Ingeniería del software, procesos ágiles y análisis de requisitos

Resumen de contenidos

- Fases del desarrollo de una aplicación: ingeniería del SW y metodologías.
- Conceptos clásicos de Ingeniería del Software; modelo en cascada.
- Procesos iterativos: características y ventajas.
- Características y fases del proceso unificado
- Otros procesos ágiles: características.
- Técnicas de análisis y captura de requisitos.

Ingeniería del software

- **Ingeniería de software** es la disciplina o área de la informática que ofrece métodos y técnicas para desarrollar y mantener software de calidad.
- Inicialmente la programación era una tarea cercana al arte. Se descubrió rápidamente que eran necesarias metodologías que sistematizaran y estandarizaran los procesos de creación de software.

La crisis del software

- Englobó a una serie de sucesos que se venían observando en los proyectos de desarrollo de software (décadas de los 60,70 y 80):
  - Programación considerada más como un **arte**, que una disciplina procedimental y metódica.
  - Los proyectos no terminaban en **plazo**.
  - Los proyectos no se ajustaban al **presupuesto** inicial.
  - Baja **calidad** del software generado.
  - Software que no cumplía las **especificaciones**.
  - Código **inmantenible** que dificultaba la gestión y evolución del proyecto.
  - Ejemplos de **fracasos** épicos y letales (F18, Therac 25)

Ciclo de vida

- Es un modelo teórico que detalla las fases por las que ha pasar el desarrollo de un software, así como de las tareas a realizar en cada una de ellas.
- Ha habido muchas propuestas de distintos ciclos de vida, algunas más acertadas que otras, algunas compuestas de varias y otras inspiradas en algunas de las anteriores.

Modelo en cascada

- **Extraído de los procesos ingenieriles clásicos.**
- **Fases:**
  - Análisis
  - Diseño
  - Implantación
  - Pruebas y mantenimiento
- **Problemas.**
  - No apropiado al mundo del software.
  - No guiado por requisitos
  - Muy rígido. No hay marcha atrás. Necesaria mucha seguridad en cada fase
  - Proyectos muy largos. Cliente no percibe el fruto de su inversión hasta pasado mucho tiempo.

Otros ciclos de vida "clásicos"

- **Cascada con realimentación**
  - Intenta mejorar el modelo en cascada introduciendo marcha atrás entre las distintas fases.
  - Introduce demasiada complejidad contra la sencillez del modelo en cascada, muchas veces no justificable.
- **Prototipos**
  - Se invierte tiempo del desarrollo construyendo “maquetas” que ayudan al cliente a comprender las dimensiones del proyecto y concretar sus funcionalidades.
  - El inconveniente es que esos prototipos cuestan tiempo y dinero a veces no justificable.
- **Espiral**
  - Las fases no se suceden de forma lineal. Se cicla alrededor de un conjunto de 4 fases, que conllevan cierta complejidad, pero que sentaron las bases de los actuales procesos iterativos

Ciclos de vida iterativos

- Basados en mini-proyectos, llamados iteraciones.
- A diferencia de los modelos prototipados, aquí el prototipo es parte del producto final.
- Guiados por requisitos y el riesgo.
- Productos finales más fiables.
- Gran interacción con el usuario: garantía del éxito.
- Más adecuado a tecnologías modernas orientadas a objetos.

Metodologías

- Definen con detalle las fases, tareas y documentación a generar para llevar a cabo un proyecto de desarrollo informático.
- Se basan en algún tipo de ciclo de vida o modelo, o una mezcla de varios.
- Cada empresa, país o administración suele comprometerse con una determinada metodología para sus desarrollos.
- De esta manera, se habitúan a un modo de trabajo homogéneo para proyectos heterogéneos (mismos diagramas, mismos documentos, mismas métricas, etc.)
- Ejemplos de metodologías:
  - ***Merise*** (gobierno francés)
  - ***Métrica*** (gobierno español-adm.públicas)
  - ***Marte*** (telefónica España)
  - Ágiles (usadas por muchas empresas actualmente):
    - UP (***RUP***, ***AUP***, ***OpenUP***)
    - eXtreme Programming (***XP***)
    - ***SCRUM***

La ingeniería del SW en la actualidad

- Liderada por las llamadas “metodologías ágiles”
- Fundamentadas en el desarrollo iterativo
- Desarrolladas para subsanar las complejidades y carencias de las ya existentes.
- No son tan estructuradas ni estrictas, pero tampoco son caóticas.
- Enfatizan la comunicación “cara a cara” por encima de la documentación.
- Aprovechan y potencian las nuevas tecnologías de programación (OOP, AOP, SOA, frameworks, etc)

Algunas metodologías ágiles

- Adaptive Software Development (ASD).
- Agile Unified Process (AUP).
- Crystal Clear.
- Essential Unified Process (EssUP).
- Feature Driven Development (FDD).
- Lean Software Development (LSD).
- Kanban.
- Open Unified Process (OpenUP).
- Programación Extrema (XP).
- **Scrum**.

El manifiesto ágil

- Principios en los que se basa cualquier metodología que se considere a sí misma como ágil.

1. Valorar al individuo y las interacciones con el equipo de desarrollo más que a las actividades y herramientas.
2. Desarrollar SW que funcione, más que conseguir una buena documentación.
3. Favorecer la colaboración con el cliente por encima de la negociación de contratos.
4. Responder a los cambios más que seguir una planificación

Proceso unificado

- Desarrollado en los capítulos 2, 4, 7 y 8 de “UML y patrones” de Craig Larman (Prentice Hall)
- ***U.P.*** (Unified process): Metodología basada en el desarrollo iterativo, desarrollada por Jacobson, Rumbaugh y Booch (padres de la OOP)
- ***R.U.P.*** (Rational unified process). Adaptación y refinamiento detallado de UP, realizado por la empresa “Rational”
- Las ***metodologías ágiles*** (muy de moda) como “Extreme programming (XP)” o “Scrum” están basadas en UP (o en AUP), y a la postre en procesos iterativos.

