BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

BUAP

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

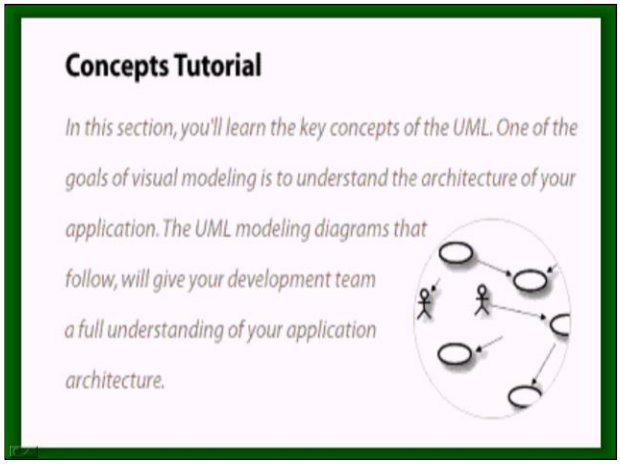
INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

APUNTES DE LA MATERIA DE INGENIERÍA SW II
OTOÑO 2022

VI.5.- CONCEPTOS DEL UML

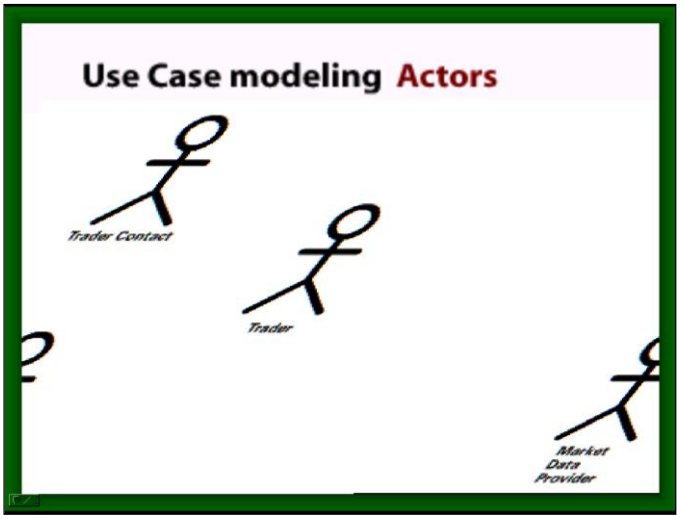
VI.5.1. MODELADO DE CASOS DE USO.

En esta sección se hablará de cómo modelar la funcionalidad inmersa en un sistema software utilizando para ello los denominados *casos de uso y actores*. A través de los casos de uso se puede representar de forma gráfica el comportamiento de un sistema software. Un diagrama de casos de uso ilustra las relaciones existentes entre los actores y los casos de uso del sistema software. Las técnicas de los casos de uso ayudan a definir el límite del sistema.



VI.5.1.2. ACTORES

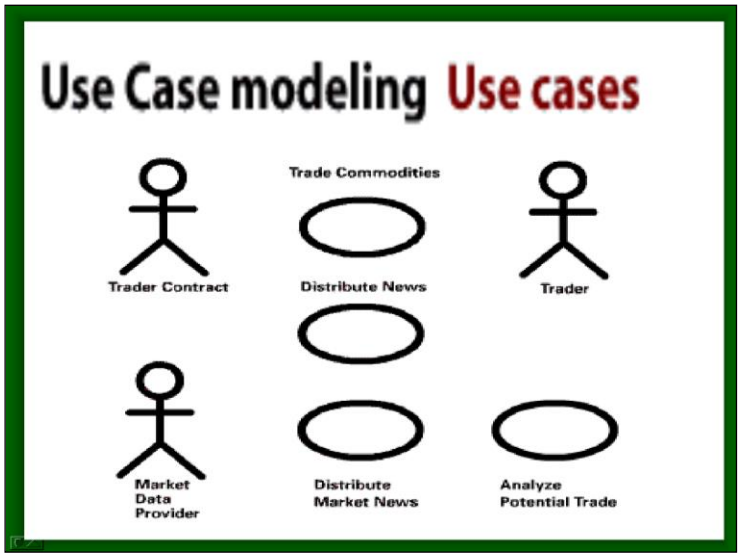
Un actor es alguien o algo (identidad) que actúa sobre el sistema software. Los actores representan cualquier cosa que pueda intercambiar información con el sistema. Supongamos que se tiene un producto software de comercio. Se pueden identificar en principio dos actores que representan una interacción humana: el vendedor (Trader) y el comerciante contactado (Trader Contact). Con ello se identifica entonces un actor más que representa la interacción con el sistema externo: el proveedor de mercado de datos (Market Data Provider).



VI.5.1.3. CASOS DE USO

Un sistema software debe satisfacer las necesidades de sus usuarios. Dichas necesidades se capturan como *casos de uso*. Un caso de uso es un camino o una forma de usar un sistema. Un caso de uso también se define como un patrón de comportamiento de un sistema. Cada caso de uso es una secuencia de transacciones relacionadas entre sí. Dicha secuencia es llevada a cabo entre el actor y el sistema mediante un diálogo. Toda la colección de los casos de uso especifica todos los caminos o formas que se tienen para utilizar un sistema software. Los casos de uso se detectan identificando a los actores y definiendo qué es lo que pueden hacer cada uno de ellos con el sistema en cuestión. Para el caso del ejemplo (producto software de comercio) se identifican cuatro casos de uso:

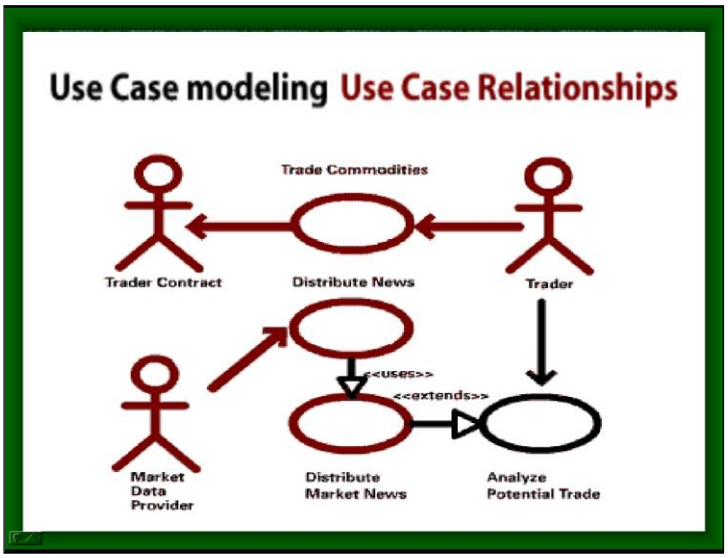
- Noticias (Distribute News)
- Nuevos Mercados (Distribute Market News)
- Mercancías de Mercado (Trade Commodities)
- Análisis del potencial de Mercado (Analyze Potential Trade)



