FRLP

Diseño de Sistemas

AyDOO utilizando lenguajes de modelado UML-OCL y aplicando proceso de desarrollo RUP, con soporte de herramientas CASE

Versión unificada y actualizada – Mayo/2016 – by Leopoldo Nahuel

2

FRLP

Diseño de Sistemas

Repaso de notación UML:

diagramas
elementos de los diagramas
vista que conforman los diagramas

Lenguaje Unificado de Modelado

¿ Qué es entonces UML ?

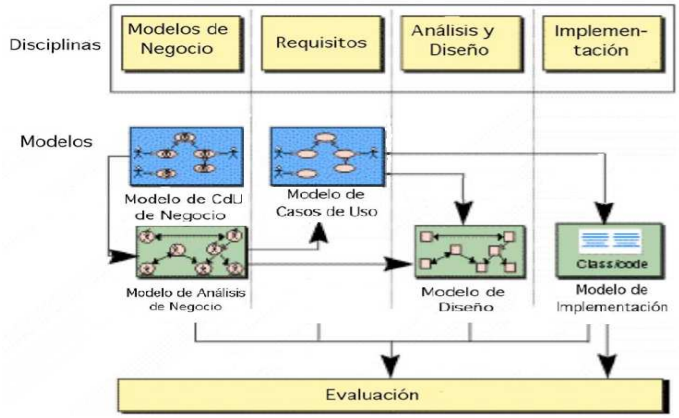
UML es un lenguaje gráfico estándar para:

- visualizar
- especificar
- construir
- documentar

los artefactos de un sistema software y/o hardware.

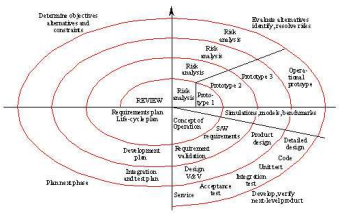
Lenguaje Unificado de Modelado

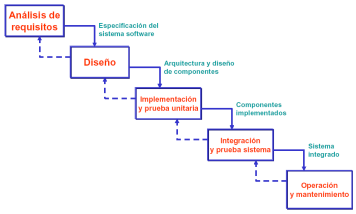
- Útil en las diferentes etapas del ciclo de vida del desarrollo de sistemas.

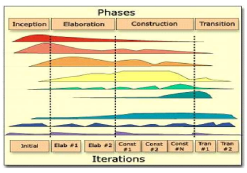


Lenguaje Unificado de Modelado

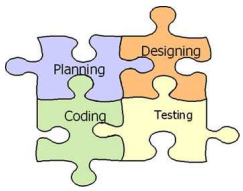
- Independiente del proceso de desarrollo de software.







MSF



Lenguaje Unificado de Modelado

- Independiente del lenguaje de implementación.

C++

python™

Microsoft .net

Ada

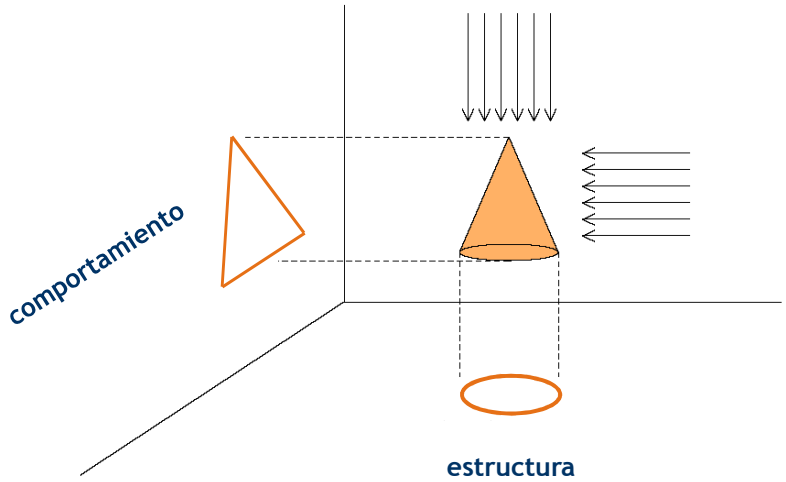
Java

¿Qué elementos definen un Método de Modelado?

- Un estilo → O.O.
- Un lenguaje → UML
- Un proceso → UP
- Herramientas → Together / Poseidon / Rational Modeler

Visión General de UML

- Definición de Vistas como proyección

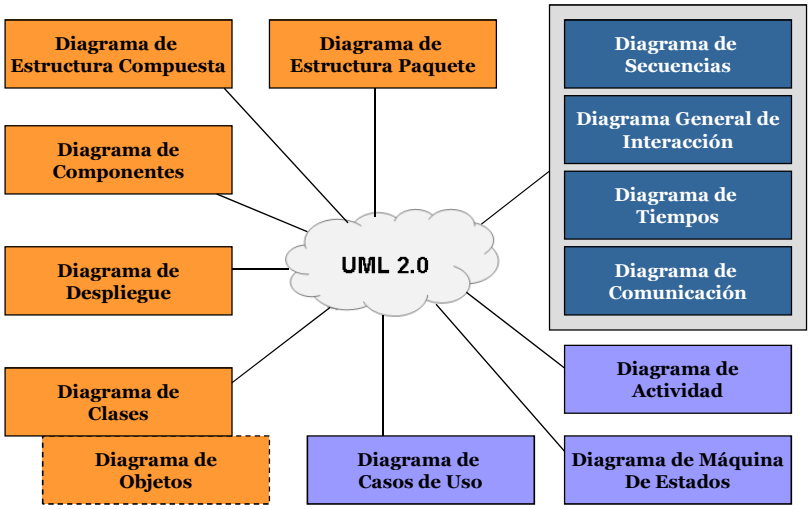


Modelado de la Arquitectura de un sistema

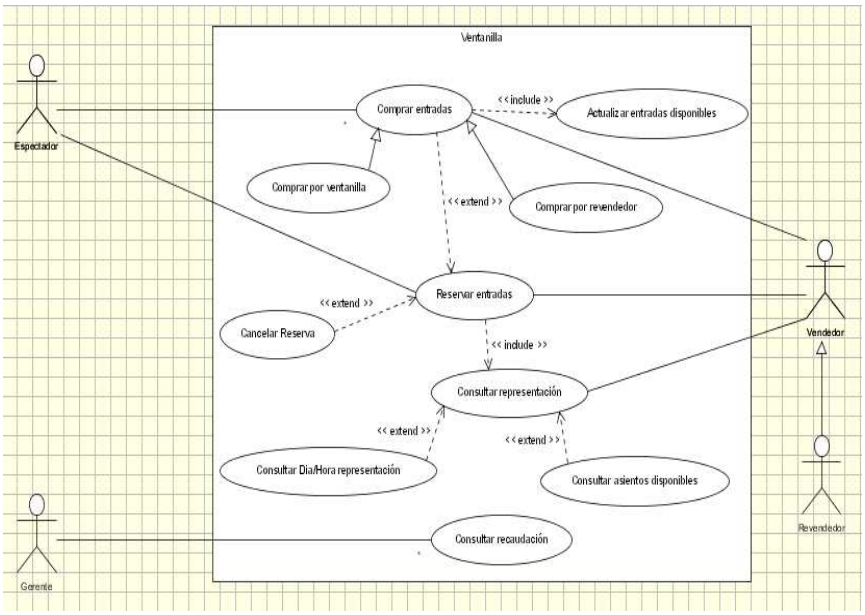
UML: bloques de construcción

- Elementos
  - Estructurales
    - Clase - Interfaz - Estructura Compuesta - Caso de Uso
    - Clase Activa - Componente - Nodo
  - De Instancia
    - Objetos
  - De Comportamiento
    - Interacción - Máquina de Estados - Actividad
  - De Agrupación
    - Paquetes - Frame
  - De Anotación
    - Notas
- Relaciones
  - Dependencia
  - Asociación
  - Generalización
- Diagramas
  - Diagrama de Clases
  - Diagrama de Objetos
  - Diagrama de Estructura Compuesta
  - Diagrama de Paquetes
  - Diagrama de Componente
  - Diagrama de Despliegue
  - Diagrama de Casos de Uso
  - Diagrama de Máquina de Estados
  - Diagrama de Actividades
  - Diagrama de Secuencia
  - Diagrama de Comunicación
  - Diagrama de Tiempos
  - Diagrama de Vista Global

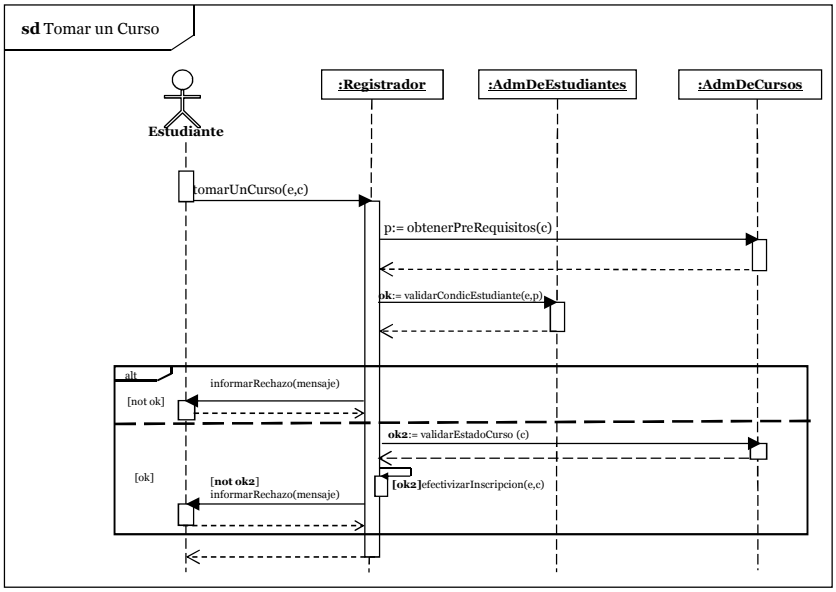
Diagramas de UML 2.0



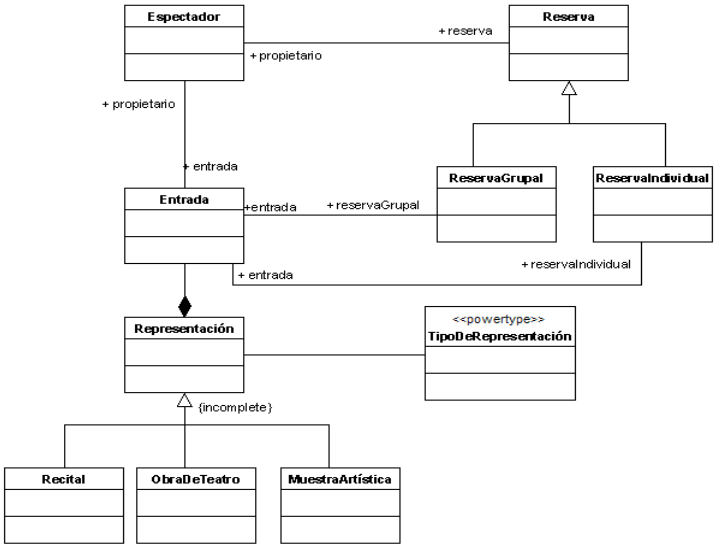
¿ Nombre del diagrama UML ?



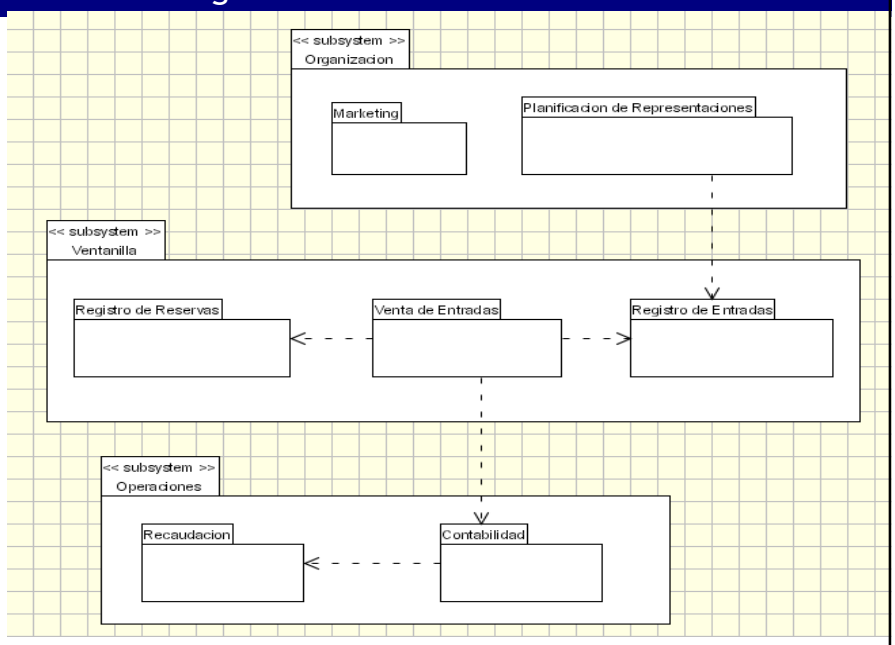
¿ Nombre del diagrama UML ?



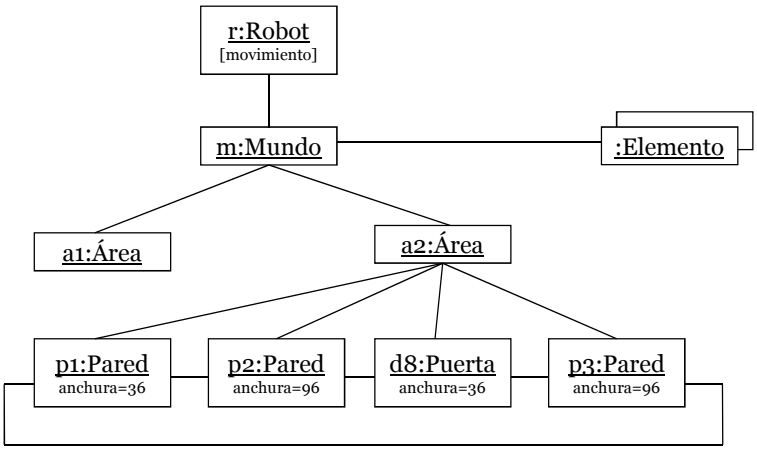
¿ Nombre del diagrama UML ?



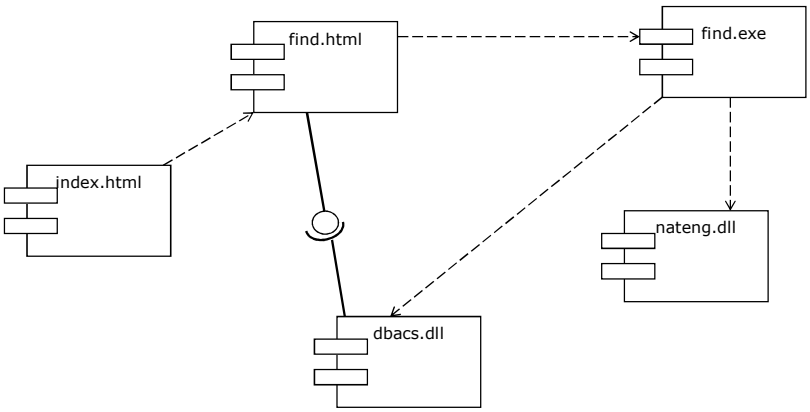
¿ Nombre del diagrama UML ?



