Resumen final de Diseño de sistemas de información

Unidad 1: Fundamentos del Diseño

Diagrama del desarrollo

Los requisitos del sistema establecidos mediante los modelos de información, funcional y de comportamiento alimentan el paso del Diseño

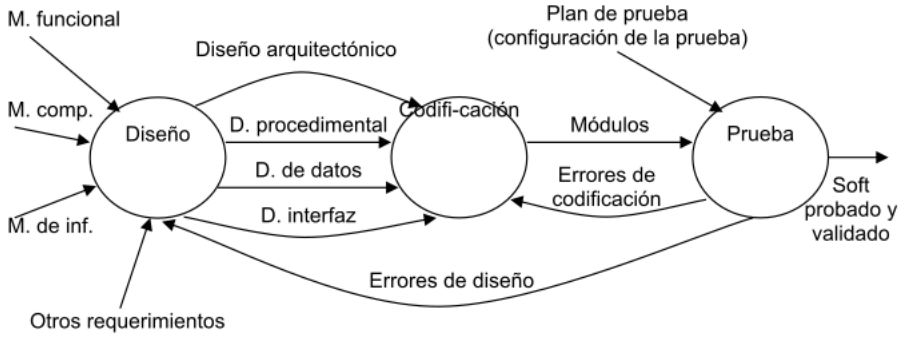

Figura 1: Diagrama del desarrollo

El plan de prueba se desarrolla desde el análisis. El diseño es el primer paso en la fase de desarrollo de cualquier producto o sistema de ingeniería. Podría definirse como el proceso de aplicar distintas técnicas, métodos y herramientas con el objetivo de **definir** un dispositivo, proceso o sistema, con el suficiente nivel de detalle para permitir su realización física (codificación).

El objetivo del diseñador es producir un modelo o representación de una entidad que se va a construir más adelante, combina la intuición y los criterios en base a la experiencia y un conjunto de heurísticas y/o principios que guían la forma en que se desarrolla el modelo.

El diseño constituye la base para la construcción y el mantenimiento del software.

Niveles de diseño desde el punto de vista técnico

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 arquitectónico.

Diseño procedimental: transforma los elementos estructurales en una descripción procedimental del software.

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.

Diseño de interfaz: establece la disposición y los mecanismos para la interacción hombre - máquina. Una interfaz implica un flujo de información y un tipo específico de comportamiento. Por lo tanto, los diagramas de flujo de datos y de control proporcionan la información necesaria para el diseño de la interfaz.

Niveles de diseño desde el punto de vista de gestión

Diseño preliminar: se centra en la transformación de los requisitos en los datos y la arquitectura del software.

Diseño detallado: se acerca más a la implementación. Se ocupa del refinamiento de la representación arquitectónica que lleva a una estructura de datos detallada y las representaciones algorítmicas del software.

El proceso de diseño

El diseño de software es un proceso iterativo a través del cual se traducen los requisitos en una representación del software. A lo largo del proceso de diseño, se evalúa la calidad del diseño con una serie de revisiones técnicas formales. Hay tres características que sirven de directrices para la evaluación de un buen diseño:

1. El diseño debe ser una guía que puedan leer y entender los que construyan el código y los que prueban y mantienen el software.
2. El diseño debería proporcionar una completa idea de lo que es el software, enfocando los dominios de datos, funcional y de comportamiento desde la perspectiva de la implementación.

Princicios del diseño: el diseño de software es tanto un proceso como un modelo. El proceso de diseño es una secuencia de pasos que hacen posible que el diseñador describa todos los aspectos del software que se va a construir. Es sólo un recetario.

- El proceso de diseño no deberá utilizarse: 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 del diseño.
- El diseño deberá poderse rastrear hasta el modelo de análisis: 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: el tiempo de diseño se deberá invertir en la representación verdadera de ideas nuevas y en la integración de esos patrones ya existentes.
- El diseño deberá minimizar la distancia intelectual entre el software y el problema: 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. Se integra si se tiene cuidado a la hora de definir interfaces entre los componentes del diseño.
- El diseño deberá'estructurarse para admitir cambios.
- El diseño deberá estructurarse para degradarse poco a poco: debe 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.
- 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).

Fundamentos del diseño

Se ha establecido un conjunto de conceptos fundamentales para el diseño del software.