Jacobson, Rumbaugh y Booch

- Padres de la O.O.P., el U.P. y gran parte de las teorías y tecnologías actuales de desarrollo.

Ivar **Jacobson**

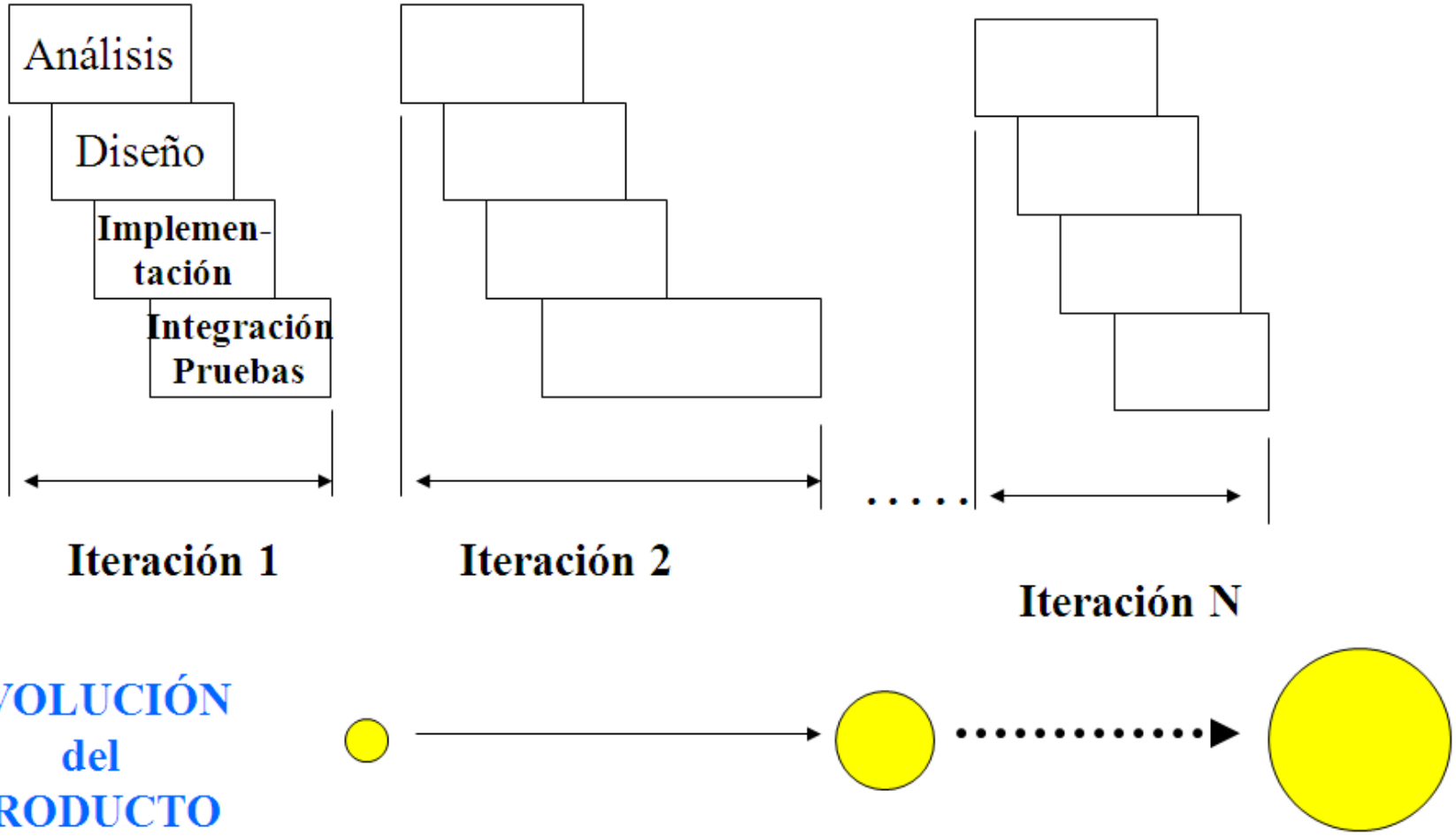
James **Rumbaugh**

Grady **Booch**

Desarrollo iterativo

- Principio común a todas las metodologías ágiles
- Desarrollo organizado en **mini-proyectos** llamados ITERACIONES.
- Cada iteración dura aprox. **2-6 semanas**.
  - En equipos grandes (cientos de programadores), pueden ser de hasta 6 meses, pero es la excepción, e incluso en estos casos, cada subequipo divide su iteración en subiteraciones.
- **No son prototipos**. Son refinamientos del sistema final.
- Al final de cada iteración se marcan los objetivos y duración de la siguiente.
- (U.P.) Dirigido por el **riesgo**.
- (U.P.) Centrado en la arquitectura.