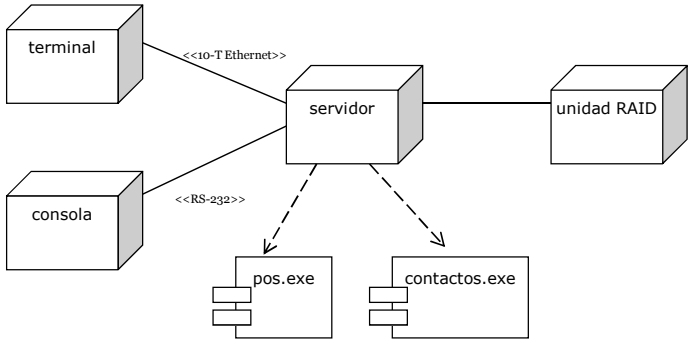
¿ Nombre del diagrama UML ?



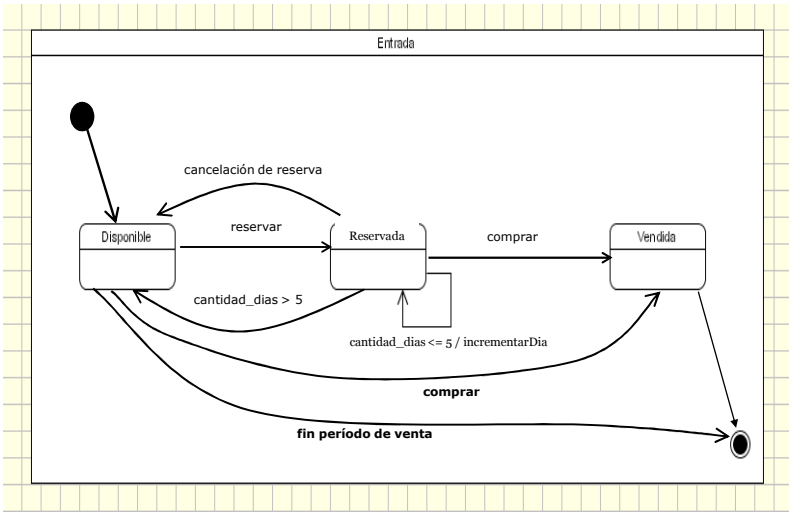
¿ Nombre del diagrama UML ?



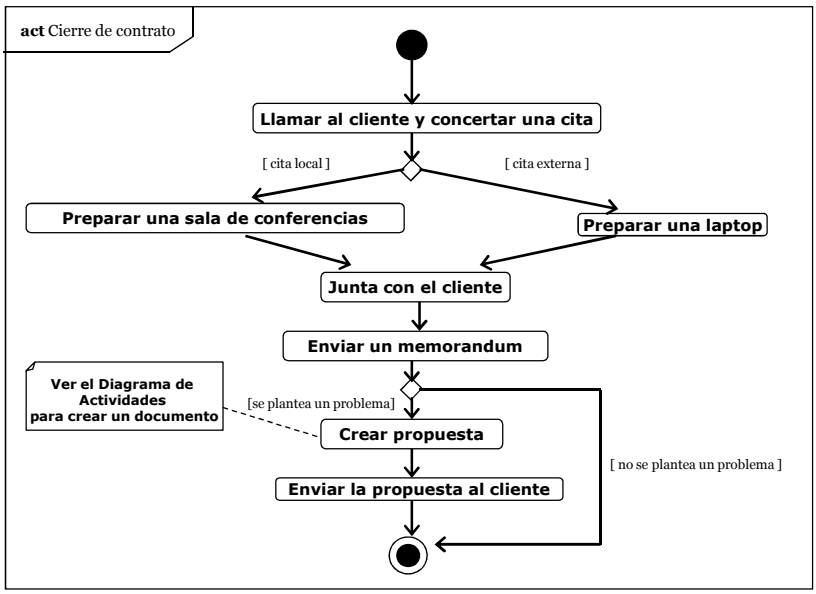
¿ Nombre del diagrama UML ?



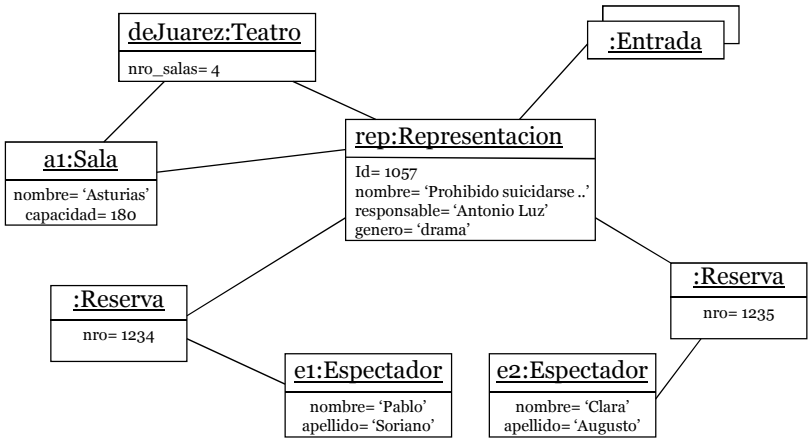
¿ Nombre del diagrama UML ?



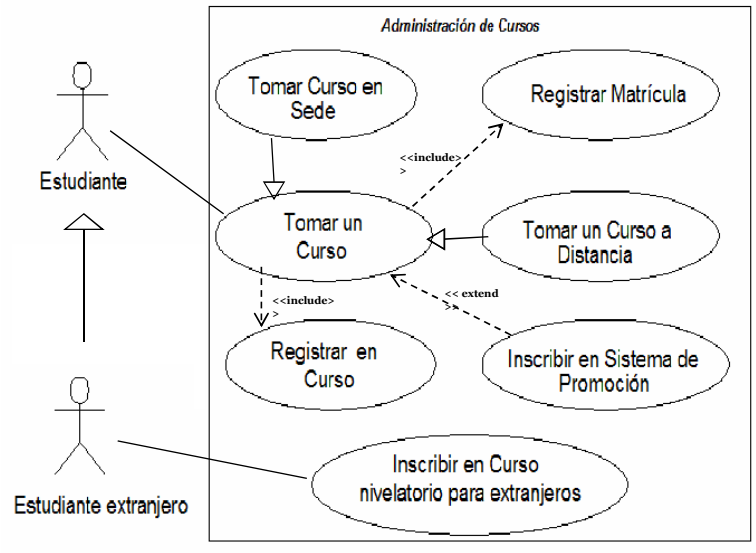
¿ Nombre del diagrama UML ?



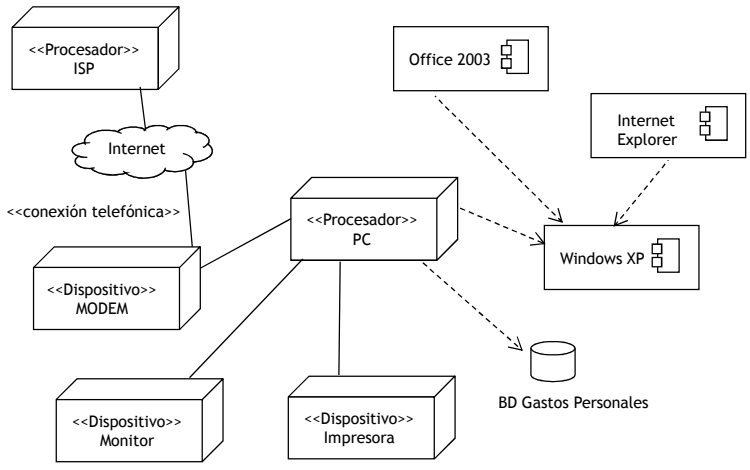
¿ Nombre del diagrama UML ?



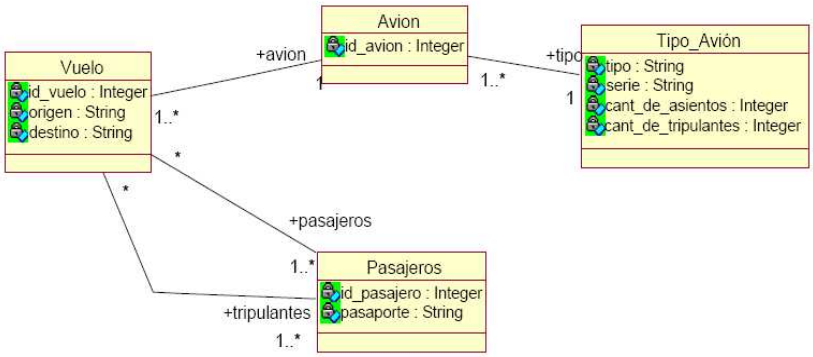
¿ Nombre del diagrama UML ?



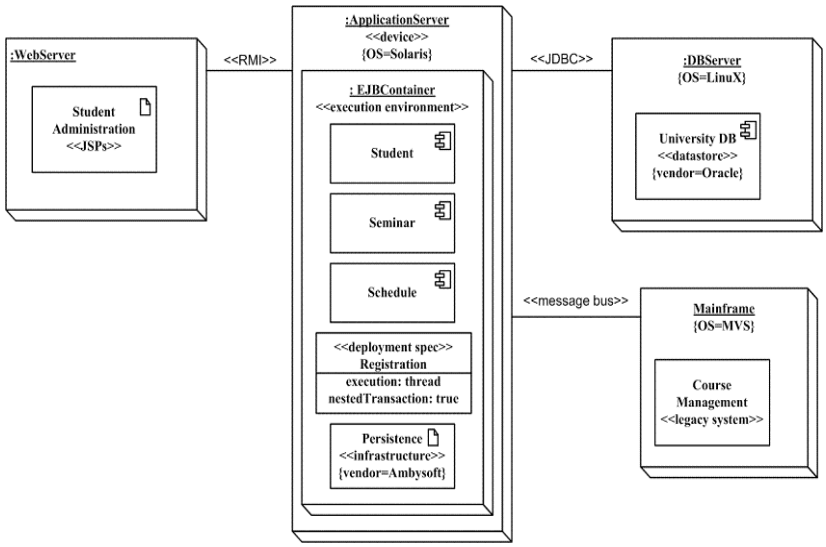
¿ Nombre del diagrama UML ?



¿ Nombre del diagrama UML ?



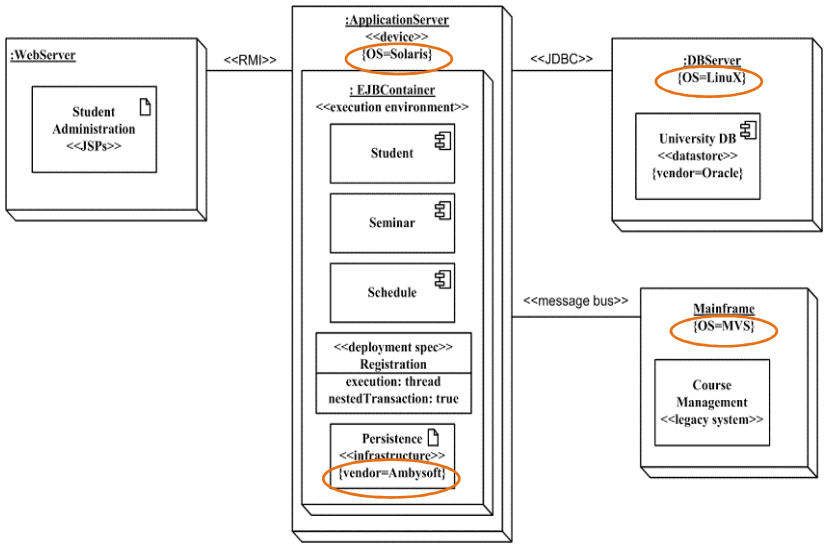
¿ Nombre del diagrama UML ?



a) ¿ Qué representa el estereotipo **<<JDBC>>** en el diagrama ?

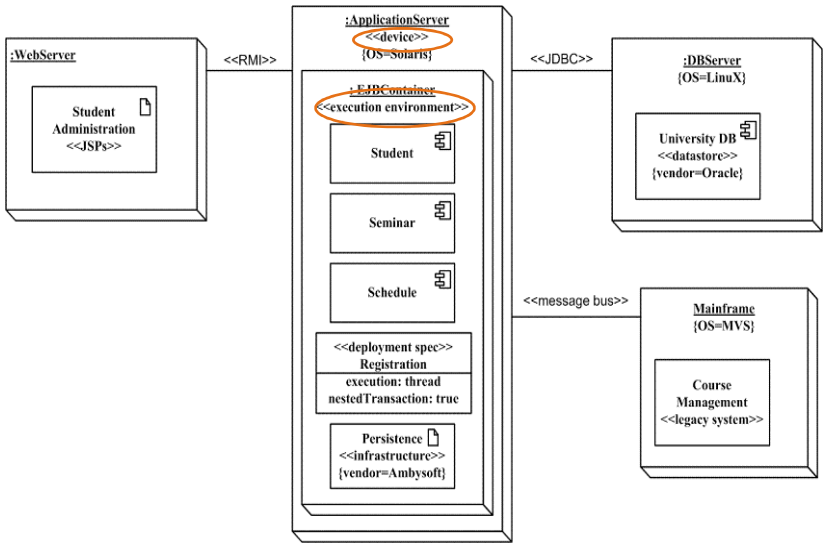
→ Una ruta de comunicación.

¿ Nombre del diagrama UML ?



b) Marque con un círculo los valores etiquetados que se encuentran enriqueciendo el diagrama de despliegue.

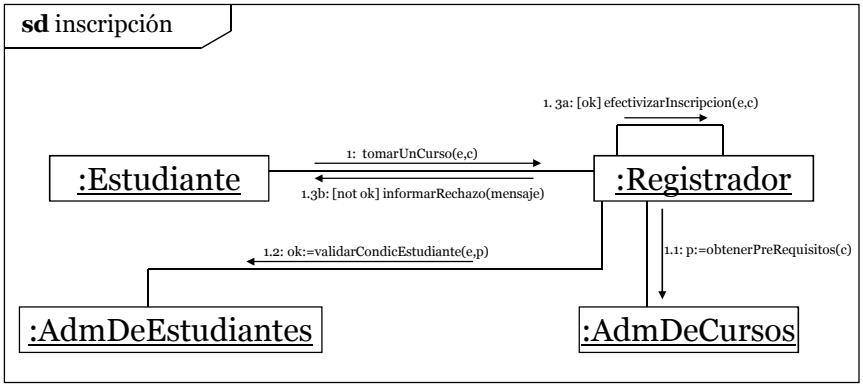
¿ Nombre del diagrama UML ?



