	0.121	0.501
A4					x	0.107	0.460
A5					x	0.103	0.401
A6					x	0.103	0.401
A7			x			0.144	0.230
A8				x		0.017	0.351
A9				x		0.121	0.332
A10				x		0.017	0.351
A11			x			0.027	0.211
A12				x		0.107	0.401
A13				x		0.107	0.402
A14					x	0.118	0.501
A15				x		0.107	0.350
A16				x		0.107	0.350
A17				x		0.107	0.350
A18				x		0.114	0.352
A19				x		0.022	0.301

Tabla 3 Calificación Promedia

Se calcula el Índice de aceptación del proyecto (IA).

IA = S (P × c) / 10

IA = 8.089 / 10

IA = 0.80

9. Y por último se determina la probabilidad de éxito de la propuesta

Rangos predefinidos de Índice de Aceptación

IA > 0,7 Existe alta probabilidad de éxito.

0,7 > IA > 0,5 Existe probabilidad media de éxito.

0,5 > IA > 0,3 Probabilidad de éxito baja.

0,3 > IA Fracaso seguro.

Por lo que se puede concluir que la propuesta tiene una alta probabilidad de éxito.

Después de aplicada la encuesta se determina que los expertos estuvieron de acuerdo en que era muy importante el procedimiento para guiar al diseñador de aplicaciones en el proyecto SUA.

Teniendo en cuenta la completitud de la secuencia de actividades y de los artefactos propuestos, fueron señaladas como correctas, pues se tiene la documentación necesaria y suficiente que garantiza que no se retrase esta etapa y tampoco haga pesado el procedimiento.

Conclusiones

En este capítulo se ha mostrado todo el proceso de validación de la propuesta, proceso que ayudó a mejorar la investigación, pues el trabajo con los expertos no sólo aportó valores numéricos sobre la investigación en los modelos, sino que también brindó Análisis de los Resultados que hicieron madurar más la propuesta y luego de analizar los resultados se obtuvo una alta probabilidad de éxito de la propuesta.

Conclusiones Generales

Para llevar a cabo este trabajo se realizó un estudio teórico profundo con el objetivo de obtener información y alguna vía de solución propuesta por otras investigaciones que se asemejarán a nuestra problemática, varias fueron las ideas encontradas pero ninguna resolvía en totalidad nuestra necesidad.

El desarrollo de esta investigación tuvo gran repercusión en la preparación profesional de las autoras por los conocimientos que aportó a la misma, lo que ha permitido llegar a las siguientes conclusiones, de manera que se evidencia el cumplimiento a los objetivos propuestos:

Se realizó una propuesta de actividades encaminada hacia las clasificaciones establecidas dentro de la Universidad, así como momentos necesarios para comenzar a aplicarlas en el proyecto SUA. Se validó la propuesta mediante el método de expertos, logrando una alta probabilidad de éxito.

Recomendaciones

Comenzar a aplicar el procedimiento de diseño propuesto en todos los proyectos de la Universidad que no presentan o ejecuten un procedimiento para realizar el diseño.
Destinar un trabajador para cada actividad propuesta.

El trabajo de diploma puede ser utilizado como material de estudio por cualquier persona interesada en conocer acerca de las actividades a realizar por el diseñador de aplicaciones.

Referencia Bibliográfica

eumed.com Grupo de investigación eumednet (SEJ-309) de la Universidad de Málaga, con el apoyo de la Fundación Universitaria Andaluza Inca Garcilaso
Disponible en
http://www.eumed.net/libros/2009c/584/Descripcion%20de%20las%20metodologias%20existentes%20para%20el%20desarrollo%20de%20software.htm
http://www.willydev.net/descargas/prev/TodoAgil.Pdf'

Beck, K... "Extreme Programming Explained. Embrace Change", Pearson Educación, 1999. Traducido al español como: "Una explicación de la programación extrema. Aceptar el cambio", Addison Wesley, 2000.

