
8.1. Presentación del capítulo

A continuación en este capítulo trata sobre la actividad principal del análisis orientado a objetos, encontrar las clases de análisis. Si quiere entender la actividad de UP en la que se encuentran las clases de análisis, diríjase al siguiente apartado. Si necesita saber qué es una clase de análisis, consulte el apartado de su mismo nombre.

En este capítulo se describe cómo encontrar clases de análisis. Presentamos tres técnicas específicas: análisis nombre/verbo, análisis CRC y estereotipos de clase de análisis RUP, y también una consideración general de otras fuentes posibles para clases.

8.2. Actividad UP. Analizar un caso de uso

Los resultados del workflow de UP Analizar un caso de uso que se muestra en la figura 8.2 son clases de análisis y realizaciones de caso de uso. En este capítulo nos centramos en clases de análisis. Consideramos las realizaciones de caso de uso en el capítulo 12; éstas son colaboraciones entre objetos que muestran cómo sistemas de objetos que interactúan pueden realizar el comportamiento del sistema expresado en los casos de uso.

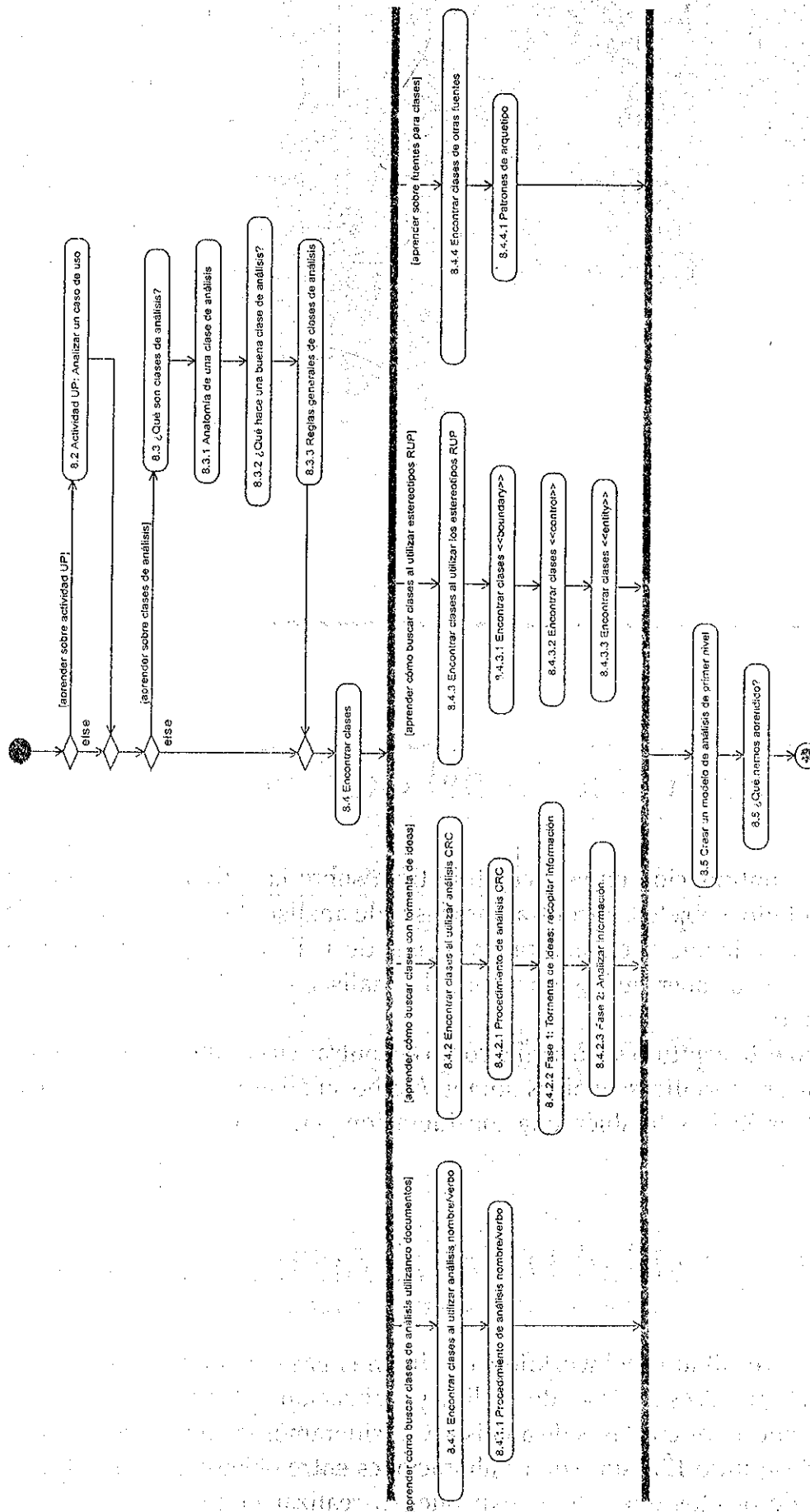


Figura 8.1.

