DISEÑANDO EL SISTEMA

QUE ES DISEÑO

Diseño es el proceso creativo de transformación del problema en una solución; la descripción de una solución también se denomina diseño.

Se utiliza la especificación de requerimientos para definir el problema. Entonces, podemos declarar que algo es una solución a un problema si satisface todos los requerimientos planteados en la especificación. En muchos casos, el número de soluciones es prácticamente ilimitado. Se dice que la naturaleza de una solución puede cambiar al describirse o al implementarse.

Diseño conceptual y diseño técnico

El diseño realmente es un proceso iterativo que consta de dos partes. Primero se elabora un diseño conceptual o un diseño del sistema que le dice al cliente, exactamente, que hará dicho sistema. Una vez que el cliente aprueba el diseño conceptual, se vuelca este diseño en un documento mucho más detallado, el diseño técnico, que permite que los constructores comprendan el hardware y software concretos que se necesitan para resolver el problema del cliente. El proceso es iterativo por que en la realidad los diseñadores se mueven de una a otra de las diferentes actividades que abarcan la comprensión de los documentos , fomulación de soluciones posibles, la comprobación de algunos aspectos de soliciones como la factibilidad, la presentación de alternativas a los clientes y la documentacion de diseño para los programadores.

Ambos documentos de diseño describen el mismo sistema. En consecuencia el diseño conceptual se concreta en las funciones del sistema y el diseño técnico describe la forma que tomará el sistema.

Un sistema que da definido por sus límites, atributos, entidades y relaciones. El diseño conceptual describe cada uno de estos aspectos del sistema respondiendo a las siguientes cuestiones:

- ¿De donde provienen los datos?
- ¿Qué les sucede a los datos en el interior del sistema?
- ¿Qué les parecerá el sistema a los usuarios?
- ¿Qué elecciones se ofrecerá a los usuarios?
- ¿Cuál es la temporización de los eventos?
- ¿Qué aspecto tendrán los informes y pantallas?

El diseño conceptual describe el sistema en un lenguaje que el cliente puede comprender en lugar de los términos técnicos o jerga de computación.

Un buen diseño conceptual debe contener las siguientes características:

- Se describe en el lenguaje del cliente.
- No contiene expresiones de jerga técnica.
- Describe claramente las funciones del sistema.
- Es independiente de la implementación.
- Está vinculado con los documentos sobre requerimientos.

El diseño técnico describe la configuración del hardware , las necesidades del software, las interfaces de comunicación, las entradas y salidas del sistema, la arquitectura de red y cualquier otro aspecto que inicia en las transformación de los requerimientos en una solución para el problema del cliente. Por lo general incluye los siguientes ítems:

- Una descripción de los principales componentes del hardware y de sus funciones.
- La jerarquía y funciones de los componentes del software.
- Las estructuras de datos y el flujo de los datos.

DESCOMPOSICIÓN Y MODULARIDAD

Diseñar un sistema es determinar un conjunto de componentes y de interfaces entre componentes que satisfagan un conjunto específico de requerimientos. Wasseerman (1995) sugiere que los diseños se crean bajo cinco enfoques:

1. Descomposición modular. Esta construcción se basa en la asignación de funciones a los componentes comenzando con las descripciones de alto nivel, elabora explicaciones sobre lo que se va a implementar y de cómo estará organizado cada componente y su relación con otros componentes.
2. Descomposición orientada por datos. En este caso el diseño se basa en las estructuras externas de los datos.
3. Descomposición orientada por eventos. Este diseño se basa en los eventos que el sistema debe manejar y usa la información sobre el modo en que estos eventos cambian el estado del sistema.
4. Diseño “de afuera – hacia adentro”. Es un enfoque de caja negra basado en las entradas en el sistema que realiza el usuario.
5. Diseño orientado a objetos. Este diseño identifica clases de objetos y sus interrelaciones.

Independientemente del enfoque de diseño adoptado, cada tipo de descomposición separa el diseño en partes componible denominadas módulos o componentes. Se dice que un sistema es modular cuando cada una de las actividades la realiza exactamente un único componente, y en donde además están bien definidos cada una de sus entradas y salidas.

El componente esta bien definido solamente cuando cada salida es un resultado del funcionamiento, y cuando ninguna entrada pasa a ser salida sin haber sido trasformada de algún modo por el componente.