1. **Abstracción:** es la visión del problema a diferentes niveles de detalles y desde diferentes perspectivas. Es una capacidad que debe tener el diseñador. Puede ser de datos (determinada colección de datos que describen un objeto, *cheque nómina*), procedimental (determinada secuencia de instrucciones que tiene una función limitada y específica, *palabra pase en una puerta*) o de control (implica un mecanismo de control de programa, sin especificar los detalles internos, *semáforo de un S.O*).
2. **Refinamiento:** significa plantear un problema a un nivel de abstracción alto e ir conociéndolo en más detalle en distintos niveles más bajos de abstracción. Este refinamiento de especificaciones

termina cuando todas las instrucciones se expresan en términos de cualquier lenguaje de programación. El refinamiento se puede hacer sobre distintos aspectos, no sólo la programación.

3. **Arquitectura:** la arquitectura determina la relación entre piezas de programa. Un objetivo del diseño del software es crear una versión arquitectónica de un sistema. Esta versión sirve como estructura desde la que se pueden llevar a cabo actividades de diseño más detalladas.
4. **Jerarquía de control:** representa la organización (a menudo jerárquica) de componentes del programa (módulos). No representa aspectos procedimentales del software.

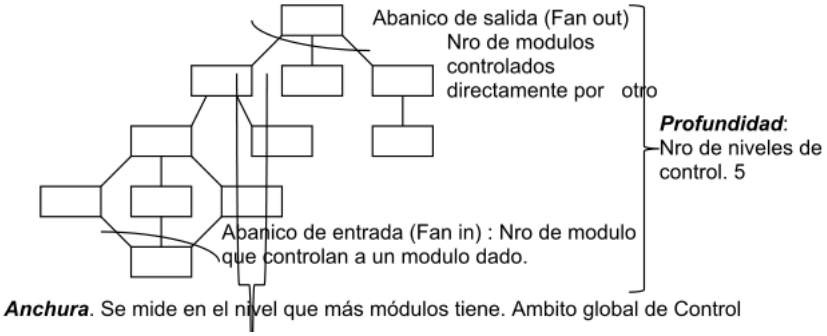

Figura 2: Jerarquía de módulos

5. **Modularidad:** es un atributo del software que lo hace manejable intelectualmente. El esfuerzo de desarrollo de un modulo individual disminiye conforme aumenta en número de módulos, sin embargo el esfuerzo asociado a las interfaces entre los módulos, va creciendo, esto nos lleva a una curva de esfuerzo total.

Hay un número M de módulos que resultaría en un costo de desarrollo mínimo, pero no tenemos la sofisticación necesaria para predecir M con seguirad.

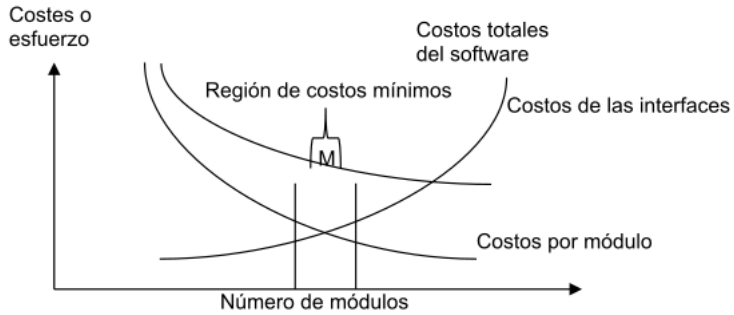

Figura 3: Costo mínimo

6. **Estructuras de datos:** es una representación de la relación lógica entre los elementos individuales de datos.
7. **Estructuras de control:**
8. **Ocultación y encapsulamiento de la información:** el principio de ocultación de información sugiere que los módulos se caractericen por decisiones de diseño que hagan que cada uno se oculte de los demás. Con otras palabras, se deberían especificar y diseñar los módulos para que la información (procedimientos y datos) contenida dentro de ellos sea inaccesible a otros módulos que no la necesiten.

Diseño Modular: un diseño modular reduce la complejidad, facilita los cambios y hace más fácil la implementación al fomentar el desarrollo en paralelo de diferentes partes de un sistema.

Independencia funcional: es un atributo de los módulos que determina o describe hasta que punto el módulo cumple con una única función específica, y no necesita para eso de otros de módulos. Un módulo funcionalmente independiente es un módulo'bien encapsulado.

El software con módulos independientes, es fácil de desarrollar porque su función puede ser partida y se simplifican las interfaces, son más fáciles de mantener, y se reduce la programación de errores y se fomenta la reutilización de los módulos. La independencia se mide usando dos criterios cualitativos: cohesión y acoplamiento.