Progreso de las iteraciones



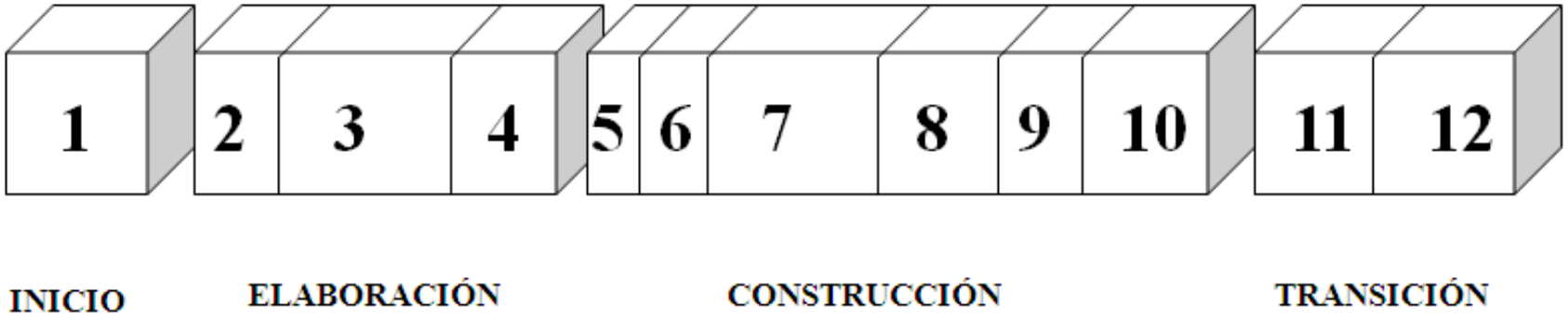
Ventajas del desarrollo iterativo

- Progreso visible en las primeras etapas
- Retroalimentación temprana y frecuente.
- (UP) Reducción de riesgos altos (técnicos, requisitos, objetivos, usabilidad, etc.)
- Más ajustado a las necesidades reales del cliente.
- Equipo de desarrollo motivado al ver progreso.
- El conocimiento adquirido en una iteración se puede reutilizar en el resto del desarrollo.

Fases del proceso unificado

- **INICIO**: Visión aproximada, análisis del negocio, alcance, estimaciones imprecisas
- **ELABORACIÓN**: Visión refinada. Construcción del núcleo central (+riesgo). Estimaciones más realistas
- **CONSTRUCCIÓN**: Implementación resto requisitos (- riesgo)
- **TRANSICIÓN**: Pruebas beta, despliegue.

Iteraciones dentro del U.P.



Artefacto

- Define ,de forma genérica, un producto concreto, fruto del trabajo en el desarrollo de un proyecto de software.
- Son los “entregables” que recibe el cliente en contraprestación al dinero que paga.
- Normalmente se suele relacionar con un documento (especificación de requisitos, diagrama de arquitectura, etc)
- No tienen tamaño fijo: pueden ocupar desde una página hasta cientos de ellas.
- No son sólo literatura: podrían ser también diagramas, dibujos o ficheros multimedia.
- El propio código fuente y el ejecutable se pueden considerar artefactos. De hecho es el más importante de todos (el que más le interesa al cliente)

Artefactos en el Proceso Unificado

Disciplina	Artefacto	Inicio I1	Elaboración E1 … En	Construcción C1 … Cn	Transición T1 … n
Modelado de negocio	**- Modelo de dominio**		c		
Requisitos	• **Casos de uso** • **Visión** • **Espec. complementaria** • **Glosario**	c c c c	r r r r		
Diseño	• **Modelo de diseño** • **Doc. Arq.SW** • **Modelo de datos**		c c c	r r	
Implementación	**- Mod.implementación**		c	r	r
Gestión proyecto	**- Plan desarrollo SW**	c	r	r	r
Pruebas	**- Mod. de pruebas**		c	r	
Entorno	**- Marco de desarrollo**	c	r		

Inicio: características

- Objetivo
  - Visión y análisis
  - ¿Viable?
  - ¿Comprar y/o construir?
  - Estimación aprox. coste: 10K -100K o millones?
  - ¿Seguimos adelante?
- Similar plantearse prospección petrolífera
- Duración breve o incluso inexistente
- Sólo es necesario definir en detalle un 10% (aprox.) de los casos de uso
- No hay imposiciones restrictivas de documentación: generar sólo los artefactos que sean interesantes.

Inicio: artefactos

- **Visión y análisis del negocio:** Objetivos y restricciones de alto nivel. Análisis de negocio, e informe para toma de decisiones.
- **Modelo de casos de uso:** Requisitos funcionales y no funcionales relacionados
- **Especificación complementaria:** Otros requisitos
- **Glosario:** Terminología clave del dominio.
- **Lista riesgos y plan gestión riesgo:** Técnicos, recursos, planificación e ideas para mitigarlos.
- **Prototipos y prueba de conceptos:** Para clarificar visión
- **Plan de iteración:** Qué hacer en la primera iter.de la elaboración.
- **Plan de fase y desarrollo de SW:** Estimación poco precisa fase elaboración. Herramientas, personas, formación y otros recursos.
- **Marco de desarrollo:** Describe pasos UP y artefactos adaptados a este proyecto

Elaboración: objetivos

- (Pág 103 Larman)
- Descubrir y estabilizar la mayoría de los requisitos.
- Reducir o eliminar los riesgos importantes.
- Implementar y probar elemento básicos de la arquitectura.

Elaboración: características deseables

- **Duración** de 2-4 iteraciones de 2-4 semanas cada una.
- **Iteraciones** fijas, breves, dirigidas por riesgo
- Comenzar a **programar** pronto
- Diseñar, implementar y probar de manera adaptable **partes básicas y arriesgadas** de la arquitectura.
- **Probar** desde el principio, a menudo y realísticamente.
- **Adaptar** en base a la **retroalimentación** procedente de pruebas, usuarios y desarrolladores.
- Escribir mayoría de los **casos de uso** y otros requisitos en detalle, a través de **talleres,** uno por iteración.

