HORA 1

Introducción al UML

El UML (Lenguaje Unificado de Modelado) es una de las herramientas más emocionantes en el mundo actual del desarrollo de sistemas. Esto se debe a que permite a los creadores de sistemas generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas.

En esta hora se tratarán los siguientes temas:

- Por qué es necesario el UML
- La concepción del UML
- Diagramas del UML
- Para qué tantos diagramas

TÉRMINO NUEVO En el contexto de este libro considere a un *sistema* como una combinación de software y hardware que da una solución a un problema de negocios. El *desarrollo de sistemas* es la creación de un programa para un *cliente*, este último es quien tiene el problema que debe ser resuelto. Un *analista* es el que documenta el problema del cliente y lo comunica a los *desarrolladores*, que son los programadores que generarán el programa que resolverá el problema y lo distribuirán en equipos de computación.

La comunicación de la idea es de suma importancia. Antes del advenimiento del UML, el desarrollo de sistemas era, con frecuencia, una propuesta al azar. Los analistas de sistemas intentaban evaluar los requerimientos de sus clientes, generar un análisis de requerimientos en algún tipo de notación que ellos mismos comprendieran (aunque el cliente no lo comprendiera), dar tal análisis a uno o varios programadores y esperar que el producto final cumpliese con lo que el cliente deseaba.

Dado que el desarrollo de sistemas es una actividad humana, hay muchas posibilidades de cometer errores en cualquier etapa del proceso, por ejemplo, el analista pudo haber malentendido al cliente, es decir, probablemente produjo un documento que el cliente no pudo comprender. Tal vez ese documento tampoco fue comprendido por los programadores quienes, por ende, pudieron generar un programa difícil de utilizar y no generar una solución al problema original del cliente.

¿Alguien se preguntará por qué muchos de los sistemas en uso son ineficientes, engorrosos y difíciles de utilizar?

Por qué es necesario el UML

En los principios de la computación, los programadores no realizaban análisis muy profundos sobre el problema por resolver. Si acaso, garabateaban algo en una servilleta. Con frecuencia comenzaban a escribir el programa desde el principio, y el código necesario se escribía conforme se requería. Aunque anteriormente esto agregaba un aura de aventura y atrevimiento al proceso, en la actualidad es inapropiado en los negocios de alto riesgo.

Hoy en día, es necesario contar con un plan bien analizado. Un cliente tiene que comprender qué es lo que hará un equipo de desarrolladores; además tiene que ser capaz de señalar cambios si no se han captado claramente sus necesidades (o si cambia de opinión durante el proceso). A su vez, el desarrollo es un esfuerzo orientado a equipos, por lo que cada uno de sus miembros tiene que saber qué lugar toma su trabajo en la solución final (así como saber cuál es la solución en general).

Conforme aumenta la complejidad del mundo, los sistemas informáticos también deberán crecer en complejidad. En ellos se encuentran diversas piezas de hardware y software que se comunican a grandes distancias mediante una red, misma que está vinculada a bases de datos que, a su vez, contienen enormes cantidades de información. Si desea crear sistemas que lo involucren con este nuevo milenio ¿cómo manejará tanta complejidad?

La clave está en organizar el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan y convengan con él. El UML proporciona tal organización.

Un arquitecto no podría crear una compleja estructura como lo es un edificio de oficinas sin crear primero un anteproyecto detallado; asimismo usted tampoco podría generar un complejo sistema en un edificio de oficinas sin crear un plan de diseño detallado. La idea

es que así como un arquitecto le muestra un anteproyecto a la persona que lo contrató, usted deberá mostrarle su plan de diseño al cliente. Tal plan de diseño debe ser el resultado de un cuidadoso análisis de las necesidades del cliente.

Otra característica del desarrollo de sistemas contemporáneo es reducir el periodo de desarrollo. Cuando los plazos se encuentran muy cerca uno del otro es absolutamente necesario contar con un diseño sólido.