Metodologías Ágiles
Amaro Calderón, Sarah Damaris, Valverde Rebaza. Jorge Carlos Trujillo – Perú 2007
Disponible en http://www.seccperu.org/files/Metodologias%20Agiles.pdf

Metodologías Ágiles en el Desarrollo de Software
José H. Canos, Patricio Letelier y Ma. Carmen Penadés DSIC -Universidad Politécnica de Valencia

Boehm, Barry W., "A Spiral Model of Software Development and Enhancement", IEEE Computer, Vol.21, No 15 (Mayo 1988), pp.61-72.

Beck, Kent, Extreme Programming Explained, Addison-Wesley the XP Series,
2000. Grupo ISSI, Metodologías Ágiles en el Desarrollo de Software, Artículo de Grupo ISSI (Noviembre 2003). Disponible en: http://issi.dsic.upv.es/tallerma/actas.pdf

Métricas en el desarrollo del Software
Querétaro, marzo de 2006 ISO 9126-3: Métricas Internas de la Calidad del Producto de Software catarina.udlap.mx/u_dl_a/tales/documentos/lis/

Fowler, Martín UML, gota a gota Addison Wesley Longman de México, SA de CV México 1.999
http://jms32.eresmas.net/tacticos/UML/UML07/UML0701.html#UML07CuandoUtilizarDiagramasPaquetes

aprendeuml.com Formación, entrenamiento y soporte Copyright 2004 Joseph Villalta Marzo
http://www.vico.org/aRecursosPrivats/UML_TRAD/talleres/mapas/UMLTRAD_101A/LinkedDocuments/SeleccionCASE_vvc.pdf

Humphrey, W. S. (1989). Managing the software process. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

ISO/IEC. (2001). Software engineering -- product quality -- part 1: Quality model. Geneva, Switzerland: ISO/IEC.

Meyer, B. (1988). Object-oriented software construction (1 ed.): Pren-tice Hall.

Miller, B. K., Hsiao, P., & Kung, C. (1999). Object-oriented architecture measures. Paper presented at the Hawaii International Con-ference on System Sciences.

Garzás, J., & Piattini, M. (2005). Ontology for micro architectural design knowledge. Software, IEEE, 22(2), 28.

BARBACCI, Mario. 2005. IEEE Argentina. IEEE. [En línea] IEEE, 22 de julio de 2005. [Citado el: 9 de febrero de 2008.] http://WWW.ieee.org.ar/noticiasdetalle.asp?IDNoticia=115.CS-08.

BOOCH, Grady. 1998. Object-Oriented Analysis and Design with applications Segunda Edición. Santa Clara: Addison-Wesley, 1998.

BOOCH, GRADY y RUMBAUGH, JAMES. 1995. Unified Method for Object-Oriented Development. S.l.: Rational Software Corporation, 1995. Overview.

CABRERO, Daniel, Garzás, Javier y PIATTINI, Mario. 2008. Técnica de Mejora del Mantenimiento de Software Basada en Valor. España: s.n., 2008.

CANOS, José H., LETELIER, patricio y PENADES, Carmen. 2003. Metodologías Ágiles en el desarrollo de Software. Universidad Politécnica de Valencia (DSIC). España: Ingeniería de Software y Sistemas de Información (ISSI), 2003.

LARMA, Craig. 2004. Applying UML and Patterns. S.l.: Prentice Hall, 2004.

PRESSMAN, ROGER S. 2001. Ingeniería del Software. Un enfoque practico. [Trad.] Darle Ince. Quinta Edición.

Jacobson, I. 1998. "Applying UML in the Unified Process" Presentación. Rational Software. Presentación disponible en http://www.rational.com/uml como UMLconf.zip

Conallen, J. "UML Extension for Web Applications 0.91" Drafted Conallen, Inc. 22-Mar-1999 Disponible en: http://www.conallen.com/technologyCorner/webextension/WebExtension091.htm

Manejo de Errores Usando Excepciones Java Ricardo Lou Torrijos 21 de Junio 2004 http://www.programacion.com/articulo/manejo_de_errores_usando_excepciones_java_98/15

Excepciones 2001 Juan Antonio de la Puente http://laurel.datsi.fi.upm.es/~ssoo/STR/Excepciones.pdf

AMEDOLA, L. 2003. La confidencialidad desde el Diseño. Proyectos de Mantenimiento. Departamento de Proyectos de Ingeniería, PMM Institute for Learning. Universidad Politécnica de Valencia. España.