Elaboración: tareas arquitecturales

- **Diseño e implementación “ancho y superficial”**
  - Identificar procesos, capas, paquetes, subsistemas, interfaces alto nivel.
  - Implementación parcial para probar las conexiones.
- **Refinamiento interfaces locales y remotas (106)**
  - P.ej. Interfaz acceso a sistemas de contabilidad de terceros
  - Provocar fallos en interfaces para estabilizarlas
- **Integración de los componentes existentes**
- **Implementación de escenarios simplificados (caso de éxito) de CdUs.**
  - Necesaria la clasificación previa de las iteraciones y los requisitos (CdU) por niveles de riesgo (técnico), cobertura y naturaleza crítica

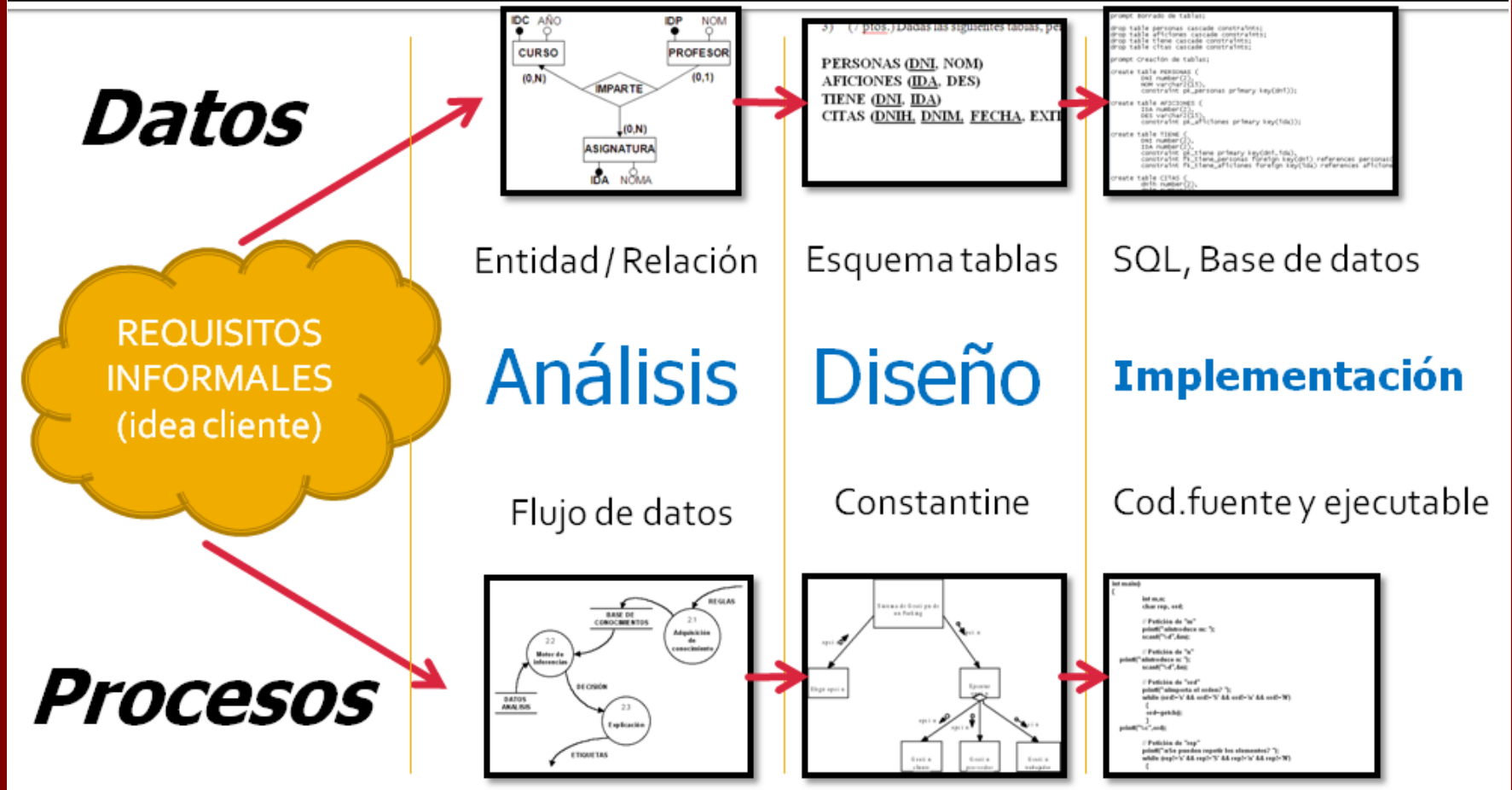
Elaboración: artefactos

- **Modelo de dominio:** Visualización conceptos dominio.
- **Modelo de diseño:** DCD, diagramas interacción, paquetes, etc.
- **Doc. Arquitectura SW:** Resumen ideas clave arquitectura.
- **Modelo de datos:** Esquemas BD, estrategias transformación obj.
- **Modelo de pruebas:** Descripción qué se probará y cómo
- **Modelo implementación:** Código, BD, etc.
- **Guiones CdU, prototipos UI:** Descripción UI, caminos de navegación, modelos de facilidad uso, etc.

Técnicas clásicas de análisis y captura de requisitos

Ingeniería de requisitos

Fases de desarrollo (enfoque estructurado)



Técnicas de análisis de requisitos (ingeniería de requisitos)

- Se utilizan para fijar de manera sólida las funcionalidades que ha de materializar la aplicación informática a desarrollar.
- Para que los requisitos sean correctos, han de ser medibles, comprobables, sin ambigüedades o contradicciones, etc.
- Se centran en la pregunta ¿qué? evitando plantear la pregunta ¿cómo?

Fases de recogida de requisitos

- **Obtener requisitos:** A través de entrevistas o comunicación con clientes o usuarios, para saber cuáles son sus deseos.
- **Analizar requisitos:** Detectar y corregir las carencias comunicativas, transformando los requisitos obtenidos de entrevistas y requisitos, en condiciones apropiadas para ser tratados por el diseño.
- **Documentar requisitos:** Igual que todas las etapas, los requisitos deben estar debidamente documentados. (diag.E/R, DFD's)
- **Verificar / validar los requisitos:** Comprobar que los requisitos implementados se corresponden con lo que inicialmente se pretendía. No más de dos-tres acciones de recogida de requisitos sobre el mismo cliente en esta primera fase. Firma del cliente.