Hay otro aspecto de la vida moderna que demanda un diseño sólido: las adquisiciones corporativas. Cuando una empresa adquiere a otra, la nueva organización debe tener la posibilidad de modificar aspectos importantes de un proyecto de desarrollo que esté en progreso (la herramienta de desarrollo, el lenguaje de codificación, y otras cosas). Un anteproyecto bien diseñado facilitará la conversión. Si el diseño es sólido, un cambio en la implementación procederá sin problemas.

La necesidad de diseños sólidos ha traído consigo la creación de una notación de diseño que los analistas, desarrolladores y clientes acepten como pauta (tal como la notación en los diagramas esquemáticos sirve como pauta para los trabajadores especializados en electrónica). El UML es esa misma notación.

La concepción del UML

El UML es la creación de Grady Booch, James Rumbaugh e Ivar Jacobson. Estos caballeros, apodados recientemente "Los tres amigos", trabajaban en empresas distintas durante la década de los años ochenta y principios de los noventa y cada uno diseñó su propia metodología para el análisis y diseño orientado a objetos. Sus metodologías predominaron sobre las de sus competidores. A mediados de los años noventa empezaron a intercambiar ideas entre sí y decidieron desarrollar su trabajo en conjunto.

Las horas 2, "Orientación a objetos", y 4, "Uso de relaciones", tratan de la orientación a objetos. Los conceptos de orientación a objetos tienen un papel fundamental en el desarrollo de este libro.

En 1994 Rumbaugh ingresó a Rational Software Corporation, donde ya trabajaba Booch. Jacobson ingresó a Rational un año después; el resto, como dicen, es historia.

Los anteproyectos del UML empezaron a circular en la industria del software y las reacciones resultantes trajeron consigo considerables modificaciones. Conforme diversos corporativos vieron que el UML era útil a sus propósitos, se conformó un consorcio del UML. Entre los miembros se encuentran DEC, Hewlett-Packard, Intellicorp, Microsoft, Oracle, Texas Instruments y Rational. En 1997 el consorcio produjo la versión 1.0 del UML y lo puso a consideración del OMG (Grupo de administración de objetos) como respuesta a su propuesta para un lenguaje de modelado estándar.

El consorcio aumentó y generó la versión 1.1, misma que se puso nuevamente a consideración del OMG. El grupo adoptó esta versión a finales de 1997. El OMG se encargó de la conservación del UML y produjo otras dos revisiones en 1998. El UML ha llegado a ser el estándar de facto en la industria del software, y su evolución continúa.

Diagramas del UML

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos. En lugar de indicarle a usted cuáles son los elementos y las reglas, veamos directamente los diagramas ya que los utilizará para hacer el análisis del sistema.

Este enfoque es similar a aprender un idioma extranjero mediante el uso del mismo, en lugar de aprender sus reglas gramaticales y la conjugación de sus verbos. Después de un tiempo de hablar otro idioma se le facilitará la conjugación de verbos y la comprensión de las reglas gramaticales.

TÉRMINO NUEVO La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como *modelo*. El modelo UML de un sistema es similar a un modelo a escala de un edificio junto con la interpretación del artista del edificio. Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

A continuación se describirán brevemente los diagramas más comunes del UML y los conceptos que representan. Posteriormente, en la parte I verá cada uno de los diagramas con mayor detenimiento. Recuerde que es posible generar híbridos de estos diagramas y que el UML otorga formas de organizarlos y extenderlos.

Diagrama de clases

Piense en las cosas que le rodean (una idea demasiado amplia, pero ¡inténtelo de cualquier forma!). Es probable que muchas de esas cosas tengan atributos (propiedades) y que realicen determinadas acciones. Podríamos imaginar cada una de esas acciones como un conjunto de tareas.

