CONCEPTOS Y DISEÑO DE REDES

Se dice que en cualquier proceso de diseño existen dos fases importantes: la diversificación y la convergencia. La diversificación es la adquisición de un repertorio de alternativas, de un material primitivo de diseño: componentes, soluciones de componentes y conocimiento, todo dentro de catálogos, de libros de texto y en la mente. En la convergencia, el diseñador elige y combina los elementos adecuados y extraídos de este repertorio para satisfacer los objetivos del diseño, de la misma manera a como se establece en el documento de los requisitos, y de la manera en que se acordó con el cliente. La segunda fase es la eliminación gradual de cualquier configuración de componentes excepto de una en particular, y de aquí la creación del producto final.

El diseño es una representación significativa de ingeniería de algo que se va a construir. Se puede hacer el seguimiento basándose en llos requisitos del cliente, y al mismo tiempo la calidad se puede evaluar y cotejar con el conjunto de criterios predefinidos para ontener un diseño bueno. El nivel de datos y de arquitectura, el diseño se centra en los patrones de la misma manera como se aplican en la aplicación que se va a construir. En el nivel de la interfaz, es la ergonomica humana la que dicta nuetro enfoque de diseño; mientras que en el nivel de componentes, un enfoque de programación conduce a diseños de datos y procedimientos eficaces.

El diseño comienza con el modelo de los requisitos.Se transforma este modelo y se obtienen cuatro niveles de detalles de diseño:

- Estructura de datos
- Arquitectura del sistema
- Representación de la interfaz
- Detalles a nivel de componentes

Dentro de cada una de las actividades dels diseño de sistema se aplican los principios y conceptos y principios básicos que llevan a obtener una alta calidad. Cada una de estas partes del diseño forma el producto obtenido del proceso de diseño. En cada una de estas se revisan los productos del diseño de software en cuanto a claridad, corrección finalización y consistencia y se comparan con los requisitos y unos con otros. Por otra parte, las metodologias de diseño de software carecen de la profundidad, flexibilidad y naturaleza cuantitativa que se asocian normalmente a las disciplinas de diseño de ingeniería más clásicas. Si existen métodos para el diseño y se pueden aplicar notaciones de diseño.

DISEÑO DE SOFTWARE E INGENIERÍA DEL SOFTWARE

El diseño del software se encuentra en el núcleo técnico de la ingeniería del software y se aplica independientemente del modelo de diseño de software que se utilice. Cada actividad transforma la información de manera que dé lugar por último a un software de computadora validado. Mediante uno de los muchos métodos de diseño (que se abarcarán en capítulos

posteriores) la tarea de diseño produce un diseño de datos, un diseño arquitectónico, un diseño de interfaz y un diseño de componentes.

El diseño de datos transforma el modelo del dominio de información que se crea durante el análisis en las estructuras de datos que se necesitarán para implementar el software. Es posible que parte del diseño de datos tenga lugar junto con el diseño de la arquitectura del software. A medida que se van diseñando cada uno de los componentes del software, van apareciendo más detalles de diseño.

El diseño arquitectónico define la relación entre los elementos estructurales'principales del software, los patrones de diseño que se pueden utilizar para lograr los requisitos que se han definido para el sistema, y las restricciones que afectan a la manera en que se pueden aplicar los patrones de diseño arquitectónicos.

El diseño de la interfaz describe la manera de comunicarse el software dentro de sí mismo, con sistemas que interoperan dentro de él y con las personas que lo utilizan. Una interfaz implica un flujo de información (por ejemplo, datos y/o control) y un tipo específico de comportamiento.

El diseno a nivel de componentes transforma los elementos estructurales de la arquitectura del software en una descripción procedimental de los componentes del software. La información que se obtiene de EP, EC yde DTE sirve como base para el diseño de los componentes.

El diseño es la única forma de convertir exactamente los requisitos de un cliente en un producto o sistema de software finalizado.

EL PROCESO DE DISEÑO

Es un proceso iterativo mediante el cual los requisitos se traducen en un «plano» para construir el Software. El diseño se representa a un nivel alto de abstracción -un nivel que puede rastrearse directamente hasta conseguir el objetivo del sistema específico y según unos requisitos más detallados de comportamiento, funcionales y de datos-. A medida que ocurren las iteraciones del diseño, el refinamiento subsiguiente conduce a representaciones de diseño a niveles de abstracción mucho más bajos.