García Peñalvo, Francisco José, y otros. 2004. Modelo de Calificación de Assets del repositorio Giro. Valladolid: s.n., 2004

Gracia, Joaquín. 2004 .Podredumbre del software. Ingeniero software. [En línea] 22 de agosto de 2004. [Citado el: 18 de marzo de 2008.] http://www.ingenierosoftware.com/analisisydiseño/pobredumbre.php.

Humphrey, Watts S. 2001. Introducción al proceso de software personal. Andrés Otero

Humphrey, W. S. (1989). Managing the software process. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Arregui, J.J.O. Revisión Sistemática de Métricas de Diseño Orientado a Objetos. 2005 [cited; Available from: http://is.ls.fi.upm.es/doctorado/Trabajos20042005/Olmedilla.pdf.

Anexos

Anexo 1 Reuniones de planificación y control

UCI Universidad de las Ciencias Informáticas	Minuta de reunión

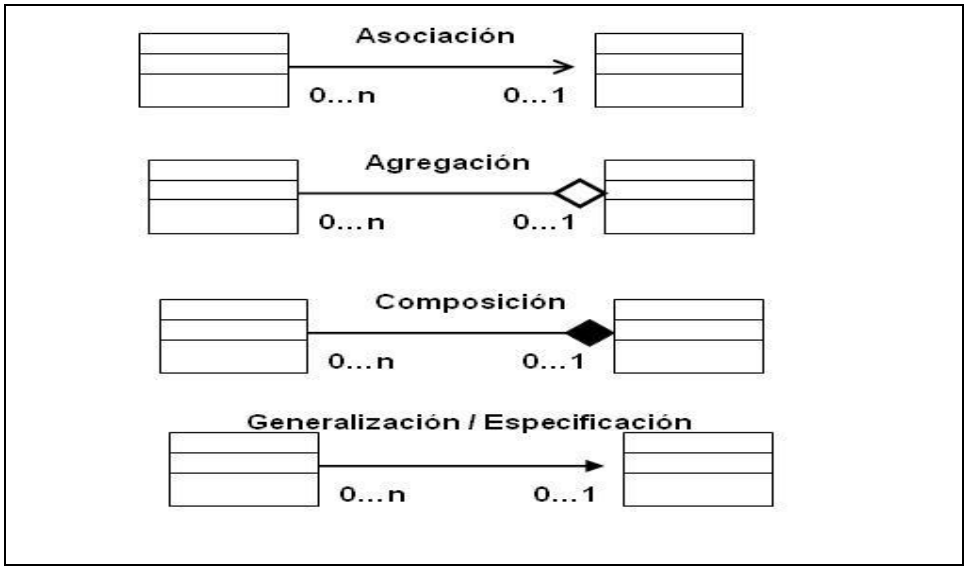
Autor	**Nombre y Apellidos**	Fecha	**DD/MM/AAAA**
Lugar	**Lugar de la Reunión**	Hora Inicio	**HH:MM**
Proyecto	**Proyecto / Subproyecto**	Hora Terminación	**HH:MM**
Asunto	**Asunto de la Reunión**		
Asistentes	**Nombre y Apellidos, correo@electronico.com** **Nombre y Apellidos, correo@electronico.com**		

Acuerdos Tomados

No	Acuerdo	Responsable	Fecha Cumplimiento

Puntos Tratados

Anexo 2 Relaciones entre clases de diseño.



Anexo 3 Documento de Definición de estándares del proceso de diseño

Documento de Definición de estándares del proceso de diseño

Definición de estándares
[Su objetivo es definir los estándares desde los que deberá partir el proceso, los cuales recogerán aspectos relacionados con.]

Métodos de trabajos.

Estilos de diseño.

Plantillas para el desarrollo de artefactos.

Modelos de calidad.

Métricas de diseño

Principios organizativos

Anexo 4 Documento de Diseño Arquitectónico

1.1 Arquitectura actual

[Describe la arquitectura del sistema que será remplazado. Si no existe un sistema previo, en esta sección se puede incluir una referencia a las arquitecturas que se adoptan en la actualidad para sistemas similares]

2.2 Arquitectura propuesta

[Documenta el modelo de diseño del sistema para el nuevo sistema.]

