Una Introducción al UML

El Modelo Lógico

Autor: Geoffrey Sparks, Sparx Systems, Australia

Traducción: Fernando Pinciroli (Solus S.A., Argentina) y Aleksandar Orlic (Craftware Consultores Ltda., Chile)

www.sparxsystems.com.ar - www.sparxsystems.cl

Tabla de Contenidos

El Modelo Lógico

El modelo lógico se usa en el UML para modelar los elementos estructurales estáticos. Captura y define los objetos, entidades y bloques de construcción de un sistema. Las clases son los moldes genéricos a partir de los que se crean los objetos en tiempo de ejecución del sistema. Los componentes (se discuten en "El modelo de componentes") se construyen a partir de las clases. Las clases (y las interfaces) son los elementos de diseño que corresponden a los artefactos de software codificados o desarrollados. Este artículo describirá algunas características del modelo de clases, cubrirá la notación del UML para describir las clases/objetos y dará un ejemplo del uso de la notación.

Introducción al UML

El Lenguaje Unificado de Modelado (UML) es, tal como su nombre lo indica, un lenguaje de modelado y no un método o un proceso. El UML está compuesto por una notación muy específica y por las reglas semánticas relacionadas para la construcción de sistemas de software. El UML en sí mismo no prescribe ni aconseja cómo usar esta notación en el proceso de desarrollo o como parte de una metodología de diseño orientada a objetos.

El UML soporta un conjunto rico en elementos de notación gráficos. Describe la notación para clases, componentes, nodos, actividades, flujos de trabajo, casos de uso, objetos, estados y cómo modelar la relación entre esos elementos. El UML también soporta la idea de extensiones personalizadas a través elementos estereotipados.

El UML provee beneficios significativos para los ingenieros de software y las organizaciones al ayudarles a construir modelos rigurosos, trazables y mantenibles, que soporten el ciclo de vida de desarrollo de software completo.

Este artículo enfoca el modelo lógico o estático.

En los libros mencionados en la sección de lectura recomendada se puede encontrar más información sobre el UML y de los documentos de especificación del UML que se pueden encontrar en las paginas de recursos de UML del OMG (*Object Management Group*) www.omg.org/technology/uml/ y www.omg.org/technology/documents/formal.

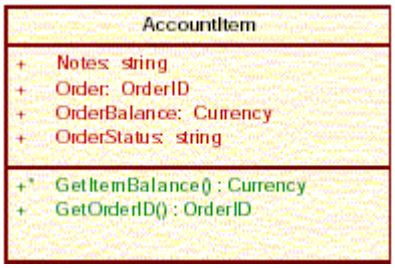
El Modelo Lógico

Un modelo lógico es una vista estática de los objetos y las clases que cubren el espacio de análisis y diseño. Típicamente, un modelo de dominio es una vista más pobre, de alto nivel de los objetos de negocio y de las entidades, mientras que el modelo de clases es un modelo más riguroso y enfocado al diseño. Esta discusión describe principalmente el modelo de clases.

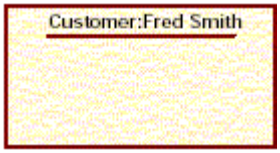
Objetos y Clases

Una clase es un elemento estándar del UML, que se usa para especificar el patrón del que se producirán los objetos en tiempo de ejecución. Una clase es una especificación; un objeto es una instancia de una clase. Las clases se pueden heredar de otras clases (es decir, heredan todo el comportamiento y el estado de sus padres y agregan nueva funcionalidad propia), pueden tener otras clases como atributos, pueden delegar sus responsabilidades a otras clases e implementar interfaces abstractas. El modelo de clases está en el núcleo del desarrollo y del diseño orientados a objetos; expresa el estado persistente y el comportamiento del sistema. Una clase encapsula el estado (los atributos) y ofrece los servicios para manipularlo (el comportamiento). Un buen diseño orientado a objetos limita el acceso directo a los atributos de la clase y ofrece los servicios que manipulan a solicitud del solicitante. Este ocultamiento de los datos y exposición de los servicios asegura que las modificaciones de los datos se realizan sólo en un lugar y de acuerdo con reglas específicas; para grandes sistemas la cantidad de código que tiene acceso directo a los elementos de datos en muchos sitios es extremadamente alto.

