Programación Avanzada

Diseño

Contenido

- Introducción
- Objetivos
- Arquitectura Lógica
- Diseño de Bajo Nivel

Introducción

- Durante el análisis un caso de uso fue reformulado en términos de interacciones entre los actores y el sistema (DSS)
- El efecto de cada mensaje fue especificado en forma precisa (Contrato)
- Es el momento de definir **cómo** hace el sistema internamente para resolver cada una de las operaciones del sistema

Introducción (2)

- La Arquitectura de Software busca expresar la estructura global de una aplicación
- El nivel de abstracción empleado para expresar dicha estructura es mayor que el empleado para detallar la solución al problema de software planteado
- El objetivo de la arquitectura no es detallar la solución adoptada sino que es proveer una visión global de la misma para simplificar su comprensión

Introducción (3)

- Uno de los puntos de vista desde donde se suele estudiar la estructura de una aplicación es la **estructura interna**
- La arquitectura desde ese punto de vista se denomina **Arquitectura Lógica**
- A definir **cómo** se resuelven las operaciones del sistema en esta estructura se le llama **diseño de bajo nivel** (o simplemente **diseño**)

Objetivos

- Definir la estructura interna del sistema a construir
- Realizar el diseño de las operaciones del sistema

Arquitectura Lógica

- La arquitectura lógica se define como un conjunto de componentes lógicos relacionados entre sí, con responsabilidades específicas
- Estos componentes se obtienen a partir la sucesiva partición del sistema en componentes con responsabilidades más concretas (enfoque top-down)

Arquitectura Lógica (2)

- Existen **guías** de particionamiento
- Cada una de ellas propone
  - Un tipo de partición particular
  - Asignación de responsabilidades a los componentes resultantes
- Se denominan estilos o patrones de arquitectura
- La elección del estilo a aplicar depende del tipo de sistema que se esté construyendo

Aspectos de una Aplicación

- El diseño de un sistema de software comprende la resolución de múltiples aspectos de una aplicación
- La forma en que esos aspectos sean resueltos determina la flexibilidad del diseño
- Desde un punto de vista lógico es preferible separar el diseño de aspectos diferentes para
  - Permitir que evolucionen independientemente
  - Simplificar el problema y tener mejor visibilidad de las partes que componen la aplicación

Aspectos de una Aplicación (2)

- Diseñar e implementar en forma conjunta diferentes aspectos
  - Usualmente simplifica la arquitectura, pero
  - Complica el diseño
- La separación de aspectos generalmente
  - Complica la arquitectura (estructura general)
  - Simplifica el diseño de cada uno al permitir enfocarse en cada aspecto por separado

Aspectos de una Aplicación (3)

```
class Persona {
      //atributos

      void mostrar() {
            write(atributos);
      }

      void procesar(entrada) {
            //hacer algo con entrada y atributos
      }

      void guardar() {
            write(arch,atributos);
      }
}
```

Aspectos de presentación

Aspectos de lógica de la aplicación

Aspectos de acceso a la persistencia de datos

Aspectos de una Aplicación (4)

- En el ejemplo anterior se detectan fragmentos de código con diferentes propósitos en una misma clase
  - Código para procesar la información existente que implementa la lógica de la aplicación,
  - Código de interacción con el usuario, y
  - Código que sirve para almacenar los datos en un medio persistente

Aspectos de una Aplicación (5)

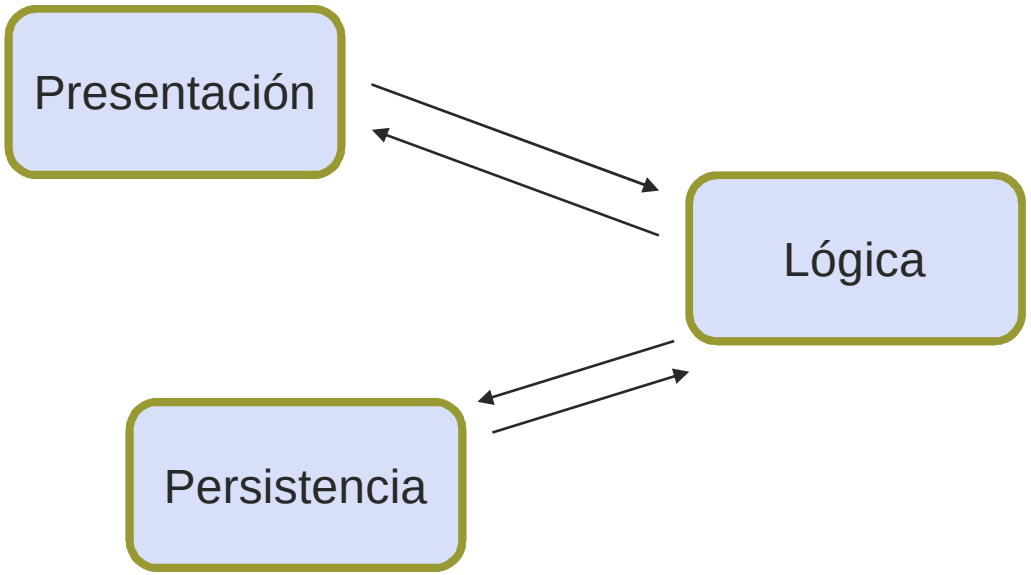
- Esto es común a la mayoría de los sistemas de información interactivos
- Se puede decir que estas aplicaciones abarcan básicamente tres aspectos
  - **Presentación:** incluye todo lo referente a la interacción del sistema con los usuarios en el mundo exterior
  - **Lógica:** se encarga del procesamiento particular que el sistema deba realizar sobre la información que maneja
  - **Persistencia:** consiste en el almacenamiento persistente de dicha información