VI.5.1.4. RELACIONES ENTRE CASOS DE USO

Una vez identificados y diagramados los casos de uso, se añaden relaciones al diagrama para ilustrar las interacciones de los actores con los casos de uso. Los casos de uso son utilizados por los actores y estos últimos pueden utilizar las capacidades especificadas en otros casos de uso. En el ejemplo del sistema comercial dentro del diagrama de casos de uso, el actor Trader o vendedor participa en dos casos de uso distintos: el caso de uso Análisis del Potencial de Mercado (Analyze Potential Trade) y el caso de uso Mercancías de Mercado (Trade Commodities). Las etiquetas *"usos" y "extensiones"* son relaciones especiales entre casos de uso; a saber son ejemplos de estereotipos (propiedades que pueden añadirse a elementos de un modelo simple para especializarlo).

Nuevamente en nuestro ejemplo, el vendedor o Trader inicia el caso de uso Mercancías de Mefrcado (Trade Commodities) y este caso de uso a su vez llama al actor Comerciante Contactado (Contact Trader). Mientras tanto el Proveedor de mercado de datos (Market Data Provider) inicia el caso de uso Noticias (Distribute News) donde se añade un comportamiento proporcionado por el caso de uso Nuevos Mercados (Distribute Market News).



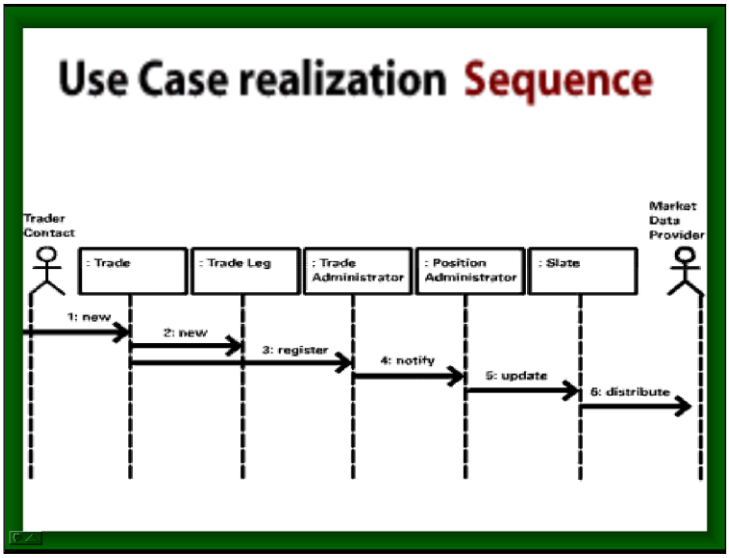
VI.5.2. REALIZACION DE CASOS DE USO

Por un lado los diagramas de casos de uso presentan la parte externa del sistema (es decir, casos de usabilidad del producto software), pero por otro, necesitamos establecer colaboraciones entre los procesos para poder describir como los casos de uso se realizan como interacciones creando así lo que se denominan como sociedades de objetos. Las realizaciones de los casos de uso se llevan a cabo a través de dos formas:

- Vía diagramas de secuencia, o bien
- Vía diagramas de colaboración

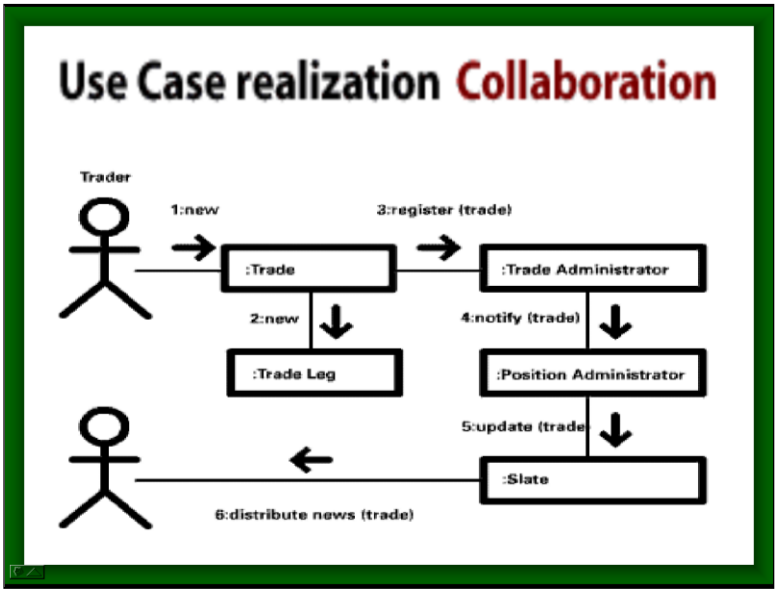
VI.5.2.1. REALIZACIÓN DE CASOS DE USO MEDIANTE SECUENCIAS

Imagine la realización de un caso de uso en donde el actor interactúa con los objetos en un modelo dado. Un diagrama de secuencias despliega interacciones entre objetos en un lapso de tiempo dado. Los rectángulos representan objetos. Las líneas verticales discontinuas bajo cada rectángulo representan las veces y el conjunto de eventos o respuestas realizadas entre los objetos en cuestión. Las flechas horizontales representan los mensajes entre los objetos participantes. Dichos mensajes ilustran la funcionalidad de los casos de uso. En el ejemplo de nuestro sistema comercial, el diagrama de secuencias representa la interacción de objetos necesaria para crear un nuevo Trade (comercio o mercado).



VI.5.2.2. REALIZACIÓN DE CASOS DE USO MEDIANTE COLABORACIONES

Un diagrama de colaboración es una representación gráfica alternativa de la realización de un caso de uso. Dicho diagrama ilustra las interacciones entre objetos organizadas alrededor de ellos y de sus ligas con algún otro objeto. El diagrama de colaboración representa en el caso del ejemplo que venimos tratando, un nuevo caso de uso para el Trade o comercio y su realización. Con estos conceptos el modelar casos de uso implica el visualizar los procesos existentes en una determinada empresa o negocio.