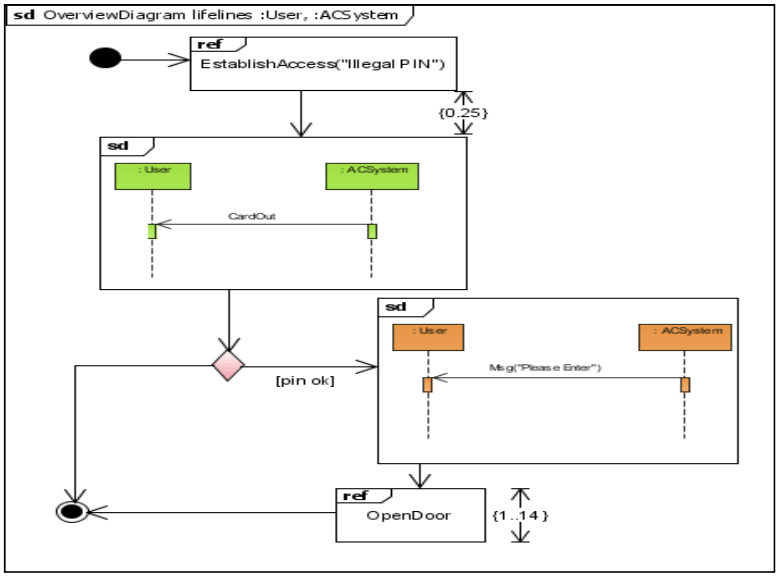
a) ¿ Qué indican los estereotipos adicionados a los nodos descriptos ?

→ El **tipo de dispositivo** físico empleado en el modelo de implementación.

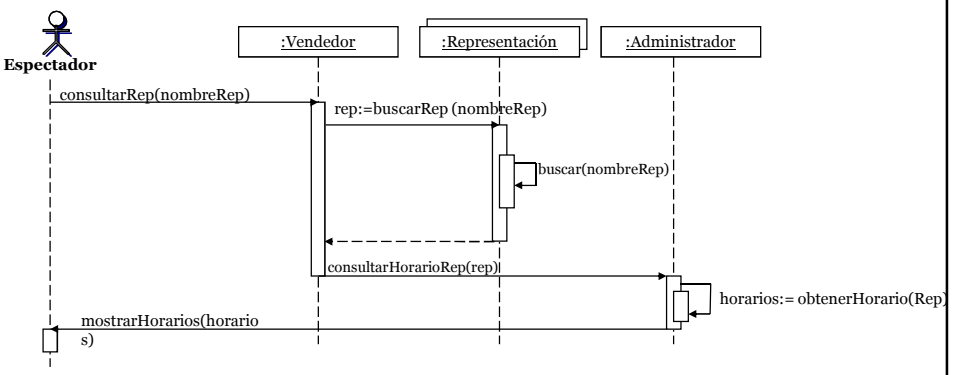
¿ Nombre del diagrama UML ?



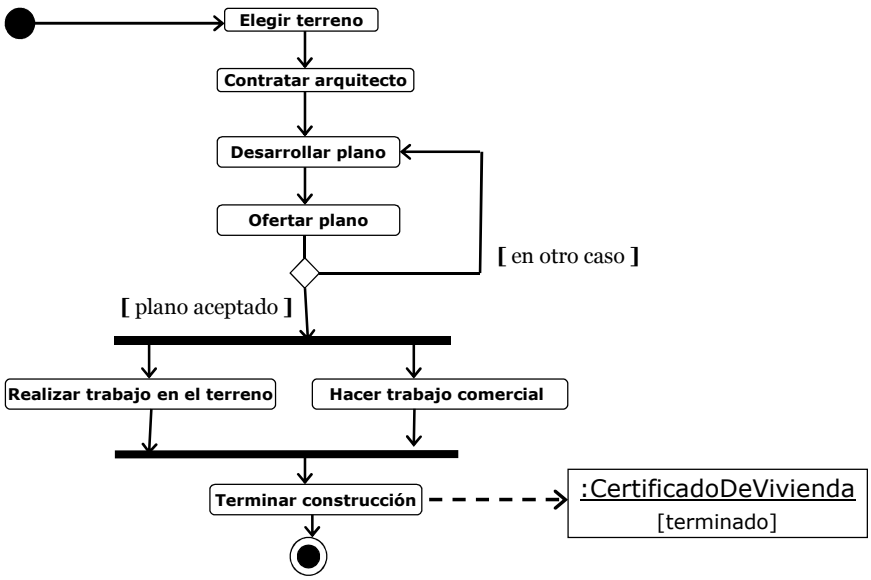
¿ Nombre del diagrama UML ?



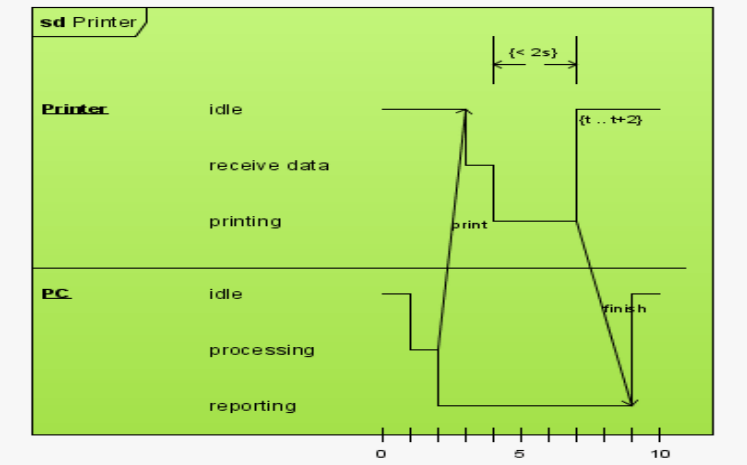
¿ Nombre del diagrama UML ?



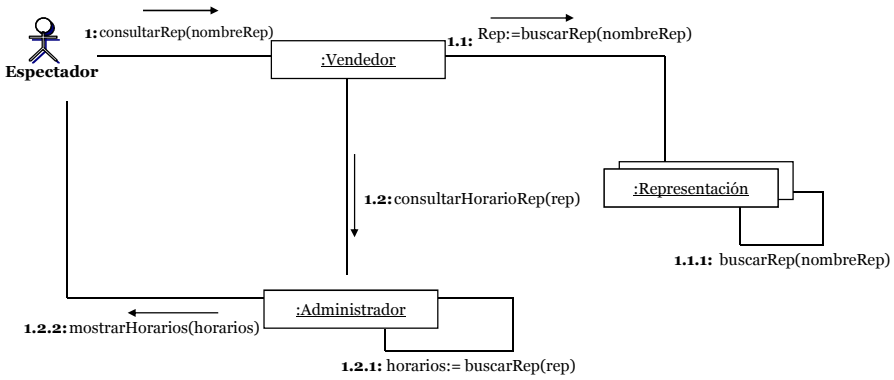
¿ Nombre del diagrama UML ?



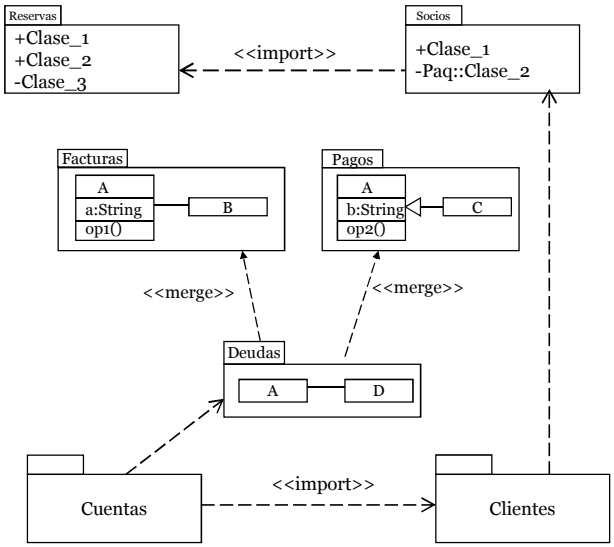
¿ Nombre del diagrama UML ?



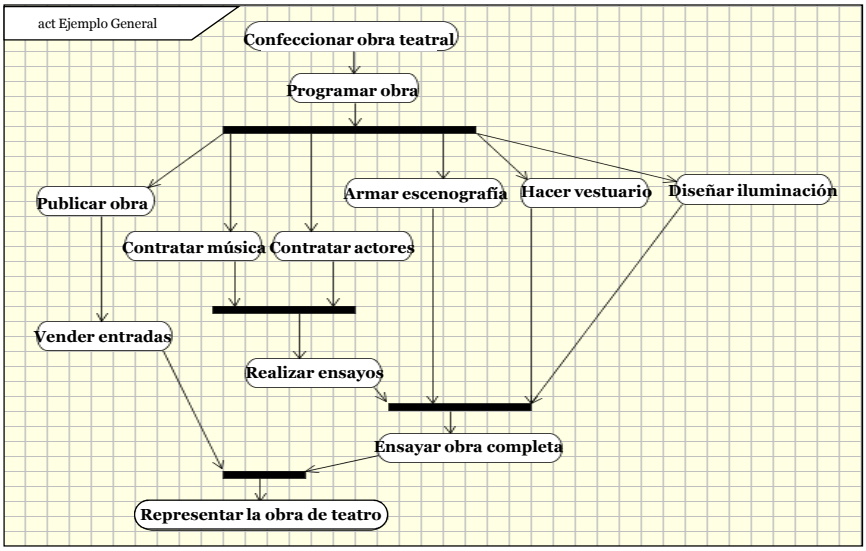
¿ Nombre del diagrama UML ?



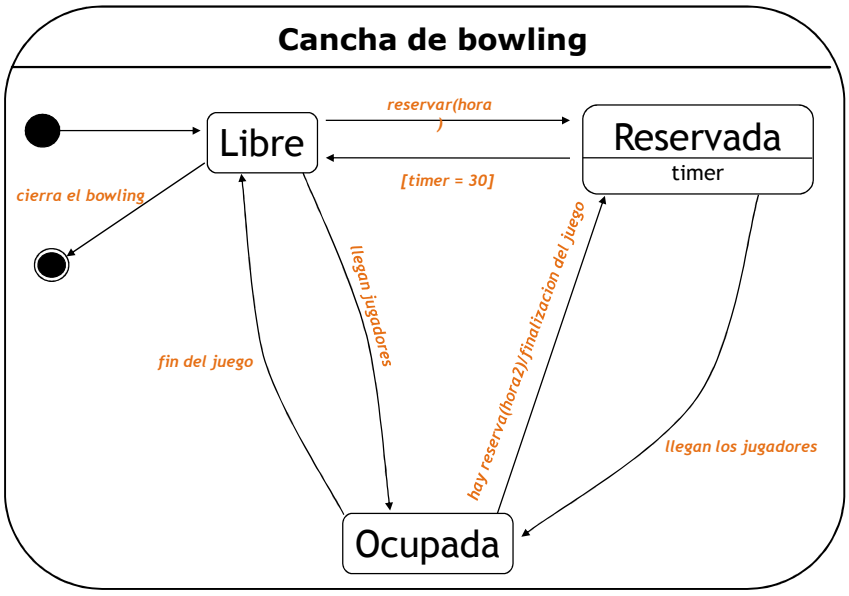
¿ Nombre del diagrama UML ?



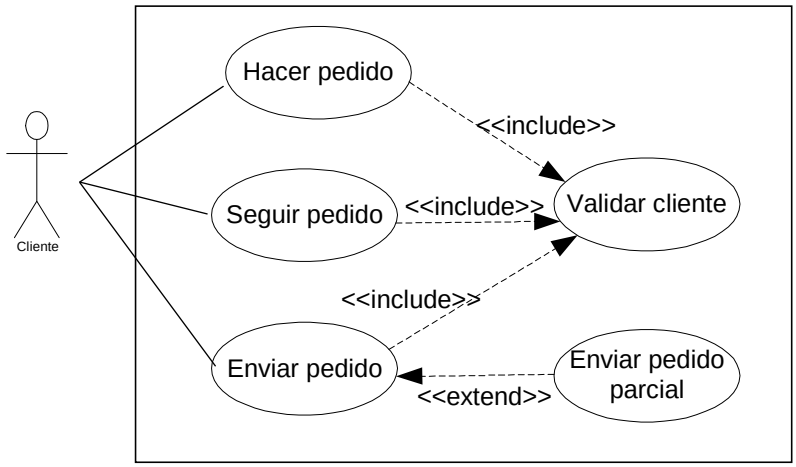
¿ Nombre del diagrama UML ?



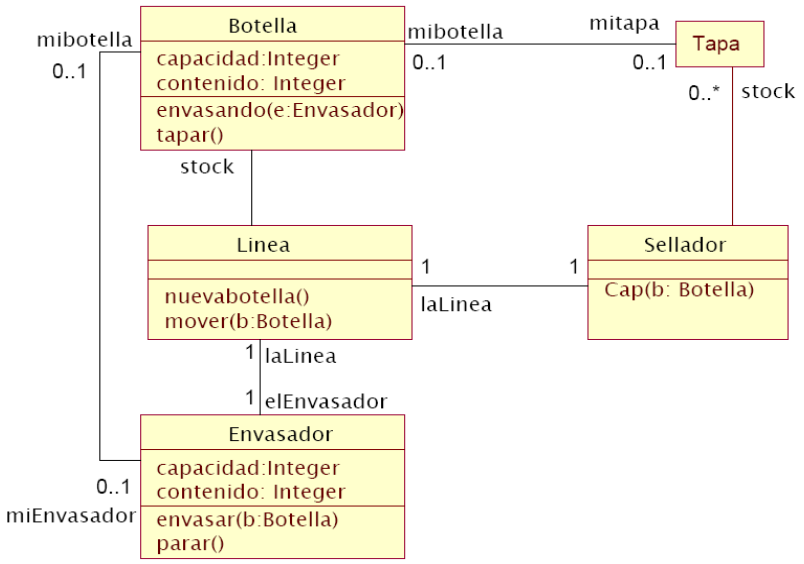
¿ Nombre del diagrama UML ?



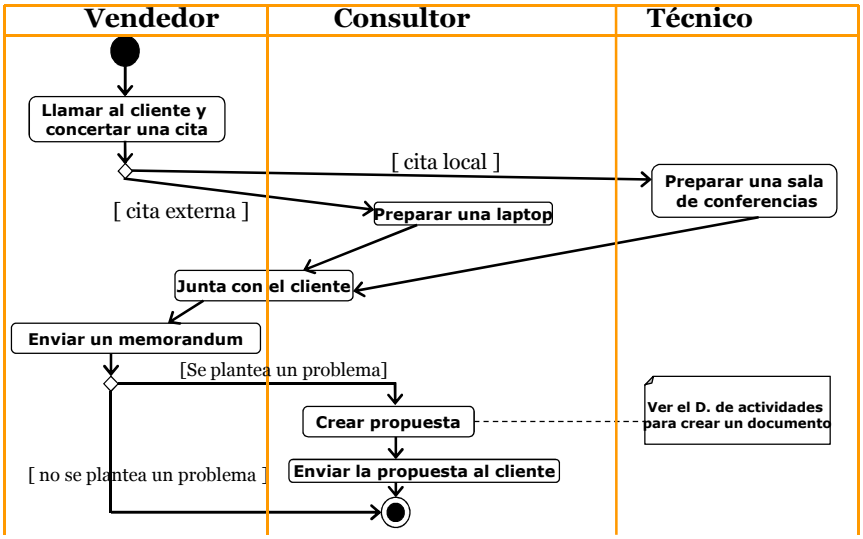
¿ Nombre del diagrama UML ?