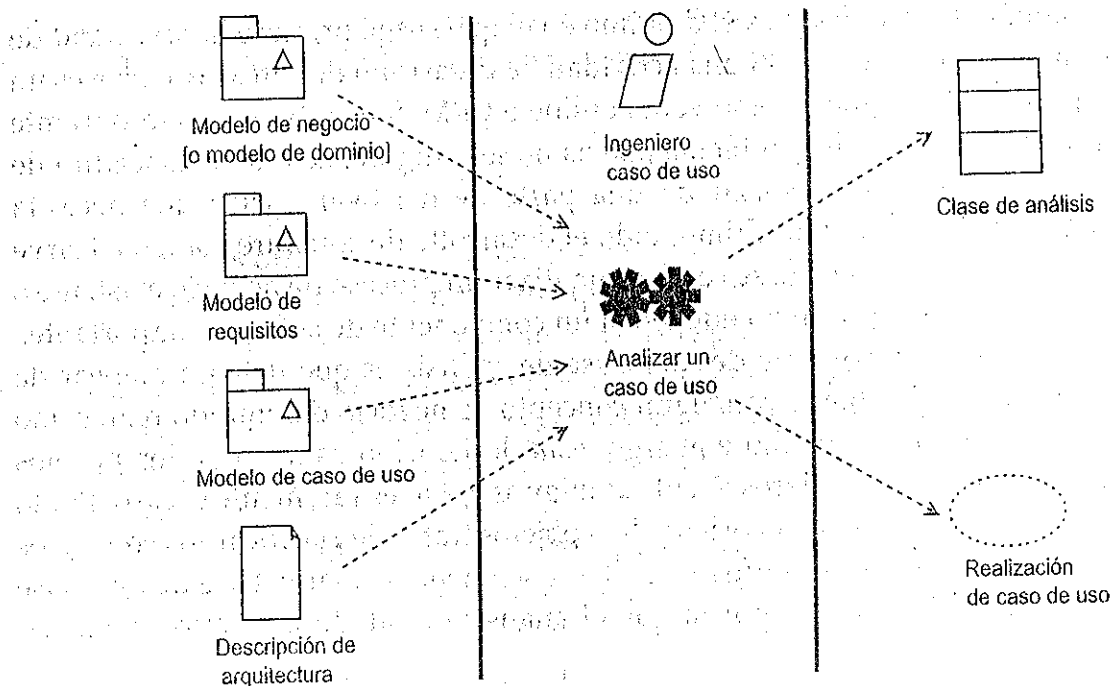


Figura 8.2. Adaptada de la figura 8.25 [Jacobson '1] con permiso de Addison-Wesley.

Merece la pena mirar las entradas en el workflow Analizar un caso de uso.

- Modelo de negocio: Puede o no tener un modelo de negocio disponible que esté relacionado con el sistema que está modelando. Si lo tiene, éste es una excelente fuente de requisitos.
- Modelo de requisitos: Hemos descrito la creación de este modelo en el capítulo 3. Estos requisitos (mostrados en gris para indicar que el artefacto se modifica a partir de la figura original) proporcionan entrada de utilidad para el proceso de modelado de caso de uso. En particular, los requisitos funcionales sugerirán casos de uso y actores. Los requisitos no funcionales sugerirán aquello que puede tener en mente cuando construya el modelo de caso de uso.
- Modelo de caso de uso: Hemos tratado la creación del modelo de caso de uso en los capítulos 4 y 5.
- Descripción de arquitectura: Una instantánea de los aspectos importantes desde el punto de vista de la arquitectura del sistema. Puede incluir extractos de los modelos UML incorporados en un texto explicativo. Esto lo crean los arquitectos con entradas de analistas/diseñadores:

8.3. ¿Qué son clases de análisis?

Las clases de análisis son clases que:

- Representan una abstracción en el dominio del problema.
- Deberían mapearse con conceptos de negocio del mundo real (y estar nombrados en consecuencia).

El dominio del problema es el dominio en el que surge primero la necesidad de un sistema de software (de ahí una actividad de desarrollo de software). Esta es un área específica del negocio como venta online o CRM. Sin embargo, es importante indicar que el dominio del problema podría no ser ninguna actividad específica de negocio, sino que podría surgir de una parte de hardware físico que necesita software para utilizarlo. Por último, todo el desarrollo de software comercial sirve alguna necesidad de negocio, sea esa automatizar un proceso de negocio existente o desarrollar un nuevo producto que tenga un componente de software importante.

El aspecto más importante de una clase de análisis es que debería mapear de forma clara y nada ambigua con algún concepto de negocio del mundo real, como cliente, producto o cuenta. Sin embargo, esta declaración asume que los propios conceptos de negocio son claros y nada ambiguos y esto es raramente el caso. Por lo tanto, es el trabajo del analista orientado a objetos tratar de clarificar los conceptos de negocio inapropiados o confusos en algo que pueda formar la base de una clase de análisis. Esta es la razón por la que el análisis orientado a objetos puede ser difícil.

Por lo tanto, el primer paso para crear software orientado a objetos es aclarar el dominio del problema. Si contiene conceptos de negocio claramente definidos y tiene una estructura funcional sencilla, la solución está ahí para cogerla. La mayor parte de este trabajo se realizará en el workflow de requisitos en las actividades de captar requisitos y crear un modelo de caso de uso y glosario del proyecto. Sin embargo, mucha más aclaración ocurre en la construcción de clases de análisis y realizaciones de casos de uso.

Es importante que todas las clases en el modelo de análisis sean clases de análisis en lugar de clases que surgen de consideraciones de diseño (el dominio de solución). Cuando llega a un diseño detallado, puede encontrar que las clases de análisis se convierten en una o más clases de diseño.

Aunque en el capítulo anterior empezamos considerando objetos específicos, ahora entenderá que el verdadero objetivo del análisis orientado a objetos es encontrar las clases de esos objetos. De hecho, encontrar las clases correctas de análisis es la clave del análisis y diseño orientado a objetos. Si las clases no son correctas en análisis, entonces el resto del proceso de ingeniería de software, que se promulga en los workflows de requisitos y análisis, estará en peligro. Por lo tanto, es totalmente crucial que dedique tiempo al workflow de análisis para asegurarse de que el conjunto correcto de clases de análisis se ha identificado. Su tiempo estará bien empleado ya que con toda seguridad le ahorrará tiempo más adelante.

En este libro, nos centramos en el desarrollo de sistemas de negocio, ya que esto es en lo que están más implicados la mayoría de los analistas y diseñadores orientados a objetos. Sin embargo, el desarrollo de sistemas incorporados es un caso especial de desarrollo normal de negocio y se aplican los mismos principios. Los sistemas de negocio están dominados por requisitos funcionales y por lo tanto son las actividades de requisitos y análisis las que son más difíciles. Los sistemas incorporados están dominados por requisitos no funcionales que surgen del hardware en el que el sistema está incorporado. En este caso el análisis tiende a ser sencillo pero el diseño puede ser difícil. Los requisitos son importantes para todos los tipos de

sistemas, y para algunos sistemas incorporados, como controladores para máquinas de rayos X, pueden ser una cuestión de vida o muerte.