VI.5.3. MODELADO DE CLASES

En esta sección se ilustra el modelado de los diagramas de clases, así como el uso de los diagramas de transición de estados como herramientas poderosas y flexibles para visualizar tanto estructura como comportamiento de un sistema software.

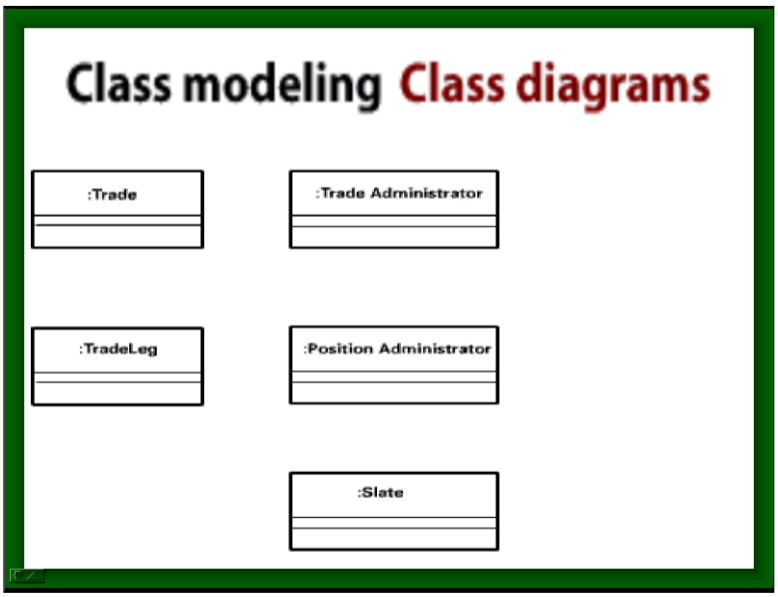
VI.5.3.1. DIAGRAMAS DE CLASES

La estructura de un sistema software surge cuando uno descubre los objetos que intervienen en dicho sistema. Un diagrama de clases ilustra las clases existentes dentro de un producto software así como las relaciones entre ellas. Por tanto, la naturaleza estática de un sistema se representa mediante los diagramas de clases. Las responsabilidades de los casos de uso se localizan en los objetos. La agrupación de dichos objetos en clases ayuda a manejar la complejidad de un sistema. Una clase por tanto, es una colección de objetos que comparten:

- una estructura común
- un comportamiento común
- unas relaciones comunes
- semánticas también comunes

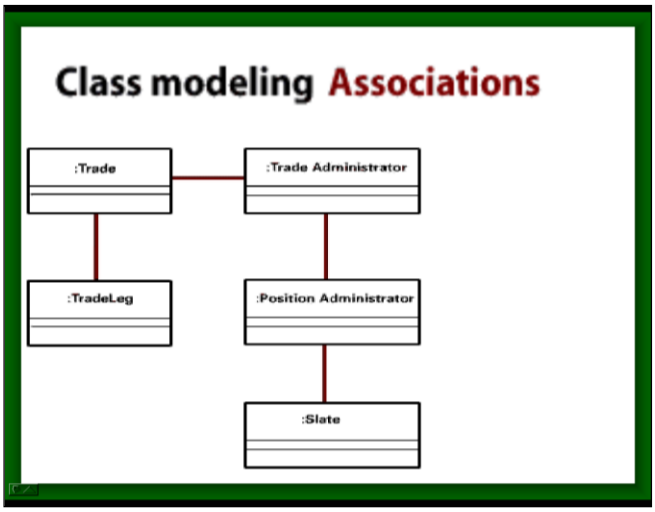
En nuestro ejemplo, en base al caso de uso “Crear un nuevo mercado o comercio” (Create New Trade) utilizamos objetos que forman parte de 5 clases:

La clase Trade (Comercio o Mercado)
La clase Trade Leg (Etapa de Comercio)
La clase Trade Administrator (Administrador de Mercado o comercio)
La clase Position Administrator (Posición del Administrador)
La clase Slate (lista de candidatos)



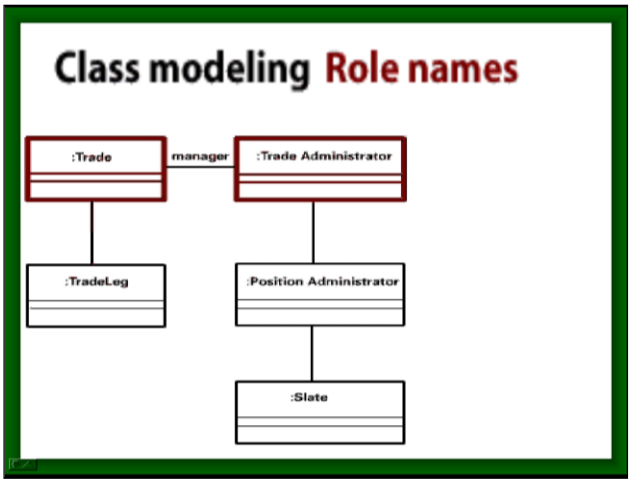
VI.5.3.2. ASOCIACIONES

Las relaciones entre clases proporcionan un camino para la comunicación entre objetos. Una asociación es un tipo de relación. Las asociaciones entre clases son modeladas con líneas bidireccionales que establecen la relación entre clases. Los diagramas de secuencia y/o colaboración se examinan para determinar las ligas que puedan existir entre los objetos que se requieren existan para definir el comportamiento del sistema (por ejemplo, cuando dos objetos necesitan comunicarse, se puede establecer una liga entre ellos).



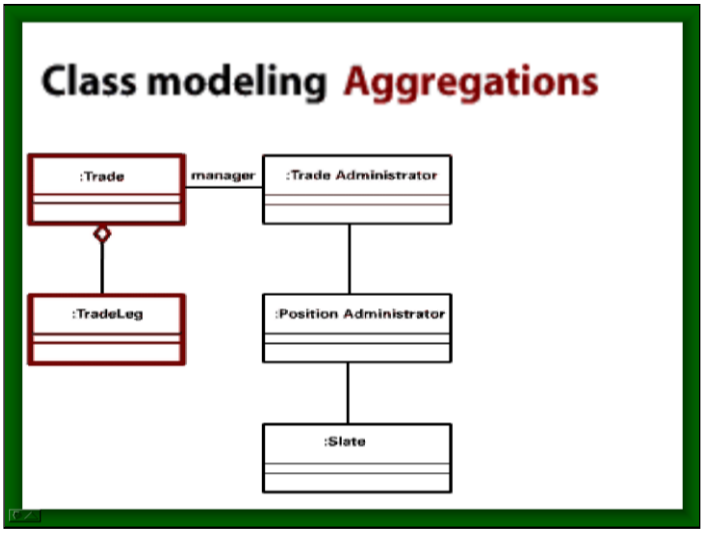
VI.5.3.3. ROLES

Uno de los propósitos de un diagrama de clases es la comunicación. Para ello es necesario añadir información a una relación de asociación. El establecimiento de comunicación en una relación es un role. Un role describe el propósito o capacidad para que una clase se pueda asociar con otra. El nombre de un role se coloca generalmente a lo largo de la línea de asociación que lo define. En el ejemplo del sistema comercial, la clase Trade Administrator (Administrador de mercado o comercio) juega el role de *manager* en la asociación con la clase Trade (Comercio o Mercado).