ESTILOS ARQUITECTONICOS Y ESTRATEGIAS

Shaw y Garlan (1996) sugieren que la arquitectura del software es el primer paso en la producción de un diseño de software y sugieren los siguientes niveles de diseño:

1. Arquitectura, asocia las capacidades del sistema identificadas en las especificación de requerimientos, con los componentes del sistema que se implementarán.
2. Diseño de código, comprende algoritmos y estructuras de datos, y los componentes son primitivas del lenguaje de programación, tales como números, caracteres, punteros e hilos de control.
3. Diseño ejecutable, remite el diseño de código en un nivel de detalle todavía más bajo, trata sobre la asignación de memoria, los formatos de los datos o la configuración de bits.

Tuberías y filtros

El sistema del tipo tuberías y filtros se utiliza cada vez que se compila un programa. Los filtros están en secuencia lineal: análisis de léxico, comprobación gramatical, análisis semántico y generación de código.

Los sistemas de tipo tuberías y filtros tiene algunas propiedades importantes:

- Los diseñadores pueden entender todos los efectos del sistema sobre la entrada y salida, así como la composición de los filtros.
- Dado que pueden vincularse dos filtros cualesquiera, son fácilmente reutilizables en otros sistemas.
- La evolución de los sistemas es simple, pueden incorporarse nuevos filtros y removerse los antiguos con relativa sencillez.
- Debido a la independencia de los filtros, los diseñadores pueden simular el comportamiento del sistema y analizar propiedades tales como el rendimiento.
- Se pueden ejecutar de forma concurrente.

Diseño orientado a objetos

El diseño basa do en objetos tiene dos características significativas: el objeto puede preservar la integridad de la representación de los datos, y la representación de los datos puede ocultarse a los restantes objetos del sistema. La última característica hace que la implementación sea fácil de cambiar sin perturbar el resto del sistema.

Sin embargo un objeto debe conocer la existencia de otros a fin de poder interactuar con ellos.

Invocación implícita

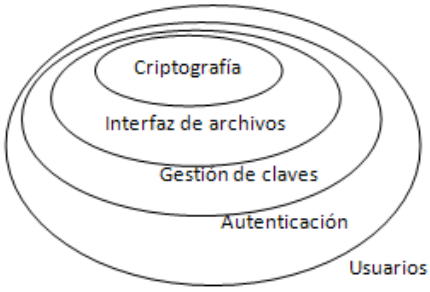
Este modelo está determinado por eventos y se basa en el concepto de difusión. En lugar de invocar directamente a un procedimiento, un componente enuncia que se ha producido uno o más eventos. Luego, otros componentes pueden asociar un procedimiento con dichos eventos y el sistema invoca a todos los procedimientos registrados.

El intercambio de datos en este tipo de sistemas debe hacerse por medio de datos compartidos en un repositorio.

Este tipo de diseño se utiliza frecuentemente en redes de conmutación de paquetes, o con sistemas basados en actores, en bases de datos para asegurar la consistencia, y en interfaces de usuario para separar la representación de los datos de las aplicaciones que los manejan.

Estratificación (modelos y capas)

Los modelos estratificados están organizados de la siguiente manera:



Arquitectura Estratificada de Seguridad

Las capas son jerárquicas; cada una presta servicios a la capa inmediata superior y actúa como cliente de la que queda encerrada. En algunos sistemas, todas las capas tienen acceso a una parte o a la totalidad de los servicios de las otras capas; en otros, el acceso de una capa queda limitado sólo a las adyacentes. El diseño incluye protocolos que establecen como interactuará cada par de capas.

En la parte más interna, criptografía, se encuentran las funciones de encriptar y descifrar una clave que se utiliza en el esquema básico de de encriptación del sistema. En la segunda capa, la

interfaz a nivel de archivo, se encripta y desencripta un archivo. El tercer estrato soporta la administración de claves, permite que un componente aplique su firma a un archivo, verifica la firma y hace el cómputo del algoritmo de hash que permite el acceso al archivo. Finalmente la cuarta capa se encarga de la autenticación. Esta capa maneja un archivo de claves de acceso que se almacenan de forma encriptrada, y por otra parte es la que solicita la identidad del usuario y la clave de acceso asignada.

Repositorios

Un repositorio tiene dos clase de componentes: un almacenamiento central de datos y un conjunto de componentes que operan sobre el almacenamiento para almacenar, recuperar o actualizar información.