Aspectos de una Aplicación (6)

- Incluir los tres aspectos en una misma clase no resulta flexible
- La clase completa queda dependiente de
  - La forma en que los datos son mostrados u obtenidos del usuario, y además de
  - La forma en que los datos son almacenados
- Es deseable establecer una separación de dichos aspectos
- Es decir, mantener el código referente a cada aspecto en clases separadas

Aspectos de una Aplicación (7)

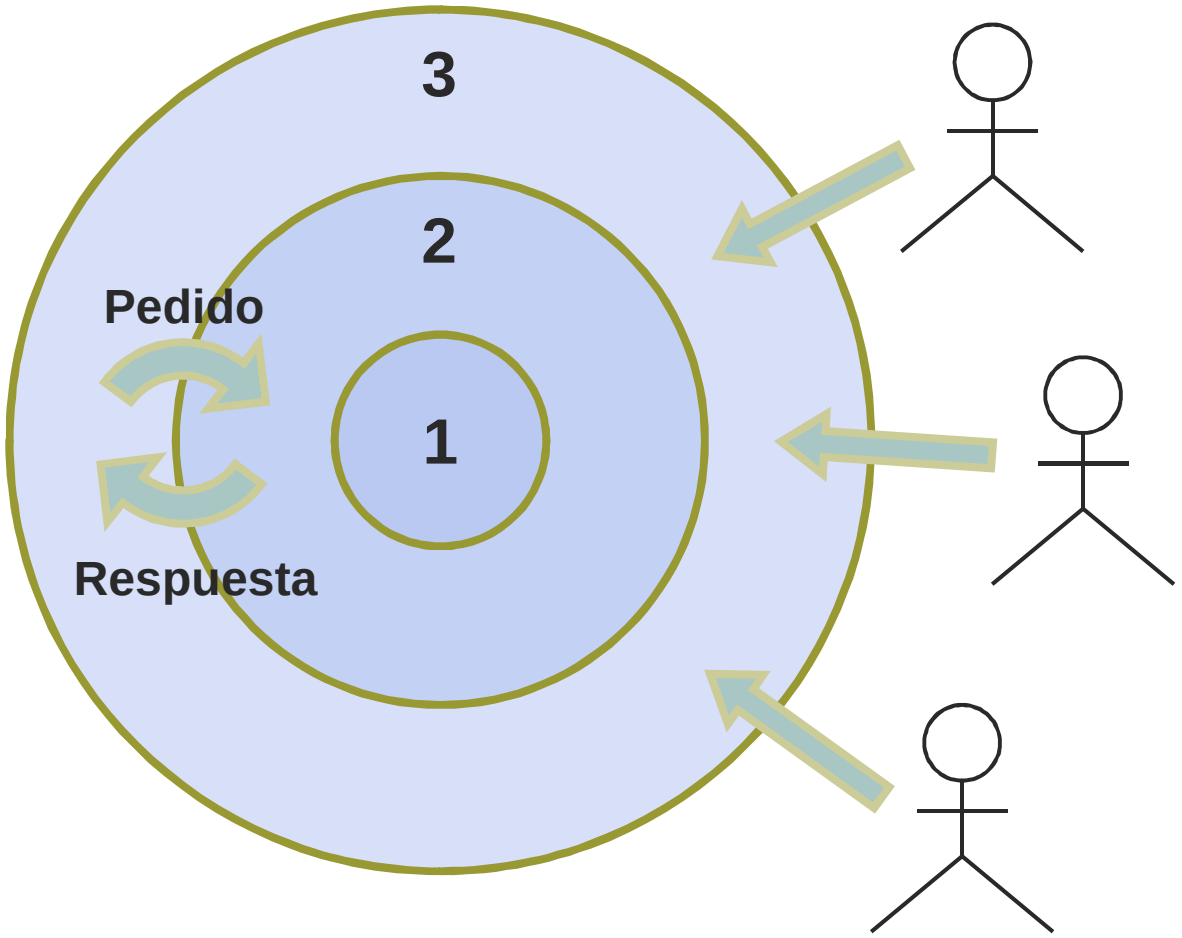
- Esto sugiere un criterio concreto de partición de componentes



