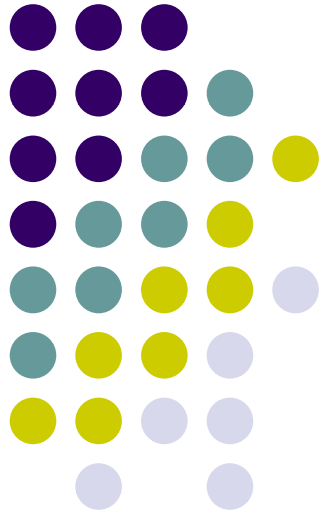
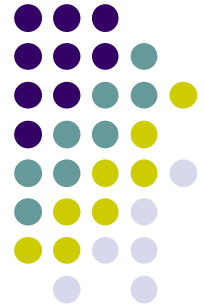
Metodologías de Diseño y Programación

Análisis

Especificación del Comportamiento del Sistema

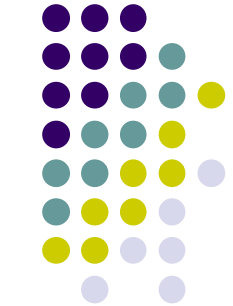


Contenido



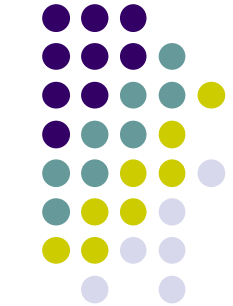
- Introducción
- La Clase Sistema
- Interacciones con el Sistema
- Contratos de Software

Introducción



- Durante esta actividad de análisis se busca describir en forma precisa cuál debe ser el comportamiento esperado del sistema
- Se trabaja sobre el Modelo de Casos de Uso
  - Viendo al sistema como una unidad
  - Se definen los protocolos que caractericen el uso del sistema por parte de los actores en cada escenario de los casos de uso
  - El comportamiento completo del sistema es especificado al especificar cada mensaje de los protocolos

Introducción (2)



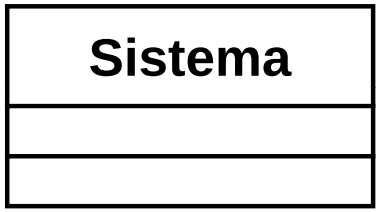
- Cada escenario de los casos de uso a analizar es entendido en términos de una interacción entre los actores involucrados y el sistema
- Al describir el significado de cada uno de los mensajes identificados en cada interacción se está especificando el comportamiento del sistema
- Nos enfocamos en qué es lo que debe hacer el sistema ante cada mensaje
- La forma en cómo el sistema resuelve internamente un mensaje será definida durante la etapa de diseño

La Clase Sistema

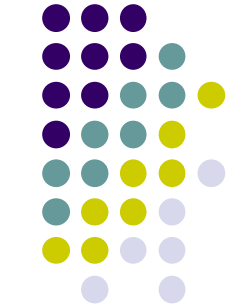
- Durante esta actividad el sistema será considerado como un objeto
  - Que es instancia de una clase
  - Que tiene operaciones (puede recibir mensajes)
  - Que tiene un estado
- En todo Modelo de Casos de Uso se asume que existe una clase Sistema
- Existe una única instancia de esta clase y representa al “sistema entero”

La Clase Sistema (2)

- Existe una **única instancia** de esta clase la cual representa al “sistema entero”



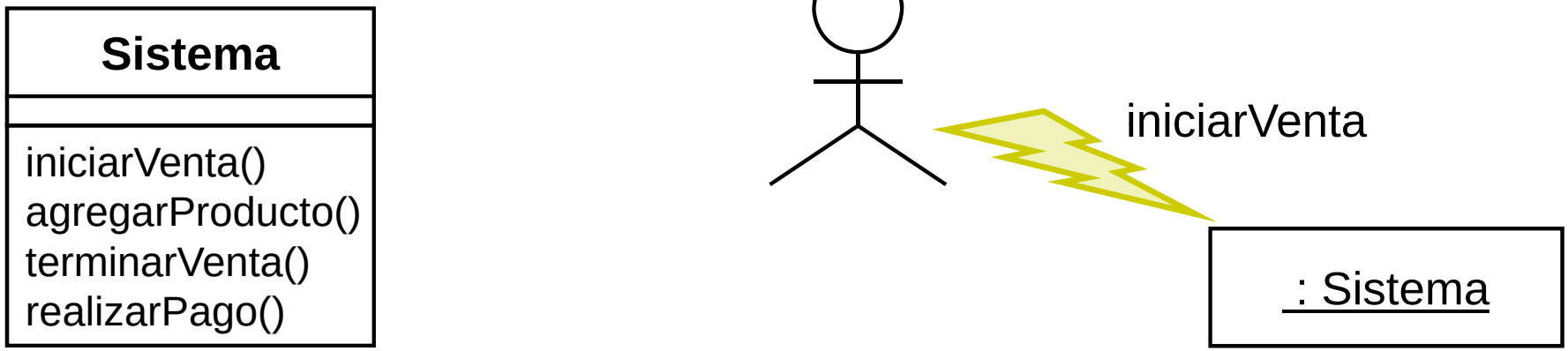
La Clase Sistema (3)



- Las **operaciones** de esta clase permiten que el sistema reciba mensajes de los actores
  - Se identifican al definir los protocolos que representan los escenarios de los diferentes casos de uso
  - Durante el análisis no se busca diseñarlas
  - Su semántica es definida en términos del efecto que deben tener sobre el estado del sistema

