

### Referencia

- El Lenguaje Unificado de Modelado. Grady Booch, James Rumbaugh e Ivar Jacobson. Addison Wesley, 1999.
  - □ Capítulos 16 y 17



# Referencia requerimientos no funcionales

- Object Oriented Software Engineering.
   Bernd Bruegge y Allen H.Dutoit.
   Prentice Hall, 2000
  - □ Capítulo 4, pág. 100–106, 118-119
- Software Requirements. Karl.
   E.Wiegers. Microsoft Press, 1999.
  - □ Capítulo 9, pág. 153-162
  - □ Capítulo 11

3



### Agenda

- Requerimientos funcionales
  - □ Levantamiento de requerimientos
  - □ Casos de Uso (Requerimientos Funcionales)
- Requerimientos no funcionales
  - □ Diferencias requerimientos funcionales, no funcionales y pseudo requerimientos
  - □ Clasificación de los requerimientos no funcionales y pseudo requerimientos



### Requerimientos

- Un requerimiento es una característica que el sistema DEBE tener o es una restricción que el sistema DEBE satisfacer para ser aceptada por el cliente.
- Levantamiento de requerimientos es la especificación del sistema en términos que el cliente entienda, de forma que se constituya en el contrato entre el cliente y los desarrolladores.



### Requerimientos funcionales

- Describen la interacción entre el sistema y su ambiente independientemente de su implementación.
- El ambiente incluye al usuario y cualquier otro sistema externo que interactúa con el sistema.



### Levantamiento de

### Requerimientos

- Para el levantamiento se pueden utilizar dos conceptos:
  - □ Escenarios
    - Describen un ejemplo del uso del sistema en términos de una serie de interacciones entre el usuario y el sistema
  - □ Casos de uso
    - Es una abstracción que describe una clase de escenarios.
  - □ Ambos deben ser escritos en lenguaje natural para que sean entendidos por el usuario.

7



### **Actividades**

- Identificación de actores
  - □ Diferentes tipos de usuario (no personas en particular)
- Identificación de escenarios
  - Observar al usuario y desarrollar un conjunto de escenarios detallados para la funcionalidad típica que debe proveer el sistema.
- Identificación de casos de uso
  - ☐ Son abstracciones que describen todos los casos posibles descritos en los escenarios.



### **Actividades**

- Identificación de relaciones entre casos de uso
  - ☐ Eliminar redundancias entre los casos de uso.
  - ☐ Hacer que la especificación del sistema sea consistente.

9



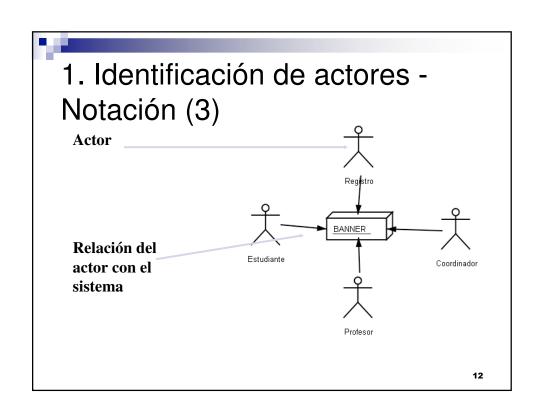
### 1. Identificación de actores (1)

- Un actor representa un conjunto coherente de roles, que son jugados por una persona, un dispositivo de hardware o incluso otro sistema al interactuar con nuestro sistema.
- Se identifican son roles, es decir usuarios que realizan un conjunto de actividades definidas respecto a la funcionalidad del sistema.



### Identificación de actores -Preguntas (2)

- Cuáles usuarios están soportados por el sistema para desarrollar su trabajo?
- Cuáles usuarios ejecutan las funciones principales del sistema?
- Cuáles usuarios desempeñan funciones secundarias, como mantenimiento y administración?
- El sistema interactúa con hardware externo o software?





# 2. Identificación de escenarios(1)

- Un escenario es una descripción narrativa de cómo las personas hacen las cosas y muestran como tratarían de hacer uso del sistema.
- El escenario es una descripción concreta, enfocada e informalmente descrita de una única característica del sistema desde el punto de vista de un único actor.