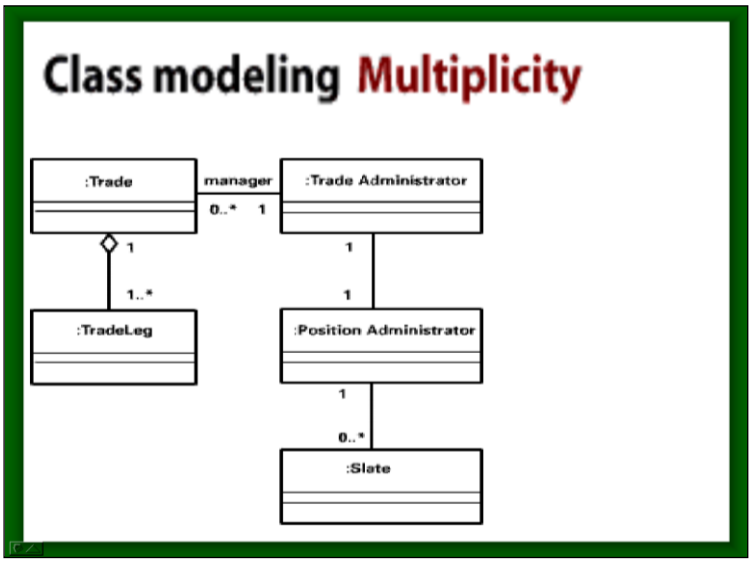
VI.5.3.4. AGREGACIONES

Una agregación es una forma especial de asociación. Las agregaciones ilustran la relación que hay entre un todo y sus partes. Siguiendo con el ejemplo, La clase Trade Leg (Etapa de Comercio) es una parte de la clase Trade (Comercio o Mercado). Esta asociación a final de cuentas es una agregación.



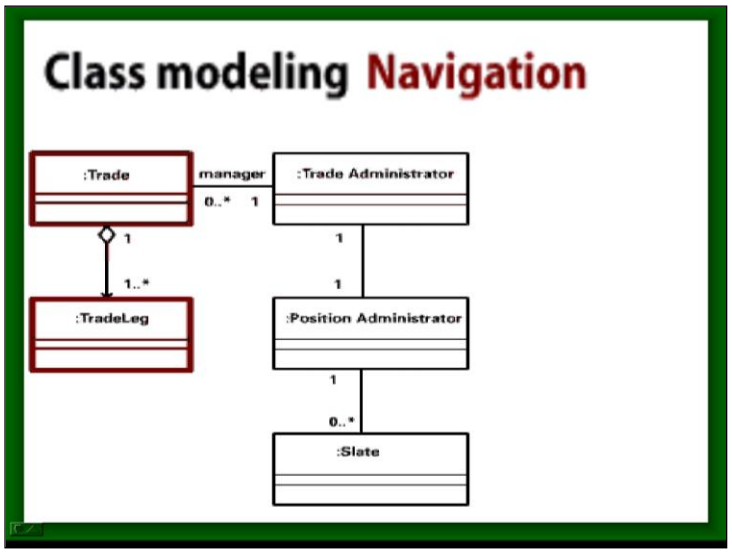
VI.5.3.5. MULTIPLICIDAD

Una vez establecidas las relaciones de asociación y agregación, será necesario conocer cuantos (muchos, varios, uno, etc.) objetos pueden participar en dichas relaciones. La multiplicidad es el número de instancias de una clase que están relacionadas con una instancia de la otra clase de la relación que se define. Para cada asociación y agregación, existe una multiplicidad formada por dos partes: una para cada extremo de la relación que se define. Por ejemplo: En el sistema del comercio, un objeto de la clase Trade Administrator (Administrador de Mercado o Comercio) se relaciona con cero o más objetos de la clase Trade (Comercio o Mercado); y un objeto de la clase Trade es asociado con exactamente un objeto de la clase Trade Administrator (Administrador de Mercado o Comercio). Similarmente un objeto Trade *se compone de* uno o más objetos Trade Leg y un objeto Trade Leg *es parte de* exactamente un objeto Trade.



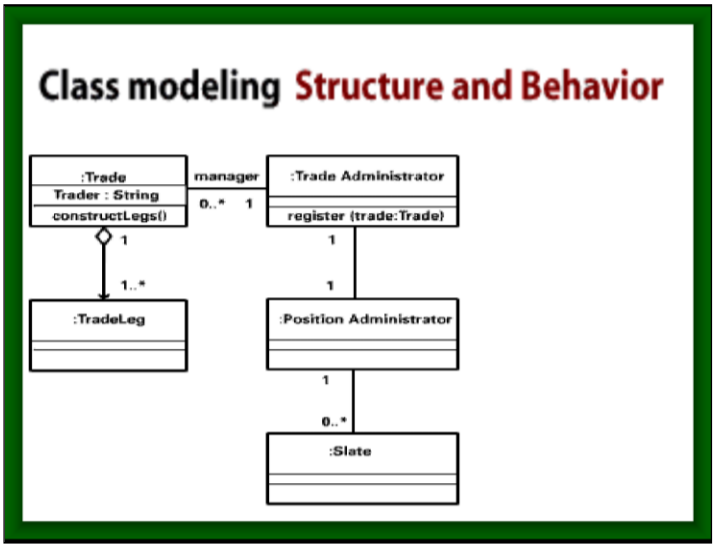
VI.5.3.6. NAVEGACIÓN

Las asociaciones y agregaciones son bidireccionales por default, pero puede ser deseable restringir la navegación o flujo en una sola dirección. Si la navegación se restringe entonces será necesario añadir al esquema gráfico una flecha que indique la dirección de dicha navegación donde corresponda. En nuestro ejemplo, un Trade puede llamar a un Trade Leg pero no a la inversa.