Arquitectura en Capas

- Una Arquitectura en Capas es la arquitectura de un sistema que haya sido particionado según el estilo de Capas
  - Define diferentes “niveles” de elementos
  - Los elementos de un mismo nivel tienen responsabilidades de abstracción similar
  - Los elementos de un nivel están para atender los pedidos de los elementos del nivel superior

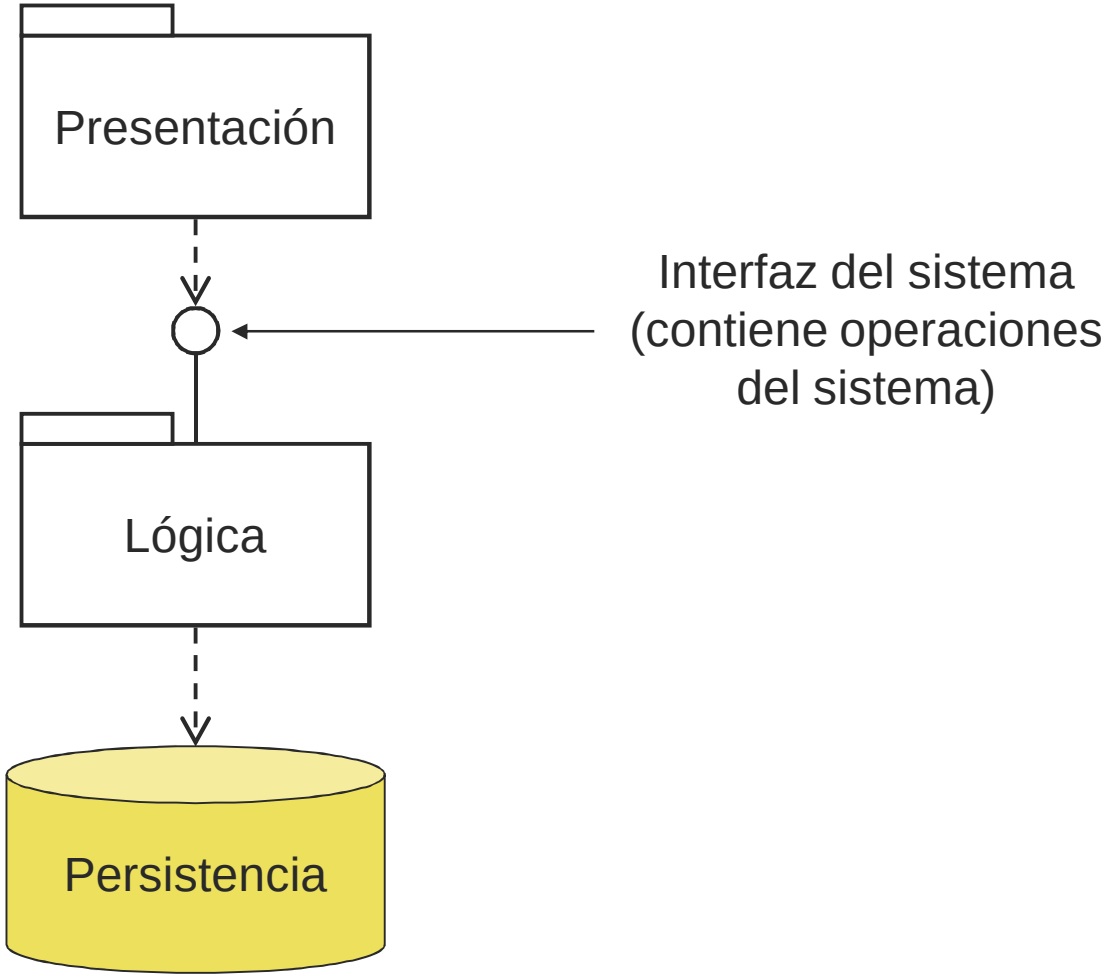
Arquitectura en Capas (2)



Arquitectura en Capas (3)

- La partición definida anteriormente es compatible con el estilo de Capas
- Se definen por lo tanto las siguientes capas
  - Presentación
  - Lógica
  - Persistencia
- Los actores utilizan solamente la capa de presentación
- La capa de persistencia no requiere de los servicios de ninguna otra

Arquitectura en Capas (4)