Muchos sistemas están organizados como repositorios: bibliotecas de componentes reutilizables, grandes bases de datos y motores de búsqueda, entre otros. Una ventaja de este tipo de arquitectura es su accesibilidad; la representación de los datos, permite que estén al alcance diferentes fabricantes de modo que éstos pueden desarrollar sus propias herramientas de acceso al repositorio, sin embargo estas características conllevan una desventaja: los datos compartidos deben adoptar una forma aceptable para todas las fuentes de conocimiento, a un cuando estas fuentes son en sí mismas radicalmente diferentes.

Interpretes

Un interprete toma una cadena de caracteres denominada pseudocódigo y la convierte en código verdadero que luego se ejecuta, el interprete esta compuesto de cuatro partes:

1. Una memoria para contener el pseudocódigo a ser interpretado.
2. Un motor de interpretación para convertir el pseudocódigo y simular el programa que representa.
3. El estado actual del motor de interpretación.
4. El estado actual del programa que se está simulando.

Control de procesos

Los sistemas de control de procesos se caracterizan no solo por el tipo de componente sino también por las relaciones que mantienen entre ellos. El propósito de un sistema de control de procesos es mantener las propiedades específicas de las salidas del proceso dentro o cerca de los valores de referencia especificados, denominados puntos fijos o valores de calibración.

El sistema de control basado en software más común comprende uno de los dos tipos de lazo cerrado: retroalimentación o prealimentación. Un sistema retroalimentado mide una variable controlada, como puede res la temperatura y ajusta el proceso de concordancia para mantener el valor de la variable cerca o dentro de la calibración. En la prealimentación el sistema intenta anticipar los futuros efectos sobre la variable controlada, midiendo otras variables de proceso que pueden actuar como buenos indicadores.

PROBLEMAS EN LA CREACIÓN DEL DISEÑO

Modularidad y niveles de abstracción

Combinando componentes modulares y niveles de abstracción se obtienen varias vistas diferentes del sistema. Los componentes de más alto nivel dan la oportunidad de visualizar la solución como un todo, ocultando detalles que de lo contrario podrían desviar la atención. En una descomposición funcional, esta vista puede mostrar las funciones principales que el sistema debe realizar. Si se requiere más detalle acerca de una porción del sistema, será suficiente desplazar la atención hacia los niveles menores de abstracción.

Al descender a un nivel más bajo, la modularidad proporciona la flexibilidad que se necesita para comprender que debe hacer el sistema, seguir el flujo de datos y las funciones a través de el y descubrir las áreas de complejidad.

Diseño colaborativo

En la mayor parte de los proyectos el diseño no está creado por una sola persona, por lo común se trabaja en equipo, asignando diferentes partes del diseño a diferentes personas.

Uno de los mayores problemas para llevar adelante un diseño en equipo es cómo encarar las diferencias en experiencia, comprensión y preferencias personales de los integrantes del equipo.

Cuando la industria del software procure reducir costos y maximizar la productividad apelando el desarrollo de software mediante grupos colaborativos ubicados por el mundo, seguramente aumentará la importancia de la comprensión del comportamiento grupal. Yurdon (1994) identifica cuatro etapas en este tipo de desarrollo distribuido:

1. En la primera etapa, el proyecto se realiza en un único sitio con desarrolladores provenientes de otros países, pero instalados en el lugar.
2. En la segunda etapa, los analistas locales determinan los requerimientos del sistema. Luego estos requerimientos se entregan a grupos foráneos de diseñadores y programadores para continuar el desarrollo.
3. En la tercera etapa, desarrolladores externos construyen productos y componente genéricos que son usados mundialmente.
4. En la cuarta etapa, los desarrolladores externos construyen productos que saquen ventajas de sus áreas individuales de experiencia.

Diseño de la interfaz de usuario

El diseño de la interfaz de usuario puede tener dificultades, dado que personas diferentes tienen estilos diferentes de percepción, comprensión y trabajo.

Marcus (1993) presenta muchas de las cuestiones involucradas en el diseño de la interfaz de usuario. Puntualiza que una interfaz debe de atender algunos elementos clave:

- Metáforas: términos fundamentales, imágenes y conceptos que deben ser reconocidos y aprendidos.
- Un modelo mental: la organización y representación de datos, funciones, tareas y roles.
- Las reglas de navegación para el modelo: como moverse entre datos, funciones, actividades y roles.
- Aspecto: las características de presentación del sistema que transmiten información al usuario.
- Sensación: las técnicas de interacción que proporcionan una experiencia atractiva apara el usuario.

