Los Patrones como un Medio del Diseño Orientado a Objetos

*Ricardo Armando Machorro Reyes**

Introducción

Un sistema orientado a objetos se compone de objetos que envían mensajes a otros objetos para que lleven a cabo las operaciones. En los contratos se incluye una conjetura inicial óptima sobre las responsabilidades y las poscondiciones de las operaciones. Los diagramas de interacción describen gráficamente la solución a partir de los objetos en interacción que estas responsabilidades y poscondiciones satisfacen.

La calidad de diseño de la interacción de los objetos y la asignación de responsabilidades presentan gran variación. Las decisiones poco acertadas dan origen a sistemas y componentes frágiles y difíciles de mantener, entender, reutilizar o extender. Una implementación hábil se funda en los principios cardinales que rigen un buen diseño orientado a objetos. En los patrones GRASP se codifican algunos de ellos, que se aplican al preparar los diagramas de interacción, cuando se asignan las responsabilidades o durante ambas actividades.

GRASP es el acrónimo de General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades). El nombre se eligió para indicar la importancia de *captar* estos principios, si se quiere diseñar eficazmente el software orientado a objetos.

*Maestro en Ciencias con especialidad en Informática por la UPIICSA. Profesor de las Academias de Computación de la UPIICSA. Consultor Especialista de sistemas en el área de Ingeniería de Software. Correo electrónico: rickmr@hotmail.com.

Actividades y dependencias

Los patrones a los que nos referimos se aplican durante la elaboración de los diagramas de interacción, al asignar las responsabilidades a los objetos y al diseñar la colaboración entre ellos.

Los diagramas de interacción bien diseñados son muy útiles

Diagramas de colaboración:

- Los diagramas de interacción son algunos de los artefactos más importantes que se preparan en el análisis y diseño orientados a objetos.
- Es muy importante asignar acertadamente las responsabilidades al momento de elaborar los diagramas de interacción.
- El tiempo y el esfuerzo que se dedican a su elaboración, así como un examen riguroso de la asignación de responsabilidades, deberían absorber parte considerable de la fase de diseño de un proyecto.
- Los patrones, principios y expresiones especializadas codificados sirven para mejorar la calidad del diseño.

Los principios del diseño que se requieren para construir buenos diagramas de interacción pueden codificarse, explicarse y utilizarse en forma metódica. Esta manera de entender y usar los principios del diseño se funda en los *patrones con que se asignan las responsabilidades.*

Responsabilidades y métodos

Booch y Rumbaugh definen la responsabilidad como "un contrato u obligación de un tipo o clase". Las responsabilidades se relacionan con las obligaciones de un objeto respecto a su comportamiento. Esas responsabilidades pertenecen, esencialmente, a las dos categorías siguientes:

1. Conocer
2. Hacer

Entre las responsabilidades de un objeto relacionadas con hacer se encuentran:

- Hacer algo en uno mismo.
- Iniciar una acción en otros objetos.
- Controlar y coordinar actividades en otros objetos.

Entre las responsabilidades de un objeto relacionadas con conocer se encuentran:

- Estar enterado de los datos privados encapsulados.
- Estar enterado de la existencia de objetos conexos.
- Estar enterado de cosas que se puede derivar o calcular.

Las responsabilidades se asignan a los objetos durante el diseño orientado a objetos. Por ejemplo, puede declararse que "una *Venta* es responsable de imprimirse ella misma" (un hacer) o que "una *Venta* tiene la obligación de conocer su fecha" (un conocer). Las responsabilidades relacionadas con "conocer" a menudo pueden inferirse del modelo conceptual por los atributos y asociaciones explicadas en él.

La granularidad de la responsabilidad influye en su traducción a clases y métodos. La responsabilidad de "brindar acceso a las bases relacionales de datos" puede incluir docenas de clases y cientos de métodos. En cambio, la de "imprimir una venta" tal vez no

incluya más que un método o unos cuantos.

Responsabilidad no es lo mismo que método: los métodos se ponen en práctica para cumplir con las responsabilidades. Éstas se implementan usando métodos que operen solos o en colaboración con otros métodos y objetos. Así, la clase *Venta* podría definir uno o varios métodos que imprimen una instancia *Venta;* digamos el método *imprimir*. Para cumplir con esa responsabilidad, la *Venta* puede colaborar con otros objetos, entre ellos el envío de un mensaje a los objetos *VentasLineadeProducto,* pidiéndoles que se impriman ellos mismos.