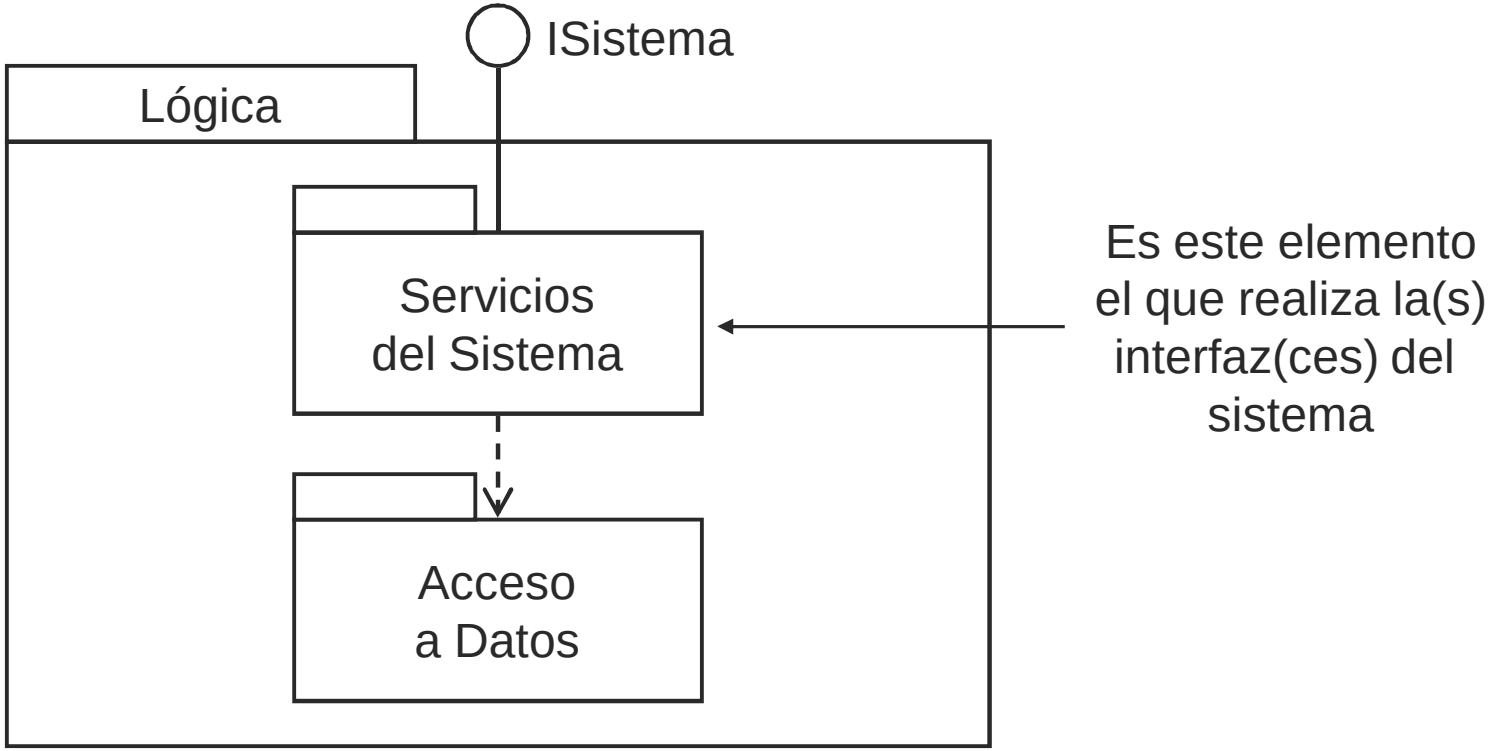


Arquitectura en Capas (5)

- ¿Qué hay en cada capa?
  - Presentación: clases que se encargan de capturar la entrada de los usuarios y mostrar información
  - Lógica:
    - Clases que describen los objetos que procesarán la información para satisfacer los casos de uso del sistema
    - Clases que permiten a las anteriores acceder a los datos
  - Persistencia: datos del sistema que necesiten ser preservados (texto plano, base de datos, etc.)