2.2.1. Visión global.

[Presenta la estructura global de la arquitectura del sistema y una breve descripción de la asignación de funcionalidad de cada subsistema. Descripción de la arquitectura software elegida para el sistema (patrón arquitectónico)]

2.2.2. Diseño de la Arquitectura

[En este apartado se establece la descomposición en subsistemas y las responsabilidades asignadas a cada uno. Corresponde a los productos obtenidos de las fases de análisis y diseño arquitectónico. En la sección de topología del sistema se describirán la asignación del software al hardware –diagrama de despliegue- y a los componentes – diagrama de componentes. La sección de "Gestión de la Persistencia" describe la infraestructura de almacenamiento requerida y los datos que serán almacenados por el sistema. La sección de "Aspectos Globales y de Seguridad" se describen las políticas de seguridad y acceso que serán establecidas en el sistema – mecanismos de autentificación, encriptación, certificados, etc.- y los aspectos de sincronización, concurrencia e inicialización de subsistemas. Por último, la sección de "Aspectos de rendimiento y tamaño" establecerá los acuerdos de nivel de servicio establecidos con el usuario en relación a tiempos de respuesta aceptables, productividad y capacidad de almacenamiento requerido.]

Anexo 5 Documento de Evaluación de Métricas de Diseño

Documento de Evaluación de Métricas de Diseño

Evaluación

[Su objetivo es evaluar métricas de calidad orientadas al diseño de software]

Familia de métricas empleadas

[Especificar la familia de métricas empleadas y el resultado de los cálculos realizados]

Anexo 6 Documento de Estudio del Estado del Arte de Reutilización

1 Marco Teórico

[Es la posición o corriente que asume el investigador ante el tema como fundamentación teórica para la justificación de su trabajo científico y la formulación de hipótesis. Aquí se concreta el trabajo de la preparación previa y se hace un análisis detallado del estado del arte, de lo que hay en la bibliografía nacional e internacional sobre el tema que se investiga, destacando la posición del investigador al respecto, haciendo su valoración crítica de cada una de las posiciones que existen.]

1.1 Alcance

[La posibilidad de reutilizar código para minimizar el tiempo y los costes de desarrollo].

1.2 Análisis sobre elementos reutilizables

[Evaluación de la compatibilidad de lo que se reutilice respecto a los principios y estándares definidos por el equipo de diseño y la organización en sentido general].

2. Tendencia tecnológica de avanzada

[Describir, cuando proceda, las tendencias tecnológicas de avanzada, los titulares de las tecnologías, las soluciones protegidas en el país cuya explotación infringiría los derechos de los titulares tanto en Cuba como en otros países.]

2.1 Modelo Teórico

[Es una representación ideal del objeto de investigación de acuerdo con la concepción que se tiene de la investigación a partir del análisis de la bibliografía, las indagaciones realizadas y la experiencia del investigador y da origen a la hipótesis que es el núcleo de ese modelo teórico.]

Anexo 7 Compendio de Patrones de Diseño

Documento Compendio de Patrones de Diseño

1 Compendio de patrones

[Definir el paquete de patrones de diseño que podrán emplearse en la construcción del diseño, sobre la base de la tecnología y los paradigmas que fungen como pilares para el desarrollo del sistema en sentido general].

Anexo 8 Repositorio de Reutilización

Documento Repositorio de Reutilización

1Almacén de modelos de software

[Aquí se debe almacenar funcionalidades imprescindibles, documentos, todo lo que realicen los trabajadores del proyecto y que sean lo suficientemente abstractos como para poder ser usados en más de un sistema.]

Anexo 9 El informe General del proceso de Diseño

Informe General del Proceso de diseño

Equipo de Diseño

[Como esta conformado el equipo de diseño, como trabajan.]

Artefactos

[Visión general de los artefactos más importantes.]

Anexo 10 Registro de Revisión Técnica

Registro de Revisión Técnica

Registro de los resultados

[Su objetivo es registrar los resultados de haber realizado una revisión al diseño o a una sección de este.]

