

Apuntes

DSI

Autor: Alejandro Rodríguez González

Email: jalo.javier@gmail.com

Tema 1. Fundamentos del DSI: Sistemas de información

1. Concepto de sistema

Un **sistema** es un conjunto de cosas que ordenadamente relacionadas entre si contribuyen a un objetivo, un **modelo** formado por una serie de elementos interrelacionados entre si, que opera en un **entorno** cambiante y con unos determinados objetivos.

Elementos:

Componentes del sistema, las **relaciones** entre los componentes determina su estructura. Los **objetivos** del sistema. Su **entorno** es aquello que lo rodea, y sus **límites** la frontera entre lo que es sistema y es entorno.

2. Sistema de información

Es un conjunto de **procesos** que operan sobre una colección de **datos** según las necesidades de la **organización** (*empresa*) y **recopilan, elaboran y distribuyen** la información necesaria para las **operaciones** de dicha empresa y para las actividades de **dirección y control** para desempeñar su actividad de acuerdo a su **estrategia de negocio**. El objetivo del *SI* es proporcionar **información de calidad**.

3. SI Automatizado (SIA)

Un SI no necesita estar basado en el uso de ordenadores para existir, existe siempre, mecanizado o no. La aplicación del ordenador a los *SI* produce los **Sistemas de información basados en computadora o Sistemas de información automatizados (SIA)**.

Algunos ejemplos de *SI* son:

- Sistemas ERP (*Enterprise Resource planning*)
- Sistemas CRM (*Customer relationship management*)
- Sistemas de flujo de trabajo (*Workflow management systems*)
- Sistemas de trabajo cooperativo (*Groupware systems*)
- Sistemas de comercio electrónico (*E-commerce Systems*)
- Sistemas EDI (*Electronic Data Interchange*)

4. La estrategia en el SI empresarial

El **SI empresarial** es el sistema encargado de coordinar los flujos y registros necesarios para llevar a cabo las funciones de una empresa de acuerdo con su planteamiento o estrategia de negocio (*algo fundamental*).

5. Elementos de un SI empresarial

- Las personas (roles: usuarios directos, indirectos, etc.)
- La información
- Los procedimientos y las prácticas habituales de trabajo.
- El equipo de soporte



6. Niveles de un SI empresarial

6.1 Sistema de transacciones

Este nivel incluye el procesamiento de las actividades **diarias**, los acontecimientos **rutinarios o periódicos** que afectan a la organización. Los **procedimientos** de tratamiento son fácilmente comprensibles y descriptibles en **detalle**. Aparecen **pocas excepciones a los procedimientos normales**, y existe un **gran volumen de transacciones** y una **gran similitud** entre estas.

6.2 Nivel operativo de dirección

Se realiza un análisis de los resultados para **tomar decisiones a corto plazo** y de consecuencias limitadas. Se preocupa del **análisis de los resultados**, sobre todo relacionado con las **transacciones** para tomar decisiones **a corto plazo y de consecuencias limitadas**. Se trabaja con información **repetitiva, centrada en el pasado**, con **datos de origen interno**, con **formato bien estructurado y detallado y preciso**.

6.3 Nivel táctico de dirección

Se realiza un análisis de resultados para toma de decisiones **a medio plazo**. Pretende **mejorar el rendimiento** basándose en **análisis de informes**: Resúmenes con **medidas estadísticas, de excepciones, específicos...**

6.4 Nivel estratégico de dirección

La información debe aparecer en formato **muy resumido (> 1 año < 5 años)**. Se puede encontrar en **formatos muy variables y de fuentes externas**. Se usa la información para decidir las **líneas maestras de la organización a largo plazo**.

7. Flujos de información

- **Verticales (Jerarquía):**
 - **Ascendentes:** De subordinado a superior.
 - **Descendente:** De jefe a subordinado.
- **Horizontales (Coordinación):** Mismo nivel de autoridad.
- **Formales e informales**

8. Tipos de SIE

Existen **otras estructuras organizativas**, otras formas de analizar una compañía sin atender a su estructura jerárquica de procesar la información (*por funciones o departamentos, división geográfica, etc.*).

8.1 Operating Information Systems (OIS)

Sistema de procesamiento de transacciones / *OLTP (O-line transactionnel Processing)*.

Es un entorno donde las respuestas se darán en un tiempo aceptable y serán consistentes. Es la parte del SI dedicada al procesamiento de operaciones rutinarias **diarias** (*transacciones*).

8.2 Management Information System (MIS – SI para la dirección)

Pertenece a los niveles operativo, **táctico** (*con este está más ligado*) y estratégico. Debe proporcionar información para **tomar decisiones, NORMALMENTE** (*que puede no ser siempre*) de **control administrativo**.

Proporciona soporte en las tareas bien estructuradas, con procesos **bien definidos**.

8.3 Decision Support System (DSS – Sistema de apoyo a la decisión)

Es un sistema informático **interactivo** que ayuda al usuario a **tomar decisiones** para resolver problemas **no estructurados**. Orientado a facilitar generalmente **decisiones estratégicas**.

9. Principales subsistemas

- **Subsistema de recursos humanos:** Gestión de información relacionada con la plantilla, ejecución de la nómina.
- **Subsistema de gestión comercial:** Ventas y función de comercialización.
- **Subsistema de gestión contable y financiera.**
- **Subsistema de control de almacén**
- **Otros subsistemas:** De automatización de oficinas y de producción.

Al construir un sistema ha de tenerse unos principios básicos:

Descomposición: Un sistema complejo es necesario descomponerlo en varios subsistemas para componerlo. La descomposición debe mantener el principio de **cohesión funcional**.

Simplificación: Organizar los subsistemas reduciendo el número de interconexiones.

Desacoplamiento: Lograr que los subsistemas sean lo más independientes posible.

Máxima cohesión, mínimo acoplamiento y poca cargabilidad.

10. Modelo de SIA

10.1 Los ERP

El ERP es un **marco de trabajo** con varios tipos de aplicaciones (*administrativas, de recursos y de planificación*). Los ERP **unifican e integran** los principales procesos de negocio.

El mundo donde se desarrollan los servicios necesarios para implantar un ERP es un mundo *complejo, dinámico, innovador, inestable y con reglas de negocio no tradicionales*.

Un 60% de los proyectos de implantación de ERP son fallidos.

10.2 Características de los ERP

Un ERP aporta un **marco estable, más portable, más adaptable** y ayuda a **consolidar y unificar** las funciones de negocio e **integra gran variedad de tecnologías y los procesos que conllevan**. Ayuda a **crecer**.

Sus objetivos **no son técnicos, son de negocio**:

- Mejorar funcionamiento de la empresa
- Mayor control sobre el negocio
- Soporte de toma de decisiones

Un ERP es facilitador de mejora empresarial, pero **no es** la respuesta en si mismo.

10.7 El futuro de las aplicaciones de ERP

El objetivo a largo plazo es **obtener más flexibilidad en las operaciones** pudiendo determinarse **cuatro objetivos** necesarios para conseguir esto.

1. **Componentes y no módulos**: Las aplicaciones deben funcionar independientes y poder ser re combinadas y ajustables a nuevos modelos de negocio.
2. **Migración incremental**: Productos rápido donde no haya que esperar largos periodos para completar el proyecto.
3. **Configuración de los ERP dinámicos más que estáticos**: Sistemas de configuración **dinámicos** para que **personal no técnico** pueda ajustar cambios de funcionalidad.
4. **Gestión de las relaciones de proveedores y socios**: Deben modelizar y supervisar los procesos que afecten a la actividad del negocio donde quiera que esto ocurra.

11. CRM: El cliente como centro de la empresa

CRM es un modelo estratégico y tecnológico cuyo objetivo es automatizar y mejorar los procesos de negocio asociados a la gestión de la relación de la empresa con el cliente. CRM es **maximizar la información disponible de nuestros clientes** para construir relaciones rentables y duraderas e incrementar el conocimiento que se tiene de cada cliente.

12. ¿Por qué DW y DM?

DW (*Data Warehouse*) es una colección de datos **orientados al tema, integrados, no volátiles e historizados, organizados** para el apoyo de un proceso de ayuda a la decisión. Se guarda toda la información útil en un único lugar.

Los **datos** deben formatearse y unificarse para llegar a un estado coherente (**datos integrados**), no se actualizan nunca y se referencian temporalmente (**datos historizados**).

DW es la base de *OLAP*, *DSS*, *DM*.

Las aplicaciones de **DM (Data Mining)** son: Marketing, finanzas, fabricación.

13. El servicio de informática

13.1 El comité de informática

Es el elemento con **mayor poder de decisión** entre los implicados por las actividades informáticas. Su **estrategia y planificación** se basa en aprobar planes estratégicos, desgloses de aplicaciones, valoración de aplicaciones y asignar responsables para realizar aplicaciones aprobadas.

Sus reuniones son **anuales, trimestrales o cuando el presidente lo considere oportuno**.

Tema 2. Fundamentos del DSI: Métodos y procesos para el desarrollo de SI

1. El papel de las metodologías

Todo proceso está integrado por **actividades** y estas por **tareas**. El método puede ser improvisado, personalizado, etc. El conjunto de métodos definidos para desarrollar un producto forma una **metodología**.

Las Metodologías nos describen (*quien (roles), que (método), cuando (proceso) y como (técnica)*).

Los métodos indican **como** construir el SW y abarcan un amplio espectro: **planificación, análisis, diseño, arquitectura, codificación, prueba y mantenimiento**.

Las **herramientas** suministran soporte automático para los métodos.

Los **procedimientos** permiten enlazar métodos y herramientas. Sirven para definir la **secuencia**, las **entregas**, los **controles** y las **directrices**.

Las **técnicas** son guías que ayudan al analista a completar las tareas de desarrollo del sistema.

2. Elementos y dimensiones de una metodología

Hay tres dimensiones:

- **Tecnología:** Conceptos, notación, técnicas y herramientas para el modelado.
- **Proceso:** Pasos a realizar y resultados obtenidos en cada paso.
- **Organización:** Como organizar las personas para acomodar el proceso.

2.1 Dimensión tecnológica

- **Conceptos:** Orientación a objetos y diseño estructurado.
- **Notación y técnicas:** Modelos requeridos, representación de modelos, expresividad y legibilidad de la notación.

2.2 Dimensión proceso

Pasos a realizar y resultados producidos en cada paso.

Los procesos son **iterativos** y los "entregables" evolucionan.

2.3 Dimensión organización

Necesario **enfoque industrial** para producir SW. Modelo para medir la capacidad de la organización: *CMMI*.

Además de notación, proceso, organización y herramientas existen:

- Guías de estimación de costes
- Guías de elaboración de entregables
- Métricas
- Políticas y procesos para asegurar calidad del SW
- Programas de formación
- Descripción de roles

3. Características y clases de Metodologías

Características deseables en toda metodología:

Existencia de reglas predefinidas, cobertura total del ciclo de desarrollo y **mantenimiento**, verificaciones intermedias, herramientas *CASE*, etc.

Según su orientación se pueden dividir en:

- **Orientación a los tratamientos/procesos:** Empiezan identificando las funciones a informatizar en la realidad.
- **Orientación a los resultados:** Identifican lo que piden los usuarios para identificar datos de entrada y tratamientos.
- **Orientación a datos:** Identifican entidades y sus relaciones deduciendo así los procesos y resultados.
- **Orientación a objetos:** Enfoque mixto de entidades (datos) y tratamientos (métodos).
- **Orientación a sucesos o eventos:** Identifica los sucesos que hacen que se disparen los tratamientos.

4. Metodologías estructuradas

4.1 Orientadas a procesos

Fundadas sobre el modelo básico: entrada/**proceso**/salida. Se enfocan al proceso y **generalmente** las especificaciones se basan en *DFDs*, diccionarios de datos y especificaciones de procesos.

Particularmente en la **metodología de Yourdon/Constantine** se basan en realizar los *DFD* del sistema, el diagrama de estructuras, evaluar el diseño y preparar este para la implantación.

4.2 Orientadas a datos

Modelo básico: entrada/proceso/salida. **Estructura de control jerárquica.**

5. Metodologías orientadas a objetos

Tratan **procesos y datos** de forma conjunta. En *OO* se examina el **dominio** del problema como un conjunto de objetos que interactúan entre sí. Varias generaciones:

- **Primera generación:** Revolucionarias o puras y Sintetistas o evolutivas
- **Segunda generación:** *RUP*, *AGIL*.

5.1 Metodología OMT (Doc)

5.2 Modelo Rational Unified Process (*RUP*)

Es un proceso de *SW* basado en uso de *UML*. Usa **patrones (templates)** y **herramientas** a lo largo de todo el ciclo de vida. Provee fácil acceso a una **base de conocimiento**.

5.2.1 Desarrollo iterativo

Permite un entendimiento **incremental** del problema a través de refinamientos. Habilita fácil **retroalimentación** de usuario con **metas específicas**. Los **requerimientos** se capturan y comunican mediante casos de uso los cuales son instrumentos importantes de **planificación**.

5.2.2 Estructura de *RUP*

El proceso se puede describir en dos dimensiones, o a lo largo de dos ejes:

Eje horizontal: Representa el **tiempo** y muestra el aspecto dinámico del proceso, expresado en términos de *ciclos, fases, iteraciones y metas*.

Eje vertical: Representa el aspecto estático del proceso, está constituido por **las disciplinas**.

5.3 Métrica V3

Cubre desarrollo estructurado y OO

6. Proceso estructurado vs. OO

7. Plan de implantación de una metodología.

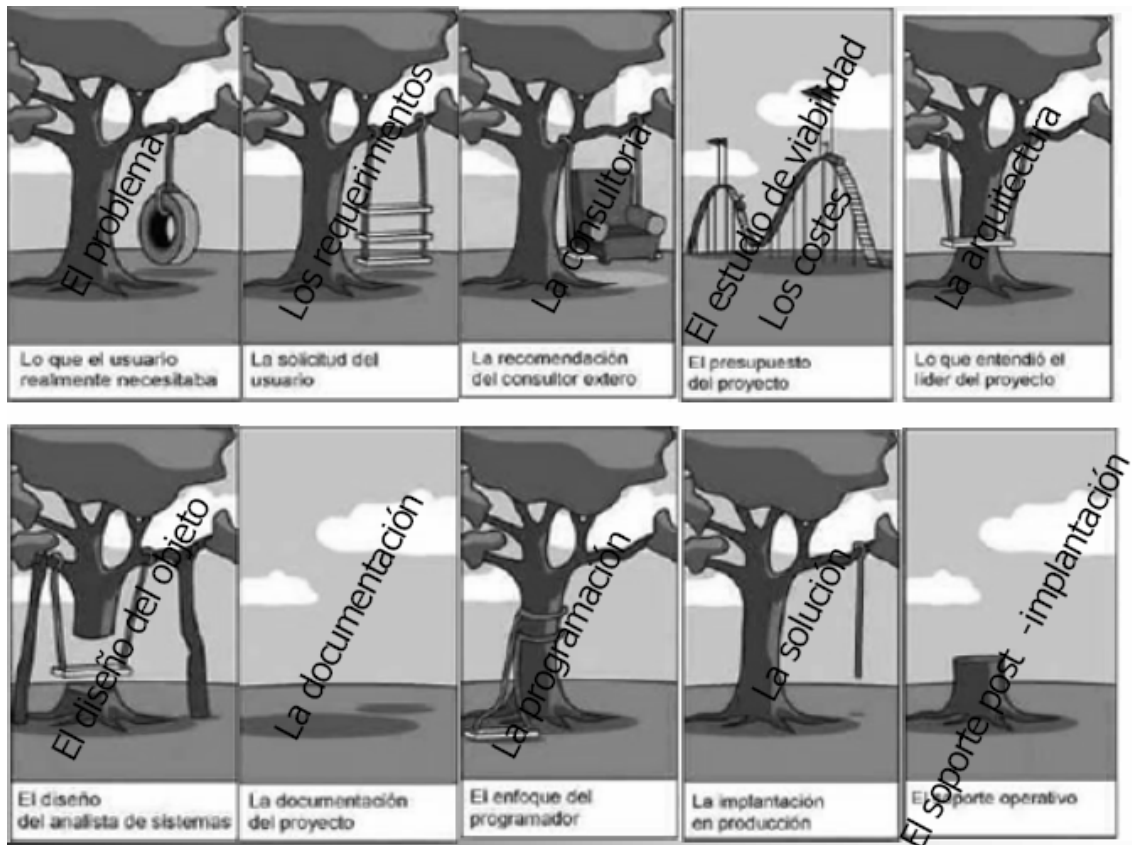
1. **Plan de normalización y adaptación al entorno:** Procedimientos, responsabilidades, documentos...
2. **Plan de formación a los miembros de la organización:** Seleccionar profesionales expertos.
3. **Plan de proyectos:** Seleccionar un proyecto piloto y evaluar y modificar la normalización, adaptación...
4. **Plan de seguimiento:** Definir, cuantificar y controlar el cumplimiento de los objetivos.

Tema 2b. Fundamentos del DSI: Métodos y procesos para el desarrollo de SI

1. El papel de las metodologías

Todo proceso productivo o de desarrollo está integrado por **actividades** y estas por **tareas**.

El método puede ser **improvisado**, **personalizado** o estar sujeto a diferentes niveles de homologación. Toda **tarea** se realiza mediante la aplicación de un **método**. La metodología ayuda a modelar el producto a desarrollar o producir y también a transformar el **modelo** en una **solución** y comprobar que la solución es un producto que responde a los requerimientos y también a mantenerlo por deficiencias o innovaciones.



El papel de las metodologías

Los métodos nos sirven para construir modelos, y estos para construir soluciones.

Metodología de desarrollo: Es un sistema.

Es un conjunto de elementos ordenadamente relacionados para conseguir un fin que nos sirven para especificar como dividir un proyecto en etapas, que actividades o tareas debemos realizar en cada etapa, artefactos que generan, entradas salidas y secuencias de las tareas, restricciones, etc.

La metodología nos describe quien (*roles*), que (*método*), cuando (*proceso*) y como (*técnica*). Hay unos pasos y procedimientos que deben seguirse para el desarrollo de software:

- Como se debe dividir un proyecto en **etapas**.
- Que **tareas** se llevan a cabo en cada etapa.
- **Heurísticas** para llevar a cabo dichas tareas.
- Que **salidas** se producen y cuando se deben producir.
- Que **restricciones** se aplican.
- Que **herramientas** se van a usar.
- Como se **gestiona y controla un proyecto**.

Los métodos:

Indican **como** construir el software y abarcan un amplio espectro: **planificación, análisis, diseño, arquitectura, codificación, prueba y mantenimiento**.

Las herramientas:

Suministran soporte automático para los métodos. Las hay para cada método mencionado. Herramientas *CASE*.

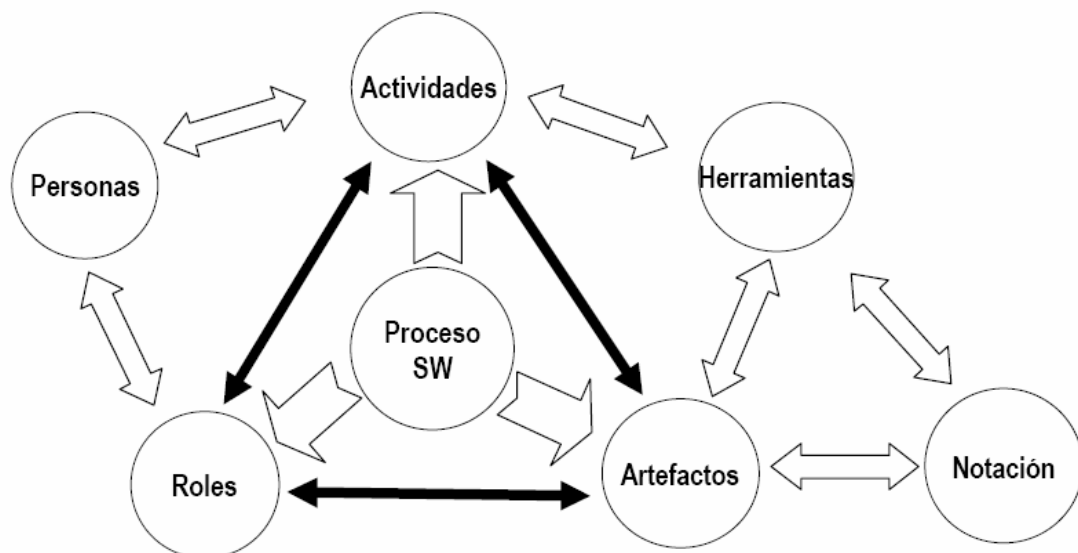
Los procedimientos:

Permiten y facilitan el enlazar los métodos y las herramientas, favoreciendo un desarrollo racional y temporal del software. Sirven para definir la **secuencia** de aplicación de los métodos, las **entregas** que requieren y los **controles** que ayudan a asegurar la calidad y coordinar cambios y las **directrices** que ayudan a los gestores del *SW* a evaluar el progreso.

Las técnicas:

Son las guías que ayudan al analista a completar las tareas de desarrollo del sistema.

2. Elementos y dimensiones de una metodología



Elementos y dimensiones de una metodología

Dimensión tecnológica:

Conceptos (OO, Diseño estructurado), Notación y técnicas (*Modelos requeridos y su representación*), Herramientas usadas para el modelado.