Las clases se representan usando la siguiente notación:



y los objetos con la notación que sigue:



Características de una Clase

Observe que la clase tiene tres áreas diferentes:

El nombre de la clase (y el estereotipo, si corresponde)
El área de los atributos (es decir los elementos de datos internos)
El comportamiento; privado y público

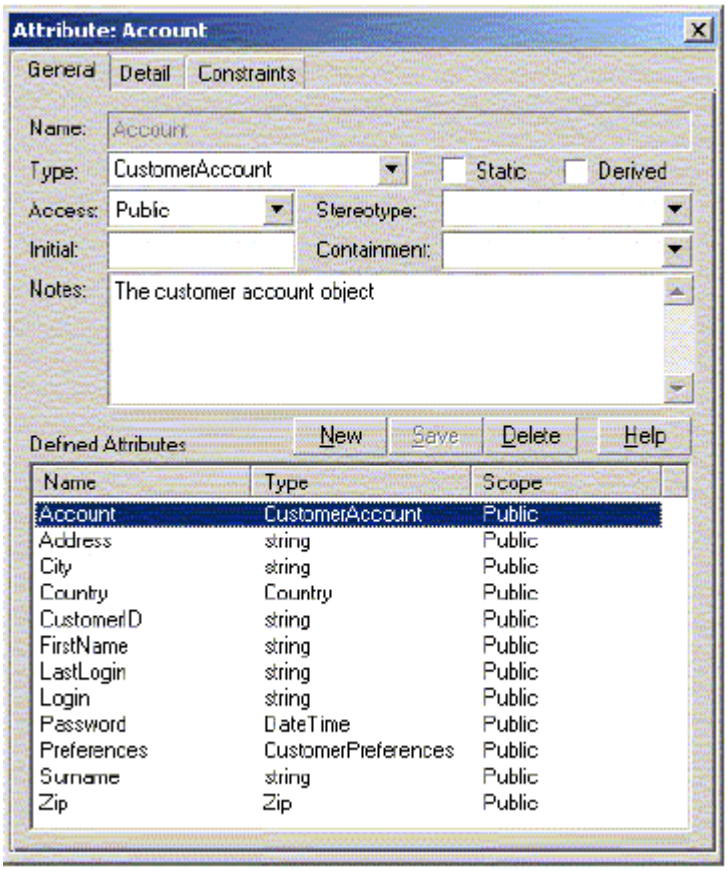
La imagen siguiente muestra estas tres áreas (los atributos en rojo, las operaciones en verde):

Los atributos y los métodos pueden ser marcados como:
Privados (*private*), indicando que no son visibles para los solicitantes fuera de la clase
Protegidos (*protected*), son visibles sólo para las clases hijas
Públicos (*public*), son visibles para todos.

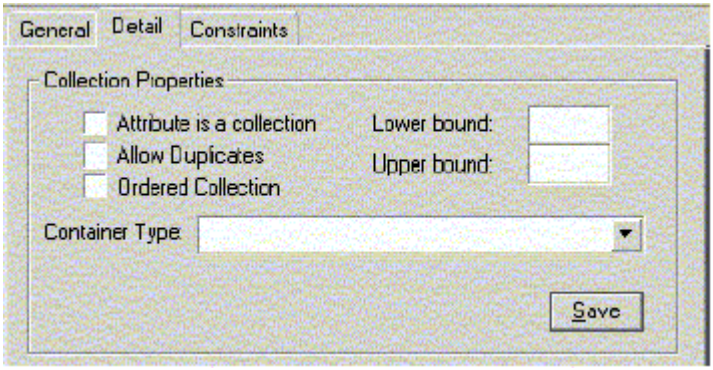
Las convenciones textuales siguiente corresponden a la notación:

Símbolo	Significado
+	Público
-	Privado
#	Protegido
$	Estático
/	Derivado (atributo – no estándar)
*	Abstracto (atributo – no estándar)