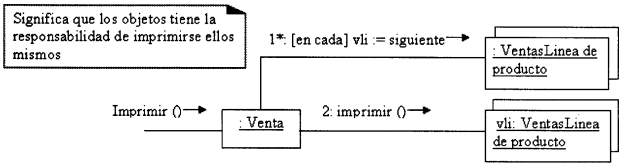

Figura 1. Las responsabilidades y los métodos.

Las responsabilidades y los diagramas de interacción

En los artefactos del UML, las responsabilidades (implementadas como métodos) suelen tenerse en cuenta al momento de preparar los diagramas de interacción.

La figura 1 indica que a los objetos *Venta* se les ha asignado la responsabilidad de imprimirse ellos mismos, la cual se llama con un mensaje *imprimir* y se cumple con el método correspondiente *imprimir*. Más aún, para atender esta responsabilidad hay que colaborar con los objetos *VentasLineadeProducto,* pidiéndoles que impriman.

En resumen, los diagramas de interacción muestran las decisiones referentes a la asignación de responsabilidades entre los objetos. Cuando se preparan, se toman decisiones sobre la asignación que se reflejan en los mensajes que son enviados a varias clases de objetos. En el presente artículo expondremos ampliamente los principios fundamentales -expresados en los patrones GRASP- para guiar las decisiones sobre la asignación de responsabilidades. Esas decisiones se reflejarán después en los diagramas de interacción.

Patrones

Los diseñadores expertos en orientación a objetos (y también otros diseñadores de software) van formando un amplio repertorio de principios generales y de expresiones que los guían al crear software. A unos y a otras podemos asignarles el nombre de patrones, si se codifican en un formato estructurado que describe el problema y su solución, y si se les asigna un nombre. A continuación ofrecemos un ejemplo de patrón.

Nombre del patrón: Experto.

Solución: Asignar una responsabilidad a la clase que tiene la información necesaria para cumplirla.

Problema que resuelve: ¿Cuál es el principio fundamental en virtud del cual asignaremos las responsabilidades a los objetos?

En la terminología de objetos, el patrón es una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos; en teoría, indica la manera de utilizarlo en circunstancias diversas. Muchos patrones ofrecen orientación sobre cómo asignar las responsabilidades a los objetos ante determinada categoría de problemas.

Expresado lo anterior con palabras más simples, el patrón es una pareja de problema y solución con un nombre, y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas.

"El patrón de un individuo puede ser la estructura primitiva de otra persona", máxima con que en la tecnología de objetos se explica la vaguedad de los patrones. En nuestro estudio de los patrones no abordaremos el tema de lo que conviene llamar patrón; nos centraremos en el valor pragmático de utilizar ese estilo como medio de presentar y recordar los principios tan útiles de la ingeniería del software.

Los patrones no suelen contener ideas nuevas

Los patrones no se proponen descubrir ni expresar nuevos principios de la ingeniería del software. Todo lo contrario: intentan codificar el conocimiento, las expresiones y los principios *ya existentes:* cuanto más trillados y generalizados, tanto mejor. En consecuencia, los patrones GRASP no introducen ideas novedosas; son una mera codificación de los principios básicos más usados.

Los patrones tienen nombre

En teoría, todos los patrones poseen nombres muy sugestivos. El asignar nombre a un patrón, a un método o a un principio ofrece las siguientes ventajas:

- Apoya el agrupamiento y la incorporación del concepto a nuestro sistema cognitivo y a la memoria.
- Facilita la comunicación.

Darle nombre a una idea compleja -digamos a un patrón- es un ejemplo de la fuerza de la abstracción: convierte una forma compleja en una forma simple con sólo eliminar los detalles. Por tanto, los patrones GRASP poseen nombres concisos como *Experto, Creador, Controlador*.