Sugiere 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 se presentarán las siguientes directrices:

1. Un diseño deberá presentar una estructura arquitectóque (1) se haya creado mediante Patrones de diseño reconocibles, (2) que esté formada Por componentes que exhiban características de buen diseño

   Y (3) 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.

La evolución del diseño del software es un proceso continuo que ha abarcado las últimas cuatro décadas. El primer trabajo de diseño se concentraba en el critero de desarrollo de programas modulares y métodos para refinar las estructuras del software de manera descendente. Los aspectos procedimentales de la definición de diseño evolucionaron en una filosofía denominada programación estructurada. Un trabajo posterior propuso métodos para la coversión del flujo de datos o estructura de datos en una definición de diseño. Por otra parte existen enfoques de diseño más recientes que proponen un método orientado a objetos. Hoy en día, se ha hecho hincapiéen un diseño de software basado en la aruitectura del software.

PRINCIPIOS DEL DISEÑO

El diseño de software es tanto un proceso como un model. El proceso de diseño es una secuencia de pasos que hacen posible que el diseñador describa todas los aspectos del software que se va construir. Sin embargo, es importante destacar que el proceso de diseño simplemente no es un recetario.

Un conocimiento creativo, experiencia en el tema, un sentido de lo que hace que un software sea bueno, y un compromiso general con la calidad son factores críticos de éxito para un diseño competente.

El modelo de diseño es el equivalente a los planes de un arquitecto para una casa. Comienza representando la totalidad de todo lo que se va a construir y refina lentamente lo que va a proporcionar la guía para construir cada detalle. De manera similar, el modelo de diseño que se crea para el software proporciona diversas visiones diferentes de software de computadora.

Lista de principios:

En el proceso de diseño no deberá utilizarse «orejeras. Un buen diseñador deberá tener en cuenta enfoques alternativos, juzgando todos los que se basan en los requisitos del problema, los recursos disponibles para realizar el trabajo y los conceptos de diseño.

El diseño deberá poderse rastrear hasta el modelo de análisis. Dado que un solo elemento del modelo de diseño suele hacer un seguimiento de los múltiples requisitos, es necesario tener un medio de rastrear cómo se han satisfecho los requisitos por el modelo de diseño.

El diseño no deberá inventar nada que ya esté inventado. Los sistemas se construyen utilizando un conjunto de patrones de diseño, muchos de los cuales probablemente ya se han encontrado antes. Estos patrones deberán elegirse siempre como una alternativa para reinventar. Hay poco tiempo y los recursos son limitados. El tiempo de diseño se deberá invertir en la representación verdadera de ideas nuevas y en la integración de esos patrones que ya existen.

El diseño deberá «minimizar la distancia intelectual entre el software y el problema como si de la misma vida real se tratara. Es decir, la estructura del diseño del software (siempre que sea posible) imita la estructura del dominio del problema.
El diseño deberá presentar uniformidad e integración.Un diseño es uniforme si parece que fue una persona la que lo desarrolló por completo. Las reglas de estilo y de formato deberán definirse para un equipo de diseño antes de comenzar el trabajo sobre el diseño.
Un diseño se integra si se tiene cuidado a la hora de definir interfaces entre los componentes del diseño.
El diseño deberá estructurarse para admitir cambios.Los conceptos de diseño estudiados en la sección siguiente hacen posible un diseño que logra este principio.
El diseño deberá estructurarse para degradarse poco a poco, incluso cuando se enfrenta con datos, sucesos o condiciones de operación aberrantes. Un software bien diseñado no deberá nunca explotar como una <<bomba» Deberá diseñarse para adaptarse a circunstancias inusuales, y si debe terminar de funcionar, que lo haga de forma suave.
El diseño no es escribir código y escribir código no es diseñar. Incluso cuando se crean diseños procedimentales para componentes de programas, el nivel de abstracción del modelo de diseño es mayor que el código fuente. Las únicas decisiones de diseño realizadas a nivel de codificación se enfrentan con pequeños datos de implementación que posibilitan codificar el diseño procedimental.
El diseño deberá evaluarse en función de la calidad mientras se va creando, no después de terminarlo.
El diseño deberá revisarse para minimizar los errores conceptuales (semánticos). A veces existe la tendencia de centrarse en minucias cuando se revisa el diseño, olvidándose del bosque por culpa de los árboles. Un equipo de diseñadores deberá asegurarse de haber afrontado los elementos conceptuales principales antes de preocuparse por la sintaxis del modelo del diseño.