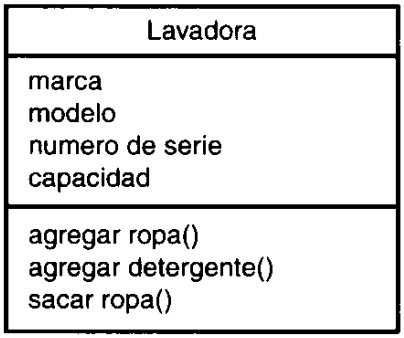
TÉRMINO NUEVO También se encontrará con que las cosas naturalmente se albergan en categorías (automóviles, mobiliario, lavadoras...). A tales categorías las llamaremos clases. Una *clase* es una categoría o grupo de cosas que tienen atributos y acciones similares. He aquí un ejemplo: cualquier cosa dentro de la clase Lavadoras tiene atributos como son la marca, el modelo, el número de serie y la capacidad. Entre las acciones

de las cosas de esta clase se encuentran: "agregar ropa", "agregar detergente", "activarse" y "sacar ropa".

La figura 1.1 le muestra un ejemplo de la notación del UML que captura los atributos y acciones de una lavadora. Un rectángulo es el símbolo que representa a la clase, y se divide en tres áreas. El área superior contiene el nombre, el área central contiene los atributos, y el área inferior las acciones. Un diagrama de clases está formado por varios rectángulos de este tipo conectados por líneas que muestran la manera en que las clases se relacionan entre sí.

FIGURA 1.1

El símbolo UML de una clase.



¿Qué objetivo tiene pensar en las clases, así como sus atributos y acciones? Para interactuar con nuestro complejo mundo, la mayoría del software moderno simula algún aspecto del mundo. Décadas de experiencia sugieren que es más sencillo desarrollar aplicaciones que simulen algún aspecto del mundo cuando el software representa clases de cosas reales. Los diagramas de clases facilitan las representaciones a partir de las cuales los desarrolladores podrán trabajar.

A su vez, los diagramas de clases colaboran en lo referente al análisis. Permiten al analista hablarle a los clientes en su propia terminología, lo cual hace posible que los clientes indiquen importantes detalles de los problemas que requieren ser resueltos.

Diagrama de objetos

TÉRMINO NUEVO

Un objeto es una instancia de clase (una entidad que tiene valores específicos de los atributos y acciones). Su lavadora, por ejemplo, podría tener la marca Laundatorium, el modelo Washmeister, el número de serie GL57774 y una capacidad de 7 Kg.

La figura 1.2 le muestra la forma en que el UML representa a un objeto. Vea que el símbolo es un rectángulo, como en una clase, pero el nombre está subrayado. El nombre de la instancia específica se encuentra a la izquierda de los dos puntos (:), y el nombre de la clase a la derecha.

FIGURA 1.2
El símbolo UML del objeto.

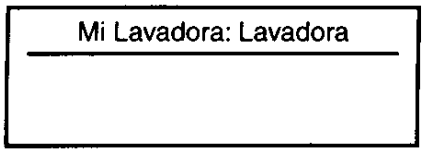
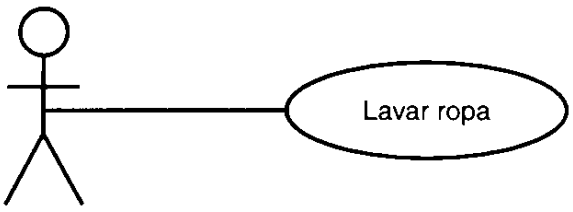


Diagrama de casos de uso

TÉRMINO NUEVO Un caso de uso es una descripción de las acciones de un sistema desde el punto de vista del usuario. Para los desarrolladores del sistema, ésta es una herramienta valiosa, ya que es una técnica de aciertos y errores para obtener los requerimientos del sistema desde el punto de vista del usuario. Esto es importante si la finalidad es crear un sistema que pueda ser utilizado por la gente en general (no sólo por expertos en computación).

Posteriormente trataremos este tema con mayor detalle; por ahora, le mostraré un ejemplo sencillo. Usted utiliza una lavadora, obviamente, para lavar su ropa. La figura 1.3 le muestra cómo representaría esto en un diagrama de casos de uso UML.