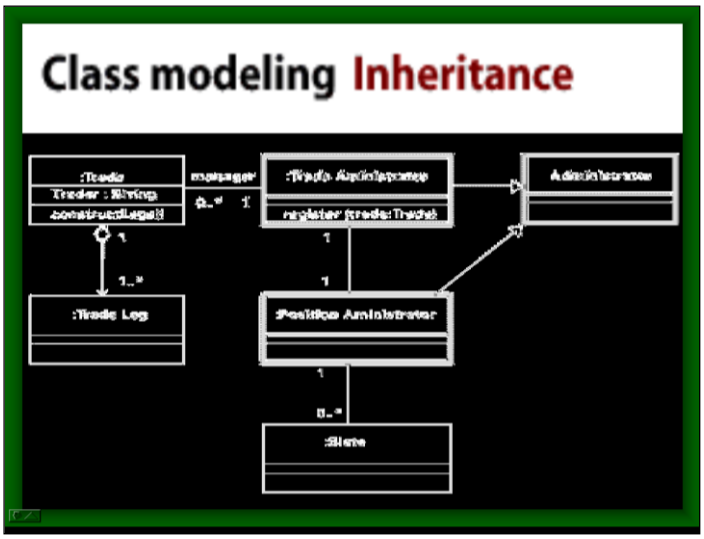
VI.5.3.7. ESTRUCTURA Y COMPORTAMIENTO

Los diagramas de clase representan la estructura y comportamiento de cada clase que los conforman. La estructura de una clase es descrita mediante un conjunto de atributos. Un atributo es una definición de dato a través de un campo o variable que es utilizada a través de instancias de clases. Estos atributos son definidos en el segundo compartimiento de la representación gráfica de una clase en el diagrama de clases.
Por otro lado, el comportamiento de una clase es representado por sus operaciones. Una operación es un servicio que puede ser utilizado por un objeto afectando su comportamiento. Las operaciones son descritas en el tercer compartimiento de la representación gráfica de una clase en el diagrama de clases .
En el ejemplo que hemos estado siguiendo, cada objeto Trade tiene un atributo llamado Trader y una operación llamada constructLeg.



VI.5.3.8. HERENCIA

La herencia es otro tipo de relación. La herencia define una relación de clases donde una clase determinada comparte la estructura y/o comportamiento de una o más clases distintas. Esta relación define una jerarquía de abstracciones en donde una subclase hereda de una o más superclases.
En el ejemplo del sistema comercial un objeto Trade Administrator y un objeto Position Administrator heredan de la clase Administrator. Esta relación por una parte define una estructura común para estos objetos: comportamiento y/o relaciones de nivel Administrator y por la otra define elementos únicos propios de la clase Trade Administrator y de la clase Position Administrator. Con todos estos conceptos vistos podemos entonces visualizar la estructura de la solución de un problema automatizado en un sistema software.



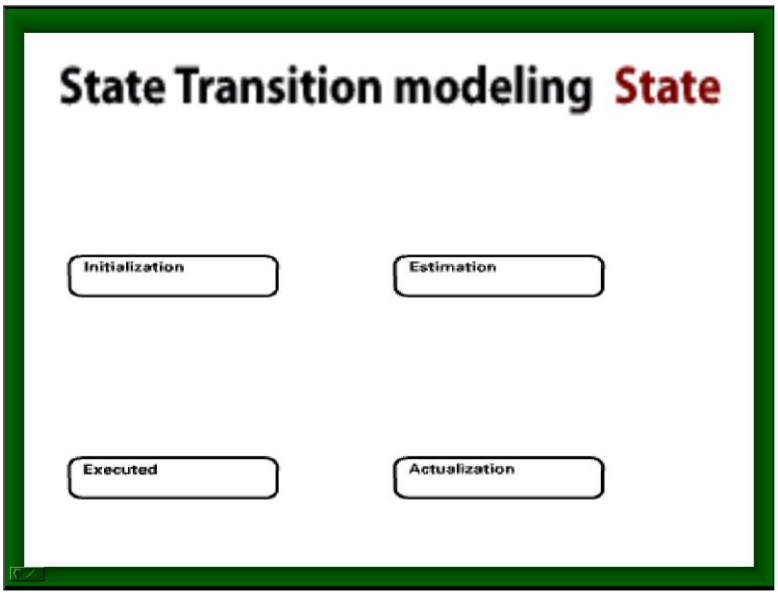
VI.5.4. MODELADO DE TRANSICIÓN DE ESTADOS

En la mayoría de las veces sino es que en todas, es deseable modelar el comportamiento de una determinada clase de objetos, especialmente si la clase demuestra un comportamiento dinámico significativo. Los diagramas de transición de estados son conocidos como diagramas de flujos de estados que pueden ser creados para dichas clases de objetos determinadas. Se puede decir que un diagrama de transición de estados ilustra la historia que puede vivir una clase dada e identifica los eventos que causan su transición de un estado a otro así como las acciones que resultan del cambio de ese estado.

VI.5.4.1. ESTADOS

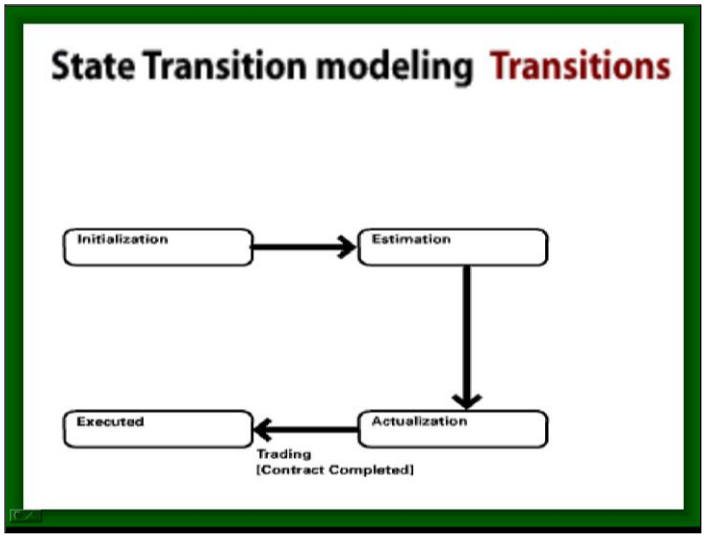
Un estado es una situación durante la vida de un objeto la cual satisface alguna condición requerida, ejecuta alguna actividad deseada o bien espera a que ocurra algún evento posible. En el ejemplo que hemos estado siguiendo, el comportamiento dinámico de un Trade se puede especificar a través de cuatro estados:

- El estado inicialización
- El estado Estimación
- El estado Actualización
- El estado Ejecución



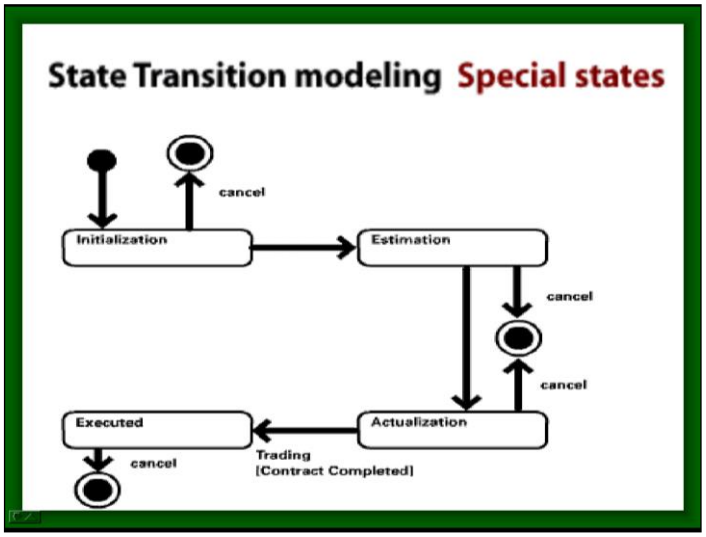
VI.5.4.2. TRANSICIONES

Una transición es un cambio de un estado original a un estado sucesor que ocurre como resultado de algún estímulo. Las transiciones pueden ser automáticas o bien etiquetadas con un evento, condición y/o acción. En el ejemplo del sistema comercial, se tienen algunos tipos de trancisiones, la transición Inicialización-Estimación y Estimación-Actualización, las cuales se dan de manera automática; mientras que la transición Actualización-Ejecución es más complicada y solo se da mediante una condición de guarda que es etiquetada como un evento. Básicamente si un objeto se encuentra en el estado de Actualización y éste recibe un evento “traded” entonces podrá cambiar al estado Ejecución sólo si la condición “contact complete” es verdadera.



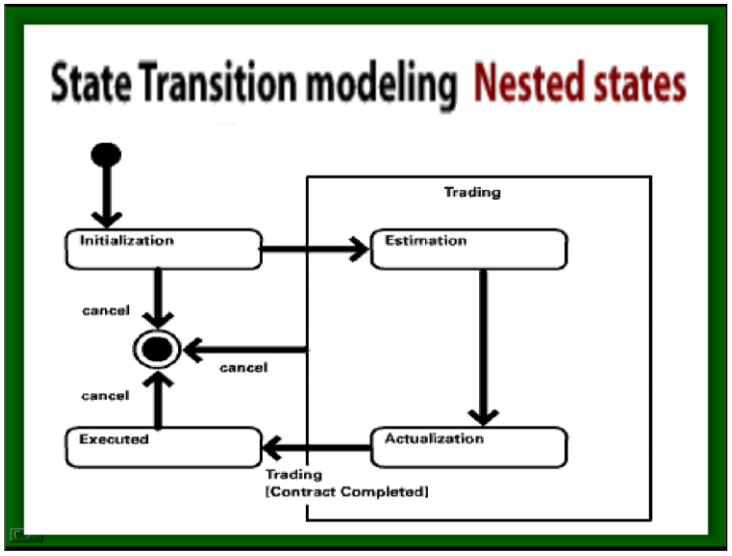
VI.5.4.3. ESTADOS ESPECIALES

Existen dos estados especiales en un diagrama de transición de estados: el estado de INICIO y el estado FINAL o de PARO. Sólo existe un estado inicial o de inicio en un diagrama de transición de estados, pero éste puede tener múltiples estados finales.
En el ejemplo, un Trade principia en el estado de Inicialización cuando éste es creado. El Trade se mueve entonces de un estado a otro durante su periodo de vida, Sin embargo el Trade puede ser cancelado dentro de cualquier estado en el que se encuentre. Si el evento cancelar es recibido, las transiciones del objeto Trade irán al estado final.



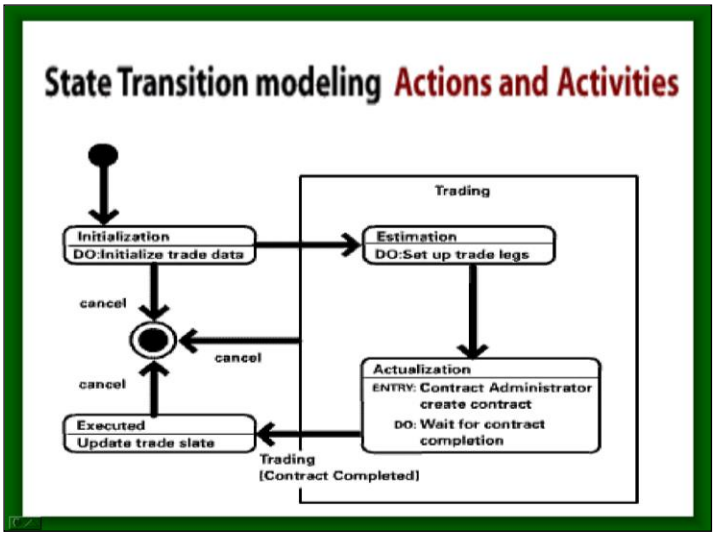
VI.5.4.4. ESTADOS ANIDADOS

Los estados pueden ser anidados. Las clases anidadas simplifican los diagramas complejos. No hay límite en el número de estados anidados en un diagrama. Dentro del ejemplo los estados de Estimación y Actualización se encuentran anidados y son por tanto subestados del estado Trading.



VI.5.4.5. ACCIONES Y ACTIVIDADES

Una actividad es un comportamiento realizado por un objeto mientras éste se encuentra dentro de un estado dado. Una actividad puede ser una acción interna, o puede ser un evento que se envía a otro objeto. En el ejemplo de siempre, la actividad "Update Trade Slate" se lleva a cabo cuando el Trade se encuentra en el estado de Ejecución. La conclusión de esta parte es que los diagramas de transición de estados se utilizan para modelar y visualizar el comportamiento de un sistema software a través de sus clases.