CONCEPTO DEL DISEÑO

Durante las ultimas décadas se ha experimentado la evolución de un conjunto de conceptos fundamentales de diseño de software. Aunque el grado de interes en cada concepto ha variado con los años, todos han experimntado el paso del tiempo.
Cuando se tiene en consideración una solución modular a cualquier problema, se pueden exponer muchos niveles de abstracción. En el nivel más alto de abstracción, la solución se pone como una medida extensa empleando el lenguaje del entorno del problema. En niveles inferiores de abstracción, se toma una orientación más procedimental. La terminología orientada a problemas va emparejada con la terminología orientada a la implementación en un esfuerzo por solucionar el problema. Finalmente, en el nivel más bajo de abstracción, se establece la solución para poder implementarse directamente.
A medida que vamos entrando en diferentes niveles de abstracción, trabajamos para crear abstracciones pro.cedimentales y de datos. Una abstracción procedimental es una secuencia nombrada de instrucciones que tiene una función específica y limitada.
Una abstracción de datos es una colección nombrada de datos que describe un objeto de datos. En el contexto de la abstracción procedimental abrir, podemos definir una abstracción de datos llamada puerta. Al igual que cualquier objeto de datos, la abstracción de datos para puerta

acompañaría a un conjunto de atributos que describen esta puerta (por ejemplo, tipo de puerta, dirección de apertura, mecanismo de apertura, peso, dimensiones).
La abstracción de control es la tercera forma de abstracción que se utiliza en el diseño del software. Al igual que las abstracciones procedimentales y de datos, este tipo de abstracción implica un mecanismo de control de programa sin especificar los datos internos.
El refinamiento paso a paso es una estrategia de diseño descendente propuesta originalmente por Niklaus Wirth. El desarrollo de un programa se realiza refinando sucesivamente los niveles de detalle procedimentales.
Una jerarquía se desarrolla descomponiendo una sentencia macroscópica de función (una abstracción procedimental) paso a paso hasta alcanzar las sentencias del lenguaje de programación.
La diferencia se encuentra en el nivel de detalle de implementación que se haya tomado en consideración, no en el enfoque.
El refinamiento verdaderamente es un proceso de elaboración. Se comienza con una sentencia de función (o descripción de información) que se define a un nivel alto de abstracción. Esto es, la sentencia describe la función o información conceptualmente, pero no proporciona
información sobre el funcionamiento interno de la información. El refinamiento hace que el diseñador trabaje sobre la sentencia original, proporcionando cada vez más detalles a medida que van teniendo lugar sucesivamente todos y cada uno de los refinamientos (elaboración).
El software se divide en componentes nombrados y abordados por separado, llamados frecuentemente módulos, que se integran para satisfacer los requisitos del problema.
Se ha afirmado que <<lam odularidad es el Único atributo del software que permite gestionar un programa intelectualmente».
Capacidad de empleo de componentes modulares. Si un método de diseño permite ensamblar los componentes de diseño (reusables) existentes en un sistema nuevo, producirá una solución modular que no inventa nada ya inventado.
Conjunto de propiedades que deberán especificarse como parte de un diseño arquitectónico:
Propiedades estructurales. Este aspecto de la representación del diseño arquitectónico define los componentes de un sistema (por ejemplo, módulos, objetos, filtros) y la manera en que esos componentes se empaquetan e interactúan unos con otros.
La jerarquia de control, denominada también estructura de programa, representa la organización de los componentes de programa (módulos) e implica una jerarquía de control. No representa los aspectos procedimentales del software, ni se puede aplicar necesariamente a todos los estilos arquitectónicos.
La división horizontal de la arquitectura proporciona diferentes ventajas:
Proporciona software más fácil de probar conduce a un software más fácil de mantener propaga menos efectos secundarios proporciona software más fácil de ampliar.
Los módulos del nivel superior deberán llevar a cabo funciones de control y no realizarán mucho trabajo de procesamiento.
La estructura de datos es una representación de la relación lógica entre elementos individuales de datos. La estructura dicta las alternativas de organización, métodos de acceso, grado de capacidad de asociación y procesamiento de la información.