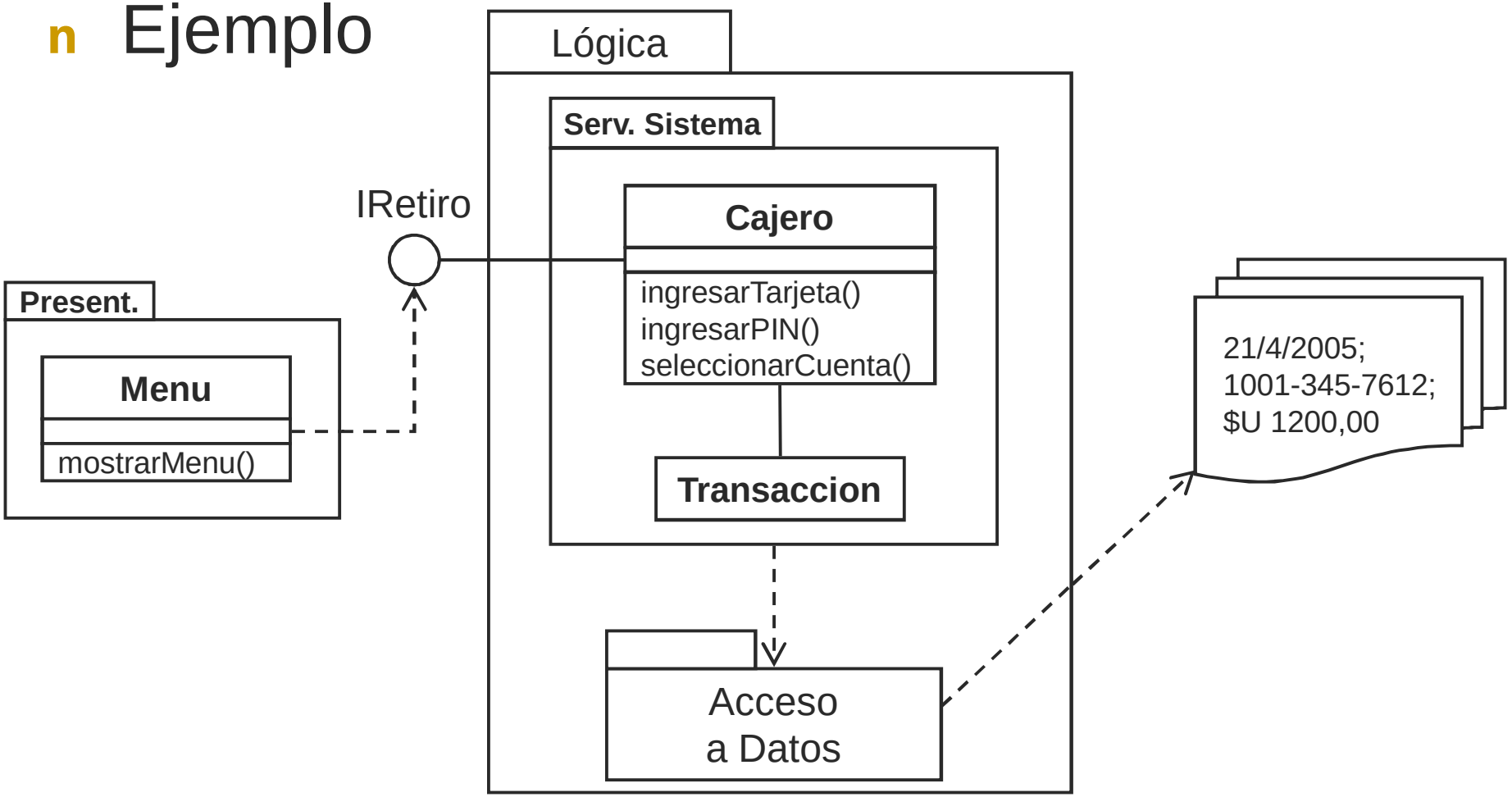
Arquitectura en Capas (6)

- Usualmente la capa lógica es refinada de la siguiente manera



Arquitectura en Capas (7)