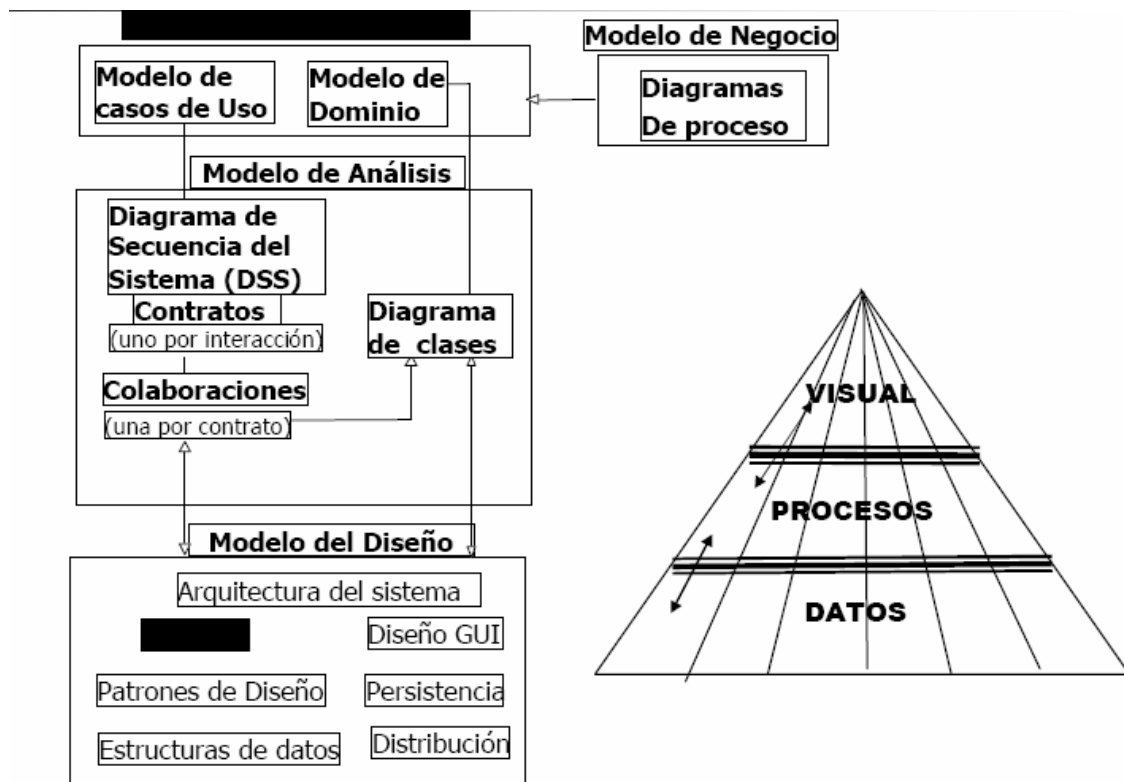
Dimensión proceso:

Conjunto de pasos a realizar y resultados obtenidos en cada paso (*entregables*). Pueden tener diferentes escalas de tiempo (*creación de modelos, desarrollo de sistemas, creación de componentes reutilizables*). Elementos del proceso:

- Secuencia de etapas
- Entradas y salidas de cada etapa
- Roles implicados
- Interacción entre las etapas
- Heurísticas
- Mecanismos para seguimiento, validación y verificación

Los procesos son **iterativos** y los entregables evolucionan. Conviene centrarse en aspectos **críticos** en las primeras iteraciones para minimizar riesgos.

Proceso RUP:



Ejemplo Proceso RUP

Además de **notación, proceso, organización y herramientas** hay guías de **estimación de costes**, para la elaboración de **entregables, métricas**, políticas y procesos de aseguramiento de **calidad**, programas de **formación**, descripción de **roles**.

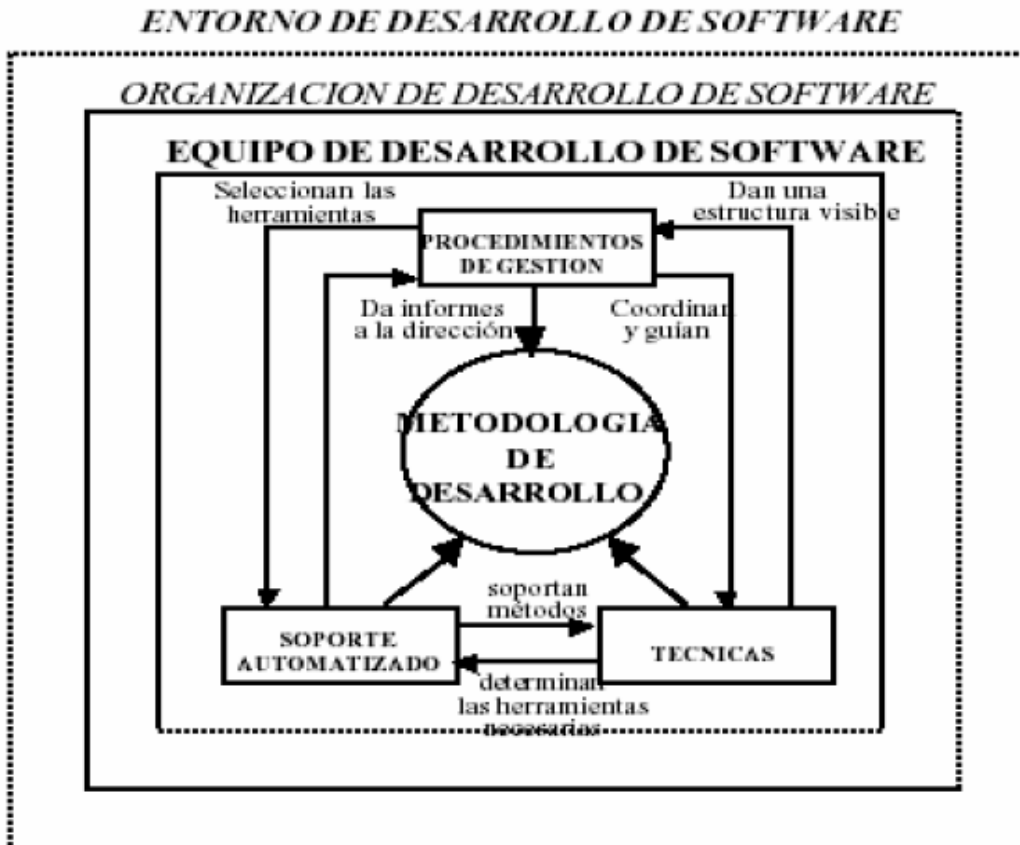
Dimensión organización: Como organizar las personas para acomodar el proceso.

3. Características y clases de metodologías

Las características **deseables** son:

- Existencia de reglas predefinidas
- Cobertura total del ciclo de desarrollo y mantenimiento
- Verificaciones intermedias
- Planificación y control
- Comunicación efectiva

- Utilización sobre un abanico amplio de proyectos
- Fácil formación
- Herramientas CASE
- Actividades para mejorar proceso de desarrollo
- Soporte al mantenimiento
- Soporte de reutilización de software



- **Orientación a tratamientos o procesos:** Empieza por identificar en la realidad las funciones a informatizar.
- **Orientación a los resultados:** Empieza por identificar lo que piden los usuarios para deducir datos de entrada y tratamientos. Se suele usar de forma implícita por los que se niegan a usar métodos, programadores de microinformática y usuarios de 4gl.
- **Orientación a datos:** Empieza por identificar entidades y sus relaciones, deduciendo los procesos que los emplean y los resultados obtenibles.
- **Orientación a objetos:** Enfoque mixto de entidades (*datos*) y tratamientos (*métodos*).
- **Orientación a sucesos o eventos:** Empieza por identificar los sucesos que hacen que se disparen los tratamientos, los cuales se transmiten por mensajes a los objetos correspondientes.

4. Metodologías estructuradas

4.1 Metodologías orientadas a procesos



Metodologías orientadas a procesos

Se funda en el modelo básico: entrada/**proceso**/salida. Enfoca la parte del proceso.

Generalmente las especificaciones se basan en:

- Diagrama de flujo de datos
- Diccionario de datos
- Especificaciones de procesos

Particularmente, en la metodología de Yourdon/Constantine:

- Realizar DFD del sistema
- Realizar diagrama de estructuras
- Evaluar el diseño
- Preparar el diseño para implantación

4.2 Metodologías orientadas a objetos

Datos jerárquicos:

Fundadas en el modelo básico **entrada/proceso/salida**. La **estructura de control** del programa debe ser jerárquica y se debe derivar de la **estructura de datos**. El proceso de diseño consiste en:

- Definir **primero** las estructuras de datos de E/S.
- Mezclarlas todas en una **estructura jerárquica de programa**.
- Ordenar detalladamente la **lógica procedimental** para que se ajuste a esta estructura.
- El diseño lógico debe preceder y estar separado del físico.

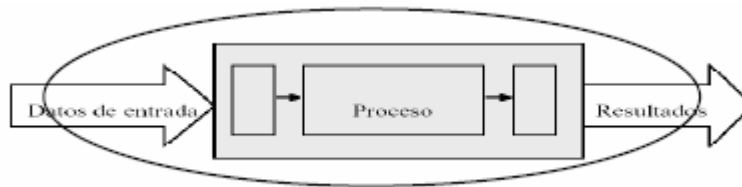
Datos NO jerárquicos:

Los datos son el corazón del SI, que son la parte esencial del sistema porque son **más estables** que los procesos que actúan sobre ellos. Los procesos derivan de una definición inicial de los datos. El modelo está formado por el conjunto de **entidades** básicas y las **interrelaciones** entre ellas.

Cuatro etapas:

1. **Planificación:** Construir una **arquitectura** de la información y una **estrategia** que soporte los **objetivos** de la organización.
2. **Análisis:** Comprender las áreas del negocio y determinar requisitos del sistema.
3. **Diseño:** Establecer el comportamiento del sistema deseado.
4. **Construcción:** Construir sistemas que cumplan los tres niveles anteriores.

5. Metodologías orientadas a objetos



Metodologías orientadas a objetos

Trata **procesos y datos** conjuntamente. En la OO se examina el **dominio** del problema como un conjunto de objetos que interactúan entre sí. Abstracción, ocultación de información y **modularidad**. Las técnicas estructuradas influyen en estas metodologías. En la OO se propugna un enfoque unificador.

Primera generación:

- **Revolucionarias o puras:** Rompen con las metodologías estructuradas *OOD* de Booch, CRC/RDD de Wirfs y Brock.
- **Sintetistas o evolutivas:** Las metodologías estructuradas son la base para el desarrollo *OO OMT* de Rumbaugh, *SYNTHESIS* de Bailin.

Segunda generación:

- Método unificado (RUP)
- Ágil

Antes de UML:

- **Métodos dirigidos por los datos:** *OMT, FUSION*
- **Métodos dirigidos por responsabilidades:** *RDD, OBA*
- **Métodos dirigidos por casos de uso:** *OOSE/Objectory*
- **Métodos dirigidos por estados:** Shlaer y Mellor

Después de UML:

- **Evolución métodos clásicos:** Métrica V3
- **Métodos ágiles:** *XP, Crystal Methods, AUP, SCRUM, DSDM, RAD.*
- **Métodos marco adaptables:** *OPEN, RUP, OUP*

5.1 Metodología OMT

Esta metodología propone usar las técnicas de modelo de objetos, dinámico y funcional. Parte de la idea de usar los mismos conceptos y notación a lo largo de todo el ciclo de vida.

Modelo de Objetos (*M de O*):

Describe la estructura de los objetos de un sistema (**identidad, relaciones con otros objetos, atributos y operaciones**). El objetivo al construir un M de O es capturar aquellos conceptos del mundo real que sean importantes para una aplicación. Se representa gráficamente mediante **diagramas** que contienen **clases** de objetos. Las **acciones** de los diagramas de estado se corresponden con **funciones** procedentes del modelo funcional; los **sucesos** de un *DE* pasan a ser las **operaciones** que se aplican a objetos dentro del modelo de objetos.

Modelo dinámico (*MD*):

Describe aquellos aspectos del sistema que tratan la **temporización y secuencia** de las **operaciones** y la organización de **sucesos y estados**. El *MD* captura el **control**, aquel aspecto de un sistema que describe las secuencias de operaciones que se producen. Se representa gráficamente mediante los **diagramas de estado**. Cada uno de estos diagramas muestra el estado y las secuencias de sucesos para una clase de objetos.

Modelo funcional (MF):

Describe aquellos aspectos del sistema que tratan las **transformaciones de valores**. El *MF* captura **lo que hace** el sistema, independientemente de cuando se haga o la forma. Se representa mediante el **diagrama de flujo de datos** que muestran las dependencias entre los valores y el cálculo de valores de salida a partir de los de entrada y de funciones. Las **funciones** se invocan como **acciones** en el modelo dinámico y se muestran como **operaciones** que afectan a objetos en el modelo de objetos.

Relaciones entre los modelos:

El **modelo de objetos describe la estructura de datos** sobre el que operan los modelos **dinámico y funcional**. Las **operaciones** del modelo de objetos se corresponden con **sucesos** en el modelo dinámico y con **funciones** en el modelo funcional. El **modelo dinámico** describe la **estructura de control** de los objetos, muestra decisiones que dependen del valor de los mismos. El **modelo funcional** describe **funciones** pedidas por operaciones en el modelo de objetos y acciones en el modelo dinámico.

5.2 Modelo RUP

Es un proceso de software basado en *UML*. Usa **herramientas y patrones** a lo largo de todo el ciclo de vida. Provee a cada miembro del equipo un fácil acceso a una **base de conocimiento** con guías, plantillas, herramientas, etc. **Crea y mantiene modelos** en lugar de enfocarse la producción de una gran cantidad de papeles de documentación.

Dados los **sistemas de software sofisticados** de la actualidad, no es posible hacer de manera secuencial la definición y el diseño de la solución completa. El descubrimiento de defectos en fases posteriores de diseño da como resultado un aumento en los costos y/o cancelación del proyecto.

El desarrollo iterativo:

Permite un entendimiento **incremental** del problema a través de refinamientos. Habilita **retroalimentación**. Las **metas específicas** permiten que el equipo de desarrollo mantenga su atención en producir resultados. El **progreso** es medido según avanzan las implementaciones.

Los casos de uso dirigen todo el proceso:

Licitación, organizar y documentar la **funcionalidad** y restricciones. Los **requerimientos** se capturan y comunican mediante casos de uso, los cuales son instrumentos importantes de **planificación**.



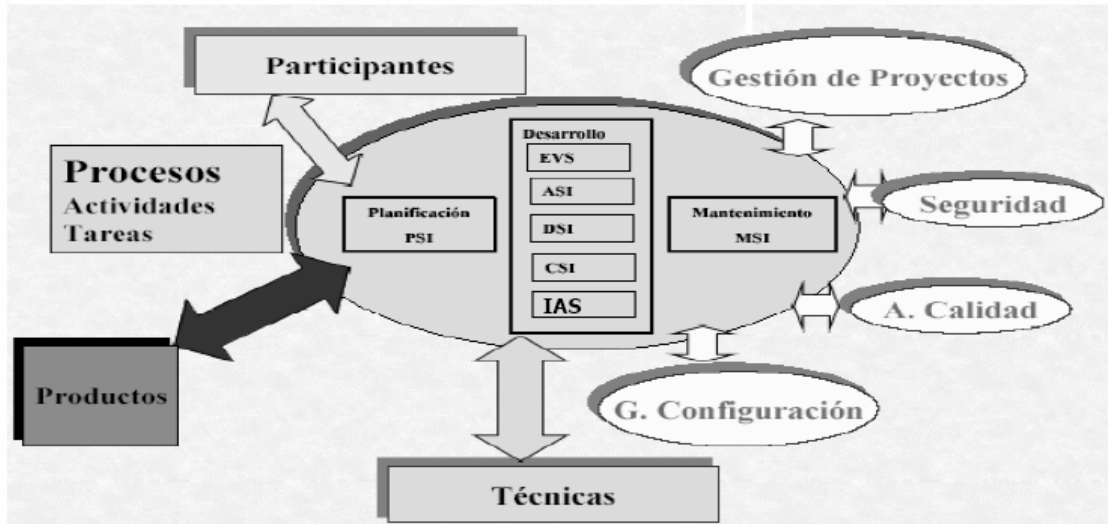
Modelo RUP

Estructura de RUP:

Iteraciones: Cada fase de *RUP* se puede descomponer en iteraciones. Una iteración es un ciclo de desarrollo completo dando como resultado una entrega de producto ejecutable (*interna o externa*).

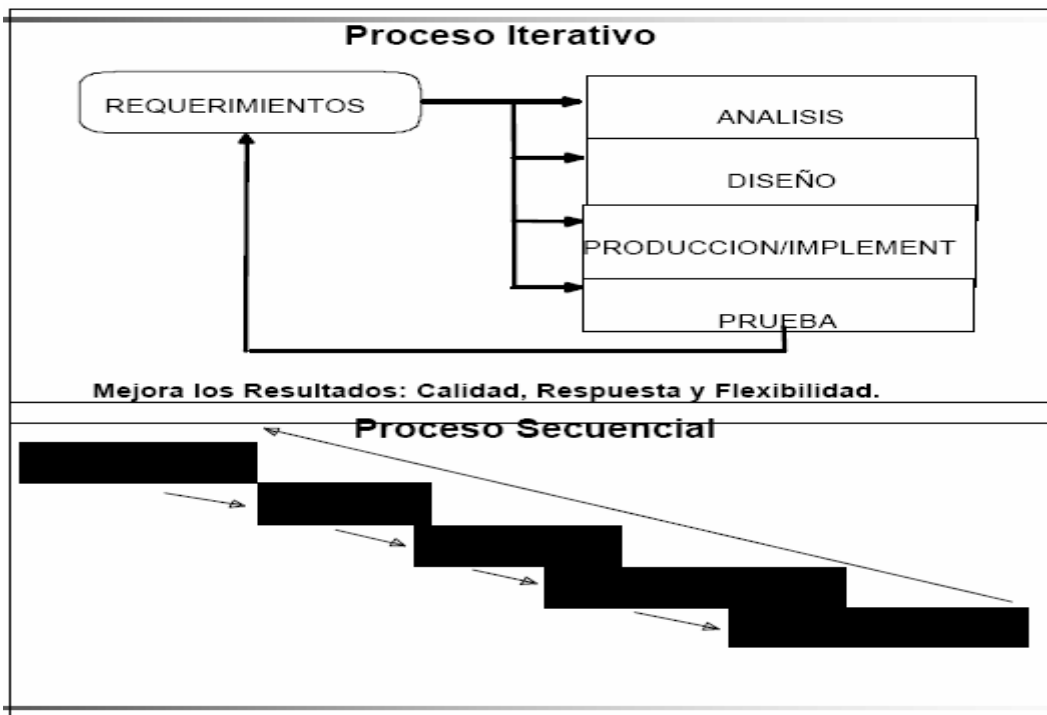
Métrica V3:

Cubre desarrollo estructurado y OO. Facilita la realización de los procesos de apoyo u organizativos.

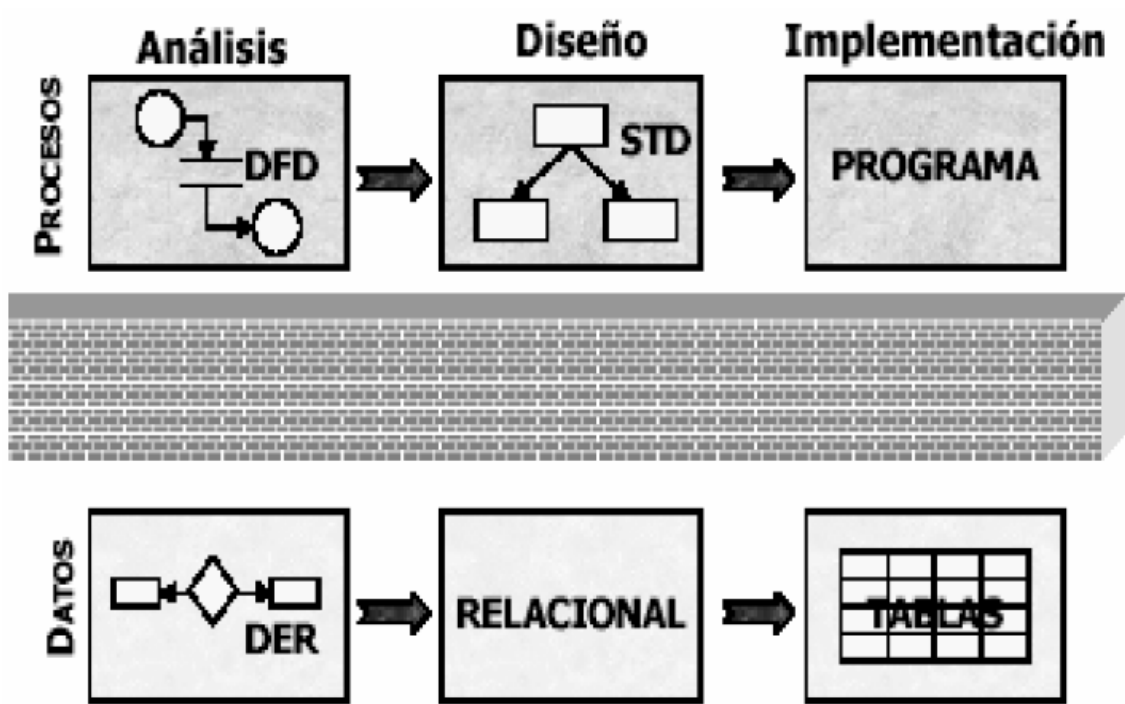


Métrica V3

6. Proceso estructurado Vs OO



Proceso iterativo y secuencial



Proceso estructurado Vs OO

Tema 3. Lenguaje unificado de modelado (*UML*)

1. Introducción

UML combina notaciones de los modelados OO, de datos, componentes y workflows.

2. El metamodelo UML

Existen ciertos mecanismos de extensión en UML:

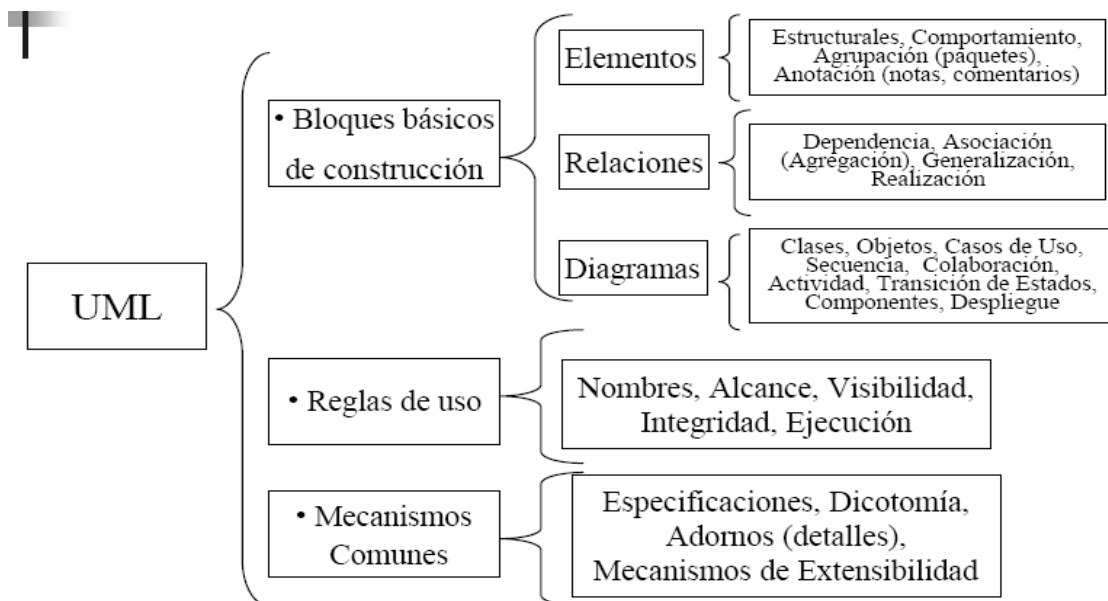
- Stereotypes
- Constraints
- Tagged Values

Alguno de los **inconvenientes** de UML es que **falta integración** respecto a **otras técnicas**, pero **facilita** la *estandarización, validación, prototipos, etc.*

UML permite conectarse a lenguajes de programación (*ingeniería directa e inversa*) y documentar todos los artefactos de un proceso de desarrollo.