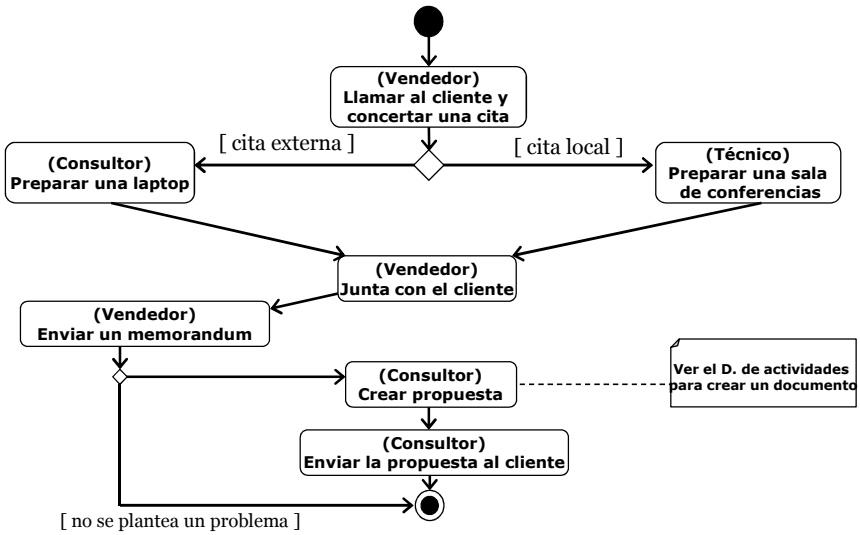
¿ Nombre del diagrama UML ?



¿ Nombre del diagrama UML ?



¿ Nombre del diagrama UML ?



Utilidades de UML

UML permite:

- Definir los límites del sistema y sus principales funciones, mediante **Casos de Uso y actores.**
- Ilustrar el funcionamiento de un caso de uso mediante **Diagramas de Interacción**
- Representar la estructura de un sistema mediante **Diagramas de Clases.**
- Modelar el comportamiento de los objetos mediante **Diagramas de Máquinas de Estados.**

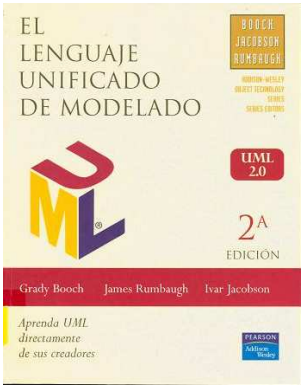
Utilidades de UML

UML permite:

- Describir la arquitectura de la implementación física con **Diagramas de Componentes y Despliegue.**
- Extender la funcionalidad de elementos estándar mediante **estereotipos.**
- Proveer **base formal** para los diagramas (metamodelo, OCL).

Bibliografía y textos recomendados

- Para repasar o revisar en detalle temas modelado con UML, se sugiere leer secciones (a demanda de sus inquietudes) de los libros:



44

- **INTRODUCCIÓN**
  - Repaso Notación UML
  - **Análisis y Diseño Orientado a Objetos**
  - Desarrollo Iterativo y el Proceso Unificado
  - Caso de estudio: el sistema de punto de venta
- **ETAPA INICIAL DEL PROCESO DE DESARROLLO**
- **ELABORACION EN LA ITERACION 1**
- **ELABORACION EN LA ITERACION 2**

45

Diseño de Sistemas

Análisis y Diseño Orientado a Objetos (AyDOO)

Análisis y Diseño OO

- **Análisis**

  El **análisis** pone énfasis en una **investigación del problema** y los **requisitos**, en lugar de ponerlo en la solución.

  → *explorar y descubrir los objetos en el dominio del problema*

- **Diseño**

  El **diseño** pone énfasis en una **solución conceptual** que **satisface los requisitos**, en lugar de ponerlo en la implementación.

→ *definición de los objetos software y en cómo colaboran para satisfacer los requerimientos*

47

48

FRLP

Diseño de Sistemas

Desarrollo Iterativo y el Proceso Unificado

¿Qué es el Proceso Unificado de Rational?

- Es, esencialmente, un proceso de desarrollo de software.
- También puede verse como un producto.
- O como un framework que puede especializarse para:
  - Sistemas software
  - Áreas de aplicación
  - Organizaciones
  - Tamaños de proyectos

Creado por Rational Software Corporation (ahora parte de IBM)

50

Claves del Proceso Unificado

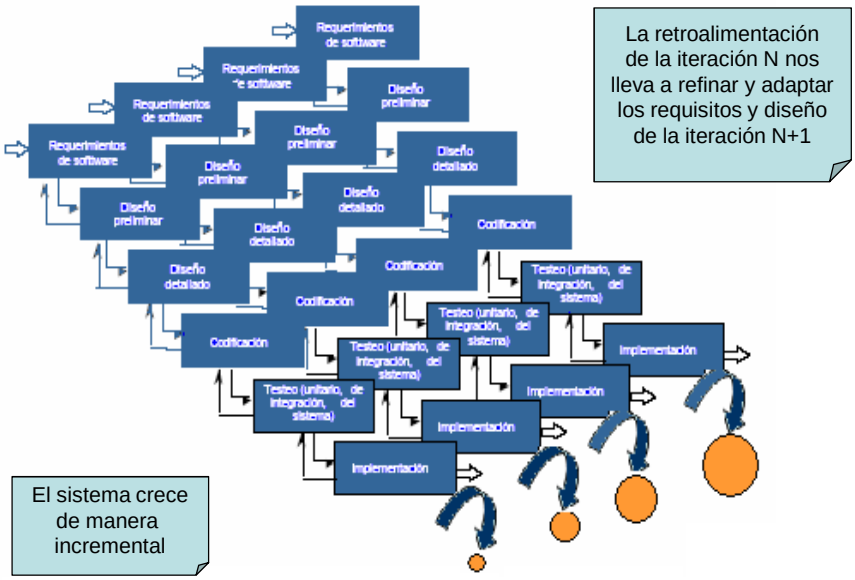
- Es iterativo e incremental
- Dirigido por los casos de uso.
- Centrado en la arquitectura

Además:

- Está basado en componentes
- Utiliza UML, una parte esencial del RUP

51

La idea más importante del RUP: desarrollo iterativo

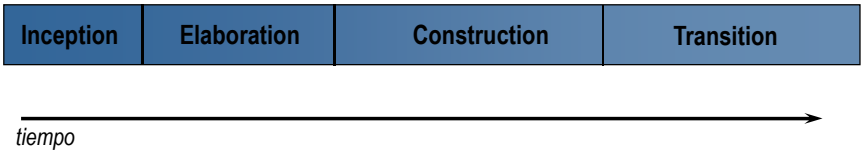


La retroalimentación de la iteración N nos lleva a refinar y adaptar los requisitos y diseño de la iteración N+1

El sistema crece de manera incremental

52

Fases del Proceso Unificado

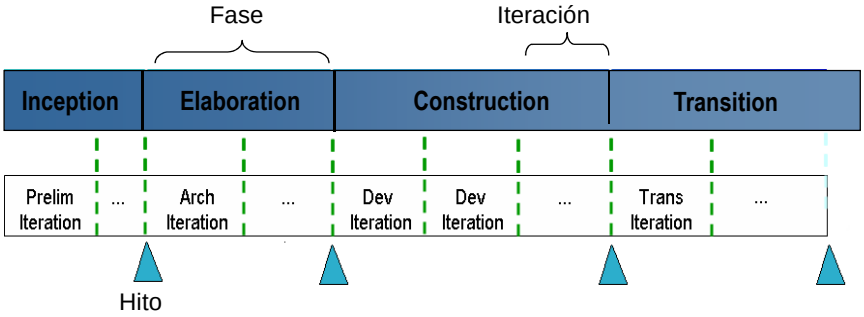


- Inception — Define el alcance del proyecto y desarrolla los casos de uso iniciales.
- Elaboration — Planea el proyecto, especifica características, define la arquitectura
- Construction — Construye el producto
- Transition — Transfiere el producto a los usuarios

53

Fases e iteraciones

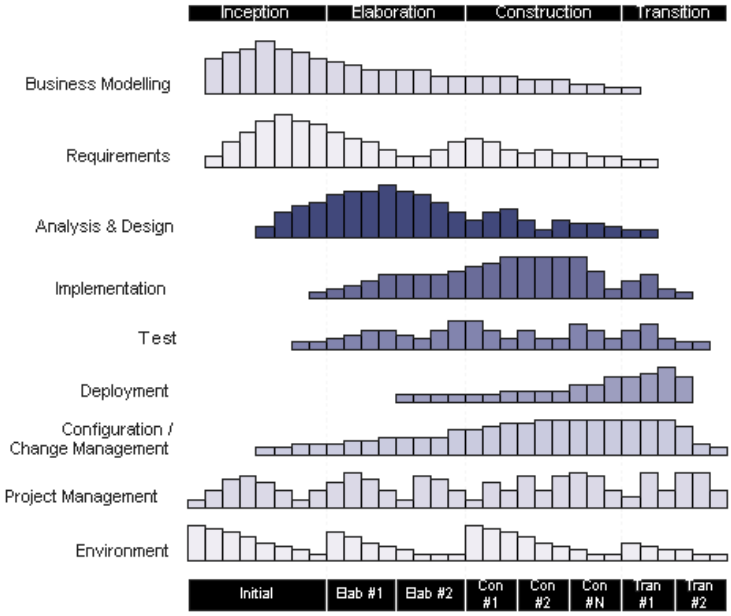
Una iteración es una secuencia de actividades con un plan establecido y criterios de evaluación, que resultan en un producto ejecutable.



Cada fase concluye en un hito

54

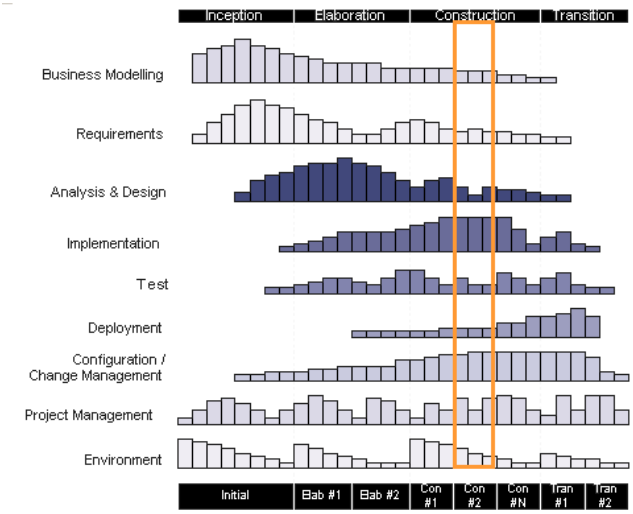
Disciplinas del Proceso Unificado



Aunque una iteración incluye trabajo en varias disciplinas, el esfuerzo relativo y el énfasis cambia en el tiempo

55

Una iteración



En una iteración se recorren todos los flujos de trabajo.

56

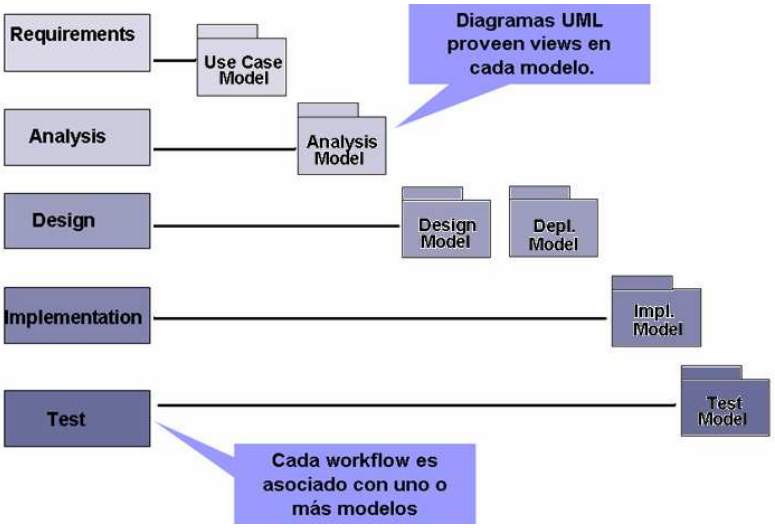
Proceso Iterativo e Incremental

Cada iteración comprende:

- Planificar la iteración (estudio de riesgos)
- Análisis de los Casos de Uso
- Diseño de opciones arquitectónicas
- Codificación y pruebas.
- Evaluación de la entrega ejecutable
- Preparación de la entrega

57

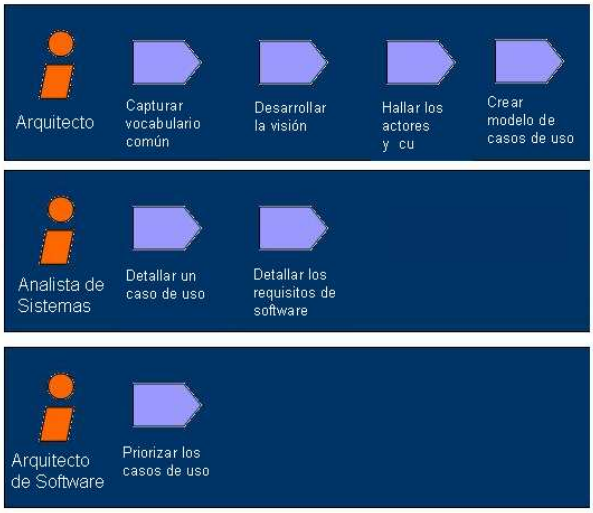
Disciplinas y Modelos



58

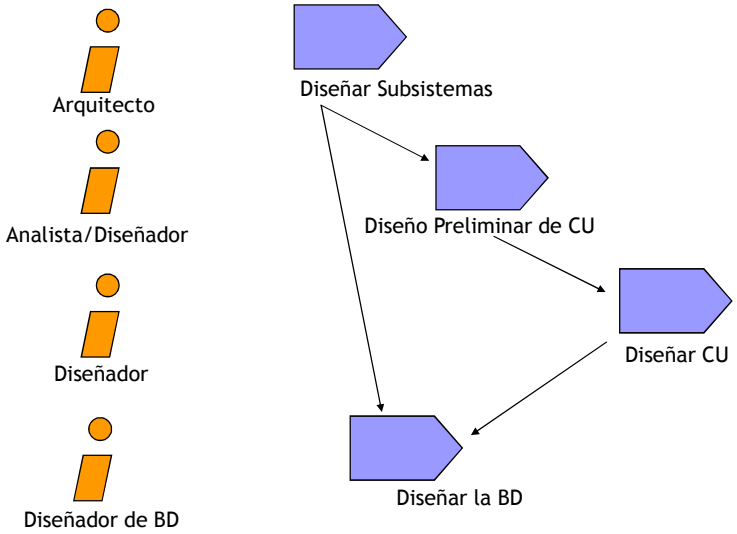
Workers en el RUP

Un *worker* define el **comportamiento** y **responsabilidades** de una persona o un conjunto de personas que trabajan como un equipo.



