FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

CURSOS INSTITUCIONALES

BANCO DE MÉXICO

ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS CON UML

Del 02 al 20 de Octubre del 2000

APUNTES GENERALES

Ing. Alexei S. Dezotti Ruiz
Palacio de Minería
Octubre/2000

ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS CON UML

Auditorio y Objetivos

- **Dirigido a**
  - Personas con la necesidad de aprender las caracteristicas y métodos de la tecnologia de objetos, principalmente aquellas que desarrollan sistemas complejos.
- **Objetivos**
  - Al finalizar el curso, los participantes podran:
    - Explicar el proceso de desarrollo iterativo e incremental
    - Definir los requerimientos de un sistema desde el punto de vista del usuario
    - Crear un modelo orientado a objetos del comportamiento y de los aspectos estructurales de los requerimientos de un sistema
    - Crear una arquitectura logica de un sistema
    - Diseñar un sistema aplicando los conceptos de abstraccion, encapsulamiento, herencia, polimorfismo y patrones

Contenido

- Introduccion
  - Antecedentes del analisis y diseño orientado a objetos y UML
- Desarrollo Iterativo e Incremental
  - Ciclo de vida del desarrollo de sistemas por medio de una aproximacion iterativa e incremental
- Comportamiento del Sistema
  - Analisis de requerimientos a través de Casos de Uso (Use case)
  - Desarrollo de escenarios
- Objetos y Clases
  - Definición de objetos, clases, estereotipos y paquetes

La Crisis del Software

- El Departamento de Vehículos a Motor de California invirtió mas de $43 mdd en un sistema que fusionaba los Sistemas de Licencias a Conductores del estado y el Registro de Vehículos
  - El sistema fue abandonado sin haberlo usado
- American Airlines realizo sin exito un esfuerzo de $165 mdd para ligar su software de reservacion de vuelos con los sistemas de reservación de Marriott, Hilton y Budget
- El sistema de control de equipaje del Aeropuerto de Denver costo millones de dólares, sobretodo debido al retraso en la abertura del aeropuerto

Fallas de software reportadas por W. Wayt Gibbs en el numero de Septiembre de 1994 de la revista Scientific American

Algunos ejemplos extremos, PERO hay varios desastres similares en escala menor

La Crisis del Software (cont.)

- En Marzo de 1989, Arthur Andersen reportó
  - Mas de $300 mil mdd por año invertidos en actividades de software comercial en los Estados Unidos
  - Solo el 8% del software entregado brinda resultados y funciona
- ¿Cuáles son las razones de la crisis del software?
  - Constantes cambios en los requerimientos
  - Fallas en el manejo de riesgos
  - Complejidad del software

Cambios en los requerimientos

- Los requerimientos del negocio se definen en ciclos de desarrollo mas pequeños
  - Los usuarios esperan mas en terminos de flexibilidad
- Los requerimientos iniciales generalmente estan pobremente definidos

Fallas en el Manejo de Riesgos

- El ciclo de vida en cascada puede retrasar la identificación del problema
- No hay prueba de que el sistema funcionara, sino hasta el final del ciclo de vida
- El resultado, el maximo riesgo

Complejidad del Software

- Esta creciendo la demanda de software de negocios
- Nadie entiende el sistema en su totalidad
- Deben mantenerse los sistemas anteriores, pero los desarrolladores de los mismos ya se han ido

Poder de la Tecnología de Objetos

- Un paradigma único
  - Los usuarios, analistas, diseñadores e programadores utilizan el mismo lenguaje
- Facilita la re-utilización de arquitectura y código
- Los modelos reflejan de manera más cercana al mundo real
  - Describe con mayor precisión los procesos y datos
  - Descomposicion basada en partición natural
  - Mas facil de entender y mantener
- Estabilidad
  - Un cambio en los requerimientos no significa cambios masivos en el sistema en desarrollo

Ejemplo: Ventas

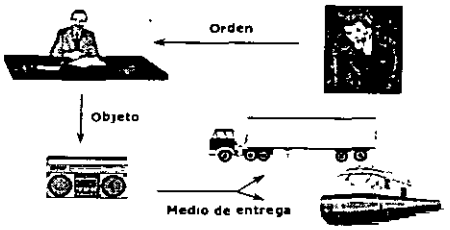
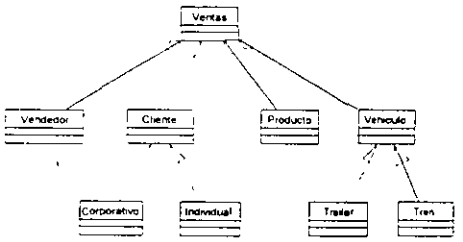
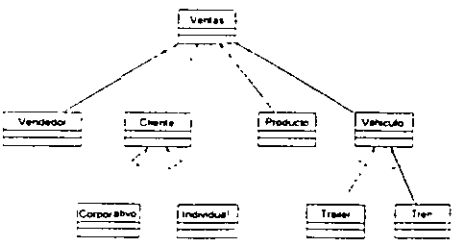


Diagrama de Clases de Ventas

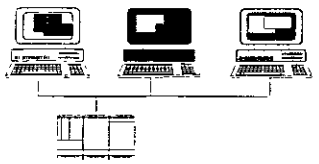


Efecto en el Cambio de Requerimientos



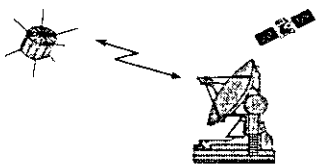
¿Dónde se está usando OO?

- Sistemas basados en GUI
  - La metodología OO facilita el diseño e implementacion de sistemas con interfaz grafica de usuario (GUI)



¿Dónde se está usando OO?

- Sistemas Inmersos
  - Los metodos OO permiten desarrollar sistemas inmersos y de tiempo real con mayor calidad y flexibilidad

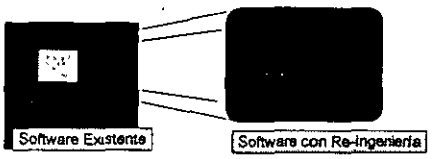


¿Dónde se está usando OO?

- Computo Cliente/Servidor
  - La metodología OO pude encapsular informacion de mainframe en objetos, permitiendo obtener aplicaciones pequeñas

¿Dónde se está usando OO?

- En la Re-ingeniería
  - Los metodos OO permiten hacer re-ingeniería a partes del sistema, protegiendo las inversiones hechas en aplicaciones de software existentes



Análisis y Diseño Orientado a Objetos

OOA → **OOD**

Desarrollo del modelo de requerimientos

Agregar decisiones de diseño y de detalle

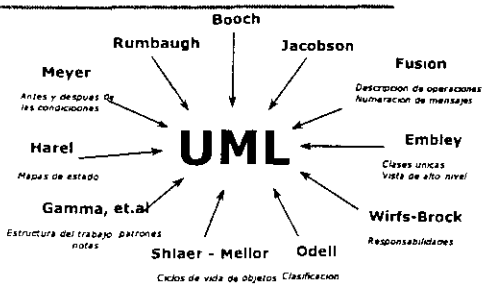
Perspectiva del Usuario

Perspectiva del Desarrollador

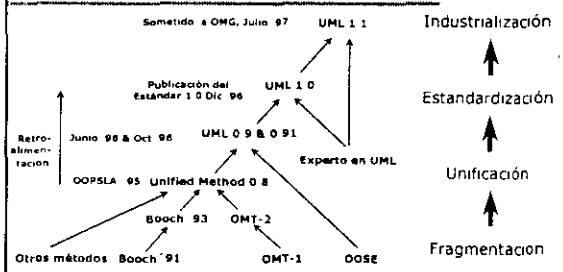
¿Qué es UML?

- El Lenguaje de Modelado Unificado (Unified Modeling Language, UML) es descrito en "The Unified Modeling Language for Object-Oriented Development" escrito por Grady Booch, Jim Rumbaugh, e Ivar Jacobson
  - Disponible en http:\\www rational com
- Basado en las experiencias personales de los autores
- Incorpora contribuciones de otras metodologías
- Sometido aprobación a la OMG por Rational Software, Microsoft, Hewlett-Packard, Oracle, Texas Instruments, MCI Systemhouse y otros

Entradas al UML



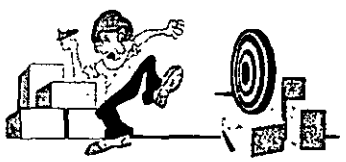
Evolución del UML



Beneficios de UML

- Ofrece un proceso de modelado sin fallas durante el análisis, para diseñar la implementacion
- Define una notacion expresiva y consistente
  - Facilita la comunicacion con otros
  - Ayuda a señalar omisiones e inconsistencias
  - Soporta tanto analisis como diseño de pequeños y grandes sistemas

Desarrollo Iterativo e Incremental



Objetivos: Desarrollo Iterativo e Incremental

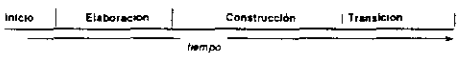
- Usted podra
  - Definir un proceso de desarrollo iterativo e incremental
  - Listar las fases, los productos y las actividades principales para cada fase de un proceso de desarrollo iterativo e incremental
  - Definir una iteración y listar sus actividades

¿Qué es Desarrollo Iterativo e Incremental?

- Desarrollo iterativo e incremental es el proceso de construir sistemas de software en pasos pequeños
- Beneficios
  - Reduccion del riesgo basándose en la retroalimentacion temprana
  - Mayor flexibilidad para acomodar requerimientos nuevos o cambios en los mismos
  - Incremento de la calidad del software

Ciclo de vida del Software

- El ciclo de vida del software se divide en una serie de ciclos de desarrollo, donde la salida de un ciclo de desarrollo es la generación de un producto de software
- Cada ciclo es una sucesion de fases
  - Inicio
  - Elaboracion
  - Construccion
  - Transicion



Fase de Inicio

- Proposito
  - Establecer casos de uso de un sistema nuevo o para la actualizacion importante de un sistema existente
- Productos requeridos
  - Requerimientos esenciales para el proyecto
  - Valoracion del riesgo inicial
- Productos opcionales
  - Un prototipo conceptual
  - Un modelo inicial del dominio (avance de un 10% - 20%)

Fase de Elaboración

- Propósito
  - Analizar el dominio del problema
  - Establecer una base arquitectónica sólida
  - Manejar los elementos de mayor riesgo del proyecto
  - Desarrollar un plan comprensivo que muestre como se completara el proyecto

Fase de Elaboración (cont.)

- Productos
  - Un modelo del comportamiento del sistema, que incluya el contexto del sistema, escenarios y un modelo del dominio (avance de un 80%)
  - Una arquitectura de ejecutables
  - Una visión del producto base de acuerdo al modelo de dominio
  - Una valoracion revisada del riesgo
  - Un plan de desarrollo
  - Criterios de evaluacion
  - Publicar descripciones
  - Un manual de usuario preliminar (opcional)
  - Estrategia de prueba
  - Plan de pruebas

Fase de Construcción

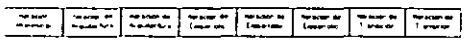
- Proposito
  - Desarrollar un producto de software completo, de forma incremental, que ya este en transicion a la comunidad de usuarios
- Productos
  - Una serie de ejecutables liberados
  - Prototipos de comportamiento
  - Resultados que aseguren calidad
  - Documentacion del sistema y del usuario
  - Plan de desarrollo
  - Criterio de evaluacion para al menos la siguiente iteracion

Fase de Transición

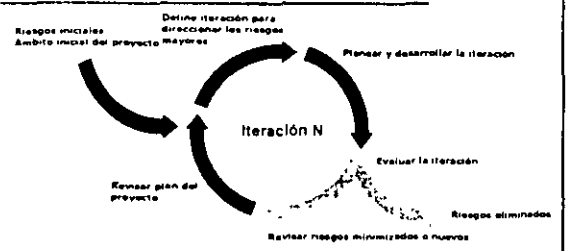
- Propósito
  - Hacer la transición del producto de software a la comunidad de usuario
- Productos
  - Una serie de ejecutables liberados
  - Resultados que aseguren calidad
  - Documentación del sistema y del usuario actualizados
  - Análisis "postmortem" del desempeño del proyecto

¿Qué es una Iteración?

- Una iteración es un loop o ciclo de desarrollo que desemboca en una liberación de un subconjunto del producto final
- Cada iteración pasa a través de todos los aspectos del desarrollo del software
  - Análisis de requerimientos
  - Diseño
  - Implementación
  - Pruebas
  - Documentación
- Cada liberación iterativa es una "pieza" totalmente documentada del sistema final



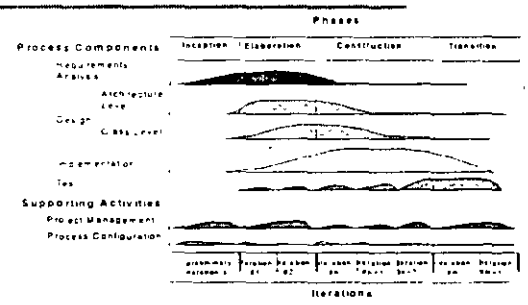
Reducción del Riesgo a través de Iteraciones



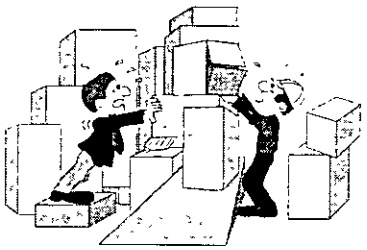
Planeación de Iteraciones

- Identificar y asignar prioridades a los riesgos del proyecto
- Seleccionar un pequeño numero de escenarios que ejemplifiquen los riesgos de mayor prioridad
- Los escenarios seleccionados son utilizados por
  - **Los desarrolladores, para identificar lo que se va a implementar en la iteración**
  - **Los evaluadores, para desarrollar planes y procedimientos de prueba para la iteracion**
- Al final de la iteracion
  - **Determinar los riesgos que han sido reducidos o eliminados**
  - **Determinar la posibilidad de nuevos riesgos descubiertos**
  - **Actualizacion del plan de iteraciones siguientes**

Reunión de todos los elementos



Comportamiento del Sistema



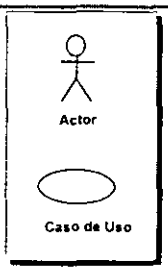
Objetivos: Comportamiento del Sistema

- Usted será capaz de
  - Definir el comportamiento de un sistema
  - Definir los casos de uso y actores
  - Entender como documentar los casos de uso
  - Usar un diagrama de casos de uso para mostrar los actores, casos de uso y sus interacciones
  - Definir escenarios para los casos de uso

¿Qué es el Comportamiento del Sistema?

