Unified Modeling Language

Modelos Dinámicos

Prof. Daniel Riesco

®

Aspectos Dinámicos de Modelado de Sistemas

Diagramas de Interacción: conjunto de objetos y sus relaciones incluyendo mensajes que pueden ser enviados entre ellos.

- Diagrama de Colaboración: organización estructural de objetos que envían y reciben mensajes
- Diagrama de Secuencia: orden de tiempo entre los mensajes

Diagrama de Estados: modela una máquina de estados

Diagrama de Actividad: diagrama de flujo que muestra el control de una actividad a otra actividad

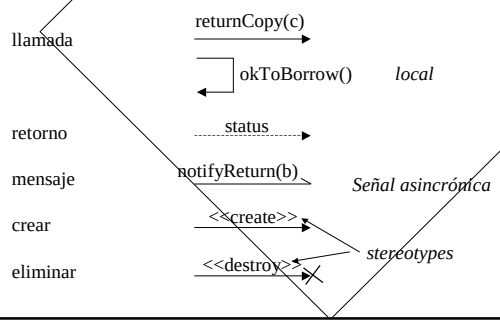
Diagramas de Colaboración

- Un diagrama de colaboración es un grafo de referencias a objs y enlaces con flujo de mensajes ligados a los enlaces.
- Una colaboración es un cjto de participantes (roles) y relaciones que son útiles para un propósito dado.
- Un colaboración puede estar unida a una operación o un caso de uso para describir el contexto en que su comportamiento ocurre. Atada a una clase se utiliza para definir su estructura estática.
- No muestra el tiempo con una dimensión más, sino a través de números.
- Design Patterns:
  - son colaboraciones parametrizadas
  - en cada uso del patrón, clases reales son sustituidas por parámetros en la definición del patrón.

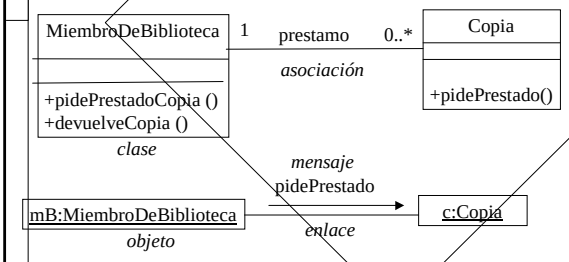
Diagramas de Colaboración (II)

- Notación:
  - objects: {new}, {destroyed}, {transient}
  - Enlaces:
    - <<parameter>>, <<local>>, <<self>>, ...
    - navegación según enlace
  - Rol de colaboración liga a un objeto o enlace
    - classRoleName : ClassifierName
- Flujos de mensajes
  - formatos de flechas
  - [pred/][entero|nbre:][rep/cond] [Ret:=] Mens [Arg]

Acciones sobre Objetos

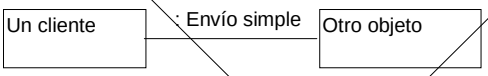


Enlaces



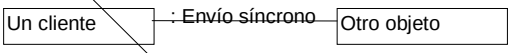
Sincronización

- Las formas de sincronización describen la naturaleza de los mecanismos de comunicación entre objetos
- Envío de mensaje simple:
  - Un solo hilo de ejecución
  - El objeto activo pasa el control al objeto pasivo



Sincronización (Cont.)

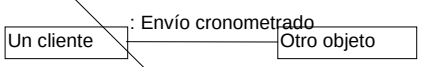
- Envío de mensaje síncrono:
  - Sólo se desencadena la operación cuando el servidor acepta el mensaje
  - El cliente se bloquea hasta que el servidor lo acepta o rechaza



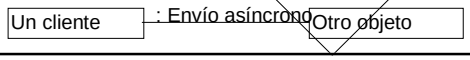
- Envío de mensaje esperado:
  - Se desencadena la operación sólo si el servidor está previamente esperando el mensaje

Sincronización (Cont.)