Anexo 11 Modelo de Subsistemas

1. Diseño de Subsistemas

1.1. Subsistemas Específicos

1.1.1. [Nombre del Subsistema Específico 1]

Propósito

[Descripción de por qué el subsistema existe. Este atributo debe dar la razón de la creación del subsistema. Como ser la funcionalidad específica y los requerimientos de performance por los cuales fue creado. También describe requerimientos especiales que se deben lograr con él que no están incluidos en la especificación de requerimientos del software.]

Función

[Declara qué hace el subsistema. Establece la transformación aplicada a las entradas del subsistema para producir la salida deseada.]

Subordinados

[Se identifican los objetos de diseño y subsistemas de diseño que componen el subsistema que se describe. Se propone representar esta información con un diagrama de paquetes.]

Dependencias

[Descripción de la relación de este subsistema con otros subsistemas de diseño. Identifica los usos o la necesidad de presencia de la relación para un subsistema. Describe la naturaleza de cada interacción incluyendo características como tiempo y condiciones de la interacción. Estas interacciones pueden involucrar la iniciación, orden de ejecución, datos compartidos, creación, duplicación, uso o almacenamiento. Se propone representar esta información con una tabla de dependencias.]

Subsistema del que depende	Naturaleza de interacción	Características
[Identificación del subsistema del que depende]	*[Condiciones para que se realice la interacción]*	*[Características de la interacción, como ser, pasaje de parámetros, mensajes, datos compartidos, etc.]*

Interfases

[Descripción de cómo los otros subsistemas interactúan con éste. Describe los métodos de interacción y las reglas que gobiernan éstas interacciones. Los métodos de interacción incluyen los mecanismos para invocar o interrumpir al subsistema, para comunicarse a través de parámetros, áreas comunes de información o mensajes, y para acceso directo a información interna. Las reglas que gobiernan la interacción incluyen el protocolo de comunicaciones, formato de la información, valores aceptables, y el significado de cada valor.

Este atributo también debe dar una descripción de los rangos de entrada, el significado de entradas y salidas, el tipo y formato de cada entrada y salida, y los códigos de error de salida. Se propone representar esta información con una lista del tipo:]

Interacción	Método de interacción	Reglas de la interacción
[Objeto que implementa la interacción]		

1.1.2. [Nombre del Subsistema Específico 2]

...

1.2. Subsistemas de Soporte

1.2.1. [Nombre del Subsistema de soporte 1]

Propósito
Función
Subordinados
Dependencias
Recursos
Interfases

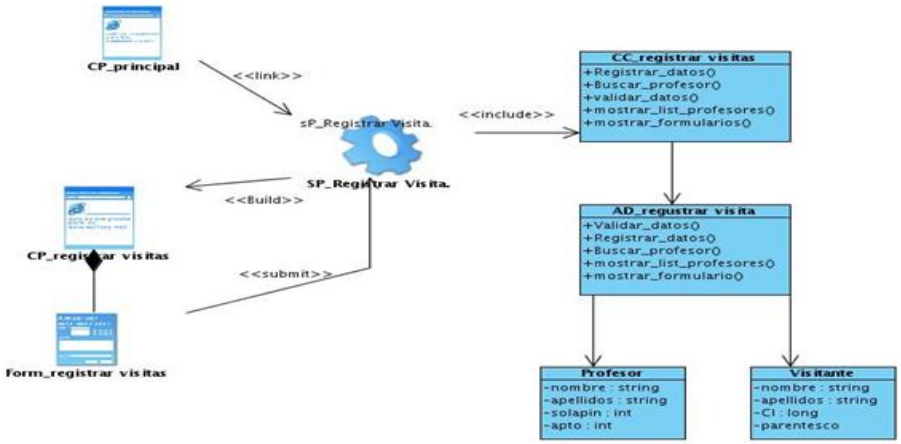
1.2.2. [Subsistema de soporte 2]

Anexo 12 Diagramas de clases del Diseño

1 Diseño de Clases

[Su objetivo es mostrar como están diseñadas las clases del sistema y sus interacciones].

Ej.