59

Trabajadores y flujo de trabajo de la etapa de diseño



60

No se entendió el UP cuando… (1)

- Se piensa que *Inicio* = Requisitos, *Elaboración* = Diseño, y *Construcción=Implementación*.
- Se intenta definir la mayoría de los requisitos antes de comenzar el diseño o la implementación.
- Se intenta definir la mayoría del diseño antes de comenzar la implementación.
- Se cree que el objetivo de la *Elaboración* es definir modelos de manera completa y cuidadosa, que se traducen a código durante la *Construcción*.
- Se piensa que adoptar el UP significa hacer muchas actividades posibles y crear muchos documentos.

61

No se entendió el UP cuando… (2)

- Se cree que una iteración debe durar al menos 4 meses;
- Se intenta planificar un proyecto en detalle de principio a fin;
- Se intentan predecir todas las iteraciones y lo que debería ocurrir en cada una de ellas;
- Se piensa que realizar las actividades de diseño y crear los diagramas UML constituyen el momento para definir diseños y modelos de manera completa.

62

Contenidos del Curso

- Módulo I: Introducción
- **Módulo II: Etapa Inicial del Proceso de Desarrollo**
- Módulo III: Elaboración de la Iteración 1
- Módulo IV: Elaboración de la Iteración 2

63

- **INTRODUCCIÓN**
- **ETAPA INICIAL DEL PROCESO DE DESARROLLO**
  - **Fase Inicio**
  - Comprensión de los requisitos
  - Modelos de Casos de Uso
  - Identificación de otros requisitos
  - Del Inicio a la Elaboración
- **ELABORACION DE LA ITERACION 1**
- **ELABORACION DE LA ITERACION 2**

64

Diseño de Sistemas

Fase Inicio

Fase Inicio-Objetivos

- Definir la visión y obtener una estimación.
- Corta duración.

Vislumbrar el alcance del producto,
visión y análisis del negocio

Si se decidió que se encarará el proyecto y que éste es viable, en esta fase se comenzará con los primeros talleres de requisitos y se planeará la primera iteración, cambiando rápidamente a la *Elaboración*.

66

Fase Inicio-Artefactos

Artefacto	Comentario
Visión y Análisis del Negocio	Describe objetivos y restricciones; proporciona un informe para la toma de decisiones
Modelo de Casos de Uso	Describe requisitos funcionales y no funcionales
Especificación complementaria	Describe otros requisitos
Glosario	Describe términos del dominio
Lista de Riesgos y Plan de Gestión de Riesgos	Describe riesgos técnicos, del negocio, recursos, ideas para mitigarlos.
Plan de Iteración	Describe qué hacer en la primera iteración
Plan de Desarrollo de Software	Estimación (poco precisa) de duración y esfuerzo en Elaboración
Marco de Desarrollo	Describe pasos del UP y artefactos adaptados para el proyecto

67

No se entendió la fase *Inicio* cuando …

- Se piensa que la secuencia de trabajo es:
  1°)definición de requisitos;
  2°) diseño de la arquitectura;
  3°) implementación.
- No hay artefacto de Visión o Análisis del Negocio.
- No se identificaron la mayoría de los nombres de los casos de uso y los actores.
- Se escribieron todos los casos de uso en detalle.
- No se escribió ningún caso de uso en detalle.
- Se intentan definir todos o casi todos los requisitos

68

- **INTRODUCCIÓN**
- **ETAPA INICIAL DEL PROCESO DE DESARROLLO**
  - Fase Inicio
  - **Comprensión de los requisitos**
  - Modelos de Casos de Uso
  - Identificación de otros requisitos
  - Del Inicio a la Elaboración
- **ELABORACION DE LA ITERACION 1**
- **ELABORACION DE LA ITERACION 2**

69

Diseño de Sistemas

Comprensión de los Requisitos

Comprensión de los Requisitos

- ¿Qué entendemos por requisitos?

Los **requisitos** son **capacidades** y **condiciones** con las cuales debe ser conforme el sistema.

El RUP fomenta un conjunto de buenas prácticas, entre ellas, la *gestión de requisitos*, que hace referencia a definir **"un enfoque sistemático para encontrar, documentar, organizar y seguir la pista de los requisitos cambiantes del sistema"**

La visión de un proceso iterativo es utilizar un filosofía que acepte el cambio y la retroalimentación como motores centrales en el descubrimiento de los requisitos.

71

Tipos de Requisitos

Los requisitos se clasifican según el modelo FURPS+.

• Funcional (*Functional*)	Características, capacidades, seguridad.
• Usabilidad (*Usability*)	Factores humanos, ayuda, documentación.
• Fiabilidad (*Reliability*)	Frecuencia y recuperación de fallos, grado de previsión.
• Rendimiento (*Performance*)	Tiempos de respuesta, precisión, disponibilidad, uso de los recursos.
• Soporte (*Supportability*)	Adaptabilidad, facilidad de mantenimiento, configurabilidad.

¿Y qué significa el '**+**' de FURPS+ ?
Indica requisitos adicionales tales como:

- Implementación
- Interfaz
- Legales

72

- **INTRODUCCIÓN**
- **ETAPA INICIAL DEL PROCESO DE DESARROLLO**
  - Fase Inicio
  - Comprensión de los requisitos
  - **Modelos de Casos de Uso**
  - Identificación de otros requisitos
  - Del Inicio a la Elaboración
- **ELABORACION DE LA ITERACION 1**
- **ELABORACION DE LA ITERACION 2**

73

FRLP

Diseño de Sistemas

Modelo de Casos de Uso

Modelo de CU-Algunas características

- Definición de RUP:

Un conjunto de instancias de los casos de uso modelados, donde cada instancia es una secuencia de acciones que un sistema ejecuta, produciendo un resultado observable de valor para un actor particular.

- Se define en la disciplina Requisitos del RUP
- La idea es utilizar casos de uso de "caja negra"
- Los casos de uso son documentos de texto

75

Casos de Uso- Tipos de formalidad

Existen tres tipos o grados de formalidad:

- Breve

  Es un resúmen conciso que no ocupa más de un párrafo. Se describe el escenario principal con éxito (curso normal).

- Informal

  La descripción puede abarcar varios párrafos, pero no demasiados, especificando varios escenarios. Se caracteriza por un estilo informal de escritura.

- Completo

  Es el formato más elaborado, ya que se describen con detalle todos los pasos y variaciones (curso normal y alternativos). Cuenta con otras secciones como pre y post condiciones, curso de error, etc.

76

Documentación de Casos de Uso – FORMATO COMPLETO

Composición general *no* estandarizada:

- Resumen de la Identificación.
- Descripción de Recursos (Hardware y Software).
- Definición de Conversación.
- Interfaz de usuario
- Otros diagramas

Título
Descripción contextual
Objetivo
Iniciación del CU
Fecha de Creación
Analista/Diseñador a cargo
Desarrollador a cargo
Prioridad
Número de versión
Última fecha de modificación
Actores involucrados
Relaciones con otros CU

Precondiciones
Postcondiciones
Conversación
- Curso Normal
- Curso Alternativo
- Curso de Error

Flujos de Entrada
Flujos de Salida

Diagramas de Estados
Diagrama de Actividades
Otros.

77

Ejemplo- Formato Breve

Caso de uso: **Procesar venta**

Un cliente llega a una caja con artículos para comprar. El cajero utiliza el sistema PDV para registrar cada artículo comprado. El sistema presenta una suma parcial y detalles de cada línea de venta. El cliente introduce los datos del pago, que el sistema valida y registra. El sistema actualiza el inventario. El cliente recibe un recibo del sistema y luego se va con los artículos.

78

Ejemplo- Formato Informal

Caso de uso: **Gestionar devoluciones**

***Escenario principal de éxito*: un cliente llega a una caja con artículos para devolver. El cajero utiliza el sistema PDV para registrar cada uno de los artículos devueltos …**

Escenarios alternativos:

Si se pagó con tarjeta de crédito y se rechaza la transacción de reembolso a su cuenta, informar al cliente y pagarle en efectivo.

Si el identificador del artículo no se encuentra en el sistema, notificar al cajero y sugerir la entrada manual del código de identificación.

……

79

Ejemplo- Formato Completo

Caso de uso: **Procesar venta**

Actor principal: Cajero

Personal involucrado e intereses:

Cajero: Quiere entradas rápidas y sin errores de pago (se deducen de su sueldo)

Compañía: quiere registrar las transacciones sin errores y satisfacer los pedidos de los clientes.

……

Precondiciones:

- El cajero se identifica y se valida.

Postcondiciones:

- Se creó la venta.
- El impuesto se calculó de manera correcta.
- Se generó el recibo.

80

Conversación del C.U. “Comprar producto”

Acciones del actor	Respuestas del sistema	
	Curso Normal	Curso Alternativo
1-El Cliente llega a una caja con mercadería a a comprar		
	2- Comenzar una nueva venta	
	3- Introducir el identificador del artículo	3,1-Identificador no válido -> Marcar el error y rechazar la entrada. 3,6-Se quiere eliminar un artículo. 3,6.1- Introducir identificador del artículo 3,6.2- Mostrar suma parcial actualizada
	4- Registrar línea de venta. Presentar descripción del artículo y precio	4.a-Se genera el precio de otro artículo, no el ingresado .....
	5- Presentar el total con los impuestos	
	Se repiten los pasos 3-4 hasta que se indique	
	6-Indicar el total al cliente y solicitar pago	
7- Elige la forma de pago: a- efectivo b- tarjeta de crédito		
	8-Con el pago, registrar la venta. Enviar la información de la venta y el pago al sist de Contabilidad externo y al sist Inventario	
	9-Emitir el recibo	

81

Descubriendo actores y casos de uso

Los casos de uso se definen para satisfacer los objetivos de usuario de actores principales.

Definimos como procedimiento básico:

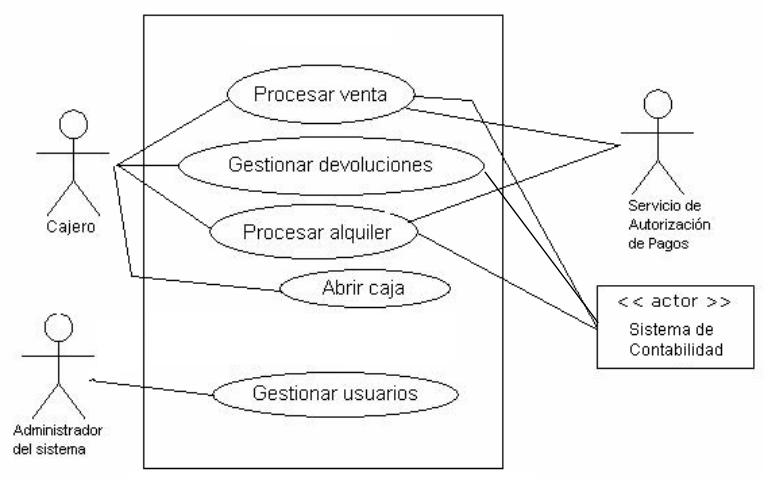
1. Elegir el límite del sistema
2. Identificar los actores principales (actores de apoyo)
3. Identificar los objetivos de usuario de cada actor
4. Definir los casos de uso

El actor principal y los objetivos de usuario dependen del límite del sistema

82

Diagramas de casos de uso

Proporciona información visual concisa del sistema, los actores externos y cómo lo utilizan



83

Casos de uso en las fases del UP

Inicio

Se escriben en formato breve, los CU complejos o arriesgados.
Luego, entre el 10% y el 20% de éstos, se escriben en formato completo.
Objetivo: tener una idea del alcance, riesgo, viabilidad técnica, para decidir si seguir o no.

Elaboración

Se priorizan los casos de uso por su complejidad o riesgo.
Incrementalmente, se van completando y se identifican la mayoría de los requisitos.
Entre el 80% y 90% se escriben con detalle.

Construcción

La mayor parte de los requisitos se "estabilizaron".
Aún se escribirán casos de uso menores

84

- **INTRODUCCIÓN**
- **ETAPA INICIAL DEL PROCESO DE DESARROLLO**
  - Fase Inicio
  - Comprensión de los requisitos
  - Modelos de Casos de Uso
  - **Identificación de otros requisitos**
  - Del Inicio a la Elaboración
- **ELABORACION DE LA ITERACION 1**
- **ELABORACION DE LA ITERACION 2**

Diseño de Sistemas

Identificación de otros requisitos

Más artefactos

- **Especificación Complementaria**

  Se documentan otros requisitos, los de calidad por ejemplo, información y restricciones de todo el sistema.

  Adiciona: requisitos FURPS+, informes, restricciones, etc.

- **Glosario**

  Almacena los términos y definiciones del dominio.

- **Visión**

  Documenta por qué se propuso el proyecto, problemas, posibles soluciones; características del sistema

87

- **INTRODUCCIÓN**
- **ETAPA INICIAL DEL PROCESO DE DESARROLLO**
  - Fase Inicio
  - Comprensión de los requisitos
  - Modelos de Casos de Uso
  - Identificación de otros requisitos
  - **Del Inicio a la Elaboración**
- **ELABORACION DE LA ITERACION 1**
- **ELABORACION DE LA ITERACION 2**

88

Diseño de Sistemas

Del Inicio a la Elaboración

¿ Qué sucedió en el Inicio ?

- La fase puede durar no más de una semana (en general)
- Los artefactos creados deberían ser breves e incompletos.
- Algunos artefactos posibles:
  - La mayoría de los actores, algunos objetivos y casos de uso, con los nombres
  - La mayoría de los casos de uso escrito en formato breve, y del 10 al 20 % en formato completo.
  - La mayoría de los requisitos de calidad y/o de riesgo identificados
  - Escritura de la primera Visión
  - Escritura de la primera Especificación Complementaria
  - Lista de riesgos
  - Arquitectura candidata
  - Plan para la primera iteración (de la Elaboración)

90

¿ Cómo continuamos con la Elaboración ?

- Se realizan iteraciones breves y de duración fija.
- La mayoría de los casos de uso se escriben en detalle
- Se comienza a programar pronto
- Diseñar, implementar y probar las partes básicas y arriesgadas de la arquitectura.
- Probar desde el principio.
- Adaptar en base a la retroalimentación procedente de las pruebas, usuarios y desarrolladores.
- Planificar la siguiente iteración.

Construir el núcleo central de la arquitectura, resolver los elementos de alto riesgo, definir la mayoría de los requisitos, y estimar la planificación y los recursos globales.

91

No se entiende la fase *Elaboración* cuando …

- La duración es superior a unos pocos meses, en la mayoría de los proyectos.
- Sólo comprende una iteración.
- El resultado no es una arquitectura ejecutable; no hay programación (código).
- Se intenta tener un diseño completo y en detalle antes de programar.
- No se realizan pruebas realistas en las primeras etapas.
- La mayor parte de los requisitos se definieron antes de la Elaboración.