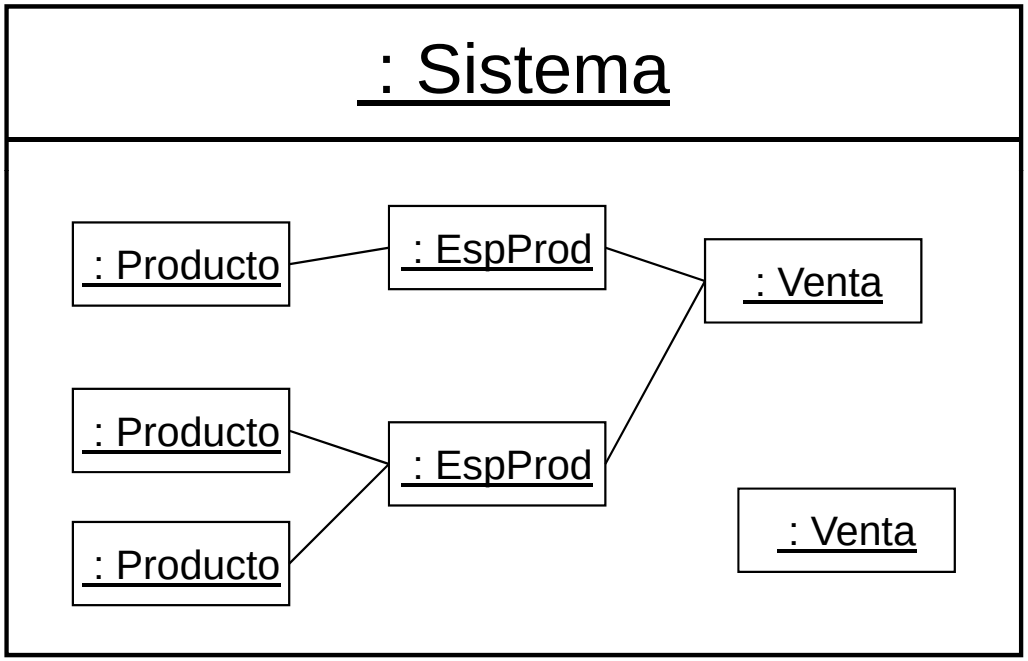
La Clase Sistema (4)

- Un actor puede enviar mensajes al sistema “invocando” sus operaciones



La Clase Sistema (5)

- En esta actividad el **estado** del sistema se asume como una configuración de objetos válida respecto al Modelo de Dominio



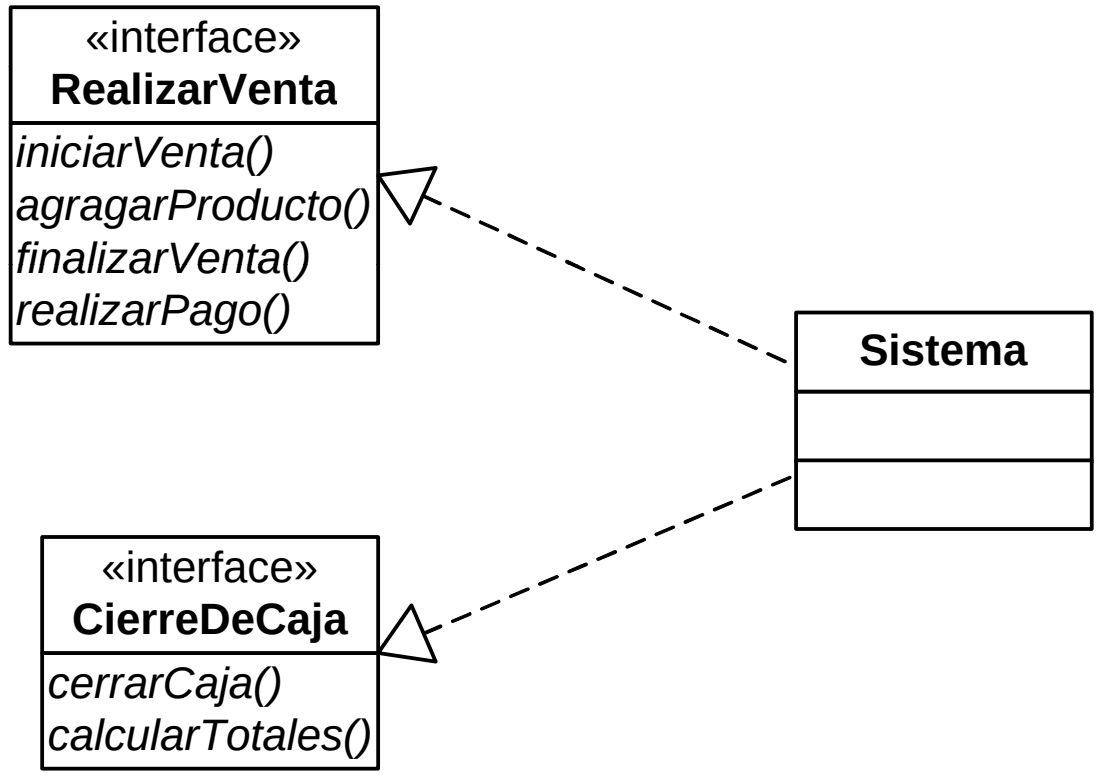
La Clase Sistema (6)

- Dado que no todos los actores participan en todos los casos de uso la visibilidad sobre las operaciones del sistema debe ser limitada
- La clase sistema podría realizar diferentes interfaces
- Cada interfaz contendría las operaciones utilizadas en un (conjunto de) caso(s) de uso
- Las operaciones se encontrarían organizadas y los actores verían al sistema a través de las interfaces que corresponden

La Clase Sistema (7)