La asignación de nombre a los patrones mejora la comunicación

Cuando se le da nombre a un patrón, un simple nombre nos permite discutir con otros un

principio en todos sus aspectos. Considere la siguiente conversación entre dos diseñadores de software que emplean una nomenclatura común de patrones *(Experto, Separacion Vista-Modelo* y otros términos afines) para tomar una decisión sobre un diseño:

Alfredo: "En tu opinión, ¿a qué objeto deberíamos asignar la responsabilidad de imprimir una *Venta?* Creo que convendría asignarla a *Separacion Vista-Modelo.* ¿Qué te parece una *VistadeReportedeVentas?"*

Teresa: "En mi opinión, *Experto* es una mejor solución por ser una impresión simple y porque Ventas tiene toda la información que se requiere en salida impresa. Creo que le asignaremos esta responsabilidad a la *Venta.*

Alfredo: "Está bien, hagámoslo."

Agrupar las expresiones especializadas y los principios del diseño con nombres del uso común facilitan la comunicación y le confiere a la búsqueda un nivel de abstracción más alto.

GRASP: patrones de los principios generales para asignar responsabilidades

Resumimos a continuación la introducción anterior:

- Asignar correctamente las responsabilidades es muy importante en el diseño orientado a objetos.
- La asignación de responsabilidades a menudo se realiza en el momento de preparar los diagramas de interacción.
- Los patrones son parejas de problema/solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades.

Dicho esto, podemos proceder a examinar los patrones GRASP.

Pregunta: ¿Qué son los patrones GRASP?

Respuesta: Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Es importante entender y poder aplicar estos principios durante la preparación de un diagrama de interacción, pues un diseñador de software sin mucha experiencia en la tecnología de objetos debe dominarlos cuanto antes: constituyen el fundamento de cómo se diseñará el sistema.

La notación del UML para los diagramas de clase

Los diagramas de clase en UML presentan las clases de software en contraste con los conceptos de dominio. La casilla de clase consta de tres secciones; la tercera contiene los métodos de la clase, como se aprecia en la figura 2.

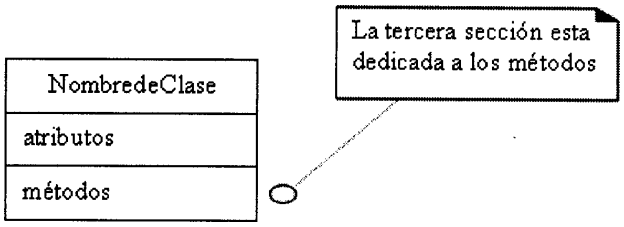

Figura 2. Las clases de software muestran los nombres de los métodos.

Experto

Solución: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Problema: ¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos?

Un modelo de clase puede definir docenas y hasta cientos de clases de software, y una aplicación tal vez requiera el cumplimiento de cientos o miles de responsabilidades. Durante el diseño orientado a objetos, cuando se definen las interacciones entre los objetos, tomamos decisiones sobre la asignación de responsabilidades a las clases. Si se hacen en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se nos presenta la oportunidad de reutilizar los componentes en futuras aplicaciones.

Ejemplo. En la aplicación del punto de venta, alguna clase necesita conocer el gran total de la venta.

Comience asignando las responsabilidades con una definición clara de ellas.

A partir de esta recomendación se plantea la pregunta:

¿Quién es el responsable de conocer el gran total de la venta?

Desde el punto de vista del patrón Experto, deberíamos buscar la clase de objetos que posee la información necesaria para calcular el total. Examinemos detenidamente el modelo conceptual parcial de la figura 3.

¿Qué información hace falta para calcular el gran total? Hay que conocer todas las instancias *VentasLineadeProducto* de una venta y la suma de sus subtotales. Y esto lo conoce únicamente la instancia

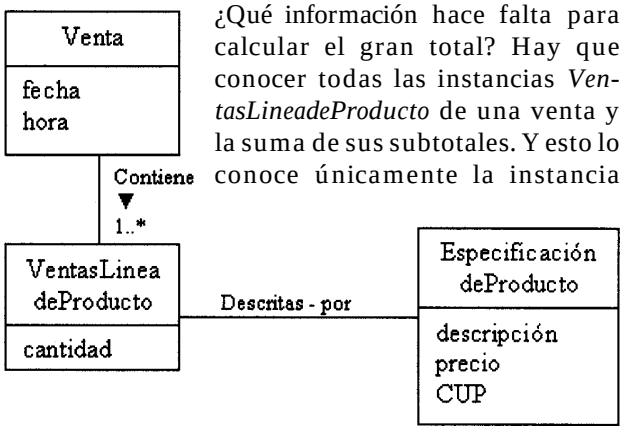

Figura 3. Asociaciones de la Venta.