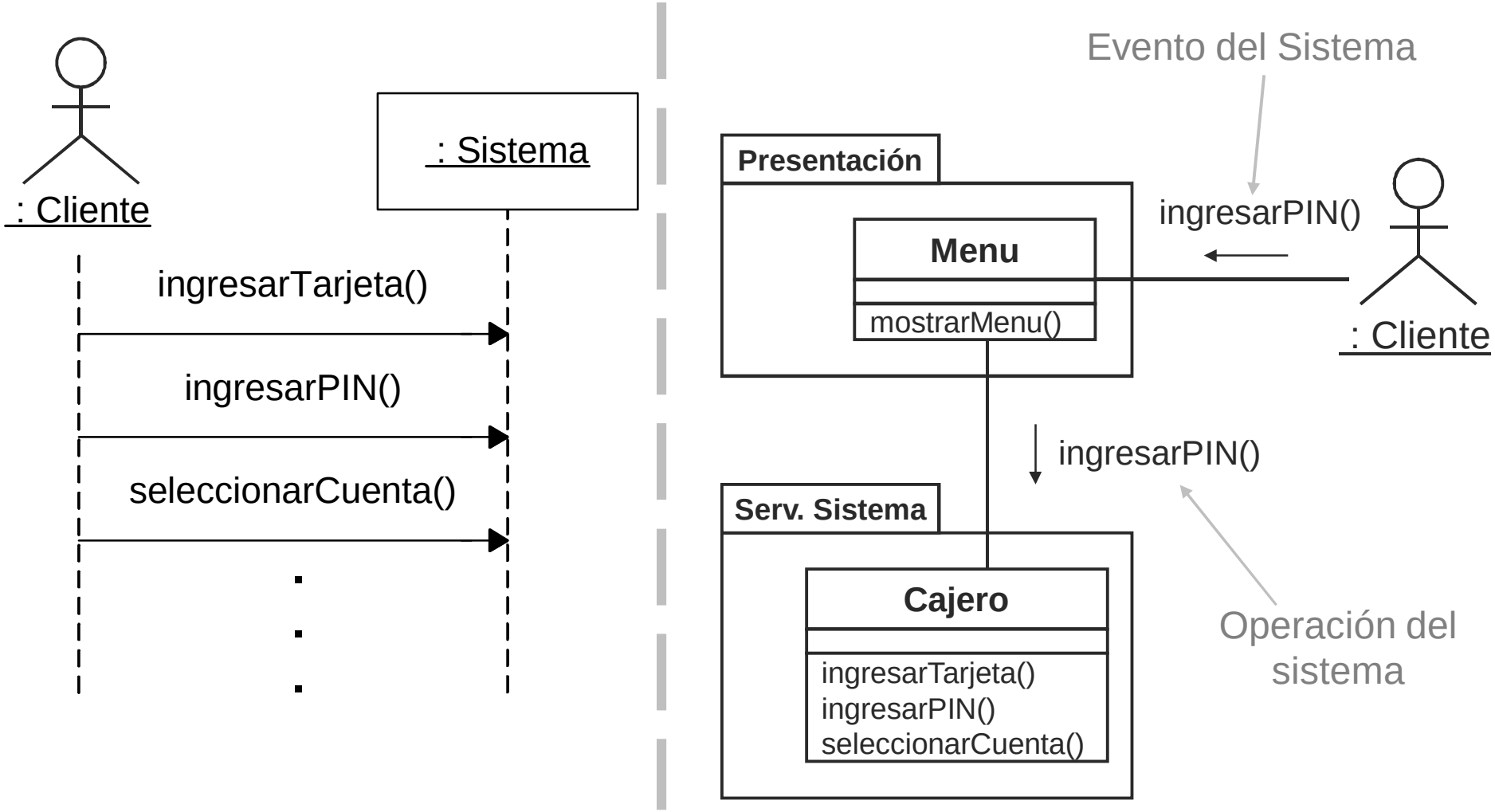
- Ejemplo



Operaciones del Sistema

- Los Diagramas de Secuencia del Sistema ilustran la forma en que los actores realizan “invocaciones” sobre el sistema
- Al estudiar la Arquitectura Lógica es posible profundizar en los detalles de cómo se realizan dichas invocaciones

Operaciones del Sistema (2)



Implementación

```
// pertenece en forma lógica a la Capa de Presentación
class Menu {
      IRetiro atm;

      void mostrarMenu() {
            // leer en t el número de tarjeta
            atm.ingresarTarjeta(t);
            // leer en p el número de PIN
            atm.ingresarPIN(p);
            // leer en c el número de cuenta
            atm.seleccionarCuenta(c);
            .
            .
}
```

Implementación (2)

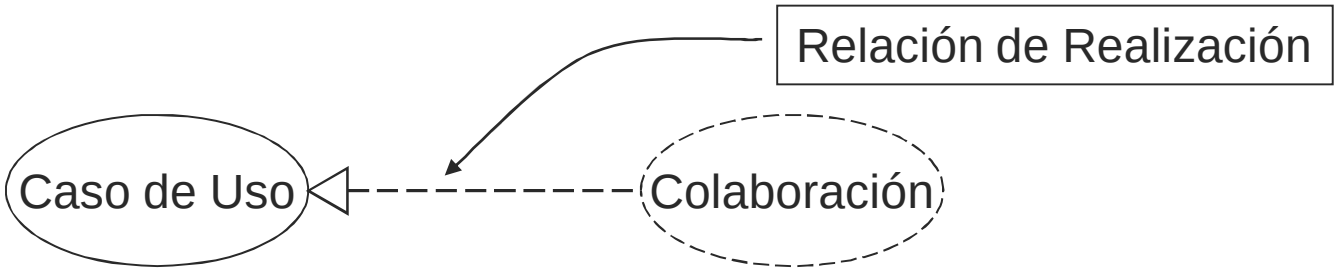
```
// pertenecen en forma lógica a la Capa Lógica
interface IRetiro {
      void ingresarTarjeta();
      void ingresarPIN();
      void seleccionarCuenta();
}

class Cajero realize IRetiro {
      public void ingresarTarjeta() {...}
      public void ingresarPIN {...}
      public void seleccionarTarjeta() {...}
      .
      .
}
```

