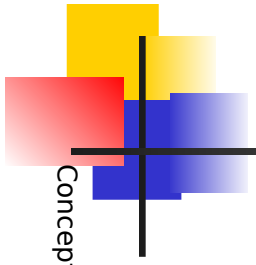

Análisis y Diseño del Software

Análisis de Requisitos

La **Ingeniería de requerimientos** es el proceso de descubrir, refinar, modelar y especificar el SW.

- En el análisis de los requisitos, se refinan en detalle los requisitos del sistema y el rol del software. Se crean modelos de datos, flujo de información y control, y de comportamiento. Se analizan soluciones alternativas.

- El análisis de requisitos es una tarea de ingeniería del software que cubre la brecha entre la especificación de los requerimientos del sistema y el diseño del software.

- El análisis de requisitos permite al ingeniero de software especificar las características operacionales del sistema (función, datos y rendimientos), definir interfaces y establecer las restricciones que debe cumplir el software.

Áreas de esfuerzo en el Análisis de Requisitos

El análisis de requisitos del SW puede dividirse en cinco áreas de esfuerzo:

(1) Reconocimiento del problema

(2) Evaluación y síntesis

(3) Modelado

(4) Especificación

(5) Revisión

Áreas de esfuerzo en el Análisis de Requisitos

1. Reconocimiento del problema

Reconocer los elementos básicos del problema tal y como los percibe el cliente/usuario. (Análisis del Problema).

2. Evaluación del problema y Síntesis de la solución

¿Qué datos produce y consume el sistema,

qué funciones debe realizar el sistema,

qué interfaces se definen,

qué restricciones son aplicables?.

Describir el problema y lograr sintetizar una solución global.

Áreas de esfuerzo en el Análisis de Requisitos

3. Modelado

El analista crea el modelo de análisis para entender el flujo de datos y de control, el tratamiento funcional, el comportamiento operativo y el contenido de la información.

4. Especificación

El modelo sirve de base para la creación de una especificación del software.

Especificación con más detalle de la solución elegida.

Las técnicas de especificación se realizan en lenguaje natural, lenguaje formal o prototipos.

Especificaciones incompletas e inconsistentes provocan frustración, confusión, baja calidad, mayor costo, retrasos.

Áreas de esfuerzo en el Análisis de Requisitos

5. Revisión

La revisión de la especificación de requisitos del software es llevada a cabo por el desarrollador y por el cliente.

Los revisores intentan asegurarse de que la especificación sea completa, consistente y correcta.

La especificación resulta en un «contrato» para el desarrollo del software entre el desarrollador y el cliente.

Los cambios requeridos por el cliente significarán una modificación de este contrato, y por lo tanto podrán aumentar los costos y prolongar los plazos de entrega del proyecto.

Principios generales del Análisis

- Representar y entender el dominio de información de un problema.
- Definir las funciones que realizará el software.
- Representar el comportamiento del software, como consecuencia de acontecimientos externos.
- Dividir los modelos que representan información, función y comportamiento de manera que se descubran los detalles por capas (o jerárquicamente).
- El proceso de análisis debe ir desde la información esencial hasta el detalle de la implementación.

Resultado del Análisis de Requisitos

Se obtiene la **Especificación de requisitos del software** (en el lenguaje del desarrollador) que contiene:

- Descripción detallada de lo que requiere el cliente.
- Establece una base para el diseño del Software.
- Define y especifica los requisitos que podrán ser validados una vez construido el software.

Diseño del Software

El **Diseño del Software** es la primera de las tres actividades técnicas en el desarrollo de software: **diseño**, **generación de código** y **prueba**.

El **Modelo de Diseño** está constituido por:

1. **Diseño de Datos:** Transforma el modelo del dominio en las estructuras de datos que se necesitarán para implementar el SW.

2. **Diseño Arquitectónico:** Define la relación entre los elementos estructurales del software y las restricciones (que afectan la aplicación de los patrones de diseño y arquitectónicos).

Diseño del Software

3. **Diseño de la Interfaz:** Describe la forma de comunicarse el SW entre sí, la interoperabilidad con otros sistemas y con las personas que lo utilizan.

4. **Diseño a nivel de Componentes:** Transforma los elementos estructurales de la arquitectura del SW en una descripción procedimental de los componentes del software.