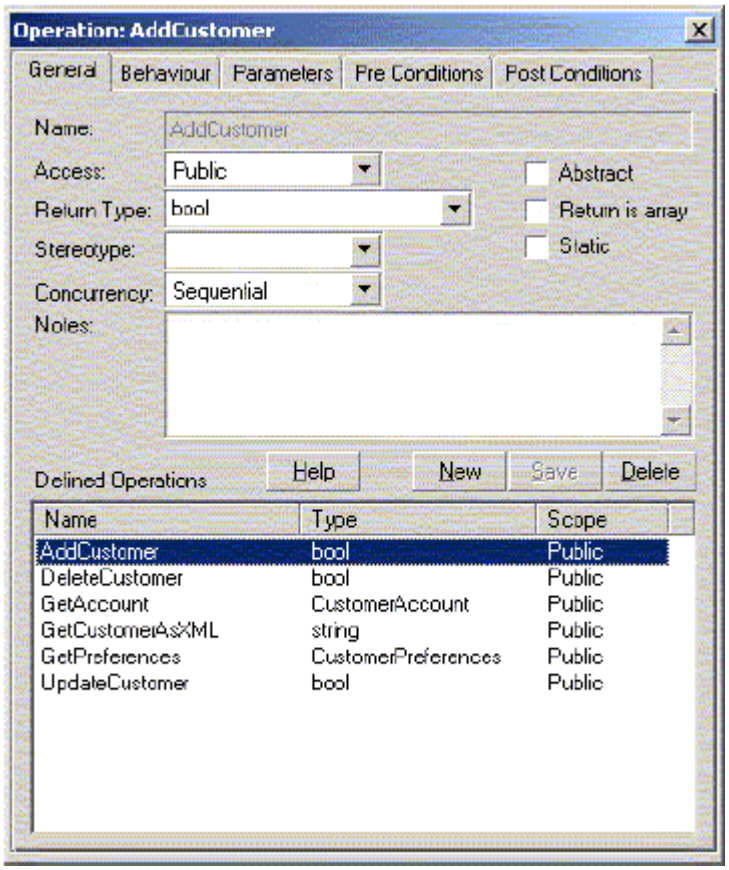
Una herramienta CASE le permitirá establecer numerosos detalles de los atributos, como muestra el ejemplo a continuación:



Y algunos detalles adicionales:



De la misma manera para las operaciones de las clases:



Interfaces

Las interfaces son clases abstractas con sólo comportamientos. Especifican el contrato operacional que un implementador acepta satisfacer. Por ejemplo, se puede definir una interfaz llamada *IPerson* con los métodos para acceder a la dirección (*Address*) y a la Edad (*Age*) como en el ejemplo a continuación:

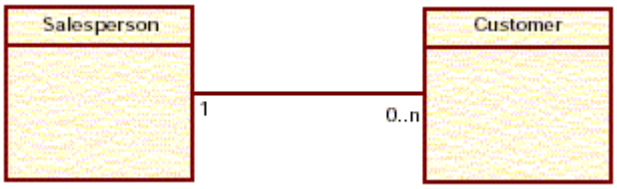
La interfaz no se puede instanciar en tiempo de ejecución; es un elemento abstracto. En su lugar, se puede definir una clase que 'implemente' esta interfaz. De esta forma, la clase se obliga a proveer implementaciones de los métodos de la interfaz con la firma exacta que se especifica en la definición de la Interfaz. Esto permite que numerosas clases de diferentes jerarquías y con diferentes superclases y diferentes comportamientos sean tratadas todas como del tipo IPerson (si implementan la interfaz) en tiempo de ejecución.

Relaciones

Los elementos lógicos se pueden relacionar en una gran variedad de maneras. Las siguientes son las más comunes:

Asociación

Las relaciones de asociación capturan las relaciones estáticas entre las entidades. Generalmente se relacionan a un objeto que tiene una instancia de otro como atributo o estando relacionado en términos de posesión (pero no estando compuesto de). Por ejemplo, un Vendedor tiene una asociación a sus Clientes (como en el dibujo de más abajo), pero un Vendedor no está 'compuesto de' Clientes. Ambos extremos del enlace de la asociación pueden asumir un 'rol' como parte de la asociación; por ejemplo el Vendedor puede asumir el rol de 'Vendedor de alfombras' si hay condiciones especiales que impliquen este rol.

