El análisis de requisitos proporciona como resultado la especificación de las características operativas del software, indica la interfaz de éste con otros elementos del sistema, y establece las restricciones que limitan al software. El análisis de los requisitos permite al profesional construir sobre los requisitos básicos establecidos durante la etapa de obtención de los mismos.

Introducción

El análisis de requisitos describe las tareas que deben llevarse a cabo para examinar los requisitos con el propósito de delimitarlos y definir exactamente cada uno de ellos. Se trata fundamentalmente de:

- Detectar y resolver conflictos entre requisitos.
- Delimitar el software y establecer con qué elementos externos interacciona.
- Elaborar los requisitos del sistema para obtener, a partir de ellos, los requisitos del software a desarrollar.

El análisis de los requisitos enlaza la definición de alto nivel del sistema, desde la perspectiva del usuario (perspectiva necesariamente externa), con el diseño del software que permitirá implementar dicho sistema (una visión interna de cómo se comportará el software). Es absolutamente imprescindible entender exactamente qué tienen en mente los clientes y los usuarios para, basándose en la especificación de dichas necesidades, construir el sistema que los clientes desean y no otro.

Actividades durante el análisis

Clasificación y priorización de los requisitos

La *clasificación de los requisitos* consiste en establecer un conjunto de categorías y situar cada requisito en ellas. La forma de clasificar los requisitos no es siempre la misma, existiendo diversos criterios con diferente nivel de aceptación. Así, pueden clasificarse en funcionales o no funcionales, del proceso o del producto, por prioridad, derivados (p. e., de otro requisito) o impuestos (p. e., por el sistema existente), por su ámbito, por su volatilidad, etc.

En cuanto a la *priorización*, debe determinarse la importancia relativa de cada requisito en relación con los demás. Para computar este valor de importancia, habitualmente se utiliza el método de preguntar a clientes y usuarios acerca del grado de satisfacción que les proporcionaría la implementación de las funcionalidades, restricciones o calidades del requisito, junto con el grado de insatisfacción que les produciría el que dicho requisito no fuera implementado. Usando estos datos como entrada del cálculo, se obtendría un valor de prioridad para el requisito.

Modelado conceptual

El modelado conceptual tiene como objetivo fundamental facilitar el entendimiento de los requisitos mediante su representación en un lenguaje o notación que “comprendan” quienes van a tratar con los requisitos. El modelado no es parte de la elaboración de una solución: simplemente es una representación que permite comprender el problema.

La elaboración de un modelo conceptual puede enfocarse desde diversas perspectivas, cada una de ellas aportará información de un aspecto diferente, pero complementario con los demás, del sistema. Así, será posible elaborar modelos conceptuales de datos, (muestran las entidades del dominio del problema, sus relaciones y dependencias), de flujo de control (describen las interacciones de procesos) y de datos, entre otros.

Para el modelado conceptual de los requisitos pueden utilizarse diferentes notaciones. No obstante, la existencia de un lenguaje común de modelado facilita la comunicación y evita esfuerzos de formación, pues elimina la necesidad de conocer cada notación particular de cada proyecto en el que se participa. El uso de un lenguaje de modelado como UML ofrece la ventaja de no admitir ambigüedades, eliminando la existencia de diferentes interpretaciones.

Situación de los requisitos en la arquitectura del sistema

En un cierto momento del desarrollo debe establecerse qué elementos del sistema software van a ser los responsables de satisfacer las demandas planteadas por los distintos requisitos. A estas tareas, es a lo que se denomina *situación de los requisitos en la arquitectura del sistema*. A menudo, esta tarea de situación de requisitos en la arquitectura permite descubrir nuevos requisitos derivados de la implementación de un conjunto concreto de requisitos por parte de un componente específico del sistema, por lo que resultan de especial importancia.

Negociación de los requisitos

Muchos autores consideran que la etapa de análisis de requisitos debería denominarse etapa de "análisis y negociación de requisitos". Esto es así porque durante el análisis de los requisitos surgen a menudo conflictos de intereses entre las diferentes partes involucradas. En la negociación de los requisitos, se deberían detectar y resolver posibles conflictos entre requisitos, definir de manera precisa cuáles son los límites del sistema y cómo éste debe interaccionar con su entorno. La resolución de conflictos tiene que ver con la eliminación de los problemas entre las diferentes necesidades de los actores implicados en el proyecto: solicitud de características incompatibles, conflictos entre requisitos funcionales y no funcionales, y otras. La solución más habitual es tomar una decisión de consenso que satisfaga suficientemente a todos los interesados, aunque todos cedan, en cierta medida, en sus pretensiones iniciales.

Especificación de requisitos

La especificación de requisitos consiste en la completa descripción de los requisitos del sistema a desarrollar. En dicha especificación se incluye, para cada requisito, información complementaria que permite gestionarlo e interpretarlo, información que a menudo se denomina "atributos de los requisitos".

La especificación de requisitos implica, con frecuencia, la construcción de un modelo (o varios) del sistema desde el punto de vista de los usuarios, que incluya todos y cada uno de los requisitos obtenidos.