El actor Cajero usará al sistema solamente a través de esta interfaz

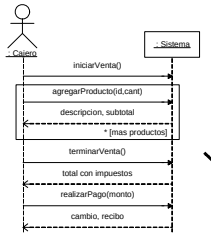
El actor Supervisor usará al sistema solamente a través de esta interfaz



Interacciones con el Sistema

- Los casos de uso describen la forma en que actores utilizan al sistema para cumplir con sus objetivos
- Es necesario expresar estas ideas desde un punto de vista técnico
- Para ello se definen protocolos que determinan la interacción entre los actores y el sistema en cada escenario de cada caso de uso
  - Cada protocolo es expresado mediante un Diagrama de Secuencia del Sistema (DSS)

Interacciones con el Sistema (2)



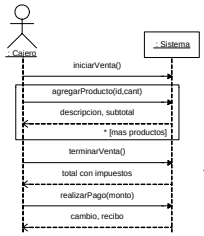
Caso de Uso 1

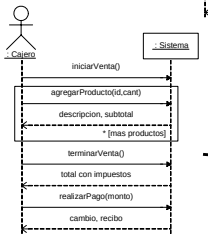
Esc. Típico

Esc. Alternativo 1

.
.
.

Esc. Alternativo *n*





Los DSSs se incluyen en la secc. "Comportamiento" del modelo

Modelo de Casos de Uso

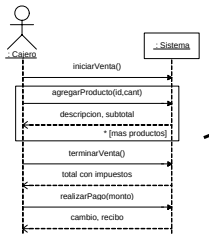
Caso de Uso 2

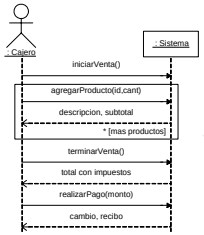
Esc. Típico

Esc. Alternativo 1

.
.
.

Esc. Alternativo *m*

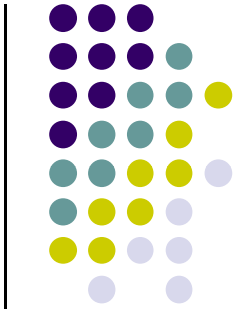




Un DSS que define la interacción entre los actores y el sistema en el escenario dado

Interacciones con el Sistema

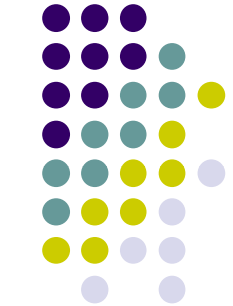
Eventos del Sistema



- Un evento del sistema
  - Es un estímulo externo,
  - Es generado por un actor, y
  - Ante el cual el sistema debe reaccionar
- Las acciones de los actores (sobre el sistema) descritas en los casos de uso sugieren los eventos del sistema
- Es necesario considerar la definición de evento del sistema para identificarlos

Interacciones con el Sistema

Eventos del Sistema (2)



- Ejemplo
  - "El Cliente llega a la caja con artículos para comprar"
    Es un evento externo pero no afecta al sistema
    ⇨ No es un evento del sistema
  - "El Cajero ingresa el identificador del producto"
    Es un estímulo externo generado por un actor ante el cual el sistema debe reaccionar
    ⇨ Es un evento del sistema

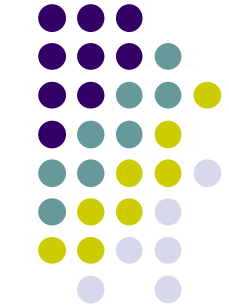
Interacciones con el Sistema

Operaciones del Sistema

- Los eventos del sistema (generalmente) disparan una operación del sistema
- Estas operaciones son ejecutadas por la “instancia sistema” en resupuesta a la ocurrencia de un evento del sistema
- Las operaciones del sistema de un escenario de un caso de uso permiten definir la interacción entre los actores y el sistema

Interacciones con el Sistema

Operaciones del Sistema (2)



- Las operaciones del sistema pueden tener asociados parámetros
- Ejemplo
  - "El Cajero ingresa el identificador del producto" representa un evento que dispara la operación

```
void agregarProducto(String ident)
```

  - "El Cajero … hasta terminar los productos" representa un evento que dispara la operación

```
void terminarVenta()
```