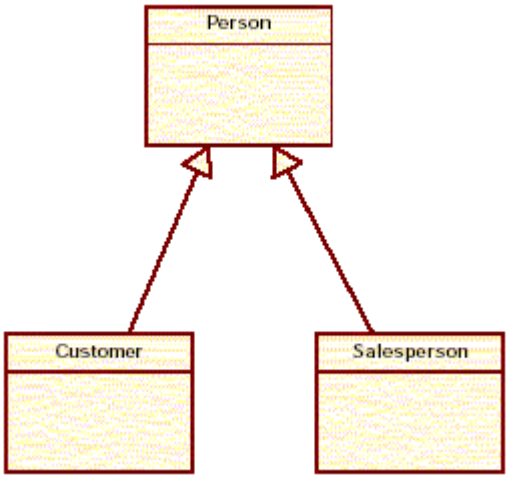


Agregación

Las relaciones de agregación definen relaciones todo/parte. Por ejemplo, un viaje en tren puede estar compuesto por varios tramos del viaje. Así, el viaje completo agrega o está compuesto de tramos separados. La forma más fuerte de agregación es la composición e infiere que una clase no sólo colecciona a otra clase sino que realmente está compuesta de esta colección. El ejemplo de más abajo muestra que una tienda (*Store*) agrega o reúne una cantidad de departamentos (*Departments*) y que hay una relación todo/parte; es decir, una tienda está compuesta de (in partes) esta colección de departamentos.

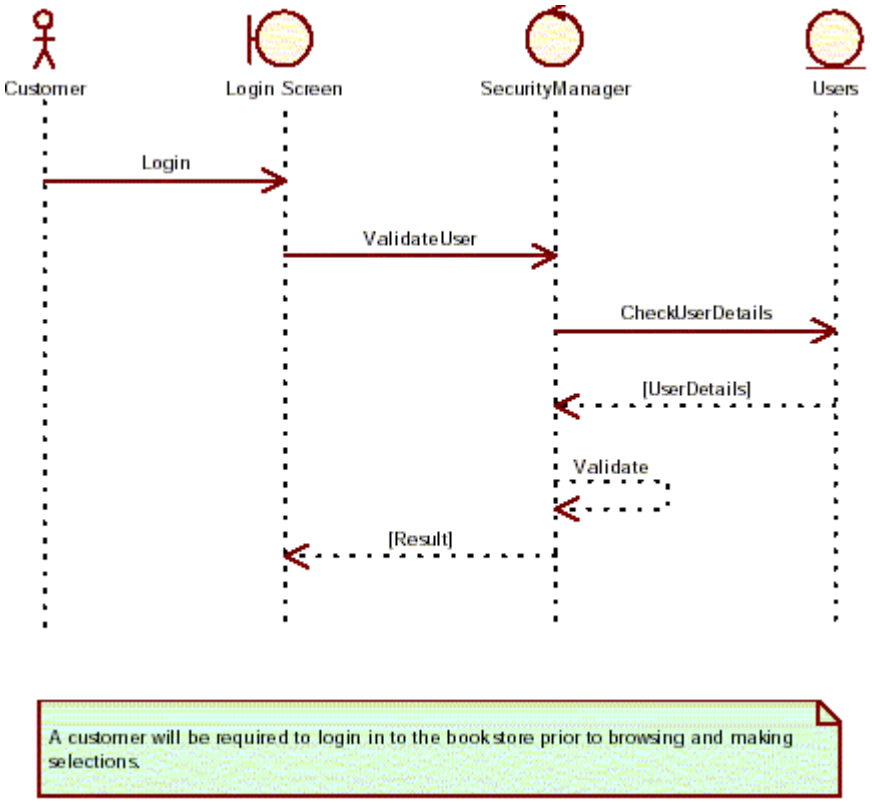
Herencia

La herencia describe la relación jerárquica entre las clases. Describe el 'árbol genealógico'. Las clases pueden heredar los atributos y el comportamiento de una clase padre (que, a su vez, puede ser el hijo de otra clase). Este árbol de características y comportamiento heredados brinda al diseñador la posibilidad de colectar la funcionalidad común en clases raíces (ancestros) y refinar y especializar este comportamiento en una o más generaciones (hijos). Los especificadores de alcance (público, protegido y privado) determinan qué elementos se pueden heredar y cuáles no son visibles. El siguiente diagrama muestra un nivel de herencia:



Dinámica

Las relaciones dinámicas son los mensajes que se pasan entre las clases (los objetos durante el tiempo de ejecución). Un diagrama de secuencias ilustra este paso de mensajes y la secuencia en la que ocurre. El dibujo de más abajo muestra la secuencia de mensajes que se pasan a lo largo del tiempo cuando un usuario ingresa a una tienda de libros en línea. Estos elementos del modelo tienen una asociación entre sí que es reflejada por el paso de mensajes en tiempo de ejecución.