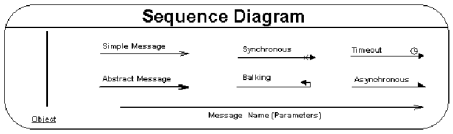
- Envío de mensaje cronometrado:
  - El cliente estará bloqueado durante un tiempo dado esperando la atención del mensaje por el destinatario
  - Tras el tiempo fijado el cliente se libera



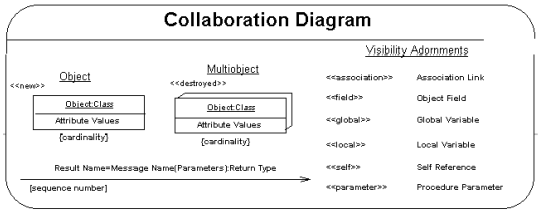
- Envío de mensaje asíncrono:
  - Un mensaje asíncrono no interrumpe la ejecución del cliente
  - El cliente envía el mensaje sin saber cuándo ni si se llevará a cabo



Tipos de Mensajes



Elementos del Diagrama de Colaboración



Acceso a la BD (I)

```
import java.sql.*;

public class Cliente {
  public Statement getStmt() {
  }
  public void insert(Statement stmt) {
  }
  public void print(Statement stmt) {
  }
```

Acceso a la BD (II)

```
public Statement getStmt() {
   String url = "jdbc:odbc:BDEjemplo";
   Connection con;
   try {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   } catch(java.lang.ClassNotFoundException e) {
      System.err.print("ClassNotFoundException: ");
      System.err.println(e.getMessage());
      }
   try {
     con = DriverManager.getConnection(url, "Usu",
   "Psswd");
     return con.createStatement();
    } catch(SQLException ex) {
        System.err.println("SQLException: " +
        ex.getMessage()); }
```

Acceso a la BD (III)

```
public void insert(Statement stmt) {
  try {
       stmt.executeUpdate("insert into CLIENTES " +
       "values('PEPE', 123)"); ...
    } catch(SQLException ex) {
       System.err.println("SQLException: " +
       ex.getMessage());
  }
}
```

Acceso a la BD (IV)

```
public void print(Statement stmt) {
   String query = "select NBRE from CLIENTES";
   try {
        ResultSet rs = stmt.executeQuery(query);
        System.out.println("Clientes:");
        while (rs.next()) {
                String s = rs.getString("NBRE");
                System.out.println(s);
        }
   } catch(SQLException ex) {
        System.err.println("SQLException: " +
        ex.getMessage());
} }}
```

Diagrama de Secuencia: Cambio a Phd

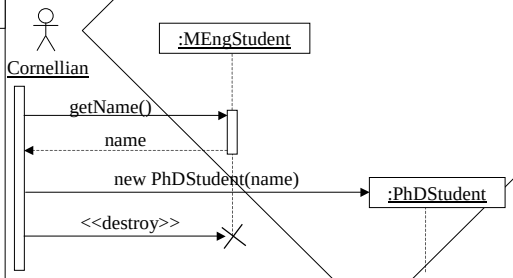


Diagrama de Secuencia: Prestamo de Libro

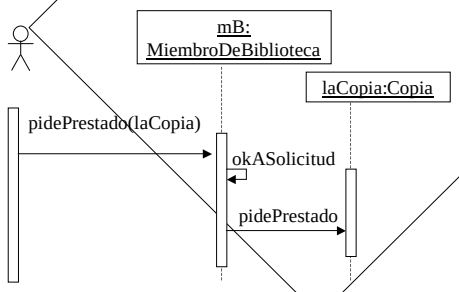


Diagrama de Herencia de Clases

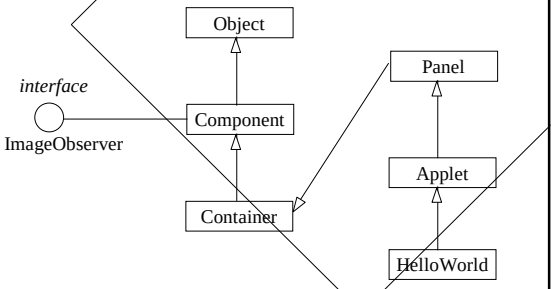
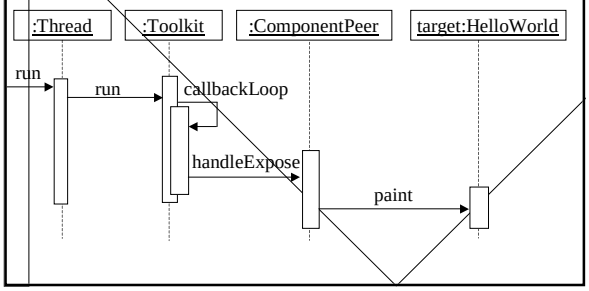
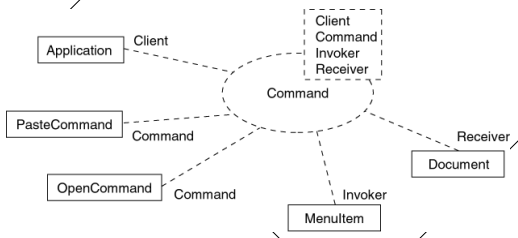