FIGURA 1.3
Diagrama de casos de uso UML.



TÉRMINO NUEVO A la figura correspondiente al Usuario de la lavadora se le conoce como actor. La elipse representa el caso de uso. Vea que el actor (la entidad que inicia el caso de uso) puede ser una persona u otro sistema.

Diagrama de estados

En cualquier momento, un objeto se encuentra en un estado en particular. Una persona puede ser recién nacida, infante, adolescente, joven o adulta. Un elevador se moverá hacia arriba, estará en estado de reposo o se moverá hacia abajo. Una lavadora podrá estar en la fase de remojo, lavado, enjuague, centrifugado o apagada.

El diagrama de estados UML, que aparece en la figura 1.4, captura esta pequeña realidad. La figura muestra las transiciones de la lavadora de un estado al otro.

El símbolo que está en la parte superior de la figura representa el estado inicial y el de la parte inferior el estado final.

FIGURA 1.4

Diagrama de estados UML.

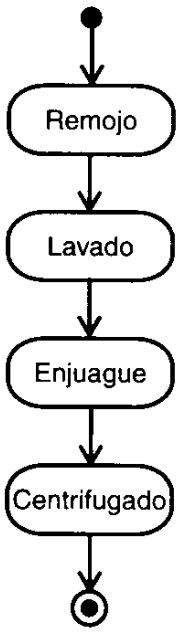


Diagrama de secuencias

Los diagramas de clases y los de objeto representan información estática. No obstante, en un sistema funcional los objetos interactúan entre sí, y tales interacciones suceden con el tiempo. El diagrama de secuencias UML muestra la mecánica de la interacción con base en tiempos.

Continuando con el ejemplo de la lavadora, entre los componentes de la lavadora se encuentran: una manguera de agua (para obtener agua fresca), un tambor (donde se coloca la ropa) y un sistema de drenaje. Por supuesto, estos también son objetos (como verá, un objeto puede estar conformado por otros objetos).

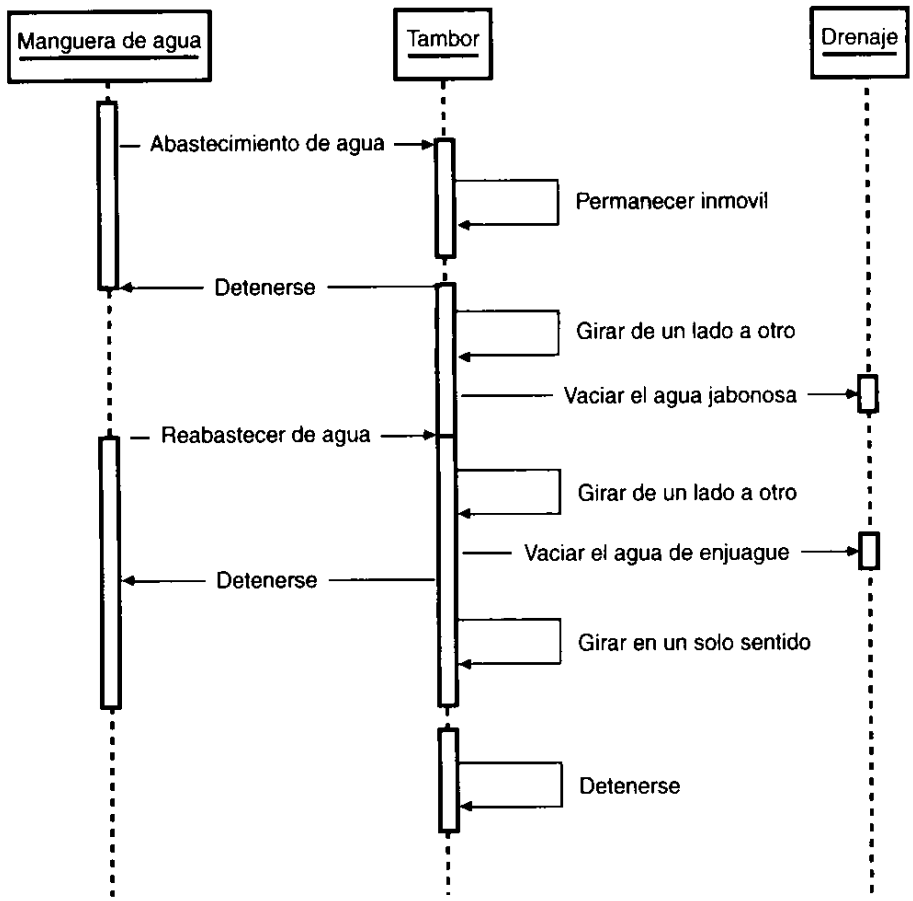
¿Qué sucederá cuando invoque al caso de uso Lavar ropa? Si damos por hecho que completó las operaciones "agregar ropa", "agregar detergente" y "activar", la secuencia sería más o menos así:

1. El agua empezará a llenar el tambor mediante una manguera.
2. El tambor permanecerá inactivo durante cinco minutos.
3. La manguera dejará de abastecer agua.
4. El tambor girará de un lado a otro durante quince minutos.
5. El agua jabonosa saldrá por el drenaje.
6. Comenzará nuevamente el abastecimiento de agua.
7. El tambor continuará girando.

8. El abastecimiento de agua se detendrá.
9. El agua del enjuague saldrá por el drenaje.
10. El tambor girará en una sola dirección y se incrementará su velocidad por cinco minutos.
11. El tambor dejará de girar y el proceso de lavado habrá finalizado.

La figura 1.5 presenta un diagrama de secuencias que captura las interacciones que se realizan a través del tiempo entre el abastecimiento de agua, el tambor y el drenaje (representados como rectángulos en la parte superior del diagrama). En este diagrama el tiempo se da de arriba hacia abajo.

FIGURA 1.5
Diagrama de secuencias UML.



Por cierto, volviendo a las ideas acerca de los estados, podríamos caracterizar los pasos 1 y 2 como el estado de remojo, 3 y 4 como el estado de lavado, 5 a 7 como el estado de enjuague y del 8 al 10 como el estado de centrifugado.

Diagrama de actividades

Las actividades que ocurren dentro de un caso de uso o dentro del comportamiento de un objeto se dan, normalmente, en secuencia, como en los once pasos de la sección anterior. La figura 1.6 muestra la forma en que el diagrama de actividades UML representa los pasos del 4 al 6 de tal secuencia.

FIGURA 1.6
Diagrama de actividades UML.

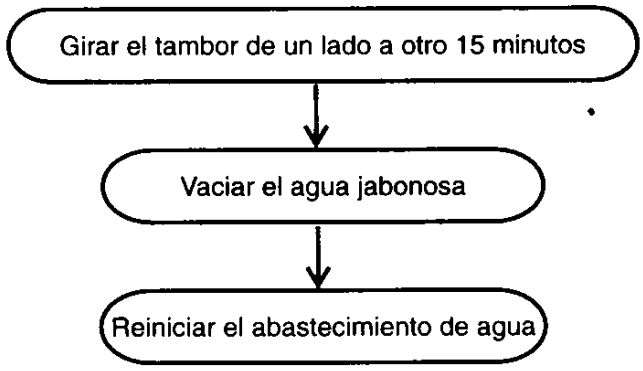


Diagrama de colaboraciones

Los elementos de un sistema trabajan en conjunto para cumplir con los objetivos del sistema, y un lenguaje de modelado deberá contar con una forma de representar esto. El diagrama de colaboraciones UML, diseñado con este fin, se muestra en la figura 1.7. Este ejemplo agrega un cronómetro interno al conjunto de clases que constituyen a una lavadora. Luego de cierto tiempo, el cronómetro detendrá el flujo de agua y el tambor comenzará a girar de un lado a otro.

FIGURA 1.7
Diagrama de colaboraciones UML.