Un elemento escalar es la estructura de datos más simple. Como su nombre indica, un elemento escalar representa un solo elemento de información que puede ser tratado por un idFntificador; es decir, se puede lograr acceso especificando un sola dirección en memoria.
La estructura de programa define la jerarquía de control sin tener en consideración la secuencia de proceso y de decisiones. El procedimiento de software se centra en el procesamiento de cada módulo individualmente El principio de ocultación de información sugiere que los módulos se caracterizan por las decisiones de diseño que (cada uno) oculta al otro. En otras palabras, los módulos deberán especificarse y diseñarse de manera que la información (procedimiento y datos) que está dentro de un módulo sea inaccesible a otros módulos que no necesiten esa información.
Ocultación significa que se puede conseguir una modularidad efectiva definiendo un conjunto de módulos independientes que se comunican entre sí intercambiando sólo la información necesaria para lograr la función del software.

DISEÑO MODULAR EFECTIVO
El concepto de independencia funcional es la suma de la modularidad y de los conceptos de abstracción y ocultación de información.
La independencia se mide mediante dos criterios cualitativos: la cohesión y el acoplamiento. La cohesión es una medida de la fuerza relativa funcional de un módulo.
El acoplamiento es una medida de la independencia relativa entre los módulos.
Un módulo cohesivo lleva a cabo una sola tarea dentro de un procedimiento de software, lo cual requiere poca interacción con los procedimientos que se llevan a cabo en otras partes de un programa.
En la parte inferior (y no deseable) del espectro, encontraremos un módulo que lleva a cabo un conjunto de tareas que se relacionan con otras débilmente, si es que tienen algo que ver. Tales módulos se denominan coincidentalmente cohesivos. Un módulo que realiza tareas relacionadas lógicamente (por ejemplo, un módulo que produce todas las salidas independientemente del tipo) es lógicamente cohesivo. Cuando un módulo contiene tareas que están relacionadas entre sí por el hecho de que todas deben ser ejecutadas en el mismo intervalo de tiempo, el módulo muestra cohesión temporal.
El acoplamiento es una medida de interconexión entre módulos dentro de una estructura de software. El acoplamiento depende de la complejidad de interconexión entre los módulos, el punto donde se realiza una entrada o referencia a un módulo, y los datos que pasan a través de la interfaz. Cuando los módulos están atados a un entorno externo al software se dan niveles relativamente altos de acoplamiento. El grado más alto de acoplamiento, acoplamiento de contenido, se da cuando un módulo hace uso de datos o de información de control mantenidos dentro de los límites de otro módulo.

HEURISTICA DE DISEÑO PARA UNA MODULARIDAD EFECTIVA
La estructura de programa se puede manipular de acuerdo con el siguiente conjunto de heurísticas:
1. Evaluar la «primera iteración» de la estructura de programa para reducir al acoplamiento y mejorar la cohesión.
II. Intentar minimizar las estructuras con un altogrado de salida; esforzarse por la entrada a

medida que aumenta la profundidad.
III. Mantener el ámbito del efecto de un módulo dentrodel ámbito de control de ese módulo.
IV. Evaluar las interfaces de los módulos para reducir la complejidad y la redundancia, y mejorar la consistencia.
V. Definir módulos cuya función se pueda predecir,pero evitar módulos que sean demasiado restrictivos.
VI. Intentar conseguir módulos de «entrada controlada, evitando «conexiones patológicas».

EL MODELO DEL DISEÑO

Crearemos un modelo de diseño que se tambalee fácilmente con vientos de cambio al establecer una base amplia en el diseño de datos, mediante una región media estable en el diseño arquitectónico y de interfaz, y una parte superior aplicando el diseño a nivel de componentes.

DOCUMENTACIÓN DEL 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.
La Especificación del diseño contiene una referencia cruzada de requisitos. El propósito de esta referencia cruzada (normalmente representada como una matriz simple) es: (1) establecer que todos los requisitos se satisfagan mediante el diseño del software, y (2) indicar cuales son los componentes críticos para la implementación de requisitos específicos.
La última sección de la Especificación del diseño contiene datos complementarios.
Será aconsejable desarrollar un Manual preliminar de Operacionesllnstalación e incluirlo como apéndice para la documentación del diseño.