Venta; por tanto, desde el punto de vista del Experto, *Venta* es la clase correcta de objeto para asumir esta responsabilidad; es el *experto en información.*

Como se señaló con anterioridad, es dentro del contexto de la preparación de los díagramas de interacción (por ejemplo, los de colaboración) donde surgen estas cuestiones concernientes a la responsabilidad. Imagine que vamos a empezar a dibujar diagramas para asignar responsabilidades a los objetos. El diagrama parcial de colaboración y el de clases de la figura 4 describen gráficamente las decisiones que hemos adoptado hasta ahora.

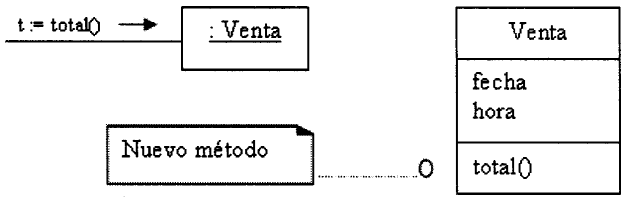

Figura 4. Diagrama parcial de colaboración.

Todavía no terminamos. ¿Qué información hace falta para determinar el subtotal de la línea de productos? Se necesitan *VentasLineadeProducto.cantidad* y *EspecificaciondeProducto.precio. VentasLineadeProducto* conoce su cantidad y su correspondiente *EspecificaciondeProducto;* por tanto, desde la perspectiva de patrón Experto, *VentasLineadeProducto* debería calcular el subtota1; es el *experto* en *información.*

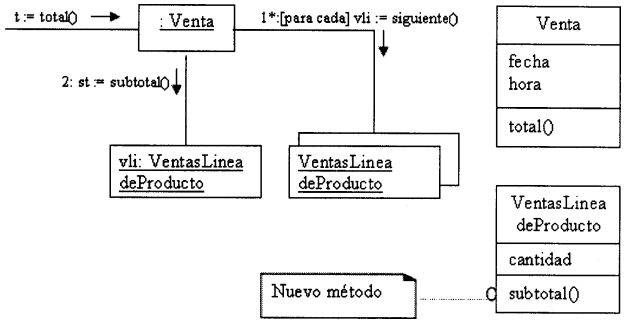

Figura 5. Cálculo del total de la venta.

VentasLineadeProducto no puede cumplir la responsabilidad de conocer y dar el subtotal, si no conoce el precio del producto. *EspecificaciondeProducto* es un Experto en información para contestar su precio; por tanto, habrá que enviarle un mensaje preguntándole el precio. El diseño correspondiente aparece en la figura 6.

En conclusión, para cumplir con la responsabilidad de conocer y dar el total de la venta, se asignaron tres responsabilidades a las tres clases de objeto así:

Clase	*Responsabilidad*
Venta	Conoce el total de la venta
VentasLineadeProducto	Conoce el subtotal de la línea de producto
EspecificaciondeProducto	Conoce el precio del producto

El donde las responsabilidades se consideraron y se decidieron fue el dibujo de un diagrama de colaboración. Después, la sección dedicada a métodos en un diagrama de clases puede resumir los métodos.

El principio en virtud del cual se asignaron las responsabilidades fue el patrón Experto: lo colocamos junto con el objeto que posee la información necesaria para cumplirlas.

Explicación. Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen.

Nótese que el cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos. Ello significa que hay muchos expertos "parciales" que colaboraron en la tarea. Por ejemplo, el problema del total de la venta exigirá finalmente la colaboración de tres clases. Siempre que la información se encuentre esparcida en varios objetos, éstos habrán de interactuar a través de mensajes para compartir el trabajo.

El patrón Experto da origen a diseño donde el objeto de software realiza las operaciones que normalmente se aplican a la cosa real que representa; a esto, Peter Coad lo llama estrategia de "Hacerlo yo mismo". Por ejemplo, en el mundo real una venta no nos indica su total sin ayuda de aparatos electromecánicos; se trata de un concepto inanimado. Alguien calcula el total de la venta. Pero en el terreno del software orientado a objetos, todos los objetos del software están "vivos" o "animados", y pueden asumir responsabilidades y hacer cosas. Básicamente, hacen cosas relacionadas con la información de que disponen. A esto yo lo llamo principio de "Animación" en el diseño orientado a objetos; es como estar en una caricatura donde todo tiene vida.