Técnicas

- **Observación** y documentación previa.
- **Encuestas**.
- **Entrevistas** (personales, grupos-talleres)
  - Directivos
  - Usuarios
  - Responsables de terceras partes involucradas.
- **Prototipos**

Observación (1/3)

- Observar los procedimientos operacionales actuales. Ver el sistema en acción.
- Planificar las tareas a observar.
- Hacer una lista de todo lo que se desea saber.
  - Elaborar preguntas sobre situaciones no esperadas o no cubiertas por procedimientos estándares.

Observación (2/3)

- Observar todos los pasos en una transacción y anotar los documentos, inputs, outputs y procesos involucrados.
- Examinar cada formulario, registro e informe.
- Considerar cada usuario del sistema y la interacción con otros usuarios, qué información reciben.

Observación (3/3)

- Hablar con las personas que reciben los informes y observar si éstos están completos, a tiempo, certeros, y si son útiles.
- Tener presente que las operaciones tal vez no funcionen normalmente porque los trabajadores puedan estar nerviosos.

Cuestionarios y encuestas

- Son documentos que contienen una serie de preguntas estándar que se les envía a muchos individuos.
- Incluyen
  - encabezado con el título
  - propósito
  - nombre y teléfono de la persona de contacto
  - fecha límite para completarlo y cómo devolverlo.

Reglas para redactar encuestas (1/3)

- Corto, amigable y sencillo
- Proporcionar instrucciones que anticipen preguntas.
- Arreglar las preguntas en orden lógico de menos a más complejo.
- Utilizar términos simples para evitar malos entendidos.

Reglas para redactar encuestas (2/3)

- No dirigir las contestaciones ni dar pistas para obtener respuestas deseadas.
- Limitar el uso de preguntas abiertas que son difíciles de tabular.
- Limitar las preguntas relacionadas a la seguridad en el empleo u otros asuntos negativos, aunque sin evitar temas sensibles, si éstos parecen decisivos en la aplicación.

Reglas para redactar encuestas (3/3)

- Incluir una sección de comentarios al final de la encuesta.
- Probar la encuesta en un pequeño grupo de personas antes de distribuirla a un grupo mayor.

Entrevistas (1/7)

- Determinar las personas que se van a entrevistar.
  - Estructuras formales / informales
  - Entrevistas individuales / grupales
  - ¿Gerencia en la entrevista?
- Establecer los objetivos de la entrevista.
  - Lista de tópicos a discutir

Entrevistas (2/7)

- Desarrollar las preguntas de la entrevista.
  - Lista estándar de preguntas
  - Preguntas Abiertas – estimula respuestas no estructuradas y espontáneas
    - ¿Funciona el sistema apropiadamente?
    - ¿Cómo se realiza esta tarea?
    - ¿Por qué esa tarea se realiza de esa forma?

Entrevistas (3/7)

- Preguntas Cerradas – limitan o restringen la respuesta.
  - ¿Cuántos ordenadores hay?
  - ¿Cuánto tiempo se puede dedicar a esta labor como máximo?
- Preguntas con Rango de Respuestas - preguntas cerradas cuya respuesta es limitada o establecida por una escala numérica.

Entrevistas (4/7)

- Prepararse para la entrevista.
  - Establecer día y hora, y un recordatorio de confirmación.
  - Enviar lista de preguntas esenciales. Evitamos así, que ciertas preguntas se queden sin respuesta por falta de datos.
  - Lugar de la entrevista, oficina del entrevistado o sitio neutral.

Entrevistas (5/7)

- Realizar la entrevista.
  - Presentación, descripción del proyecto, objetivos de la entrevista.
  - Dar suficiente tiempo para la respuesta. Establecer armonía.
  - Escuchar atentamente.
  - Seguir el orden preparado para la entrevista.
  - Realizar un resumen de los puntos principales de la entrevista.

Entrevistas (6/7)

- Documentar la entrevista.
  - Limitar el tomar notas.
  - Hacer anotaciones al finalizar la entrevista.
    - Las entrevistas se olvidan en 30 minutos. No planificar entrevistas yuxtapuestas.
  - Explicar uso de grabadoras.
  - Enviar resumen de la entrevista.

Entrevistas (7/7)

- Evaluar la entrevista.
  - Identificar tendencias, inclinaciones.
    - Respuestas incompletas.
    - Distorsión de hechos.
    - Evitar dar información voluntaria.

Prototipos

- Son maquetas incompletas del producto final.
- Normalmente incluyen gran parte de la interfaz de usuario, y algunas funcionalidades o simulaciones de las mismas.
- Sirven para obtener del cliente “feedback” sobre del producto final (al menos de su apariencia).
  - Sugerirá nuevas funcionalidades que faltan, o algunas que sobran.
  - Defectos en el “workflow”
  - Aspecto del interfaz de usuario.

Tipos de requisitos

- **Funcionales**: lo que el usuario necesita que haga el software.
- **No funcionales**: limitaciones, restricciones (plataformas, consideraciones sobre rendimiento, etc.)

FURPS +

- Clasificación FURPS+
  - **F**UNCTIONAL: características, capacidades, seguridad
  - **U**SABILITY: factores humanos, ayuda, documentación.
  - **R**ELIABILITY: Frecuencia fallos, capacidad recuperación
  - **P**ERFORMANCE: tiempo respuesta, productividad, disponibilidad, uso recursos
  - **S**UPPORTABILITY: Adaptabilidad, mantenimiento, internacionalización, configurabilidad

+

  - **Implementación:** Limitación recursos HW, SW, lenguajes, herramientas
  - **Interfaz:** Restricciones para interacción con sistemas externos.
  - **Operaciones:** Gestión de la puesta en marcha del sistema
  - **Empaquetamiento:**
  - **Legales:** Licencias, etc.