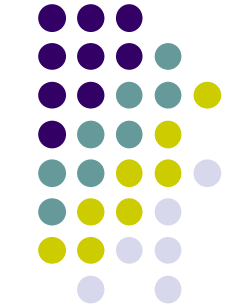
Interacciones con el Sistema

Diag. de Secuencia del Sistema

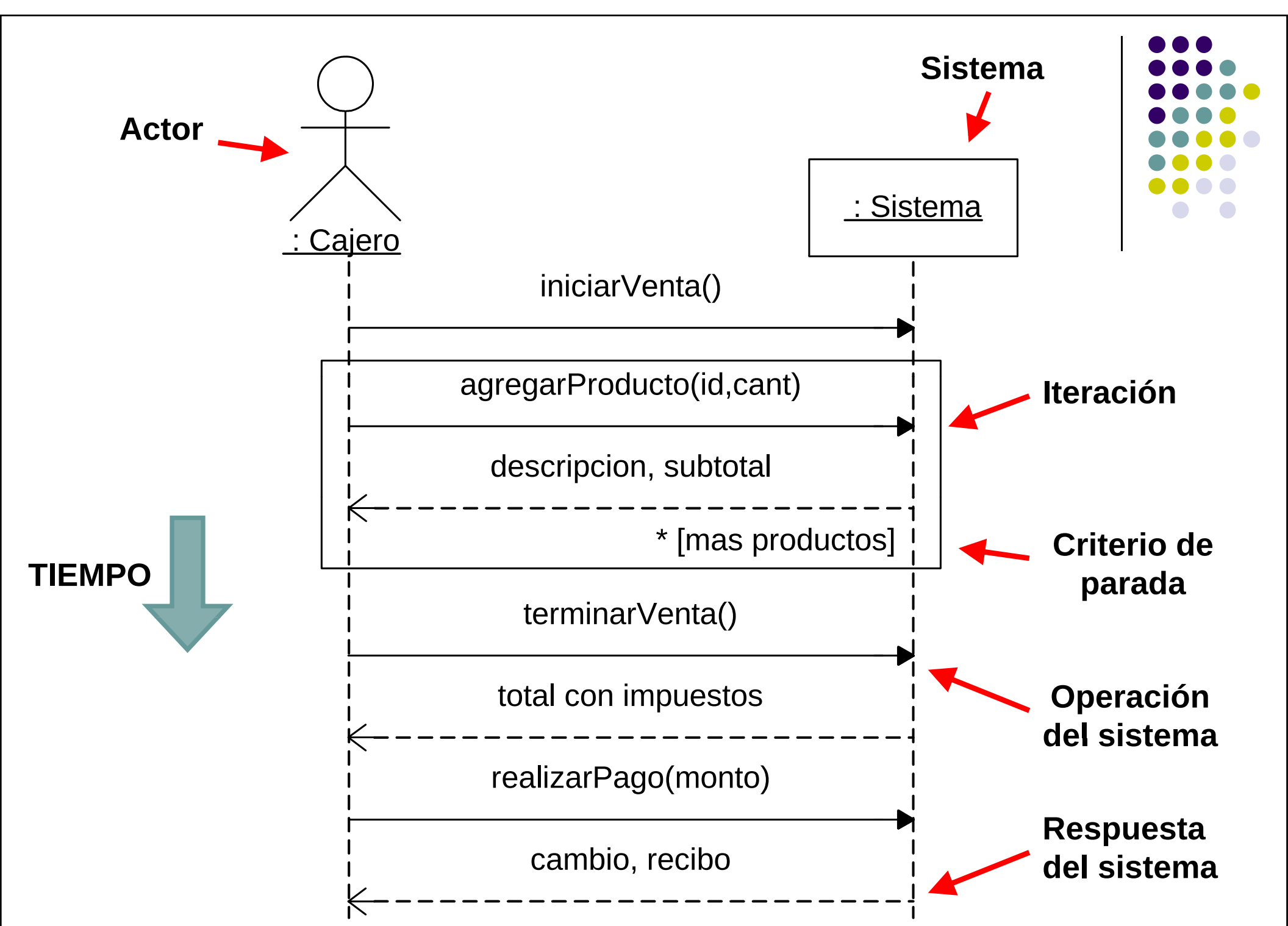
- Es un artefacto incluido en el Modelo de Casos de Uso que define e ilustra la interacción entre los actores y el sistema en un escenario de un caso de uso
- Incluye
  - Una instancia representando a cada participante (sistema y actores)
  - Los mensajes enviados entre ellos en el escenario correspondiente (con sus respuestas)
- Usualmente se define
  - Uno para el escenario típico de un caso de uso
  - Uno (opcionalmente) para cada escenario alternativo de interés

Interacciones con el Sistema

Diag. de Secuencia del Sistema (2)

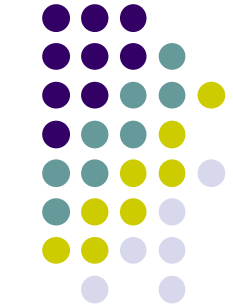


- Los diagramas de secuencia del sistema definen la conversación entre los actores y el sistema pero enfocándose en los mensajes que el sistema recibe
- Es posible incluir además mensajes enviados desde el sistema hacia los actores
  - Sin embargo esto no forma parte del conjunto de servicios que el sistema brinda (y cuya especificación es el objetivo de la presente actividad)


Actor
: Cajero
Sistema
: Sistema
iniciarVenta()
agregarProducto(id,cant)
descripcion, subtotal
* [mas productos]
Iteración
Criterio de parada
TIEMPO
terminarVenta()
total con impuestos
Operación del sistema
realizarPago(monto)
cambio, recibo
Respuesta del sistema

Interacciones con el Sistema

Sugerencias



- Definición de un DSS

1. Incluir una instancia que represente al sistema como una unidad
2. Identificar cada actor que participe en el escenario considerado e incluir una instancia para cada uno
3. De la descripción del caso de uso identificar aquellos eventos que los actores generen y sean de interés para el sistema e incluir cada uno de ellos como un mensaje
4. Opcionalmente, incluir junto a cada mensaje una descripción