El metamodelo define la notación a través de diagramas de la propia notación y con OCL (*Object Constraint Language*).

Un **metamodelo** describe de manera formal los elementos de modelado y la sintaxis y semántica de la notación para manipularlos. En el **metamodelado** la idea fundamental es que las entidades del modelo (clases) jueguen el papel de plantilla (visto como una clase) y el de instancia (*vista como un objeto*).



Organización UML

3. Modelos y diagramas

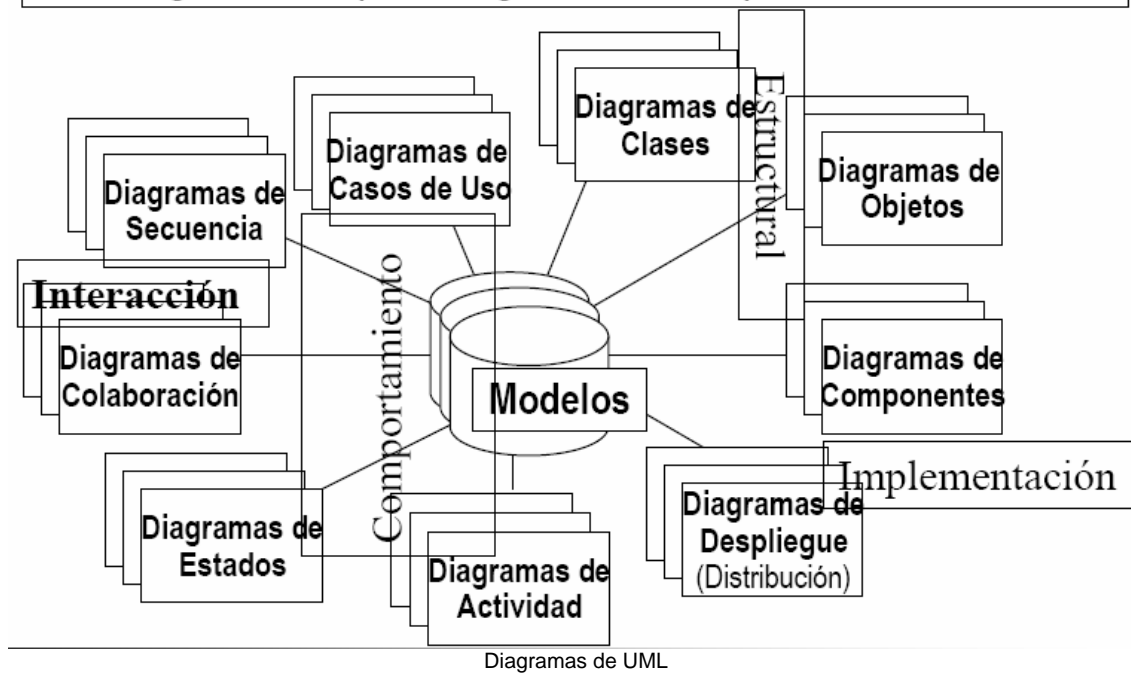
Un **modelo** captura una vista de un sistema del mundo real. Es una abstracción del sistema y describe completamente los aspectos del sistema relevantes al propósito del modelo.

Un **diagrama** es una representación gráfica de una colección de elementos de modelado.

El **código fuente** del sistema es el modelo más detallado del sistema (*y ejecutable*).

4. Diagramas de UML

Los diagramas expresan gráficamente partes de un modelo

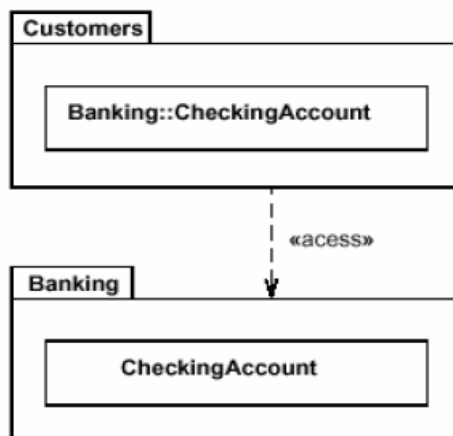


5. Paquetes

Los paquetes ofrecen un mecanismo para la organización de modelos/subsistemas agrupando elementos. Cada paquete corresponde a un **submodelo** (*subsistema*) del modelo (*sistema*).

Un paquete **puede contener otros paquetes, sin límite de anidamiento** pero cada elemento pertenece a sólo un paquete.

Una clase de un paquete puede aparecer en otro paquete por importancia. El paquete a la vez que agrupa puede **encapsular**. El operador "::" permite designar una clase definida en un contexto distinto al actual.



Acceso a clase de paquete externo

6. Diagramas de casos de uso

Un **actor** representa una persona u otro sistema que interactúa con el sistema. Un **caso de uso** define una secuencia de acciones que realiza un sistema dando como resultado un valor observable para un actor. Los actores **no son parte del sistema**.

Casos de uso es una **técnica para capturar información**.

7. Diagramas de secuencia

Representa una interacción entre los objetos de un caso de uso, escenario.

8. Diagrama de colaboración

Es una interacción entre los objetos de un caso de uso o escenario. Papel que juega un objeto en el sistema.

9. Diagrama de clases

Una clase es un descriptor de un conjunto de objetos.

Un **objeto es una entidad discreta, con identidad, estado y comportamiento invocable**.

Una clase tiene una **multiplicidad** que especifica cuantas instancias de ella pueden existir.

9.1 Interfaces

Una interfaz es la descripción del comportamiento de objetos **sin proporcionar su implementación** o estado. Las interfaces no poseen implementación y contienen operaciones pero no atributos. Equivale a **una clase abstracta sin atributos ni métodos**. Una interfaz, aparte de con `<<interface>>` se puede representar con **un circulito con el nombre de la interfaz situado debajo del símbolo**.

9.2 Diagrama de clases

Es el diagrama principal para el análisis y el diseño.

10. Diagrama de estados

Un Diagrama de Estados muestra la secuencia de estados por los que pasa bien un caso de uso, bien un objeto a lo largo de su vida, o bien todo el sistema. En él se indican qué eventos hacen que se pase de un estado a otro y cuáles son las respuestas y acciones que genera.

En cuanto a la representación, un diagrama de estados es un grafo cuyos nodos son estados y cuyos arcos dirigidos son transiciones etiquetadas con los nombres de los eventos.

11. Diagrama de actividad

El Diagrama de Actividad es un diagrama de flujo del proceso multi-propósito que se usa para modelar el comportamiento del sistema. Los diagramas de actividad se pueden usar para modelar un Caso de Uso, o una clase, o un método complicado.

Un diagrama de actividad es parecido a un diagrama de flujo; la diferencia clave es que los diagramas de actividad pueden mostrar procesado paralelo (*parallel processing*).

12. Diagrama de componentes

Se muestra la estructura del software y la dependencia entre componentes. Un componente es un grupo de clases cohesionadas.

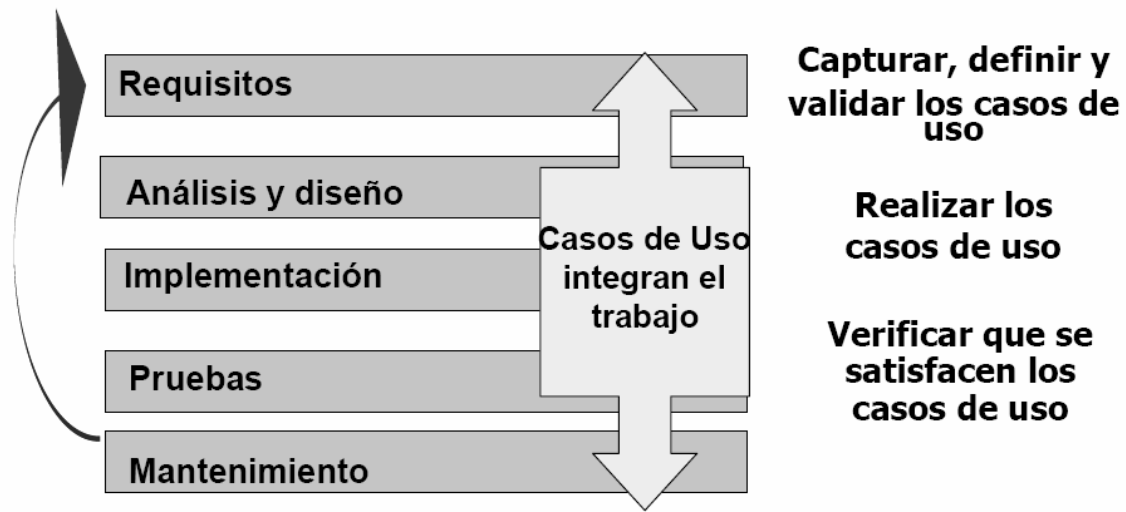
13. Diagrama de despliegue

Es un diagrama de distribución de los componentes *SW* en los nodos y su comunicación. Cada nodo puede contener instancias de componentes.

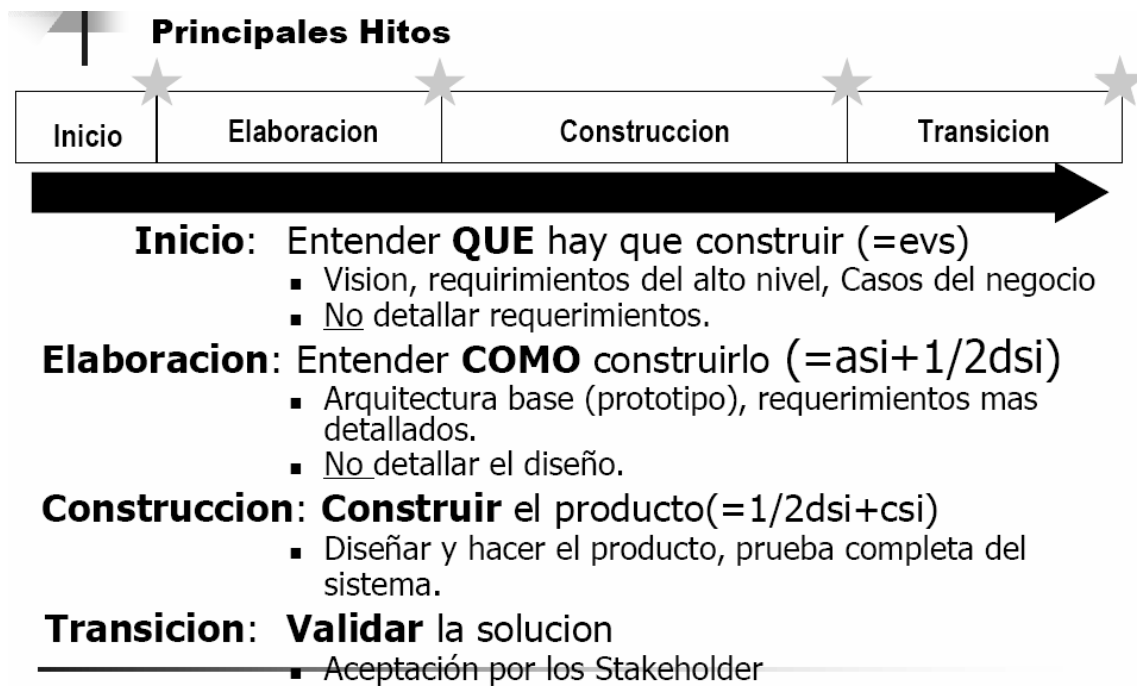
Tema 5. Metodología RUP

1. Introducción a RUP

RUP es una **metodología** que aplica las mejores prácticas para el desarrollo de software. Un **proceso de ingeniería de software** bien definido y estructurado. Se caracteriza por ser **dirigida por los casos de uso, iterativos e incrementales y estar centrada en la arquitectura**. Se suele aplicar a las **mejores prácticas**.



Metodología RUP dirigida por CDU



Fases del desarrollo iterativo

Iteraciones y fases:

Cada fase se refiere a la actividad dominante. Una iteración es una secuencia de **actividades diferentes con un plan** y unos criterios de evaluación establecidos, que dan como resultado una reléase ejecutable.

Ciclo de vida iterativo:

Se basa en la evolución de prototipos ejecutables que se muestran a los usuarios y clientes. En el ciclo de vida iterativo a cada iteración se reproduce el ciclo de vida en cascada a

menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

Fase Inicio: Saber **QUE** hay que construir.

- Preparar el documento Visión y el de casos del negocio.
- Desarrollar **requerimientos de alto nivel** del proyecto.
- Gestionar **alcance del proyecto**

Fase Elaboración: Saber **COMO** construirlo

- **Detallar los requerimientos** lo necesario (~80% completo)
- **Producir una arquitectura** ejecutable y estable
- Arquitectura dirigida por casos de uso clave.
- **Verificar calidad de la arquitectura.**
- **Evaluar continuamente** casos del negocio, riesgos y plan de desarrollo.

Fase Construcción: Construir el modelo

- **Completar requerimientos y modelo de diseño.**
- Diseñar, implementar y probar cada componente.
- Construcción diaria o semanal con procesos automáticos.
- **Probar** cada construcción.
- **Liberar software** con funcionalidad completa.
- Producir descripciones de reléase.

Fase Transición: Distribuir a usuarios finales

- Producir releases incrementales.
- Actualizar manuales de usuario y documentación.
- Actualizar descripciones de reléase.
- Execute cut-over (transición ordenada y sincronizada).
- Conduct post-mortem project analysis

Proceso centrado en la arquitectura:

La arquitectura de un sistema es la organización o **estructura** de sus **partes** más relevantes. Una arquitectura ejecutable es una implementación parcial del sistema, construida para demostrar algunas funciones y propiedades. RUP establece refinamientos sucesivos de una arquitectura ejecutable construida como un prototipo evolutivo.

Mejores prácticas y principios clave:

- Desarrollar solo **lo necesario**.
- **Minimizar documentos** de trabajo.
- Ser **flexible**.
- Aprender pronto de los **errores**.
- Revisar los **riesgos**.
- Establecer **objetivos** medibles para ver el progreso.
- **Automatizar**.

Noción de proceso:

¿Qué es un proceso de desarrollo de SW?

Define quien debe hacer qué, cuando y cómo debe hacerlo.



No existe un proceso de software universal. Las características de cada proyecto exigen que el proceso sea configurable.

Rol de UML en RUP

RUP se desarrolló mano a mano con UML. La mayoría de los artefactos en RUP tienen una representación UML. RUP también incluye una guía de los conceptos UML.

2. Las seis mejores prácticas de RUP

RUP describe como utilizar de forma efectiva los **procedimientos** probados en el desarrollo de software por equipos de desarrollo software.

2.1 Administración de requerimientos

Aspectos de la gestión de los requerimientos

- **Analizar** el problema (*stakeholders y fronteras*).
- **Comprender** las necesidades del usuario.
- **Definir** el sistema.
- **Gestionar** el alcance.
- **Refinar** la definición.
- **Gestionar** el cambio de requerimientos.

Licitar, organizar, y documentar la funcionalidad y restricciones requeridas. Llevar un registro y documentación de cambios y decisiones.

¿Que especifican los requerimientos?

La gestión de requerimientos **NO ES FACIL**, porque los requerimientos no son siempre obvios, proceden de varias fuentes, no siempre son fáciles de expresar con palabras, relacionan unos productos con otros en el proceso de ingeniería del software, tiene propiedades o valores únicos, cambian y son difíciles de controlar.

2.2 Desarrollo iterativo de software

Dados los sistemas de software sofisticados de la actualidad, **no** es posible hacer de forma **secuencial** la definición completa del problema, diseñarla, construir el software y por último probarlo.

El descubrimiento de defectos en fases posteriores de diseño da como resultado un **aumento en los costos y/o la cancelación del proyecto.**

Características del desarrollo iterativo:

- Técnica usada para obtener producto mediante **releases**, cada una generada en una iteración.

- Cada iteración se centra en la definición, análisis, diseño, construcción y prueba de un conjunto de requerimientos.
- Permite un **entendimiento incremental** del problema a través de refinamientos.
- Habilita una fácil **retroalimentación** de usuario.
- Establecer **metas específicas** permite al equipo de desarrollo mantener su atención en producir resultados.
- El **progreso es medido** conforme avanzan las implementaciones.

2.3 Arquitectura basada en componentes

Se enfoca en el desarrollo temprano de una arquitectura ejecutable robusta. Es derivada a partir de los **CDU** más importantes.

Arquitecturas elásticas y basadas en componentes:

Elásticas:

- Permiten hacer frente a los requerimientos actuales y futuros.
- Mejoran la extensibilidad.
- Permiten la reusabilidad.
- Encapsulan las dependencias del sistema.

Basada en componentes:

- Componentes reusables o personalizables.
- Seleccionados de la oferta comercial.
- El software existente evoluciona de forma incremental.
- Evita los efectos en cascada de los cambios.

2.4 Modelado visual de software

Captura la estructura y comportamiento de arquitecturas y componentes. Además muestra como encajan de forma conjunta los elementos del sistema y mantiene la consistencia entre un diseño y su implementación promoviendo una comunicación no ambigua.

2.5 Verificación continua de la calidad del software

Crea **pruebas para cada escenario** para asegurar que todos los requerimientos están propiamente implementados y además **verifica la calidad** del software con respecto a los requerimientos basados en la confiabilidad, funcionalidad, desempeño de la aplicación y del sistema.

2.6 Control de cambios del software (*el mantenimiento en la construcción*)

Controlar, llevar un registro y monitorizar cambios para permitir un desarrollo iterativo. Establece espacios de trabajo seguros para cada desarrollador y provee el aislamiento de los cambios hechos en otros espacios de trabajo. Además controla todos los artefactos de software, modelos, código, documentos, etc.

Aspectos de la gestión de cambios:

- **Change Request Management (CRM):** Organización requerida para asegurar el coste y el impacto de los cambios pedidos.

- **Configuration Status Reporting:** Descripción del estado del producto y su ciclo de vida.
- **Configuration Management (CM):** Descripción de la estructura del producto y sus versiones.
- **Change Tracking:** Realización detallada del CRM.
- **Versión Selection:** Asegura la implementación de CM.
- **Software Manufacture:** Automatización del proceso de compilación, prueba, empaquetamiento y distribución.

3. Estructura de RUP

El proceso puede describirse en dos dimensiones o a lo largo de dos ejes.

Horizontal: Representa tiempo y muestra el aspecto dinámico del proceso, expresado en términos de ciclos, fases, iteraciones y metas. Un **ciclo** se compone por fases. Una **fase** por iteraciones. Y una **iteración** contiene metas.

Vertical: Representa el aspecto estático del proceso: **Las disciplinas** descritas en términos de actividades, artefactos, trabajadores y flujos de trabajo.

Iteraciones:

Una iteración es como un mini-proyecto. Cada fase en RUP es al menos una iteración. Es una secuencia de diferentes actividades contenidas en un plan base y sometidas a criterios de valoración, que tienen como resultado final una reléase interna o externa.

Componentes clave de las nueve disciplinas de RUP:

- **Roles:** Un rol define la conducta y responsabilidad de un individuo o grupo de responsables de actividades y artefactos.
- **Actividades:** Una actividad es una tarea de la cual se responsabiliza un rol y describe los pasos requeridos para crear o modificar uno o varios artefactos.
- **Artefactos:** Un artefacto es una entrada y/o salida de una actividad.

4. Fases en RUP (*visión temporal*)

- **Inicio:** Define el alcance del proyecto.
- **Elaboración:** Plan del proyecto, especificación de características (*features*), arquitectura base (*baseline*).
- **Construcción:** Construir el producto.
- **Transición:** Transición del producto a la comunidad del usuario.

Fase de Inicio:

Propósito:

- Establecer **CDU de negocio** para un nuevo sistema o actualización importante de un sistema existente.
- Especificar el **alcance** del proyecto.
- Estimar el **coste y el plan**.
- Identificar **riesgos** potenciales.
- Preparar el **entorno de soporte** del proyecto.

Resultado:

- Visión general de los **requerimientos** del proyecto.

- Casos de uso del negocio inicial incluyendo evaluación de **riesgos** y estimación de **recursos**.

Fase de elaboración:

Propósito:

- Analizar el **dominio** del problema.
- Establecer una buena **arquitectura**.
- Lidar con los elementos de **riesgos más altos** del proyecto.
- Desarrollar un **plan** comprensivo mostrando como el proyecto será completado.

Resultado:

- Un **modelo del dominio** y de CDU 80% completo.
- **Requerimientos** suplementarios que capturen los requerimientos no funcionales y los que no estén asociados a un CDU específico.
- Una lista de **riesgos** revisada.

Fase de construcción:

Propósito:

- **Desarrollar incrementalmente** producto de SW completo el cual estará listo para ser transferido al usuario.
- Minimizar el coste de desarrollo.
- Asegurar la **calidad**.

Productos:

- Un **modelo completo de diseño** y CDU.
- **Liberaciones de productos ejecutables de funcionalidad incremental**.
- **Documentación** de usuario.
- Una **liberación beta** del producto.

Fase de transición:

Propósito:

Hacer la transición final del producto de SW al usuario.

Productos:

- **Liberaciones ejecutables** de producto.
- **Pruebas beta** para validar el nuevo sistema vs. las expectativas del usuario.
- Manuales de usuario actualizados.
- **Documentación** de desarrollo actualizada.

5. Workflow en RUP

Iteraciones (*Mini proyectos*):

Cada fase en RUP se puede descomponer en iteraciones. Una iteración es un ciclo de desarrollo completo dando como resultado una entrega de producto ejecutable (*interna o externa*).

Modelos y flujos de trabajo:

Una mera enumeración de todos los trabajadores, actividades y artefactos no constituyen un proceso. Se necesita una forma de **describir secuencias** significativas que produzcan algún resultado válido, y que muestre la interacción entre trabajadores.

Un **flujo de trabajo** es una secuencia de actividades que producen un resultado de valor observable. En términos de UML pueden ser expresados como un **diagrama de secuencia, de colaboración o de actividad**.