Nombre del escenario	Consultar listado de cursos
Instancias de los usuarios participantes	Pepito: Profesor
Flujo de eventos	<ol> <li>Pepito ingresa al sistema indicando sus datos.</li> <li>El sistema indica un menú dando cada una de las posibilidades del sistema.</li> <li>Pepito indica que quiere sacar un listado de un curso.</li> <li>El sistema solicita ingresar la información del código, sección y semetre de la materia.</li> <li>Pepito ingresa 21251, 02, 2001-1.</li> <li>El sitema devuelve la información correspondiente al curso indican el nombre, carnet, carrera y correo electrónico de cada uno de los alumnos.</li> </ol>



# 2. Identificación de escenarios(3)

- Escenarios actuales
- Escenarios visionarios
- Escenario de evaluación
- Escenarios de entrenamiento

15



### 2. Identificación de escenarios (4)

- Cuáles son las tareas que los actores requieren desempeñar con el sistema?
- Qué información requiere el actor? Quién crea esa información? Puede ser modificada o eliminada? Por quién?
- Qué cambios externos necesita el actor que el sistema debe informar? Qué tan seguido? Cuándo?
- De cuáles eventos necesita el actor ser informado? Con qué periodicidad?



# Identificación de casos de uso (1)

- Capturar el comportamiento deseado del sistema en desarrollo, sin tener que especificar cómo se implementa este comportamiento.
- Los casos de uso proporcionan un medio para que los desarrolladores, los usuarios finales del sistema y los expertos del dominio lleguen a una comprensión común del sistema.
- Un escenario es la instancia de un caso de uso.
- Los casos uso no deben ser excesivamente genéricos ni demasiado específicos.
- Requerimiento funcional del sistema

17



# 3. Identificación de casos de uso (2)

- Es una descripción de un conjunto de secuencias de acciones, incluyendo variantes, que ejecuta un sistema para producir un resultado observable de valor para un actor.
- Se utilizan verbos por lo general en infinitivo que representan las acciones del usuario con el sistema, por lo que siempre debe existir un tipo de usuario(actor) que lo utilice.

# 3. Identificación de casos de uso (3)

19

# 3. Identificación de casos de uso (4)

- Relaciones entre actores y casos de uso representan el flujo de la información durante el caso de uso.
- Representa que funcionalidad puede ser realizada por un actor en particular.
- También es un proceso cíclico donde cada vez se busca refinar cada vez más los casos de uso que finalmente responderá a los requerimientos funcionales.



# 3. Identificación de casos de uso (5)

- El comportamiento de un caso de uso se puede especificar describiendo un flujo de eventos en forma textual.
- Además de incluir cómo y cuándo empieza y acaba el caso de uso.
- Se incluye cuándo interactúa con los actores
- Indicar qué información se intercambian
- Se describe el flujo básico
- Se describe los flujos alternativos.

21



# 3. Identificación de casos de uso (6)

- Flujo de eventos principal
  - 1. El sistema pide al cliente un número de identificación personal.
  - 2. El cliente introduce su id.
  - 3. El sistema comprueba entonces la id. Para ver si es válido.
  - Si la identificación es válida el sistema acepta la entrada.



# 3. Identificación de casos de uso (7)

- Flujo de eventos excepcional
  - □ El cliente puede cancelar una transacción en cualquier momento, reiniciando el caso de uso.
  - □ No se efectúa ningún cambio en la cuenta del cliente.
- Flujo de eventos excepcional
  - □ Paso 2: El cliente puede borrar su id en cualquier momento antes de introducirlo.

23



# 3. Identificación de casos de uso (8)

- Flujo de eventos excepcional
  - □ Paso 4: Si el cliente introduce un id inválido, el caso de uso vuelve a empezar.
  - □ Paso 4: Si el cliente introduce tres veces seguidas un id inválido, el sistema cancela la transacción completa, impidiendo que el Cliente utilice el cajero durante 24 horas.



# 4. Identificar relaciones entre casos de uso(1)

### Extends

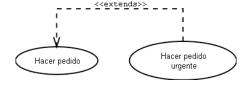
- □ Un caso de uso extiende otro caso de uso, si el caso de uso extendido incluye el comportamiento del otro bajo ciertas condiciones.
- ☐ Se utiliza para modelar la parte de un caso de uso que el usuario puede ver como comportamiento opcional del sistema.
- □ Se separa el comportamiento opcional del obligatorio.