8.3.1. Anatomía de una clase de análisis

Las clases de análisis deberían presentar un conjunto de atributos de "muy alto nivel". Estos indican los atributos que tendrán las clases de diseño resultantes. Podríamos decir que las clases de análisis capturan atributos candidatos para las clases de diseño. Las operaciones de clase de análisis especifican, en un alto nivel, los servicios clave que debe ofrecer la clase. En diseño, se convertirán en operaciones implementables. Sin embargo, una operación a nivel de análisis se desglosará en más de una operación a nivel de diseño.

Ya hemos tratado la sintaxis UML para clases en gran detalle en el capítulo 7, pero en análisis, solamente se utiliza un pequeño subconjunto de esa sintaxis. Por supuesto, el analista es libre de añadir cualquier adorno que considere necesario para hacer que el modelo sea más claro. Sin embargo, la sintaxis básica de una clase de análisis siempre evita los detalles de implementación. Después de todo, en análisis estamos tratando de capturar lo más importante o idea principal.

Una forma mínima para una clase de análisis consta de lo siguiente:

- **Nombre:** Es obligatorio.
- **Atributos:** Los nombres de atributo son obligatorios aunque solamente un subconjunto importante de atributos candidatos se pueden modelar en este punto. Los tipos de atributo se consideran opcionales.
- **Operaciones:** En análisis, las operaciones pueden ser declaraciones de responsabilidades de la clase a muy alto nivel. Los parámetros de operación y tipos de retorno solamente se muestran donde son importantes para entender el modelo.
- **Visibilidad:** Generalmente no se muestra.
- **Estereotipos:** Se pueden mostrar si mejoran el modelo.
- **Valores etiquetados:** Se pueden mostrar si mejoran el modelo.

Un ejemplo se proporciona en la figura 8.3.

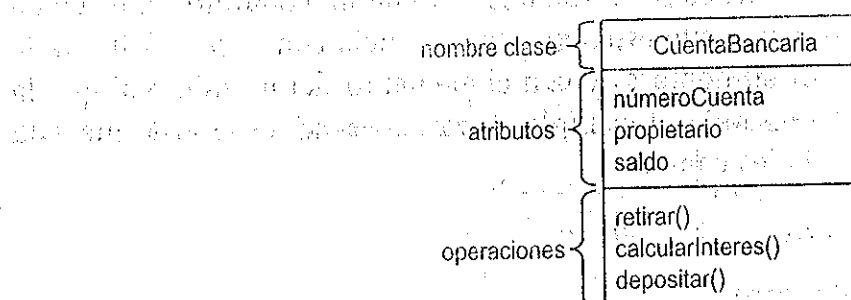


Figura 8.3.

La idea de una clase de análisis es que trata de capturar la esencia de la abstracción y deja los detalles de la implementación hasta que llega al diseño.

8.3.2. ¿Qué hace una buena clase de análisis?

Podemos resumir lo que hace una buena clase de análisis en los siguientes puntos:

- Su nombre refleja su intención.
- Es una abstracción que modela un elemento específico del dominio del problema.
- Mapea en una característica claramente identificable del dominio del problema.
- Tiene un pequeño conjunto de responsabilidades bien definidas.
- Tiene una alta cohesión.
- Tiene un bajo acoplamiento.

En análisis, trata de modelar un aspecto del dominio del problema de forma precisa y concisa desde la perspectiva del sistema que trata de construir. Por ejemplo, si modela un cliente en un sistema bancario, querrá capturar el nombre, dirección del cliente, etc., pero no es probable que esté interesado por su preferencia de ventana o pasillo en un avión. Tiene que centrarse en aquellos aspectos del mundo real que son importantes desde la perspectiva del sistema que está creando.

Puede hacerse una primera idea de si una clase es una "buena" clase simplemente por su nombre. Si considera un sistema de comercio electrónico, *Cliente* parece que se referirá a algo bastante preciso en el mundo real y sería un buen candidato para una clase. *CarroCompra* también parecería una buena abstracción; sabemos casi de forma intuitiva cuál será su semántica. Sin embargo, algo como *VisitanteSitioWeb* parece tener una semántica bastante vaga y en realidad suena como un rol que un *Cliente* desempeña en relación con el sistema de comercio electrónico. Siempre debería tratar de buscar una abstracción "concisa"; algo que tiene una semántica clara y obvia.

Una responsabilidad es un contrato u obligación que una clase tiene con sus clientes. De forma esencial, una responsabilidad es un servicio que una clase ofrece a otras clases. Es crucial que sus clases de análisis tengan un conjunto cohesivo de responsabilidades que directamente estén en concordancia con el propósito de la clase (según se expresa por su nombre) y con el elemento del mundo real que la clase está modelando. Regresando al ejemplo *CarroCompra*, esperaríamos que esta clase tuviera responsabilidades como:

- Añadir artículo a carro.
- Eliminar artículo de carro.
- Mostrar artículos en carro.

Este es un conjunto cohesivo de responsabilidades, todas sobre mantener una colección de artículos que el cliente ha elegido. Es cohesivo porque todas las responsabilidades trabajan hacia el mismo objetivo, mantener el carro de compra del cliente. De hecho, podríamos resumir estas tres responsabilidades como una responsabilidad de muy alto nivel denominada "mantener carro".

Ahora, podría también añadir las siguientes responsabilidades a CarroCompra:

- Validar tarjeta de crédito.
- Aceptar pago.
- Imprimir recibo.