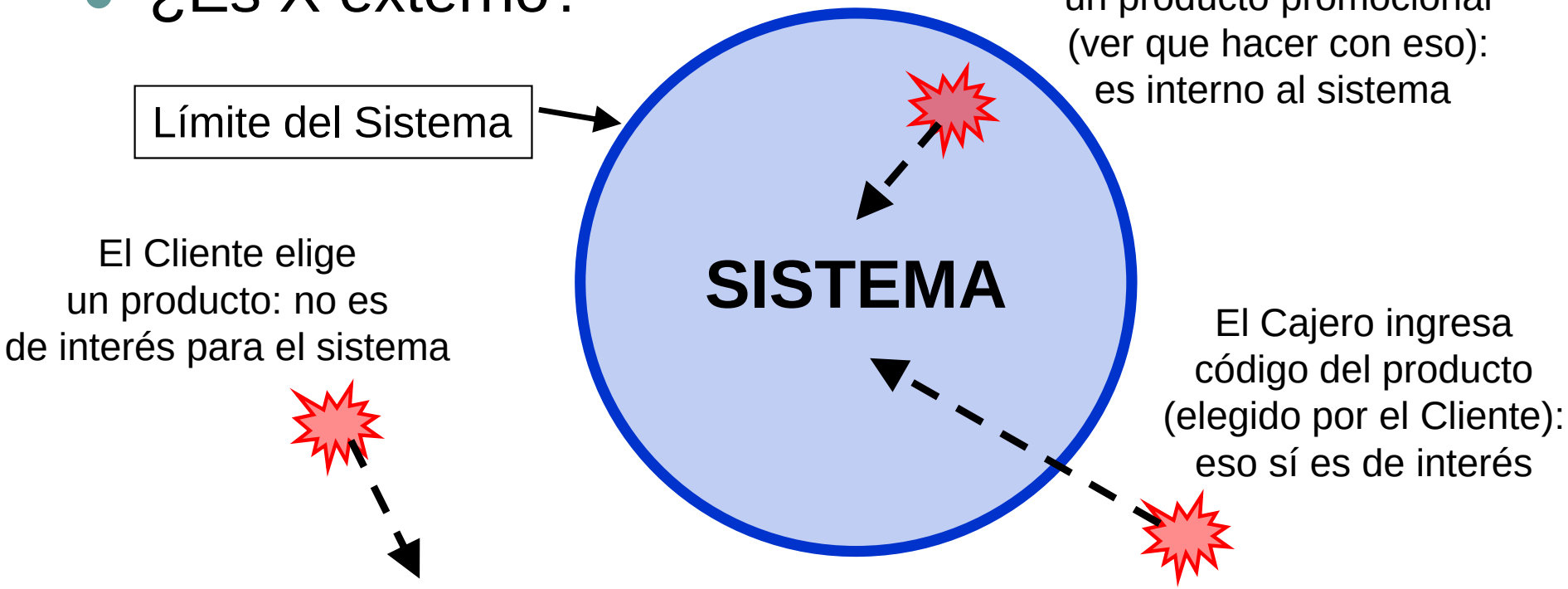
Interacciones con el Sistema

Sugerencias (2)

- Límite del sistema
  - Para identificar eventos del sistema es útil pensar en el límite del sistema
  - El límite suele determinarse para que coincida con el sistema de software (y el de hardware también)
  - Buscar aquello que ocurra fuera de ese límite y que además lo atraviese

Interacciones con el Sistema
Sugerencias (3)

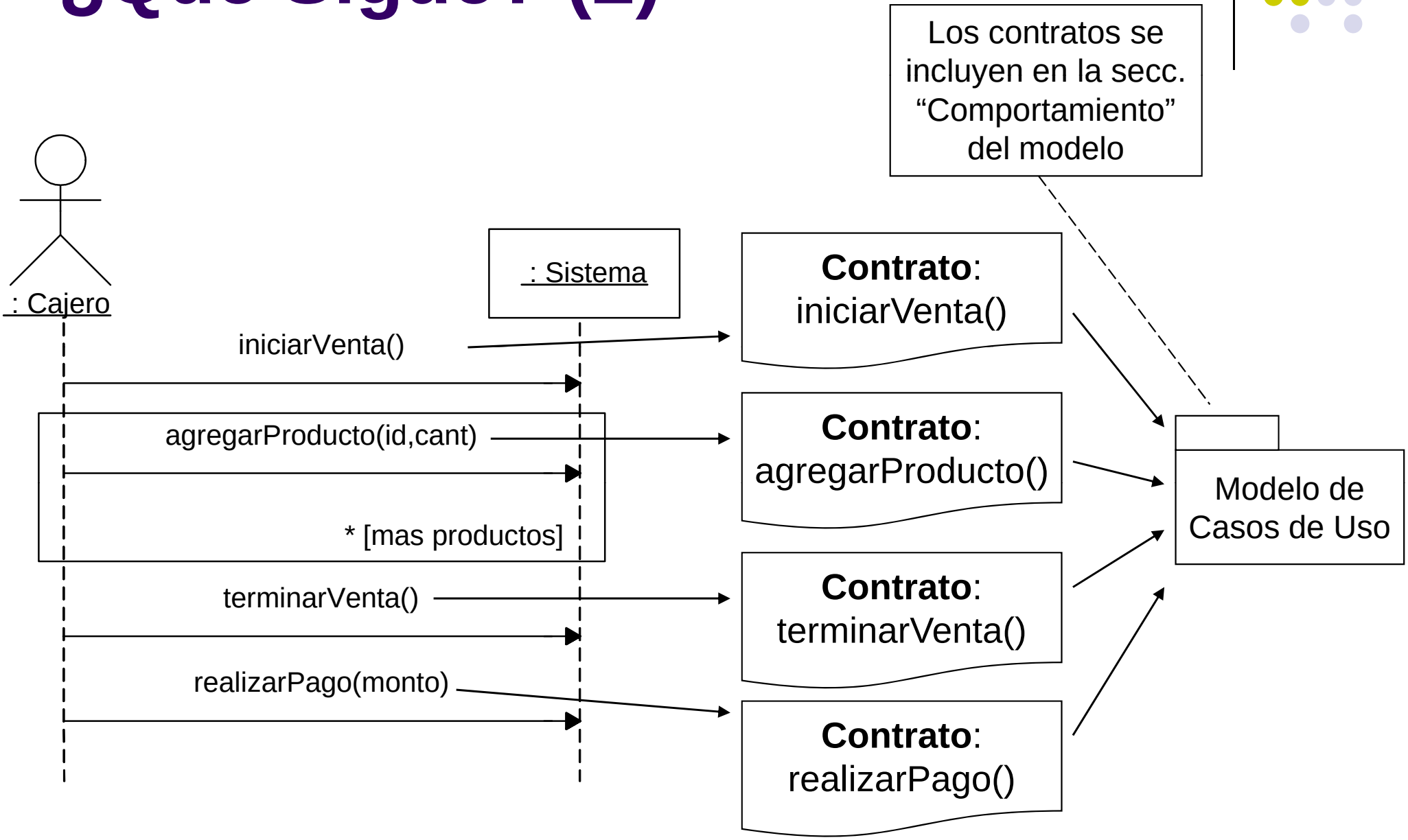
- Límite del sistema
  - ¿Es responsabilidad del sistema reaccionar ante el evento X?
  - ¿Es X externo?