El patrón Experto -como tantas otras cosas en la tecnología de objetos- ofrece una analogía con el mundo real. Acostumbramos asignar responsabilidad a individuos que disponen de la información necesaria para llevar a cabo una tarea. Por ejemplo, en una empresa ¿quién será el encargado de preparar un estado de pérdidas y ganancias? El empleado que tiene acceso a toda la información necesaria para elaborarlo: quizá el director financiero. Y del mismo modo que los objetos del software colaboran porque la información está esparcida, lo mismo sucede con el personal de una empresa. El director financiero podrá pedir a los encargados de cuentas por cobrar y de cuentas por pagar que

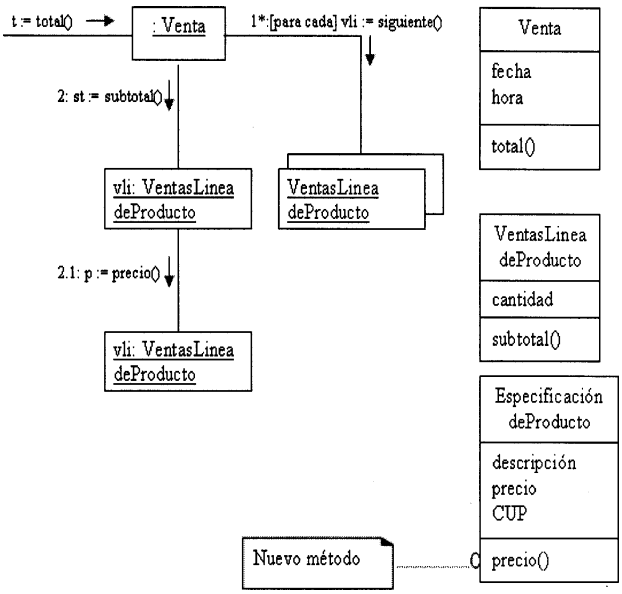

Figura 6. Cálculo del total de Venta.

generen reportes individuales sobre créditos y deudas.

Beneficios

- Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento.

- El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase "sencillas" y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

Otras formas de designar este patrón son "Juntar responsabilidades y la información", "Lo hace el que conoce", "Animación", "Lo hago yo mismo", "Unir los servicios a los atributos sobre los que operan".

Creador

Solución. Asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:

- B agrega los objetos A.
- B contiene los objetos A.
- B registra las instancias de los objetos A.
- B utiliza específicamente los objetos A.
- B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A).

B es un *creador* de los objetos A.

Si existe más de una opción, prefiera la clase B que *agregue* o *contenga* la clase A.

Problema. ¿Quién debería ser responsable de crear una nueva instancia de alguna clase?

La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella. El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilizabilidad.

Ejemplo. En la aplicación del punto de venta, ¿quién debería encargarse de crear una instancia *VentasLineadeProducto?* Desde el punto de vista del patrón Creador, deberíamos buscar una clase que agregue, contenga y realice otras operaciones sobre este tipo de instancias. Examine atentamente el modelo conceptual parcial de la figura 7.

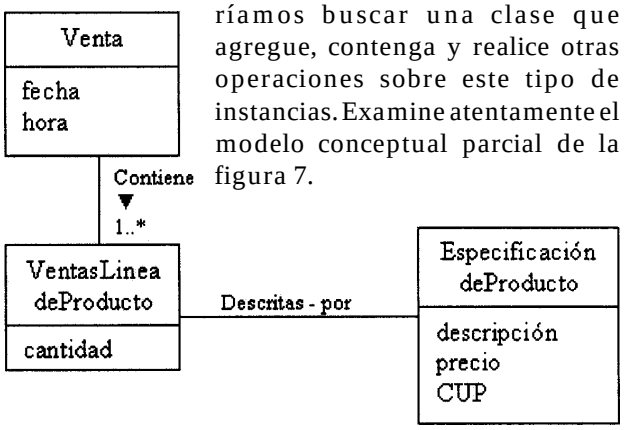

Figura 7. Modelo conceptual parcial.

Una *Venta* contiene (en realidad, agrega) muchos objetos *VentasLineadeProducto;* por ello, el patrón Creador sugiere que *Venta* es idónea para asumir la responsabilidad de crear las instancias VentasLineadeProducto.