Diseño de Bajo Nivel

- Tenemos definida la estructura interna del sistema a construir (Arquitectura Lógica)
- A partir de dicha estructura definimos **cómo** se resuelven internamente cada una de las operaciones del sistema
- En este curso, el diseño de bajo nivel estará enfocado en la capa lógica

Objetivos

- Diseñar Colaboraciones que realicen los Casos de Uso del sistema
  - Se busca diseñar una colaboración por cada caso de uso (o varios de ellos juntos)
  - Una colaboración realiza un conjunto de casos de uso cuando define su solución
  - Esta relación es la misma que se puede definir entre una interfaz y un conjunto de clases



Colaboración

- Una Colaboración está compuesta por
  - **Una Estructura:** que indica
    - Las clases de objetos que participan en la solución de los casos de uso
    - Los atributos de las mismas y sus relaciones
    - Las operaciones que pueden ser invocadas sobre sus instancias
  - **Interacciones:** que definen la forma en que objetos de las clases dadas se comunican para obtener el resultado deseado

Colaboración (2)

- La estructura de la colaboración indica **quién** participa y sus propiedades
- Las interacciones de la colaboración indican **cómo** los participantes logran el resultado
- Las colaboraciones aparecen especificadas en el **Modelo de Diseño**

Colaboración (3)

- Por lo tanto una Colaboración que realice un conjunto de Casos de Uso contendrá
  - La estructura de los participantes
  - Una interacción en términos de dichos participantes para cada operación del sistema
    - En cada interacción se detalla la forma en que la operación del sistema es resuelta

Enfoque

- Existen dos enfoques para diseñar una colaboración
  - Definir primero la estructura y luego generar las diferentes interacciones respetándola
  - Definir “libremente” las interacciones y luego definir la estructura necesaria para que éstas puedan ocurrir
- En el curso seguiremos el segundo enfoque

Actividades

- Para lograr los objetivos planteados realizaremos las siguientes actividades
  - Diseño de interacciones
  - Diseño de la estructura

Diseño de Interacciones

- Consiste en definir comunicaciones entre objetos que permitan resolver operaciones del sistema
- Esta definición se realiza “libremente”
  - Los protagonistas aparecen “sugeridos” en el Modelo de Dominio
  - El resultado es el especificado en el contrato de la operación del sistema a diseñar
- La libertad está dada en los mensajes que los protagonistas se puedan enviar entre sí
- Herramienta: Diagrama de Comunicación

Diseño de la Estructura

- Consiste en especificar completamente la estructura necesaria para que todas las interacciones puedan ocurrir
  - Se busca especificar la estructura de una colaboración
  - Por lo tanto es necesario considerar todas las interacciones del caso de uso que la colaboración realiza
  - Recordar que se define una interacción por cada operación del sistema
- Herramienta: Diagrama de Clases de Diseño

Consideraciones

- Durante la etapa de diseño de una metodología iterativa e incremental se obtienen un conjunto de colaboraciones que comprenden todos los casos de uso del sistema
- En consecuencia, se obtiene un diagrama de comunicación por operación del sistema y un conjunto de DCDs, uno por colaboración
- Los DCD pueden requerir algún tipo de revisión general de alguien con una visión global de la solución a los efectos de eliminar inconsistencias

Consideraciones (2)

- Durante el diseño la idea clave es la de asignación de responsabilidades
- La asignación de responsabilidades se realiza (en parte) **definiendo operaciones** para los participantes de la solución
- Es posible definir diferentes interacciones para lograr un mismo efecto
- Esto es asignando responsabilidades en maneras diferentes

Consideraciones (3)

- A pesar de que pueden existir varias soluciones no todas tienen buenas cualidades (flexibilidad, extensibilidad, adaptabilidad, etc.)
- Buscaremos encontrar soluciones que además presenten buenas cualidades
- Para ello utilizaremos criterios de asignación de responsabilidades
- Estos criterios buscan evitar la toma de malas decisiones al momento de asignar responsabilidades

Modelo de Diseño

- El Modelo de Diseño es una abstracción de la solución lógica al problema
- Incluye todas las clases de objetos (y otros elementos) que conforman la estructura necesaria para el funcionamiento del sistema
- Dichos elementos pueden estar eventualmente organizados en paquetes de diseño

Modelo de Diseño (2)

- El Modelo de Diseño también incluye las interacciones que realizan los casos de uso
- Una interacción está expresada en términos de elementos de diseño del modelo