6. Elementos en RUP

6.1 Roles (*Who?*)

Los **roles** realizan actividades que tienen artefactos de entrada y salida. Define el comportamiento y las responsabilidades de un individuo, o de un conjunto que trabaja en equipo. Los miembros de un equipo pueden vestir diferentes sombreros (*Cada miembro puede jugar más de un rol y un rol puede ser jugado por más de un miembro*).

6.2 Trabajadores

6.3 Actividad (*How?*)

Es **una parte de trabajo** que realiza un rol. Se repite tantas veces como sea necesario en una iteración.

6.4 Artefacto (*What?*)

Un documento, modelo, o elemento de un modelo producido, modificado, o usado por un proceso. La responsabilidad de los roles. Está normalmente sometido al control de configuración. Puede contener otros artefactos.

Es el resultado parcial o final que es producido y usado durante el proyecto. Entradas y salidas de las actividades.

Economía de artefactos:

Producir solamente los artefactos que vayan a **usarse**. Guardar los artefactos con la herramienta más apropiada. Usar informes para extraer resúmenes de información de los modelos para revisarlos.

6.5 Roles, actividades y artefactos

Diagramas.

6.6 Disciplinas

Es un conjunto de actividades relacionadas con un área de interés principal. Los artefactos obtenidos en cada disciplina evolucionan a lo largo del proceso iterativo. Las disciplinas producen modelos compartidos y guían el desarrollo iterativo.

6.7 RUP Workflows

En RUP, cada disciplina contiene un **workflow**. Un **workflow** es un flujo condicional de tareas de alto nivel (*Workflow Details*) que produce un resultado de valor observable.

7. Las disciplinas de RUP

7.1 Disciplina de Modelo de Empresa

Propósito:

Entender los problemas básicos de la organización e identificar las mejoras potenciales asegurándose que cliente y usuario tienen una opinión común, derivando los **requerimientos** del sistema para soportar los objetivos de la organización y comprendiendo la **estructura** y la **dinámica** de la organización para la que se va a desarrollar el sistema.

Modelo de Empresa y desarrollo del software:

El modelo de empresa actúa como:

Entrada a los requerimientos: El **modelo de CDU** de la empresa ayuda a comprender los requerimientos del sistema y a identificar los CDU del sistema.

Entrada al análisis y al diseño: Las **entidades** de la empresa del **modelo de análisis** de la empresa ayudan a identificar las clases del modelo de análisis.

7.2 Requerimientos

Propósito:

Establecer y mantener acuerdos con los clientes y otros stakeholders sobre lo que hará el sistema. Suministrar a los desarrolladores la mejor descripción de los requerimientos, definir las fronteras del sistema, suministrar las bases para la planificación de las iteraciones de los contenidos técnicos, suministrar una base para la estimación de costes y tiempo del desarrollo del sistema y definir la interfaz de usuario del sistema, centrándose en las necesidades y objetivos de los usuarios.

Tipos de requerimientos:

- **Elementos:** Usados para el alcance del proyecto.
- **Requerimientos funcionales:** Especifican las interacciones de los usuarios con el sistema.
- **Requerimientos suplementarios:** Especifican requerimientos de funcionalidad, usabilidad, etc.

7.3 Disciplina Análisis y Diseño

Propósito:

Transforma los requerimientos en el diseño del sistema, desarrolla una arquitectura robusta para el sistema y adapta el diseño para que encaje en el entorno de implementación.

El artefacto principal es el modelo de diseño.

Un **modelo de diseño** es un modelo de objetos que describe la realización de los CDU. Sirve como una abstracción del modelo de implementación y su código fuente. Se usa como una entrada esencial en las actividades de implementación y prueba.

7.4 Disciplina de implementación

Propósito:

Implementa clases y objetos en términos de componentes y código fuente, define la organización de los componentes en términos de subsistemas de implementación, prueba los componentes desarrollados como unidades y crea un sistema ejecutable.

El artefacto principal es el modelo de implementación.

El **modelo de implementación** está formado por **componentes y subsistemas a implementar**. Los **componentes** incluyen **componentes distribuidos, tales como ejecutables y los componentes desde los que los productos son generados, tales como código fuente**.

Construir o empaquetar es genera una versión operacional de un sistema o de parte de el. Demuestra un subconjunto de capacidades que suministra el producto final y es una parte del ciclo de vida iterativo. Además suministra puntos de revisión y ayuda a descubrir problemas de integración lo más pronto posible.

7.5 Disciplina de prueba

Propósito:

Encontrar y documentar **defectos de calidad** del SW. Suministra la **validación de las especificaciones** realizadas para el diseño y los requerimientos mediante una demostración. Se realiza una **validación de las funciones** del producto SW respecto a como fueron diseñadas y una **validación de que los requerimientos** han sido implementados correctamente. La disciplina de prueba se centra principalmente en **evaluar o asegurar la calidad** mediante una serie de prácticas. En muchos casos actúa como un **servicio prestado a las demás disciplinas**.

7.6 Disciplina de despliegue (Distribución)

Propósito:

Gestionar las actividades asociadas con garantizar que el producto de software este disponible para usuarios finales.

7.7 Disciplina de configuración y gestión de cambios

Propósito:

Controlar los cambios y mantener la integridad de los artefactos del proyecto.

Gestión de la configuración (CM):

Herramientas de gestión de la configuración: Soportan a los miembros del equipo del proyecto.

Gestión de las peticiones de cambio (CRM):**Dos tipos básicos:**

- **Defectos:** CRM soporta a todos los miembros del equipo en la gestión de los procesos.
- **Petición de mejoras:** CRM soporta a la dirección del control de cambios en la administración, evaluación y disposición de las propuestas de cambios del proyecto.

7.8 Disciplina de Project Management

Propósito:

Suministrar **una estructura** para gestionar proyectos de software, una **guía práctica** para planificar, dirigir, ejecutar y controlar proyectos y una **estructura para gestionar el riesgo**.

Caso de empresa:

Estimar los costes de desarrollo basándose en la estructura actual del directorio del producto y las decisiones realizadas por el arquitecto.

Estimación de los beneficios del proyecto para establecer el ROI.

7.9 Disciplina de Entorno

Propósito:

Centrarse en las actividades necesarias para configurar el proceso del proyecto. Define que mejoras son realistas en los aspectos de **procesos, herramientas, conocimientos y capacidades y problemas y posibles objetivos de mejora, actuales**.

8. Características del proceso en RUP

8.1 Proceso dirigido por los CDU

Diagramas.

8.2 Proceso Iterativo e Incremental

El ciclo de vida iterativo se basa en la evolución de prototipos ejecutables que se muestran a usuarios y clientes.

Respecto a la **duración de una iteración**, esta se inicia con la planificación y los requerimientos y finaliza con una reléase.

Las actividades se encadenan en una mini-cascada con un alcance limitado por los objetivos de la iteración.

Cada iteración comprende varias fases:

- **Planificar** la iteración.
- **Análisis** de CDU y escenarios.
- **Diseño** de opciones arquitectónicas.
- **Codificación y pruebas**.
- **Evaluación**.
- **Preparación**.

8.3 Proceso centrado en la arquitectura

Una arquitectura ejecutable es una implementación parcial del sistema, construida para demostrar algunas funciones y propiedades.

Tema 4: Fundamentos del DSI:

Metodología métrica

1. Introducción

Métrica es una metodología para la planificación, desarrollo y mantenimiento de sistemas de información. Métrica V3 ha sido concebida para abarcar el desarrollo completo de SI, sea cual sea su complejidad y magnitud.

Origen:

Métrica v2 proviene de SSADM v4, y Métrica v3 proviene de Métrica v2 y Eurométodo a su vez de diversos estándares.

Objetivos:

- **Abierta:** Apta para administración y empresas privadas.
- **Formal:** Establece pasos necesarios.
- **Orientada a obtener productos de calidad:** Énfasis en revisiones.
- **Flexible:** Adaptable a distintos proyectos, entornos, etc.
- **Sencilla:** Práctica y fácil de usar.

La estructura de un proyecto (ciclo de vida) se divide en:

- Procesos
- Actividades
- Tareas

Se basa en un enfoque orientado a proceso, actividad, tarea.
Distingue entre procesos principales (Planificación, desarrollo y mantenimiento) y de Interfaz (Calidad, seguridad, gestión y configuración).

Basada en el ciclo de vida secuencial o cascada.

Proceso:

Se divide en actividades, de 3 tipos: Comunes, solo AE, solo OO.

Actividad:

Descripción general y conjunto de tareas.

Tarea:

Descripción, productos (in, out), técnicas y participantes.

Al ser un estándar nacional se garantiza su evolución, recoge lo mejor de otras (SSADM, Merise), es flexible y aplicable a proyectos con diferentes ciclos de vida, y usa el mismo marco metodológico que el Eurométodo V1. Posee integración del enfoque estructurado y del OO (UML), incluye arquitectura cliente/servidor desde el principio, y en diseño usa la arquitectura de soporte como clave para reutilización.

1.1 Introducción: Organización

Para cada actividad y tarea de las mismas se especifican los participantes, los productos a obtener y las técnicas/prácticas a usar durante su desarrollo.

PSI (Planificación de sistemas de información)

DSI (Desarrollo de sistemas de información)

EVS (Estudio viabilidad del sistema)

ASI (Análisis sistema información)

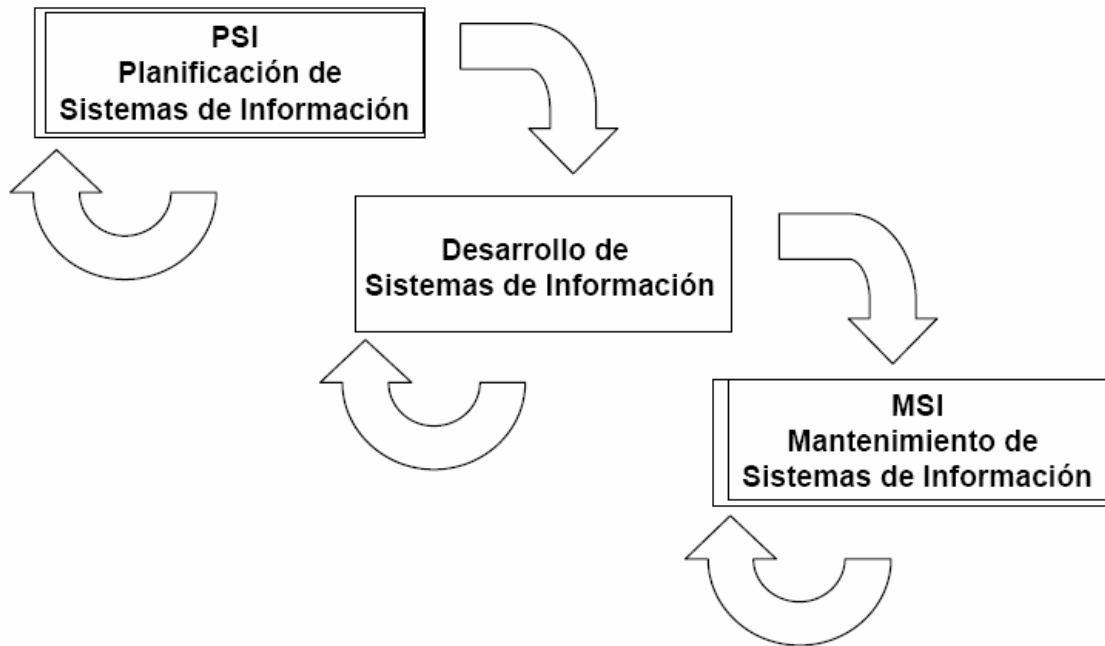
DSI (Diseño del sistema de información)

CSI (Construcción del sistema de información)

IAS (Implantación y aceptación del sistema)

MSI (Mantenimiento de sistemas de información)

2. Procesos



Procesos

Planificación de sistemas de información (PSI):

Proporciona un **marco estratégico** de referencia para los SI de un determinado ámbito de la organización.

Desarrollo de sistemas de información (DSI):

Contiene **todas las actividades y tareas** que se deben llevar a cabo para desarrollar un sistema, desde análisis de requisitos hasta instalación del SW.

Mantenimiento de Sistemas de información (MSI):

Obtener nuevas versiones del SI a partir de las peticiones de mantenimiento.

DSI es la parte más grande.

2.2 Procesos: DSI

Estudio de viabilidad (EVS):

Analizar un conjunto de **necesidades**, para **proponer una solución a corto plazo** teniendo en cuenta las posibles **restricciones** (económicas, técnicas, legales y operativas).

Análisis del sistema de información (ASI):

Conseguir **especificación detallada** del SI a través de un **catálogo de requisitos** y una serie de modelos.

Diseño del sistema de información (DSI):

Obtener la definición de la arquitectura del sistema y del entorno tecnológico, y la especificación detallada de los componentes del SI.

Construcción del Sistema de información (CS):

Construcción y prueba de los componentes. Desarrollo de procedimientos de operación y seguridad y de los manuales (de usuario y de explotación).

Implantación y aceptación del sistema de información (IAS):

Entrega y aceptación del sistema. Incluye actividades para el paso a producción del sistema.

3. Interfaces

Las interfaces definen actividades de tipo **organizativo o de soporte** al proceso de desarrollo. Las de Métrica V3 son las que se verán a continuación.

3.1 Interfaces Métrica V3: Gestión de proyectos.

Planificación, seguimiento y control de actividades y recursos humanos y materiales que intervienen en el desarrollo de un SI. La interfaz de gestión de proyectos de métrica v3:

- Acorde con eurométodo
- Proyectos de nuevos desarrollos
- Proyectos de ampliación y mejora

3.1.1 Actividades:

3.1.1.1: Actividades inicio proyecto (GP):

Después de concluir el proceso *EVS* se realizan las actividades de:

- Estimación de esfuerzo
- Planificación del proyecto

3.1.1.2: Actividades de seguimiento y control (GPS):

Comprende desde la **asignación** de las tareas hasta su **aceptación** interna, incluyendo la **gestión de incidencias y cambios** en los requisitos. El seguimiento y control se realiza durante los procesos de **Análisis, Diseño, Construcción, Implantación y Aceptación, y Mantenimiento**.

3.1.1.3: Actividades de finalización del proyecto:

Al concluir el proyecto se realizan las tareas propias de **cierre del proyecto y registro de la documentación de gestión**.

3.2 Interfaces Métrica v3: Seguridad

Se deben incorporar **mecanismos de seguridad** por cada proceso.

Para el *PSI*: *PSI-SEG*
Para el *EVS*: *EVS-SEG*
etc.

3.3 Interfaces Métrica v3: Aseguramiento de la calidad (CAL):

Se debe evaluar la **calidad del proceso y del producto** para reducir, eliminar y prevenir deficiencias.

Del *EVS* al *MSI* (Se excluye el *PSI*), cada proceso va ligado a su aseguramiento de calidad:

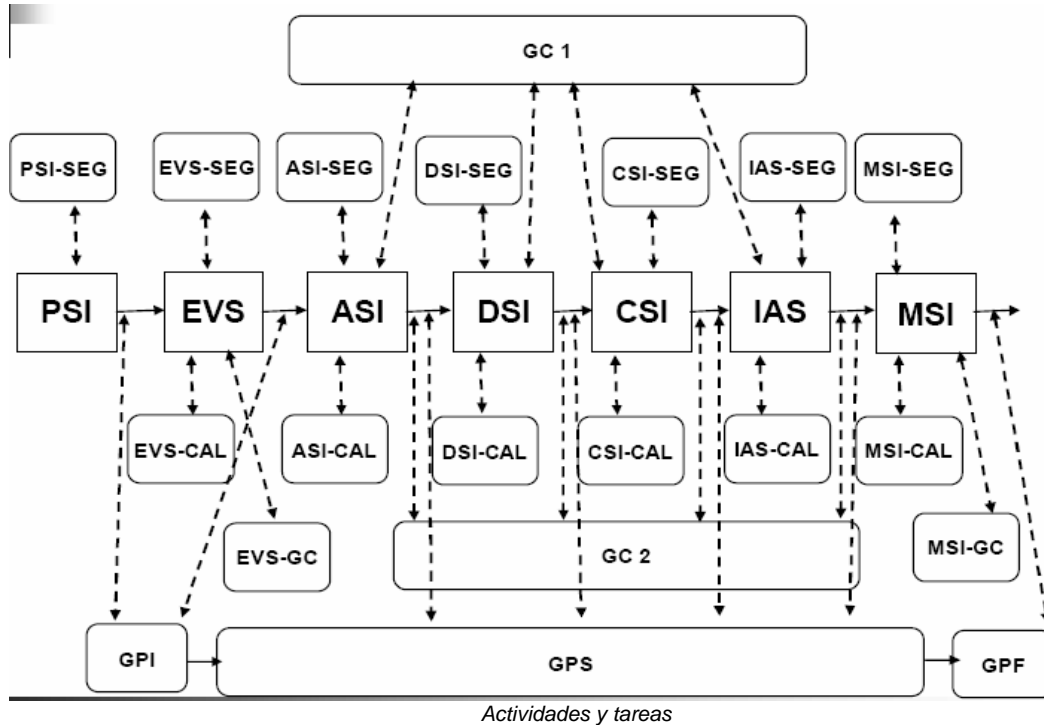
EVS: *EVS-CAL*
ASI: *ASI-CAL*
etc.

3.4 Interfaces Métrica v3: Gestión de la configuración (GC):

Se deben identificar, definir y controlar los cambios en la configuración del sistema.

4. Actividades y tareas

Cada proceso e interfaz está compuesto por actividades, y a su vez esta por tareas.



Recomendable echar un vistazo a los ejemplos que vienen en las transparencias.

5. Productos

Son las entradas y salidas de una tarea. Los productos establecen relaciones de dependencia entre tareas. Para su elaboración se usan técnicas o practicas específicas. Los productos externos son aquellos elaborados por tareas externas a las definidas en métrica v3.

En las transparencias se pueden ver los distintos productos de los procesos del DSI.

6. Técnicas y prácticas

Se usan en la elaboración de productos y mejoran la **productividad** y aseguran la **calidad** de los productos. Una **técnica** es un conjunto de heurísticas y/o procedimientos que utiliza una notación específica. Una **práctica** representa un medio para la consecución de unos objetivos específicos de forma rápida, segura y precisa.

6.1 Técnicas

6.1.1 Técnica de desarrollo:

- Diagramas *UML*, de estructura, de flujo de datos
- Modelados de proceso de la organización (*SADT*)
- Modelo entidad/relación (*ER*)
- Normalización
- Optimización
- Reglas de transformación de un diagrama *ER* o de clases a un modelo relacional

6.1.2 Técnicas de gestión de proyectos

- Análisis coste/beneficio
- Técnicas de estimación (Puntos función, Staffing Size)
- Técnicas de planificación (PERT, Diagrama de Gantt, Estructura de descomposición de trabajo, diagramas de extrapolación)

6.2 Practicas

Análisis de impacto, Catalogación, Cálculo de accesos, Caminos de acceso, Diagramas de representación, Factores críticos de éxito, Impacto en la organización, Presentaciones, Prototipado, Pruebas, Revisión formal, Revisión técnica, Sesiones de trabajo

Estructurado	OO
Modelo ER Extendido	Casos de uso
Normalización	Diagramas de clases
Regla obtención modelo físico	Diagrama Interacción de objetos
DFD	DTE
DTE	Diagrama de paquetes
Diagrama descomposición funcional	Diagrama de despliegue
Optimización del diseño	Diagrama de componentes
Caminos de acceso	Prototipazo
Cálculo de accesos	Optimización del diseño
Prototipazo	Reglas de transformación

7. Participantes

Los participantes del proyecto (*skateholders*) son aquellos que se verán afectados por el SI o por el proceso de desarrollo. Se clasifican en perfiles considerando **funciones y responsabilidades afines** y el conocimiento necesario respecto la metodología.

Perfiles establecidos:

7.1 Participantes: Perfil directivo

En esta categoría se agrupan los siguientes tipos de participantes:

- Comité de dirección
- Comité de seguimiento
- Directores de usuario
- Usuarios expertos

7.2 Participantes: Perfil jefe de proyecto

En esta categoría se agrupan los siguientes tipos de participantes:

- Jefe de proyecto
- Responsables de Implantación, Mantenimiento, Operación, Sistemas, Seguridad y Calidad

7.2 Participantes: Perfil consultor

En esta categoría se agrupan los siguientes tipos de participantes:

- Consultores (A secas, informáticos, de las tecnologías de la información y de sistemas de información).
- Especialista en comunicaciones
- Técnicos (De sistemas y de comunicaciones)

7.3 Participantes: Perfil analista

En esta categoría se agrupan los siguientes tipos de participantes:

- Analista
- Administrador de bases de datos
- Equipos de Arquitectura, Formación, Implantación, Operación, Seguridad, Soporte técnico y proyecto
- Grupo de aseguramiento de la calidad

8. Tipos de proyectos

- **Proyecto Grande (PG):** Desarrollo de un **sistema a medida** de gran tamaño o complejidad.
- **Proyecto pequeño (PP):** **Sistema a medida** de tamaño y/o complejidad limitado.
- **Desarrollo modular (DM):** **Sistema a medida** estructurable en distintos módulos relativamente independientes.
- **Mantenimiento de Sistemas (MS):** Cubre la entrega de un proyecto basado en la **modificación de un sistema existente**
- **Basado en paquete (BP):** Entrega de **sistemas mixtos** donde una parte estará basada en un paquete estándar y puede ser necesario algún tipo de desarrollo para parametrizar o mejorar su integración con otros sistemas.

- **Prototipado (PT): Sistema a medida** donde la especificación funcional no es muy detallada y se puede usar un entorno tecnológico avanzado que permita construir un sistema preliminar que se pueda probar. 3 tipos:
 - **Clase 1:** Simulación de **diálogos de pantalla**. Diseño y enlace de pantallas.
 - **Clase 2: Prototipado rápido.** Entorno tecnológico definido. Prototipo se reutiliza en Diseño y construcción. El manual de usuario se hace en Análisis.
 - **Clase 3:** Prototipos evolutivos. Construcción **incremental**. Creación de un prototipo inicial que funcione y se refina añadiendo nuevas funcionalidades.