25



# 4. Identificar relaciones entre casos de uso(2)

**Extends** 





# 4. Identificar relaciones entre casos de uso(3)

### Include

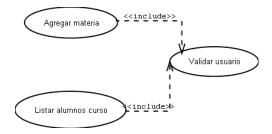
- □ Indica que en el flujo de eventos del caso de uso base se incluye el comportamiento del otro caso de uso.
- □ Factorizar comportamiento común (NO hacer descomposición funcional).
- SOLAMENTE se hace cuando la parte común es utilizada por otro caso de uso o cuando es utilizada por otro actor.

27



# 4. Identificar relaciones entre casos de uso(4)

### Include





# 4. Identificar relaciones entre casos de uso(5)

### Generalización

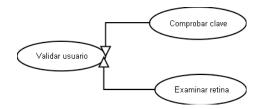
- Cuando algunos casos de uso tienen algo en común y puede ser abstraído a otro, mucho más general.
- □ El caso de uso hijo hereda el comportamiento y el significado del caso de uso padre.
- ☐ El hijo puede añadir o redefinir el comportamiento del padre.
- ☐ El hijo puede ser colocado en cualquier lugar donde aparezca el padre.

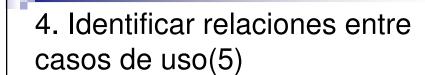
29



# 4. Identificar relaciones entre casos de uso(6)

### Generalización





- Comúnmente se utiliza la generalización para actores y no para casos de uso.
  - □ Los superactores y subactores interactúan con el caso de uso.
  - □ Existe una descripción considerable común entre los subactores que de otra manera se duplicaría.

31

# 4. Identificar relaciones entre casos de uso(7) Consignar Retirar Consultar Saldo Bloquear Cuenta



### Documentación del caso de uso

Identificador:	Indispensable/Deseable	Prioridad	
Nombre Caso de Uso:			
Autor:			
Fecha			
Categoría (Visible/No visible):	Actores involucrados:		
Resumen:			
Curso Básico Eventos:			
Caminos Alternativos:			
Caminos de Excepción:			
Puntos de Extensión:			
Pre - Condiciones:			
Post- Condiciones:			
Criterios de Aceptación			
Borrador de Interfaz Gráfica			

33



### Documentación del Caso de Uso

- Identificación del caso de uso: Unica
- Indispensable/Desable: Necesidad del requerimiento
- Prioridad: Alta/Media/Baja definida por el usuario
- Nombre de caso de uso: Iniciar con un verbo en infinitivo



# Documentación del Caso de Uso

- Autores: Persona(s) que elabora(ron) el formato.
- Fecha
- Descripción: Describe la interacción que ocurre en el caso de uso (contexto)
- Curso básico de eventos: Describe los pasos que los actores y el sistema deben seguir para completar la meta del caso de uso (No ocurre ningún error)

35



### Documentación del Caso de Uso

- Caminos alternativos: camino correcto que ocurre como parte integral del caso de uso.
- Caminos de excepción: Muestran procesamiento no común, en especial cuando ocurren errores.
- Puntos de extensión: Cuando se utiliza la relación de extends. Indica en que punto exacto se extiende la funcionalidad bajo ciertas condiciones



# Documentación del Caso de Uso

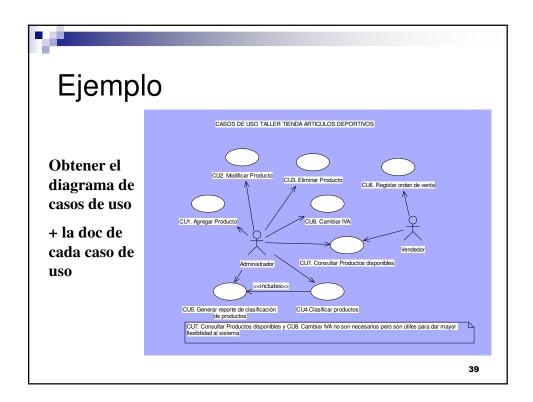
- Precondiciones: Cosas que han debido ocurrir antes de iniciar la interacción.
   Parte del contrato entre el caso de uso y el mundo de afuera.
- Postcondición: Cuando el caso de uso termina exitosamente se debe satisfacer.