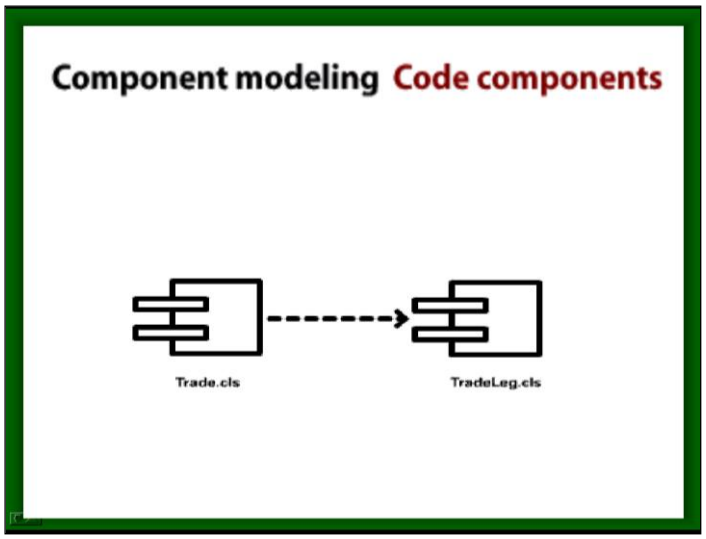


VI.5.5. MODELADO DE COMPONENTES

En esta sección se ilustrará la forma de usar los diagramas de componentes para definir organizaciones y dependencias sobre componentes software (valga la redundancia), incluyendo aquellos que tienen que ver con el código fuente, con los tiempos de ejecución del sistema y con aquellos involucrados en la ejecución del producto.
Los diagramas de componentes proporcionan las instrucciones de modelado necesarias para visualizar la naturaleza física de un producto de programación.

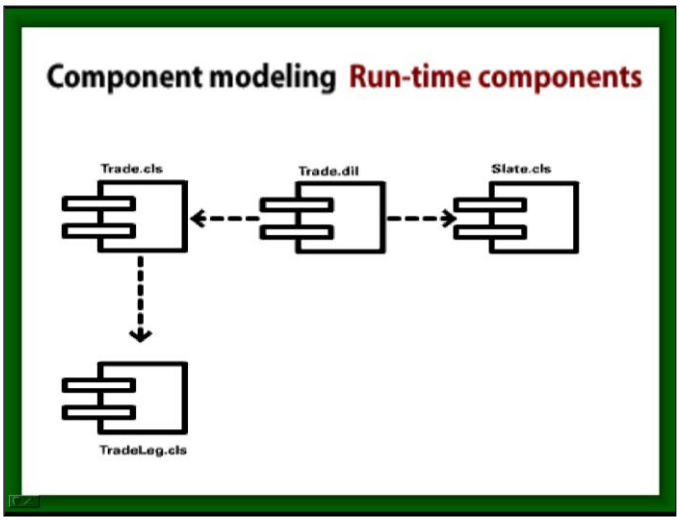
VI.5.5.1. COMPONENTES DE CODIGO

Los componentes de código fuente ilustran las dependencias de compilación entre archivos. Estos tipos de dependencias se definirán en un lenguaje específico.
En el ejemplo, los archivos que representan la clase Trade son dependientes de aquellos que representan a la clase Trade Leg.



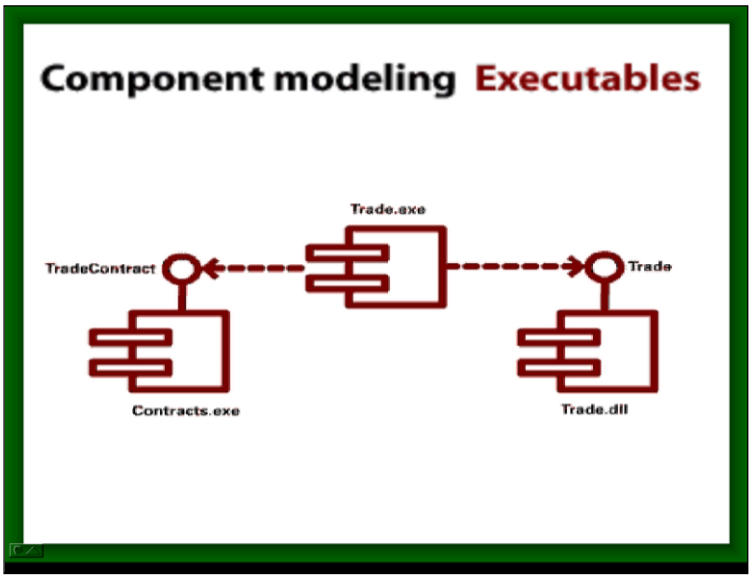
VI.5.5.2. COMPONENTES DE TIEMPO DE EJECUCIÓN

Estos componentes ilustran el mapeo de clases en librerías en tiempo de ejecución (Applets de JAVA, componentes de Active-X y librerías dinámicas). Con estos diagramas ilustramos en nuestro ejemplo que la librería Trade depende de las clases Trade y Slate.



VI.5.5.3. COMPONENTES DE EJECUCIÓN

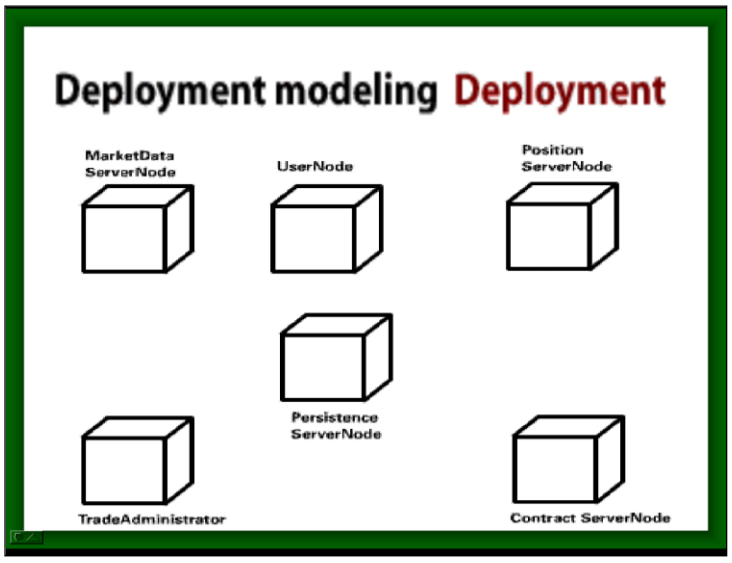
Los componentes de ejecución ilustran las interfaces y el llamado de dependencias en los archivos ejecutables**.** En el ejemplo, el diagrama revela que el ejecutable Trade llama al ejecutable Contacts vía la interface Trade Contact. A su vez la librería Trade es llamada vía la interface Trade.