Problemas (cliente)

- No tiene claro lo que desean
- No se involucran en la elaboración de requisitos escritos
- Insisten en nuevos requisitos después de que el coste y la programación se hayan fijado.
- La comunicación con los usuarios es lenta
- No participan en revisiones o son incapaces de hacerlo.
- No comprenden los problemas técnicos
- No entienden el proceso del desarrollo

Problemas (analista)

- Uso de terminología ambigua en la redacción de los documentos de requisitos
- Sobreespecificación de los requisitos
- Escritura poco legible, voz pasiva, abuso de negaciones
- Uso de verbos en condicional, expresiones subjetivas
- Ausencia de términos y verbos del dominio de la aplicación (regla del cartógrafo)

CASOS de USO

Técnicas, documentación y diagramas UML

Casos de uso

- Ideados por Jacobson en 1986
- Son historias narradas del uso del sistema para alcanzar un objetivo concreto.
- El modelo de casos de uso compila todas esas “historias”.
- Formato breve:

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.

CdU:elementos y formato

- Son simples y útiles. Cuanto más se compliquen, peor.
- Elementos
  - **Actor:** Algo con comportamiento (persona, sistema "externo", organización, etc.). Existen actores principales, secundarios y pasivos.
  - **Escenario:** Secuencia específica de acciones e interacciones entre los actores y el SuD (system under description). Existe un escenario de **éxito** y un conjunto de escenarios alternativos (p.45)
- Formato completo (ver)

CdU: otros requisitos

- Requisitos especiales:
  - No funcionales
  - Atributo de calidad
  - Restricción que se relaciona de forma especial con el CdU
  - Rendimiento, fiabilidad, facilidad de uso y restricciones de diseño (en dispositivos E/S)
  - Junto con su CdU y en la “Especificación complementaria”
- Lista de tecnología y variaciones de datos.
  - Tecnología que nos vemos obligados a utilizar según el entorno.
  - Formatos que nos vemos obligados por ley o por necesidades del negocio a implementar.

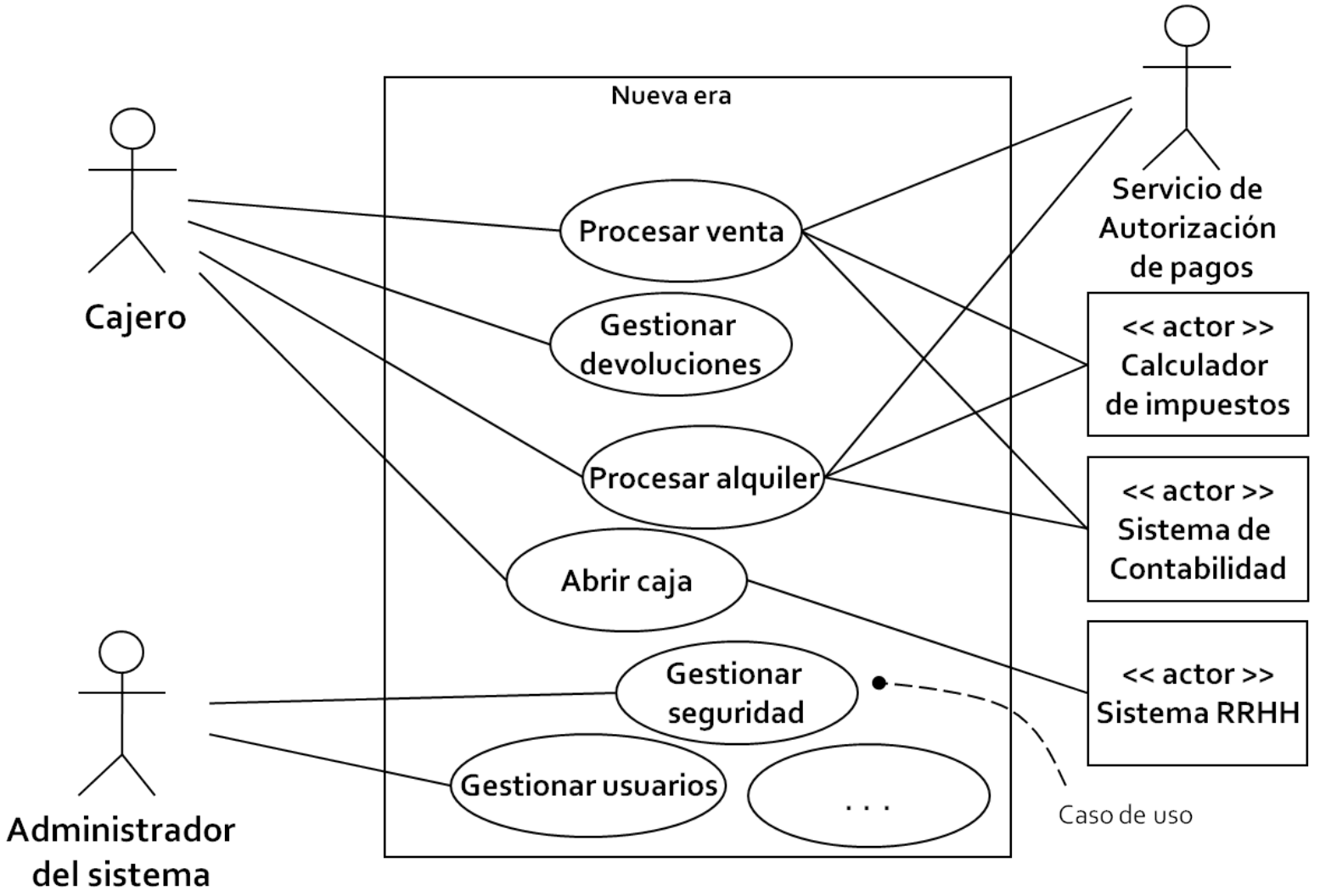
Procesos elementales de negocio (E.B.P.)