Pero estas responsabilidades no parecen encajar con el propósito o semántica intuitiva de los carros de compra. No son tan cohesivas y se deberían asignar en cualquier otra parte, quizás a la clase EmpresaTarjetaCredito, una clase Facturar, y una clase ImprimirRecibo. Es importante distribuir responsabilidades de forma apropiada sobre las clases de análisis para maximizar la cohesión dentro de cada clase. Por último, las buenas clases tienen la cantidad mínima de acoplamiento con otras clases. Medimos el acoplamiento entre clases por el número de otras clases con las que una clase dada tiene relaciones. Una distribución uniforme de responsabilidades entre clases tenderá a resultar en un acoplamiento bajo. La localización de control o de muchas responsabilidades en una sola clase tiende a incrementar el acoplamiento a esa clase. Consideramos formas de maximizar la cohesión y minimizar el acoplamiento en el capítulo 15.

8.3.3. Reglas generales de clases de análisis

Aquí tiene algunas reglas generales para crear clases de análisis bien estructuradas.

- De tres a cinco responsabilidades por clase: Normalmente las clases se deberían mantener lo más simples posibles, y esto normalmente limita el número de responsabilidades que pueden soportar a entre tres y cinco. Nuestro ejemplo anterior de CarroCompra es un buen ejemplo de una clase con un número pequeño y manejable de responsabilidades.
- Ninguna clase permanece sola: La esencia de buen análisis y diseño orientado a objetos es que las clases colaboran entre sí para proporcionar beneficio a los usuarios. Como tal, cada clase se debería asociar con un pequeño número de otras clases con las que colabore para distribuir el beneficio deseado. Las clases pueden delegar parte de sus responsabilidades a otras clases "de ayuda" que están dedicadas a esa función específica.
- Tenga cuidado de muchas clases muy pequeñas, algunas veces puede ser difícil encontrar el equilibrio correcto: Si el modelo parece tener muchas clases pequeñas con una o dos responsabilidades cada una, entonces debería examinar esto detenidamente con idea de consolidar algunas de las clases pequeñas en más grandes.

- **Tenga cuidado de pocas clases pero muy grandes:** Lo contrario de lo anterior es un modelo que tiene pocas clases, donde muchas de ellas tienen un gran número (>5) de responsabilidades. La estrategia aquí es examinar estas clases en turno y ver si cada una se puede descomponer en dos o más clases más pequeñas con el número apropiado de responsabilidades.
- **Tenga cuidado de los "functoids":** Un "functoid" es una función procedimental normal disfrazada como una clase. Grady Booch contó la anécdota de un modelo de un sistema muy sencillo que tenía miles de clases. En un examen más detallado, toda clase tiene exactamente una operación denominada `tonto()`. Los functoid son siempre un peligro cuando los analistas acostumbrados a la técnica de la descomposición funcional de arriba abajo se acercan al análisis y diseño orientado a objetos la primera vez.
- **Tenga cuidado de las clases omnipotentes:** Estas son clases que parecen hacerlo todo. Busque clases con "sistema" o "controlador" en su nombre. La estrategia para tratar con este problema es ver si las responsabilidades de la clase omnipotente se encuentran en subconjuntos cohesivos. Si es así, quizá cada uno de estos conjuntos cohesivos de responsabilidades se pueda destacar en una clase aparte. Estas clases más pequeñas colaborarán para implementar el comportamiento ofrecido por la clase omnipotente original.
- **Evite árboles de herencia muy profundos:** La esencia de diseñar una buena jerarquía de herencia es que cada nivel de abstracción en la jerarquía debería tener una finalidad bien definida. Es fácil añadir muchos niveles que no sirven para ningún propósito de utilidad. De hecho, un error común es utilizar la herencia para implementar un tipo de descomposición funcional donde cada nivel de abstracción tiene solamente una responsabilidad. Esto conduce a un modelo complejo difícil de entender. En análisis, la herencia solamente se utiliza donde existe una jerarquía de herencia clara y obvia que surge directamente del dominio del problema.

En este último apartado tenemos que aclarar lo que queremos decir por árbol de herencia "profundo". En análisis, donde las clases representan elementos de negocio, "profundo" serían tres niveles de herencia o más. Esto es porque los elementos de negocio tienden a formar jerarquías de herencia que son amplias en lugar de profundas. En diseño, donde el árbol consta de clases del dominio de solución, la definición de "profundo" depende del lenguaje de implementación al que se dirija. En Java, C++, C#, Python y Visual Basic, consideramos tres o más niveles como profundo. Sin embargo, en Smalltalk los árboles de herencia pueden ser mucho más profundos que esto debido a la estructura del sistema Smalltalk.

8.4. Encontrar clases

En el resto de este capítulo consideramos el tema principal del análisis y diseño orientado a objetos, encontrar las clases de análisis.

Como destaca Meyer en *Object Oriented Software Construction* [Meyer 1], no existe un algoritmo sencillo para encontrar las clases de análisis correctas. Si un algoritmo así existe, entonces equivaldrá a un medio infalible de diseñar software orientado a objeto y esto es tan improbable como encontrar un medio infalible de probar los teoremas matemáticos.

Aún así existen técnicas probadas y testadas que llevan hacia una buena respuesta y las presentamos aquí. Estas implican analizar texto y entrevistar a usuarios y expertos de dominio. Pero a pesar de todas las técnicas, encontrar las clases "correctas" depende de la perspectiva, habilidad y experiencia del analista.

8.4.1. Encontrar clases al utilizar análisis nombre/verbo

El análisis nombre/verbo es una forma muy sencilla de analizar texto para tratar de encontrar clases, atributos y responsabilidades. En esencia, los nombres y frases nominales en el texto indican clases o atributos de clases, y los verbos y frases verbales indican responsabilidades u operaciones de una clase. El análisis nombre/verbo se lleva utilizando muchos años y funciona bien ya que está basado en un análisis directo del lenguaje del dominio del problema. Sin embargo, tiene que estar pendiente de sinónimos y homónimos ya que esto puede dar lugar a clases falsas.