1.2 Descripción

[Breve descripción de lo que hace cada método por clase.]

Nombre	Descripción
[Nombre del método]	*[Breve descripción de lo que hace el método.]*

Anexo 13 Diagramas de Estado e Interacción

Diagrama de Interacción

[Su objetivo es ilustrar el modo en el que los objetos interaccionan por medio de mensajes y se dividen en dos tipos de diagramas: los de secuencia y los de colaboración.]

Diagramas de Secuencia

[Su objetivo es mostrar una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo. El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo. Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren.]

Ej.

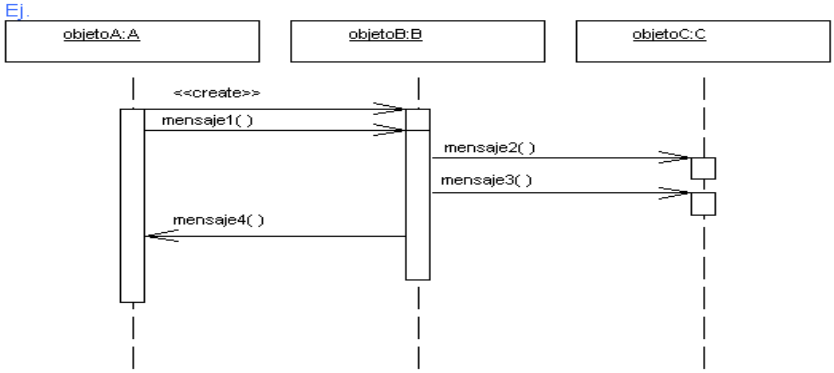


Diagrama de Colaboración

[Su objetivo es ilustra las iteraciones entre objetos en un formato de grafo o red, en el cual los objetos se pueden colocar en cualquier lugar del diagrama].

Ej.

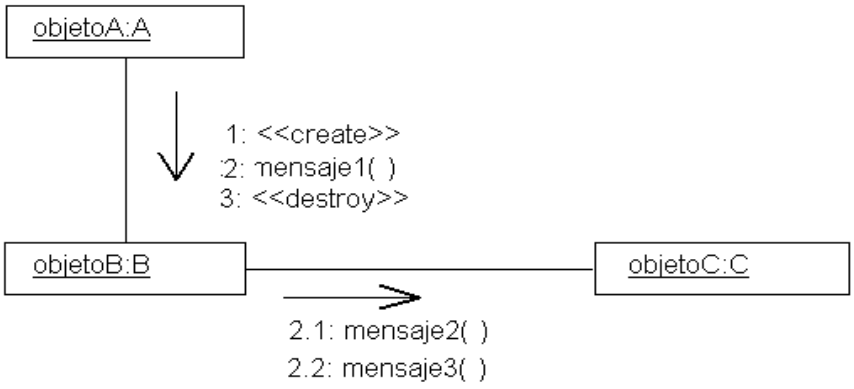
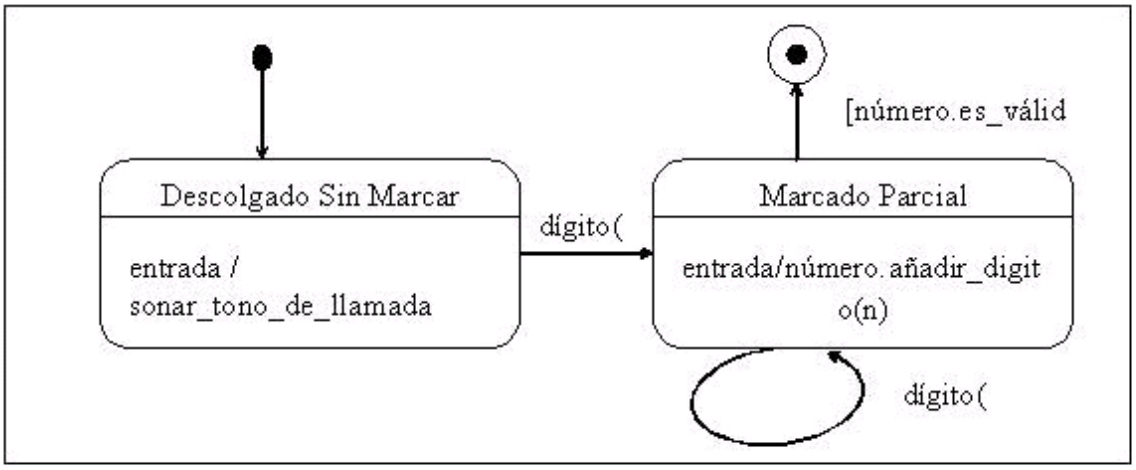