Según el estándar 830-1998 de IEEE para la especificación de requisitos del software, una especificación debe ser: *Completa, verificable, consistente, modificable, Susceptible de permitir seguimientos, utilizable durante las fases de operación y mantenimiento*, y *no debe contener ambigüedades.*

De manera general, los requisitos se plasman en un documento denominado especificación de requisitos del software, comúnmente conocido por sus siglas en inglés SRS. Para sistemas sencillos la elaboración de un único documento es suficiente, pero en sistemas complejos es habitual la elaboración de tres documentos distintos como parte de esta actividad: el documento de definición del sistema, el documento de requisitos del sistema y, finalmente, la especificación de requisitos del software (SRS).

Análisis orientado a objetos

La conclusión de lo dicho hasta ahora es que, el *Análisis* pone énfasis en la *investigación* del problema y los requisitos, en vez de ponerlo en una solución.

"Análisis" es un término amplio, es más adecuado calificarlo como *análisis de requisitos* (un estudio de los requisitos) o *análisis de objetos* (un estudio de los objetos del dominio).

Durante el *análisis orientado a objetos*, se presta especial atención a encontrar y describir los objetos, o conceptos, en el dominio del problema,

La finalidad del análisis orientado a objetos es crear una descripción del dominio desde la perspectiva de la clasificación de objetos. Una descomposición del dominio conlleva una identificación de los conceptos, atributos y asociaciones que se consideran significativas. El resultado se puede expresar en un *modelo del dominio*, que se ilustra mediante un conjunto de diagramas que muestran los objetos o conceptos del dominio.

Modelo de análisis

El modelo de análisis está formado por dos modelos (artefactos) fundamentales:

- *Modelo estático*: Modela los conceptos básicos en el dominio de negocio a través de las denominadas clases de análisis. A menudo, a este modelo se le denomina también modelo del dominio, modelo conceptual o diagrama de conceptos. Para su representación se usa el diagrama de clases de UML
- *Modelo de comportamiento (o dinámico)*: Ilustra cómo las instancias de clases de análisis pueden interactuar para realizar el comportamiento del sistema especificado por los casos de uso. Este modelo está formado por un conjunto de diagramas de secuencia del sistema y un conjunto de contratos de las operaciones principales del sistema. Las herramientas que se utilizan para su representación son los diagramas de secuencia de UML y los contratos de Larman.

Los puntos de partida para la elaboración de ambos modelos son la lista estructurada de requisitos funcionales y el modelo de casos de uso.

Todo sistema es diferente, y es difícil generalizar sobre los modelos de análisis. Aún así, para un sistema de tamaño y complejidad moderada existen probablemente de cincuenta a cien clases de análisis en el modelo de análisis. Cuando se construye el modelo de análisis, es de vital importancia limitarse solamente a aquellas clases que son parte del vocabulario del dominio del problema. Es fundamental tratar de mantener el modelo de análisis como una declaración concisa y sencilla de la estructura y comportamiento del sistema. Todas las decisiones de implementación se deben dejar para las etapas de diseño e implementación.

Obtención del modelo estático

Un modelo conceptual muestra clases conceptuales significativas en el dominio de un problema; es el artefacto más importante que se crea durante el análisis orientado a objetos. Los casos de uso son un importante artefacto del análisis de requisitos, pero no forman parte del proceso de orientación a objetos. Ponen de relieve una vista de procesos del dominio. La identificación de un conjunto de objetos o clases conceptuales es una parte esencial del análisis orientado a objetos, y forma parte del estudio del dominio del problema. UML contiene notación, en forma de diagramas de clases, para representar los modelos del dominio.

La idea general es que un modelo conceptual es una representación de las clases conceptuales del mundo real, no de componentes software. No se trata de un conjunto de diagramas que describen clases software, u objetos software con responsabilidades.

Proceso general

La etapa esencial en el análisis orientado a objetos es la descomposición de un dominio de interés en clases conceptuales individuales u objetos. Un *modelo conceptual* es una representación *visual* de las clases conceptuales u objetos del mundo real en un dominio de interés.

Utilizando la notación UML, un modelo conceptual se representa con un conjunto de *diagramas de clases* en los que no se define ninguna operación y pueden mostrar: Objetos del dominio o *clases conceptuales, asociaciones entre las clases conceptuales*, y *atributos de las clases conceptuales.*

En la transparencia 10, se muestran los pasos a seguir para la elaboración de un modelo conceptual

Identificar e incorporar conceptos

El objetivo es crear un modelo conceptual de clases conceptuales significativas del dominio de interés. La tarea central es, por lo tanto, identificar las clases conceptuales relacionadas con el escenario que se está diseñando.

No se debe pensar que un modelo conceptual es mejor si contiene pocas clases conceptuales; suele ser verdad justamente lo contrario. Es normal obviar clases conceptuales durante la etapa de identificación inicial, y descubrirlas más tarde al considerar los atributos y asociaciones. No se debe excluir una clase conceptual porque los requisitos no indiquen ninguna necesidad obvia para registrar información sobre ella, o porque la clase conceptual no tiene atributos. Es válido tener clases conceptuales sin atributos, o clases conceptuales con un rol puramente de comportamiento en el dominio, en lugar de un rol de información.