- El comportamiento del sistema es como este actua y reacciona a su entorno
  - La actividad aparentemente visible y comprobable de un sistema
- El comportamiento del sistema se captura en casos de uso
  - Describen al sistema, su ambiente y las relaciones entre el sistema y su ambiente

Conceptos Importantes en el Modelado de Casos de Uso



- Un actor representa cualquier cosa que interactúa con el sistema
- Un caso de uso es una secuencia de acciones que un sistema desempeña y que produce un resultado observable por un actor

¿Qué es un Modelo de Casos de Uso?

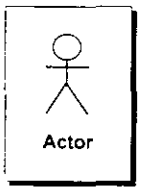
- Un modelo de casos de uso es una representación de las funciones intencionales del sistema (casos de uso) y sus alrededores (actores)
- El mismo modelo de casos de uso se emplea en el análisis de requerimientos, diseño y pruebas

El objetivo principal del modelo de casos de uso es comunicar la funcionalidad y el comportamiento del sistema hacia el cliente o usuario final

Beneficios de un Modelo de Casos de Uso

- El modelo de casos de uso
  - Se utiliza para comunicarse con los usuarios finales y expertos del dominio
    - Proporciona una etapa previa al desarrollo de sistemas
    - Asegura el entendimiento mutuo de los requerimientos
  - Se utiliza para identificar
    - ¿Quien interactuara con el sistema y que debe hacer el sistema?
    - ¿Qué interfaz debe tener el sistema?
  - Se utiliza para verificar
    - Que se capturen todos los requerimientos
    - Que los desarrolladores hayan entendido los requerimientos

Actores

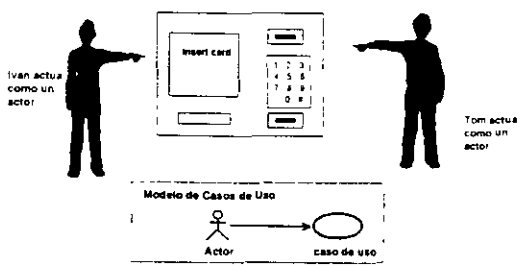


- Los actores no son parte del sistema, representan roles que un usuario del sistema puede ejecutar
- Un actor puede intercambiar información activamente con el sistema
- Un actor puede ser un recipiente pasivo de información
- Un actor puede representar a una persona, a una maquina o a otro sistema

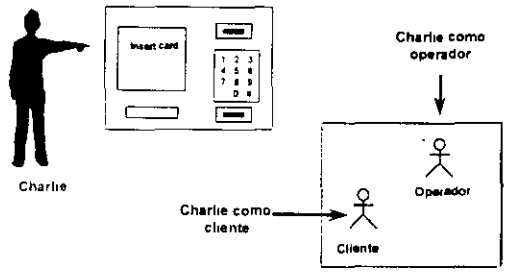
Identificación de Actores: Preguntas Útiles

- ¿Quien está interesado en cierto requerimiento?
- ¿En qué parte de la organización se usará el sistema?
- ¿Quién proveerá al sistema con información, la usará y/o borrará?
- ¿Quien usará esta función?
- ¿Quien le dara soporte y mantenimiento al sistema?
- ¿El sistema usa una fuente externa?
- ¿Que actores necesitan los casos de uso?
- ¿Puede un actor desempeñar roles diferentes?
- ¿Varios actores desempeñan el mismo rol?

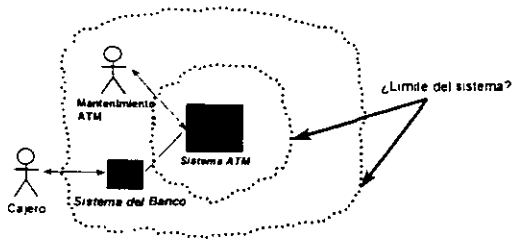
Instancias de Actores



Un usuario puede actuar como varios actores



Límites de los actores y el sistema



Casos de Uso

- Un caso de uso modela un diálogo entre actores y el sistema
- Un actor inicia un caso de uso para invocar cierta funcionalidad del sistema
- Un caso de uso es un flujo de eventos completo y significativo
- El conjunto de todos los casos de uso, representa todas las formas posibles de uso del sistema

Identificación de Casos de Uso: Preguntas Útiles

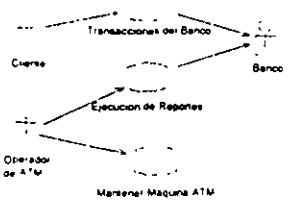
- ¿Cuales son las tareas que realiza este actor?
- ¿El actor creará, almacenara, cambiara, borrará o leerá información en el sistema?
- ¿Qué caso de uso creará, almacenará, cambiará, borrará o leerá esta información?
- ¿Necesitará el actor informar al sistema sobre cambios externos repentinos?
- ¿Necesitara el actor recibir información en relación a ciertas ocurrencias en el sistema?
- ¿El sistema proporciona al negocio el comportamiento correcto?
- ¿Que casos de uso van a darle soporte y mantenimiento al sistema?
- ¿Pueden todos los requerimientos funcionales ser ejecutados por los casos de uso?

Fuentes de Información para los Casos de Uso

- Declaracion de especificaciones del sistema
- Definicion del problema a resolver
- Literatura relevante al dominio
- Entrevistas con expertos del dominio
- Conocimiento personal del dominio o experiencia
- Sistemas Anteriores o Legados

Diagrama Casos de Uso

- Se dibuja un ***diagrama de casos de uso*** para ilustrar los casos de uso y los actores que interactuan enviandose estimulos el uno al otro

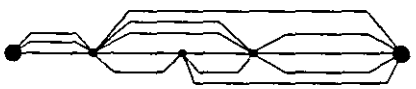


Documentación de un Caso Uso

- Los casos de usos se documentan con:
  - Una breve descripción
    - **Se expone el propósito del caso de uso en unas cuantas líneas**
  - Flujo de eventos detallado
    - **Descripción del flujo primario y los flujos alternos de eventos que ocurren desde el inicio el casos de uso**
  - La documentacion debe leerse como un diálogo entre el actor y el caso de uso
- Ambas partes de la documentación deben estar escritos en terminos que el cliente entienda

Flujo de Eventos en un Caso de Uso

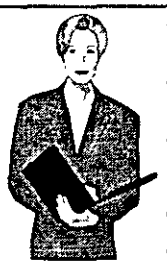
- Cada caso de uso
  - Tiene una secuencia de transacciones normal o básica
  - Debe tener varias secuencias alternativas de transacciones
  - Generalmente tiene secuencias de excepcion a transacciones que manejan situaciones erróneas
  - Tambien debe tener pre y post condiciones bien definidas



Flujo de Eventos en un Caso de Uso (cont.)

- Describe sólo los eventos que pertenecen al caso de uso, y no lo que ocurren en otros casos de uso
- Evitar el uso de terminologia vaga como: "por ejemplo", "etc " e "informacion"
- El flujo de eventos debera describir:
  - ¿Cómo y cuando inicia y termina el caso de uso?
  - ¿Cuándo interactua el caso de uso con los actores?
  - ¿Que informacion se intercambia entre un actor y el caso de uso?
  - No describe los detalles de la interfaz de usuario
  - Describe el flujo básico de eventos
  - Cualquier flujo de eventos alterno

¿Quién lee la documentación asociada a los Casos de Uso?



- Clientes: aprueban lo que el sistema debe hacer
- Usuarios: ganan entendimiento del sistema
- Desarrolladores: documento de comportamiento del sistema
- Examinadores: examinan el flujo de eventos
- Analistas o Diseñadores: proporciona las bases para el análisis y diseño
- Evaluador: se usa como base para la prueba de requerimientos
- Líder de Proyecto: proporciona elementos para la planeación de proyectos
- Escritor Técnico: base para la escritura de la guía de usuario

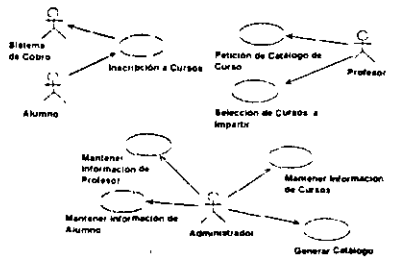
Ejemplo: Inscripción a Cursos

- Al inicio de cada semestre, los alumnos solicitan un catálogo de que contiene la lista de los cursos que se impartiran en el semestre, en el cual se incluyen tambien datos relacionados como profesor, departamento y pre-requisitos
- El sistema nuevo deberá permitir que los alumnos seleccionen cuatro cursos para el semestre que inicia Ademas, cada alumno indicara dos cursos alternativos en caso de que no pueda ser asignada la primera seleccion Los nuevos cursos tendran un maximo de diez alumnos y un minimo de tres Un curso con menos de tres alumnos sera cancelado Una vez que el proceso de inscripcion se ha completado para un alumno, el sistema de registro envía la informacion al sistema de cobros, para que el alumno pueda pagar por el semestre

Ejemplo: Inscripción a Cursos (cont.)

- Los profesores deben ser capaces de ingresar al sistema para indicar que cursos van a impartir Tambien podrán ver que alumnos están inscritos en sus cursos
- Para cada semestre, hay un periodo en el que los alumnos pueden cambiar su horario Los alumnos deben ser capaces de ingresar al sistema durante este tiempo para agregar o cancelar cursos

Diagrama de casos de uso



1. Breve Descripción: Caso de Uso Inscripción a Cursos

1.1 Breve Descripción

- Este caso de uso es iniciado por un alumno. Proporciona la capacidad para que un alumno cree, borre, modifique y/o revise un horario de curso para un semestre dado

2. Flujo de Eventos: Casos de Uso Inscripción a Cursos

2.1 Pre-condiciones
Ninguna

2.2 Flujo Principal
Este casos de uso inicia cuando un alumno introduce el numero de id de alumno. El sistema verifica que el numero de id de alumno sea valido (E-1) y permite que el alumno seleccione el semestre actual o uno futuro (E-2). El sistema permite que el alumno seleccione la actividad deseada **Crear, Revisar, Modificar, Imprimir, Borrar** o **Salir**

Si la actividad seleccionada es:
A-1 Crear. Subflujo Crear un Horario Nuevo
A-2 Revisar. Subflujo Revisar un Horario
A-3 Modificar. Subflujo Modificar un Horario
A-4 Imprimir. Subflujo Imprimir un Horario
A-5 Borrar. Subflujo Borrar un Horario
Salir. termina el caso de uso

2. Flujo de Eventos: Casos de Uso Inscripción a Cursos (cont.)

2.3 Flujos Alternos

A-1 Crear: Subflujo Crear un Horario Nuevo:
El sistema despliega una pantalla de horario en blanco. Los alumnos introducen los 4 cursos primarios y 2 cursos alternativos (E-3). El alumno envía su requerimiento de cursos. Por cada curso primario seleccionado el sistema revisa que se satisfagan los pre-requisitos (E-4) y agrega al alumno al curso si éste está abierto (E-5). El sistema imprime el horario del alumno (E-6) y envía esta información al sistema de cobros para procesarla (E-7). El Caso de Uso inicia de nuevo.

A-2 Revisar: Subflujo Revisar un Horario:
El sistema recupera (E-8) y despliega la información para todos los cursos a los cuales el alumno se registró: nombre del curso, numero del curso, numero de lugares del curso, días de la semana, hora, lugar y numero de horas necesarias. Cuando el alumno indica que el o ella ha terminado su revisión, el Caso de Uso inicia de nuevo.

2. Flujo de Eventos: Casos de Uso Inscripción a Cursos (cont.)

A-3 Modificar: Subflujo Modificar un Horario:
El sistema revisa que la fecha límite para los cambios no haya expirado (E-9). El sistema recupera (E-8) y despliega la siguiente información para todos los cursos a los cuales el alumno se inscribió: nombre del curso, numero del curso, numero de lugares del curso, días de la semana, hora, lugar y numero de horas necesarias. El sistema permite que el alumno seleccione la actividad que deseada. Borrar Curso, Agregar Curso o Salir

Si la actividad seleccionada es
A-6 Borrar Curso: Subflujo Borrar un Curso.
A-7 Agregar Curso: Subflujo Agregar un Curso.
Salir, el sistema imprime el horario del alumno (E-6) y el Caso de Uso inicia de nuevo

2. Flujo de Eventos: Casos de Uso Inscripción a Cursos (cont.)

A-4 Imprimir Subflujo Imprimir un Horario
El sistema imprime el horario del alumno (E-6) El Caso de Uso inicia de nuevo

A-5 Borrar: Subflujo Borrar un Horario
El sistema recupera (E-8) y despliega la información actual del horario. El sistema pide al usuario que confirme la eliminación del horario. Si se confirma, se borra el horario del sistema Si la eliminación no se confirma, la operación se cancela y el Caso de Uso inicia de nuevo

A-6 Borrar Curso. Subflujo Borrar un Curso:
El alumno introduce el numero del curso para borrarlo. El sistema pide al usuario que confirme la eliminación del curso. Si se confirma, se borra el curso del horario del alumno. Si la eliminación no se confirma, la operación se cancela y el caso de uso inicia de nuevo

2. Flujo de Eventos: Casos de Uso Inscripción a Cursos (cont.)

A-7 Agregar Curso: Subflujo Agregar un Curso:
El alumno introduce el número de curso para agregarlo El sistema revisa que se satisfagan los pre-requisitos (E-4) y agrega el alumno al grupo si el curso esta abierto (E-5). El caso de uso inicia de nuevo.