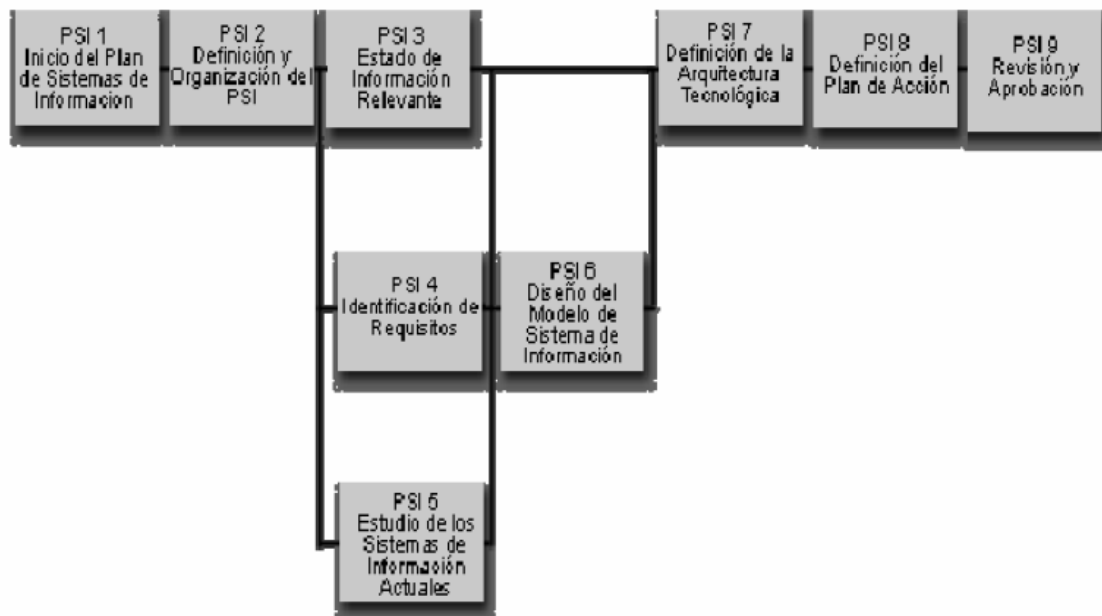
9. Proceso PSI

El objetivo es tener un marco de referencia para el desarrollo del SI que responda a los **objetivos estratégicos de la organización**. Se necesita:

- Una descripción crítica de la **situación actual**.
- **Arquitectura** de la información de alto nivel.
- **Modelos** que constituyen la arquitectura de la información
- Propuesta de **proyectos**
- Propuesta de **calendario y estimación de recursos**
- Plan de seguimiento y cumplimiento de lo propuesto bajo perspectiva estratégica y operativa.

9.1 PSI: Resumen

Se deben estudiar las **necesidades** de información de los procesos de la organización, definir los **requisitos** generales, obtener los **modelos** conceptuales de información y de SI, evaluar las **opciones** y proponer un entorno, elaborar un **calendario** y planificar en detalle los procesos más próximos.



Estructura PSI

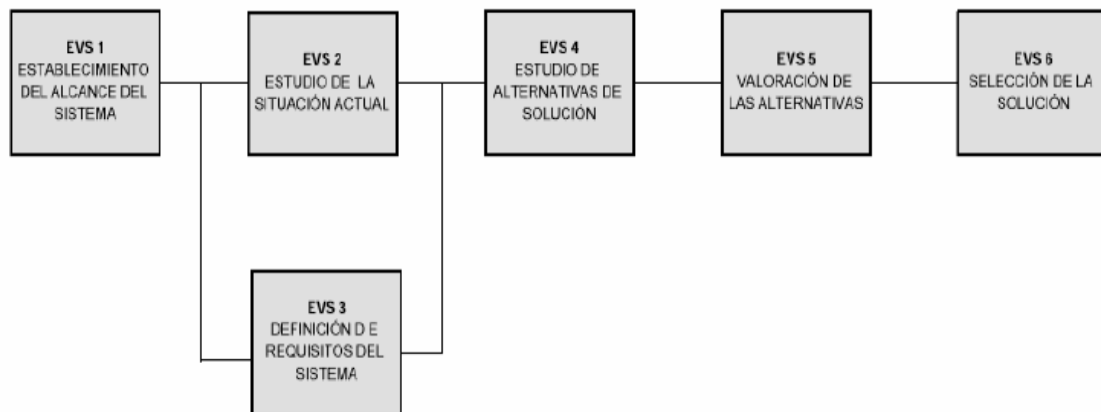
10. Proceso EVS:

El objetivo es analizar las necesidades y proponer una **solución a corto plazo** basada en criterios económicos, técnicos, legales y operativos. Se han de **identificar los requisitos** a satisfacer.

10.1 EVS: Resumen

Se estudian los requisitos a satisfacer y si procede la situación actual. Se planean **alternativas de solución** (a medida, basadas en productos SW, mixtas) y para cada alternativa se valora el impacto en la organización, la inversión a realizar y los riesgos asociados. Se evalúan las alternativas y se selecciona la más adecuada para definirla con detalle y establecer su planificación.

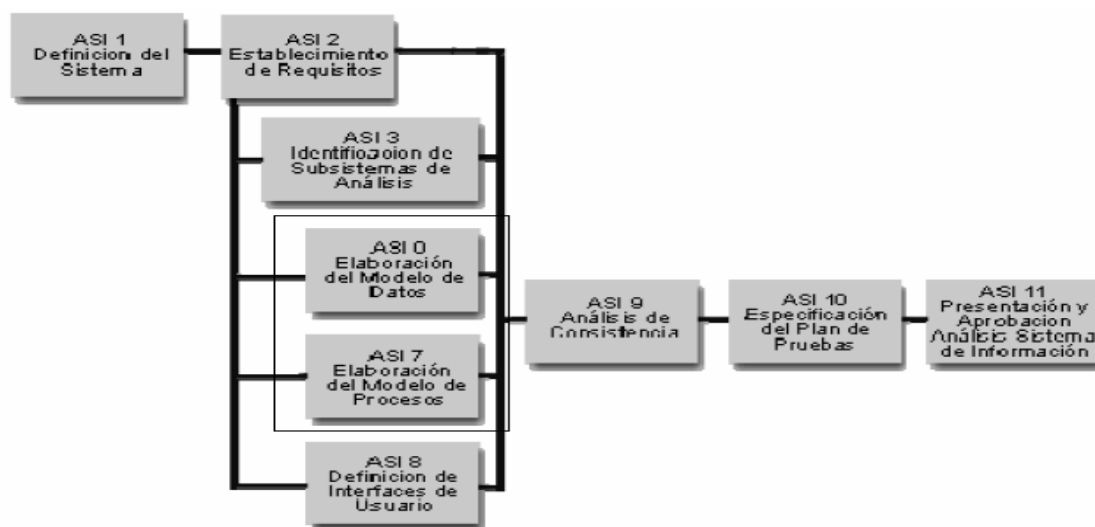
Si la justificación económica es obvia, el riesgo técnico bajo, se esperan pocos problemas legales y existe una alternativa clara, este proceso se orienta a la especificación de requisitos, descripción del nuevo sistema y planificación.



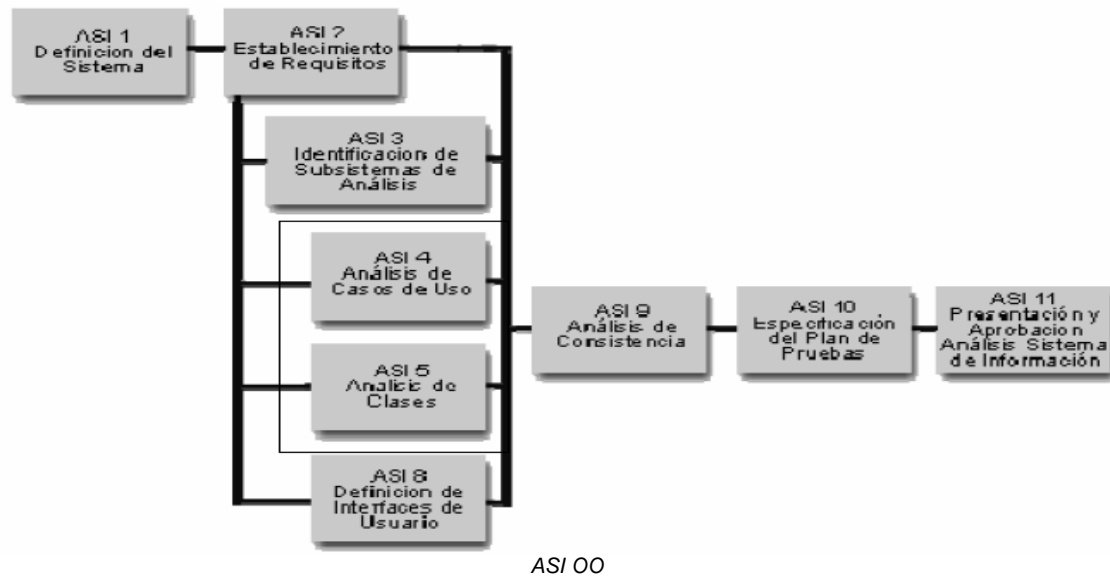
Estructura EVS

11. Proceso ASI

Su objetivo es obtener una **especificación detallada del SI** y de sus **interfaces con otros sistemas** que satisfaga las necesidades de información de los usuarios y sirva de base para el diseño. **Integra actividades de análisis estructurado y OO**. Se refinan los productos obtenidos en el EVS, y la **participación activa de los usuarios es una condición imprescindible para el análisis del SI** ya que es una garantía de **que los requisitos identificados son comprendidos e incorporados al sistema y por lo tanto será aceptado**.



ASI Estructurado



11.1. ASI 1: Definición del sistema

Productos que se generan:

Común:

Catálogo de requisitos generales, Glosario, Catálogo de estándares y de normas, de usuarios (participantes y finales), entorno tecnológico del sistema y plan de trabajo.

En AE: Contexto del sistema, modelo conceptual de datos

En AOO: Modelo del negocio/dominio

11.2. ASI 2: Establecimiento de requisitos

El objetivo es la definición, análisis y validación de los requisitos. Se completa el catálogo de requisitos.

Los **modelos gráficos de requisitos**: casos de uso (obligatorio en AOO, opcional en AE).

Las tareas se realizan de forma iterativa y con continuas realimentaciones y solapamientos.

11.2.1: ASI 2.1: Obtención de requisitos

- **Sesiones de trabajo con los usuarios:** Catálogos de requisitos y modelo de casos de uso.
- **Requisitos funcionales:** Con casos de uso (obligatorios en AOO): Actores, casos de uso y descripción.
- **Requisitos no funcionales:** Restricciones del entorno, niveles de servicio del sistema (Rendimiento, seguridad, implantación, disponibilidad).

11.2.2: ASI 2.2: Especificación de casos de uso

Para cada caso de uso se especifica el escenario principal, las pre-post condiciones, las interfaces de usuario y los escenarios secundarios.

11.2.3: ASI 2.3: Análisis de requisitos

Su objetivo es detectar inconsistencias, ambigüedades o duplicidad de información revisando las prioridades, relacionando con requisitos e identificando relaciones entre casos de uso.

11.2.4: ASI 2.4: Validación de requisitos

El objetivo es que los usuarios validen el catalogo de requisitos y los casos de uso.

11.3: ASI 3: Identificación de subsistemas de análisis

El objetivo es la descomposición del sistema en subsistemas en paralelo con el resto de actividades de generación de modelos de análisis. En *AE* se corresponde con el *DFD 0*. Se asignaran los requisitos y casos de uso a cada uno de los subsistemas identificados.

11.4: ASI 4: Análisis de los casos de uso

Solamente para Análisis orientado a objetos

El objetivo es **identificar las clases** cuyos objetos se necesitan para realizar un caso de uso. Se hace para **cada uno de los casos de uso** contenidos en los subsistemas identificados en *ASI 3*.

11.5: ASI 5: Análisis de clases

Solamente para Análisis orientado a objetos

El objetivo es describir cada una de las clases de la actividad anterior, *ASI 4*.

11.6: ASI 6: Elaboración del modelo de datos

Solamente para Análisis estructurado

Se usa como técnica el modelo *E-R* extendido y se completa el modelo conceptual de datos además de elaborar y normalizar el modelo lógico (*hasta 3FN*).

11.7: ASI 7: Elaboración del modelo de procesos

Solamente para Análisis estructurado

Se usa como técnicas *DFS* y matriciales. *ASI 7* en paralelo y con realimentaciones con *ASI 2, 6 y 8*. Para cada subsistema (*ASI 3*) se establece el conjunto de procesos que satisfacen las necesidades de usuario. Se usará un enfoque *top-down*. Se especifican con detalle las interfaces con otros *SI*.

11.8: ASI 8: Definición de interfaces de usuario:

Se especifican los estándares y directrices a tener en cuenta y se definen los formatos de pantalla, diálogos, informes y formularios.

11.9: ASI 9: Análisis de consistencia y especificación de requisitos

El objetivo 1 es garantizar la calidad de los modelos verificando la calidad técnica de cada modelo, el análisis de consistencia entre modelos y la validación de los modelos.

El objetivo 2 es elaborar la **ERS (Especificación de requisitos software)**: Introducción, ámbito y alcance, participantes, requisitos, visión general, referencia de productos a entregar y plan de acción.

11.10: ASI 10: Especificación del plan de pruebas

Se inicia la definición del plan de pruebas

11.11: ASI 11: Presentación y aprobación del análisis del SI

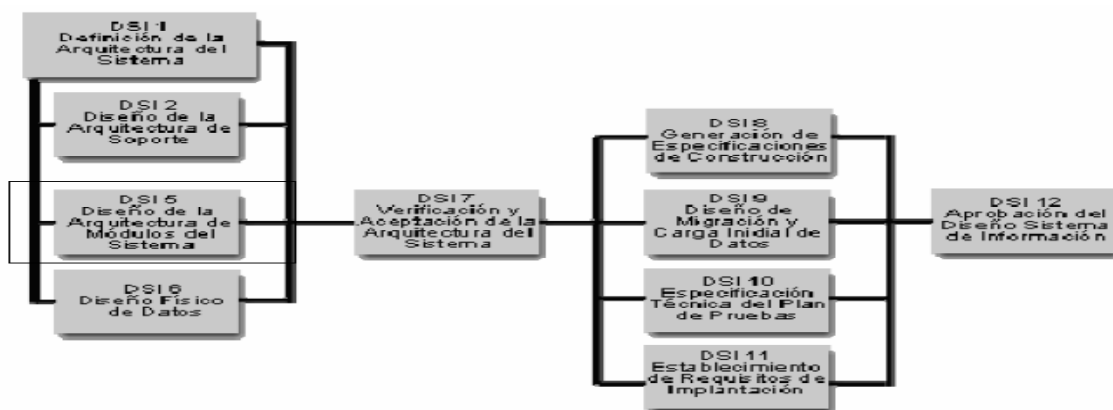
Se presenta el análisis del SI al comité de dirección para su aprobación final.

12. Proceso DSI

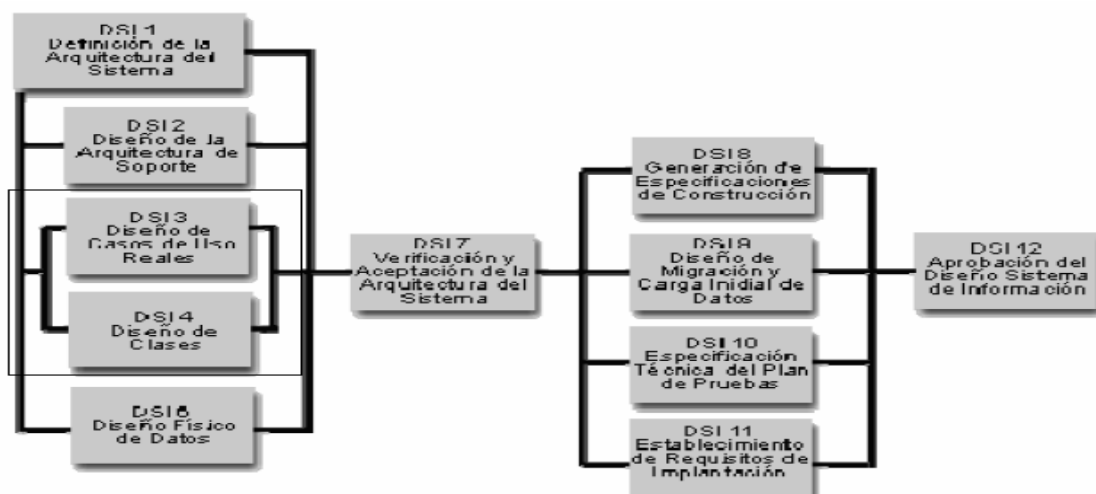
Sus objetivos son la definición de la arquitectura del sistema, del entorno tecnológico, la especificación de los componentes, del plan de pruebas, la definición de los requisitos de implantación y el diseño de los procedimientos de migración y carga inicial (si procede).

Las actividades de este proceso se agrupan en 2 grandes bloques:

- En el primer bloque se obtiene el diseño de detalle del SI
- En el segundo se general las especificaciones necesarias para construir el SI.



DSI Estructurado



DSI OO

12.1: DSI 1: Definición de la arquitectura del sistema

Se define en la arquitectura general del SI las particiones física y la descomposición lógica en subsistemas de diseño.

12.1.1: Definición de niveles en la arquitectura:

Se definen las principales particiones físicas del SI y sus componentes (gestores de datos, tipos puesto cliente, impresión, teleproceso, servidores, comunicaciones).

12.1.2: Identificación de subsistemas de diseño:

Hay **subsistemas específicos** como las funcionalidades propias del SI, o de soporte (servicios comunes, etc.). Los subsistemas se ubican en nodos y se especifica el entorno tecnológico.

12.2: DSI 2: Diseño de la arquitectura de soporte.

El objetivo es el diseño de los subsistemas de soporte identificados en *DSI 1*.

12.3: DSI 3: Diseño de casos de uso reales:

El objetivo es la identificación de las clases que intervienen **en los casos de uso**. Partiendo de los escenarios del ASI se detallan y se buscan comportamientos comunes además de realizar una revisión de la interfaz. Las tareas de esta actividad se realizan en paralelo con las de *DSI 4*.

12.4: DSI 4: Diseño de clases

Solamente para diseño Orientado a Objetos

El objetivo es transformar el modelo de clases lógico del análisis en un modelo de clases de diseño. Se identifican nuevas clases en el diseño.

12.5: DSI 5: Diseño de la arquitectura de módulos del sistema

Solamente para diseño estructurado

El objetivo es diseñar para cada uno de los subsistemas la estructura modular de los procesos que lo integran. Se parte del modelo de procesos obtenido en ASI y el catalogo de requisitos y se usa como técnica un diagrama de estructura de cuadros Constantine (*DEC*). Se realiza un diseño detallado de la interfaz de usuario, la cual ha de corresponderse con la estructura modular.

12.6: DSI 6: Diseño físico de datos

El objetivo es definir la estructura física de los datos que usará el sistema a partir del modelo lógico de datos normalizado o del modelo de clases.

12.7: DSI 7: Verificación y aceptación de la arquitectura del sistema.

El objetivo es garantizar la calidad de las especificaciones del diseño y su viabilidad.

12.8: DSI 8: Generación de especificaciones de construcción.

El objetivo es a partir del diseño anterior generar las especificaciones para la construcción del *DSI* incluyendo la especificación del entorno de construcción, la descripción de componentes, la especificación detallada de componentes y la especificación de la estructura física de datos.

12.9: DSI 9: Diseño de la migración y carga inicial de datos

Los objetivos es que se realice solo cuando es necesaria una carga inicial o migración. Se debe definir el entorno tecnológico del proceso, se dará una estimación de las capacidades para determinar infraestructuras, definir procedimientos a seguir, y revisión de los planes ya elaborados.

12.10: DSI 10: Especificación técnica del plan de pruebas

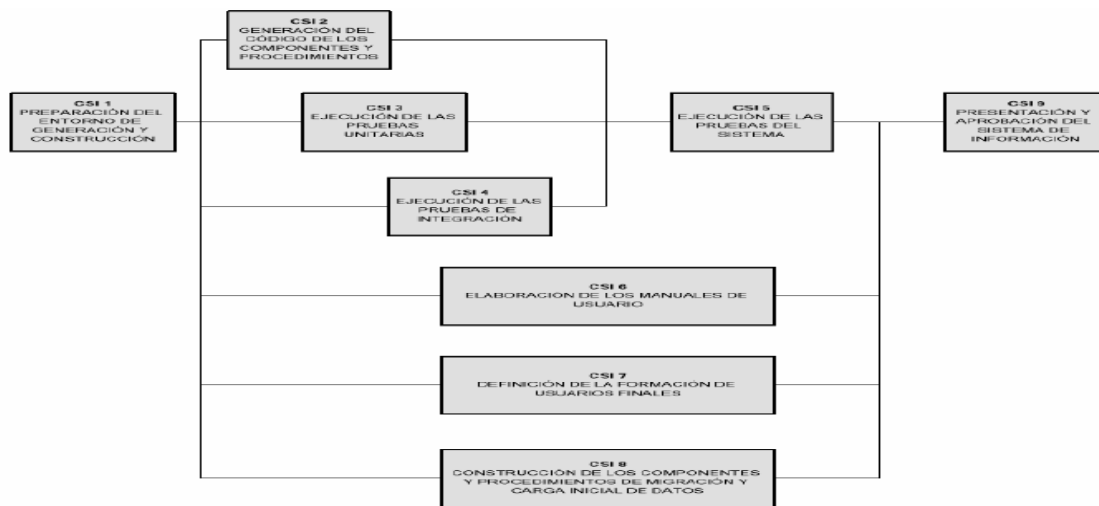
El objetivo es especificar en detalle el plan de pruebas del *S/* para los niveles de pruebas unitarias, de integración, de implantación y de aceptación.

13. Proceso CSI

El objetivo es codificar y probar los componentes del *S/* y escribir los manuales de usuario y explotación.

13.1. CSI: Resumen

Se prepara el entorno de construcción (bases de datos, herramientas, puestos de trabajo...), se codifican los componentes, se realizan las pruebas unitarias, de integración, y del sistema.