La meta de estos elementos y de la interfaz de usuario es contribuir a que el usuario consiga un rápido acceso al contenido de sistemas complejos, sin perdida de comprensión mientras se desplaza a través de la información".

Pero a fin de diseñar interfaces efectivas y confortables, debemos considerar dos cuestiones claves: cultura y preferencia.

- **Cuestiones culturales.** Muchos grupos de usuarios están integrados por personas provenientes de culturas diferentes; por ende, no se puede suponer que creando un

sistema para cada una de las grandes culturas se asegurará la adopción del sistema o su uso correcto. Para hacer que el sistema sea multicultural, las interfaces podrían diseñarse en dos etapas. Primero se eliminan todas las referencias culturales específicas o brechas, para hacer que la interfaz resulte tan "internacional" como sea posible.

La segunda etapa toma el diseño ya libre de brechas y lo amolda a las culturas que lo van a utilizar. Es importante recordar que la cultura es determinada no sólo por la nacionalidad, sino también por la región, sexo, profesión, edad o corporación.

- **Preferencias del usuario.** Algunos aspectos del diseño dependen de las preferencias del usuario, ya sea solo o como miembro de un grupo de trabajo.

Concurrencia

En muchos sistemas las acciones se llevan a cabo concurrentemente, es decir, no secuencialmente.

Los sistemas secuenciales por lo común utilizan una corriente simple de ejecución para controlar los eventos, pero los sistemas concurrentes exigen diseños mucho más complejos. Uno delos mayores problemas de los sistemas concurrentes es asegurar la consistencia de los datos entre componentes que se ejecutan al mismo tiempo.

Una forma de manejar la concurrencia es establecer una cantidad de tiempo asignado para la ejecución de cualquier acción. De esta forma, una temporización cuidadosa puede asegurar que acciones no interfieran unas con otras.

Idealmente se debe aspirar a diseñar los sistemas de manera que sean correctos, sin depender de la temporización de las solicitudes. Afortunadamente, hay dos maneras de haces esto: monitoreo y guardianes.

- Un monitor es un objeto abstracto o componente que controla la exclusión mutua de un proceso particular. Para prevenir problemas de temporización el monitor habitualmente se complementa con un probador de condición, para garantizar que las condiciones son las correctas para invocar el procesos solicitado.
- Guardianes. Un guardia es una tarea que se ejecuta permanentemente; su único propósito es controlar el acceso a un recurso encapsulado (es decir, a una estructura de datos o un proceso), el guardián también incluye un probador de condiciones que lo asiste en la toma de decisiones de control de acceso.

Patrones de diseño y reutilización

Un patrón de diseño nombra, abstrae e identifica aspectos claves de una estructura común de diseño que lo hacen útil para la creación de un diseño reutilizable. El patrón de diseño identifica las clases e instancias participantes, sus roles y colaboraciones de responsabilidades.

CARACTERISTICAS DE UN BUEN DISEÑO

Independencia de los componentes

Para reconocer y medir el grado de independencia de de los componentes de un diseño se aplican dos conceptos: acoplamiento y cohesión.

- **Acoplamiento.** Se dice que dos componentes están altamente acopados cuando existe mucha dependencia entre ellos. Los componentes poco acoplados tienen algunas dependencias, pero las interconexiones entre ellos son débiles. Los componentes no acoplados tienen interconexiones con el resto; son completamente independientes.

- **Cohesión.** En contraste con la interdependencia entre componentes, la cohesión se refiere al grado de "adhesivo" interno con que se ha construido el componente. Cuanto más cohesivo es un componente, más relacionadas están las partes internas de él, tanto entre ellas como en relación al propósito global.

Identificación y tratamiento de excepciones

En el diseño se incluirá el manejo de excepciones de modo que el sistema dirija cada excepción en una forma satisfactoria que no degrade las funciones del sistema.

Las excepciones más comunes son:

- Fracaso al proporcionar un servicio.
- Proporcionar un servicio o datos erróneos.
- Corrupción de datos.

