2

**Diseño**

El objetivo del diseño es producir un modelo o representación que tenga resistencia, funcionalidad y belleza. El modelo de diseño proporciona detalles sobre arquitectura del software, estructuras de datos, interfaces y componentes que se necesitan para implementar el sistema. Permite modelar el sistema o producto que se va a construir. En primer lugar, debe representarse la arquitectura del sistema o producto. Después se modelan las interfaces que conectan al software con los usuarios finales, con otros sistemas y dispositivos, y con sus propios componentes constitutivos. Por último, se diseñan los componentes del software que se utilizan para construir el sistema. El producto final es un modelo de diseño que agrupa las representaciones arquitectónicas, interfaces en el nivel de componente y despliegue. A medida que esto ocurre, se evalúan las alternativas, algunas se rechazan, se converge en “una configuración particular de componentes y, con ello, en la creación del producto final.

**Diseño en el contexto de la ingeniería de software**

El diseño del software comienza una vez que se han analizado y modelado los requerimientos, es la última acción de la ingeniería de software dentro de la actividad de modelado y prepara la etapa de construcción (generación y prueba de código).

El diseño de datos o clases transforma los modelos de clases en realizaciones de clases de diseño y en las estructuras de datos que se requieren para implementar el software.

El diseño de la arquitectura define la relación entre los elementos principales de la estructura del software. La representación del diseño de la arquitectura se obtiene del modelo de los requerimientos.

El diseño de la interfaz describe la forma en la que el software se comunica con los sistemas que interactúan con él y con los humanos que lo utilizan, los modelos de escenarios de uso y de comportamiento dan mucha de la información requerida para diseñar la interfaz.

La importancia del diseño del software se resume en una palabra: calidad.

**El proceso de diseño**

El diseño de software es un proceso iterativo.

Lineamientos y atributos de calidad del software:

* Debe implementar todos los requerimientos explícitos contenidos en el modelo de requerimientos y dar cabida a todos los requerimientos implícitos que desean los participantes.
* Debe ser una guía legible y comprensible para quienes generan el código y para los que lo prueban y dan el apoyo posterior.
* Debe proporcionar el panorama completo del software, y abordar los dominios de los datos, las funciones y el comportamiento desde el punto de vista de la implementación
* Lineamientos de la calidad: Establecer los criterios técnicos de un buen diseño.
* Atributos de calidad: FURPS: funcionalidad, usabilidad, confiabilidad, rendimiento y mantenibilidad.

1. Funcionalidad: se califica de acuerdo con el conjunto de características y capacidades del programa, la generalidad de las funciones que se entregan y la seguridad general del sistema.
2. Usabilidad: se evalúa tomando en cuenta factores humanos, la estética general, la consistencia y la documentación.
3. Confiabilidad: se evalúa con la medición de la frecuencia y gravedad de las fallas, la exactitud de los resultados que salen, el tiempo medio para que ocurra una falla (TMPF), la capacidad de recuperación ante ésta y lo predecible del programa.
4. Rendimiento: se mide con base en la velocidad de procesamiento, el tiempo de respuesta, el uso de recursos, el conjunto y la eficiencia.
5. Mantenibilidad: combina la capacidad del programa para ser ampliable (extensibilidad), adaptable y servicial, y además que pueda probarse, ser compatible y configurable (capacidad de organizar y controlar los elementos de la configuración del software) y que cuente con la facilidad para instalarse en el sistema y para que se detecten los problemas.

**La evolución del diseño de software**

Los aspectos de procedimiento del diseño evolucionaron hacia una filosofía llamada programación estructurada. Los aspectos de procedimiento del diseño evolucionaron hacia una filosofía llamada programación estructurada, sin embargo, todos cuentan con:

1. un mecanismo para traducir el modelo de requerimientos en una representación del diseño.
2. Una notación para representar las componentes funcionales y sus interfaces.
3. Una heurística para mejorar y hacer particiones.
4. Lineamientos para evaluar la calidad.

Sin importar el método de diseño que se utilice, debe aplicarse un conjunto de conceptos básicos al diseño en el nivel de datos, arquitectura, interfaz y componente.

**Conceptos de diseño**

* Abstracción: muchos niveles de abstracción. En el más elevado se enuncia una solución en términos gruesos. En niveles más bajos de abstracción se da la descripción más detallada de la solución. Una *abstracción de procedimiento* es una secuencia de instrucciones que tienen una función específica y limitada. Una *abstracción de datos* es un conjunto de éstos con nombre que describe a un objeto de datos.
* Arquitectura: La arquitectura del software alude a la estructura general de éste y a las formas en las que ésta da integridad conceptual a un sistema, es decir, la organización de los componentes de un programa.

1. Propiedades estructurares: define los componentes de un sistema (módulos, objetos, filtros, etc.) y la manera en la que están agrupados e interactúan unos con otros
2. Propiedades extra funcionales: La descripción del diseño arquitectónico debe abordar la forma en la que la arquitectura del diseño satisface los requerimientos de desempeño, capacidad, confiabilidad, seguridad y adaptabilidad, así como otras características del sistema.
3. Familias de sistemas relacionados: Debe basarse en patrones repetibles que es común encontrar en el diseño de familias de sistemas similares

Los *modelos estructurales* representan la arquitectura como un conjunto organizado de componentes del programa. Los *modelos de marco* aumentan el nivel de abstracción del diseño. Los *modelos dinámicos* abordan los aspectos estructurales de la arquitectura del programa e indican cómo cambia la estructura o la configuración del sistema en función de eventos externos. Los *modelos del proceso* se centran en el diseño del negocio o proceso técnico al que debe dar acomodo el sistema. Por último, los *modelos funcionales* se usan para representar la jerarquía funcional de un sistema

