MODELADO COMO UNA TÉCNICA DE DISEÑO ORIENTADA A OBJETO

A.S. MARÍA JOSÉ REINA
Centro de Procesamiento de Datos
Prof. Titular Cátedra Análisis de Sistemas.

Un modelo es una abstracción de algo con el propósito de entenderlo antes de construirlo, porque el modelo omite los detalles no esenciales, lo que hace que sea más sencillo de manipular, que la entidad original. La abstracción es una capacidad fundamental del ser humano que le permite manejar sistemas muy complejos. El desarrollo de sistemas de hardware y de software no es una excepción a ello. Para construir sistemas complejos, el desarrollador debe abstraer diferentes vistas del sistema, construir un modelo con notación precisas, verificar que el modelo satisfaga los requerimientos del sistema y gradualmente sumar detalles para transformar el modelo en una implementación.

a. Metodología para técnicas de modelado objeto (OMT)

La metodología consiste en construir un *modelo de* una aplicación y luego ir sumándole detalles durante el diseño del sistema. Llamemos a este acercamiento TÉCNICAS DE MODELADO OBJETO (OTM). Esta metodología combina tres formas distintas de ver el modelo de sistemas:

. El *modelo Objeto* representa el aspecto estático y estructural del sistema.

. El *modelo dinámico* representa el aspecto temporal y control del sistema.

. El *modelo funcional* representa el aspecto de transformación y funciones del sistema.

Los diferentes modelos no son completamente independientes, un sistema es más que un conjunto de partes independientes, pero cada modelo puede ser examinado por sí solo. La interconexión entre los modelos es limitada y explícita.

El *análisis* es el primer paso de la metodología OMT, consiste en

crear modelo del mundo real que sea preciso, concreto y entendible. Antes de crear o construir cualquier cosa compleja. El constructor debe entender los requerimientos y el entorno del mundo real donde él existirá.

El análisis comienza con una sentencia generada por el cliente o el desarrollados Esta *sentencia problema* es informal e incompleta. El análisis la vuelva más precisa y expone las ambigüedades e inconsistencias.

El modelo de análisis ubica los 3 aspectos de los objetos: *estructura estática* (modelo objeto), *secuencia de iteraciones* (modelo dinámico) y *transformación de datos* (modelo funcional). No para todos los problemas estos 3 submodelos tienen igual importancia. Todos los problemas tienen *modelos objetos* útiles derivados de las entidades del mundo real. Problemas concernientes a iteraciones y tiempos, como de interfaces y procesos de control, tienen importantes *modelos dinámicos.* Problemas que tiene cómputos significativos, como compiladores y cálculos de ingeniería, tiene importantes *modelos funcionales.* El análisis no es un proceso mecánico. El analista debe comunicarse con el encuestado para clarificar ambigüedades y mal conceptos. El modelo de análisis facilita la comunicación precisa.

b. Modelado de Objeto

El primer paso en el análisis de los requerimientos es la construcción de un *modelo objeto.* El modelo objeto muestra la estructura estática de los datos del sistema del mundo real. Describe los *objetos clase* y sus relaciones entre ellos. El modelo objeto es el primero en definirse porque las estructuras estáticas son mejor definidas, menos dependientes de los detalles de la aplicación y más estables que la solución que las envuelve.

Pasos para llevar a cabo la construcción de un modelo objeto:

Los objetos incluyen entidades físicas como: casas, empleados y máquinas. Así como también conceptos como: transacciones, plan de pagos, asignación de asientos, etc. No todas las clases son explícitas en una sentencia problema. Por lo general son sustantivos.

Habrá que descartar las clases innecesarias o incorrectas de acuerdo al siguiente criterio:

- *Clases redundantes:* si dos clases expresan la misma información, el nombre más descriptivo se elige.

- *Clase irrelevante:* si la clase tiene poco o nada que ver con el problema, deberá ser eliminada.

-*Atributos:* son nombres que describen objetos individuales, por Ej.: nombre, edad, peso y domicilio. Si la existencia independiente de una propiedad es importante, deja de ser atributo para convertirse en una clase.

-*Operación:* si un nombre describe un operación aplicada a los objetos y no manipula sobre sus propios derechos, no es una clase.

-*Roles:* el nombre de una clase suele reflejar su naturaleza intrínseca y no el rol que ésta juega.