Flujos de Excepción

E-1. Se introdujo un número id de alumno inválido El usuario puede re-introducir el numero id o finalizar el caso de uso

E-2: Se introdujo un semestre inválido. El usuario puede re-introducir el semestre o finalizar el casos de uso.

E-3: El numero de curso no es válido. El usuario puede re-introducir el numero válido o finalizar el caso de uso.

E-4: Los pre-requisitos no fueron satisfechos por el usuario Se le informa al alumno que un curso no puede ponerse en el horario y el motivo De ser posible, se sustituye con un curso alterno. El caso de uso continua

2. Flujo de Eventos: Casos de Uso Inscripción a Cursos (cont.)

E-5: E curso seleccionado esta cerrado De ser posible, se substituye con un curso alterno El caso de uso continua.

E-6. No es posible imprimir el horario. La información se guarda y se le informa al usuario que la petición de impresión del horario debe ser reintroducida. El caso de uso continua.

E-7: No es posible enviar la información al sistema de cobro. El sistema guarda toda la información de cobro y la re-envia despues al sistema de cobros. El caso de uso continua

E-8: El sistema no puede recuperar la información del horario. El caso de uso inicia nuevamente

E-9. El sistema informa al usuario que no se puede modificar un horario El caso de uso inicia nuevamente.

¿Qué son los escenarios?

- Un escenario es una instancia de un casos de uso
  - Es un flujo determinado de eventos en un caso de uso
- Cada caso de uso tiene multiples escenarios
  - Escenarios primarios (happy path scenarios)
    - **Flujo normal, la forma en la que debe trabajar el sistema**
  - Escenarios secundarios
    - **Excepciones del escenario primarios**

Escenario para el Caso de Uso Inscripción a Cursos

- John introduce el numero id de alumno 369 52 3449 y el sistema lo valida. El sistema pregunta que a semestre desea inscribirse John le indica al sistema el semestre actual y elige crear un horario nuevo
- De una lista de cursos disponibles, John selecciona los siguientes 4 cursos primarios: English 101, Geology 110, World History 200, y College Algebra 110. Después selecciona 2 cursos alternos. Music Theory 110 y Introduction to Java Programming 180.
- El sistema determina que John tiene todos los pre-requisitos necesarios y lo agrega a las listas de los cursos correspondientes.
- El sistema indica que la actividad esta completa El sistema imprime el horario del alumno y envia la información de cobro de cuatro cursos primarios al sistema de cobros para que sea procesado

Escenarios Secundarios

- ¿Qué escenarios secundarios podrían considerarse para el caso de uso: "Inscripción a Cursos"?

Escenarios Secundarios (cont.)

- Algunos escenarios secundarios a considerarse:
  - El alumno no seleccionó los 4 cursos primarios
  - Algún curso primario no esta disponible
  - Los cursos primarios y secundarios no están disponibles
  - No se puede agregar el alumno a la lista de un curso
  - No se puede crear el horario del alumno

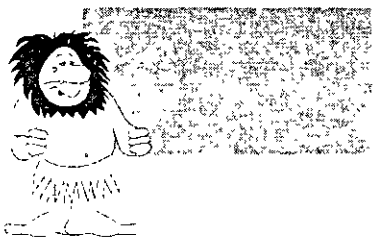
¿Cuántos escenarios se necesitan?

- Respuesta sencilla: tantos como se necesiten para entender el desarrollo del sistema
- Sugerencia:
  - Escenarios Primarios
    - Dedicar aproximadamente el 80% del tiempo a estos escenarios
  - Escenarios Secundarios
    - Elaborar algunos de los escenarios secundarios interesantes y de alto riesgo

Ejercicio: Comportamiento del Sistema

- Utilice el problema que proporciona el instructor
  - Dibujar un diagrama de casos de uso
  - Escribir una definición para cada actor
  - Para un caso de uso
    - Escribir una breve descripción (dos sentencias máximo)
    - Escribir el flujo de eventos
    - Listar algunos escenarios posibles

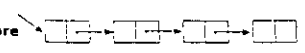
Objetos y Clases



Objetivos: Objetos y Clases

- **Usted podrá:**
  - Definir y dar ejemplos de objetos
  - Definir y dar ejemplos de clases
  - Describir las relaciones entre clases y objetos
  - Definir estereotipos

¿Qué es un objeto?

- **Informalmente, un objeto representa una entidad, ya sea física, conceptual o software**
  - Entidad física 
  - Entidad conceptual 
  - Entidad de software 

Una definición más formal

- Un objeto es un concepto, abstracción, o cosa con límites bien definidos y significado para una aplicación
- Un objeto es algo que tiene:
  - Estado
  - Comportamiento
  - Identidad

Un Objeto tiene Estado

- El estado de un objeto es una de las condiciones posibles en las que un objeto puede existir
- El estado de un objeto normalmente cambia con el paso del tiempo
- El estado de un objeto generalmente se implementa por una serie de propiedades (llamadas atributos), con los valores de las propiedades, más las relaciones que el objeto pueda tener con otros objetos

Profesora Clark

Nombre	Joyce Clark
Empleado ID	867138
Fecha de contrato	March 21, 1887
Estado	Tenured

Un Objeto tiene Comportamiento

- El comportamiento determina como actua y responde un objeto
- El comportamiento define como responde un objeto a peticiones de otros objetos
- El comportamiento visible de un objeto se modela por un conjunto de mensajes a los que puede responder (las operaciones que el objeto puede desempeñar)

Un Objeto tiene Identidad

- **Cada objeto tiene identidad única, aún si el estado es idéntico al de otro objeto**

Profesor "J Clark" imparte Algebra

Profesor "J Clark" imparte Algebra

Profesor "J Clark" imparte Algebra

¿Qué son las clases?

- Hay varios objetos identificados en cualquier dominio
- Una clase es una descripción de un grupo de objetos con propiedades comunes (atributos), comportamiento común (operaciones), relaciones comunes con otros objetos (asociaciones y agregaciones) y semánticas comunes
  - Un objeto es una instancia de una clase
- Una clase es una abstracción en la que ella:
  - Enfatiza características relevantes
  - Suprime otras características
- La abstracción nos ayuda a manejar la complejidad

Ejemplo de Clase

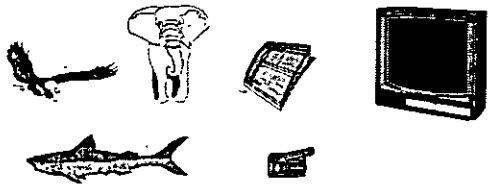
Clase
Curso

Estructura
Nombre
Ubicación
Días ofrecidos
Horas de créditos
Hora inicio
Hora termino

Comportamiento
Agregar un alumno
Borrar un alumno
Obtener catalogo de cursos
Determinar si esta lleno

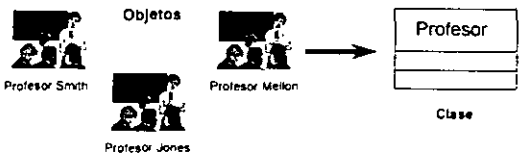
Clases de Objetos

- ¿Cuántas clases ve?



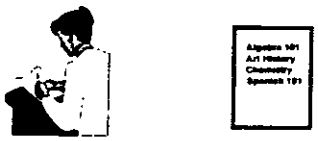
Relaciones entre Clases y Objetos

- Una clase es una definición abstracta de un objeto
  - Define la estructura y comportamiento de cada objeto en la clase
  - Sirve como una plantilla para crear objetos
- Los objetos pueden agruparse en clases



Guía para identificar Clases

- Una clase debe capturar una y solo una llave de abstraccion
- Mala abstraccion la clase Alumno sabe la informacion del alumno y su horario para el semestre actual
- Buena abstraccion Separar las clases en una para alumno y otra para Horario del alumno



¿Cómo nombrar a una Clase?

- **El nombre de una clase debe ser un nombre en singular que caracterice de la mejor forma a la abstracción**
- **La dificultad al nombrar a una clase puede indicar que una abstracción está pobremente definida**
- **Los nombres deben venir directamente del vocabulario del dominio**

Guía de estilo para nombrar Clases

- Una guia de estilo debe dictar convenciones de nombres para clases
- Ejemplo de Guia de Estilo
  - Las clases se nombran usando sustantivos en singular
  - Los nombres de clases empiezan con una letra mayuscula
  - No se usan palabras subrayadas
    - **Los nombres compuestos de palabras multiples se ponen juntas y la primera letra de cada palabra adicional se escribe en mayúscula**
  - Ejemplo: Alumno, Profesor, SistemaCobro

Definición de Semántica de Clases

- Despues de nombrar una clase, se debe hacer una breve y concisa descripcion de la clase
  - Enfocarse en el proposito de la clase y no en la implementacion
- El nombre de la clase y la descripcion forman la base de un diccionario del modelo

Busque el "QUÉ" y no el "CÓMO"

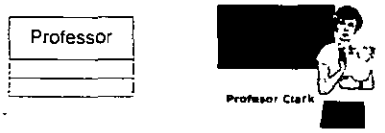
Ejemplo de Entradas al Diccionario del Modelo

- Nombre: StudentInformation
  - Definicion Informacion relacionada a una persona registrada para asistir a clases en la Universidad
- Nombre. Course
  - Definicion de Trabajo. Una clase ofrecida por la Universidad

Al ir estudiando más el problema, se descubren clases y se mejoran las definiciones de las anteriores, agregándolas al diccionario del modelo

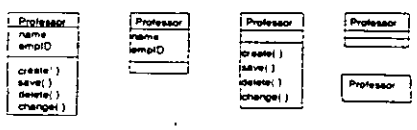
Representación de Clases

- Una clase se representa usando un rectangulo con tres divisiones



Divisiones de Clase

- Una clase comprende tres secciones
  - La primera sección contiene el nombre de la clase
  - La segunda sección muestra la estructura (atributos)
  - La tercera sección muestra el comportamiento (operaciones)
- La segunda y tercera sección pueden suprimirse si no es necesario que sean visibles en el diagrama

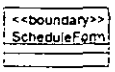


Estereotipos

- Un estereotipo es un nuevo tipo de elemento de modelado que extiende la semantica del metamodelo
  - **Deben estar basados en tipos o clases existentes en el metamodelo**
- Cada clase puede tener como maximo un estereotipo
- Estereotipos comunes
  - **Clase Boundary**
  - **Clase Entity**
  - **Clase Control**
  - **Clase Exception**
  - **Metaclase**
  - **Clase Utility**
- Los Estereotipos se muestran en la parte donde se escribe el nombre de la clase entre << >>

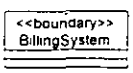
Clases Boundary

- Una clase boundary modela la comunicacion entre los alrededores del sistema y sus funciones internas
- Clases boundary tipicas
  - **Windows (interfaz de usuario)**
  - **Protocolo de Comunicación (interfaz del sistema)**
  - **Interfaz de impresora**
  - **Sensores**
- En el escenario de "Inscripción a Cursos", se crea una pantalla de horario (ScheduleForm) para aceptar información del usuario



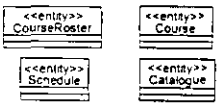
Interfaz con otros sistemas

- Una clase boundary se usa tambien para modelar una interfaz con otro sistema
- Las caracteristicas importantes de este tipo de clases son
  - **La información que va a pasarse al otro sistema**
  - **El protocolo de comunicación que se use para "hablar" con el otro sistema**
- En el escenario "Inscripcion a Cursos", la información debe enviarse al sistema de cobros (BillingSystem)
  - **Se crea una clase llamada BillingSystem para mantener la interfaz del sistema externo**



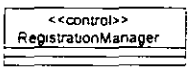
Clases Entity

- Una clase entity modela información y comportamiento asociado que es generalmente de larga vida (persistente)
  - **Puede reflejar un fenómeno de la vida real**
  - **También puede necesitarse para las tareas internas del sistema**
  - **Los valores de sus atributos son proporcionados generalmente por un actor**
  - **Su comportamiento es independiente de los alrededores**
- Clases entity en el caso de uso "Inscripcion a Cursos"
  - **Course**
  - **Schedule**
  - **Catalogue**
  - **CourseRoster**

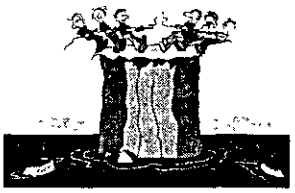


Clases Control

- Una clase control modela el comportamiento de control específico a uno o más casos de uso
- Una clase control
  - **Crea, inicia y borra objetos controlados**
  - **Controla la secuencia o coordinación de ejecución de objetos controlados**
  - **Controla elementos actuales para clases controladas**
  - **Es, la mayor parte de las veces, la implementación de un objeto intangible**
- En el escenario "Inscripción a Cursos", la clase RegistrationManager controla el proceso de inscripcion



Interacción de Objeto



Objetivos: Interacción de Objeto

- **Usted podrá:**
  - **Usar diagramas de secuencia y colaboración para mostrar las interacciones de los objetos**

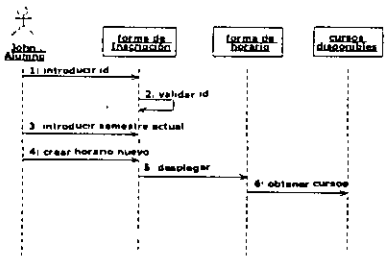
¿Qué es un Diagrama de Interacción?