Cohesión: es la fuerza con la que están unidas las sentencias de un módulo para cumplir la función. Es el grado en el cual los componentes de un módulo (las instrucciones individuales) son necesarios y suficientes para llevar a cabo una sola función bien definida. Cuando todas las sentencias son indispensables entonces la cohesión es alta.

Acomplamiento: es el grado de interdependencia que existe entre los módulos. La situación ideal sería que no haya interdependencia. El acomplamiento es el grado en el cual los módulos se relacionan entre sí. Mientras más fuerte sea el acomplamiento entre módulos en un sistema, más difícil es implantarlo y mantenerlo. A menor acoplamiento, mejor.

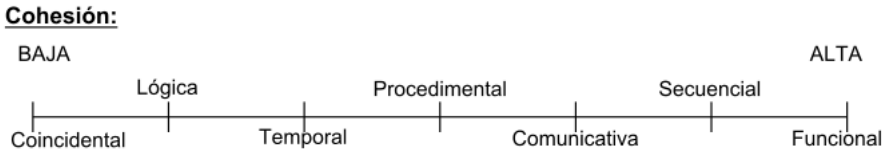

Figura 4: Cohesión

Figura 5: Acomplamiento

La calidad como objetivo

La importancia del diseño del software se puede decir con una sola palabra: calidad. El diseño es el lugar donde se fomenta la calidad en el desarrollo del software. El diseño nos proporciona representaciones del software en las que se puede valorar la calidad.

La calidad es un atributo del software que determina la concordancia con atributos de utilidad, mantenibilidad y portabilidad. Los factores que afectan a la calidad del software se pueden categorizar en dos grandes grupos: factores que se pueden medir directamente (defectos por punto de función) y factores que se pueden medir sólo indirectamente (facilidad de uso). En todos los casos debe aparecer la medición. Debemos comparar el software con un standard y llegar a una conclusión sobre la calidad.

Algunos factores son:

1. Corrección: hasta dónde satisface un programa su especificación y logra los objetivos de la misión del cliente.
2. Fiabilidad: hasta dónde se puede esperar que un programa lleve a cabo su función pretendida con la exactitud requerida.
3. Eficiencia: hasta dónde el programa cumple su objetivo con la mejor utilización de los recursos.
4. Integridad: hasta dónde se puede controlar el acceso al software o alos datos por personas no autorizadas.
5. Facilidad de uso: el esfuerzo necesario para aprender, operar, preparar los datos de entrada e interpretar las salidas de un pograma.

6. Facibilidad de mantenimiento: el esfuerzo necesario para localizar y arreglar un error en un programa.
7. Flexibilidad: el esfuerzo necesario para modificar un programa.
8. Facilidad de prueba: el esfuerzo necesario para probar un programa para asegurarse de que realizar su función pretendida.
9. Portabilidad: el esfuerzo necesario para transferir el programa de un entorno de sistema hardware y/o software a otro.
10. Reusabilidad: hasta dónde se puede volver a usar un programa (o partes de él) en otras aplicaciones.
11. Interoperatividad: el esfuerzo necesario para acoplar un sistema con otro.

Unidad 2: Niveles de diseño

Niveles

Diseño de datos: la actividad principal del diseño de datos es seleccionar representaciones lógicas de objetos de datos (estructuras de datos) identificadas durante la fase de definición y especificación de requisitos. La estructura de datos ha sido siempre una parte importante del diseño de software.

Diseño arquitectónico: el objetivo del diseño arquitectónico es desarrollar una estructura de programa modular y representar las relaciones de control entre los módulos. Además, el diseño arquitectónico combina la estructura del programa y las estructuras de datos, definiendo interfaces que permiten el flujo de datos a través del programa.

Diseño procedimental: el diseño procedimental se realiza después de los diseños de datos, arquitectónico y de interfaz. En un mundo ideal, la especificación procedimental necesaria para definir los detalles de los algoritmos se expresaría en un lenguaje natural. Debe especificar los detalles procedimentales sin abigüedades.

Diseño de la interfaz con el usuario: es la definición de interacción hombre - máquina. Estos mecanismos de interacción incluyen también a los dispositivos. La interfaz es la frontera entre el usuario y la aplicación del sistema (el punto donde la computadora y el individuo interactúan). Sus características influyen en la eficiencia del usuario, al igual que en la frecuencia de errores cuando se introducen datos o instrucciones.