Restricciones, Requisitos, Estados y Escenarios

Un elemento del modelo lógico y los atributos, asociaciones y operaciones que posee pueden ser especificados mucho más con las restricciones, los requisitos y los escenarios.

Restricciones

Son las reglas contractuales que se aplican a un elemento o a sus características. Típicamente caen en uno de tres tipos:

1. Pre-condiciones; deben ser verdaderas antes de que opere o de que exista un objeto
2. Post-condiciones; deben ser verdaderas luego del uso o de la destrucción de un objeto
3. Invariantes; deber ser siempre verdaderas para la vida y uso de un objeto

Requisitos

Especifican qué debería ser capaz de hacer una clase, qué información deberá ser persistente, qué comportamiento se soporta y qué relaciones se requieren. Por ejemplo, una clase Persona puede requerir el almacenamiento del Nombre, Edad, Sexo, Fecha de Nacimiento, etc.

Estados

Las clases se pueden entender como poseedoras de diferentes estados alo largo del tiempo (por ejemplo, un Cliente está Navegando, Seleccionando, Comprando, etc). El UML permite al diseñador modelar estos estados usando las Cartas de Estados. Ellas definen los estados legales en los que un objeto puede estar, junto con las transiciones, las condiciones que inician las transiciones y las condiciones de guarda que previenen las transiciones si aquellas no son satisfechas.

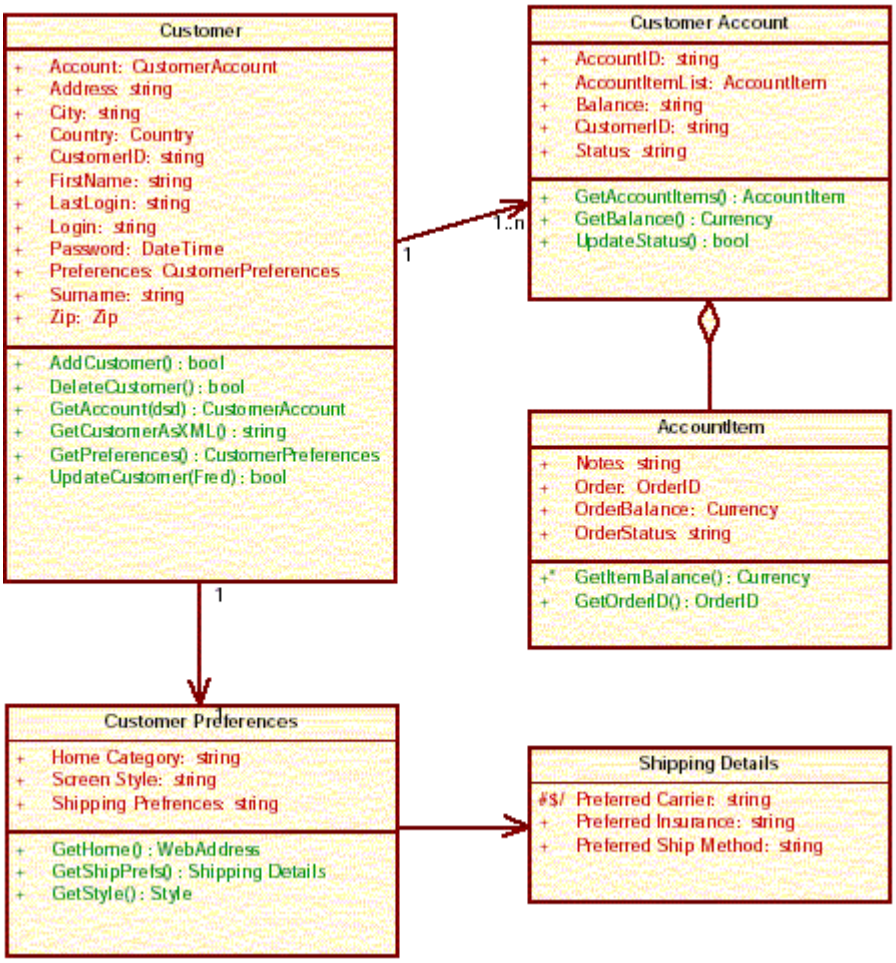
Escenarios

Los escenarios son representaciones textuales (o gráficas) descriptivas de cómo se usa una clase a lo largo del tiempo en una forma cooperativa.