- Un diagrama de interaccion es una representacion grafica de las interacciones entre objetos
- Hay dos tipos de diagramas de interaccion
  - Diagramas de Secuencias
    - **Un diagrama de secuencias están ordenado de acuerdo al tiempo**
  - Diagramas de Colaboración
    - **Un diagrama de colaboración incluyen el flujo de datos**
  - Cada uno provee un punto de vista diferente de la misma interaccion

¿Qué es un Diagrama de Secuencias?

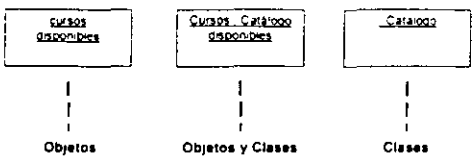
- Un diagrama de secuencia muestra interacciones de objetos ordenados en el tiempo
- El diagrama muestra
  - Los objetos que participan en la interaccion
  - La secuencia de mensajes intercambiados
- Un diagrama de secuencias contiene:
  - Objetos con sus "lineas de vida"
  - Mensajes intercambiados entre objetos en orden secuencial
  - *Enfoque de control (Focus of Control) (opcional)*

Un Diagrama de Secuencias



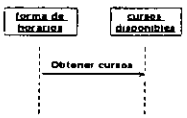
¿Cómo nombrar a los Objetos en un Diagrama de Secuencias?

- Los objetos se dibujan como rectangulos con los nombres subrayados
- Las "lineas de vida" se representan con lineas de guiones descendentes



Mostrar las interacciones entre objetos

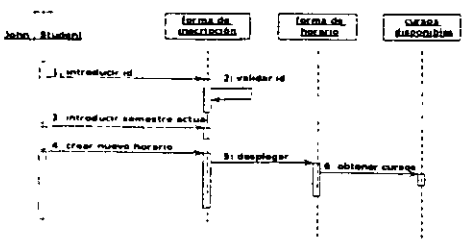
- La interaccion de objetos se indica con flechas horizontales que se dirigen desde la linea vertical que representa al objeto cliente hasta la linea que representa al objeto proveedor
- Las flechas horizontales se etiquetan con un mensaje
- El orden en que ocurren los mensajes es indicado por la posicion vertical, con el más cercano en la parte superior
- La numeracion es opcional, ya que la orden se basa en la posicion vertical



¿Qué es el Enfoque de Control?

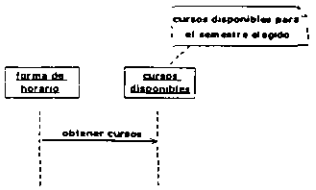
- El Enfoque de Control representa el tiempo relativo en el que el flujo de control se enfoca sobre un objeto
  - Representa el tiempo en que un objeto dirige sus mensajes
- El Enfoque de Control puede mostrarse en un diagrama de secuencia

Enfoque de Control



Notas

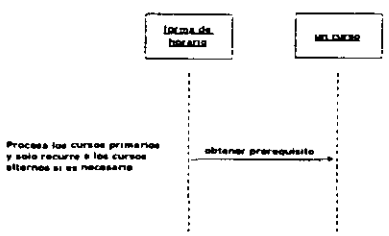
- Las notas pueden agregarse para agregar más información al diagrama



Scripts en Diagramas de Secuencias

- Para escenarios complejos, los diagramas de secuencias pueden ser mejorados mediante el uso de scripts
- Un script se escribe a la izquierda de un diagrama de secuencia con la secuencia de pasos alineados a las interacciones del objeto
- Los scripts se pueden escribir en lenguaje natural o en pseudo codigo

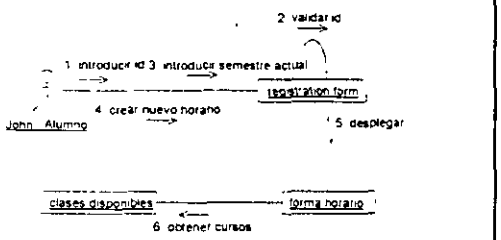
Ejemplo de Script



Diagramas de Colaboración

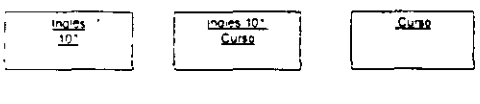
- Un diagrama de colaboración es una forma alternativa de representar el intercambio de mensajes de un conjunto de objetos
- El diagrama muestra las interacciones organizadas entorno a los objetos y a sus relaciones
- Un diagrama de colaboración contiene:
  - Objetos
  - Ligas entre objetos (relaciones)
  - Mensajes intercambiados entre objetos
  - Flujo de datos entre objetos, si hay alguno

Ejemplo de un Diagrama de Colaboración



Representación de Objetos en un Diagrama de Colaboración

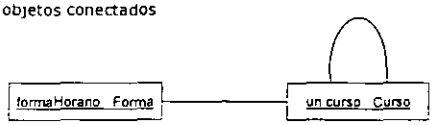
- Los objetos se dibujan como rectángulos con nombres subrayados



Objetos **Objetos y Clases** **Clases**

Representación de Ligas en un Diagrama de Colaboración

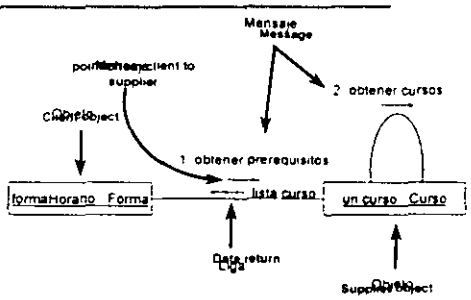
- Una liga de interacción en un diagrama de colaboración se representa como una línea que conecta iconos de objetos
- Una liga indica que hay una ruta de comunicación entre objetos conectados



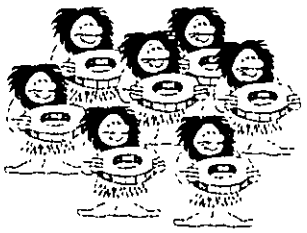
Anotaciones de Liga

- Una liga de interacción en un diagrama de colaboración se puede anotar con:
  - Una flecha apuntando del objeto cliente al objeto proveedor
  - El nombre del mensaje con una lista opcional de parametros y/o valores de retorno
  - Un numero opcional que muestre el orden relativo en el cual se envían los mensajes

Notación de Liga



Identificación de Clases



Objetivos: Identificación de Clases

- Usted podrá:
  - Discutir el análisis de casos de usos
  - Identificar objetos y clases llevando a cabo el analisis casos de usos
  - Usar tarjetas CRC para descubrir clases
  - Diagramar un escenario usando diagramas de interaccion
  - Crear paquetes
  - Crear diagramas de clase iniciales

¿Qué es un Análisis Casos de Uso?

- El análisis casos de uso es el proceso de estudiar los casos de usos para descubrir objetos y clases para el desarrollo del sistema
  - Los escenarios se detallan y se representan gráficamente en diagramas de interacciones
    - **Se crean clases entity, boundary y control**
  - Las clases se agrupan en paquetes
  - Se crean diagramas de clases

Identificación de Objetos Entity

- Los objetos entity se identifican al examinar los sustantivos en los escenarios
- Los sustantivos encontrados pueden ser
  - Objetos
  - Descripcion del estado de un objeto
  - Entidades externas y/o Actores
  - Otros

Filtrado de Sustantivos

- Cuando se identifican sustantivos, debe estar consciente de que:
  - Varios términos se pueden referir al mismo objeto
  - Un término se puede referir a más de un objeto
  - El lenguaje natural es muy ambiguo
- Este acercamiento puede identificar muchos objetos sin importancia
  - La lista de sustantivos debe filtrarse

Observar los Sustantivos

- El siguiente expresion fue escrita para un sistema de bancario
  - "Los requerimientos legales se tomaran en cuenta"
- Si SOLO se consideraran los sustantivos, ¿qué pasaria?

Línea principal: Cada sustantivo debe considerarse en el contexto del dominio del problema -- no puede considerarse por sí mismo

Escenario: "Crear horario"

- John introduce el número id de alumno 369 52 3449 y el sistema valida el número. El sistema pregunta que semestre. John indica el semestre actual y elige crear un horario
- De una lista de cursos disponibles, John selecciona los cuatro cursos primarios English 101, Geology 110, World History 200, y College Algebra 110. Despues selecciona los cursos alternos Music Theory 110 y Introduction to Java Programming 180
- El sistema determina que John tiene todos los pre-requisitos necesarios al examinar el registro del alumno y lo agrega a la lista de los cursos
- El sistema indica que la actividad se ha completado. El sistema imprime el horario del alumno y envia informacion de cobro para cuatro cursos al sistema de cobro para procesarlo

Sustantivos del Escenario "Crear Horario"

- John
- Sistema
- Semestre
- Horario
- Geology 110
- College Algebra 110
- Cursos alternos
- Music Theory 110
- Prerequisitos necesarios
- Horario de estudiante
- Información de cobro
- Cuatro cursos
- Sistema de cobro
- Numero de ID 369523449
- Numero
- Semestre actual
- Lista de cursos disponibles
- Cursos primarios
- English 101
- Introduction to Java Programming 180
- World History 200
- Registro de estudiante
- Lista del curso
- Actividad

¿Qué sustantivos deben filtrarse?

Decisiones de Filtrado

- John -- filtrado (actor)
- Número de ID 369523449 -- filtrado (propiedad del alumno)
- Sistema -- filtrado (lo que se está construyendo)
- Numero -- filtrado (lo mismo que el numero id del alumno)
- Semestre -- filtrado (estado -- cuando la selección aplica)
- Semestre actual -- filtrado (igual que el semestre)
- Horario -- candidato a objeto
- Lista de cursos disponibles -- candidato a objeto
- Cursos primarios -- filtrado (estado de un curso seleccionado)
- English 101 -- candidato a objeto
- Geology 110 -- candidato a objeto
- World History 200 -- candidato a objeto
- College Algebra 110 -- candidato a objeto

Decisiones de Filtrado

- Cursos alternos -- filtrado (estado de un curso seleccionado)
- Music Theory 110 -- candidato a objeto
- Introduction to Java Programming 180 -- candidato a objeto
- Prerequisitos necesarios -- filtrado (curso como otros cursos identificados)
- Registro de estudiante -- candidato a objeto
- Lista del curso -- candidato a objeto
- Actividad -- filtrado (Expresión en inglés)
- Horario de estudiante -- filtrado (igual que el nuevo horario)
- Información de cobro -- candidato a objeto
- Cuatro cursos -- filtrado (información necesitada por la información de cobro)
- Sistema de cobro -- filtrado (actor)

Candidatos a Objetos en el Escenario

- **Horario** -- lista de cursos por semestre para un alumno
- **Lista de cursos disponibles** -- lista de todos los cursos que se imparten en un semestre
- **English 101** -- una oferta para un semestre
- **Geology 110** -- una oferta para un semestre
- **World History 200** -- una oferta para un semestre
- **College Algebra 110** -- una oferta para un semestre
- **Music Theory 110** -- una oferta para un semestre
- **Introduction to Java Programming 180** -- una oferta para un semestre

Candidatos a Objetos en el Escenario

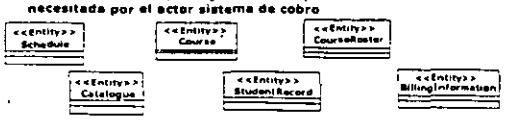
- **Registro de estudiante** -- una lista de cursos que el alumno tomo en semestres previos
- **Lista del curso** -- lista de alumnos para una oferta de curso especifica
- **Información de cobro** -- información que necesita el actor sistema de cobro

Creación de Clases

- Los objetos entity encontrados se agrupan en clases
  - Basado en estructura y/o comportamiento similar
- Esto es sólo un intento inicial
  - Las clases pueden cambiar al examinar mas escenarios

Clases Candidatas Entity -- Escenario "Crear Horario"

- **Horario (Schedule) -- lista de cursos para un semestre para un alumno**
- **Catálogo (Catalogue) -- lista de todos los cursos que se imparten en un semestre**
- **Curso (Course) -- una oferta para un semestre**
- **RegistroEstudiante (StudentRecord) -- lista de cursos previamente tomados**
- **ListaCurso (CourseRoster) -- lista de alumnos para una oferta especifica de curso**
- **InformacionCobro (BillingInformation) -- información necesitada por el actor sistema de cobro**



4.