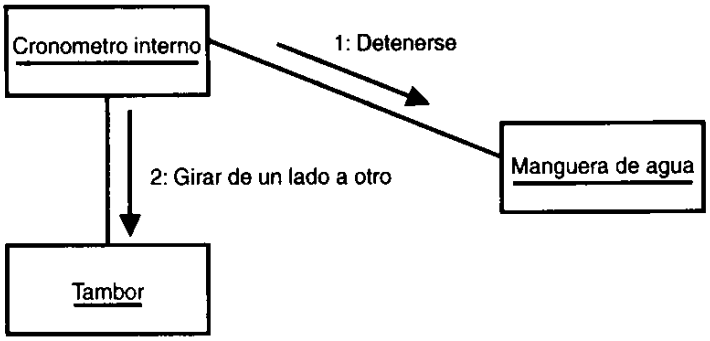


Diagrama de componentes

Este diagrama y el siguiente dejarán el mundo de las lavadoras, dado que están íntimamente ligados con los sistemas informáticos.

El moderno desarrollo de software se realiza mediante componentes, lo que es particularmente importante en los procesos de desarrollo en equipo. Sin extenderme mucho en este punto le mostraré, en la figura 1.8, la manera en que el UML representa un componente de software.

FIGURA 1.8
Diagrama de componentes UML.

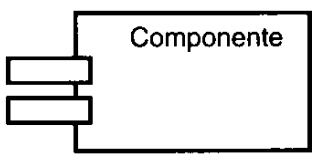
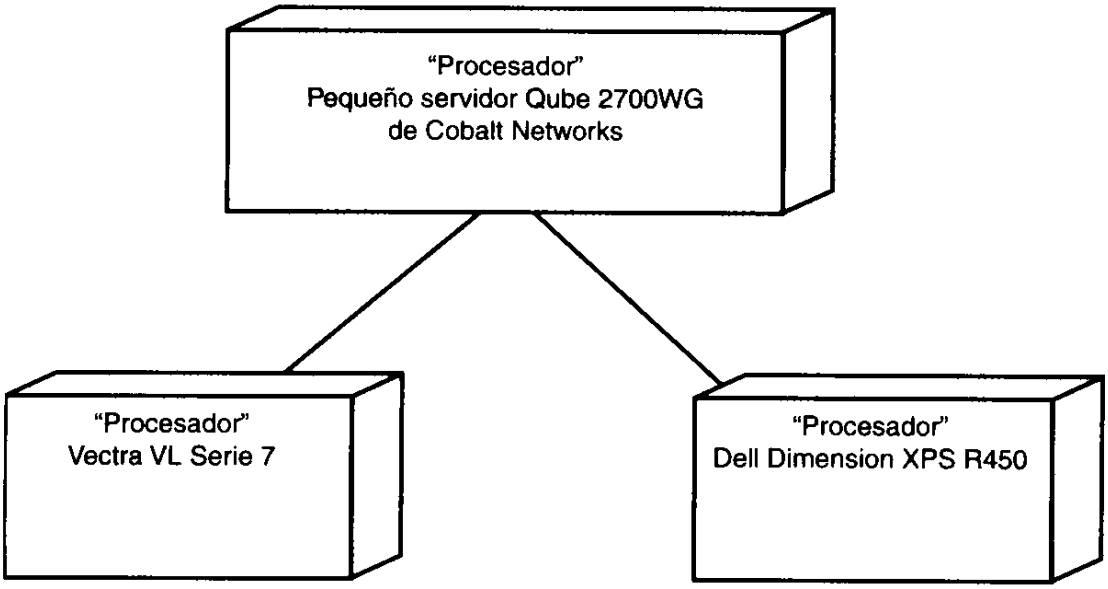


Diagrama de distribución

El diagrama de distribución UML muestra la arquitectura física de un sistema informático. Puede representar los equipos y dispositivos, mostrar sus interconexiones y el software que se encontrará en cada máquina. Cada computadora está representada por un cubo y las interacciones entre las computadoras están representadas por líneas que conectan a los cubos. La figura 1.9 presenta un ejemplo.