Cada una de las excepciones que se identifican puede manejarse según uno de los siguientes tres criterios:

1. Reintentar, se restaura el sistema a su estado precio y se intenta realizar el servicio utilizando una estrategia diferente.
2. Corregir, se restaura el sistema a su estado previo y se intenta realizar el servicio utilizando la misma estrategia.
3. Informar, se restaura el sistema a su estado previo, se informa el problema a un componente de tratamiento de error y no se proporciona el servicio.

Existen varias técnicas que pueden usarse para detectar excepciones mientras el código está ejecutando:

- Suma y dígitos de control, para doble comprobación de exactitud de datos y cálculos.
- Vínculos redundantes, incluyendo punteros hacia atrás y hacia delante.
- Temporizadores.

Prevención de defecto y tolerancia.

¿Cómo se produce los defectos? Cuando una persona comete un error, este error humano da como resultado un defecto en algún producto del software.

Distinguimos defectos de fallas. Una falla es un desvío del sistema respecto de su comportamiento requerido.

Es importante destacar que no todo defecto corresponde a una falla, dado que las condiciones bajo las cuales un defecto resulta en una falla del sistema pueden no producirse jamás.

Una de las características de un buen diseño es la forma en que previene o tolera los defectos. En lugar de operar hasta que el sistema falle para corregir el problema, los diseñadores pueden anticipar lo que puede ocurrir y construir el sistema para que reacciones de manera aceptable.

- **Detección activa de defectos.** Cuando se diseña un sistema que espera hasta que ocurra una falla durante la ejecución, se está poniendo en práctica la detección pasiva de defectos. Sin embargo, si se controla periódicamente la presencia de síntomas de defectos, se intenta anticipar cuando se producirán las fallas, se está realizando una detección activa de fallas.
- **Corrección de defectos.** Una vez descubierto el defecto, se le debe corregir. La corrección de defectos es la compensación del sistema ante la presencia de un defecto. Por lo general, la corrección de defectos subsana el daño producido, así como modifica el producto para eliminar tal defecto. Se puede optar entre detener el sistema cuando el defecto lo afecta de algún modo o simplemente registrar la

existencia de la falla, apuntar el estado del sistema en el momento de producirse la falla y arreglar el daño después.

TECNICAS PARA LA MEJORA DEL DISEÑO

Reducción de la complejidad.

Cuando se crea un diseño se desea simplificar su estructura tanto como sea posible sin cambiar la naturaleza de la solución. Esta simplificación permite comprender la solución más fácilmente. Como los atributos de calidad superior del diseño se van trasladando el resto del proceso de desarrollo, el software y el hardware deberían ser más fáciles de diseñar, construir y mantener.

Diseño por contrato

Meyer propone un enfoque particular para el diseño denominado diseño por contrato, para contribuir a asegurar que el diseño satisface sus especificaciones. comienza considerando un sistema software como un conjunto de componentes que se comunican, y cuya interacción está basada en una especificación exactamente definida de lo que se supone que hace cada componente. Estas especificaciones, denominadas contratos, ordenan como un componente interactúa con otros componentes y sistemas. Esta especificación no puede garantizar la exactitud de un componente, pero establece una buena base para la prueba y la validación.

Diseño con prototipos

El prototipado proporciona muchas ventajas en al etapa de diseño. En la etapa de diseño, un prototipo de factibilidad permite determinar si la solución propuesta realmente resolverá el problema en trato. Por lo tanto, el prototipo de diseño alienta a la comunicación entre el grupo y con los clientes para explorar áreas de incertidumbre que se presentan a medida que se piensa como diseñar una solución. De esta forma se resuelven muchas cuestiones ante de dar comienzo a la codificación, y se evita la creación de problemas adicionales durante las pruebas.

Casi siempre, un prototipo omite muchos defectos de los detalles de funcionalidad y rendimiento del sistema real, lo que permite acotarlo estrechamente a un aspecto particular del sistema y tener mayor comprensión de él.

Análisis de árbol de defectos

El análisis de árbol de defectos es un método originalmente desarrollado para el programa misilístico US Minuteman, razona sobre el diseño, ayudándonos a descomponerlo y a ver las situaciones que podrían llevar a una falla.

Los árboles de defectos se construyen para representar el camino lógico que lleva del defecto a su causa., estos árboles se usan para dar soporte a la corrección de defectos o a la tolerancia, dependiendo de la estrategia de diseño que se haya elegido.