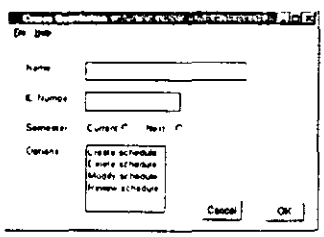
Identificación de Clases Boundary

- Examinar cada par: actor/escenario y crear clases boundary obvias
  - Durante el diseño, la clase se refinará en base a los mecanismos de la interfaz de usuario elegida
- Ejemplo
  - Al alumno se le presentan diferentes opciones en el caso de uso "Inscripción a Cursos"
    - Se crea una clase boundary llamada RegistrationForm para permitir que el alumno seleccione la opción deseada
  - El alumno debe introducir información del curso al sistema en el escenario "Crear Horario"
    - Se crea una clase boundary llamada ScheduleForm para mantener la información que el alumno introduce

Clases Candidatas Boundary -- Escenario "Crear Horario"

Prototipo de Ventana

- Los prototipos de ventanas pueden crearse para comunicar la apariencia y percepcion de la clase boundary al usuario



Identificación de Clases Boundary

- Las clases boundary también se crean por la comunicación de sistema-a-sistema
  - Puede ser otro sistema de software o una pieza de hardware (impresoras, alarmas, etc)
- Las clases boundary se agregan para describir el protocolo de comunicación elegido

Clases Candidatas Boundary -- Escenario "Crear Horario"

- El horario del alumno se imprime en el escenario "Crear Horario"
  - Se crea una clase boundary Printer
- La información de cobro se envía al Sistema de Cobro en el escenario "Crear Horario"
  - Se crea una clase boundary BillingSystem

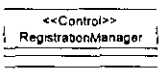
Identificación de Clase Control

- Las clases control contienen típicamente información secuencial
  - Precaución las clases control NO deben desempeñar las responsabilidades que pertenecen típicamente a las clases entity y/o boundary
- En este nivel de análisis, una clase control se agrega típicamente para cada caso de uso
  - Responsable por el flujo de eventos en el caso de uso
- Este es solo un breve inicio
  - Cuanto más casos de uso y escenarios se desarrollen, pueden eliminarse, dividirse o combinarse las clases de control

Reglas para el Caso de Uso "Inscripción a Cursos"

- Se agrega una clase control llamada RegistrationManager
  - Recibe información de la clase boundary ScheduleForm
  - Para cada curso seleccionado
    - **Pide los prerequisitos del curso**
    - **Revisa para asegurarse de que todos los prerequisitos de un curso se tomaron al preguntar a StudentRecord si un prerequisito de curso se había completado**
    - **Sabe que hacer si no se tiene un prerequisito**
    - **Pregunta si el curso está abierto**
    - **Pide a Course que agregue al alumno (si el curso está abierto)**
    - **Sabe que hacer si no están disponibles 4 cursos**
    - **Crea los objetos StudentSchedule y BillingInformation**
    - **Pide al BillingSystem que envíe la BillingInformation**

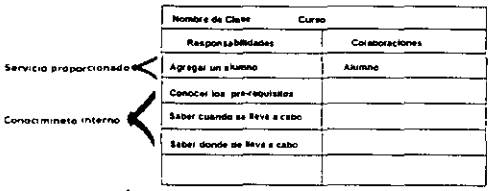
Clase Candidata Control -- Caso de Uso "Inscripción a Cursos"



Tarjetas Responsabilidad-Colaboración de Clases

- Las clases también pueden descubrirse usando tarjetas responsabilidad-colaboración de clases (Class-Responsibility-Collaboration Cards, CRC)
  - **Introducidas por Ward Cunningham y Kent Beck at OOPSLA en 1989**
- Una tarjeta CRC es una tarjeta de 3" x 5" que muestra
  - **El nombre y descripción de la clase**
  - **Las responsabilidades de la clase**
    - **Conocimiento interno de la clase**
    - **Servicios proporcionados por la clase**
  - **Los colaboradores para las responsabilidades**
    - **Un colaborador es una clase cuyos servicios necesitan una responsabilidad**

Tarjeta CRC para la Clase Curso



Una sesión de tarjeta CRC

- Un grupo de personas ejecutan un escenario
- Se crea una tarjeta para cada objeto en el escenario
- Se le asigna un grupo de tarjetas a cada participante
  - La persona se convierte en la "clase"
- Los participantes actúan a los escenarios definidos
- Se anotan responsabilidades y colaboraciones en las tarjetas
- Se crean tarjetas para los objetos descubiertos

Beneficios de las Tarjetas CRC

- Al completar mas y mas escenarios, emergen los patrones de colaboracion
- Las tarjetas pueden arreglarse fisicamente para representar estas colaboraciones cerradas
- Esto puede ayudar a identificar jerarquías de generalizacion/especialización o jerarquías de agregacion entre las clases
- Las tarjetas CRC son más efectivas para grupos que desconocen las tecnicas OO, ya que ellos
  - Evitan enfocarse a elementos OOP
  - Evitan generalizacion prematura
  - Fortalecen "object think"

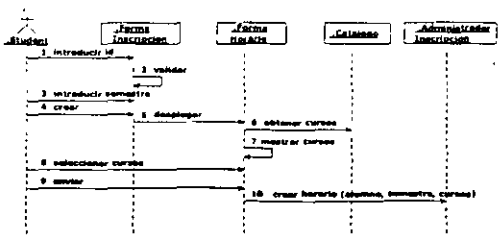
¿Cómo lo estoy haciendo?

- Las cosas van bien si...
  - Todas las clases tienen nombre significativos, específicos del dominio
  - Cada clase tiene un pequeño grupo de colaboradores
  - No hay clases "indispensables" (una clase que colabora con todos necesita ser redefinida)
  - La información para cada clase se ajusta bien en una tarjeta de 3X5
  - Las clases pueden manejar un cambio en requerimientos
- Las cosas van mal si...
  - Varias clases no tienen responsabilidades
  - Una sola responsabilidad se le asigna a varias clases
  - Todas las clases colaboran con todas las clases

Diagramación de Escenarios

- Al descubrir objetos y clases, se documentan en diagramas de interacción
  - Estos diagramas puede ser, un diagrama de secuencias o un diagrama de colaboración
- Los diagramas de interaccion contienen el flujo de eventos para un escenario dado
  - Los nombres de objetos son generales
    - e.g., un alumno en lugar de John
  - Los nombres de objetos pueden omitirse si no se necesitan para la comunicación
  - Se agregan notas y/o scripts de ser necesario

Diagrama de Secuencias para el Escenario "Crear Horario"



Paquetes en el Sistema de Inscripción

- Las clases en el Sistema de Inscripción se pueden agrupar en tres paquetes:
  - University Artifacts, Business Rules, e Interfaces
- UniversityArtifacts
  - Schedule, Catalogue, CourseOffering, StudentRecord, CourseRoster, y Billing Information
- BusinessRules
  - RegistrationManager
- Interfaces
  - RegistrationForm, ScheduleForm, BillingSystem, and Printer

¿Qué es un Diagrama de Clases?

- La vista lógica se hace de varios paquetes y clases
- Un diagrama de clases es la vista lógica de algunos (o todos) los paquetes y clases
  - **Generalmente hay varios diagramas de clase**
- El diagrama de clases principal es típicamente una vista lógica de los paquetes a alto nivel
- Cada paquete posee su propio diagrama de clases principal
- Los diagramas de clases adicionales se agregan como sea necesario
  - **Vista de las clases participando en un escenario**
  - **Vista de las clases "privadas" en el paquete**
  - **Vista de una clase, sus atributos y operaciones**
  - **Vista de una jerarquía de herencia**

Diagrama de Clases Principal para el Sistema de Inscripción

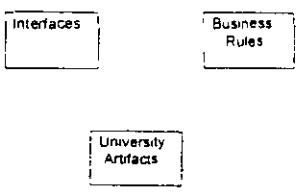


Diagrama de Clases Principal de University Artifacts

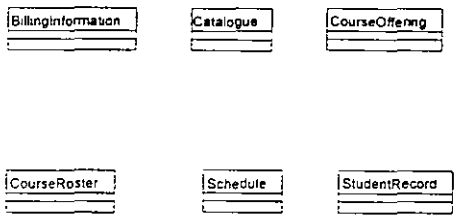


Diagrama de Clases Principal de Interfaces

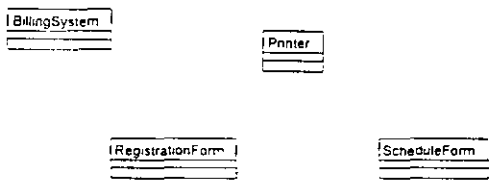
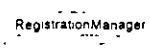


Diagrama de Clases Principal de Business Rules



Ejercicio: Identificación de Clases

- Tome un caso de uso desarrollado en la lección previa
  - Diagrame al menos un escenario en un diagrama de interaccion
    - **Cree clases entity, boundary y/o control que sean necesarias**
    - **Escriba una definición para cada clase**
- Cree paquetes para el modelo
- Coloque las clases descubiertas en paquetes
- Cree diagramas de clase iniciales

Relaciones

Objetivos: Relaciones

- Usted podra:
  - Nombrar los dos importantes tipos de relaciones entre clases, asociacion y agregación
  - Definir asociación y representarla en diagramas de clases
  - Usar nombres de asociación y nombres de rol para clarificar las asociaciones
  - Definir y especificar la multiplicidad de una asociación
  - Definir agregación y representarla en diagramas de clases
  - Definir y representar una asociación reflexiva o agregada
  - Usar clases de asociación
  - Definir calificadores y representarlos en diagramas de clases
  - Descubrir relaciones a partir de los diagramas de escenario

La Necesidad de Relaciones

- Todos los sistemas abarcan varias clases y objetos
- Los objetos contribuyen al comportamiento del sistema colaborando unos con otros
  - La colaboracion se realiza a traves de las relaciones
- Hay dos importantes tipos de relaciones durante el analisis
  - Asociacion
  - Agregacion

Asociaciones

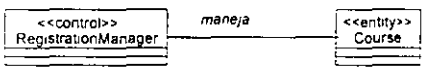
- Una asociacion es una conexión semantica bi-direccional entre clases
  - Esto implica que hay una liga entre objetos entre las clases asociadas
- Las asociaciones se representan en diagramas de clase por una linea que conecta las clases asociadas
- La informacion puede fluir en cualquier direccion o en ambas direcciones a traves de la liga

Navegación

- Una asociación es una relacion bi-direccional
  - Dada una instancia de RegistrationManager hay un objeto asociado Course
  - Dada una instancia de Course hay un objeto asociado RegistrationManager

Nombrando Asociaciones

- Una asociacion se debe nombrar para esclarecer su significado
- El nombre se representa con una etiqueta que se pone a lo largo de la linea de asociación, entre los iconos de clases
- Un nombre de asociación es generalmente un verbo o una frase con verbo

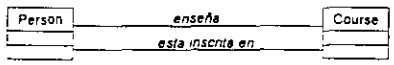


Definición de Roles

- Un rol denota el proposito o capacidad en la que una clase se asocia con otra
- Los nombres de roles son tipicamente sustantivos o frases con sustantivo
- Un nombre de rol se pone a lo largo de la linea de asociacion cerca de la clase que modifica
  - En uno o en ambos extremos de una asociacion se pueden tener roles

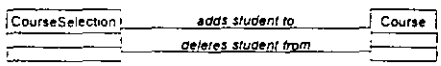
Asociaciones Múltiples

- Puede existir mas de una asociacion entre dos clases
- Si hay más de una asociación entre dos clases se les DEBE de nombrar



- Las asociaciones múltiples deben cuestionarse

Asociaciones Múltiples (cont.)



¿Modelo bueno o malo?

Multiplicidad para Asociaciones

- Multiplicidad es el numero de instancias de una clase relacionada a UNA instancia de otra clase
- Para cada asociación, hay dos decisiones de multiplicidad que tomar una por cada extremo de la asociacion
- Por ejemplo, en la conexión entre Person jugando el rol maestro y Course
  - Para cada instancia de Person, varios (i e , cero o más) Courses deben impartirse
  - Para cada instancia de Course, exactamente una instancia de Person es maestro

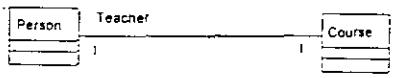
Indicadores de Multiplicidad

- Cada extremo de una asociacion tiene indicadores de multiplicidad
  - Indica el numero de objetos que participan en la relación

Muchos	*
Exactamente uno	1
Cero o más	0..*
Uno o más	1..*
Cero o uno	0..1
Rango específico	2..4

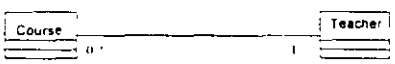
Ejemplo: Multiplicidad

- La multiplicidad expone varias hipótesis ocultas sobre el problema que se está modelando
  - ¿Puede estar un maestro en sabático?
  - ¿Puede tener un curso dos maestros?



¿Qué significa Multiplicidad?

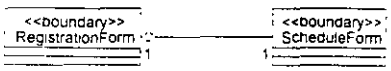
- La multiplicidad debe responder a dos preguntas
  - ¿La asociacion es obligatoria u opcional?
  - ¿Cual es el numero minimo y maximo de instancias que pueden ligarse a una instancia?



¿Qué le dice este diagrama?

Agregación

- La agregacion es una forma especializada de asociacion en la que un todo se relaciona con sus partes
  - La agregacion es conocida como la relacion "parte de" o "contiene a"
- Una agregación se representa como una asociación con un diamante en el extremo de la liga, del lado de la clase que denota el agregado (todo)
- La multiplicidad se representa de la misma manera que otras asociaciones

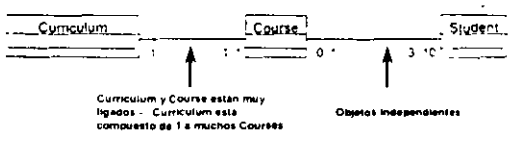


Pruebas de Agregación

- ¿Se usa la frase "parte de" para describir relaciones?
  - **Una Puerta es "parte de" un Carro**
- ¿Se aplican algunas operaciones en el todo y automaticamente a sus partes?
  - **Al mover el Carro, se mueve la Puerta**
- ¿Se propagan algunos valores de atributos del todo a todas o algunas de sus partes?
  - **El Carro es azul, la Puerta es azul**
- ¿Hay una asimetria intrinseca a la relacion donde una clase se subordina a la otra?
  - **Una Puerta es parte de un Carro, un Carro NO es parte de una Puerta**

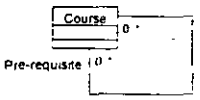
¿Asociación o Agregación?