¿Qué Sigue?

- Una vez identificadas las operaciones del sistema es posible especificar su comportamiento
- Esta especificación expresa el efecto que una operación tendrá sobre el sistema
- Para ello se realizará un Contrato de Software para cada operación del sistema

¿Qué Sigue? (2)

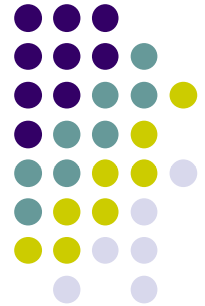

Los contratos se incluyen en la secc. “Comportamiento” del modelo
: Cajero
: Sistema
iniciarVenta()
agregarProducto(id,cant)
* [mas productos]
terminarVenta()
realizarPago(monto)
Contrato: iniciarVenta()
Contrato: agregarProducto()
Contrato: terminarVenta()
Contrato: realizarPago()
Modelo de Casos de Uso

Contratos de Software

- Un contrato de software especifica el comportamiento o efecto de una operación
- La especificación es declarativa y no imperativa
- Esta técnica está basada en las ternas de Hoare en las que
  - Se describen propiedades del resultado, en lugar de
  - Dar un conjunto de pasos o instrucciones que indiquen cómo calcularlo

Contratos de Software

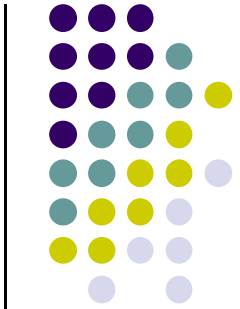
Ternas de Hoare



- Las ternas de Hoare tienen la forma {P} S {Q} donde
  - S es el programa u operación que se desea especificar
  - P (precondición) expresa condiciones que es necesario asumir para la ejecución de S
  - Q (postcondición) expresa condiciones que deben ser satisfechas por el resultado de la ejecución de S
- A partir de P y Q es posible diseñar una implementación para S

Contratos de Software

Ternas de Hoare (2)



- Ejemplo
  - Sean X y Y valores enteros:

$$\{x=X \wedge y=Y \wedge y>0\}$$
$$S$$
$$\{x=y*q+r \wedge r<x \wedge r<y \wedge r\geq 0\}$$

  - Esta especificación no dice cómo es el programa S sin embargo indica qué es lo que hace

Contratos de Software

Ternas de Hoare (3)

- Una implementación de S podría ser

```
q := x div y;
r := x mod y;
```

- Notar que muchas otras implementaciones podrían satisfacer la especificación
- En general se busca el programa que haciendo lo menos posible satisfaga la especificación

Contratos de Software

Enfoque de Contratos

- El contrato de una operación es un contrato entre partes
  - Consumidor de la operación: quién la invoca
  - Proveedor de la operación: quién la implementa
- Determina derechos y obligaciones para cada una de las partes

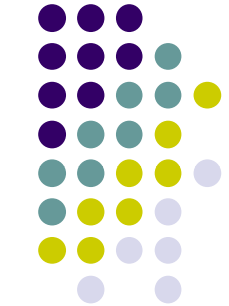
Contratos de Software

Enfoque de Contratos (2)

	Obligaciones	Derechos
Consumidor	Satisfacer precondición	Obtener la postcondición satisfecha
Proveedor	Satisfacer postcondición	Procesamiento más simple al poder asumir como satisfecha la precondición

Contratos de Software

Enfoque de Contratos (3)



- El Consumidor se compromete a satisfacer la precondición al invocar la operación
  - Si la satisface: tiene derecho a exigir que la postcondición se satisfaga
  - Si no la satisface: no se le garantiza la correctitud del resultado de la invocación
- Por esta razón es responsabilidad del Consumidor saber cuándo invocar a la operación (y manejar en forma adecuada el resultado)

Contratos de Software

Enfoque de Contratos (4)

- El Proveedor se compromete a satisfacer la postcondición al finalizar la operación solamente cuando la precondición fue satisfecha al momento de la invocación
- El compromiso no comprende el caso en que la precondición no fue satisfecha
  - En ese caso el Proveedor puede devolver un valor arbitrario y el Consumidor tiene que aceptarlo y saber qué hacer con él

Contratos de Software

Enfoque de Contratos (5)

- Ejemplo “Autorización de Documento”
  - Precondición: el documento está en la oficina antes de la hora 10
  - Postcondición: el documento está firmado por el Gerente a la hora 18
- Consumidor
  - “Yo te traigo el documento a las 10, pero a las 18 lo quiero firmado”
- Proveedor
  - “Yo te hago firmar el documento para las 18, pero lo necesito antes de las 10”

Contratos de Software

Enfoque de Contratos (6)