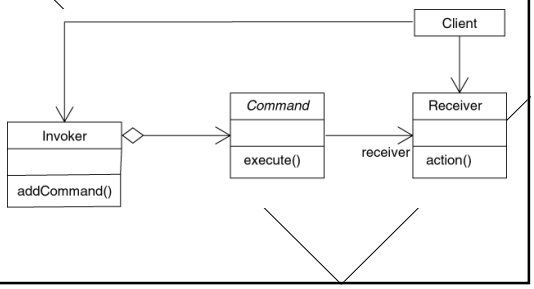
Diagrama de Secuencia: Mecanismo Paint



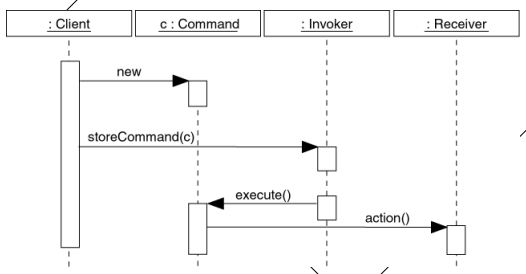
Modelando Patrones de Diseño



Modelado de Patrones de Diseño (cont.)



Modelado de diseño de Patrones (cont.)



Concurrencia (I)

```
public class MyThread2 extends Thread {
ThreadApplet3 applet; boolean forward;
int count; int increment; int end; int position;
MyThread2(ThreadApplet3 applet, boolean forward) {
this.applet = applet; this.forward = forward; }
public void run() { InitCounter(); DoCount(); }
protected void InitCounter() {
   if (forward) {
 count = 0; increment = 1; end = 1000; position = 120;
   } else {
 count = 1000;increment = -1; end = 0;position = 180;}}
protected void DoCount() {
while (count != end) {
  count = count + increment;
  String str = String.valueOf(count);
  applet.SetDisplayStr(str, position);
  try sleep(100);
  catch (InterruptedException e) { } } } }
```

Concurrencia (II)

```
public class ThreadApplet3 extends Applet {
  MyThread2 thread1; MyThread2 thread2;
  String displayStr; Font font; int position;
 public void init() {
  font=new Font("TimesRoman",Font.PLAIN, 72);
  setFont(font);displayStr="";position = 120;
  thread1 = new MyThread2(this, true);
  thread2 = new MyThread2(this, false);}
 public void start() {
  if (thread1.isAlive()) thread1.resume();
  else thread1.start();
  if (thread2.isAlive()) thread2.resume();
  else thread2.start(); }
 public void stop() {
  thread1.suspend(); thread2.suspend();}
 public void destroy() {
  thread1.stop();thread2.stop();}
 public void paint(Graphics g) {
   g.drawString(displayStr, 50, position);}
synchronized public void SetDisplayStr(String
   str, int pos) {
 displayStr = str; position=pos; repaint();}}
```

Diagramas de Estados

- Muestran la secuencia de estados de un objeto o una interacción durante su tpo de vida en respuesta a un estímulo recibido (David Harel's statecharts)
- Es atachado a una clase o un método
- Estado: es una condición durante el tpo de vida de un obj o interacción durante el cual satisface alguna condición, ejecuta alguna acción o espera algún evento. Un obj permanece en el estado por un tpo finito (no instantáneo)
- Acciones: son atómicas, no interrumpibles.
- Estados compuestos.
- Eventos: **when** (cond), **after** (tpo) o nbre (args)
- Transición simple:
  - nbreEvento (args) `[`guard-cond`]` `/` acción ^ clausulaSend
- Transición compleja: representa una sincronización y/o separación del control dentro de threads concurrentes

Diagrama de Estados

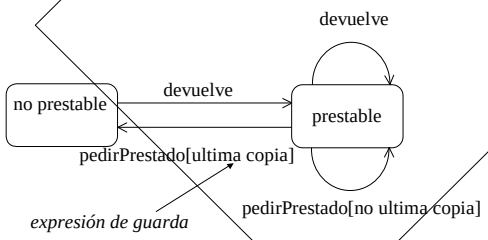
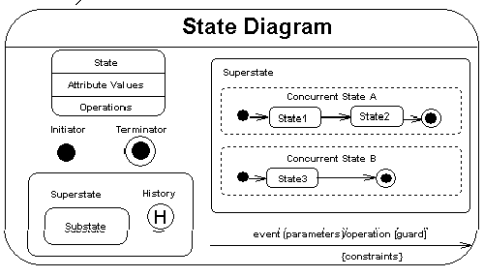