También debería tener mucho cuidado si el dominio del problema está mal definido. En este caso, trate de recopilar tanta información del dominio de tantas personas como le sea posible. Busque dominios de problemas similares fuera de su organización.

Quizás el aspecto más difícil del análisis nombre/verbo es encontrar las clases "ocultas". Estas son clases que son intrínsecas al dominio del problema pero puede que nunca se mencionen explícitamente. Por ejemplo, en un sistema de reservas para una empresa de vacaciones, oír a los grupos de decisión hablar sobre reservas, etc., pero la abstracción más importante, Pedido, es posible que nunca se mencione explícitamente si no existe en los sistemas de negocio actuales. Normalmente sabe cuando ha encontrado una clase oculta porque todo el modelo parece encajar de repente con la introducción de esta nueva abstracción. Esto sucede sorprendentemente a menudo, de hecho, si alguna vez tenemos problemas con un modelo de análisis y no parece que tenga sentido, vamos a buscar clases ocultas. Esto nos hace formularnos preguntas y mejorar nuestro entendimiento del dominio del problema.

8.4.1.1. Procedimiento de análisis nombre/verbo

El primer paso en el análisis nombre/verbo es recopilar tanta información importante como sea posible. Algunas fuentes posibles de información son:

- El modelo de requisitos.
- El modelo de caso de uso.

- El glosario del proyecto.
- Cualquier otra cosa (arquitectura, documentos de visión, etc.).

Después de recopilar la documentación, analízela de forma sencilla al destacar (o grabar de alguna forma) lo siguiente:

- Nombres, por ejemplo, vuelo.
- Frases nominales, por ejemplo, número de vuelo.
- Verbos, por ejemplo, asignar.
- Frases verbales, por ejemplo, verificar tarjeta crédito.

Los nombres y frases nominales pueden indicar clases o atributos de clase. Los verbos y frases verbales pueden indicar responsabilidades de clases.

Si se encuentra con algún término que no entiende durante este proceso, busque una aclaración inmediata en un experto y añada el término al glosario del proyecto. Tome la lista de nombres, frases nominales, verbos y frases verbales y utilice el glosario del proyecto para solucionar cualquier sinónimo y homónimo. Esto crea una lista de clases, atributos y responsabilidades candidatos.

Una vez que tiene esta lista hace una asignación tentativa de los atributos y responsabilidades a las clases. Puede realizar esto al incorporar las clases en una herramienta de modelado y añadir las responsabilidades como operaciones a las clases. Si ha encontrado algún atributo candidato, puede asignarlo también a las clases. También podría haberse hecho una idea de las relaciones entre ciertas clases (los casos de uso son una buena fuente de estos), por lo que puede añadir algunas asociaciones candidatas. Esto le proporciona un primer modelo de clase que puede mejorar con más análisis.

8.4.2. Encontrar clases al utilizar análisis CRC

Una buena forma de hacer que el usuario se implique en la búsqueda de clases es utilizar el análisis CRC, CRC significa Clase, Responsabilidades y Colaboradores. Esta técnica utiliza la herramienta de análisis más potente del mundo, el post-it. Tan popular es el método CRC que existe una historia de que una vez una empresa comercializó post-it ya marcados con nombres de clase, responsabilidades y colaboradores.

Empieza marcando algunos post-it como muestra la figura 8.4. La nota se divide en tres compartimentos. En el compartimiento superior graba el nombre de la clase candidata; en el compartimiento izquierdo, las responsabilidades; y en el derecho los colaboradores. Los colaboradores son otras clases que pueden colaborar con esta clase para realizar una parte de la funcionalidad del sistema. El compartimiento de colaboradores proporciona una forma de grabar relaciones entre clases. Otra forma de capturar relaciones (que preferimos) es pegar las notas en una pizarra y dibujar líneas entre las clases colaboradoras.

Nombre clase: CuentaBancaria	
Responsabilidades: Mantener saldo	Colaboradores: Banco



Figura 8.4.

8.4.2.1. Procedimiento de análisis CRC

El análisis CRC siempre se debería utilizar junto con análisis nombre/verbo de casos de uso, requisitos, glosario y otra documentación relevante, a menos que el sistema sea muy sencillo. El procedimiento de análisis CRC es sencillo y la clave está en separar la información que se recopila del análisis de información. CRC se ejecuta mejor como una actividad de dos fases.

8.4.2.2. Fase 1. Tormenta de ideas: recopilar información

Los participantes son analistas, grupos de decisión y expertos de dominio orientados a objeto. También necesita un facilitador. El procedimiento es el siguiente:

1. Explicar que se trata de una verdadera tormenta de ideas.
 1. Todas las ideas se aceptan como buenas ideas.
 2. Las ideas se graban pero no se debaten, nunca se discute sobre algo, simplemente se escribe y se sigue adelante. Todo se analizará más adelante.
2. Pedir a los miembros del equipo que nombren los elementos que funcionan en su dominio de negocio, por ejemplo, cliente, producto.
 1. Escribirlo en un post-it; se trata de una clase candidata o atributo de una clase.
 2. Pegar la nota en una pared o pizarra.
3. Pedir al equipo que indique las responsabilidades que pudieran tener; grabarlas en el compartimiento de responsabilidades de la nota.
4. Trabajar con el equipo, tratar de identificar clases que podrían trabajar conjuntamente. Reorganizar las notas en la pizarra para reflejar esta organización y dibujar líneas entre ellas. De forma alternativa, grabar a los colaboradores en el compartimiento de colaboradores de la nota.

8.4.2.3. Fase 2. Analizar información

Los participantes son analistas y expertos de dominio orientados a objetos. ¿Cómo decide qué post-it deberían convertirse en clases y cuáles deberían convertirse en

atributos? Como se ha indicado anteriormente, las clases de análisis deben representar una abstracción concisa en el dominio del problema. Ciertos post-it representarán conceptos clave de negocio y claramente necesitan convertirse en clases. Otras notas pueden convertirse en clases o atributos. Si una nota parece ser una parte de otra nota, ésta es una buena indicación de que representa un atributo. Igualmente, si una nota no parece ser particularmente importante o tiene un comportamiento poco interesante, vea si se puede hacer un atributo de otra clase.