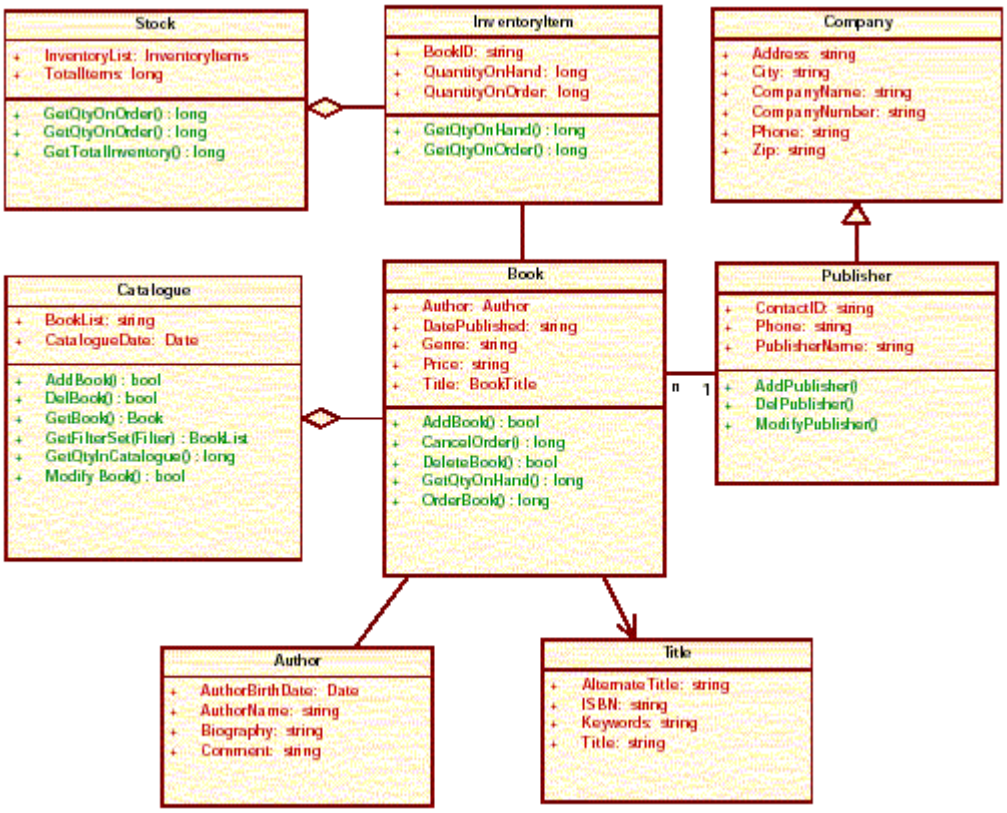
Juntando las Piezas

Ejemplo de Diagrama de Clases

Los ejemplos siguientes muestran cómo se usa la notación para diseñar un modelo estático. El primer diagrama muestra las relaciones entre un Cliente (*Customer*), su Cuenta (*Customer Account*) y los Detalles de Cuenta (*Account Item*). Se incluyen también las preferencias del cliente. La notación muestra que un cliente puede tener una o más cuentas y que las cuenta se componen de uno o más ítems. Se detallan los atributos (o datos) asociados a cada una de la clase (en rojo) y el comportamiento soportado (en verde).