FIGURA 1.9

Diagrama de distribución UML.



Otras características

Anteriormente, mencioné que el UML proporciona características que le permiten organizar y extender los diagramas.

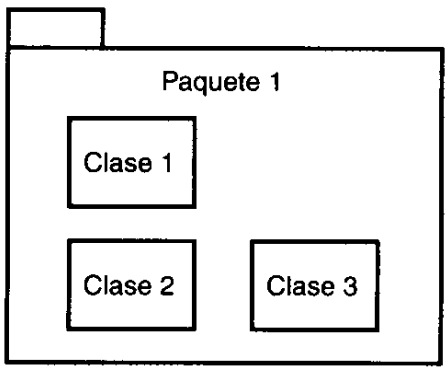
Paquetes

TERMINO NUEVO

En algunas ocasiones se encontrará con la necesidad de organizar los elementos de un diagrama en un grupo. Tal vez quiera mostrar que ciertas clases o componentes son parte de un subsistema en particular. Para ello, los agrupará en un *paquete*, que se representará por una carpeta tabular, como se muestra en la figura 1.10.

FIGURA 1.10

El paquete UML le permite agrupar los elementos de un diagrama.



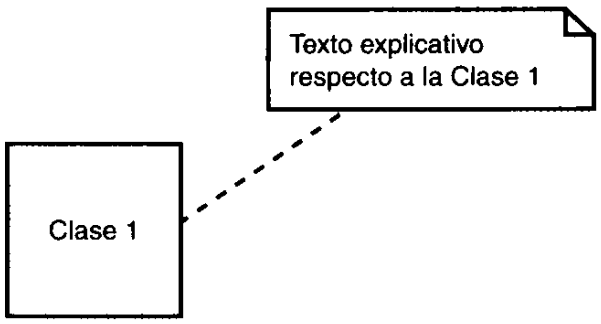
Notas

TÉRMINO NUEVO

Es frecuente que alguna parte del diagrama no presente una clara explicación del porqué está allí o la manera en que trabaja. Cuando éste sea el caso, la *nota* UML será útil. Imagine a una nota como el equivalente gráfico de un papel adhesivo. La nota es un rectángulo con una esquina doblada, y dentro del rectángulo se coloca la explicación. Usted adjunta la nota al elemento del diagrama conectándolos mediante una línea discontinua.

FIGURA 1.11

En cualquier diagrama, podrá agregar comentarios aclaratorios mediante una nota.



Estereotipos

TÉRMINO NUEVO

El UML otorga varios elementos de utilidad, pero no es un conjunto minucioso de ellos. De vez en cuando diseñará un sistema que requiera algunos elementos hechos a la medida. Los *estereotipos* o clisés le permiten tomar elementos propios del UML y convertirlos en otros. Es como comprar un traje del mostrador y modificarlo para que se ajuste a sus medidas (contrario a confeccionarse uno completamente nuevo). Imagine a un estereotipo como este tipo de alteración. Lo representará como un nombre entre dos pares de paréntesis angulares y después los aplicará correctamente.

TÉRMINO NUEVO

El concepto de una interfaz provee un buen ejemplo. Una interfaz es una clase que realiza operaciones y que no tiene atributos, es un conjunto de acciones que tal vez quiera utilizar una y otra vez en su modelo. En lugar de inventar un nuevo elemento para representar una interfaz, podrá utilizar el símbolo de una clase con «Interfaz» situada justo sobre el nombre de la clase, como se muestra en la figura 1.12.

FIGURA 1.12

Un estereotipo le permite crear nuevos elementos a partir de otros existentes.

Para qué tantos diagramas

Como puede ver, los diagramas del UML le permiten examinar un sistema desde distintos puntos de vista. Es importante recalcar que en un modelo UML no es necesario que aparezcan todos los diagramas. De hecho, la mayoría de los modelos UML contienen un subconjunto de los diagramas que he indicado.