Se comienza el análisis identificando las posibles fallas. Aunque la identificación tiene lugar durante el diseño, se consideran las fallas que podrían ser afectadas por el diseño, la operación y el mantenimiento. Se utiliza un conjunto de palabras guía, para ayudar a comprender de qué manera el sistema podría apartarse de su comportamiento esperado.

EVALUACIÓN Y VALIDACIÓN DEL DISEÑO

Una vez diseñado el sistema, se lo comprueba en dos formas diferentes. En primer lugar se asegura que el diseño satisface todos los requerimientos especificados por el cliente. Este procedimiento se conoce como validación del diseño. En entonces nos orientamos a la calidad

del diseño: la verificación involucra asegurar que se han incorporado las características del buen diseño.

Validación matemática

Asociado a cada proceso existe un conjunto de entradas, un conjunto de salidas esperadas y un conjunto de aserciones acerca del proceso mismo. Entonces, para cada uno de estos procesos se demuestra que:

- Si el conjunto de entradas está formulado correctamente, se transforma propiamente en el conjunto de salidas esperadas.
- El proceso termina sin fallas.

Medición de la calidad del diseño

Card y Glass (1990) puntualizan que la complejidad del diseño realmente comprende dos aspectos: la complejidad intrínseca del componente y la complejidad de las relaciones entre componentes. Definen una medida de la complejidad de un sistema que combina ambos criterios. De acuerdo con estos investigadores, la medida final de complejidad es la suma: C=S+D donde S es la complejidad estructural (entre componentes) y D es la complejidad de los datos (dentro del componente).

Comparación de diseños

La medición de varios atributos de diseño es útil no solo para prevenir defectos sino también para comparar dos diseños.

Una sola especificación varios diseños. Para ver cómo se pueden aplicar diferentes estilos de diseño en la solución de un mismo problema.

Shaw y Garlan presentan cuatro diseños arquitectónicos diferentes para implementar KWIC: datos compartidos, tipos abstractos de datos, invocación implícita y tubería-filtro.

Cada diseño tiene aspectos positivos y negativos. En consecuencia se necesita un método para la comparación de diferentes diseños que permita elegir el mejor para cada propósito.

Tablas de comparación. Shaw y garlan (1996) comparan los cuatro diseños construyendo una tabla donde se consideran los atributos más importantes. Cada fila representa un atributo y hay una columna para cada estilo de diseño. Un signo menos en la celda significa que el atributo representado por la fila no es un diseño para esa columna; un signo más significa que ese diseño tiene el atributo.

Otras características podrían haberse elegido o tenido en cuenta incluyen:

- Modularidad
- Facilidad de prueba
- Facilidad de mantenimiento
- Eficiencia
- Facilidad de comprensión
- Facilidad de modificación
- Consistencia

Una vez obtenida una lista completa de los atributos que se consideran importantes, se asocia un grado de importancia con cada uno de ellos, asignando un peso que corresponde a su prioridad.

Revisiones del diseño

Cuando el diseño se completa, se mantiene reuniones con los diferentes clientes para revisarlo antes de avanzar en el desarrollo. El proceso de revisión se realiza en tres etapas en correspondencia con los pasos de proceso de diseño.

Revisión del diseño. Los clientes y usuarios se reúnen para validar el diseño conceptual, es decir, para asegurarse de que todos los aspectos relativos a los requerimientos han sido apropiadamente contemplados en el diseño. Para hacerlo, se invitara a participar de la revisión a ciertas personas claves:

- Cliente (s), quien ayudo a definir los requerimientos del sistema.
- Analista (s), quien colaboro para definir requerimientos del sistema.
- Usuario (s) potencial (es) del sistema
- Diseñador (es) del sistema.
- Un moderador
- Un secretario/a
- Otros desarrolladores interesados, pero no involucrados en el proyecto.

Revisión critica del diseño. Una vez que el cliente está satisfecho con el producto propuesto, es tiempo de llevar a cabo una revisión crítica del diseño, donde se presenta una vista general del diseño técnico. Los participantes en esta revisión son:

- Analista (s), que colaboraron para definir requerimientos del sistema.
- Diseñador (es) del sistema
- Un moderador
- Un secretario/a
- Diseñador (es) de programa para este proyecto
- Un probador de sistemas
- Un analista que escribirá la documentación del sistema
- Otros desarrolladores interesados, que no estén involucrados de otra forma en este proyecto.