92

Contenidos del Curso

- Módulo I: Introducción
- Módulo II: Etapa Inicial del Proceso de Desarrollo
- **Módulo III: Elaboración en la Iteración 1**
- Módulo IV: Elaboración en la Iteración 2

93

- **INTRODUCCIÓN**
- **ETAPA INICIAL DEL PROCESO DE DESARROLLO**
- **ELABORACION EN LA ITERACION 1**
  - **Modelo de casos de uso: Representación de diagramas de secuencia**
  - Modelo del dominio
  - Modelo de casos de uso: contratos de las operaciones
  - De los requisitos al diseño en esta iteración
  - GRASP: diseño de objetos con responsabilidades
  - Modelo de Diseño
  - Modelo de Implementación
- **ELABORACION EN LA ITERACION 2**

94

Diseño de Sistemas

Representación de los diagramas de secuencia del sistema

Diagramas de secuencia del sistema (DSS)

Los casos de uso describen cómo interactúan los actores con el sistema. Durante esta interacción, un actor genera eventos sobre un sistema, solicitando alguna operación o respuesta.

Con diagramas de secuencia podemos representar las interacciones de los actores (externos al sistema) y las operaciones que inician.

Debería hacerse un diagrama de secuencia para el escenario principal de éxito del caso de uso, y los escenarios alternativos complejos o frecuentes.

La mayoría de los DSS se crean en la Elaboración, cuando es útil identificar los detalles de los eventos (operaciones que se deben diseñar que gestione el sistema)

96

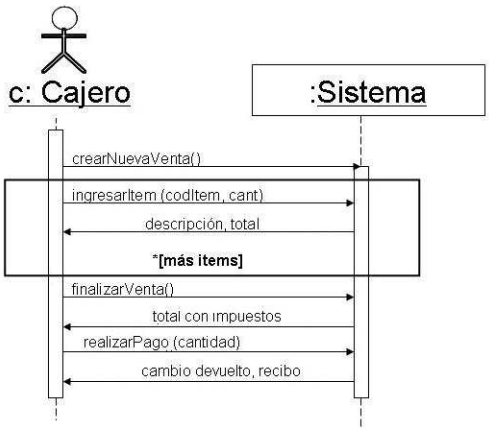
Ejemplo de un DSS

Los DSS se derivan de los casos de uso

Escenario Simple de Procesar Venta para el pago en efectivo

1.Elcliente llega al terminal PDV.
2.El cajero inicia una nueva venta.
3.El cajero ingresa el identificador del artículo.
4.El sistema registra la línea de venta y presenta la descripción del artículo, precio y suma parcial.
El cajero repite los pasos 3 y 4 hasta que se indique.
5.El sistema muestra el total con los impuestos calculados.
6.El cajero le dice al cliente el total con los impuestos calculados.
7.El cliente paga y el sistema gestiona el pago.

.....



El cajero genera los eventos del sistema crearNuevaVenta, introducirArtículo, finalizarVenta y realizarPago.

97

- **INTRODUCCIÓN**
- **ETAPA INICIAL DEL PROCESO DE DESARROLLO**
- **ELABORACION EN LA ITERACION 1**
  - Modelo de casos de uso: Representación de diagramas de secuencia
  - **Modelo del dominio**
  - Modelo de casos de uso: contratos de las operaciones
  - De los requisitos al diseño en esta iteración
  - GRASP: diseño de objetos con responsabilidades
  - Modelo de Diseño
  - Modelo de Implementación
- **ELABORACION EN LA ITERACION 2**

98

Diseño de Sistemas

Modelo del Dominio

Identificación de clases conceptuales

La tarea central es identificar las clases conceptuales relacionadas con el escenario que se está diseñando.

> Es mejor especificar en exceso un modelo del dominio con muchas clases conceptuales de grano fino que especificar por defecto

Consejos:
- Usar nombres del dominio del problema, no de la solución.
- Omitir detalles (irrelevantes o pertenecientes a futuras iteraciones)
- No inventar nuevos conceptos (del dominio de la solución).

Estrategias:
- Identificación de frases nominales.
- Utilización de una lista de categorías de clases conceptuales.

100

Identificación de clases conceptuales- Frases nominales

Encontrar conceptos (y sus atributos) mediante la identificación de los **sustantivos** en la descripción textual del dominio del problema.

Ejemplo:

1- El **Cliente** llega a un **terminal PDV** con **artículos** para comprar.

2- El **Cajero** comienza una nueva **venta**.

3- El **Cajero** ingresa el **identificador** del **artículo**.

4- El sistema registra la **línea de venta** y presenta la **descripción**, **precio** y **suma** parcial.

El Cajero repite los pasos 3-4 hasta que se indique.

5- El Cajero le dice al Cliente el total y solicita el **pago.**

6- .....

101

Identificación de clases conceptuales- Lista de categorías

Categoría de Clase Conceptual	Ejemplos
Objeto físico o tangible	Terminal PDV
Especificación de una cosa	Especificación del producto
Lugar	Tienda
Transacción	Venta, pago
Roles de la gente	Cajero, cliente
Contenedor de cosas	Almacén, catálogo, registro ventas
Cosas en un contenedor	Artículo
Otros sistemas	Sist. de Contabilidad, Sist. de Autorización de Pagos
Hechos	Venta, pago
Reglas y políticas	Política de reintegro
Registros financieros/laborales	Factura/Recibo
Manuales, documentos	Reglas de devolución, moras por alquiler

102

Construyendo el Modelo del Dominio

Pasos a seguir:

1- Listar los conceptos candidatos

2- Graficarlos en un Modelo del Dominio.

3- Agregar atributos a los conceptos.

4- Agregar asociaciones entre conceptos.

103

(1) Conceptos candidatos

- Punto de Venta
- Artículo
- Venta
- LíneaDeVenta
- CatálogoDeProducto
- EspecificaciónDeProducto
- Pago
- Cajero
- Cliente
- Factura (???)
- …

104

(2) Modelo del Dominio- Visualización de conceptos

Un **Modelo del Dominio** es una representación visual de las clases conceptuales del mundo real en un dominio de interés.

105

(3) Agregar atributos

Se identifican los atributos que son necesarios para satisfacer los requerimientos de información de los casos de uso en desarrollo.

> Los atributos en un modelo deberían ser, preferiblemente, atributos simples o tipos de datos.

Los tipos de datos de los atributos más comunes incluyen:

- Boolean
- Fecha
- Número
- String (texto)
- Hora

> Recuerde relacionar las clases conceptuales con asociaciones, no con atributos

106

Agregar atributos

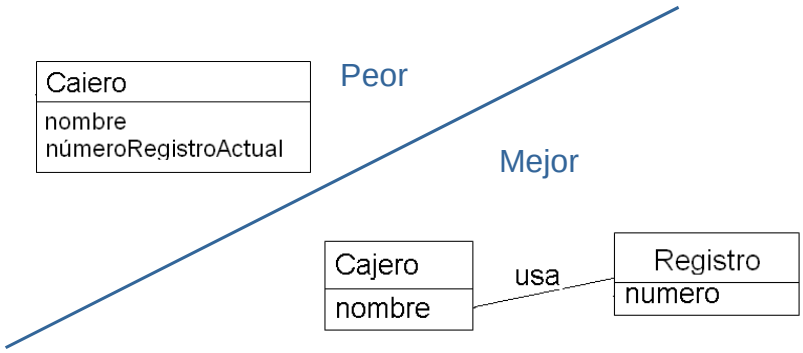
Represente lo que inicialmente podría considerarse un tipo de dato primitivo como una clase si:

- Está compuesto de secciones separadas
- Tiene operaciones asociadas
- Tiene otros atributos
- Es una cantidad con una unidad
- Es una abstracción de uno o más tipos con esas cualidades

107

Atributo como claves ajenas

¿Debería usar un atributo para relacionar clases conceptuales?

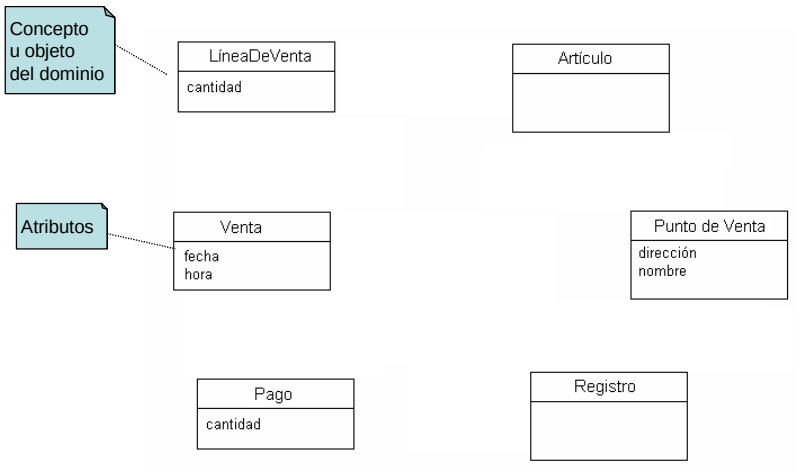


La mejor manera de expresar que un concepto utiliza a otro es, nuevamente, con una asociación; no con un atributo de clave ajena

108

Modelo del Dominio- Visualización de atributos

Un **Modelo del Dominio** es una representación visual de las clases conceptuales del mundo real en un dominio de interés.



109

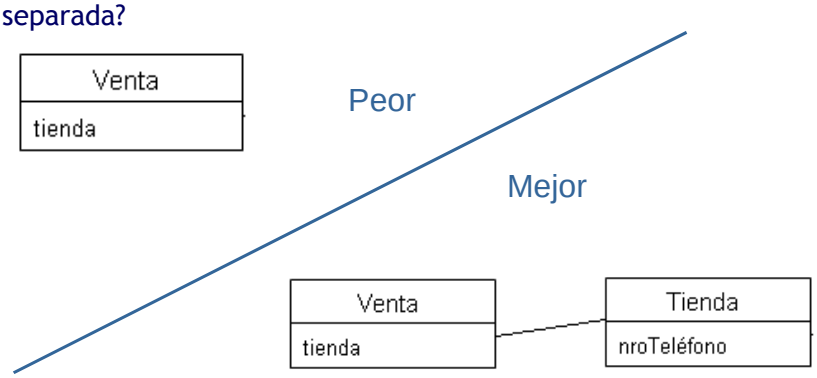
(4) Agregando asociaciones- Lista de asociaciones comunes

Categoría	Ejemplo
A es una parte física de B	No aplicable
A es una parte lógica de B	LíneaDeVenta-Venta
A está físicamente contenido en B	Terminal PDV-Tienda
A está lógicamente contenido en B	EspecificaciónDeProducto-Catálogo
A es una descripción para B	EspecificaciónDeProducto-Artículo
A es un miembro de B	Cajero-Tienda
A usa o maneja a B	Cajero-Terminal PDV
A se comunica con B	Cliente-Cajero
A está relacionado con la transacción B	Cliente-Pago Cajero-Pago
A es una transacción relacionada con otra transacción B	Pago-Venta
A es dueño de B	Tienda-Terminal PDV

110

¿ Atributo o Concepto ?

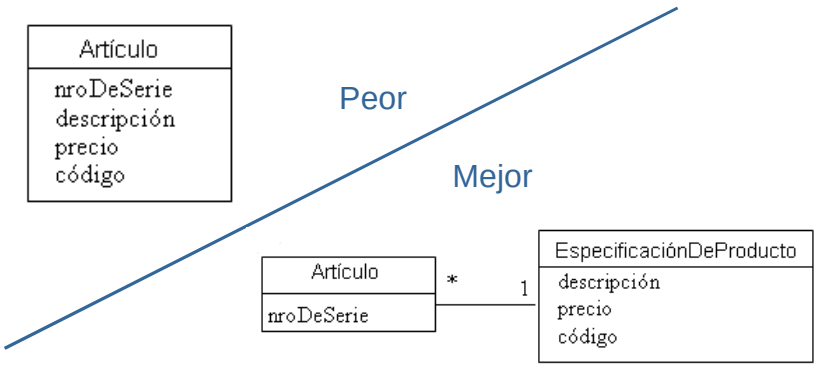
¿Debería ser la *tienda* un atributo de *venta*, o una clase conceptual separada?



Si no se consideró algún concepto X que sea número o texto en el mundo real, X es, probablemente, una clase conceptual, no un atributo.

111

¿ Todo en un concepto o un concepto con especificación?



Agregue clases conceptuales de especificación cuando necesite la descripción de un artículo o servicio, o si al eliminar las instancias que describen, se pierde información.

112

Agregando asociaciones

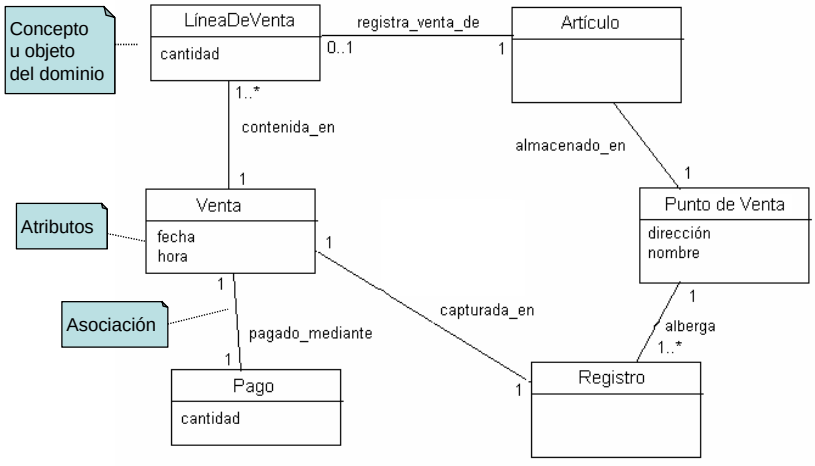
Algunos tips:

- Focalizar las asociaciones que necesitan ser preservadas por un lapso de tiempo.
- Evitar mostrar asociaciones redundantes o derivadas.
- Es más importante identificar clases conceptuales que asociaciones conceptuales.
- Demasiadas asociaciones pueden oscurecer el Modelo del Dominio.
- Recuerde agregar multiplicidades.
- Recuerde agregar roles.

113

Modelo del Dominio- Visualización de conceptos

Un **Modelo del Dominio** es una representación visual de las clases conceptuales del mundo real en un dominio de interés.



114

- **INTRODUCCIÓN**
- **ETAPA INICIAL DEL PROCESO DE DESARROLLO**
- **ELABORACION EN LA ITERACION 1**
  - Modelo de casos de uso: Representación de diagramas de secuencia
  - Modelo del dominio
  - **Modelo de casos de uso: contratos de las operaciones**
  - De los requisitos al diseño en esta iteración
  - GRASP: diseño de objetos con responsabilidades
  - Modelo de Diseño
  - Modelo de Implementación
- **ELABORACION EN LA ITERACION 2**

115

FRLP

Diseño de Sistemas

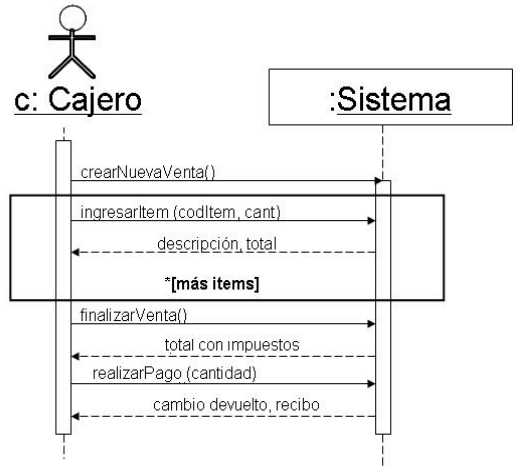
Contratos de las operaciones

Contratos: Describiendo casos de uso

Son una de las formas de describir comportamiento del sistema en forma detallada.
Describe pre y post condiciones para las operaciones.