- Ejemplo (cont.)
  - Caso 1 (ambos cumplen)
    - El documento llegó a las 9:45
    - A las 18 estaba firmado
  - Caso 2 (el consumidor no cumple)
    - El documento llegó a las 11:20
    - A las 18 no estaba firmado
  - Caso 3 (el consumidor cumple pero el proveedor no)
    - El documento llegó a las 9:10
    - A las 18 no estaba firmado

El proveedor no tiene que cumplir

Esto denota un bug en la implementación del proveedor

Contratos de Software

Enfoque de Contratos (7)

- Consumidor
  - Prefiere precondiciones débiles: implica menos trabajo
  - Prefiere postcondiciones fuertes: implica más resultados
- Proveedor
  - Prefiere precondiciones fuertes: implica menos preocupaciones
  - Prefiere postcondiciones débiles: implica menos trabajo

Contratos de Software

Enfoque de Contratos (8)

- Precondición
  - Es a lo que debe acceder el Consumidor para obtener el resultado deseado
  - Es lo que debe exigir el Proveedor para llegar al resultado
- Postcondición
  - Es a lo que accederá el Consumidor
  - Es a lo que se compromete el Proveedor

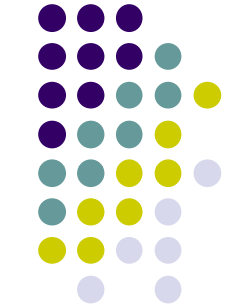
Contratos de Software

Enfoque de Contratos (9)

- Tanto las Pre- como las Post- las determina el Proveedor
- El Consumidor
  - Viendo la Post- sabe qué va a obtener (sin saber cómo)
  - Viendo la Pre- sabe a cambio de qué obtiene el resultado

Contratos de Software

Contratos de Operaciones



- Los contratos se pueden realizar para operaciones de cualquier tipo de clase
- En esta actividad las realizaremos para operaciones del sistema
- Para una operación X tendremos {P}S{Q}
  - P es la precondición de X (especificada)
  - S es el programa que implementa X (a ser diseñado más adelante en la etapa de Diseño)
  - Q es la postcondición de X (especificada)

Contratos de Software

Contratos de Operaciones (2)

- ¿Quién utiliza el contrato (partes P y Q) de una operación?
  - Un diseñador de nuestro equipo que deba diseñar S
    - Para saber qué es lo que tiene que lograr su diseño de la operación
    - En función de lo anterior para decidir cómo será el diseño de la operación (parte S)
  - Un desarrollador de otro equipo que deba invocar la operación (el diseño o implementación de S no es su responsabilidad)
    - Para saber qué es lo que la operación hace sin tener que ver el diseño o la implementación de S

Contratos de Software

Condiciones

- ¿En qué términos se expresan las pre- y postcondiciones?
  ¿Y para el caso particular de operaciones del sistema?
- En términos generales estas condiciones refieren al estado del sistema antes y después de la invocación a la operación
  - Las precondiciones refieren además a los argumentos de la operación

Contratos de Software

Condiciones (2)

- Las Precondiciones refieren al momento previo a la invocación y expresan condiciones sobre
  - Los valores de los parámetros de la operación
  - El estado del sistema
    - Que un cierto objeto existe
    - Que un cierto objeto no existe
    - Que ciertos objetos están conectados
    - Que ciertos objetos no están conectados
    - Propiedades sobre valores de atributos de objetos existentes

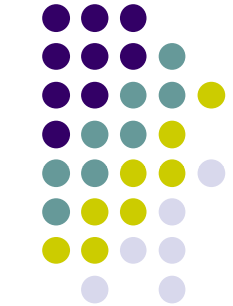
Contratos de Software

Condiciones (3)

- Las Postcondiciones refieren al momento posterior a la invocación y también expresan condiciones sobre el estado del sistema
  - Que un cierto objeto existe
  - Que un cierto objeto no existe
  - Que ciertos objetos están conectados
  - Que ciertos objetos no están conectados
  - Propiedades sobre valores de atributos de objetos existentes

Contratos de Software

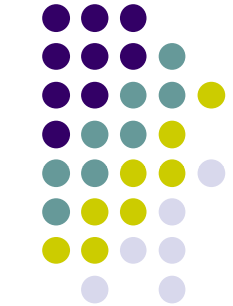
Condiciones (4)



- De un contrato para la operación X se derivan cambios en el estado del sistema
  - La creación de objetos
  - La destrucción de objetos
  - La conexión de objetos
  - La desconexión de objetos
  - La modificación del valor de atributos de objetos
- Estos cambios son el efecto de la operación X
- Un cambio se deriva al comparar la precondición con la postcondición

Contratos de Software

Condiciones (5)



- De un contrato que incluya

  Pre: El objeto *x* no existe

  Post: El objeto *x* existe

  - Se deriva que la operación crea el objeto *x*

- De un contrato que incluya

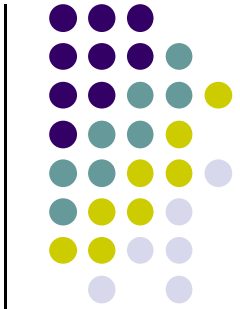
  Pre: El objeto *x* existe

  Post: El objeto *x* no existe y todos los objetos que estaban conectados a él ya no lo están

  - Se deriva que la operación destruye el objeto *x*

Contratos de Software

Condiciones (6)



- De un contrato que incluya

  Pre: Los objetos x_1 y x_2 no están conectados

  Post: Los objetos x_1 y x_2 están conectados

  - Se deriva que la operación conecta a x_1 y x_2

- De un contrato que incluya

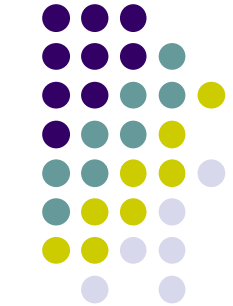
  Pre: Los objetos x_1 y x_2 están conectados