Si se tiene dudas sobre una nota, hágala una clase. Lo importante es acertar y luego cerrar este proceso; siempre puede modificar el modelo más adelante.

8.4.3. Encontrar clases al utilizar los estereotipos RUP

Una técnica de utilidad procede de RUP en la forma de estereotipos RUP. La idea es que considera tres tipos distintos de clases de análisis durante su actividad de análisis. Ésta es una forma de centrar su análisis en aspectos específicos del sistema. Consideramos esto una técnica opcional que puede utilizar para complementar las técnicas de análisis importantes, nombre/verbo y CRC presentadas anteriormente.

Tres tipos distintos de clase de análisis se pueden distinguir por los estereotipos que se muestran en la tabla 8.1.

Tabla 8.1.

Estereotipo	Icono	Semántica
<<boundary>>		Una clase mediadora entre el sistema y su entorno.
<<control>>		Una clase que encapsula comportamiento específico de caso de uso.
<<entity>>		Una clase que se utiliza para modelar información persistente sobre algo.

Vemos cómo encontrar cada uno de estos tipos de clase de análisis en los siguientes subapartados.

8.4.3.1. Encontrar clases <<boundary>>

Estas clases existen en el límite de su sistema y se comunican con actores externos. Encuentra esas clases al considerar el sujeto (límite del sistema) y descubrir qué clases median entre el sujeto y su entorno. Según RUP existen tres tipos de clase <<boundary>>:

1. Clases de interfaz de usuario: Clases que hacen de interfaz entre el sistema y los usuarios.

2. Clases de interfaz del sistema: Clases que hacen de interfaz con otros sistemas.

3. Clases de interfaz de dispositivo: Clases que hacen de interfaz con dispositivos externos como sensores.

Toda comunicación entre un actor y un caso de uso en su modelo debe estar activada por algún objeto en su sistema. Estos objetos son instancias de clases límite. Puede averiguar qué tipo de clase límite está indicado al considerar lo que representa el actor (véase la tabla 8.2).

Tabla 8.2.

Actor	Indica
Representa una persona.	Clase de interfaz de usuario.
Representa un sistema.	Clase de interfaz de sistema.
Representa un dispositivo.	Clase de interfaz de dispositivo.

Cuando una clase límite sirve más de un actor, estos actores deberían ser del mismo tipo (representando una persona, sistema o dispositivo). Puede ser una indicación de mal diseño si una clase límite sirve actores de diferentes tipos.

Puesto que todavía se encuentra en análisis, es importante mantener estas clases en el nivel correcto de abstracción. Por ejemplo, cuando se modela una clase <<boundary>> que representa una GUI, simplemente modele la ventana de nivel superior y deje todos los detalles que componen la ventana al diseño. De forma alternativa, puede introducir una clase tonta que representa toda la interfaz de usuario.

De forma similar, con clases de interfaz de sistema y clases de interfaz de dispositivo, está preocupado con capturar el hecho de que existe una clase que media entre su sistema y alguna otra cosa, pero no con los detalles específicos de esa clase. Decidirá estos detalles más adelante en el diseño.

Por ejemplo, si escribe un sistema de comercio electrónico que necesita actuar de interfaz con un sistema de inventario, puede representar la interfaz al sistema de inventario por medio de una clase denominada *InterfazInventario* que está estereotipada como <<boundary>>. Esto es suficiente detalle para un modelo de análisis.

8.4.3.2. Encontrar clases <<control>>

Estas clases son controladores, sus instancias coordinan comportamiento del sistema que corresponde a uno o más casos de uso. Encuentra clases de control al considerar el comportamiento del sistema como se describe por los casos de uso y averiguando cómo ese comportamiento se debería dividir entre las clases de análisis. El comportamiento sencillo se puede distribuir entre clases límite o entidad,

pero el comportamiento más complejo, como el procesamiento de pedidos, está mejor localizado al introducir una clase controlador adecuada como GestorPedido. Algunos moderadores (nosotros incluidos) a menudo indican las clases de control al anexar Gestor o Controlador al nombre de la clase.

El punto clave cuando se trabaja con clases de control es dejar que las clases surjan de forma natural del propio dominio del problema. Algunos moderadores introducen de forma artificial una clase de control para cada caso de uso para controlar o ejecutar ese caso de uso. Ésta es una aproximación peligrosa que lleva a un modelo que se parece más a una descomposición funcional de arriba a abajo que a un verdadero modelo de análisis orientado a objetos. De hecho, ésta es una de las razones que consideramos utilizando los estereotipos RUP como opcionales. Esto puede hacer que los modeladores principiantes vayan por mal camino.

En el mundo real, los controladores que surgen directamente del dominio del problema (en lugar de como una consecuencia de una técnica de análisis específica) tienden a acortar varios casos de uso. Un buen ejemplo podría ser un controlador como una clase Secretario que está implicada en muchos de los casos de uso que describen un sistema de registro de cursos. De forma similar, un solo caso de uso puede requerir la participación de muchas clases de control.

Si encuentra que una clase controlador tiene un comportamiento muy complicado, esto indica que puede desglosarlo en dos o más controladores más sencillos que implementa cada uno un subconjunto cohesivo de ese comportamiento. Cada una de las clases más sencillas que identifique puede ser algo que ocurre de forma natural en el dominio del problema. Por ejemplo, cuando diseña un sistema de registro de cursos, podría introducir una clase de control denominada ControladorRegistroCurso que coordina todo el proceso. Pero dicha clase tiene un comportamiento complejo, por lo que podría decidir descomponerla en un conjunto de clases de colaboración gestionando cada una de ellas uno o dos aspectos de ese comportamiento. ControladorRegistroCurso se podría descomponer en las clases Secretario, GestorCurso y GestorPersonal. Observe que cada una de estas clases representa algo que existe en el dominio del problema.