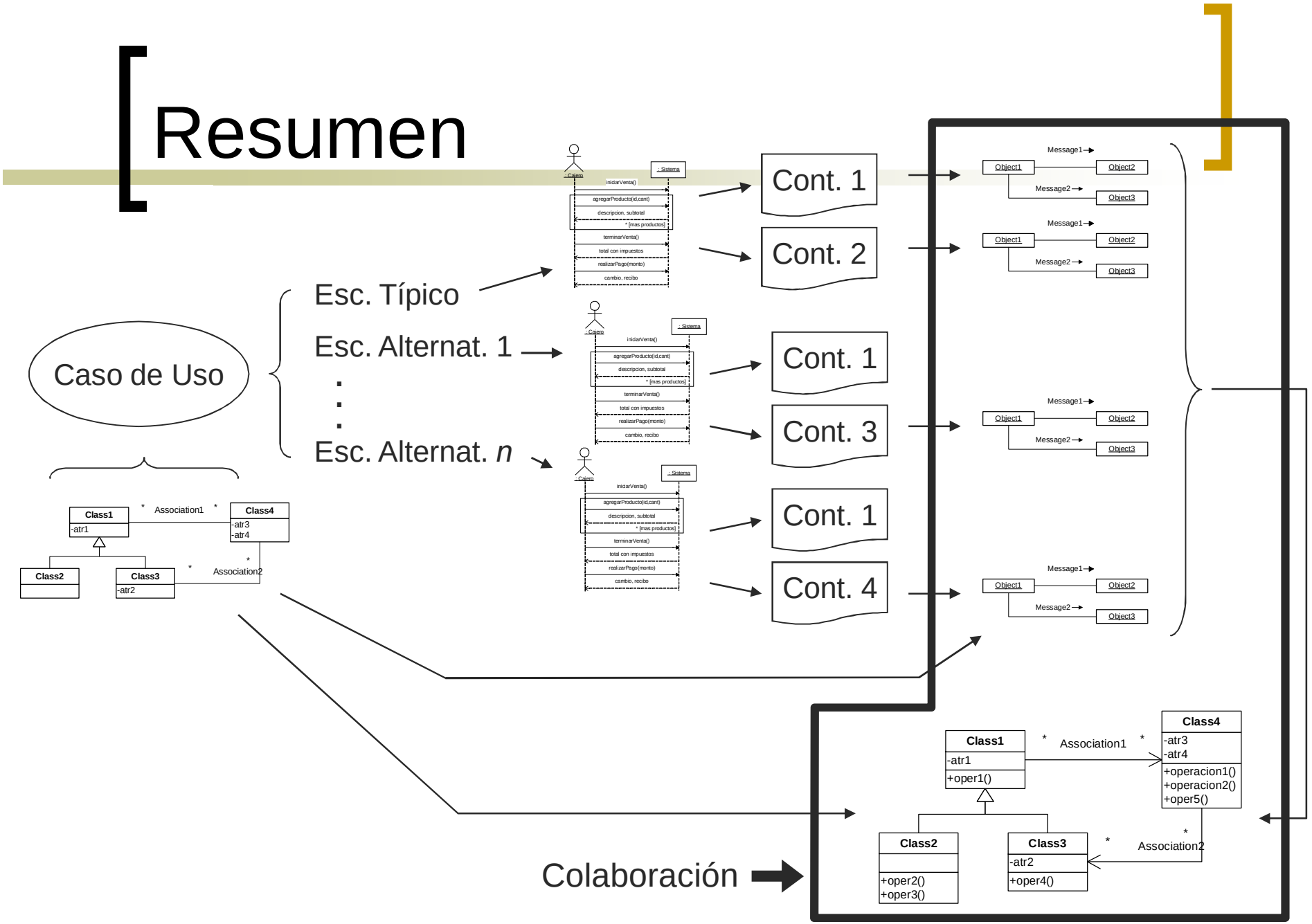
Modelo de Diseño (3)

- Contenido
  - **Introducción:** Breve descripción que sirve como introducción al modelo
  - **Clases:** Las clases del modelo
  - **Interfaces:** Las interfaces del modelo

Modelo de Diseño (4)

- Contenido (cont.)
  - **Relaciones:** Las relaciones del modelo entre clases e interfaces
  - **Colaboraciones:** Las realizaciones de casos de uso del modelo
  - **Diagramas:** Representación de los elementos del modelo

Resumen


Caso de Uso
Esc. Típico
Esc. Alternat. 1
Esc. Alternat. n
Cont. 1
Cont. 2
Cont. 1
Cont. 3
Cont. 1
Cont. 4
Class1
-atr1
Association1
Class4
-atr3
-atr4
Class2
Class3
-atr2
Association2
Message1
Message2
Object1
Object2
Object3
Colaboración
+oper1()
+operacion1()
+operacion2()
+oper5()
+oper2()
+oper3()
-atr2
+oper4()