Las palabras aisladas pueden tener diferentes interpretaciones, por ello se prepara el *diccionario de datos.*

Luego procederemos a identificar cualquier dependencia entre dos o más clases. Una referencia de una clase a otra es una *asociación.* Las asociaciones deben eliminarse del módulo de análisis para preservar un diseño libre.

Las asociaciones a menudo corresponden a verbos estáticos. Estos incluyen locación física: próximo a, etc. Se descartarán las asociaciones innecesarias o incorrectas, usando el siguiente criterio:

-*Asociación entre clases eliminadas:* si una de las clases de una asociación ha sido eliminada, la asociación debe ser eliminada.

-*Acciones:* una asociación deberá describir una propiedad estructural del dominio de la aplicación no un evento pasajero.

-*Asociación ternaria:* la mayoría de las asociaciones entre tres o más clases pueden ser descompuestas en asociaciones binarias.

- *Asociaciones derivadas:* elimine asociaciones que pueden ser definidas en términos de otras asociaciones, porque son redundantes.

Deberá identificar los *atributos,* que son propiedades de objetos individuales, como el nombre, paso, velocidad o color. Los atributos no pueden ser objetos. Use asociaciones para mostrar cualquier relación entre objetos.

Atributos usualmente corresponde a un sustantivo seguido por una frase posesiva, como "el color del auto". Obtenga primero los atributos principales, los detalles finos pueden ser adicionados después. Los atributos derivados son omitidos Ej.: edad, puede ser obtenida por la fecha de nacimiento.

Elimine los atributos innecesarios o incorrectos con el siguiente criterio:

- *Objetos:* si la existencia independiente de una entidad es importante, es un objeto. Ej.: "Jefe" es un objeto y "salario" es atributo. La distinción muchas veces depende de la aplicación.

- *Calificadores:* si el valor de un atributo depende de un contexto

particular. Ej.: "n° de empleado" no es la única propiedad de una persona que tiene dos trabajos.

El próximo paso es organizar las clases usando herencia para compartir estructura común.

Las herencias pueden ser sumadas en dos direcciones: generalizando aspectos comunes de clases en *superclases* (bottom-up) o retinando clases existentes en *subclases especializadas* (top-down).

Piense en preguntas que usted quiere hacer al sistema. Hay preguntas útiles que no se pueden responder? Esto indica que falta información. Si algo que parece simple en el mundo aparece como complejo en el modelo, debe faltar algo.

Un modelo objeto raramente es correcto después de una sola pasada. El desarrollo de un software entero es un proceso de continua interacción, diferentes partes del modelo a menudo están en diferente estado de terminación. Si una deficiencia es encontrada, es necesario volver atrás a un estado de comienzo para corregir. Algunos refinamientos solo pueden lograrse después que el modelo dinámico y funcional, han sido completados.

Es el último paso del modelado, es agrupar clases en módulos. Un módulo es un conjunto de clases que capturan algunos subconjuntos lógicos del módulo entero. Los módulos pueden ser de diferentes tamaños.

c. Modelado Dinámico

El *modelo dinámico* muestra el comportamiento dependiendo del tiempo del sistema y de los objetos en él.

Comience el análisis dinámico mirando los eventos. Luego resuma la secuencia de eventos permisibles para cada objeto en su diagrama de estado. La ejecución de algoritmos no es relevante durante el análisis si no son manifestaciones externamente visibles, los algoritmos son parte de la implementación.

El modelo dinámico es insignificante para la reposición estática de datos, como una base de datos. El modelo dinámico es importante para sistemas interactivos.

Los siguientes pasos son los a seguir para construir un modelo dinámico:

Preparar uno o más diálogos típicos entre el usuario y el sistema, para ver el comportamiento del sistema. Primero prepare escenarios

para casos "normales", interacción sin entradas inusuales o errores de condición. Luego considere casos "especiales", tales como omisión en la entrada, valores máximo y mínimo o valores repetidos. Luego considere casos de error incluyendo valores erróneos o respuestas mal logradas. *Un escenario es una secuencia de eventos.* Un *evento* ocurre siempre que la información es intercambiada entre un objeto en el sistema y un agente externo, tal como el usuario, un censor u otra tarea. Cada vez que la información es entrada al sistema o sale del sistema, un evento ocurre. Para cada evento, identifica el actor (sistema, usuario y otro agente externo).