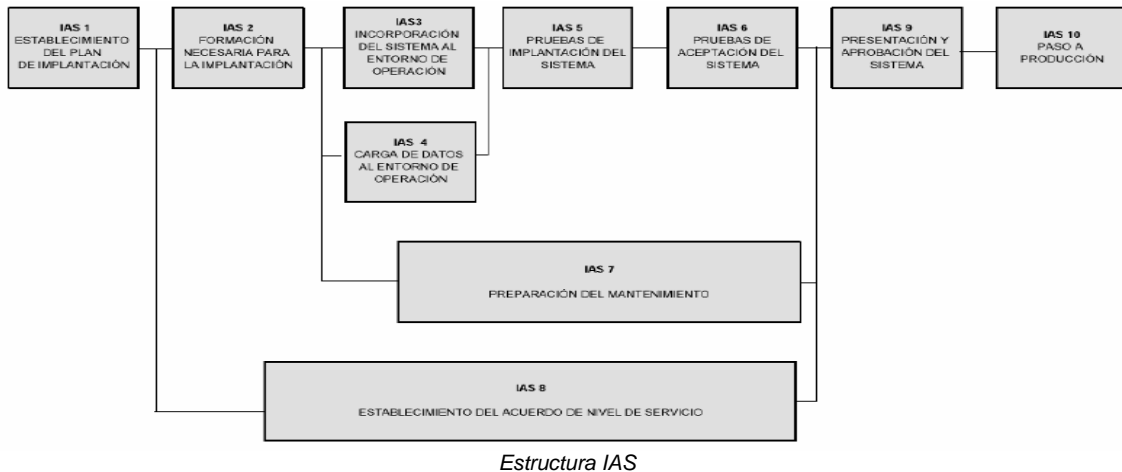
Estructura CSI

14. Proceso IAS

El objetivo es la entrega y aceptación del sistema en su totalidad y la realización de las actividades necesarias para dar paso a la producción:

Preparar el entorno de explotación, instalar componentes, activar procedimientos, carga inicial o migración de datos, pruebas de implantación, de aceptación y preparación del mantenimiento.

Las **pruebas de implantación** comprueban el comportamiento del sistema en el entorno de explotación, y las **pruebas de aceptación** tienen como objetivo que el usuario valide que el sistema se ajusta a sus necesidades.



15. Proceso MSI:

El objetivo es obtener una nueva versión del *SI* a partir de las peticiones de mantenimiento de los usuarios por algún problema detectado.

Tipos de mantenimiento:

- **Correctivo:** Cambios para corregir errores
- **Perfectivo (Evolutivo):** Expansión o cambio en necesidades de usuario
- **Adaptativo:** Modificaciones por cambio en el que el SW. opera.
- **Preventivo:** Mejorar calidad interna de los sistemas.

MSI solo mantenimiento correctivo y evolutivo.

15.1 MSI: Resumen

Se registra la petición de mantenimiento y se determina quien debe atenderla. Si la petición no es denegada:

Si el mantenimiento es correctivo: Se reproduce el problema.

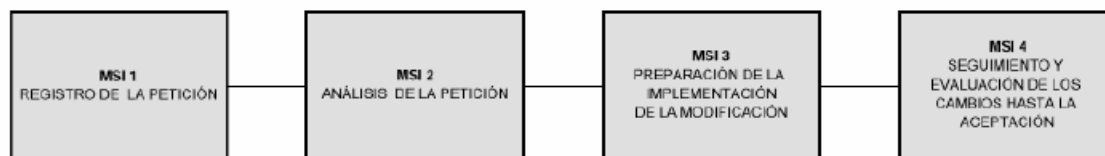
Si es evolutivo: Se estudia la viabilidad del cambio que propone el usuario.

Se analizan alternativas de solución.

Se realizan las tareas necesarias de los procesos de desarrollo *ASI*, *DSI*, *CSI*, *IAS*.

Se realizan pruebas de regresión.

Se registran los cambios realizados en el *SI* y su documentación.



Estructura MSI

16. Adaptación de la metodología del entorno

No existe un método universal o ideal, estará condicionado por el tamaño y estructura de la organización y el tipo de aplicaciones.

Tema 6: DSI con TE. Modelado de requerimientos

1. El proceso de Ingeniería de Requerimientos

El objetivo es identificar lo que espera un usuario/cliente de un sistema de software. Tiene una gran importancia dado que los **errores** en esta fase son los más numerosos y el **coste** crece exponencialmente con el retraso en subsanarlo. **Mejorar esta fase** reporta grandes beneficios.

1.1 Estudio de viabilidad

Alternativas:

- Comprar un producto SW comercial ya construido
- Desarrollar el producto internamente
- Desarrollarlo de forma externa mediante contrato (outsourcing)
- Automatizar parcialmente el sistema para reducir gastos
- **Económico:** Determinar si el beneficio compensa los costes.
- **Técnico:** Estudiar si la funcionalidad, el rendimiento, son realizables.
- **Legal:** Determinar si los requisitos violan o atenta contra alguna ley.
- **Operativa:** Determinar la implantación efectiva.
- **Especificación detallada de la alternativa**
- **Definición del plan inicial del proyecto**

1.2 Análisis de requerimientos

El objetivo es entender los requerimientos del cliente para un sistema SW. Involucra trabajo de grupo con los clientes para averiguar **el dominio de la aplicación, los servicios que el sistema debe proporcionar y las restricciones**. Debe involucrar a los **usuarios finales, administradores, ingenieros, etc...**

1.2.1 Problemas del análisis de requerimientos

Los **especialistas (skateholders) no saben lo que quieren**. Éstos expresan **requerimientos en sus términos...**

Diferentes especialistas pueden tener **requerimientos en conflictos y los factores políticos y organizacionales pueden influir en los requerimientos** los cuales cambian durante el proceso de **análisis**.

1.3 Requerimientos: Definición y especificación

Definición: Una declaración en un **lenguaje natural** incluye los diagramas de los servicios del sistema y sus límites operaciones.

Especificación de requerimientos: Un documento estructurado con detalle de los servicios.

Especificación de software: Es una descripción detallada de software.

1.3.1: Definición de requerimientos

Especifica el comportamiento externo del sistema. Incluye **requerimientos funcionales y no funcionales**. Se usa para su definición el **lenguaje natural** además de **diagramas y tablas**. Pueden presentarse problemas de **falta de claridad, confusión en los requerimientos y mezcla de requerimientos**.

1.3.2: Especificación de requerimientos:

La especificación añade detalles a la definición. Suele ser presentada mediante **modelos de sistema**.

2. Requisitos funcionales y no funcionales

Requisitos funcionales: Describen la funcionalidad o servicios que se espera que el sistema proveerá, entradas y salidas, excepciones, etc.

2.1 Requisitos funcionales:

2.1.1 Requisitos sobre la actualización de datos: Características sobre las funciones que cambian la información del sistema.

2.1.2 Requisitos sobre la estructura de la información: Características de los datos que el software maneja.

2.1.3 Requisitos sobre las consultas y los informes

2.1.4 Requisitos sobre la interacción con otros sistemas

2.2 Requisitos no funcionales

Se refieren a las propiedades emergentes del sistema como la fiabilidad, tiempo de respuesta, capacidad de almacenamiento, etc.

2.2.1 Requisitos de fiabilidad: Frecuencia y severidad de los fallos, recuperación, seguridad...

2.2.2 Requisitos de usabilidad: Facilidad de uso, homogeneidad, etc.

2.2.3 Requisitos de sostenible: Facilidad de prueba, ampliación, mantenimiento, etc.

2.2.4 Requisitos de rendimiento: Límites al rendimiento y volúmenes de información que el SW debe tratar.

2.2.5 Requisitos de seguridad: Características de control de acceso al SW y copias de seguridad.

2.2.6 Requisitos de comunicaciones: Características de la red que compone el sistema.

3. Documento de requerimientos

Es la declaración oficial de lo que es requerido para que el sistema sea desarrollado. Incluye definición y especificación de requerimientos. **No es un documento de diseño**.

3.1 Requerimientos del documento de requisitos

Debe especificar la conducta externa del sistema, límites de implantación... Ha de ser fácil de cambiar y servir como herramienta de referencia además de recordar el ciclo de vida del sistema.

3.2 Estructura del documento de requerimientos

Consta de: *Introducción, Glosario, Modelos del sistema, Definición de requerimientos funcionales y no funcionales, evolución del sistema, especificación de requerimientos, apéndices e índice.*

3.3 Validación de requerimientos

Los requerimientos definen lo que el cliente quiere realmente. La **validación es muy importante** por el coste de los errores. El prototipado es una técnica importante de validación de requerimientos.

3.4 Chequeando requerimientos

- Validación
- Consistencia
- Completo
- Realismo

4. Técnicas de recogida de información

4.1 Entrevistas/Reuniones

Las **entrevistas** son un tipo de **sesiones de trabajo** dirigidas a obtener información individualmente. Las **reuniones** pueden tener el mismo objetivo pero va dirigido a **varias personas**. Es una **forma de conversación** ideal para recoger opiniones, comentarios, etc. y son el medio para obtener información sobre:

- Requerimientos de usuario
- Funcionamiento del sistema actual
- Organización de la unidad
- Responsables y funciones de los usuarios

En una entrevista las apariencias engañan y pueden tener un contenido muy denso. Debe evitarse ambigüedad y malas interpretaciones. Su objetivo es **descubrir tantos requerimientos como sea posible**. Cinco puntos principales:

Que, como, Que cosas (con qué), Cuando, cuanto.

El entrevistador debe evitar entrevistar a gente inadecuada, hacer preguntas inadecuadas, crear malas relaciones y ha de tener en cuenta que **usuarios y analistas hablan lenguajes diferentes**. Ir con tacto.

4.1.1 Fases de una entrevista

Preparación: El entrevistador tiene que documentarse.

Desarrollo: Se divide en apertura (presentaciones, etc.). Desarrollo (Preguntas abiertas y generales y luego más concretas) y final (preguntas para ponderar la efectividad de la entrevista).

Consolidación: Documentar resultados un acta y enviarlo a los participantes.

4.1.2 Contenidos

- **Para responsables de área:** SI actuales, funciones del departamento...
- **Para resto de usuarios:** Situación actual, procesos y funciones realizadas por los sistemas actuales...
- **Directivo:** Políticas, estrategias y objetivos. Limitaciones globales.
- **Ejecutivo:** Objetivos departamentales
- **Operativo:** Funciones actuales, evaluación satisfacción trabajo, sugerencias...

4.2 JAD (*Join Application Development*):

Proceso de recolección de datos acelerados donde usuarios y analistas se reúnen en una única e intensa reunión para definir los requerimientos de los usuarios.

4.2.1 Características del JAD:

- Establecer un equipo de trabajo donde cada uno sabe sus responsabilidades.
- Pocas reuniones de larga duración y bien preparadas.
- En la sesión se elaboran modelos empleando diagramas fáciles de entender y mantener directamente en herramientas CASE.
- Al finalizar se obtiene un conjunto de modelos que tienen que ser aprobados por los participantes.

4.2.2 Perfiles en un JAD:

- **Moderador (*líder JAD*):** Amplios conocimientos de metodología de trabajo, psicología comportamiento, etc.
- **Promotor:** Persona que impulsa el desarrollo
- **Jefe de proyecto:** Responsable de la implantación.
- **Especialista en modelización:** Responsable de elaborar los modelos.
- **Desarrolladores:** Aseguran que los modelos son correctos y responden a los requisitos especificados.
- **Usuarios:** Responsables de definir los requisitos y validarlos.

4.3 Prototipado:

Consiste en la **elaboración de un modelo** o maqueta del sistema que se construye para evaluar mejor los requisitos. Su empleo permite asegurar que está bien diseñada, evaluar el rendimiento de un diseño. El prototipado funciona está basado en el ciclo de vida iterativo, supondrá una **primera versión** del sistema con funcionalidad limitada.

4.4 Tablas de decisión:

Es utilizable en análisis, diseño y programación y es la representación de las relaciones entre condiciones y sus acciones. Son fáciles de leer y comprender.

5. Técnicas y prácticas en Métrica 3

Técnicas de recogida de información

Actividades Proceso EVS:

- **EVS 1:** Establecimiento del alcance del sistema.
- **EVS 2:** Estudio de la situación actual.
- **EVS 3:** Definición de requisitos del sistema.
- **EVS 6:** Selección de la solución.

Actividades Proceso ASI:

- **ASI 1:** Definición del sistema.
- **ASI 2:** Establecimiento de requisitos.
- **ASI 6:** Elaboración del modelo de datos.
- **ASI 8:** Definición de interfaces de usuario.
- **ASI 10:** Especificación del plan de pruebas.

Actividades Proceso DSI:

- **DSI 1:** Definición de la arquitectura del sistema.
- **DSI 2:** Diseño de la arquitectura del sistema.
- **DSI 4:** Diseño de clases.
- **DSI 11:** Establecimiento de los requisitos de implantación.

Actividades Proceso IAS:

- **IAS 1:** Establecimiento del plan de implantación.
- **IAS 7:** Preparación del mantenimiento del sistema.
- **IAS 8:** Establecimiento del acuerdo de nivel de servicio.

Actividades Proceso MSI:

- **MSI 2:** Análisis de la petición.

Tema 7. DSI Con TE: Modelado de procesos

1. Introducción al DFD

Técnica de modelado: Muestra un sistema como una red de procesos conectados entre sí por flujos y almacenamiento de datos.

Su objetivo es construir un **modelo lógico** del sistema que facilite su comprensión tanto al equipo de desarrollo como a sus usuarios. Establecerá **que** funciones se deben desarrollar sin implicar **como**.

El modelo del sistema es **independiente de las restricciones** físicas del entorno lo cual facilita el mantenimiento y la portabilidad. El **diagrama de flujo de datos (DFD)** proporciona representación del sistema a nivel **lógico y conceptual**.

DFD proporciona visión general de las **funciones y transformaciones de datos**. Identifica **entradas, salidas, procesos y relaciones** con el exterior.

2. Elementos básicos del DFD



2.1 Entidad externa

Representa entes ajenos a la aplicación que aportan o reciben información y señalan los límites del sistema y sus relaciones con el entorno.

Si las entidades externas se comunican entre sí, no se contempla en el diagrama. Puede aparecer en los distintos niveles del *DFD*.

2.2 Proceso

Actividad que transforma o manipula datos. No es origen ni final de datos, solo **lugar de transformación**. Un proceso puede transformar un dato en varios. Es necesario un proceso entre una entidad externa y un almacén de datos.

2.3 Almacén de datos

Deposito de información dentro del sistema. Representa información en reposo. **No puede** crear, destruir ni transformar datos ni estar comunicado directamente con otro almacén o entidad externa. **No debe estar referido al entorno físico.**

Para facilitar **validaciones cruzadas** entre *DFDs* y *E-R* (o *DED*):

- Existe **correspondencia** entre los almacenes de datos "principales" del *DFD* y las entidades del *E-R*.
- **Cada almacén** de un *DFD* representa **una o varias entidades** del *E-R*.
- **Cada entidad** del *E-R* pertenece a **un único almacén** principal de un *DFD*.

2.4 Flujo de datos

Establece **comunicación** entre procesos, almacenes y entidades externas.

3. Construcción del DFD

Pasos:

1. Representar el diagrama de contexto (*nivel 0*)
2. Representar el *DFD* de **primer nivel indicando los subsistemas del sistema.**
3. Descomponer cada uno de los procesos del *DFD* del primer nivel.

Se recomienda usar hasta el nivel **cuatro (5 niveles)** de descomposición de diagramas.

3.1 Diagrama de contexto

Su objetivo es **realizar una declaración formal del dominio.** Las **entidades externas** aparecen por primera vez en este nivel (y no en otro).

4. Descomposición por niveles

Cada proceso se puede descomponer en un *DFD* más detallado. El *DFD* de un sistema es un **conjunto de DFDS dispuestos jerárquicamente.**

El sistema ha de contener:

- Un diagrama de contexto (primer nivel).
- Varios *DFD* en niveles intermedios.
- Varios *DFD* en el último nivel de detalle.

Puede aparecer un **proceso primitivo (PP)** que es un proceso que no necesita descomposición.

En un *DFD* completo **cada proceso tiene un número único identificador.** Los **almacenes** aparecen lo más tarde posible.

5. Reglas sintácticas

El origen y/o destino de un *FD* es siempre un proceso. Todo almacén y proceso tiene uno o más *FD* de entrada o salida. Hay una **excepción**: Un almacén puede no tener *FD* de salida por simplificación.

6. Flujos de control

En los DFDs **no se muestra el control ni orden de ejecución**. No se pueden mostrar procesos que se realizan antes que otros, ni sincronización ni periodificación. **Sirven para "habilitar/deshabilitar" procesos**. También existen **procesos de control** (*coordinan otros procesos*) y **almacenes de control** (*almacenes de eventos*).

7. Localización de los procesos

8. Técnicas de especificación de procesos primitivos

- Lenguaje estructurado
- Pre y post-condiciones
- Tablas de decisión
- Árboles de decisión
- Diagrama de flujo
- Diagrama N-S

9. Visión panorámica

10. DFD en Métrica V3

Proceso EVS:

EVS 1: Establecimiento del almacén del sistema.

EVS 1.2: Identificación del alcance del sistema.

EVS 2: Estudio de la situación actual.

EVS 2.1: Valoración del estudio de la situación actual.

EVS 2.3: Descripción de los sistemas de información existentes.

Proceso ASI:

ASI 1: Definición del sistema.

ASI 1.1: Determinación del alcance del sistema.

ASI 3: Identificación de subsistemas de análisis.

ASI 3.1: Determinación de subsistemas de análisis.

ASI 3.2: Integración de subsistemas de análisis.

ASI 7: Elaboración del modelo de procesos.

ASI 7.1: Obtención del modelo de procesos del sistema.

Tema 8. DSI con TE. Modelado de datos

1. Modelo de datos

Es un **conjunto** de conceptos, reglas y convenciones para escribir y manipular datos. Es una *abstracción*. Su objetivo es obtener una **representación de la información del sistema independientemente de las aplicaciones y dispositivos físicos**.

Como **ventajas** destacan la noción de **independencia** entre las **estructuras lógica y física** de datos.

1.1 Niveles del modelado de datos

1.1.1 Nivel conceptual

Está orientado hacia la **visión lógica** del conjunto de información que proviene del mundo real. **Que** datos almacenar y las **relaciones** entre ellos.

1.1.2 Nivel lógico global/externo

Orientado hacia el **usuario** que comprende las características lógicas de los **datos** para los **programas** de aplicación.

1.1.3 Nivel interno

Orientado hacia la máquina, comprende características de tipo físico (*también llamado nivel físico*). Abstracción mínima. **COMO se almacenan los datos**.

2. Modelo entidad-interrelación (E/R)

Sirve para establecer una **visión global** de los datos de una organización o sistema de información. Constituye el nivel conceptual de la arquitectura *ANSI*. Consiste en describir la información de la organización mediante definición de entidades, atributos y asociaciones entre ellas.

3. Elementos básicos del Modelo E/R

3.1 Entidad

Cualquier objeto real o abstracto sobre el que queremos tener información que tiene **existencia** y por si mismo se puede **identificar** de manera clara y precisa. Ej.: Objetos reales, personas, actividades del sistema, objetos abstractos...

Una entidad se representa mediante un **rectángulo** con un **nombre**.

Características:

- Cada uno de sus miembros individuales (*instancias*) se identifican **unívocamente**.
- **Cada entidad** juega una función dentro del sistema.
- **Cada entidad** puede ser descrito por uno o más datos elementales (*atributos*).

3.2 Atributos

Es cada una de las propiedades o características en una entidad o interrelación. Aquel o aquellos que identifican unívocamente cada ocurrencia de la entidad se denomina **identificador principal**.

3.3 Interrelaciones

Asociación o correspondencia entre entidades. Se denotan con un rombo con el nombre de la relación que une las entidades.

Cada instancia de la interrelación representa una asociación entre 0 o más ocurrencias de un objeto y 0 o más ocurrencias de otro objeto.

- **Grado de interrelación:** Número de entidades participantes (unitarias o reflexivas, binarias, n-arias).
- **Cardinalidad máxima o tipo de interrelación:** Número máximo de ocurrencias de cada entidad que pueden intervenir en la interrelación.

4. El modelo E/R Extendido

Incorpora ciertos mecanismos de abstracción que facilitan la representación de ciertas estructuras del mundo real.

La **generalización** permite abstraer un tipo de entidad de nivel superior (**supertipo**) a partir de varios tipos de entidad (**subtipos**). Los **atributos comunes y relaciones** de los subtipos se asignan al supertipo.

La **especialización** es la operación inversa a la generalización. En ella un supertipo se descompone en uno o varios subtipos, los cuales heredan todos los atributos y relaciones del supertipo además de tener los suyos propios.

Categorías. Se denomina categoría al **subtipo** que aparece como resultado de la **unión de varios tipos de entidad**.

La **agregación** consiste en construir un nuevo tipo de entidad como **composición** de otros y su tipo de relación y así poder manejarlo en un nivel de abstracción mayor.

La **asociación** consiste en relacionar dos tipos de entidades que normalmente son de dominios independientes, pero coyunturalmente se asocian.

5. Construcción del modelo E/R

Pasos generales:

1. Identificar tipos de **entidades**.
2. Identificar tipos de **interrelaciones**.
3. Encontrar las **cardinalidades**.
4. Identificar los **atributos** de cada tipo de entidad.
5. Identificar las **claves** de cada tipo de entidad.

6. El modelo relacional

Se correspondería al nivel lógico/externo de la arquitectura *ANSI*. Los datos se estructuran en forma de **tablas** o relaciones manteniendo la **independencia** de esta **estructura lógica con las características físicas**.

A las columnas de las tablas se las denomina **atributos**, a las filas **tuplas**. El número de atributos se denomina **grado** y el número de tuplas **cardinalidad**.

La **relación o tabla** del modelo relacional tiene ciertas características:

- No puede haber tuplas duplicadas.
- El orden de las tuplas es irrelevante.
- La tabla es plana, es decir, en el cruce de un atributo y una tupla solo puede haber un valor.
- El orden de los atributos no es significativo.
- El conjunto no vacío de atributos que identifica unívocamente cada tupla se denomina **clave candidata**. El usuario debe elegir una, y será la **clave primaria**.
- **Integridad de una entidad:** Ningún atributo que forme parte de la clave primaria puede tomar un valor nulo.
- **Descriptor de una relación:** Es un conjunto (*no vacío*) de atributos de una relación.

Se llama **clave ajena** a un atributo o conjunto de ellos de una tabla o relación que coinciden con los de la clave primaria de otra tabla o relación. Ambas claves, primarias y ajenas, toman valores del mismo dominio. Esta clave sirve para relacionar tablas.

7. Diagrama de estructura de datos (DED)

Los DED son unas técnicas de **representación gráfica** de los esquemas lógicos de datos en modelos convencionales.

Las **entidades** se representan mediante un rectángulo y las **interrelaciones** mediante una línea recta que une entidades. Esta línea puede acabar en un tridente para indicar **cardinalidad superior a 1**.