- Para poder diferenciar qué es un caso de uso y qué no (porque podrían ser a diferentes niveles), es aconsejable concentrarse en los EBP.
- E.B.P. (elementary business process)
  - **Una tarea realizada por una persona en un instante como respuesta a un evento del negocio, que añade un valor cuantificable para el negocio y deja los datos en un estado consistente.**
    - No pequeño paso como “eliminar línea de pedido” o “mostrar datos del documento”. (escenario principal de 5-10 pasos)
    - No tarda días y múltiples sesiones como “Negociar contrato con proveedor”.
  - Importante: Valor observable y datos en estado estable y consistente

Trucos para identificar CdU

- Preguntas útiles para encontrar actores principales y objetivos de usuario.
  - ¿Quién arranca / para el sistema?
  - ¿Quién gestiona a los usuarios y seguridad?
  - ¿Quién se encarga de la admin. del sistema?
  - ¿Es un actor el tiempo (Sistemas tiempo real)?
  - ¿Quién evalúa la actividad / rendimiento del sistema?
  - ¿Existe un proceso de control que reinicie el sistema si falla?
  - ¿Cómo se gestionan las actualizaciones de SW? ¿Automáticas?
  - ¿Quién evalúa los registros? ¿Se recuperan de forma remota?

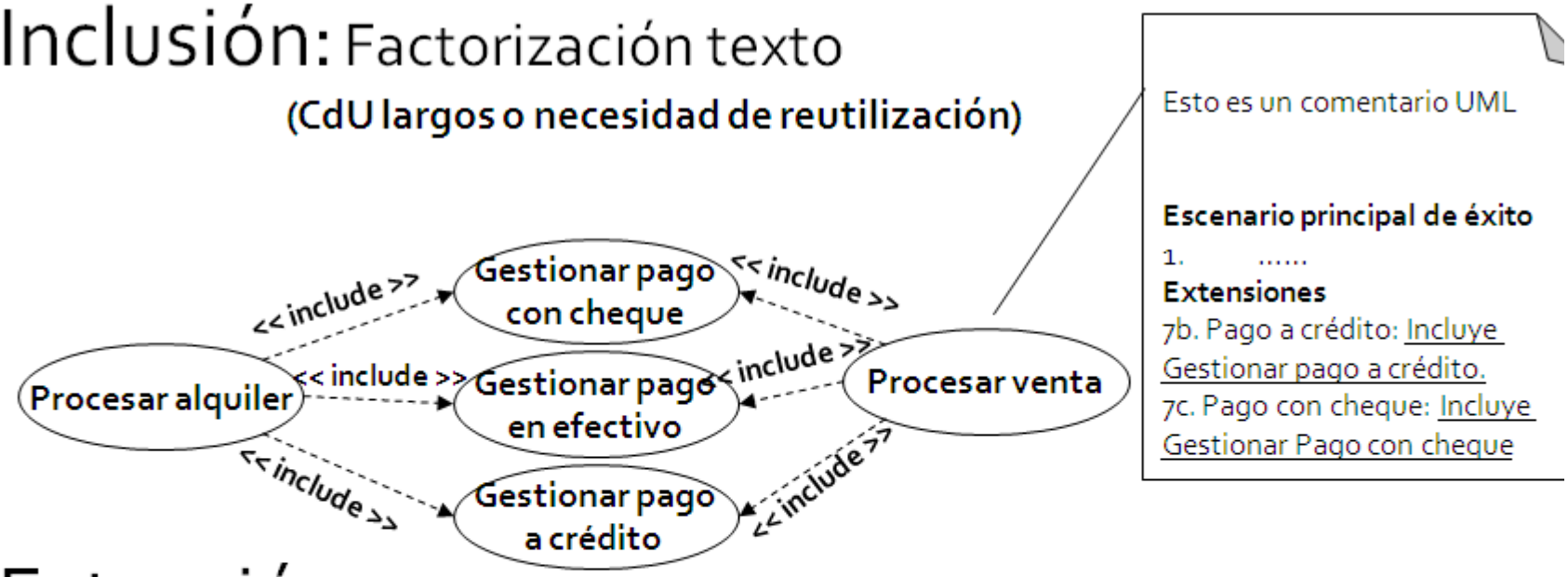
Diagramas de CdU



CdU: inclusión y extensión

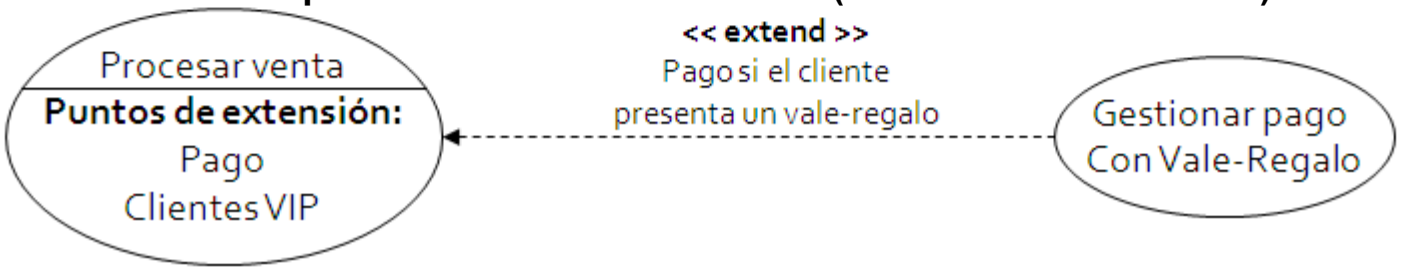
- **Inclusión:** Factorización texto

(CdU largos o necesidad de reutilización)