- Si dos objetos estan estrictamente limitados por una relacion complementaria
  - La relacion es un agregación
- Si dos objetos se consideran usualmente como independientes, y aún asi estan ligados
  - La relacion es una asociacion



Asociaciones Reflexivas

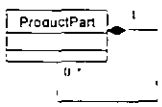
- En una asociación reflexiva, se relacionan los objetos de la misma clase
  - Indica que múltiples objetos en la misma clase colaboran juntas en otra forma



Un curso puede tener muchos pre-requisitos
Un curso puede ser pre-requisito de otros cursos

Agregaciones Reflexivas

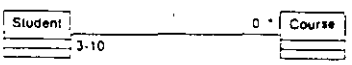
- Las agregaciones pueden tambien ser reflexivas
  - El tipo de problema clasico de productos y sus partes
- Esto indica una relación recursiva



Un objeto ProductPart está "compuesto de" cero o mas objetos ProductPart

Clase de Asociación

- Si quisieramos rastrear los grados para todos los cursos que un alumno ha tomado, entonces
- La relacion entre Student y Course es una relacion de muchos-a-muchos
- ¿Donde ponemos el atributo de grado?

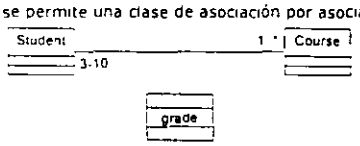


Clase de Asociación (cont.)

- El atributo grado no puede ponerse en la clase Course porque existen (potencialmente) varias ligas a objetos Student
- El atributo grado no puede ponerse en la clase Student porque existen (potencialmente) varias ligas a objetos Course
- Por lo tanto, el atributo en realidad pertenece a la liga Student-Course
- Una clase de asociación se usa para mantener dicha información

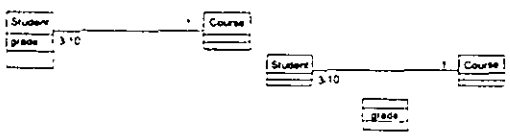
Dibujo de Clase de Asociación

- Se crea una clase de asociación usando el ícono clase
- Se conecta el ícono de la clase a la línea de asociación con una línea punteada
- La clase de asociación puede incluir múltiples propiedades de la asociación
- Solo se permite una clase de asociación por asociación



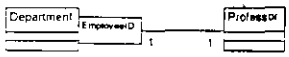
Clase de Asociación y Multiplicidad

- Las clases de asociación se emplean en asociaciones de muchos-a-muchos
- Si la multiplicidad en cualquiera de los extremos de una asociación es "a-una"
  - El atributo puede ponerse en la clase donde la relación es "a muchos", o
  - Puede aun usarse una clase asociación

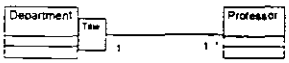


Calificadores

- Un calificador es un atributo o grupo de atributos cuyos valores dividen el conjunto de objetos relacionado a un objeto a través de una asociación



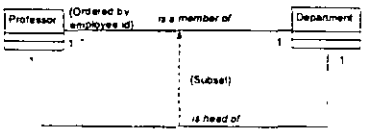
Dado un objeto Department y un valor para un Employee ID hay exactamente un objeto Professor



Dado un objeto Department y un valor para Title hay un conjunto de objetos Professor

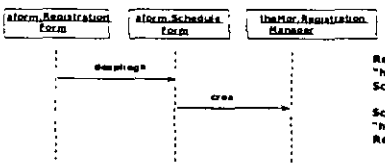
Restricciones

- Una restricción es la expresion de alguna condición que se debe preservar
  - Una restriccion se muestra como una linea punteada



Identificación de Asociaciones y Agregaciones

- Deben examinarse los escenarios para determinar si una relacion debe existir entre dos clases
  - Dos objetos pueden comunicarse solo si se "conocen" entre ellos
- Las asociaciones y/o agregaciones proporcionan una ruta de comunicacion

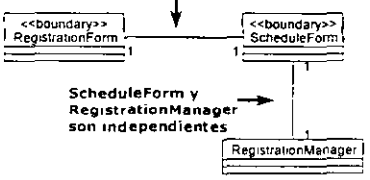


RegistrationForm "habla" con ScheduleForm

ScheduleForm "habla" con RegistrationManager

¿Asociación o Agregación?

RegistrationForm y ScheduleForm están muy ligadas -- una ScheduleForm es "parte de" la RegistrationForm



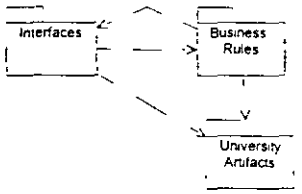
Relaciones de Paquetes

- Los paquetes se relacionan unos con otros a través de una relacion de dependencia
- Si una clase en un paquete "habla" con una clase en otro paquete entonces se agrega una relación de dependencia al nivel del paquete
- Los diagramas de escenario y de clases se evalúan para determinar las relaciones entre paquete

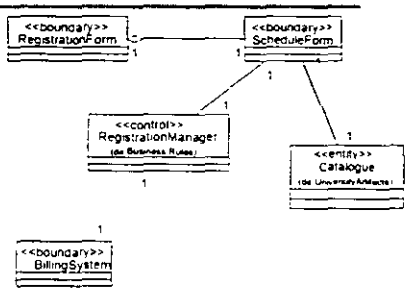
Relaciones en el Análisis y Diseño

- Durante el análisis, se establecen conexiones (asociaciones y agregaciones) entre clases
  - Estas conexiones existen debido a la naturaleza de las clases y no debido a una implementacion especifica
  - Hacer una estimacion inicial de multiplicidad para exponer hipotesis ocultas
- Los diagramas de clases se actualizan para mostrar las relaciones agregadas
- Durante el diseño:
  - Se refinan y actualizan las estimaciones de multiplicidad
  - Se avalúan y refinan las asociaciones y agregaciones
  - Se evaluan y refinan las relaciones de paquetes
  - Se maduran los diagramas de clases

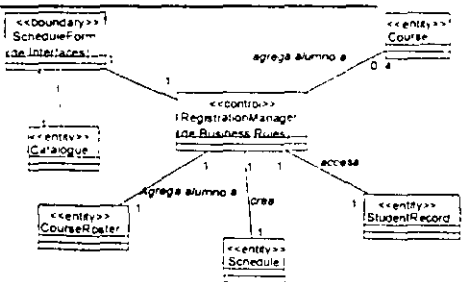
Actualización del Diagrama de Clases Principal para el Sistema de Inscripción



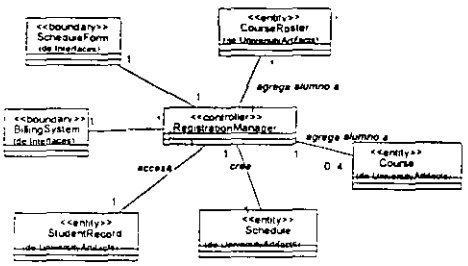
Actualización del Diagrama de Clases de Interfaces



Actualización del Diagrama de Clases de University Artifacts



Actualización del Diagrama de Clases de Business Rules



Ejercicio: Relaciones

- Usando los escenarios y diagramas de secuencias generados en lecciones anteriores
  - Actualizar los diagramas de clases mostrando relaciones entre clases
    - **Asegurarse de que se tomen las decisiones de multiplicidad inicial**
  - Agregar relaciones a los paquetes para el sistema

Operaciones y Atributos

Objetivos: Operaciones y Atributos

- Usted podra
  - Definir operaciones para clases
  - Definir atributos para clases
  - Definir encapsulación y establecer sus beneficios
  - Representar atributos y operaciones en diagramas de clase

¿Qué es una operación?

- Una clase engloba un conjunto de responsabilidades que definen el comportamiento de los objetos de esa clase
- Las responsabilidades de una clase se llevan a cabo por sus operaciones
  - Esto no es necesariamente un mapeo de uno-a-uno
    - Responsabilidad de la clase Producto -- precio de venta
    - Operaciones para esta responsabilidad
      - **Buscar información de una base de datos**
      - **Calcular el precio**
- Una operacion es un servicio que puede ser solicitado desde un objeto al comportamiento de efecto

Una operación debe desempeñar una función simple y cohesiva

Las operaciones dependen del dominio

- Listar las operaciones relevantes al dominio del problema
  - Las operaciones de la clase Persona serán diferentes dependiendo de "quién este preguntando"

Perspectiva del Banquero
recibir renta manejar cuenta recibir lineaDeCrédito

Perspectiva del Doctor
examinar tomarMedicina irAlHospital recibirFactura

Nombrando Operaciones

- Las operaciones deben nombrarse para indicar su resultado, no los pasos detras de la operación
- Ejemplos
  - calculateBalance()
    - Pobremente nombrado
      - **Indica que se debe calcular el balance -- esta es una decisión de implementación/optimización**
  - getBalance()
    - Bien nombrado
      - **Indica solamente el resultado**

Nombrando Operaciones (cont.)

- Las operaciones deben nombrarse desde la perspectiva del proveedor no del cliente
- En una gasolinera, la gasolina se recibe de la bomba
  - La bomba tiene su responsabilidad a traves de una operacion -- ¿como se le debe llamar?
    - Nombres adecuados -- dispense(), giveGas()
    - Nombre malo -- receiveGas()
      - **La bomba de la gasolina — no recibe la gasolina**

¿Qué es una operación primitiva?

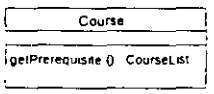
- Una operacion primitiva es una operacion que no puede ser implementada usando solamente las operaciones internas de la clase
  - Todas las operaciones de una clase son tipicamente primitivas
- Ejemplo.
  - Agregar un objeto a un conjunto -- operacion primitiva
  - Agregar cuatro objetos a un conjunto -- no primitiva
    - Se puede implementar con llamadas multiples a la operación de agregar un objeto a un conjunto

Firma de una Operación

- La firma de una operación consiste en:
  - Lista de argumentos opcional
  - Clases o valores de retorno
- Durante el análisis NO ES OBLOGATORIO llenar la firma de una operación
  - Esta información debe completarse en el diseño

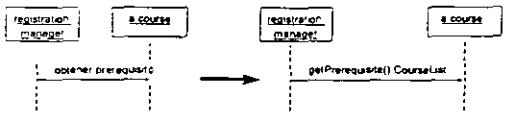
Despliegue de Operaciones

- Las operaciones se muestran en el tercer compartimiento de la clase



Obteniendo Operaciones a partir de los Diagramas de Interacción

- Los mensajes desplegados en los diagramas de secuencias y/o colaboración son generalmente operaciones de la clase receptora
  - Los mensajes se traducen en operaciones y se agregan al diagrama de clase



Descubriendo Clases Adicionales

- Si se especifica una firma de operación, es posible descubrir clases adicionales
  - Argumento en la operacion
  - Clase de retorno
- Ejemplo:
  - getPrerequisite() CourseList
  - addStudent(John StudentInfo)
- Las clases adicionales se agregan al modelo
  - Se despliegan en diagramas de clases cuando sea necesario

Descubriendo Relaciones Adicionales

- Los argumentos de una operación y/o la clase de retorno denotan una relacion entre la clase que posee la operacion y la clase del argumento y/o la clase de retorno
- Ejemplo
  - La clase CourseRoster tiene una operación addStudent(John StudentInfo)
  - Esto implica que hay una relación entre CourseRoster y StudentInfo
- Las relaciones adicionales se agregan al modelo
  - Se despliegan en diagramas de clases cuando sea necesario

¿Qué es un atributo?

- Un atributo es una definicion de dato contendido en instancias de la clase
- Los atributos no tienen comportamiento -- no son objetos
- Los nombres de atributo son sustantivos simples
  - Los nombres deben ser únicos en la clase
- Cada atributo debe tener una definición clara y concisa
- Atributos buenos para la clase Alumno
  - Name -- nombre y apellido
  - Major -- campo superior de estudios
- Atributo malo para la clase Alumno -- selectedCourses
  - Esta es una relación no un atributo

Valores de Atributo

- El valor de atributo esta dado por el estado de un objeto particular
- Cada objeto tiene un valor para cada atributo definido por su clase
- Por ejemplo, para un objeto de la clase Profesor

Los Atributos dependen del dominio

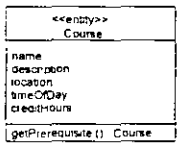
- Listar todos los atributos relevantes para el dominio del problema
  - Los atributos de una clase Persona seran diferentes dependiendo de "quien este preguntando"

Perspectiva de Banquero
nombre
direccion
fechaDeNacimiento
NumeroCuenta

Perspectiva de Doctor
nombre
direccion
fechaDeNacimiento
altura
peso

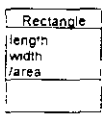
Despliegue de Atributos

- Los atributos se muestran en el segundo compartimiento de la clase



Atributos Derivados

- Un atributo derivado es un atributo cuyo valor puede calcularse en base al valor de otro(s) atributo(s)
  - Se usa cuando no hay tiempo suficiente para re-calcular el valor cada vez que sea necesario
  - Tráfico del desempeño del tiempo de corrida vs memoria requerida



Tipo de Dato, Atributo y Valor Inicial

- Cada atributo tiene
  - Tipo de Dato
  - Valor Inicial (Opcional)
- Durante el analisis NO ES OBLIGATORIO completar la definicion del atributo
  - Esta informacion debe completarse en el diseño

¿Cómo se descubren los atributos?