TERMINO NUEVO

¿Por qué es necesario contar con diferentes perspectivas de un sistema? Por lo general, un sistema cuenta con diversas personas implicadas las cuales tienen enfoques particulares en diversos aspectos del sistema. Volvamos al ejemplo de la lavadora. Si diseñara el motor de una lavadora, tendría una perspectiva del sistema; si escribiera las instrucciones de operación, tendría otra perspectiva. Si diseñara la forma general de la lavadora vería al sistema desde una perspectiva totalmente distinta a si tan sólo tratara de lavar su ropa.

El escrupuloso diseño de un sistema involucra todas las posibles perspectivas, y el diagrama UML le da una forma de incorporar una perspectiva en particular. El objetivo es satisfacer a cada persona implicada.

Resumen

El desarrollo de sistemas es una actividad humana. Sin un sistema de notación fácil de comprender, el proceso de desarrollo tiene una gran cantidad de errores.

El UML es un sistema de notación que se ha convertido en estándar en el mundo del desarrollo de sistemas. Es el resultado del trabajo hecho por Grady Booch, James Rumbaugh e Ivar Jacobson. El UML está constituido por un conjunto de diagramas, y proporciona un estándar que permite al analista de sistemas generar un anteproyecto de varias facetas que sean comprensibles para los clientes, desarrolladores y todos aquellos que estén involucrados en el proceso de desarrollo. Es necesario contar con todos esos diagramas dado que cada uno se dirige a cada tipo de persona implicada en el sistema.

Un modelo UML indica *qué* es lo que supuestamente hará el sistema, mas no *cómo* lo hará.

Preguntas y respuestas

P He visto que se refiere al Lenguaje Unificado de Modelado como "UML" y como "el UML". ¿Cuál es el correcto?

R Los creadores del lenguaje prefieren el uso de "el UML".

P Ha indicado que el UML es adecuado para los analistas. No obstante, el diagrama de distribución no parece ser algo muy útil en la fase de análisis en el desarrollo de un sistema. ¿No sería más apropiado para una fase posterior?

R En realidad nunca será demasiado pronto para empezar a pensar en la distribución (u otras cuestiones que, tradicionalmente, se dejan para fases posteriores del desarrollo). Aunque es cierto que el analista se interesa por hablar con los clientes y usuarios, en las fases tempranas del proceso el analista debería pensar en los equipos y componentes que constituirían el hardware del sistema. En algunas ocasiones, el cliente dicta esto; en otras, el cliente desea una recomendación del equipo de desarrollo. Ciertamente, un arquitecto de sistemas encontrará útil al diagrama de distribución.

P Ha mencionado que es posible hacer diagramas híbridos. ¿UML, perdón, *el* UML, impone limitaciones respecto a los elementos que podrá combinar en un diagrama?

R No. El UML no establece límites, no obstante, con frecuencia se da el caso de que un diagrama contenga un tipo de elemento. Podrá colocar símbolos de clases en un diagrama de distribución, pero ello no será muy útil.

Taller

Ya se ha iniciado en el UML. Ahora deberá reafirmar su conocimiento de esta gran herramienta al responder algunas preguntas y realizar los ejercicios. Las respuestas aparecerán en el Apéndice A, "Respuestas a los cuestionarios".

Cuestionario

1. ¿Porqué es necesario contar con diversos diagramas en el modelo de un sistema?
2. ¿Cuáles diagramas le dan una perspectiva estática de un sistema?
3. ¿Cuáles diagramas le dan una perspectiva dinámica de un sistema (esto es, muestran el cambio progresivo)?

Ejercicios

1. Suponga que creará un sistema informático que jugará ajedrez con un usuario. ¿Cuáles diagramas UML serían útiles para diseñar el sistema? ¿Por qué?
2. Para el sistema del ejercicio que ha completado, liste las preguntas que formularía a un usuario potencial y por qué las haría.