- **Extensión:** Nuevo texto para CdU estable

Dependen del caso de uso base (no tienen sentido sin él)



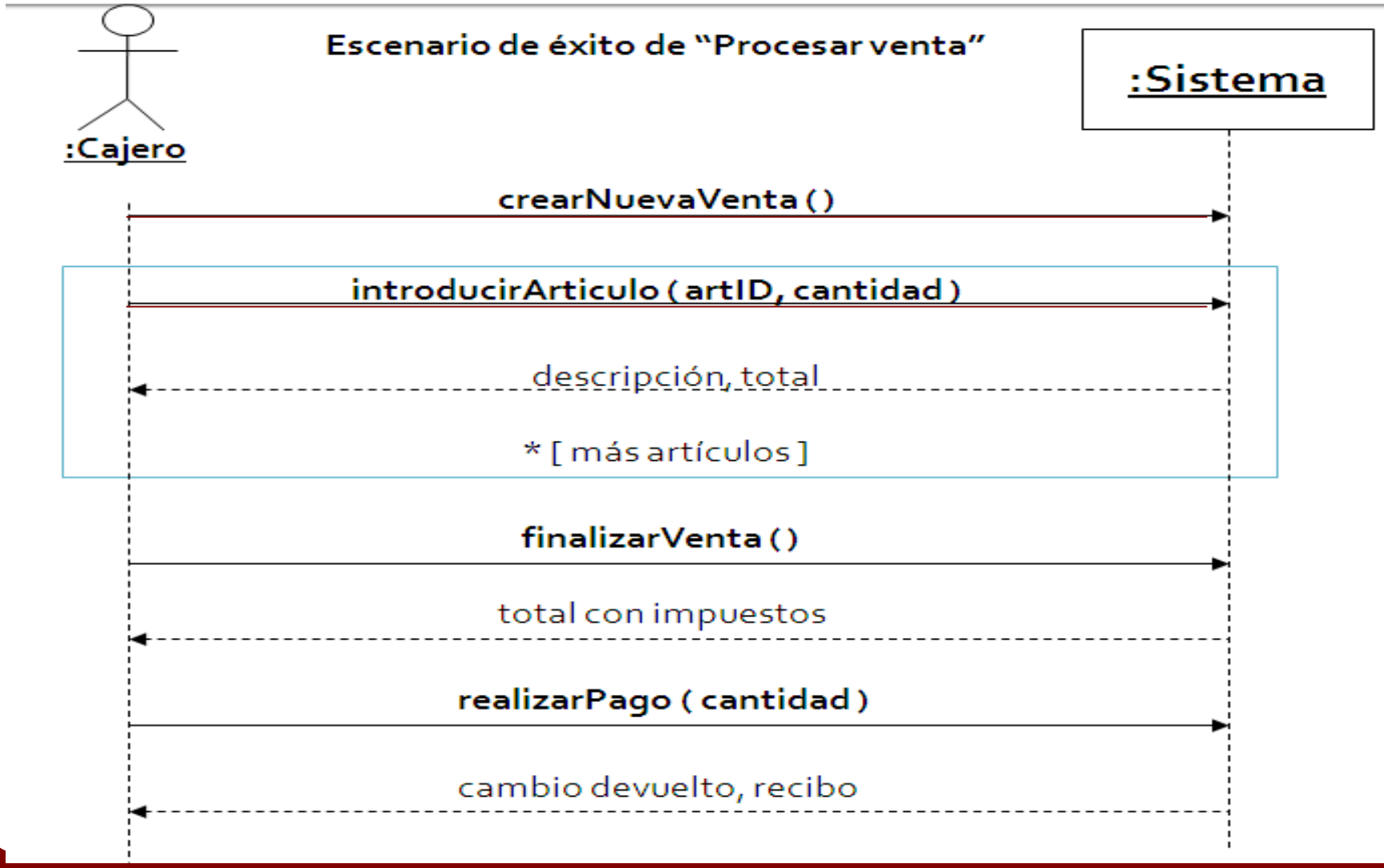
CdU: secciones

- Secciones.
  - Se utilizan sobre todo con CdU tipo CRUD (create, recover, update, delete).
  - A estos CdU se les suele llamar “Gestionar …” ó “CRUD …”
  - Nos podemos referir a ellas como CdUX.seccY
  - Si oscurecen el análisis es preferible no utilizarlas.

Diagramas de secuencia del sistema D.S.S.

- Definición
  - Definen el comportamiento del sistema como una “caja negra”.
  - Utilizan la misma notación de los diagramas de secuencia UML que utilizaremos en la etapa de diseño.
  - Es un dibujo que muestra la secuencia de acciones que realiza un actor sobre el sistema para un escenario concreto de un Caso de Uso.
  - Se realizan fundamentalmente en las fases de elaboración y construcción del UP. No se mencionan explícitamente en UP, ni en RUP

DSS (representación en UML)



Historias de usuario (1/2)

- Es otra forma de documentar requisitos, parecida a los casos de uso.
- Se podría decir que cada *HdU* documenta escenarios de un caso de uso de forma más declarativa, sin entrar en el detalle de un *CdU*
- Se escriben en una o dos frases y comienzan diciendo "Como <actor> quiero…"
- Son muy útiles porque van acompañadas de su prioridad, una estimación del esfuerzo, así como de una serie de pruebas que validarían la corrección y la consecución de dicha historia

Historias de usuario (2/2)