VI.5.6. MODELADO DE ORGANIZACIÓN

VI.5.6.1. ORGANIZACIÓN

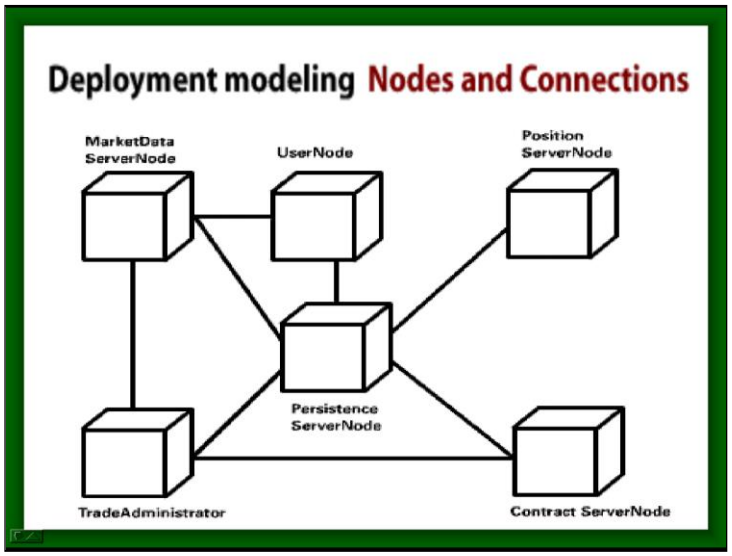
El diagrama de organización ilustra la configuración del procesamiento en tiempo de ejecución de los elementos y la vida de los procesos software. Los diagramas de organización visualizan la distribución de componentes sobre una empresa o negocio.



VI.5.6.2. NODOS Y CONEXIONES

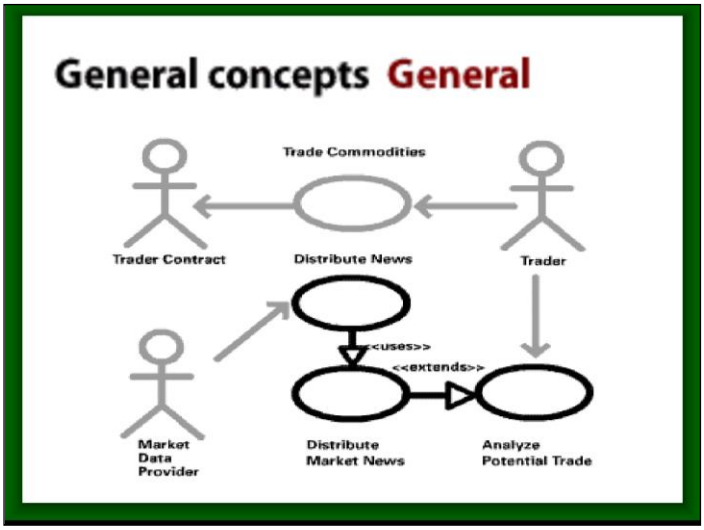
Los elementos del procesamiento en tiempo de ejecución son representados como nodos. Los nodos son conectados mediante asociaciones que indican las rutas de comunicación

entre ellos. Un proceso software se define en estos diagramas como un texto que es añadido a un nodo o grupo de ellos. En el diagrama del ejemplo se muestra que el sistema comercial se organiza a través de 6 nodos distintos. Con ello lo que tratamos de definir es la conectividad que hay entre dichos nodos.



VI.5.7. CONCEPTOS GENERALES

Algunos conceptos generales tales como *estereotipos y paquetes* extienden los diferentes diagramas vistos para soportar funcionalidad. Los *estereotipos* son elementos de modelado que pueden ser utilizados como mecanismos de comunicación adicional. Se pueden utilizar en casos de uso, clases y diagramas de componentes. Por ejemplo, se podrían utilizar las etiquetas “USES” y “EXTENDS” para refinar las asociaciones entre casos de uso. Esto en efecto son propiedades-estereotipos que pueden ser añadidos a los elementos de un modelo simple para ilustrar la especialización deseada.



VI.5.7.1. ESTEREOTIPOS

Un estereotipo es un mecanismo de clasificación. Los estereotipos pueden utilizarse para extender los elementos notacionales del UML, además de poder clasificar y extender asociaciones, relaciones de herencia, clases y componentes. Básicamente se identifican 5 estereotipos:

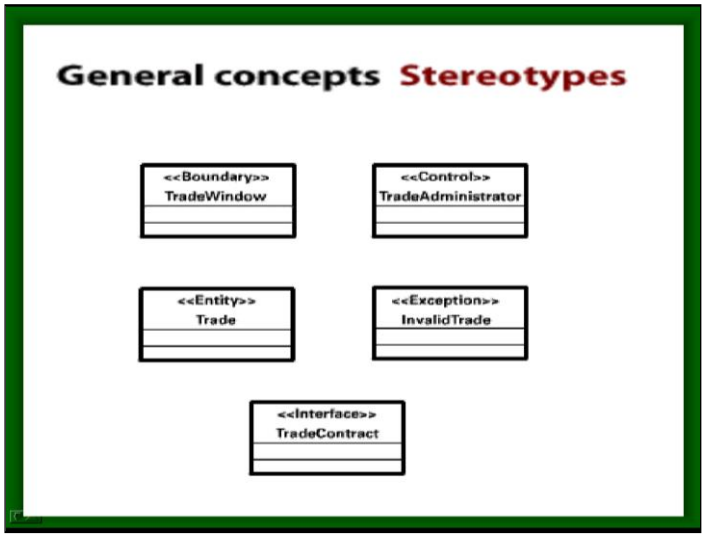
Boundary (límite).- Es una clase que representa una interfaz de la parte externa de un sistema (mundo exterior).

Control.- Clase responsable para información secuencial.

Entidad.- Clase que define una abstracción de alguna entidad del mundo real.

Excepción.- Clase que representa una condición excepcional (error serio).

Interface.- Especificador para las operaciones externas visibles.



VI.5.7.2. PAQUETES

Un paquete es un contenedor de modelado de elementos. En el diagrama de la figurase observan tres paquetes:

- El paquete Domain Trading
- El paquete Services
- El paquete Administration

Cada paquete contiene una colección de casos de uso, una colección de clases, o una colección de componentes. Las flechas punteadas indican las dependencias entre los paquetes.