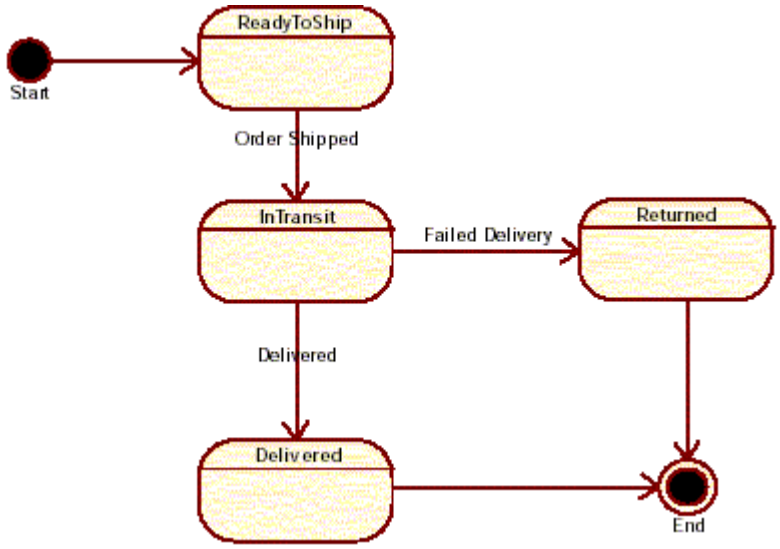


El segundo ejemplo ilustra elementos semejantes, incluyendo un vínculo desde el Editor *(Publisher)* a la Empresa *(Company)*; (indicando que un Editor es una Empresa).



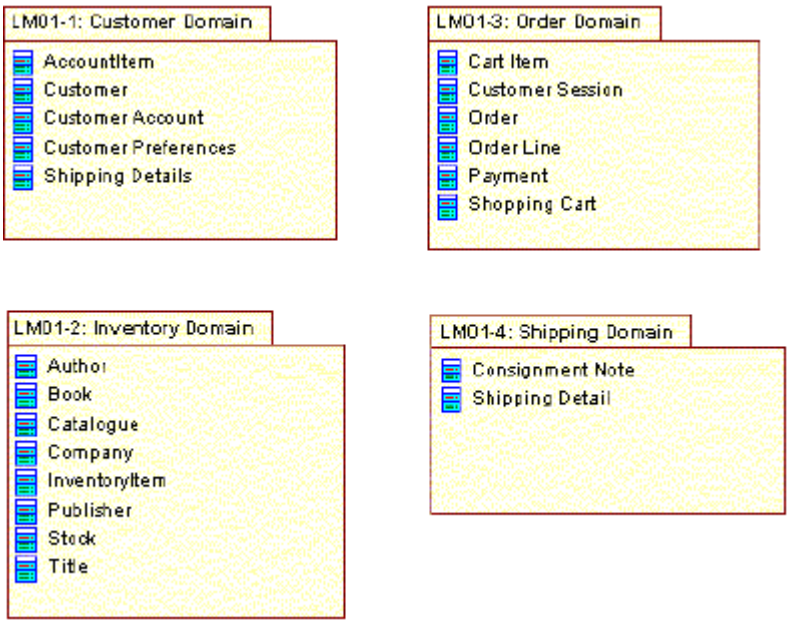
Carta de Estados de Ejemplo

El diagrama siguiente muestra los estados a través de los que puede transitar una orden de envío; desde estar lista para su envío, estar en tránsito, entregada o devuelta.



Los Paquetes

A medida que el modelo de clases avanza, las clases (y los objetos e interfaces) se pueden organizar en unidades lógicas o paquetes. Ellos juntan elementos relacionados entre sí (y en algunas implementaciones regularán la visibilidad de las operaciones -y de los atributos- por parte de elementos externos al paquete). El siguiente diagrama ilustra la reunión de clases dentro de paquetes.



Lectura Recomendada

Sinan Si Alhir, **UML in a NutShel**.
ISBN: 1-56592-448-7. Publisher: O'Reilly & Associates, Inc

Doug Rosenberg with Kendall Scott, Logical Driven Object Modeling with UML
ISBN:0-201-43289-7. Publisher: Addison-Wesley

Geri Scheider, Jason P. Winters, **Applying Use Logicals**
ISBN: 0-201-30981-5. Publisher: Addison-Wesley

Ivar Jacobson, Martin Griss, Patrik Jonsson, **Software Reuse**
ISBN:0-201-92476-5. Publisher: Addison-Wesley

Hans-Erik Eriksson, Magnus Penker, **Business Modeling with UML**
ISBN: 0-471-29551-5. Publisher: John Wiley & Son, Inc

Peter Herzum, Oliver Sims, **Business Component Factory**
ISBN: 0-471-32760-3 Publisher: John Wiley & Son, Inc