Aspectos del diseño de interfaz

Aspectos humanos: se analiza al ser humano como persona (independientemente de la aplicación).

1. Percepción:
    - Sentidos: el sentido que guía nuestro diseño es la vista.
    - Capacidad cognitiva de la lectura: capacidad de adquirir conocimiento con la lecutra. Se debe aprovechar.
    - Memoria: hay dos tipos de memoria, de corto plazo y de largo plazo. El usuario puede acordarse de los comandos.
    - Mecanismos de deducción/inducción: es otra capacidad importante del usuario. Si hace una cosa siempre de la misma manera, es natural que lo intente hacer otra vez.
2. Comportamiento:
    - Personalidad: tiene que ver con cada persona.
    - Experiencia: (novatos, intermitentes, expertos).

Aspectos técnicos: nos preocupan:

1. Dispositivos: se deben definir dispositivos de entrada (mouse, teclado, lectores, cámaras, pantallas sensibles al tacto) y de salida (impresora, pantalla).
2. Diseño de entrada - salida: problema de aumentar la productividad del usuario. Implica el diseño de la entrada y salida de datos. Formularios.
3. Niveles de ayuda: toda interfaz debe tener alguún nivel de ayuda (sensible al contexto, en línea, teclas calientes).
4. Retroalimentación: es la informaciónque le damos al usuario constantemente sobre los resultados de lo que está haciendo. (Archivos log, barra de progreso, porcentajes, manejo de errores).
5. Tipo de interfaz: existen varios tipos.
    - De comandos: DOS es una interfaz de comandos. Requiere conocimiento del usuario.
    - De menús: NORTON por ejemplo. El usuario elige entre las opciones en pantalla.
    - De pregunta/respuesta: nos pregunta y le respondemos para que pueda hacer lo que queremos.
    - De manejo directo: brinda la posibilidad de acceder a diferentes acciones combinando cosas de las otras interfaces. Tengo el control de todo (puedo abrir un menú, escribir, apretar un botón, etc).
    - De ventanas: la idea es que pueda ver distintas cosas al mismo tiempo en varias ventanas.
    - Entrada salida: formas (formulario / llenado de datos).

Diseño de la entrada: en el diseño de entradas los analistas de sistemas deciden qué datos ingresan al sistema, qué medios utilizar, la forma en que se deben disponer o codificar los datos, el diálogo que servirá de guía a los usuarios para dar entrada a los datos, validación necesaria de datos y transacciones para detectar errores y los métodos para llevar a cabo la validación de las entradas y los pasos a seguir cuando se presentan errores.

El diseño de la entrada también incluye la especificación de los medios por los que tanto los usuarios finales como los operadores darán instrucciones al sistema sobre las acciones que deben emprender.

Diseño de la salida: el término salida se refiere a los resultados e información generados por el sistema. Cuando diseñan la salida, los analistas deben determinar qué información presentar, decidir si la información será presentada en forma visual, verbal o impresa y seleccionar el medio de salida, disponer la presentación de la información en un formato aceptable y decidir cómo distribuir la salida entre los posibles destinatarios.

Diseño de controles: los diseñadores también deben anticipar los errores que se cometerán al ingresar los datos en el sistema o al solicitar la ejecución de ciertas funciones. Un buen diseño de un sistema de información ofrecerá los medios para detectar y manejar el error.

Los controles de entrada proporcionan medios para asegurar que sólo los usuarios autorizados tengan acceso al sistema, garantizar que las transacciones sean aceptables, validar los datos para comprobar su exactitud y determinar si se han omitido datos que son necesarios.

Validación de la entrada: el término general dado a los métodos cuya finalidad es detectar errores en la entrada es *validación de entradas*. Tres categorías principales de métodos'tienen que ver con la verificación de la transacción, la verificación de los datos de la transacción y el cambio de estos últimos.

Verificación de la transacción: lo primero y lo más importante es identificar todas las transacciones que no son válidas. Las transacciones pueden caer en esta categoría porque están incompletas, no autorizadas e incluso fuera de lugar.

Validación de transacciones: los pasos que el sistema sigue para asegurarse de que la transacción es aceptable reciben el nombre de validación de la transacción. Por ejemplo, no es aceptable tratar de añadir un artículo nuevo si existe ya uno con el mismo nombre y número de identificación. El analista también debe asegurar que los procesos de validación de transacciones detecten situaciones donde se envía una entrada aceptable por un usuario que no está autorizado para hacerlo.