Para la identificación de clases conceptuales existen dos estrategias fundamentales:

1. Utilización de una lista de categorías de clases conceptuales.
2. Identificación de frases nominales.

Utilización de una lista de categorías de clases conceptuales

La creación de un modelo conceptual comienza haciendo una lista de clases conceptuales candidatas. Las transparencias 12, 13, 14, y 15 contienen muchas categorías habituales que, normalmente, merece la pena tener en cuenta, aunque no en ningún orden particular.

Identificación de frases nominales

Otra técnica útil, debido a su simplicidad, es el análisis lingüístico; identificar los nombres y frases nominales en las descripciones textuales de un dominio, y considerarlos como clases conceptuales o atributos candidatos.

Se debe tener cuidado con este método; no es posible realizar una correspondencia mecánica de nombres a clases, y las palabras en lenguaje natural son ambiguas. En cualquier caso, es otra fuente de inspiración. Los casos de uso en formato completo constituyen una descripción excelente a partir de la cual extraer este análisis.

El modelo conceptual es una visualización de los conceptos y vocabulario relevantes del dominio. ¿Dónde se encuentran esos términos? En los casos de uso. En consecuencia, constituyen una rica fuente a explorar para la identificación de frases nominales.

Algunas de las frases nominales son clases conceptuales candidatas, algunas podrían hacer referencia a clases conceptuales que se ignoran, y algunas podrían ser atributos de las clases conceptúales.

Un punto débil de esta estrategia es que diferentes frases nominales podrían representar, entre otras ambigüedades, la misma clase conceptual o atributo. Se recomienda que se combine con la técnica de utilizar la lista de categorías de clases conceptuales.

En las transparencias 17, 18, 19 y 20 se muestra un ejemplo de cómo obtener las clases conceptuales mediante la identificación de frases nominales, y de cómo representar los conceptos definitivos en un diagrama de conceptos.

Identificar e incorporar asociaciones

Resulta útil identificar las asociaciones entre las clases conceptuales que son necesarias para satisfacer los requisitos de información.

Una *asociación* es una relación entre conceptos (o más concretamente, entre instancias de esos conceptos) que indica alguna conexión significativa e interesante.

Una asociación se representa como una línea entre clases con un nombre de asociación. La asociación es inherentemente bidireccional, lo que significa que desde las instancias de cualquiera de las dos clases, es posible el recorrido lógico hacia la otra.

Los extremos de la asociación podrían contener una expresión de multiplicidad que indica la relación numérica entre las instancias de las clases.

Una “flecha de dirección de lectura” opcional indica la dirección de la lectura del nombre de la asociación; no indica la dirección de la visibilidad o navegación. Si no está presente, la convención es leer la asociación de izquierda a derecha o de arriba hacia abajo, aunque no es una regla UML.

La inclusión de asociaciones debe comenzar utilizando la lista de la transparencia 22. Contiene categorías comunes que, normalmente, merece la pena tener en cuenta.

Las asociaciones son importantes, pero un error típico al crear modelos conceptuales es dedicar demasiado tiempo durante el estudio intentando descubrirlas.

Es fundamental entender que es más importante encontrar las clases conceptuales que las asociaciones. La mayoría del tiempo dedicado a la creación del modelo conceptual debería emplearse en la identificación de las clases conceptuales, no en la de las asociaciones.

Los nombres de las asociaciones deben comenzar con una letra mayúscula, puesto que una asociación es un clasificador de enlace entre las instancias.

Como criterio para identificar asociaciones, se recomienda: Centrarse en aquellas asociaciones para las que el conocimiento de la relación necesita conservarse durante algún tiempo y evitar mostrar asociaciones redundantes o derivadas.

Las transparencias 23, 24, 25, 36, 37, 38 y 29 muestran un ejemplo de cómo identificar e incorporar asociaciones en el modelo conceptual.

Agregar atributos

Un *atributo* es un valor de datos lógico de un objeto.

Los atributos se muestran en el segundo compartimiento del rectángulo de la clase. Sus tipos pueden mostrarse opcionalmente.

Tipos de atributos válidos

Hay algunas cosas que no deberían representarse como atributos, sino como asociaciones (ver transparencia 33)

Intuitivamente, la mayoría de los atributos simples son los que, a menudo, se conocen como tipos de datos primitivos, como los números. El tipo de un atributo, normalmente, no debería ser un concepto de dominio complejo, como *Venta* o *Aeropuerto*.

Los atributos en un modelo conceptual deberían ser, preferiblemente, *atributos simples* o *tipos de datos*.

No se debería utilizar los atributos para relacionar las clases conceptuales en el modelo conceptual. La violación más típica de este principio es añadir un *tipo de atributo de clave ajena*, como se hace normalmente en el diseño de bases de datos relacionales, para asociar dos tipos.

En las transparencias 34 y 35 se ilustra con un ekemlo el paso de agregar atributos a un modelo conceptual