- Se descubren atributos en el flujo de eventos de los casos de uso
  - Buscar sustantivos que no se consideraron buenos candidatos para clases
- Otros se descubren cuando la definicion de la clase se crea
- Con la ayuda de expertos en el dominio, el cual nos puede proporcionar buenos atributos

Sólo modele los atributos que sean relevantes al dominio del problema

Ejemplo: Atributos en el Problema de Inscripción a Cursos

- "Cada curso tendrá una descripción"
  - Un atributo llamado descripción se agrega a la clase Curso

Guía de Estilo para Nombrar Atributos y Operaciones

- Una guía de estilo debe dictar convenciones de nombres para atributos y operaciones
  - Proporciona consistencia a través del proyecto
  - Conduce a mas modelos y codigo que se puede mantener
- Ejemplo
  - Los atributos y operaciones inician con una letra minuscula
  - No se usan subrayados
    - Los nombres compuestos de multiples palabras se juntan y la primer letra de cada palabra adicional se escribe con mayusculas

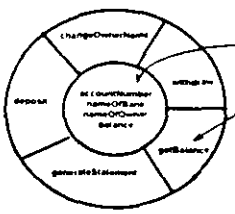
Despliegue de Atributos y Operaciones

- Los atributos y/u operaciones deben mostrarse en una clase
- Pueden crearse diagramas de clases adicionales para desplegar atributos y operaciones
  - Las relaciones típicamente no se despliegan en estos diagramas de clase

Encapsulado

- Un modo de ver una clase es la que consiste de dos partes la interfaz y la implementación
  - La interfaz puede verse y usarse por otros objetos
  - La implementación es oculta para los clientes
- Ocultar detalles de implementación de un objeto se llama encapsulado u ocultamiento de información
- El encapsulado ofrece dos tipos de protección
  - Protege el estado interno de un objeto al ser capturado por sus clientes
  - Protege el código del cliente de los cambios en la implementación del objeto

Ejemplo: Encapsulado



Sólo las operaciones proporcionadas por el objeto pueden cambiar los valores de un atributo

Las operaciones están provistas para desplegar valores de atributos que los clientes necesitan

Los clientes no pueden modificar el estado del objeto directamente

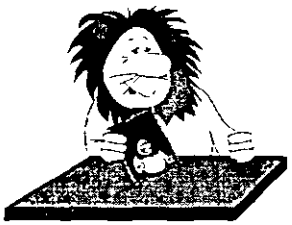
Beneficios del Encapsulado

- El código de la operación de un cliente puede utilizar la intefaz de otra clase
- El código del cliente no puede tomar ventaja de la implementación de una operación de otra clase
- La implementación puede cambiar por los siguientes motivos
  - Corregir un defecto
  - Mejorar el desempeño
  - Reflejar un cambio de política
- El código del cliente no será afectado por los cambios en la implementación, de este modo se reduce el "efecto de rizo (ripple)" en el cual la corrección de una operación fuerza la corrección correspondiente en una operación del cliente
- El mantenimiento es más fácil y menos costoso

Ejercicio: Operaciones y Atributos

- Actualizar los diagramas de secuencias en lecciones previas y transformar los mensajes en nombres de operaciones concisas, tanto como sea necesario
- Crear diagramas de clases mostrando sólo atributos y operaciones
- Agregar relaciones adicionales basadas en argumentos de operaciones y/o clases de retorno

Herencia



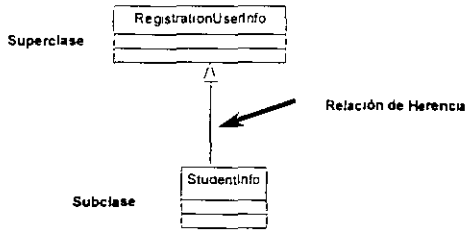
Objetivos: Herencia

- Usted podrá:
  - Definir y discutir herencia, generalización y especialización
  - Representar jerarquías de herencia en diagramas de clases
  - Entender las técnicas para encontrar herencias
  - Definir herencia múltiple

Herencia

- La herencia define una relación entre clases donde una clase comparte la estructura y/o comportamiento de una o mas clases
- La herencia define una jerarquía de abstracciones en las que una subclase hereda de una o mas superclases
  - Con la herencia simple, la subclase hereda solo de una superclase
  - Con la herencia múltiple, la subclase hereda de mas de una superclase
- La herencia es una relacion del tipo "es una" o "tipo de"

Dibujo de una Jerarquía de Herencia



Consideraciones de Herencia

- Debido a que una relacion de herencia no relaciona objetos individuales
  - La relacion no se nombra
  - La multiplicidad no tiene sentido
- Teóricamente, no hay limite en el numero de niveles en una herencia
  - En la practica, los niveles estan limitados
    - Las jerarquias tipicas de C++ son de 3 o 5 niveles
    - Las jerarquias de Smalltalk pueden ser un poco mas profundas

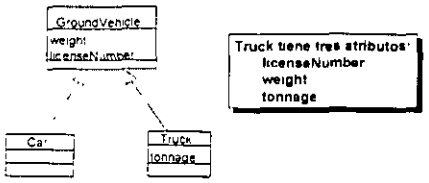
¿Qué es lo que tiene herencia?

- Una subclase hereda de sus padres
  - Atributos
  - Operaciones
  - Relaciones
- Una subclase puede
  - Agregar atributos, operaciones y relaciones
  - Redefine operaciones heredadas (¡sea cuidadoso!)

La herencia controla las similitudes entre clases

Atributos de Herencia

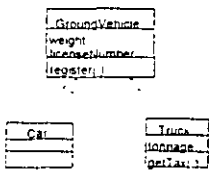
- Los atributos de definen en el mas alto nivel de la jerarquía de herencia
- Las subclases de una superclase heredan todos sus atributos
- Cada subclase puede agregar atributos adicionales



Truck tiene tres atributos:
licenseNumber
weight
tonnage

Operaciones de Herencia

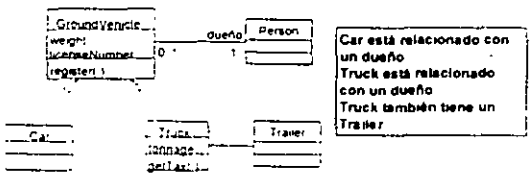
- Las operaciones se definen en el mas alto nivel de la jerarquía de herencia
- Las subclases de una superclase heredan todas las operaciones
- Cada subclase puede aumentar o redefinir operaciones heredadas



Truck tiene tres atributos:
licenseNumber
weight
tonnage
y dos operaciones
register()
getTax()

Relaciones de Herencia

- Las relaciones tambien se heredan y deben definirse en el mas alto nivel en la jerarquía de herencia
- Las subclases de una superclase heredan todas sus relaciones
- Cada subclase puede participar en relaciones adicionales

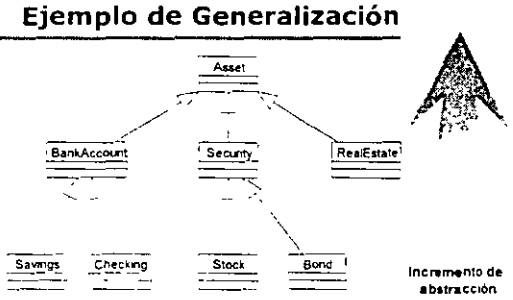


Car está relacionado con un dueño
Truck está relacionado con un dueño
Truck también tiene un Trailer

Generalización de Clases

- La generalización proporciona la capacidad de crear superclases que encapsulan la estructura y/o el comportamiento comunes a varias subclases
- Procedimiento de generalización
  - Identificar similitudes de estructura/comportamiento entre varias clases
  - Crear una superclase que encapsule el comportamiento/estructura comunes
  - Las clases originales se hacen subclases de la superclase nueva
- Las superclases son mas abstractas que sus subclases

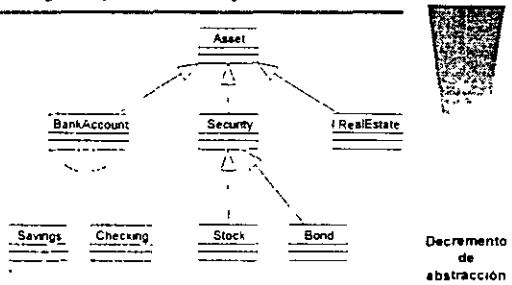
Ejemplo de Generalización



Especialización de Clases

- La especializacion proporciona la capacidad de crear subclases que representen refinamientos en los que la estructura y/o comportamiento de la superclase se agregan o modifican
- Procedimiento de Especializacion
  - Advertir que algunas instancias exhiben estructura o comportamiento especializado
  - Creacion de subclases para agrupar instancias de acuerdo a su especializacion
- Las subclases son menos abstractas que sus superclases

Ejemplo de Especialización

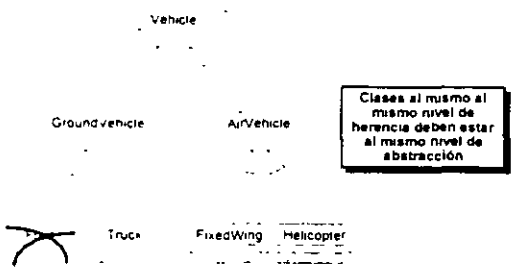


Decremento de abstracción

Jerarquías de Herencia

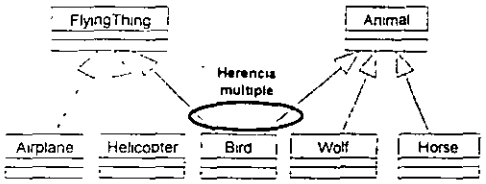
- Ambas técnicas, generalización y especialización, se usan en el desarrollo de jerarquías de herencia
- Durante el análisis, se establecen jerarquías de herencia entre abstracciones, de ser necesario
- Durante el diseño, las jerarquías de herencia se definen para:
  - Incrementar la reutilización
  - Incorporar clases de implementación
  - Incorporar librerías de clase disponibles

Niveles de Abstracción



Clases al mismo al mismo nivel de herencia deben estar al mismo nivel de abstracción

Herencia Múltiple



Bird hereda de ambos FlyingThing y Animal

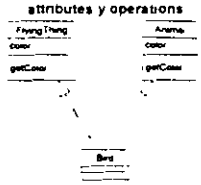
Conceptos de Herencia Múltiple

- Conceptualmente directo y necesario para modelar el mundo real correctamente
- En la practica, puede conducir a dificultades en la implementación
  - No todos los lenguajes orientados a objetos soportan herencia múltiple directamente

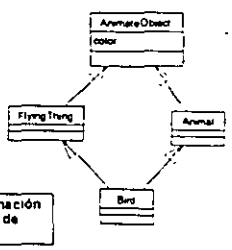
¡Use herencia múltiple sólo cuando sea necesario, y siempre con precaución!

Problemas con Herencia Múltiple

Confusiones con attributes y operations



Herencia repetida



Cada lenguaje/ambiente de programación elige modos para resolver este tipo de dificultades

Identificación de Herencia

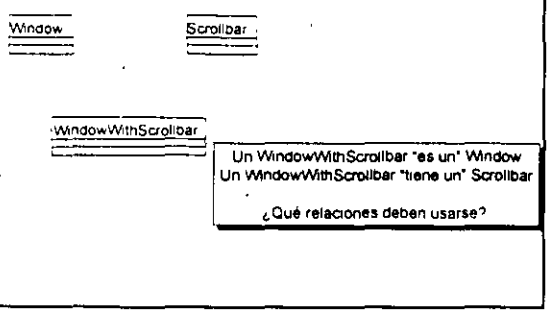
- Es importante evaluar todas las clases para encontrar posibles relaciones de herencia
  - Busque comportamiento comun (operaciones) y estado (atributos) en las clases
- Técnica de Adición
  - Agregar operaciones/atributos nuevos a la(s) subclase(s)
- Técnica de Modificación
  - Redefinir operaciones
    - Debe tener precaución de no cambiar las semánticas

Herencia vs. Agregación

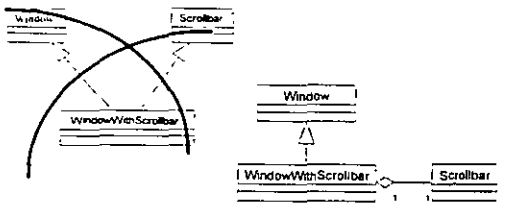
- Con frecuencia se confunde a la herencia y a la agregación
  - La herencia representa una relación "es un" o "tipo de"
  - La agregacion representa una relacion "tiene un"

Las palabras claves "es un" y "tiene un" ayudarán a determinar la relación correcta

Window y Scrollbar



Window y Scrollbar (cont.)



Un WindowWithScrollbar "es un" Window
Un WindowWithScrollbar "tiene un" Scrollbar

Herencia vs. Agregación

Herencia	Agregación
Palabras clave "es un"	Palabras clave "tiene un"
Un objeto	Relaciona objetos en clases diferentes
Se representa con una flecha	Se representa con un diamante

¿Qué es Metamorfosis?

- Metamorfosis
  1. Un cambio en forma, estructura o función, especialmente el cambio físico que sufren algunos animales, como el renacuajo a rana
  2. Cualquier cambio notorio, como en el carácter, apariencia o condición

  - Webster's New World Dictionary
  - Simon & Schuster, Inc , 1979

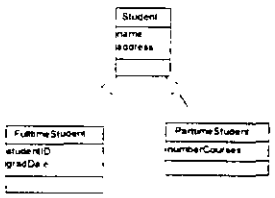
La Metamorfosis existe en el mundo
¿Como debe modelarse?

Ejemplo de Metamorfosis

- En una universidad, hay alumnos de tiempo completo y de medio tiempo
  - Los alumnos de tiempo completo tienen un numero id y una fecha de graduacion, pero los alumnos de medio tiempo no
  - Los alumnos de medio tiempo pueden tomar hasta de tres cursos. Los alumnos de tiempo completo no tienen ese limite

Una Aproximación

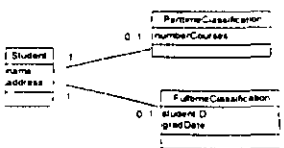
- Se puede crear una relación de herencia



¿Qué sucede si un alumno de medio tiempo desea convertirse en un alumno de tiempo completo?