La mayoría de las interacciones pueden separarse en dos partes: aplicaciones lógicas e interface usuario.

El analista debería concentrarse primeramente en el flujo y control de la información antes que el formato de presentación, el mismo programa puede aceptar entradas desde la línea de comando, archivo, mouse, etc.

Los detalles exactos no son importantes en este punto, menos aún el estilo de los mensajes. Lo importante es la información intercambiada, en cualquier forma.

Examine los escenarios para identificar los eventos externos. Eventos incluyen signos, entradas, decisiones, interrupciones, transiciones y acciones del usuario o dispositivos externos. Los pasos internos de la computadora *no son eventos.* Una acción a través del objeto que transmite información *es un evento.* La mayoría de las interacciones y operaciones Objeto-Objeto *son eventos.*

Agrupe bajo un mismo nombre de evento aquellos que producen el mismo efecto en el flujo de control, aún si los parámetros son diferentes. Uno debe decidir cuando los diferentes valores cuantitativos son suficientemente importantes como para distinguirse.

Usted debe construir un *diagrama de estado* antes de clasificar los eventos. Prepare un diagrama de estado para cada clase de objeto que lo necesite, mostrando los eventos que los objetos reciben y envían. Cada rama en el control de flujos es representada por un estado con más de una salida de transmisión.

Chequee la complementación y consistencia a nivel de sistema con el diagrama de estado para que cada clase esté completa. Cada evento debe tener un enviador y receptor, ocasionalmente el mismo objeto.

d. Modelado Funcional

El modelado funcional muestra como los valores están compuestos,

sin consideración de secuencia, decisión o estructura de objeto. El *modelo funcional* muestra que valores dependen de otros y las funciones que los relacionan. Los DFD son útiles para mostrar las dependencias funcionales. Las funciones son expresadas de diferentes formas, incluyendo lenguaje natural, ecuaciones matemáticas o seudocódigos.

Los procesos en un DFD corresponden a actividades o acciones en un diagrama de estado de clases. Los flujos en un DFD corresponden a objetos o valores de atributos de valores de un diagrama de objetos. Es conveniente construir el *modelo funcional* después de los *modelos objeto y dinámico.*

Pasos involucrados en la construcción de un modelo funcional:

Comenzar con *listar valores de entrada y salida.* Valores de entrada y salida son parámetros de eventos entre el sistema y el mundo exterior.

Construir un DFD que muestre como cada valor de salida es computado desde los valores de entrada. Los DFD sólo especifican dependencias entre operaciones, no muestran decisiones o secuencias de operación, de hecho algunas operaciones pueden ser opcionales o excluyentes mutuamente.

Cuando el DFD ha sido refinado lo suficiente, escriba una descripción de cada función.

Contraints son dependencias funcionales entre objetos que no están relacionados por una dependencia ES. Constraints puede ser sobre dos objetos al mismo tiempo, entre instancias de un mismo objeto en diferentes tiempos o entre instancias de un mismo objeto en diferentes tiempos o entre instancias de diferentes objetos en diferentes tiempos. Precondiciones en funciones son contraints que los valores de entrada deben satisfacer y post condición son constraints de los valores de salida asegurados.

Ej.: es un cajero automático un contraints sería "nunca el saldo negativo" puede ser mayor al límite del crédito permitido. Este contraints no especifica que hacer, el analista deberá incorporar el constraints en los modelos dinámico y funcional para completar ¡a especificación.

Usando esta metodología, OTM, se producen sistemas que son más estables con respecto a los cambios en los requerimientos que las aproximaciones tradicionales. No debemos olvidarnos que es un análisis iterativo. Cada iteración suma o clarificar más bien características que modificar trabajos que ya han sido realizados, por lo tanto hay menor chance de introducir inconsistencias y errores.

Bibliografía

OBJECT-ORIENTED MODELING AND DESIGN de Rumbaugh, Blaha, Premerlani, Eddy, Lorensen - Prentice Hall International.

ANÁLISIS Y DISEÑO ORIENTADO A OBJETO de J. Martin - Prentice Hall International.

OBJECT-ORIENTED SOFTWARE CONSTRUCCIÓN de Meyer - Prentice Hall International.

OBJECT-ORIENTED SYSTEM ANALYSIS de Shlaer - Yourdon Press.