  Post: Los objetos x_1 y x_2 no están conectados

  - Se deriva que la operación desconecta a x_1 y x_2

Contratos de Software

Condiciones (7)



- De un contrato que incluya

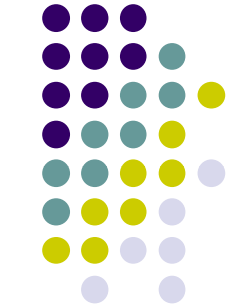
  Pre: El objeto *x* existe (y eventualmente que el valor de su atributo *a* es v_1)

  Post: El atributo *a* del objeto *x* tiene el valor v_2 (el cual puede depender de v_1)

  - Se deriva que la operación cambia el valor del atributo *a* del objeto *x*

Contratos de Software

Condiciones (8)

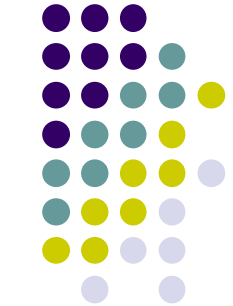


- Ejemplo para operación `contratar(banco,cod)`

> Pre: No existe un objeto de clase Empleado con el valor *cod* en el atributo 'codigo', y existe un objeto de clase Empresa con valor *banco* en el atributo 'nombre'
>
> Post: Existe un nuevo objeto de clase Empleado con el valor *cod* en el atributo 'codigo' que está conectado a uno de clase Empresa que tiene el valor *banco* en el atributo 'nombre'

  - De esto se puede derivar que la operación crea al objeto de clase Empleado y lo conecta con el de clase Empresa

Contratos de Software
Condiciones (9)



σ_1

: Empresa
nombre = “BROU”

σ_2

: Empleado
codigo = 6150

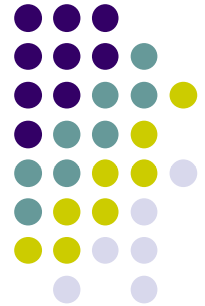
: Empresa
nombre = “BROU”

Se pasa del estado σ_1 al estado σ_2 mediante la ejecución de

```
contratar(“BROU”, 6150);
```

Contratos de Software

Condiciones (10)



- Notar que el contrato NO dice cómo debe implementarse la operación del sistema `contratar()`
- Expresa condiciones sobre el estado inicial y sobre el estado final que indican qué es lo que la operación hace

Contratos de Software

Snapshots

- Para elaborar las precondiciones y postcondiciones que componen el contrato de una operación es útil emplear figuras como las del ejemplo anterior
- Estas figuras se denominan *snapshots*
- Un snapshot es un diagrama de instancias que muestra una configuración de objetos (un estado)
- Todo snapshot debe ser válido respecto al Modelo de Dominio
  - Mostrar solamente instancias de las clases y asociaciones contenidas en él
  - Respetar todas las restricciones estructurales (multiplicidades)
  - Respetar todos los invariantes

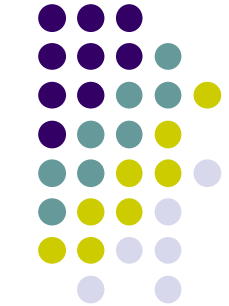
Contratos de Software

Snapshots (2)

- Para elaborar un contrato se puede utilizar snapshots
  - Uno que sirva de ejemplo representativo del estado del sistema antes de la invocación
  - Otro que sirva de ejemplo representativo del estado del sistema después de la invocación
- Estudiando el par de snapshots puede resultar más simple expresar las pre- y postcondiciones correspondientes
- Snapshots son ejemplos: no son ni pre- ni post-

Contratos de Software

Estructura de Contratos



- Un contrato es un artefacto textual que se incluye en la sección ‘Comportamiento’ del Modelo de Casos de Uso
- Está estructurado de la siguiente forma
  - **Nombre:** Cabezal sintáctico de la operación
  - **Responsabilidades:** Descripción de las responsabilidades, una idea de lo que debe realizar la operación
  - **Tipo:** Tipo del cual la operación es propiedad (en nuestro caso es siempre ‘Sistema’)

Contratos de Software

Estructura de Contratos (2)

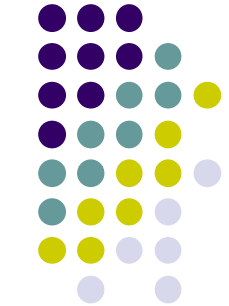
- Estructura (cont.)
  - **Referencias cruzadas:** Caso(s) de Uso a los que pertenece la operación
  - **Notas:** Comentarios generales
  - **Salida:** Resultado de la operación (sólo si es una función)
  - **Precondición:** Descripción del estado de la instancia a la que se le aplicará la operación, y otras condiciones que sea necesario asumir previo a la aplicación

Contratos de Software

Estructura de Contratos (3)

- Estructura (cont.)
  - **Postcondición:** Descripción del estado de la instancia a la que se le aplicó la operación
  - **Snapshots:** (Opcional)
    - Pares de snapshots que ejemplifiquen el estado de la instancia a la que se le aplicó la invocación, previo y posterior a la invocación
    - La invocación concreta que produce el cambio ejemplificado (mostrando los parámetros efectivos)

¿Qué Sigue?



- Hasta el momento se tienen identificadas y especificadas las operaciones del sistema para todos los casos de uso definidos
- Es posible ahora realizar un diseño en el que
  - Se identifiquen los objetos que realmente participarán en la solución
  - Se definan interacciones entre dichos objetos tal que cada una cumpla un contrato correspondiente a una operación del sistema