Metamorfosis

- Es muy difícil cambiar la clase de un objeto
- Tecnica de modelado mejorado
  - Poner la estructura y comportamiento que "cambia" en su propia clase



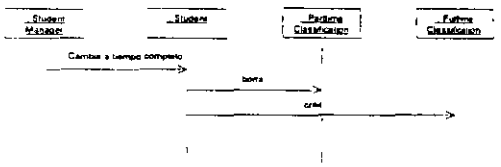
Metamorfosis (cont.)

Mary Smith
Estudiante de tiempo completo

Joe Jones
Estudiante de medio tiempo

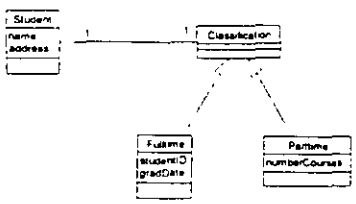
Metamorfosis (cont.)

- La metamorfosis se lleva a cabo por el objeto que "habla" con las partes cambiantes



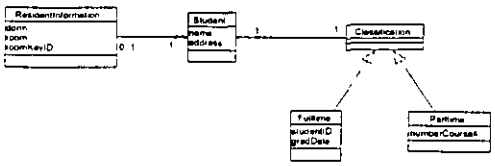
Metamorfosis y Herencia

- La herencia puede usarse para modelar estructura, comportamiento y/o relaciones comunes a las partes "cambiantes"



Metamorfosis y Flexibilidad

- Esta tecnica tambien agrega flexibilidad al modelo
- Ejemplo un alumno puede tambien vivir en el campus En este caso, hay un identificador de dormitorio, un número de habitación y un número de llave de la habitación



Ejercicio: Herencia

- Examinar las clases definidas en el problema hasta el momento y agregar herencia donde sea necesario
  - Asegurarse de considerar cualquier metamorfosis
- Actualizar diagramas de clases como sea necesario

Comportamiento de Objetos

Objetivos: Comportamiento de Objetos

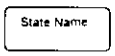
- Usted podra
  - Explicar la necesidad de los Diagramas de Transición de Estado
  - Entender cómo encontrar estados
  - Desarrollar un DTE simple que muestre
    - **Estados y Transiciones**
    - **Eventos**
    - **Condiciones de protección**
    - **Acciones y Actividades**
  - Entender el concepto de estados anidados
  - Explicar las relaciones entre diagramas de transicion de estados, diagramas de objeto/interaccion y diagramas de clase

¿Qué es un Diagrama de Transición de Estado?

- Un diagrama de transicion de estado se usa para mostrar la historia de la vida de una clase dada, los eventos que causan una transicion de un estado a otro, y las acciones que resultan de un cambio de estado
- El espacio de estados de una clase dada es la numeracion de todos los estados posibles de un objeto
- El estado de un objeto es una de las condiciones posibles en las que puede existir un objeto
  - Contiene todas las propiedades del objeto
    - Generalmente estatico
  - Mas los valores actuales de cada una de estas propiedades
    - Generalmente dinamico

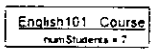
Dibujo de Estados

- Un estado se representa como un rectángulo con esquinas redondeadas en un diagrama de transición de estado



Estado y Atributos

- Los estados pueden distinguirse por los valores de ciertos atributos



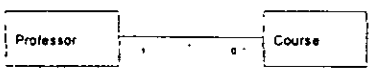
El numero maximo de alumnos por curso es 10

numStudents < 10

numStudents >= 10

Estados y Ligas

- Los estados tambien pueden distinguirse por la existencia de ciertas ligas
- Las instancias de la clase Profesor puede tener dos estados
  - Impartir cuando existe una liga a un curso
  - En sabatico cuando no existe liga



Estados Especiales

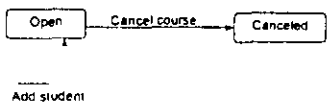
- El estado inicial es el estado introducido cuando se crea un objeto
  - Un estado inicial es obligatorio
  - Sólo un estado inicial es permitido
  - El estado inicial se representa como un círculo sólido
- Un estado final indica el final de vida de un objeto
  - Un estado final es opcional
  - Puede existir mas de un estado final
  - Un estado final se representa con un "ojo de buey"

Eventos

- Un eventos es una ocurrencia que sucede en algún punto en el tiempo
  - El estado del objeto determina la respuesta a diferentes eventos
- Ejemplo
  - Agregacion un alumno a un curso
  - Creacion de un curso nuevo

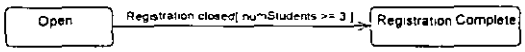
Transiciones

- Una transición es un cambio de un estado original a un estado sucesor como resultado de algunos estimulos
  - El estado sucesor tambien podria ser el estado original
- Una transicion puede tomar lugar en respuesta a un evento
- Las transiciones pueden clasificarse con los eventos



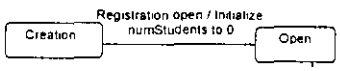
Condiciones de Seguridad

- Una condición de seguridad es una expresión booleana de valores de atributos que permiten una transición sólo si la condición es verdadera



Acciones

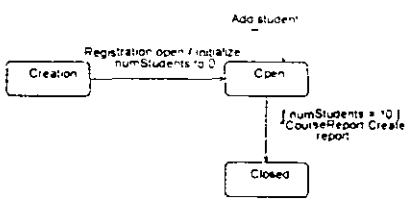
- Una accion es una operacion que se asocia a una transición
  - Toma una cantidad insignificante de tiempo para completarse
  - Se considera no-interrumpible
- Los nombres de acción se muestran en la flecha de transicion precedida por una diagonal



Add student

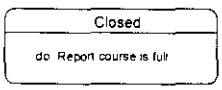
Envío de Eventos

- Un evento puede disparar el envio a otro evento
  - Se muestra como ^Class event



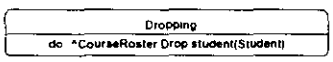
Actividades

- Una actividad es una operacion que toma tiempo para completarse
- Las actividades se asocian con un estado
- Una actividad
  - Inicia cuando se introduce el estado
  - Puede ejecutarse hasta el fin o puede ser interrumpida por una transicion que sale



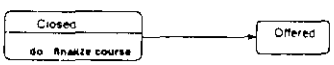
Envío de Eventos en un Estado

- Una actividad tambien puede enviar un evento a otro objeto



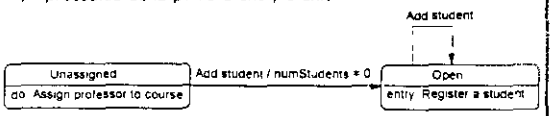
Transiciones Automáticas

- A veces, el único propósito de un estado es ejecutar una actividad
- Una transicion automática ocurre cuando se completa la actividad
- Si hay multiples transiciones automaticas
  - Se necesita una condicion de seguridad en cada transicion
  - Las condiciones deben ser mutuamente excluyentes



Acciones de Entrada y Salida

- Cuando una acción debe ocurrir sin importar como entra o sale del estado, se debe asociar la acción con el estado
  - En realidad, la acción se asocia con cada transición que entre o salga del estado
- La acción se muestra dentro del icono del estado precedida de la palabra entry o exit



Estado Anidado

- Los diagramas de transicion de estado pueden volverse complejos e inmanejables
- Los estado anidados pueden usarse para simplificar diagramas complejos
- Un superestado es un estado que incluye estados anidados llamados subestados
- Las transiciones comunes de los subestados se representan en el nivel del superestado
- Es permitido cualquier número de niveles de anidacion
- Los estados anidados pueden conducir a una reduccion sustancial de complejidad gráfica, permitiendonos modelar problemas mas grandes y complejos

Diagrama de Transición de Estado para la Clase Curso sin Estados Anidados

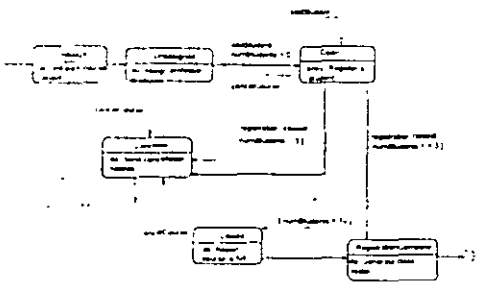
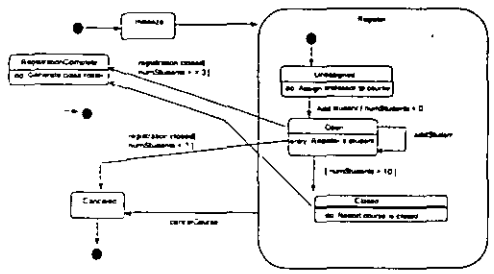


Diagrama de Transición de Estado para la Clase Curso con Estados Anidados



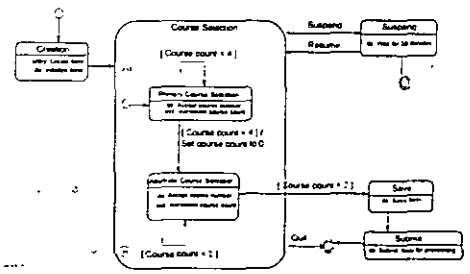
Estados Anidados con Memoria

- El uso de la característica de memoria indica que tras el regreso a un superestado, se introducirá el subestado más recientemente visitado
- Use la letra H en un círculo para denotar que la característica de memoria (history) se aplica al superestado
- Si no se usa la característica de memoria, siempre se tomara el subestado inicial cuando se introduzca el superestado

Ejemplo: Estado Anidados con Memoria

- En el sistema de Inscripcion a Cursos, la clase CourseSelection hace lo siguiente
  - Acepta cursos primarios
  - Acepta cursos alternos
- El usuario puede salir en cualquier momento
- El usuario puede suspender una sesión por un maximo de 30 minutos mientras selecciona sus cursos
- La forma se guarda despues de que se han seleccionado todos los cursos
- La forma se envia para procesarla después de que ha sido guardada

Estado Anidado con Memoria (Clase RegistrationForm)



Donde iniciar...

- Durante el análisis, primero hay que centrarse en las clases con comportamiento dinámico significativo
- Para una clase dada, busque estados posibles al:
  - Evaluar valores de atributos
  - Evaluar operaciones
  - Definirlas reglas para cada estado, e
  - Identificar las transacciones validas entre estados
- Rastrear los mensajes en los diagramas de interacciones correspondientes a un objeto que tengan que ver con el modelado de la clase
  - El intervalo entre dos operaciones puede ser un estado

Ejercicio: Comportamiento de Objetos

- Crear un diagrama de transición de estados para modelar el comportamiento dinámico de la clase Empleado con los siguientes estados
  - Apply, Employed, Leave of Absence, Terminated, Retired
- Información del estado Apply
  - Se conduce una entrevista durante este estado
  - Se sale de este estado con el evento empleo
- Informacion del estado Employed
  - El estado Employed tiene tres subestados basados en su clasificacion de pago
    - Hourly Salaried, and Commissioned
  - La clasificacion de pago puede cambiar en cualquier momento
  - Las clasificaciones de pago se crean/borran despues de entrar/salir del subestado

Ejercicio: Comportamiento de Objetos (cont.)

- Información del estado Leave of Absence
  - Un empleado puede tomar un permiso de ausencia de hasta 1 año mientras este en cualquier subestado de empleado
  - Al regresar de su permiso ingresa al mismo subestado de pago del estado Employed
- Información del estado Terminated
  - Un empleado pude ser despedido mientras este en cualquier subestado de empleado
  - Un empleado puede renunciar mientras este en cualquier subestado de empleado
  - El registro de empleado se marca como terminado en este estado
- Información del estado Retired
  - Un empleado se retira cuando tiene 65 años
  - La información de pensión se calcula en este estado

Homogeneización

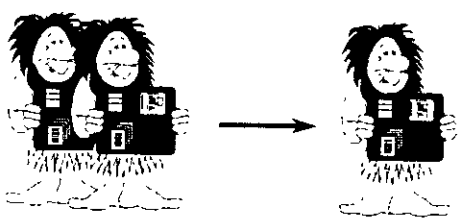


Diagrama de Secuencias para el Escenario "Crear Horario"(cont.)

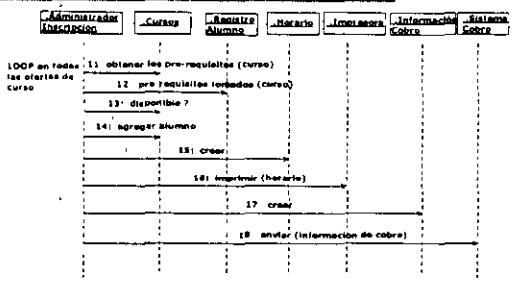
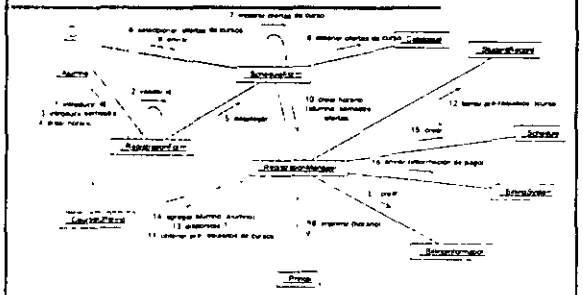


Diagrama de Colaboración para el Escenario "Crear Horario"



¿Qué es un Paquete?

- Un paquete es un mecanismo de proposito general para organizar elementos en grupos
- El numero de clases crecen al analizar los casos de uso y los escenarios
  - Las clases pueden agruparse en paquetes
    - **Proporcionan la habilidad para organizar el modelo en desarrollo**
- Un paquete se representa como una carpeta etiquetada