Las interrelaciones entre entidades serán siempre de grado 2. **En los DED solo se consideran interrelaciones de cardinalidad 1:N**. En otro tipo de cardinalidades se han de reagrupar:

- **Caso 1:1:** Se pueden agrupar las dos entidades en una sola, añadiendo los atributos de una a la otra.
- **Caso M:N:** Se creará una entidad auxiliar que sirva de nexo de unión entre las dos entidades iniciales obteniéndose dos relaciones tipo 1:N.

La **opcionalidad** se representa gráficamente con una circunferencia.

DED es, básicamente, un E-R limitado (no relaciones ternarias, solo cardinalidades 1:N, no atributos multivaluados ni compuestos).

8. Construcción del DED

Etapas de construcción (*sin haber elaborado el esquema conceptual*):

1. **Identificar las entidades.**
2. Determinar las **claves** o identificadores de las entidades.
3. Establecer y representar **las interrelaciones** entre entidades.
4. Analizar **cardinalidad y grado** de las interrelaciones.
5. Indicar en el diagrama la **opcionalidad** de la interrelación y si existe **exclusividad** entre interrelaciones.
6. Identificar y describir **los atributos** de cada entidad.
7. Verificaciones.
8. Evitar redundancias.

9. Teoría de la normalización

Elimina dependencias entre atributos que original anomalías en la actualización de la base de datos y proporciona una estructura más regular en la representación de tablas, **constituyendo el soporte para el diseño de bases de datos relacionales.**

Ayuda a encontrar **fallos del diseño** y corregirlos. Se dice que una relación está en una determinada **forma normal** si cumple un **conjunto de restricciones** sobre los atributos.

Su objetivo es representar las entidades de datos de la forma más "compacta" posible, consiguiendo:

- Reducir inconsistencias.
- Eliminar redundancias.
- Reducir el impacto de los cambios.
- Facilitar el mantenimiento.

9.1 Primera forma normal (1FN)

Una relación está en primera forma normal **si no tiene grupos repetitivos**, es decir, un atributo solo puede tomar un único valor de un dominio.

Dependencia funcional: Un descriptor Y se dice que depende funcionalmente de otro X, si y solo si **a cada valor de X le corresponde un único valor de Y**. Se representa $X \rightarrow Y$.

Dependencia funcional completa: Un descriptor Y tiene dependencia funcional completa respecto de otro X, **si depende funcionalmente de él en su totalidad**, es decir, no depende de ninguno de los posibles atributos que formen parte de X. Sea X un descriptor compuesto de X1 y X2; Y tiene dependencia funcional completa de X si cumple:

$$\begin{aligned} X &= X1+X2 \\ X &\rightarrow Y \\ X1 &\rightarrow Y \\ X2 &\rightarrow Y \end{aligned}$$

Lo que expresa: $X \rightarrow Y$

Dependencia transitiva: Un descriptor depende transitivamente uno de otro si y solo **si depende de él a través de otro** descriptor. Z depende transitivamente de X, si:

$$\begin{aligned} X &\rightarrow Y \\ Y &\rightarrow Z \\ X &\rightarrow Z \end{aligned}$$

Ya que X implica Z, pero a través de Y.

9.2 Segunda forma normal (2FN)

Una relación está en 2FN si está en 1FN y todos los atributos **que no forman parte de las claves tienen dependencia funcional completa respecto de éstas**, es decir, no hay dependencias de atributos no principales respecto de una parte de las claves.

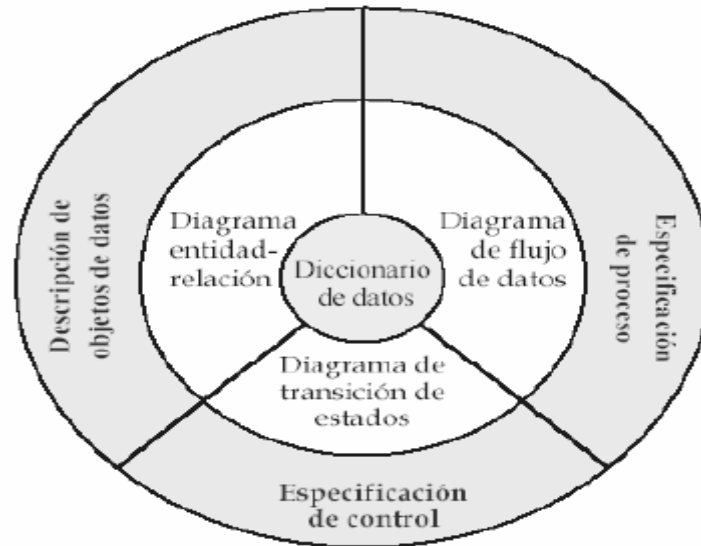
9.3 Tercera forma normal (3FN)

Una relación está en 3FN si está en 2FN y **todos sus atributos que no principales dependen directamente de la clave primaria**, es decir no hay dependencias transitivas de atributos no principales respecto de las claves. En otras palabras, si cada uno de los atributos de la entidad dependen **solo** de la clave.

10. Diccionario de datos

Es un conjunto de metadatos, es decir, de información (*datos*) sobre datos. Objetivos:

- Glosario de términos.
- Establecer terminología estándar para todos.
- Proporcionar referencias cruzadas.
- Proporcionar control centralizado para cambios.



Diccionario de datos

Definición de los datos:

- Describe el significado de los **flujos de datos** y los **almacenes** de los *DFD*'s.
- Describe la composición de las **estructuras de datos que se mueven a lo largo de los flujos**.
- Describe la composición de las **estructuras en los almacenes**.
- Describe detalles de **las relaciones** en los *DER*'s.

Elementos de dato:

- **Nombre:** Asignar nombres significativos.
- **Alias:** El mismo dato recibe varios nombres.
- **Longitud:** Cantidad de espacio necesario para cada dato.
- **Intervalo de valores:** Los valores de los datos están restringidos a un intervalo específico.

Los **datos elementales** son aquellos que dentro del contexto del usuario no tiene sentido descomponerlos.

Estructura de dato:

Se construyen sobre cuatro relaciones de componentes (*datos o estructuras*):

- **Secuencial +:** Define los componentes (*datos o estructuras*) que siempre se incluyen en una *ED*. Se realiza mediante el símbolo '=' que se define como:
 - Una expresión como $A=B+C$ (*A está compuesto de B y C*).
- **Selección []:** Define alternativas para datos o estructuras incluidas en una *ED*. Solo una de entre varias posibilidades será posible.

- **Iteración {}:** Define la repetición de un componente cero o más veces. Sirve para indicar la repetición de una cierta ocurrencia en una definición.
- **Opcional ():** Es un caso especial de iteración, es decir, una o ninguna iteración. Es aquel dato que puede o no formar parte de la composición de un dato compuesto.

Flujos de datos (*elementos*):

- **Nombre:** Nombres significativos.
- **Fuente:** Indica el proceso fuente de la información.
- **Destino:** Indica el proceso destino de la información.
- **Definición:** Explica el contenido del flujo de datos.
- **Contenido:** Describe cuales son las ED incluidas.

Almacenamientos de dato (*elementos*):

- **Nombre:** Nombres significativos.
- **Flujos entrada:** Flujos que alimentan el almacenamiento de datos.
- **Flujos de salida:** Flujos que extraen información del almacenamiento de datos.
- **Definición:** Describe el contenido del almacenamiento de datos.
- **Contenido:** Especifica el contenido del almacenamiento.

Procesos (*elementos*):

- **Nombre:** Nombres significativos.
- **Entradas:** Indica cuales son los procesos, almacenamientos de datos que ejercen de fuente de datos.
- **Salidas:** Indica cuales son los procesos, almacenamientos de datos que ejercen de destino de datos.
- **Definición:** Indica la misión del proceso.
- **Descripción:** Describe el proceso.

Entidades (*elementos*):

- **Nombre:** Nombres significativos.
- **Flujos de datos:** Indica cuales son los flujos (*E/S*) asociados.
- **Definición:** Indica quien es la entidad.
- **Volumen:** Número de componentes de la entidad.

11. Modelo E/R en Métrica V3

Proceso *EVS*:

EVS 2: Estudio de alternativas de solución.

EVS 2.3: Descripción de los subsistemas de información existentes.

Proceso *ASI*:

ASI 1: Definición del sistema.

ASI 1.1: Determinación del alcance del sistema.

ASI 6: Elaboración del modelo de datos.

ASI 6.1: Elaboración del modelo conceptual de datos.

ASI 6.2: Elaboración del modelo lógico de datos.

ASI 6.3: Normalización del modelo conceptual de datos.

Tema 9. DSI con TE. Modelado de comportamiento

1. Objetivos del modelo

Los objetivos del modelo son obtener **un registro de la secuencia de los cambios de las entidades en el tiempo, los requisitos de tratamiento, establecer estados y cambios de estado** de las entidades y poner de manifiesto las posibles **interacciones (comportamiento)**.

Disponemos de **dos tipos de modelo, y en consecuencia, dos técnicas**.

- **Modelo de eventos: HVE** (*Historia de la vida de la entidad*)
- **Modelo de estados: DTE** (*Diagrama de transición de estados*)

HVE se centra en los eventos y *DTE* en los estados.

2. Elementos del modelo

2.1 Entidades de datos

Cualquier objeto sobre el que el sistema guarda información. Se caracterizan por sus atributos. Se considera una *DTE/HVE* para cada entidad del sistema. Los efectos de los eventos de una entidad son:

- **Crear** o dar de alta la entidad en el sistema.
- **Modificar** características de la entidad (*sus atributos*).
- **Borrar** o dar de baja la entidad del sistema.

2.2 Eventos

Es cualquier **suceso que activa un proceso** que actualiza datos en el sistema. Tres tipos:

- **Exteriores al sistema.**
- **Periódicos, sin estímulo externo.**
- **Reconocidos internamente (*trigger*).**

Los eventos se asocian a las entidades.

2.3 Efectos

Describen **el resultado de la acción de un evento sobre una entidad**. Un evento puede tener **diferentes efectos sobre distintas entidades**. También puede tener **diferentes efectos sobre distintas ocurrencias de una misma entidad**.

2.4 Nodo

Se usan nodos como medio para agrupar un **conjunto de eventos** que afectan a una entidad.

2.5 Cajas vacías

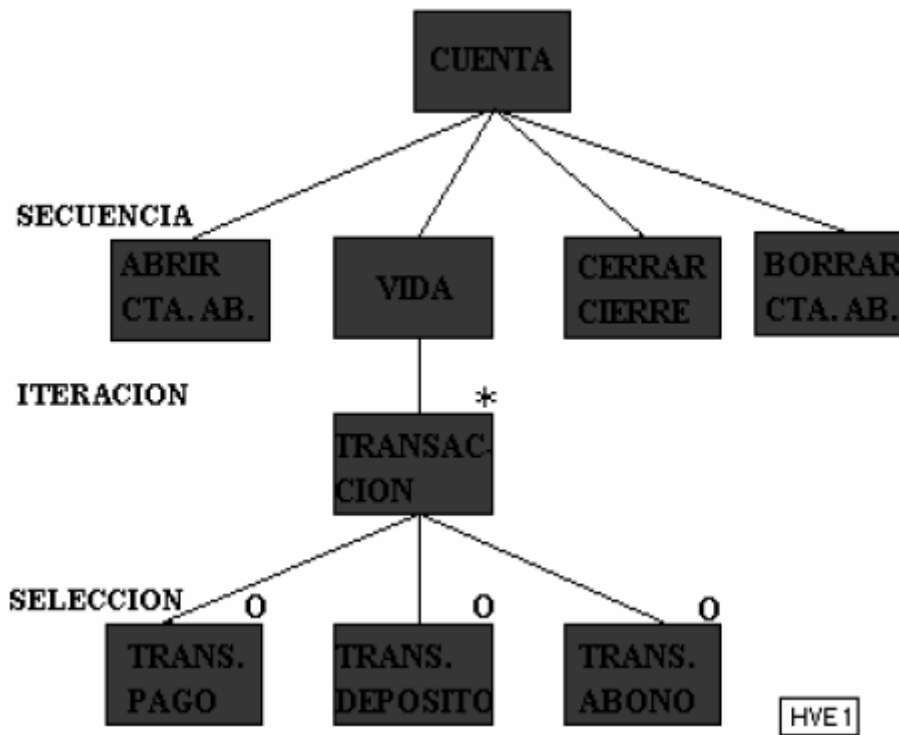
Representan el caso en que ningún evento afecta a la entidad.

3. Representación gráfica

3.1 HVE: Modelo de eventos

Muestra el **ciclo de vida** de una entidad desde su creación hasta su desaparición. Proporciona una **forma de modelar** todos los posibles **cambios** de valor de sus atributos durante la vida de la entidad y la **secuencia** en que dichas actualizaciones transcurren. Usa una notación de **diagrama de estructura** (*Diagrama de Jackson*).

Los **HVE** se representan de forma jerárquica. En la parte superior se comienza por la entidad cuyo ciclo de vida se va a representar.



Modelo de eventos

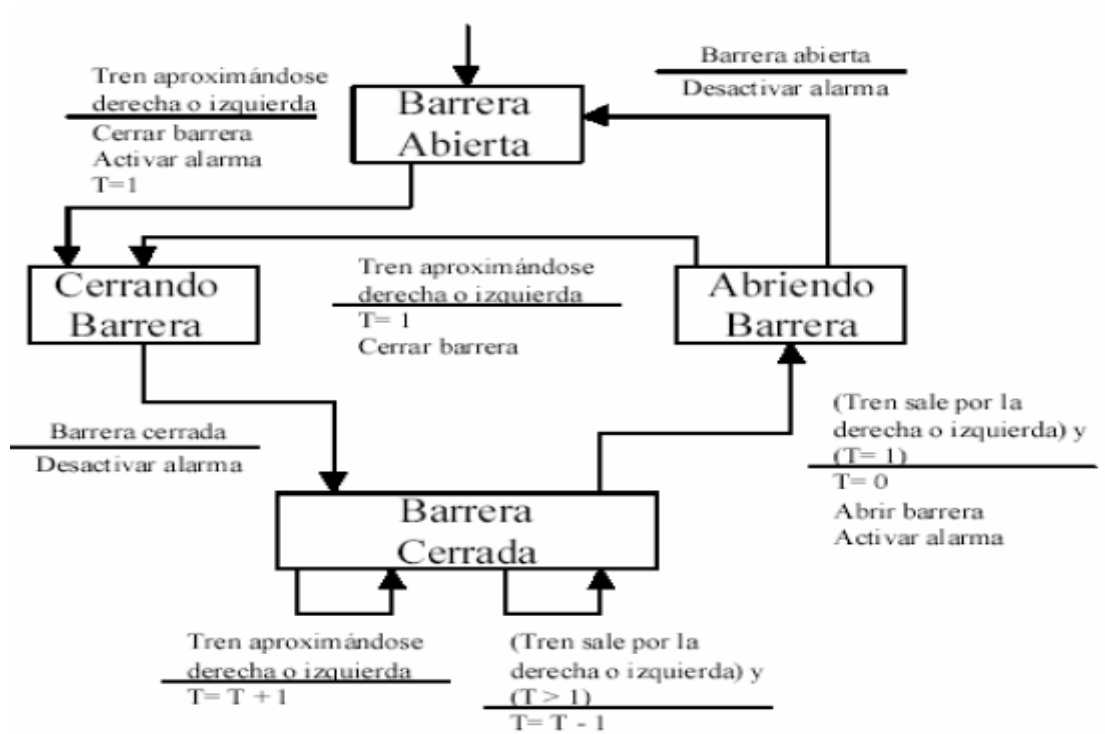
Las **estructuras paralelas** se usan en los **HVE** cuando los **eventos** pueden ocurrir de forma **simultánea** en el tiempo. Un **salto** se produce cuando un **evento** produce como resultado la alteración de la vida normal de la entidad de forma que esa entidad pueda **pasar a otro estado**.

Los **indicadores de estado** sirven para determinar el estado de una entidad antes y después de ser actualizada por un evento.

3.2 DTE: Modelo de estados

Es un diagrama que representa los estados que puede tomar un componente o sistema y que además muestra los eventos o circunstancias que implican el cambio de un estado a otro. Sus **componentes** son **estados** (*modo externo de comportamiento*) y la **transición** (*paso de un estado a otro si se cumple una condición*). Su **representación** se basa en **nodos** (*estados*) y **arcos** (*transiciones*).

El *DTE* representa la secuencia de estados y transiciones para **CADA INSTANCIA** de una entidad. Cada instancia puede estar en un estado distinto.



Ejemplo Notación DTE

4. Construcción de los modelos

Existen unos pasos generales a seguir para la construcción:

1. Identificación de los eventos
2. Construcción de la matriz entidad-evento
 - a. **Filas:** entidades
 - b. **Columnas:** eventos
 - c. **Intersecciones:**
 - i. **I:** Inserción
 - ii. **M:** Modificar
 - iii. **B:** Borrado
3. Construcción de *DTE/HVE* iniciales para todas las entidades
4. Refinamiento de las *DTE/HVE*
5. Adición de los indicadores de estado

5. Interrelación con otras técnicas

DTE se corresponde con **Vista evolutiva**.

DED se corresponde con **Vista estática**.

DFD se corresponde con **Vista dinámica**.

Se realizan ciertas **comprobaciones** como si existe un **proceso** dentro de los *DFD* del sistema que **trate cada uno de los eventos identificados** o que el modelo de de datos, representado por *DED* permita **reflejar las repercusiones que la actuación de un evento sobre una entidad tiene sobre las otra**.

Respecto a la **consistencia entre modelos...**

- Cada flujo y almacén de datos han de estar definidos en el *DD*
- Cada flujo y almacén de datos definidos en el *DD* deben aparecer en alguna parte del *DFD*.
- Cada proceso del *DFD* ha de estar asociado a un *DFD* de un nivel más bajo o relacionado con una especificación de proceso (*EP*)
- Cada *EP* debe estar asociado a un proceso de nivel mínimo.
- Entradas y salidas han de coincidir.

En la *EP*, cada referencia debe cumplir:

- Ha de ser el nombre de un flujo de datos o almacén.
- Ha de ser un término local
- Ha de aparecer en el *DD*

- Cada entrada en el *DD* ha de ser referenciada para una *EP*, *DFD* u otra entidad en el *DD*.
- Cada almacén en el *DFD* se ha de corresponder con una entidad (*tabla*) del *DED*.

Un almacén de datos corresponde a un conjunto completo de entidades de datos (una o más).

6. Modelo de comportamiento en Métrica V3

Proceso ASI

ASI 8: Definición de interfaces de usuario

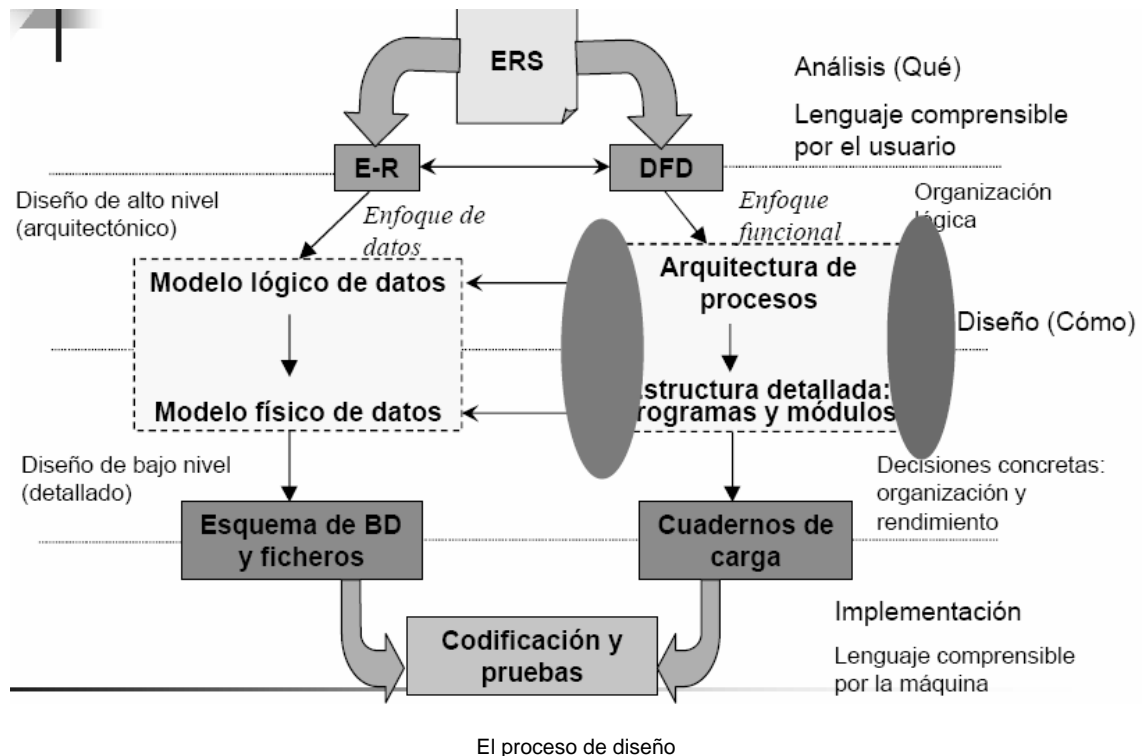
Proceso DSI

DSI 3: Diseño de casos de uso reales (*IU*).

DSI 5: Diseño de la arquitectura de módulos del sistema.

Tema 10. DSI con TE. Técnicas de diseño de procesos

1. El proceso de diseño



Existen cuatro diseños fundamentales:

- **Diseño de datos:** Transforma el modelo del dominio de la información del análisis en estructuras de datos.
- **Diseño arquitectónico:** Estructura modular de la aplicación.
- **Diseño de interfaz:** Interfaces del SW con otros sistemas y los usuarios.
- **Diseño procedimental:** Descripción procedimental de los componentes del SW.

2. Diseño estructurado

Su objetivo es desarrollar la **estructura modular** del programa y definir las **relaciones** entre módulos. Se usa el **diagrama de estructura** como técnica principal. La **documentación de entrada son DFD's + M E/R** y como estrategias de diseño se usan **análisis de transformaciones y transacciones**.

Las tareas a realizar son:

- Determinar **que módulos** implantaran los procesos terminales obtenidos en el análisis.
- Organizar la **estructura** de estos módulos y definir las **conexiones** entre los mismos.
- **Refinar** el DE aplicando cohesión, acoplamiento y guías adicionales de diseño.
- Describir el **pseudo código** para cada módulo.

Se basa en: abstracción, modularidad, encapsulamiento y ocultación de la información (No confundir con programación estructurada).