Revisión del diseño del programa. Cuando el diseño técnico resulta satisfactorio, los diseñadores de programa estarán en posición de interpretarlo como el conjunto de descripciones de diseño para los componentes reales, que deben ser codificados y probados. El equipo de revisión en este caso incluye:

- Analistas, que produjeron los requerimientos del sistema.
- Diseñadores del sistema.
- Diseñadores de programas.
- Un probador de sistemas.
- Un analista que describirá la documentación del sistema.
- Desarrolladores
- Un moderador
- Un secretario/a
- Diseñador (es) de programas para este proyecto.
- Otros desarrolladores interesados, que no estén involucrados de otra forma en el proyecto.

Valor de las revisiones del diseño. En cada tipo de revisión de diseño, la audiencia puede presentar varias preguntas importantes, incluyendo las siguientes:

1. ¿Este diseño es una solución al problema?
2. ¿El diseño es modular, bien estructurado y fácil de comprender?
3. ¿Puede hacerse algo para mejorar la estructura y la comprensión del sistema?
4. ¿El diseño es transportable hacia otras plataformas?

5. ¿Es reutilizable?
6. ¿Es fácil de modificar y expandir?
7. ¿Soporta facilidad de prueba?
8. ¿Reutiliza componentes provenientes de otros proyectos donde es apropiado?
9. ¿Maximiza el rendimiento donde es apropiado?
10. ¿Los algoritmos son los adecuados o pueden implementarse?
11. Si esta previsto desarrollar este sistema en fase, ¿tienen las "interfaces" adecuadas de modo que la transición entre etapas sea fácil?
12. ¿El diseño está bien documentado e incluye opciones de diseño y la justificación racional?
13. ¿El diseño hace referencia cruzada de componentes y datos con los requerimientos?
14. ¿Utiliza las técnicas apropiadas para el tratamiento de defectos y la prevención de fallas?

DOCUMENTACIÓN DEL DISENO

Los documentos de diseño deben contener una sección denominada justificación racional del diseño, donde se delinean las cuestiones críticas y compromisos que fueron considerados en la generación del diseño. Esta filosofía orientadora ayudara a los clientes y a otros desarrolladores a comprender como y porque encajan partes ciertas del sistema.

El diseño también contiene descripciones de los componentes del sistema. Una de las secciones debería indicar cómo interactúan los usuarios con el sistema, incluyendo lo siguiente:

- Menúes y otros formatos de presentaciones en pantalla.
- Interfaces hombre-máquina: teclas de función, descripción de pantallas sensibles al tacto, esquemas de teclados y uso del mouse e de la palanca para juegos (jostick).
- Formatos de los informes.
- Entrada: procedencia de los datos, como les da formato y el medio en que son almacenados.
- Salida: donde se envían los datos, como se les da formato y el medio en que son almacenados.
- Características funcionales generales
- Exigencias performance
- Procedimientos de archivo
- Enfoque de tratamiento de defectos.

Si el cliente lo requiere, se incluirá el diseño de los elementos que permitirán el monitoreo del rendimiento del sistema. Otras secciones de la documentación del diseño pueden tratar aspectos especiales como localización y aislamiento de defectos, configuración del sistema o medidas especiales de seguridad.

Finalmente se hace la referencia cruzada del diseño contra los requerimientos para demostrar como el diseño ha sido efectivamente derivado de ellos. Esta correspondencia obliga a verificar la consistencia y completitud, dicha referencia cruzada hará que las mejoras y modificaciones que se incorporen sean más fáciles de rastrear mas tarde.

CONCLUSIONES

Durante el desarrollo de cualquier tipo de sistema es de gran importancia el diseño ya que detras de este debe de llevar un respaldo, dicho respaldo consta de diferentes partes en los que seguramente no podrán ser los mejores de los mejores, pero si una solución que además de resolver o mejorar algún problema puede ser mejor que otras propuestas que quizas puedan ser mencionadas durante el proceso de desarrollo y que seguramente competiran con la que nosotros propongamos, es por eso que cualquier decision que tomemeos durante el diseño de un sistema, desde la forma en que se esta plantenado el problema hasta la propuesta de la GUI que será es este caso quien nis resoalde con los usurios potenciales que será a partir de ellos quienes sobre la marcha puedan decir o definir si es una buena solución o creen que no es asi, si este es el caso nuestro sistema no ha concluido de buena manera lo que dara por resultado un sistema malo.