Diseño del Software

- El **diseño del software** es un proceso iterativo donde los requisitos se traducen a un «plano» para construir el SW.
- El proceso de diseño inicia en un nivel alto de abstracción y por refinamientos sucesivos obtiene representaciones a niveles más bajos.
- La calidad de la evolución del diseño debe ser evaluada con una serie de revisiones de diseño.

El **Diseño** debe:

- Ser una guía legible y comprensible para los generadores de código y para aquellos que comprueban y dan soporte al SW.
- Proporcionar una imagen completa del SW, cubriendo su comportamiento, funciones y datos desde una perspectiva de implementación.

Conceptos de Diseño del Software

- ABSTRACCION
- REFINAMIENTO
- MODULARIDAD
- ARQUITECTURA DEL SOFTWARE
- JERARQUIA DE CONTROL
- DIVISIÓN ESTRUCTURAL
- ESTRUCTURA DE DATOS
- PROCEDIMIENTO DE SOFTWARE
- OCULTACIÓN DE INFORMACIÓN
- INDEPENDENCIA FUNCIONAL: COHESIÓN y ACOPLAMIENTO

Conceptos de Diseño del Software

- **ABSTRACCION**

Una ***abstracción procedimental*** es una secuencia nombrada de instrucciones que tiene una función específica y limitada. Ej: Abrir una puerta. **Abrir** implica una secuencia de pasos procedimentales.

Una ***abstracción de datos*** es una colección nombrada de datos que describe un objeto de datos. **Puerta** es una abstracción de datos.

Una ***abstracción de control*** implica un mecanismo de control de programa sin especificar los datos internos.

- **REFINAMIENTO**

El ***refinamiento paso a paso*** es una estrategia de diseño descendente. Se desarrolla un programa, refinando sucesivamente los niveles de detalle procedimentales.

Conceptos de Diseño del Software

- **MODULARIDAD**

El SW se divide en componentes, llamados ***módulos***, que se integran para satisfacer los requisitos del problema.

Capacidad de descomposición modular. Usar un mecanismo sistemático para descomponer el problema en subproblemas, reduce la complejidad del problema, y obtiene una solución modular más efectiva.

Capacidad de empleo de componentes modulares. Ensamblar componentes de diseño (reusables) existentes, en un sistema nuevo.

Capacidad de comprensión modular. Si un módulo se puede comprender como una unidad autónoma (sin referencias a otros módulos) será más fácil de construir y de cambiar.

Continuidad modular. Si se cambian los módulos individuales, se minimizará el impacto de los efectos secundarios de los cambios.

Protección modular. Si en un módulo se produce una condición no deseada y sus efectos se limitan a ese módulo, se minimizará el impacto de los efectos secundarios inducidos por los errores.

Conceptos de Diseño del Software

- **ARQUITECTURA DEL SOFTWARE**

La arquitectura es la estructura jerárquica de los componentes (módulos), la manera en que interactúan y la estructura de datos que van a utilizar. Los patrones arquitectónicos permiten determinar formas generales para estructurar todo el sistema.

- **JERARQUIA DE CONTROL**

La relación de control entre módulos se expresa considerando una jerarquía entre módulos y de quién controla a quién.

- **DIVISIÓN ESTRUCTURAL**

La estructura del programa se puede dividir tanto horizontal como verticalmente.

- **ESTRUCTURA DE DATOS**

Es una representación de la relación lógica entre los datos, y determina una estructura de datos para el sistema.

Conceptos de Diseño del Software

- **PROCEDIMIENTO DE SOFTWARE**

Se debe proporcionar una especificación precisa de procesamiento de cada módulo, incluyendo la secuencia de sucesos, los puntos de decisión, las operaciones repetitivas y la organización de datos.

- **OCULTACIÓN DE INFORMACIÓN**

Sugiere que los módulos se caractericen por las decisiones de diseño que cada uno oculta al otro.

- **INDEPENDENCIA FUNCIONAL:**
  COHESIÓN y ACOPLAMIENTO

Crear módulos cohesivos que tengan una única función, y con poca interacción con las funciones que realizan otros módulos. El **acoplamiento** depende de la complejidad de interconexión entre módulos. En el diseño del software, intentamos conseguir el acoplamiento más bajo posible.

Bibliografía Consultada

Pressman, Roger. Software Engineering: A Practitioner's Approach. Fifth edition. ISBN: 0071181822. McGraw Hill. 2001.

Capítulo 11: Conceptos y Principios de Análisis

Capítulo 13: Conceptos y Principios de Diseño