

8. Requisitos del software: organización y calidad

ORGANIZACIÓN

Métodos para organizar los requisitos del software

- Una buena organización de los requisitos del software es esencial, porque si no están organizados no se pueden manejar (localizar, actualizar, aplicar, comprobar...)
- Técnicas ya mencionadas
 - Tablas de referencias cruzadas
 - composición NFR-FR, transversalidad: NFR *afecta a* FR
 - consistencia: conflicto, acoplamiento, redundancia
 - Matrices de trazabilidad hacia requisitos de usuario (hacia diseño en ADD)
 - Principios de *modularidad* y *anidamiento* a los requisitos (estructuración)
- Utilizar tablas para representar explícitamente las relaciones entre los requisitos y el modelo, y entre distintas partes del modelo, por ejemplo:
 - casos de uso – requisitos derivados del caso de uso
 - requisitos – clases derivadas del requisito
 - casos de uso – clases mencionadas en el caso de uso

Organización de los requisitos por casos de uso

- Estructurar los requisitos en torno a los *servicios requeridos por los usuarios*, descritos como secuencias de operaciones
- Cada caso de uso agrupa:
 - un requisito principal descrito en el *objetivo* del caso de uso
 - uno o varios requisitos subordinados, que estarán relacionados con:
 - las *clases* de objetos que se mencionan (elementos de información)
 - las *operaciones* individuales de que consta el caso de uso
- *Ejemplo*: caso de uso “alquilar una película de videoclub”
 - las películas alquilables tienen título, director, actores, duración...
 - las películas alquilables son ordenables o filtrables según distintos criterios...

Organización de los requisitos por clases

- Estructurar los requisitos en *unidades atómicas de información y comportamiento*
- Las clases resultantes de la organización de los requisitos pueden tener mucho o nada que ver con las clases del diseño y la implementación. Si la relación es estrecha...
 - *Ventajas*: trazabilidad, una de las banderas de la OO; las clases próximas a conceptos del mundo real es más probable que sean reutilizadas
 - *Desventajas*: acoplamiento análisis-diseño, selección de clases de diseño antes de tiempo, simular innecesariamente la estructura del mundo real en el diseño
- Trazabilidad: la correspondencia entre clases de análisis y clases de diseño es *muchos-muchos*; si no puede ser *uno-uno*, al menos puede intentarse que sea *pocos-pocos*
- Procedimiento:
 - Identificar y enumerar las *clases de análisis*: las que pertenecen específicamente a la aplicación, al problema en cuestión
 - Para cada clase, describir la *funcionalidad requerida* de la aplicación que pertenezca principalmente a esa clase, en forma de atributos y operaciones
 - Atributos: unidades de información
 - Operaciones: unidades de comportamiento, reacciones frente a eventos (es mucho más fácil repartir los atributos que repartir las operaciones)
 - Anotar si se requiere la *existencia* de objetos individuales de la clase

Organización de los requisitos por estados

- Estructurar los requisitos de acuerdo a los distintos estados o modos de funcionamiento del sistema
- Ejemplos:
 - aplicación contable: estados Configuración, Ejecución, Respaldo
 - aplicación radar: estados Entrenamiento, Normal y Emergencia

Uso de herramientas para el análisis de requisitos

- Las herramientas pueden ayudar a...
 - capturar y gestionar los requisitos, por ejemplo mediante ordenación, priorización, asignación y seguimiento
 - valorar el estado del análisis de requisitos
- La información de seguimiento puede presentarse en una hoja de cálculo vía web, con hipervínculos a los documentos relevantes del proyecto
- Los hipervínculos ayudan a evitar la *duplicidad de información*: por ejemplo, la doble especificación de un requisito – poner en el código un hiperenlace al documento de requisitos donde se especifica el requisito implementado

CALIDAD**Qué sentido tiene realizar medidas de calidad en los requisitos del software**

- Los requisitos deben cumplir determinados criterios de calidad (*cuantificables*)
- Si no se exige que los requisitos cumplan los criterios de calidad, entonces más adelante no se podrá comprobar que la aplicación satisface estos mismos criterios (es decir, no se podrá *buscar la calidad en fases posteriores* del proyecto)
- La calidad de los requisitos debe ser *medida por personas distintas* de los autores
- Toda medición (o métrica) tiene un *coste añadido* en dinero y tiempo para obtenerla, almacenarla, analizarla e informar acerca de ella
 - Se trata de optimizar la relación coste/beneficio, lo cual depende de muchos factores: cultura de la organización, estado del proyecto, naturaleza del proyecto, etc.
 - No se trata de obtener mediciones sin saber para qué
 - Idea general: clasificar las mediciones en tres categorías (*bien, regular, mal*)
- Las medidas son más útiles cuando sus valores esperados se especifican por adelantado
 - Ejemplo: basados en experiencias anteriores, los requisitos se consideran completos cuando la tasa de creación y modificación es menor del 1% semanal

Métricas de calidad: cómo de bien están escritos los requisitos

- Grado de completitud (estimado a partir del ritmo de creación/modificación; aunque probablemente es imposible en la práctica decir cuál es el “último” requisito)
- Grado de consistencia (número de conflictos entre requisitos)
- Porcentaje de requisitos
 - ambiguos
 - mal clasificados (asignados a un caso de uso o a una clase equivocados)
 - no trazables
 - no priorizados
 - sin riesgo estimado
 - no comprobables
 - sin condiciones de error

Métricas de calidad: cómo de efectivos son los procesos

- Medidas de la efectividad de la inspección de requisitos
 - Porcentaje de requisitos que faltan o son defectuosos, encontrados por hora de inspección
- Medidas de la efectividad del proceso de análisis de requisitos
 - Coste por requisito específico
 - Coste medio total (tiempo total / número de requisitos)
 - Coste marginal (coste de obtener uno más)
 - Ritmo al que los requisitos son: modificados / eliminados / añadidos

Los Requisitos del Software en el Estándar ESA

- Lo que dice cada uno de los documentos
 - ESA software engineering standards (PSS-05-0)
 - Guide to applying the ESA SE-Std to projects using OO Methods (BSSC98-1)
 - Guide to the software requirements definition phase (PSS-05-03)
- Nociones importantes: *modelo lógico* (PSS-05-0, 3.3.1; BSSC98-1, 3.4.1)
 - Modelo de lo que necesita el usuario, independiente de la implementación
 - *modelo de casos de uso*: denominado en ESA *modelo de requisitos*
 - derivado del modelo de negocio
 - *modelo conceptual*: denominado en ESA *modelo de análisis*
 - derivado del modelo de dominio
- Contenido del documento