Los DSS muestran, para un escenario de un caso de uso, los eventos que los actores generan.

Estos eventos de entrada del sistema invocan *operaciones del sistema*



117

Secciones del contrato

Son una de las formas de describir comportamiento del sistema en forma detallada.

Contrato: ingresar artículo

Operación:	ingresarArtículo (codItem: ArticuloID, cant: integer)
Referencias cruzadas:	Caso de Uso: Procesar venta
Precondiciones:	hay una venta en curso
Postcondiciones:	- se creó una instancia de LineaDeVenta ldv - ldv se asoció a la venta actual - ldv.cantidad pasó a ser cant - ldv se asoció con una EspecificaciónDelProducto, en base a la coincidencia del codItem

Secciones

118

Secciones del contrato - Postcondiciones

Las postcondiciones:

- describen cambios en el estado de los objetos del Modelo del Dominio.
- son declarativas (y expréselas así)
- se dividen en categorías:
  - Creación y eliminación de instancias
  - Modificación de atributos
  - Creación o ruptura de asociaciones

119

De los requisitos al diseño

- Describir casos de uso reales
- Crear diagramas de interacción que muestran cómo los objetos se comunican con el objetivo de cumplir con los requerimientos capturados en la etapa de análisis.
- A partir de los diagramas de interacción, diseñar diagramas de clases representando las clases que serán implementadas en software.

Crear diagramas de interacción requiere la aplicación de principios para la asignación de responsabilidades y el uso de principios y patrones de diseño.

120

Contenidos

- Módulo I: Introducción
- Módulo II: Etapa Inicial del Proceso de Desarrollo
- **Módulo III: Elaboración en la Iteración 1**
- Módulo IV: Elaboración en la Iteración 2
- Módulo V: Construcción en la Iteración 1

121

INTRODUCCIÓN

ETAPA INICIAL DEL PROCESO DE DESARROLLO

ELABORACION EN LA ITERACION 1

- Modelo de casos de uso: Representación de diagramas de secuencia
- Modelo del dominio
- Modelo de casos de uso: contratos de las operaciones
- De los requisitos al diseño en esta iteración
- **GRASP: diseño de objetos con responsabilidades**
- Modelo de Diseño
- Modelo de Implementación

ELABORACION EN LA ITERACION 2

122

Diseño de Sistemas

GRASP: diseño de objetos con responsabilidades

Responsabilidades de los objetos

- Conocer
  - Conocer sus datos privados encapsulados
  - Conocer sus objetos relacionados
  - Conocer cosas derivables o calculables
- Hacer
  - Hacer algo él mismo
  - Iniciar una acción en otros objetos
  - Controlar o coordinar actividades de otros objetos

Patrones GRASP

- La habilidad para asignar responsabilidades es extremadamente importante en el diseño.
- La asignación de responsabilidades generalmente ocurre durante la creación de diagramas de interacción.
- Los patrones GRASP son pares (problema, solución) con nombre, que codifican buenos consejos y principios que ayudan a la buena asignación de responsabilidades.

GRASP = **G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns

125

Patrón Experto en Información (Experto)

Solución: Asignar una responsabilidad al experto en información (la clase que tiene la información necesaria para realizar la responsabilidad).

Ejemplo: ¿Quién tiene la responsabilidad de conocer el monto total de una venta?
… La venta

- Expresa la intuición de que los objetos hacen cosas relacionadas con la información que tienen.
- Para cumplir con su responsabilidad, un objeto puede requerir de información que se encuentra dispersa en diferentes clases → expertos en información "parcial".

126

Patrón Creador

Solución: asignar a la clase B la responsabilidad de crear una instancia de la clase A si:

- B contiene objetos A (aggregation, composite).
- B registra instancias de A.
- B tiene los datos para inicializar objetos A.
- B usa a objetos A en forma exclusiva.

Ejemplo: ¿Quién debe ser responsable de crear una LineaDeVenta?
… La venta

- La intención del patrón Creador es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Eligiéndolo como el creador se favorece el bajo acoplamiento.

127

Patrón Controlador

Solución: asignar la responsabilidad de manejar eventos del sistema a una clase que representa:

- El sistema global, dispositivo o subsistema
- Un escenario de caso de uso, en el que tiene lugar el evento del sistema

Ejemplo: ¿Quién debe ser el controlador de los eventos *ingresarArticulo* o *finalizarVenta*?
… ProcesarVentaManejador, SistemaPDV, Registro

- La intención del patrón Controlador es encontrar manejadores de los eventos del sistema, sin recargar de responsabilidad a un solo objeto y manteniendo alta cohesion.

128

Patrón Bajo Acoplamiento

Solución: asignar responsabilidades de manera que el acoplamiento permanezca lo más bajo posible.

El **acoplamiento** es una medida de dependencia de un objeto con otros.

- El alto acoplamiento dificulta el entendimiento y complica la propagación de cambios en el diseño.
- No se puede considerar de manera aislada a otros patrones, sino que debe incluirse como principio de diseño que influye en la elección de la asignación de responsabilidad.

129

Patrón Alta Cohesión

Solución: asignar responsabilidades de manera que la cohesión permanezca lo más fuerte posible.

La **cohesión** es una medida de la fuerza con la que se relacionan las responsabilidades de un objeto, y la cantidad de ellas.

- Ventaja: clases más fáciles de mantener, entender y reutilizar.
- El nivel de cohesión no se puede considerar de manera aislada a otras responsabilidades, y otros principios los patrones Experto y Bajo Acoplamiento.

130

- **INTRODUCCIÓN**
- **ETAPA INICIAL DEL PROCESO DE DESARROLLO**
- **ELABORACION EN LA ITERACION 1**
  - Modelo de casos de uso: Representación de diagramas de secuencia
  - Modelo del dominio
  - Modelo de casos de uso: contratos de las operaciones
  - De los requisitos al diseño en esta iteración
  - GRASP: diseño de objetos con responsabilidades
  - **Modelo de Diseño**
  - Modelo de Implementación
- **ELABORACION EN LA ITERACION 2**

131

Diseño de Sistemas

Modelo de Diseño

Realizaciones de casos de uso: del Análisis al Diseño

- Los casos de uso sugieren las eventos del sistema que se muestran en los diagramas de secuencia del sistema.
- Se pueden describir detalles de los efectos de los eventos del sistema en los contratos de las operaciones.
- Los eventos del sistema representan los mensajes que inician las interacciones.
- Los diagramas de interacción muestran las interacciones entre objetos software.
- Los objetos software con sus métodos y relaciones se muestran en el Diagrama de Diseño.

Análisis
Casos de Uso
+
DSS
+
Contratos
+
Patrones GRASP

Diseño
D. Secuencia
D. Colaboración
D. Clases

133

Del Análisis al Diseño- Diagramas de interacción

- Cree un diagrama de colaboración y secuencia por cada operación del sistema en desarrollo (la operación es el mensaje de partida en el diagrama).
- Si el diagrama queda complejo, sepárelo en diagramas menos complejos (uno por cada escenario).
- Use el contrato de la operación como punto de partida; piense en objetos que colaboran para cumplir la tarea (la mayoría de estos objetos están definidos en el modelo conceptual).
- Aplique los GRASP para obtener un mejor diseño.

134

Diagramas de interacción- Ejemplo

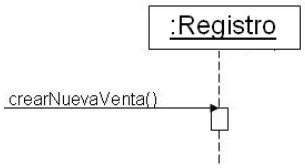
Crear una nueva venta.

… Comenzamos con el contrato de la operación.

Contrato: crear nueva venta

Operación:	crearNuevaVenta ()
Referencias cruzadas:	Caso de Uso: Procesar venta
Precondiciones:	ninguna
Postcondiciones:	- Se creó una instancia de Venta venta (creación de inst.) - venta se asoció con el Registro (formación de asoc.) - Se inicializaron los atributos de venta

… Elección de la clase controlador

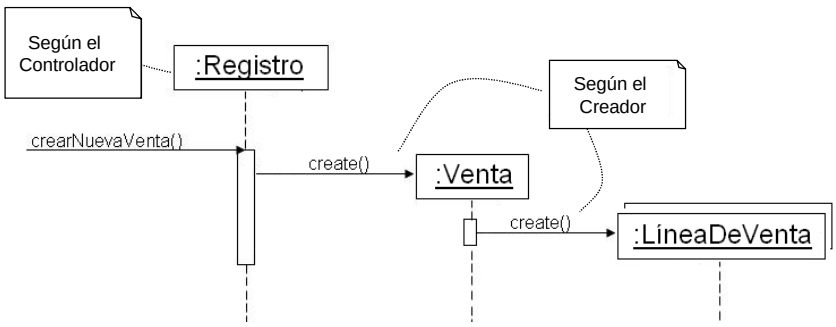


135

Diagramas de interacción- Ejemplo

Crear una nueva venta.

… Aplicando más patrones GRASP



136

Determinación de la visibilidad

Para que un objeto A envíe un mensaje a un objeto B, B debe ser visible a A.

Formas de alcanzar la visibilidad desde un objeto A a un objeto B:

- Visibilidad de atributo
- Visibilidad de parámetro
- Visibilidad local
- Visibilidad global

137

Creación de los diagramas de clases de diseño

- Los DCD muestran la especificación de clases de software de una aplicación (en lugar de clases conceptuales del dominio).
- Muestran todas las especificaciones de las clases y sus relaciones.

  Podemos encontrar:
  - Clases, asociaciones y atributos
  - Interfaces y sus operaciones
  - Métodos
  - Tipo y visibilidad de los atributos
  - Navegabilidad
  - Dependencias
- Pueden crearse en paralelo con los diagramas de interacción.

138

Creación de los diagramas de clases de diseño

- Identificar las clases que participan en los diagramas de interacción y en el Modelo del Dominio o Conceptual.
- Graficarlas en un diagrama de clases.
- Colocar los atributos presentes en el Modelo Conceptual.
- Agregar nombres de métodos analizando los diagramas de interacción.
- Agregar tipos y visibilidad de atributos y métodos.
- Agregar las asociaciones necesarias.
- Agregar roles, navegabilidad, nombre y multiplicidad a las asociaciones.

139

140

Diseño de Sistemas

Modelo de Implementación

Transformación de los diseños en código

• Mapear los artefactos de diseño a código orientado a objetos (en las primeras iteraciones, el código resultante es un prototipo del sistema real).

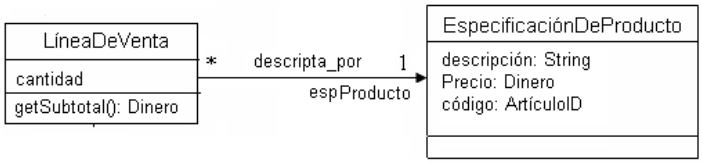
Clases		Clases
Atributos		Atributos simples
Asociaciones (roles)	→	Atributos de referencia
Métodos		Métodos
Multiobjetos		Collections

142

Transformación de los diseños en código- Ejemplo

```
public class LineaDeVenta {
        private int cantidad;
        private EspecificacionDeProducto espProducto;

        public LineaDeVenta (EspecificacionDeProducto espec, int cantidad) {
        ……
        }
        public Dinero getSubtotal(){
        ……
        }
}
```



143

Contenidos del Curso

- Módulo I: Introducción
- Módulo II: Etapa Inicial del Proceso de Desarrollo
- Módulo III: Elaboración en la Iteración 1
- **Módulo IV: Elaboración en la Iteración 2**

144

- INTRODUCCIÓN
- ETAPA INICIAL DEL PROCESO DE DESARROLLO
- ELABORACION EN LA ITERACION 1
- ELABORACION EN LA ITERACION 2
  - **La iteración 2 y sus requisitos**
  - De los requisitos al diseño en esta iteración
  - GRASP: más patrones para asignar responsabilidades
  - Organización de los paquetes del modelo de diseño
  - Modelo de Implementación

145

Diseño de Sistemas

La Iteración 2 y sus requisitos

De la iteración 1 a la iteración 2

- Se probó todo el software generado.
- Los clientes han sido involucrados para obtener retroalimentación (adaptar y clarificar requisitos).
- El sistema ha integrado y estabilizado sus subsistemas.
- Con una herramienta CASE se realizó ingeniería inversa.
- Se comenzó el análisis de **usabilidad** para las interfaces.
- Se comenzó el modelado e implementación de la base de datos.
- Se eligen los requisitos de la próxima iteración (taller de requisitos)

147

Requisitos de la iteración 2

Casos de Uso.

PDV maneja nuevos requisitos:

1. Distintos tipos de pago (tarjeta y cheque, además de efectivo)
2. Soporte a las variaciones externas de terceras partes (diferentes calculadores de impuestos).
3. Actualizar la UI cuando cambia el total de la venta.

DSS.

Inclusión de soporte para sistemas externos de terceras partes como calculadores de impuestos.

Modelo del Dominio.

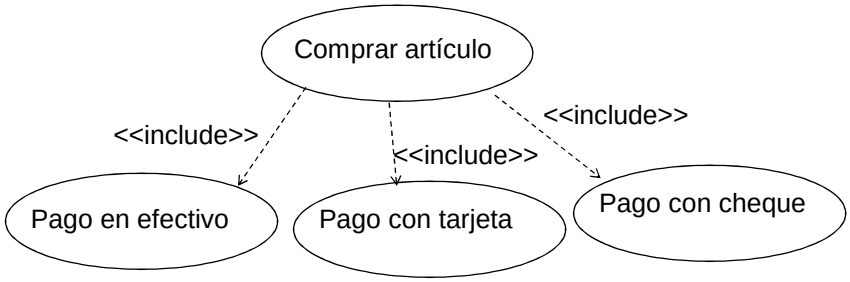
Se agregan algunos conceptos nuevos, pero se mantiene

148

Estructurando Casos de Uso

Queremos ahora que la compra de los artículos pueda abonarse de distintas formas.

¿Algo así?

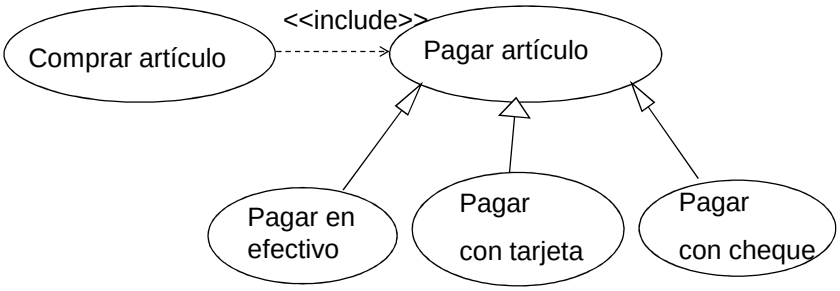


Nota: también se definen algunos casos de uso de prioridad media.

149

Estructurando Casos de Uso

O así ?



150

Extendiendo el modelo conceptual

En este segundo ciclo de desarrollo nuevos conceptos son identificados y agregados al modelo.