* Patrones: Patrón de diseño es una mezcla con nombre propio de puntos de vista que contienen la esencia de una solución demostrada para un problema recurrente dentro de cierto contexto de necesidades en competencia. Describe una estructura de diseño que resuelve un problema particular del diseño dentro de un contexto específico y entre “fuerzas” que afectan la manera en la que se aplica y en la que se utiliza dicho patrón.
* División de problemas: Sugiere que cualquier problema complejo puede manejarse con más facilidad si se subdivide en elementos susceptibles de resolverse u optimizarse de manera independiente. Un problema es una característica o comportamiento que se especifica en el modelo de los requerimientos para el software.
* Modularidad: El software se divide en componentes con nombres distintos y abordables por separado, en ocasiones llamados módulos, que se integran para satisfacer los requerimientos del problema.
* Ocultamiento de la información: El principio del ocultamiento de información sugiere que los módulos se “caractericen por decisiones de diseño que se oculten (cada una) de las demás”. En otras palabras, deben especificarse y diseñarse módulos, de forma que la información (algoritmos y datos) contenida en un módulo sea inaccesible para los que no necesiten de ella. El ocultamiento define y hace cumplir las restricciones de acceso tanto a los detalles de procedimiento como a cualquier estructura de datos local que utilice el módulo.
* Independencia funcional: Es resultado directo de la separación de problemas y de los conceptos de abstracción y ocultamiento de información. Debe diseñarse software de manera que cada módulo resuelva un subconjunto específico de requerimientos y tenga una interfaz sencilla cuando se vea desde otras partes de la estructura del programa. La independencia se evalúa con el uso de dos criterios cualitativos: la cohesión y el acoplamiento. La *cohesión* es un indicador de la fortaleza relativa funcional de un módulo. El *acoplamiento* lo es de la independencia relativa entre módulos.
* Refinamiento: es un proceso de elaboración. Se comienza con un enunciado de la función (o descripción de la información), definida en un nivel de abstracción alto.
* Aspectos: representación de una preocupación de interferencia.
* Rediseño: técnica de reorganización que simplifica el diseño (o código) de un componente sin cambiar su función o comportamiento. Es el proceso de cambiar un sistema de software en forma tal que no se altera el comportamiento externo del código [diseño], pero sí se mejora su estructura interna
* Clases de diseño:

1. Clases de usuario de la interfaz. Definen todas las abstracciones necesarias para la interacción humano-computadora.
2. Clases del dominio de negocios: Las clases identifican los atributos y servicios (métodos) que se requieren para implementar algunos elementos del dominio de negocios.
3. Clases de proceso. Implantan abstracciones de negocios de bajo nivel que se requieren para administrar por completo las clases de dominio de negocios.
4. Clases persistentes: Representan almacenamientos de datos.
5. Clases de sistemas: Implantan las funciones de administración y control del software que permiten que el sistema opere y se comunique dentro de su ambiente de computación y con el mundo exterior.

* Completa y suficiente: Debe ser el encapsulado total de todos los atributos y métodos que sea razonable esperar.
* Primitivismo: deben centrarse en el cumplimiento de un servicio para la clase.
* Mucha cohesión: Para implementarlas emplea atributos y métodos de objetivo único.
* Poco acoplamiento: La colaboración debe mantenerse en un mínimo aceptable.

**El modelo de diseño**

La *dimensión del proceso* indica la evolución del modelo del diseño conforme se ejecutan las tareas de éste como parte del proceso del software. La *dimensión de la abstracción* representa el nivel de detalle a medida que cada elemento del modelo de análisis se transforma en un equivalente de diseño y luego se mejora en forma iterativa.

* Elementos de diseño de datos: el diseño de datos crea un modelo de datos o información que se representa en un nivel de abstracción elevado.
* Elementos de diseño arquitectónico: dan la visión general del software.
  1. información sobre el dominio de la aplicación del software que se va a elaborar
  2. Los elementos específicos del modelo de requerimientos, tales como diagramas de flujo de datos o clases de análisis, sus relaciones y colaboraciones para el problema en cuestión
  3. La disponibilidad de estilos arquitectónicos.
* Elementos de diseño de la interfaz: Permiten que la información fluya hacia dentro y afuera del sistema, y cómo están comunicados los componentes que son parte de la arquitectura.

1. La interfaz de usuario (IU). incorpora elementos estéticos, elementos ergonómicos y elementos técnicos.
2. Las interfaces externas que tienen que ver con otros sistemas, dispositivos, redes y otros productores o consumidores de información. Requiere información definitiva sobre la entidad a la que se envía información o desde la que se recibe.
3. Interfaces internas que involucran a los distintos componentes del diseño. Se relaciona de cerca con el diseño de componentes

Interfaz: Es un especificador para las operaciones visibles desde el exterior [públicas] de una clase, un componente u otro clasificador (incluso subsistemas), sin especificar su estructura interna, es decir, es un conjunto de operaciones que describen alguna parte del comportamiento de una clase y dan acceso a aquéllas.

* Elementos de diseño en el nivel de componentes: describe por completo los detalles internos de cada componente. define estructuras de datos para todos los objetos de datos locales y detalles algorítmicos para todo el procesamiento que tiene lugar dentro de un componente, así como la interfaz que permite el acceso a todas las operaciones de los componentes.
* Elementos del diseño de despliegue: indican la forma en la que se acomodarán la funcionalidad del software y los subsistemas dentro del ambiente físico de la computación que lo apoyará. *Formato descriptor* esto significa que el diagrama de despliegue muestra el ambiente de computación, pero no indica de manera explícita los detalles de la configuración.

Preguntas