37



### Documentación del Caso de Uso

- Criterios de aceptación: Condiciones para que el cliente acepte el requerimiento como válido.
- Borrador de interfaz: Prototipo que muestre como se observará el requerimiento





- Describen aspectos del sistema que son visibles por el usuario que no incluyen una relación directa con el comportamiento funcional del sistema.
- Los requerimientos no funcionales incluyen restricciones como el tiempo de respuesta(desempeño), la precisión, recursos consumidos, seguridad, etc.



### Pseudo Requerimientos

- Son requerimientos impuestos por el cliente que restringen la implementación del sistema.
- Ejemplos:
  - □ Lenguaje de implementación
  - □ Plataforma en que el sistema debe ser implementado
  - □ Requerimientos del proceso y documentación (utilización de un lenguaje formal)

41



### Requerimientos no funcionales

- Requerimientos de Interfaz externa
  - □ Interfaz de usuario
    - Estándar de GUI
    - Distribución de la pantalla
    - Restricciones de resolución
    - Estándares de botones, funciones o enlaces de navegación que aparecen en cada ventana
    - Teclas "shortcut"
    - Estándares de mensajes de error



- Requerimientos de Interfaz externa
  - □ Interfaces de hardware
    - Interfaces entre componentes de hardware y software del sistema
    - Ejemplos
      - □ Periféricos soportados
      - □ Naturaleza de la información
      - □ Protocolos de comunicación a utilizar

43



### Requerimientos no funcionales

- Requerimientos de Interfaz externa
  - □ Interfaces de Software
    - Conexiones entre el producto y software externo (identificado por nombre y versión)
    - Ejemplo
      - □ Bases de datos
      - □ Sistemas operativos
      - Legacy
    - Identificar la información que comparten los componentes



- Requerimientos de desempeño
  - □ Describir el desempeño para los escenarios
  - Describir el volumen o tiempo de utilización para saber que tan importante es.
  - □ Especificar el número de usuarios concurrentes
  - Especificar el número de operaciones concurrentes
  - □ Tiempos de respuesta
  - Restricciones de tiempo para sistemas de tiempo real

45



### Requerimientos no funcionales

- Requerimientos de tolerancia a fallas (safety)
  - □ Posibles pérdidas de información
  - □ Daño de información
  - ☐ Indicar acciones potencialmente peligrosas que deben ser prevenidas
  - ☐ Identificar políticas de mantenimiento de información
  - □ Identificar regulaciones



- Requerimientos de seguridad
  - □ Protección de la información
  - □ Utilización del producto
  - Definir la autenticación o autorización del ingreso los usuarios

47



### Requerimientos no funcionales

- Requerimientos de calidad del software (usuario)
  - □ Disponibilidad
  - □ Eficiencia en el manejo de recursos
  - ☐ Flexibilidad para adicionar requerimientos al producto
  - □ Integridad
    - Protegerse ante el daño de información
    - Protección ante virus
    - Proteger información importante



- Requerimientos de calidad del software(usuario)
  - □ Interoperabilidad
  - □ Confiabilidad
  - □ Robustez
  - □Usabilidad
    - "Amigable al usuario"
  - □ Instalación

4



### Requerimientos no funcionales

- Requerimientos de calidad del software (desarrollador)
  - Mantenibilidad
    - Estándares de documentación
    - Indentación
    - Metodología de diseño
    - Estructura de directorios
    - Documentos de diseño



- Requerimientos de calidad del software (desarrollador)
  - □ Portabilidad
  - □ Reusabilidad
  - □ Facilitar pruebas

51



### Requerimientos no funcionales

- Requerimientos operación
  - □ No aumentan la capacidad funcional
  - □ Permiten un mejor uso
    - Deshacer, rehacer, copiar, pegar
    - Configuración
    - Barras de herramientas, configurar menús, cambiar font
    - Sistema de ayuda

# Requerimientos no funcionales Restricciones de diseño relación con pseudo requerimientos

- □ Estilo de arquitectura
- □ Plataforma de operación
- □ Herramientas
- Restricciones de implementación relacionados con pseudo requerimientos
  - □ Lenguaje
  - □ Librerías
  - □ Plataforma de implementación

53

# Documentación del requerimiento no funcional | Nombre | Tipo: Necesario / no necesario | Crítico: Si/No | | Descripción | Criterios de Aceptación | Criterios | C