Diagrama de Estado

[*su objetivo es representar el ciclo de vida de un objeto: los eventos que le ocurren, sus transiciones, y los estados que median entre estos eventos****].***

Ej.



Anexo 14 Mecanismo de Excepciones.

Tratamiento de excepciones

[Se deberán definir el modelo de clases que deberá al tratamiento de excepciones, además una vista de las clases que definen dicho mecanismo.]

Ej.

```
class exception {
   virtual void message () {cout << "exception thrown\n";}
};
class integer_constraint_error : public exception {
   int lower_limit;
   int upper_limit;
   int value;
  public:
   integer_constraint_error (int L,int U, int V){
      lower_limit = L;
      upper_limit = U;
      value = V;
   };
   virtual void message () {
      cout << "constraint error\n";
   };
};
class actuator_dead : public exception {...};
```

```
try {
   /* instrucciones */
}
catch (exception E) {
   /* manejador para el tipo excepción y todos sus derivados */
   E::message(); /* escribe el mensaje según el tipo de E */
}
```

Anexo 15 Especificación de mantenimiento.

Especificación de mantenimiento

Políticas de mantenimiento

[Definida una política de mantenimiento adecuada y orientada a optimizar costes.]

Planificación del mantenimiento

[Su objetivo es registrar los cambios realizados y las causas que los provocaron.]

Anexo 17 Encuesta a Expertos para la valoración de la propuesta.

Compañero (a):

En la presente tesis se propone definir las actividades a realizar por el rol de Diseñador de Aplicaciones en el proyecto Sistema Único de Aduana (SUA). Por sus conocimientos y experiencia en el tema tratado ha sido seleccionado para formar parte del equipo de expertos que se utilizan para validar el trabajo.

A continuación se presentan un conjunto de interrogantes de las cuales usted debe dar su criterio. Valore el grado de factibilidad de las mismas de acuerdo a la siguiente escala:

5 – Muy bueno
4 – Bastante Adecuado
3 – Adecuado
2 – Poco Adecuado
1 – No Adecuado

Criterios a Evaluar	Peso que usted le otorga (de 1 hasta 5)				
	1	2	3	4	5
1. **Diseñador principal**					
Definición de los estándares de diseño					
Diseño Arquitectónico					
Conformación del expediente de diseño					
Reuniones de planificación y control					
2. **Diseñador de la lógica funcional**					
Diseño de sub-sistemas					
Diseño de clases					
Diseño de la lógica funcional ***(Diagramas de Estados e interacción)***					
3. **Diseñador del modelo de datos**					
Análisis de las clases de diseño					
Diseño del modelo de datos					
4. **Diseñador de mecanismo de respaldo**					
Diseño de mecanismos de recuperación de errores					
Diseño de mecanismos de manipulación de Excepciones					
Diseño de Modelo de Persistencia					
Definición de mecanismos de mensajes a usuarios					
Diseño de mecanismo de seguridad					
Diseño de mecanismos de optimización de recursos					
5. **Responsable de integración**					
Definición de estándares de integración					
6. **Responsable de mantenimiento**					
Mantenimiento sistemático del diseño					
7. **Revisor Técnico**					
Revisiones Técnicas Formales					
8. **Analista de reutilización**					
Identificación de elementos de reutilización					
Análisis de uso de Patrones de diseño					
9. **Responsable de repositorio de reutilización**					
Administración del repositorio de reutilización					
10. **Responsable de Refactorización**					
Refactorización continua					
Documentar la actividad de Refactorización					
11. **Responsables de calidad del diseño**					
Evaluación del cumplimiento de Estándares de diseño					