Lo anterior nos permite diseñar las interacciones de los objetos en un diagrama de colaboración, según se observa en la figura 8.

Esta asignación de responsabilidades requiere definir en *Venta* un método de *hacer.LineadeProducto.*

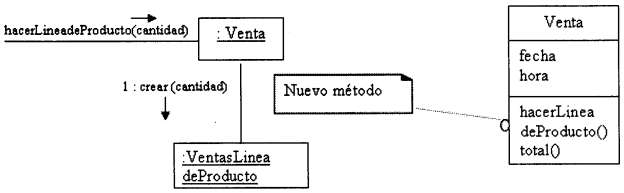

Figura 8 Creación de un objeto VentaslineadeProducto.

Una vez más, el contexto donde se consideraron y se asignaron estas responsabilidades fue el momento de dibujar un diagrama de colaboración. Después, en la sección destinada a métodos en un diagrama de clases, podemos resumir los resultados de la asignación de responsabilidades, realizadas concretamente como métodos.

Explicación

El patrón Creador guía la asignación de responsabilidades

relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.

El Agregado *agrega* la Parte, el Contenedor *contiene* el contenido, el Registro *registra*. En un diagrama de clases se registran las relaciones muy frecuentes entre las clases. El patrón Creador indica que la clase incluyente del contenedor o registro es idónea para asumir la responsabilidad de crear la cosa contenida o registrada. Desde luego, se trata tan sólo de una directriz.

Nótese que el concepto de **agregación** se utilizó al examinar el patrón Creador. Es un tema que trataremos más ampliamente en un artículo posterior; pero damos aquí una definición sucinta: la agregación incluye cosas que están en una sólida relación de parte-todo o de parte-estructura; por ejemplo, Cuerpo agrega Pierna y Párrafo agrega oración.

En ocasiones encontramos un patrón creador buscando la clase con los datos de inicialización que serán transferidos durante la creación. Éste es en realidad un ejemplo del patrón Experto. Los datos de inicialización se transmiten durante la creación a través de algún método de inicialización, como un constructor en Java que cuenta con parámetros.

Suponga, por ejemplo, que una instancia *Pago* al momento de ser creada necesita inicializarse con el total *Venta*. Puesto que *Venta* conoce el total, es un buen candidato para ser el parámetro creador de *Pago*.

Beneficios

Se brinda soporte a un bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. Es probable que el acoplamiento no aumente, pues la clase *creada* tiende a ser visible a la clase *creador*, debido a las asociaciones actuales que nos llevaron a elegirla como el parámetro adecuado.

Finalmente, repetimos algunos de los puntos ya expuestos en este artículo cuando hablamos de los diagramas de colaboración:

- Los diagramas de interacción son algunos de los artefactos más importantes que se preparan en el análisis y diseño orientados a objetos.
- Es muy importante asignar acertadamente las responsabilidades al momento de elaborar los diagramas de interacción.
- El tiempo y el esfuerzo que se dedican a su elaboración, así como un examen riguroso de la asignación de responsabilidades, deberían absorber parte considerable de la fase de diseño de un proyecto.
- Los patrones, principios y expresiones especializadas codificados sirven para mejorar la calidad del diseño.

Los principios del diseño que se requieren para construir buenos diagramas de interacción pueden codificarse, explicarse y utilizarse en forma metódica. Esta manera de entender y usar los principios del diseño se funda en los *patrones con que se asignan las responsabilidades.*

Bibliografía

Booch, G., Jacobson, I. y Rumbaugh, J. *UML Manual de referencia.* Addison Wesley. 1997.

Booch, G. *Object-Oriented analysis and Design.* Benjamin / Cummings. 1994.

Booch, G. *Object Solutions: Managing the Object-Oriented Project.* Addison-Wesley. 1996.

Coad, P. Object Models: *Strategies Patterns and Applications.* Prentice-Hall. 1995.

Fowler, M. *Analysis Patterns: Reusable Object Models.* Addison-Wesley. 1996.

Jacobson, I., et al. *Object-Oriented Software Engineering: A Use Case Driven Approach.* Addison-Wesley. 1992.

Martin,J. y Odell,J. *Object-Oriented Methods: A Foundation.* Prentice-Ha11. 1995.

Rumbaugh,J. et al. *Object-Oriented Modelling and Design.* Prentice-Hall. 1991.