Diagrama de Estado para la clase Libro

Elementos del Diagrama de Estados



Diagramas de Actividad

- Modela los aspectos dinámicos del sistema mostrando el flujo de control entre actividades.
- Una actividad es una ejecución no atómica en curso. Produce una acción.
- Diag. de Interacción -> Flujo de Control entre objetos.
- Diag. de Actividades -> Flujo de Control entre actividades.
- Capturar el camino crítico del flujo de trabajo.
- Estado de acción: son atómicos, no se pueden descomponer, se considera que su ejecución lleva un tiempo insignificante.
- Estado de actividad: no son atómicos (pueden ser interrumpidos), invierten un tiempo en ejecutarse. Un estado de acción puede verse como un caso particular de actividad.

Diagramas de Actividad

- Las Transiciones son sin disparador (inmediatas).
- Bifurcación. Una transición de entrada y varias de salida.
- División y Unión. Barra de sincronización.
- Calles (Swimlanes): Flujo de trabajo de procesos de organiz.
- Se utilizan para:
  - Modelar el Flujo de Trabajo. Se hace hincapié en las actividades tal como son vistas por los actores.
  - Modelar una operación. Modelan los detalles de computación, mostrando la bifurcación, división y unión (flujos paralelos). Muestra el flujo de las acciones de una operación.

Diagrama de Actividad: Modelado de Procesos

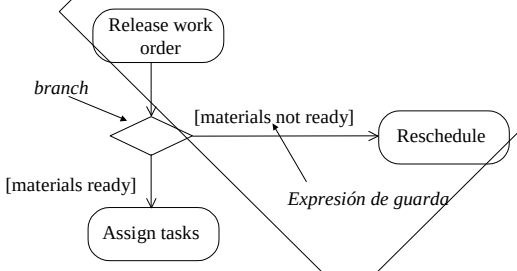
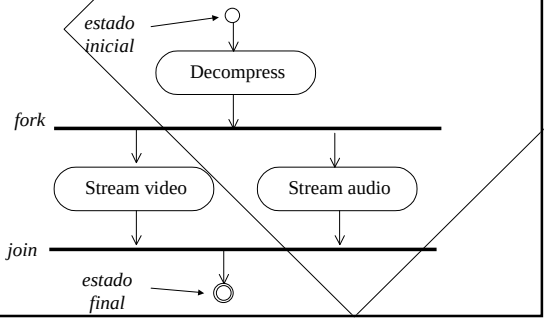


Diagrama de Actividad: Actividades Paralelas



Mecanismos de Extensión General

- Una Restricción es una relación semántica entre elementos del modelo que especifica condiciones y proposiciones que deben ser mantenidas como verdaderas. {}, OCL
- Un comentario es un texto asociado a un elem. modelo
- Una propiedad de un elemento es cualquier valor asociado a un elemento del modelo. Tag = value. Un tag representa una clase particular de propiedad.
- Un estereotipo es una nueva clase de elemento de modelado, que se introduce en tpo de modelado. Representa una subclase de un elemento de modelado existente con la misma forma pero diferente intención. Un elemento stereotipado puede tener restricciones diferentes de un clase base.

Estereotipos

- Los estereotipos permiten añadir nuevos bloques de construcción que poseen
  - características propias (un cjto propio de valores etiquetados)
  - semántica (restricciones propias)
  - notación (un icono especial).
- Son metatipos.
- Se crean cuando se garantizan que no existe ya una forma de expresar lo que se desea en UML.
- Se identifica el elemento primitivo en UML más cercano a lo que se desea modelar (clase, interfaz, nodo, asociación, ...) y se define el estereotipo. Jerarquías de estereotipos.

Propiedades - Valores etiquetados

- Permite crear nuevas propiedades a los bloques básicos o a los nuevos bloques (estereotipos).
  - No es un atributo.
  - Metadato. Se aplica al propio elemento y no a sus instancias.
  - <Tag> '=' <value>. Si es una enumeración sólo el valor.
- Se usa cuando no existe nada en UML que lo exprese.
- Se aplica la propiedad al estereotipo y se aplican las reglas de generalización.
- En general los valores etiquetados y los estereotipos deben homologarse a nivel proyecto, empresarial o corporativo para evitar la creación de ellos por parte de los desarrolladores.

Restricciones y Compartimentos

- Restricciones -- Especificar nueva semántica.
  - Texto libre u OCL.
  - Se puede usar una relación de dependencia para mostrar una conexión más explícita.
- Compartimentos extras.