Una buena forma de explorar clases controlador es imaginarse en el rol de la clase. ¿Qué haría en esa situación?

8.4.3.3. Encontrar clases <<entity>>

Estas clases modelan información sobre algo y normalmente tienen un comportamiento muy sencillo que equivale a poco más que a obtener y establecer valores. Las clases que representan información persistente como las direcciones (una clase Dirección) y personas (una clase Persona) son clases de entidad.

Las clases de entidad:

- Acortan muchos casos de uso.
- Están manipuladas por clases de control.
- Proporcionan información y aceptan información de clases límite.

- Representan elementos clave gestionados por el sistema (por ejemplo, Cliente).
- A menudo son persistentes.

Las clases de entidad expresan la estructura lógica de datos del sistema. Si tiene un modelo de datos, entonces las clases de entidad están íntimamente relacionadas con entidades o tablas en este modelo.

8.4.4. Encontrar clases de otras fuentes

Junto con el análisis nombre/verbo, análisis CRC y estereotipos RUP, merece la pena recordar que existen muchas otras fuentes potenciales de clases que se deberían considerar. Igual que busca abstracciones concisas que mapeen con elementos del mundo real en el dominio del problema, obviamente puede buscar clases en el mundo real.

- Los objetos físicos como avión, gente y hoteles pueden indicar clases.
- El papeleo es otra fuente estupenda de clases. Los recibos y pedidos, pueden indicar clases posibles. Sin embargo, debe tener mucho cuidado cuando examina todos estos papeles. En muchas empresas éste ha evolucionado con los años para soportar exactamente los procesos redundantes de negocio que el nuevo sistema podría tratar de reemplazar. Lo último que quiere hacer como analista/diseñador orientado a objetos es automatizar sistemas obsoletos basados en papel.
- Las interfaces conocidas con el mundo exterior, como pantallas, teclados, periféricos y otros sistemas, también pueden ser una fuente de clases candidatas, especialmente para sistemas incorporados.
- Entidades conceptuales son elementos que son cruciales para la operación del negocio, pero no se manifiestan como elementos concretos. Como ejemplo de esto, podría ser un ProgramaFidelidad. Claramente, el propio programa no es un elemento concreto, pero sigue siendo una abstracción cohesiva y por lo tanto se puede justificar su modelado como una clase.

8.4.4.1. Patrones arquetipo

En nuestro libro *Enterprise Patterns and MDA* [Arlow 1] describimos un conjunto de lo que denominados patrones arquetipo. Éstos son patrones de conceptos de negocio que son tan penetrantes en sistemas de negocio que creemos que son arquetipos por naturaleza.

Como tales, se pueden modelar una vez y luego reutilizar, en lugar de modelarse una y otra vez en cada sistema de negocio. La idea del libro es que puede utilizar estos patrones como tales o modificarlos para construir su modelo de análisis a partir de componentes de modelo. Denominamos a esta técnica modelado basado en componentes.

Proporcionamos los siguientes patrones de arquetipo:

- CRM
- Inventario
- Dinero
- Pedido
- Tercero
- Relación con terceros
- Producto
- Cantidad
- Regla

Cada uno de estos patrones está muy detallado. Si puede reutilizar uno de estos patrones puede ahorrarse muchos hombres/día incluso hombres/meses de trabajo. Incluso si el patrón no es del todo apropiado para lo que está tratando de modelar, puede proporcionarle ideas de utilidad para sus propias clases de análisis en lugar de partir de una página en blanco. Ésta es probablemente la forma más eficaz de encontrar clases para sus modelos.

8.5. Crear un modelo de análisis de primer nivel

Para crear un modelo de análisis de primer nivel necesita consolidar las salidas del análisis nombre/verbo, análisis CRC, estereotipos RUP y una consideración de otras fuentes de clases (especialmente patrones de arquetipo) en un solo modelo UML en una herramienta de modelado. Realice la consolidación de la siguiente manera.

1. Compare todas las fuentes de clases.
2. Consolide las clases de análisis, atributos y responsabilidades de las diferentes fuentes e incorpórelas en una herramienta de modelado.
 1. Utilice el glosario del proyecto para resolver sinónimos y homónimos.
 2. Busque diferencias en los resultados de las tres técnicas. Las diferencias indican áreas donde se podría hacer más trabajo. Resuelva estas diferencias ahora o destáquelas para más adelante.
3. Los colaboradores (o líneas entre los post-it en la pizarra) representan relaciones entre clases. Verá cómo modelar éstas en el capítulo 9.
4. Mejore el nombrado de clases, atributos y responsabilidades para seguir cualquier convención estándar de nombres que su empresa tenga, o siga las convenciones de nombre descritas en el capítulo 7.

El resultado de esta actividad es un conjunto de clases de análisis donde cada clase puede tener algunos atributos clave y debería tener entre tres a cinco responsabilidades. Éste es su primer modelo de análisis de primer nivel.

8.6. ¿Qué hemos aprendido?