Ejemplo:

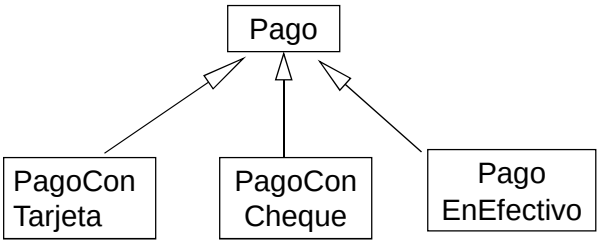
- TarjetaDeCrédito, Cheque
- PagoEnEfectivo, PagoConTarjeta, PagoConCheque
- ServicioAutorizaciónDeCrédito, etc.

151

Descubriendo jerarquías

Jerarquías generalización-especialización son analizadas y creadas.

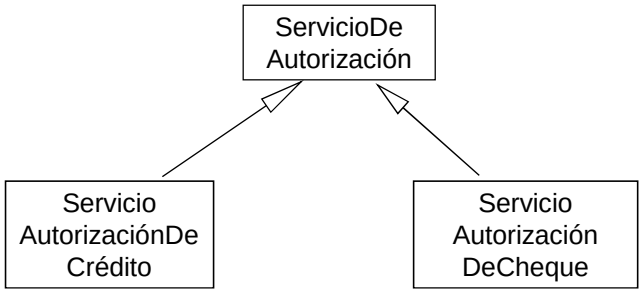
Relación ***es-un***



152

Descubriendo jerarquías

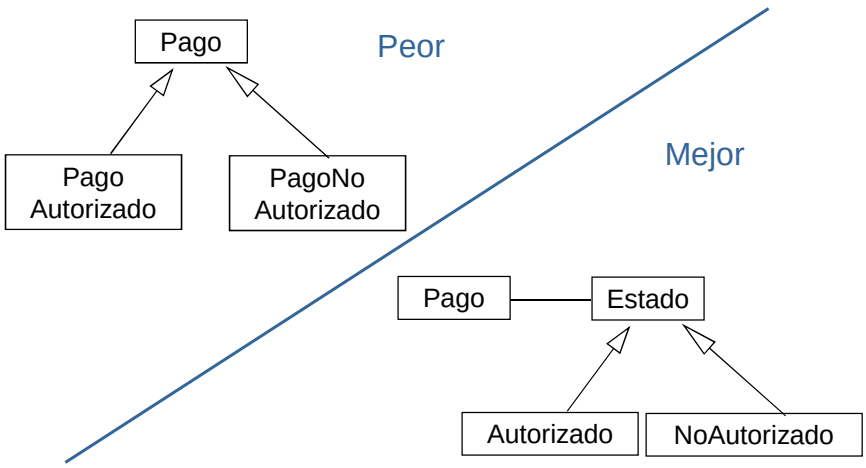
Un concepto es especializado en las subclases.
Características comunes son factorizadas en la superclase



153

Descubriendo jerarquías

¿Debería clasificar objetos por su estado, o clasificar estados?

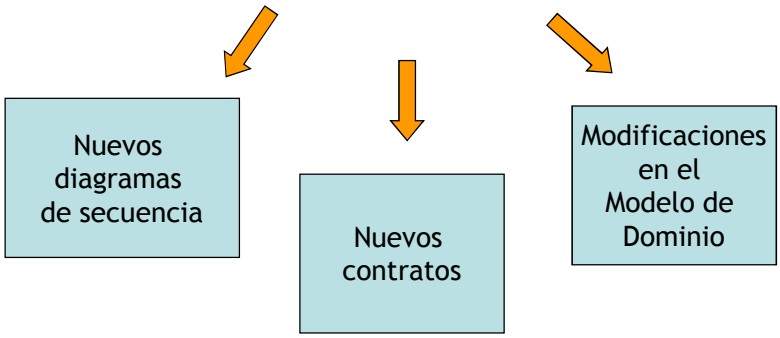


154

Agregando nuevos eventos para el sistema

En este ciclo de desarrollo se incorporan nuevos eventos para el sistema:

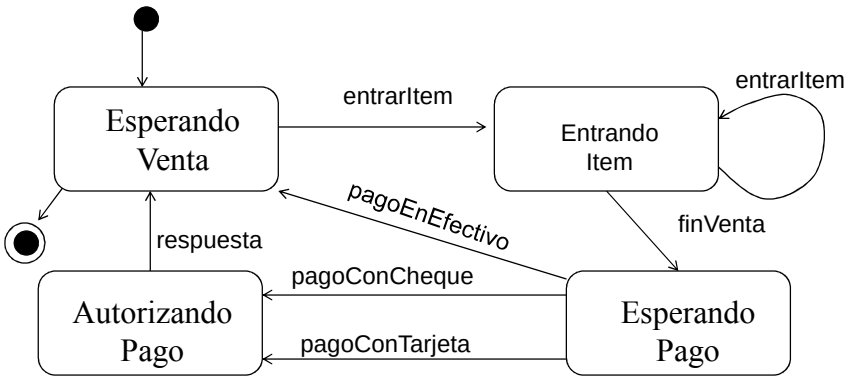
- PagarConTarjeta
- PagarConCheque



155

Modelando comportamiento con Máquinas de Estados

Caso de Uso Comprar producto



156

- **INTRODUCCIÓN**
- **ETAPA INICIAL DEL PROCESO DE DESARROLLO**
- **ELABORACION EN LA ITERACION 1**
- **ELABORACION EN LA ITERACION 2**
  - La iteración 2 y sus requisitos
  - **De los requisitos al diseño en esta iteración**
  - GRASP: más patrones para asignar responsabilidades
  - Organización de los paquetes del modelo de diseño
  - Modelo de Implementación

157

Diseño de Sistemas

De los requisitos al diseño

En resumen...

En la fase de análisis de la iteración n-ésima se realizan las siguientes actividades:

- Se agregan nuevos casos de uso.
- Se refinan - extienden - estructuran casos de uso anteriores.
- Se extiende el modelo conceptual con nuevos conceptos.
- Se revisan las asociaciones del modelo (multiplicidad, roles, etc.)
- Se re-organiza el diagrama de clases (jerarquías).
- Se refina la especificación de las funciones del sistema (diagramas de secuencia . y contratos - máquinas de estado del sistema-)

159

De los requisitos al diseño

- Describir casos de uso reales
- Crear diagramas de interacción que muestran cómo los objetos se comunican con el objetivo de cumplir con los requerimientos capturados en la etapa de análisis.
- A partir de los diagramas de interacción, diseñar diagramas de clases representando las clases que serán implementadas en software.

Crear diagramas de interacción requiere la aplicación de principios para la asignación de responsabilidades y el uso de principios y patrones de diseño.

160

- **INTRODUCCIÓN**
- **ETAPA INICIAL DEL PROCESO DE DESARROLLO**
- **ELABORACION EN LA ITERACION 1**
- **ELABORACION EN LA ITERACION 2**
  - La iteración 2 y sus requisitos
  - De los requisitos al diseño en esta iteración
  - **GRASP: más patrones para asignar responsabilidades**
  - Organización de los paquetes del modelo de diseño
  - Modelo de Implementación

Diseño de Sistemas

GRASP: diseño de objetos con responsabilidades

Patrón Polimorfismo

Solución: cuando el comportamiento varía según el tipo, asigne la responsabilidad para el comportamiento a los tipos para los que varía el comportamiento.

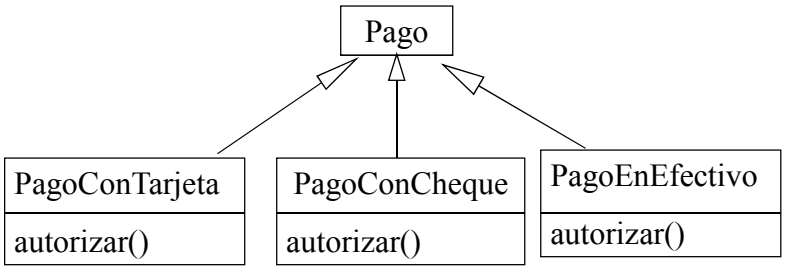
Ejemplo: El sistema PDV debe soportar distintas formas de pago.

… Como la autorización del pago varía según su tipo deberíamos asignarle la responsabilidad de autorizarse a los distintos tipos de pagos

- Nos permite sustituir objetos que tienen idéntica interfase

163

Patrón Polimorfismo



164

Patrón Fabricación Pura

Solución: asigne responsabilidades a una clase "de conveniencia" que no representa un concepto del dominio y que soporte alta cohesión, bajo acoplamiento y reutilización.

Ejemplo: El sistema PDV necesita almacenar todas las ventas en una base de datos relacional.

… Se crea una nueva clase cuya responsabilidad es la de almacenar objetos en algún tipo de medio de almacenamiento persistente.

- La venta permanece bien diseñada, con alta cohesión y bajo acoplamiento.

165

Patrón Indirección

Solución: asigne la responsabilidad a un objeto intermedio que medie entre otros componentes o servicios de manera que no se acoplen directamente.

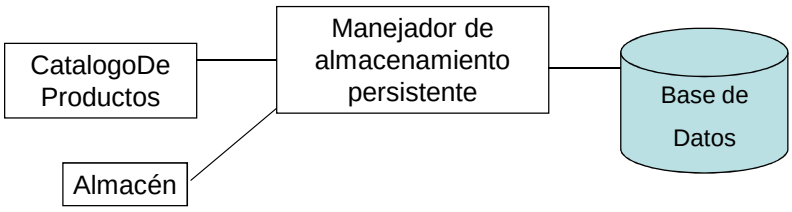
Ejemplo: El sistema PDV necesita almacenar todas las ventas en una base de datos relacional.

… Se crea una nueva clase cuya responsabilidad es la de almacenar objetos en algún tipo de medio de almacenamiento persistente.

- La clase que se ocupa del almacenamiento persistente de los objetos es un intermediario entre la *Venta* y la *base de datos*.

166

Patrones Fabricación Pura e Indirección



167

"No hables con extraños"

Evite crear diseños que recorren largos caminos de estructura de objetos y envía mensajes (habla) a objetos distantes o indirectos (extraños).

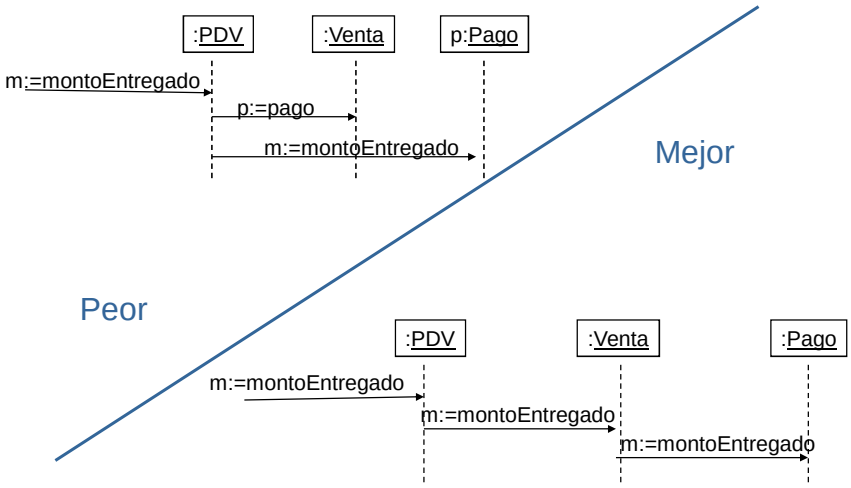
Dentro de un método sólo pueden enviarse mensajes a objetos conocidos:

- self
- un parámetro del método
- un objeto que esté asociado a self
- un miembro de una colección que sea atributo de self
- un objeto creado dentro del método

Los demás objetos son extraños (strangers)

168

“No hables con extraños”



169

- **INTRODUCCIÓN**
- **ETAPA INICIAL DEL PROCESO DE DESARROLLO**
- **ELABORACION EN LA ITERACION 1**
- **ELABORACION EN LA ITERACION 2**
  - La iteración 2 y sus requisitos
  - De los requisitos al diseño en esta iteración
  - GRASP: más patrones para asignar responsabilidades
  - **Organización de los paquetes del modelo de diseño**
  - Modelo de Implementación

170

FRLP

Diseño de Sistemas

Organización de los paquetes del modelo de diseño

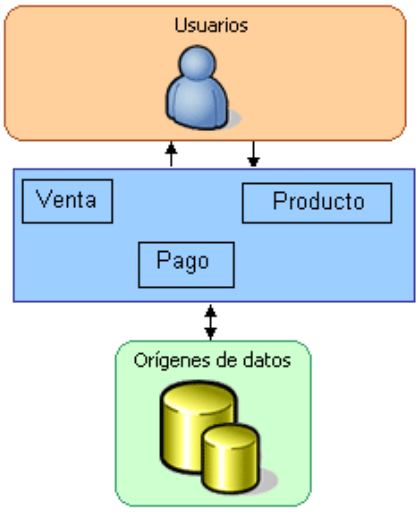
Arquitectura de tres capas

Paquetes - Partición vertical

Capa de presentación

Capa de negocio

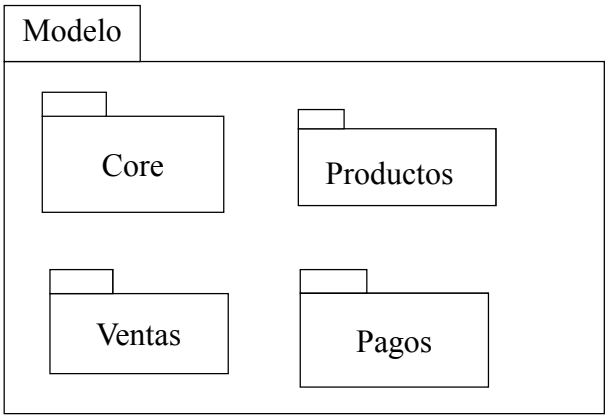
Capa de datos



172

Paquetes del dominio (modelo)

Paquetes - Partición horizontal



Las clases del modelo se diseñan independientemente de las clases de la interfaz

173

En resumen...

En la fase de diseño de la iteración n-ésima se realizan las siguientes actividades:

- Se organizan los elementos del modelo en paquetes (horizontal y vertical).
- Se asignan responsabilidades a los objetos del sistema
- Se refinan / completan los diagramas de interacción/ colaboración/máquinas de estado, mostrando la realización de los casos de uso y contratos de análisis.
- Se mejora el diagrama de clases.

174

Diseño de Sistemas

Modelo de Implementación

Transformación de los diseños en código

• Mapear los artefactos de diseño a código orientado a objetos (en las primeras iteraciones, el código resultante es un prototipo del sistema real).

Clases		Clases
Atributos		Atributos simples
Asociaciones (roles)	→	Atributos de referencia
Métodos		Métodos
Multiobjetos		Collections

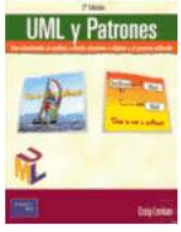
176

Diseño de Sistemas

En la siguiente iteración se deberán aplicar restricciones del dominio a través de cláusulas de validación OCL.

Bibliografía y textos recomendados

- Para repasar o revisar en detalle temas de RUP, alcance de Análisis OO y Diseño OO, se sugiere leer secciones de los libros:



178