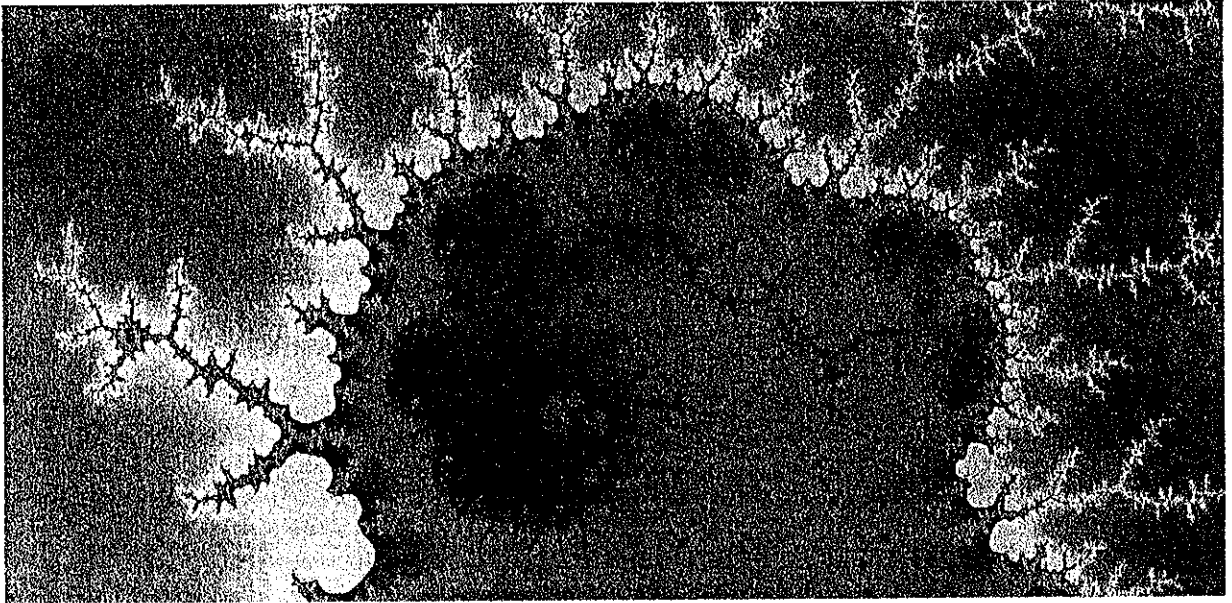
En este capítulo hemos descrito qué son las clases de análisis y cómo encontrar estas clases utilizando las técnicas de análisis nombre/verbo, tormenta de ideas CRC y un examen de otras fuentes de clases.

Ha aprendido lo siguiente:

- La actividad UP Analizar un caso de uso muestra como resultado clases de análisis y realizaciones de caso de uso.
- Las clases de análisis representan una abstracción bien definida en el dominio del problema.
 - El dominio del problema es el dominio en el que ha surgido la necesidad del sistema de software.
 - Las clases de análisis deberían mapear de forma clara con un concepto de negocio del mundo real.
 - Los conceptos de negocio a veces se tienen que aclarar durante el análisis.
- El modelo de análisis solamente contiene clases de análisis; cualquier clase que surja de las consideraciones de diseño (dominio de la solución) se debe excluir.
- Las clases de análisis incluyen:
 - Un conjunto de atributos candidatos de alto nivel.
 - Un conjunto de operaciones de alto nivel.
- ¿Qué hace que sea una buena clase de análisis?
 - Su nombre refleja su propósito.
 - Es una abstracción clara que modela un elemento específico del dominio del problema.
 - Mapea en una característica claramente identificable del dominio del problema.
 - Tiene un pequeño conjunto de responsabilidades bien definidas.
 - Una responsabilidad es un contrato u obligación que una clase tiene con sus clientes.
 - Una responsabilidad es un conjunto de operaciones semánticamente cohesivas.
 - Solamente debería haber de tres a cinco responsabilidades por clase.

- Tiene una cohesión alta; todas las características de la clase deberían ayudar a realizar su finalidad.
- Tiene bajo acoplamiento. Una clase solamente debería colaborar con un pequeño número de otras clases para realizar su propósito.
- ¿Qué hace que sea una mala clase de análisis?
 - Es un functoid; un clase con sólo una operación.
 - Es una clase omnipotente, una clase que lo hace todo. Las clases con "sistema" o "controlador" en su nombre pueden necesitar un mayor examen.
 - Tiene un árbol de herencia profundo; en el mundo real los árboles de herencia tienden a ser poco profundos.
 - Tiene una baja cohesión.
 - Tiene un alto acoplamiento.
- Análisis nombre/verbo.
 - Busca nombres o frases nominales, estos son clases o atributos candidatos.
 - Busca verbos o frases verbales, estos son responsabilidades u operaciones candidatas.
 - El procedimiento es recopilar información relevante y luego analizarla.
- El análisis CRC es una técnica potente y divertida.
 - Lo importante en el dominio del problema se escriben en post-it.
 - Cada nota tiene tres compartimentos:
 - Clase. Contiene el nombre de la clase.
 - Responsabilidades. Contiene una lista de las responsabilidades de esa clase.
 - Colaboradores. Contiene una lista de otras clases con la que esta clase colabora.
 - Procedimiento, tormenta de ideas:
 - Pida a los miembros del equipo que nombren lo que funciona en su dominio de negocio y escríbalo en post-it.
 - Pida al equipo que establezca las responsabilidades y las grabe en el compartimiento de responsabilidades de la nota.
 - Pida al equipo que identifique clases que podrían trabajar conjuntamente y dibuje líneas entre ellos o grábelo en el compartimiento de colaboradores de cada nota.

- Los estereotipos RUP se pueden utilizar para centrar la actividad de análisis en tres tipos de clase:
 - `<<boundary>>`. Una clase que media la interacción entre el sistema y su entorno.
 - `<<control>>`. Una clase que encapsula el comportamiento específico del caso de uso.
 - `<<entity>>`. Una clase que se utiliza para modelar información persistente sobre algo.
- Considere otras fuentes de clase:
 - Objetos físicos, interfaces con el mundo exterior y entidades conceptuales.
 - Patrones de arquetipo; modelado basado en componentes.
- Crear un modelo de análisis de primer nivel:
 - Compare los resultados del análisis nombre/verbo con resultados CRC y los resultados de un examen de otras fuentes de clases.
 - Resuelva sinónimos y homónimos.
 - Las diferencias entre los resultados de las diferentes técnicas indican áreas de incertidumbre.
 - Consolide resultados en un modelo de análisis de primer nivel.



9

Relaciones