3. Estrategias de diseño

Pasos a seguir para obtener un buen diseño a partir de un *DFD* de procesos primitivos. Dos estrategias: **Análisis de transformaciones y de transacciones.**

Es importante **diseñar el DE de forma que:**

- Los módulos de **nivel superior** toman las decisiones de ejecución: **Jerarquía.**
- Los de **nivel inferior** realizan la mayor parte del trabajo de entrada, de cálculo y salida: **Especialización.**

Pasos:

1. **Revisar** el modelo fundamental del sistema.
2. Determinar si el *DFD* tiene características de **transformación o transacción.**
3. Según sea de **transformación o transacción:**
 - a. **Aislar el centro de la transformación.**
 - b. **Identificar el centro de la transacción.**
4. Realizar el primer corte del diagrama de estructuras.
5. Realizar el segundo nivel de factorización.
6. Refinar la estructura del programa.

4. Técnica del diagrama de estructura

Diseño arquitectónico del sistema: Diagrama de módulos funcionales

Hay que identificar **que módulos se necesitan** así como sus inputs/outputs. Refleja la **comunicación** de datos y control y la **jerarquía** entre módulos.

El **módulo** en el diagrama de estructura es aquella parte de código "**que se puede llamar**". Representa un programa, subprograma...

El diseño estructurado **NO impone** la restricción de que un módulo tenga que ser compilado independientemente. Su tamaño ideal podría estar entre 40 y 50 líneas.

La **representación de módulos** se hace mediante cajas, y estos se **conectan** mediante una línea. Si 'A' está por encima de 'B' unido por una línea significa que A llama a B. Las estructuras repetitivas se indican dibujando una flecha en forma de bucle, y las alternativas con un rombo.

El orden de ejecución es **de izquierda a derecha y de arriba abajo.**

Diferencias entre datos y flags/controles:

Datos	Flags/Controles
Los datos se procesan	Los controles solo sirven para comunicar condiciones entre módulos
Es la información compartida por los módulos.	Los controles indican al módulo que llama la terminación <i>EOF</i> o un error.
Tienen importancia para el mundo exterior.	Los flags tienen importancia en la comunicación de la información en el interior.

5. Documentación de la comunicación entre módulos

Es importante registrar la información referente a la comunicación entre módulos (módulo que llama, llamado, parámetros formales, si son de entrada o salida...)

Se suelen usar tablas de interfaz.

6. Análisis de transformación

Es un **conjunto de pasos** que permiten obtener, a partir de un *DFD* con *características de transformación*, la **estructura del diseño de alto nivel** del sistema.

Un *DFD* con **características de transformación** es aquél en el que se pueden distinguir:

- Flujo de entrada o llegada.
- Flujo de transformación o centro de transformación.
- Flujo de salida.

Los **datos** necesarios para el sistema lo recogen los módulos de las **ramas izquierdas**, siendo los datos intercambiados en esa rama **ascendentes**.

En las **centrales** habrá movimiento de información compartida (*ascendente y descendente*).

En las **ramas de la derecha** la información será de **salida** y por lo tanto **descendente**.

Pasos a realizar en el **análisis de transformación**.

1. **Identificar el centro de transformación.**
2. Realizar el **primer nivel de factorización** o descomposición del diagrama de estructura. Identificar tres módulos **subordinados** a un módulo de control del sistema.
 - a. Módulo controlador del proceso de información de entrada (*Ce*).
 - b. Módulo controlador del centro de transformación (*Ct*).
 - c. Módulo controlador del proceso de la información de salida (*Cs*).
3. Elaborar el **segundo nivel de factorización**.
4. **Revistar la estructura del sistema** usando medidas y guías de diseño.

7. Análisis de transacción

Se aplica cuando en un *DFD* **existe un proceso que en función del flujo de llegada, determina la elección de uno o más flujos de información**. Se llama **centro de transacción** al proceso desde el que parten los posibles caminos de información.

Pasos a realizar en el análisis:

1. Identificar el centro de transacción.
2. Transformar el *DFD* en la estructura (*primer corte*) adecuada al proceso de transacciones.
3. Factorizar la estructura de cada camino de acción.
4. Refinar la estructura.

8. Ejemplo

9. Ejemplo

10. Centros de transacción implícitos

Generalmente el esquema de transacción no es tan claro, por lo tanto: El proceso de transacción no aparece explícitamente en el *DFD*.

La solución es examinar el diagrama de contexto y lista de eventos para determinar los tipos de transacciones en el sistema.

11. Métodos para la especificación de módulos (Complemento tema 6.8)

- Interfaz-función.
- Pseudo código.
- Herramientas complementarias:
 - Diagramas de flujo
 - Nassi-Schneiderman
 - Tablas y árboles de decisión

12. Acoplamiento

Es el grado de interdependencia entre módulos. Depende de la forma en la que interactúen los módulos. El objetivo es minimizar el acoplamiento (*CAJA NEGRA*).

Ventajas de un bajo acoplamiento:

Menos oportunidades para el **efecto onda**.

Cambio hecho en un módulo **afecta lo menos posible a otros módulos**.

El acoplamiento implica complejidad de la interfaz.

Acoplamiento común:

Más de dos módulos hacen referencia a un área común de datos. Dos módulos pueden estar acoplados globalmente y no estar conectados mediante una llamada.

Acoplamiento de contenido:

Ocurre cuando un modulo necesita o accede a una parte del otro y rompe la jerarquía de funcionalidad de la estructura.

13. Cohesión

Es una medida de la **relación funcional** entre los elementos de un módulo. Un módulo coherente sólo debe hacer una cosa. El objetivo es tener **módulos con alta cohesión**.

Hay varios niveles de cohesión:

- **Funcional:** Se realiza una sola función.
- **Secuencial:** La salida de una tarea sirve como entrada a la siguiente.
- **Comunicacional:** Actividades que comparten datos.
- **Procedural:** Actividades diferentes donde la ejecución fluye de una a la siguiente.
- **Temporal:** Actividades diferentes pero relacionadas por el tiempo.
- **Lógica:** Actividades con la misma categoría lógica general.
- **Coincidental o casual:** Actividades diferentes sin relaciones significativas entre ellas.

14. Heurísticas de diseño estructural

IMPORTANTE: Los **módulos superiores** deben coordinar, y los **inferiores** realizar las tareas. Se debe **reducir el número de parámetros** que intercambian los módulos tanto como sea posible, **pasar estrictamente lo necesario**, y **no agrupar parámetros sin relación**.

Se deben **factorizar** funciones cuando sea posible e **inicializar las variables lo más tarde posible**.

15. Diagramas de estructura en Métrica V3

Proceso DSI:

DSI 1: Definición de la arquitectura del sistema.

DSI 1.5: Identificación de subsistemas de diseño.

DSI 2: Diseño de la arquitectura de soporte.

DSI 2.1: Diseño de subsistemas de soporte.

DSI 5: Diseño de la arquitectura de módulos del sistema.

DSI 5.1: Diseño de módulos del sistema.

DSI 5.2: Diseño de comunicaciones entre módulos.

DSI 8: Generación de especificaciones de construcción.

DSI 8.2: Definición de componentes y subsistemas de construcción.

Tema 11. DSI con TE. Técnicas de diseño de datos

1. Conceptos

Nota: Las técnicas aquí expuestas se pueden aplicar en *OOT*, simplemente considerando la entidad como una clase.

1.1 Base de datos

Es un conjunto, colección o depósito de datos almacenados en un soporte informático. Dichos datos deben estar **interrelacionados y estructurados** de acuerdo con un modelo capaz de recoger el máximo contenido semántico.

Una *BD* consiste en una colección de datos **persistentes e independientes** usados por una organización determinada. Los **sistemas de bases de datos**, al ser un repositorio, poseen componente estática, dinámica y funcional.

1.2 Sistema gestor de bases de datos

Es un **conjunto** de programas, procedimientos, lenguajes, etc. que permiten describir, manipular y usar los datos almacenados en la base manteniendo la **integridad, confidencialidad y seguridad**. Proporciona una **interfaz entre los datos** almacenados en la *BD* y los programas de aplicación. Incluyen:

- Lenguaje de descripción de datos (*DDL*).
- Lenguaje de manipulación de datos (*DML*).
- Lenguaje de control de datos (*DCL*).
- Un modelo de datos subyacente.

2. Conceptos DB2

2.1 Tablas

Son estructuras **lógicas** mantenidas por el *DB2*. Las filas de una tabla **relacional** no tienen un orden fijo. Las **columnas** tienen el orden especificado. En la intersección de una fila con una columna existe un valor. El número de **filas** de una tabla **puede ser cero** mientras que el de **columnas debe ser al menos 1**. *DB2* soporta estos tipos de tablas:

- **Tabla base:** Creada con *CREATE TABLE* y sus datos persisten.
- **Tabla temporal:** Almacena datos temporales, existe mientras dure el proceso.
- **Tabla resultado:** Contiene un conjunto de filas que *DB2* devuelve como resultado de una consulta.

2.2 Claves

Una clave está formada por **una o más columnas**. La misma columna puede ser parte de **más de una clave**. Tipos:

- **Clave única:** No hay valores repetidos. Se define con la palabra *UNIQUE*.
- **Clave primaria:** Tipo especial de clave única, no tiene valores únicos. Una clave primaria por tabla. (*Es opcional*).
- **Clave padre:** Es una clave primaria o única en una tabla padre.
- **Clave extranjera:** Es aquella clave de una tabla hija que se relaciona con la clave padre de la tabla padre.

2.3 Índices

Es un conjunto ordenado de direcciones a filas de una tabla. Estos mejoran el rendimiento de las búsquedas y aseguran ocurrencias únicas. Un índice es un objeto separado de los datos de la tabla. Una tabla tendrá un objeto de datos y tantos de índices como se hayan creado con *CREATE INDEX*. DB2 mantiene y construye los índices automáticamente. Son usados para *Valores únicos, Claves padre, Datos clustered, Particionar tablas, Suministrar rutas de acceso, etc.*

Una **clave primaria** es candidata para ser índice único. Un índice único no admite duplicados. En tablas pequeñas **no compensa** tener índices **no-únicos**. Respecto a los índices clustering, **siempre debe existir uno**.

2.4 Vistas

Es una visión lógica de datos de una o varias tablas. No existen físicamente. Una vista sirve para ocultar datos, controlar y delimitar acceso a los datos, simplificar autorizaciones de acceso, mostrar datos resumen o combinar tablas de diferentes formas.

3. Construcción del esquema interno o físico

Su objetivo es producir un **modelo de datos físico** que tenga en cuenta la estructura de ficheros del **entorno tecnológico** donde funcionará el sistema. El **esquema de entidades** obtenido en el *M E-R/DED* validado mediante teoría de **normalización servirá para definir las tablas físicas** de la *BD*.

Definir las claves, índices u otras posibles maneras de acceder físicamente a los datos, para cumplir todos los requisitos de **acceso** y de **relaciones**.

Definir las vistas o subesquemas de la *BD* para cada componente definido a partir del modelo de procesos.

Para asegurar un acceso eficiente a los datos hay que tener en cuenta las siguientes reglas de integridad:

- **Integridad de entidad:** La clave principal no puede ser nula. Si permitimos valor nulo podría existir más de un atributo con ese mismo valor, y el acceso no sería único.
- **Integridad referencial:** Si existe clave ajena, el valor ha de ser igual al atributo clave o bien ser nulo.

Puede ocurrir que al implantar las tablas **el rendimiento** no sea el esperado y habrá que proceder a realizar una **revisión de optimización**.

La integridad requiere el mecanismo de *locks*, sin embargo esto también es fuente de problemas, se debe buscar equilibrio entre **integridad y concurrencia**.

4. Estructura física de una BD DB2.

Una BD DB2 es un conjunto de estructuras de almacenamiento del tipo *TS (Table's Space's)* e *IS (Index's Space's)*.

Un **TS** es un fichero físico que contiene tablas. Un *TS* puede ser simple, segmentado o particionado.

Un **TS segmentado** son ficheros físicos divididos lógicamente en segmentos. Cada segmento contiene un grupo de páginas de una Tabla y solo de una y además contiene el mismo número de páginas, que ha de ser múltiplo de 4. El *TS* segmentado es el más apropiado para contener varias tablas pequeñas.

Ventajas de los TS Segmentados:

- Cuando el *DB2* busca una fila de una tabla solo busca en los segmentos de la tabla, no en todo el *TS*.
- Cuando el *DB2* bloque una tabla no afecta al resto de segmentos del *TS*.
- Cuando se borra una tabla, los segmentos pueden rehusarse de forma inmediata.
- Cuando se borran filas de una tabla y quedan segmentos vacíos, se pueden rehusar por otras tablas.
- Suelen ser más eficientes que las particionadas, en especial al producirse un *TS Scan*.

Un **TS particionado** es casi lo opuesto al segmentado. En este caso el *TS* no es un fichero físico sino un conjunto de ficheros físicos, porque la tabla está almacenada en varias particiones y cada una es un fichero físico. Es una estructura ideal para *tablas enormes*. El *TS* particionado contiene una sola tabla y es un **conjunto de TS en los que la tabla se particiona**.

Ventajas de los TS particionados

- Conviene usar una clase que permita balancear las filas sin las diferentes particiones.
- Las diferentes particiones pueden pertenecer a diferentes *Storage Group's*.
- Se puede trabajar concurrentemente en las diferentes particiones con diferentes programas.

Un *TS* se crea de forma implícita cuando al crear una tabla no se indica su *TS*. Se puede crear de forma explícita con la secuencia *SQL: CREATE TABLESPACE*

Cuando se crea un índice, automáticamente se crea para él un Index Space en la BD de la tabla.

5. Reglas de transformación del modelo lógico en físico

El objetivo es obtener un modelo físico de datos a partir del lógico. Se necesita aplicar unas reglas que **conserven la semántica** del modelo lógico.

5.1 Transformación de entidades

Toda **entidad** se convierte en una **tabla** o relación que toma su nombre.

5.2 Transformación de atributos de entidades

Los **atributos** de la entidad serán las **columnas** de las tablas y el **atributo(s) identificador** principal será la **clave primaria**.

5.3 Transformación de relaciones

Según el tipo de correspondencia:

5.3.1 Relaciones 1:N: Se propaga el identificador de la entidad de cardinalidad 1 a la que es N teniendo en cuenta que:

- Si la **relación es de asociación**, la clave propagada es la **clave ajena** en la tabla donde se propagó.
- Si la **relación es dependencia**, la clave primaria de la tabla correspondiente a la entidad débil está formada por la **concatenación** de los identificadores de ambas identidades.
- Las **relaciones de agregación** se transforman del mismo modo que las 1:N.

5.3.2: Relaciones 1:1: Es un caso particular de las 1:N y por tanto se propaga la clave en las dos direcciones o a la más débil.

5.3.3: Relaciones N:M: Se transforman en **una tabla** cuya **clave primaria** será la **concatenación** de los atributos principales de las entidades que asocia.

6. Optimización

El objetivo es reestructurar el modelo físico de datos para asegurar que satisface los requisitos de rendimiento. Técnicas:

- Introducir elementos redundantes.
- **Dividir** entidades.
- **Combinar** entidades si los accesos son frecuentes en la misma transacción.
- **Redefinir** o **añadir** relaciones entre entidades para hacer más directo su acceso.
- Definir **claves secundarias** o índices para permitir caminos de acceso alternativos.

Para decidir si se debe desnormalizar o no se debe analizar **los tiempos** de respuesta requeridos, **la tasa** de actualizaciones, **las veces** que se accede **conjuntamente** a los atributos, **la longitud** de los mismos, **frecuencia y prioridad** de acceso, **tamaño** de las tablas y los requisitos de **seguridad**. **DESNORMALIZARÁ** el administrador de la **BD**.

Para **gestionar el rendimiento** del **DB2** hay que conocer el funcionamiento de sus componentes: **codificación de sentencias SQL**, **diseño de índices**, **organización de tablas**, **como son los TS**, **nivel de concurrencia y locking**, **los BP**, **los commit...**

Ante procesos lentos **primero fijarse en los recursos del sistema**. Si no hay sobrecarga ahí, entonces ir a los recursos del **DB2**.

Primero analizar **el uso del BP**, fijarse si la aplicación ha de **esperar** para escribir en los discos y fijarse si las aplicaciones **consumen más** recursos de lo habitual.

6.1 El locking

Sensibilidad a nivel de fila, página. Tabla, **TS**.

Cuando una aplicación modifica un dato el **DB2** adquiere un *lock* para prevenir que otras actúen sobre ese dato y se pierda la integridad. Cuando la aplicación que causo el *lock* finaliza su trabajo (*commit*) el **DB2** libera el bloque para que los siguientes puedan operar.

Si el *lock* no se resuelve, para los que esperan se producirá un *deadlock*. Si el que está haciendo algo no consigue finalizarlo, se producirá un *rollback*.

6.2 Los BP's

Son **áreas de memoria virtual** que de forma temporal almacén páginas de datos extraídas de **TS** o índices. Son un elemento clave en el rendimiento del **DB2** dado que es mucho más rápido buscar una página en ellos que en disco.

DB2 permite usar hasta 50 **BP** que contiene páginas de 4 **KB** y hasta 10 **BP** que contengan páginas de 8, 16 y 32 **KB**.

El número de páginas de un **BP** depende de su tamaño.

Los datos permanecen en el **BP** hasta que el **DB2** decide usar el espacio para otra página. Los **BP** pueden almacenarse en *Memoria Principal*, *Data Spaces* o *HiperSpaces*.

6.3 Compresión de datos

La compresión de datos de un *TS* puede reducir el disco necesario para almacenar datos y mejorar el rendimiento del *BP*. Esta técnica es eficiente con tablas grandes. La herramienta *DSN1COMP* nos puede ayudar a determinar si comprimir o no los datos. *CREATE TABLESPACE* tiene una cláusula *COMPRESS*.

6.4 Cuando es necesario reorganizar

- **Reor-pending status:** Cuando los *TS* o particiones están en ese estado, no hay alternativa. No se puede acceder a los datos.
- **Datos desvirtualizados (*skewed*):** Puede ocurrir con tablas particionadas cuando las particiones no están bien balanceadas.
- **Datos desorganizados o fragmentados:** Si ocurre esto puede ser necesario reorganizar *TS* y/o *IS*.

6.5 Programando con SQL

El rendimiento de una consulta depende de la ruta de acceso a los datos. Puede ser indexada o secuencial.

7. Caminos de acceso y cálculo de errores

Son técnicas que tienen por objetivo **verificar** que un **modelo de datos soporta** realizar determinadas consultas. Se puede usar en **análisis y diseño**, en diseño se usa para optimizar.

7.1 Caminos de acceso

Su objetivo es **analizar la secuencia de acceso** a los datos que realizan los módulos a través del modelo de datos.

7.2 Cálculos de acceso

El objetivo es **estimar el número de accesos** aproximado que ha de realizarse para obtener información de cada consulta, tomando como **referencia las vistas** del modelo de datos obtenidas en los análisis de los **caminos de acceso**.

8. Las técnicas de diseño de datos en Métrica V3:

Proceso DSI

DSI 6: Diseño físico de datos.

- **DSI 6.1:** Diseño del modelo físico de datos (*Se aplican reglas de transformación a entidades o clases*).
- **DSI 6.2:** Especificación de los caminos de acceso a los datos (*Se aplican técnicas de caminos de acceso y cálculo de acceso a los datos*).
- **DSI 6.3:** Optimización del modelo físico de datos (*Se aplican técnicas de optimización*).

Tema 12. DSI con TE. Técnicas de diseño de presentación

1. El modelo Vista

Su objetivo es **identificar y diseñar la parte visual** de la aplicación. Este modelo estará integrado por todas las **interfaces entre el sistema y usuario** (*pantallas, diálogos, etc.*).

Se debe dar más importancia a los **requisitos de usuario** que las definiciones técnicas. La **participación activa de los usuarios** es imprescindible.

Este modelo ofrece una visión más amplia del sistema creando simulaciones de comportamiento (**prototipos**) usando datos específicos. Ofrece unas ventajas:

- **Reducir el riesgo** en la implantación del nuevo sistema.
- **Aumentar la calidad** del producto.
- **Incrementar la productividad**.
- **Permitir** la introducción de **mejoras y modificaciones** antes de presentar los diseños definitivos.

Si no fuera factible realizar un **prototipo completo** se diseñará un **prototipo parcial**.

Una interfaz interactiva es dominada por los **aspectos dinámicos** del modelo, que son encargados de especificar el **protocolo de interacción** con el usuario. La interfaz constituye uno de los **subsistemas más importantes** y no puede ser obviado en el análisis y diseño del sistema. La interfaz de usuario debe estar **minimamente acoplada** con los objetos del dominio.

2. Método de construcción del modelo

Las actividades a realizar son:

1. Identificar las **unidades** de interface con el usuario.
2. Identificar las **relaciones**, en especial entre las unidades tipo pantalla.
3. Identificar las **secuencias** entre las diferentes unidades.
4. Identificar el **contenido**.
5. Diseñar los **formatos**.
6. Diseño del diagrama de **flujo de presentación**.
7. Diseño de la **comunicación** con la capa del dominio.
8. Construcción de **prototipos**.
9. Prueba.
10. Refinamiento.

2.1 Metodología estructurada

Para construir el modelo de presentación nos dirigimos por:

- **Modelo de datos:** Dentro de este por el **M E/R** para identificar los componentes **ventanas** o **modelo estático** y por el **Modelo de comportamiento** para identificar los **estados base** de cada ventana y sucesos que los determinan.
- **Modelo de procesos:** En sus interfaces con las entidades externas, para determinar el flujo o **modelo dinámico**.

Reglas:

- A toda entidad le corresponde al menos una ventana.
- En una relación 1:N, a la entidad del lado N le corresponderá además una ventana lista de entidades.
- Una ventana tendrá como atributos al menos los de la entidad origen.
- Cada atributo tendrá una posición en la ventana.
- Una ventana tendrá como estados al menos los de la entidad origen.
- A cada estado le corresponderá una o varias acciones de transición que cruzaremos con el *DTE*.
- Cada estado tendrá vinculado uno o varios mensajes.
- Una ventana tendrá como eventos al menos los que actúan en el *DTE* de su entidad origen.

Es útil tomar como referencia el *DFD* y el Modelo de comportamiento (*DTE* y *HVE*) como referente básico para construir el *DFP* (*Diagrama de flujo de la presentación*).

2.2 Metodología OO

El modelo dinámico estará integrado por diálogos. Una clase diálogo se corresponderá con un caso de uso. Las instancias de dialogo serán cada uno de sus escenarios.

El diálogo estará constituido por el conjunto de interacciones del sistema con el actor o actores del escenario.

Las tareas a realizar por cada instancia del dialogo o escenario son:

1. **Detallar los formatos de presentación de la interfaz.**
2. **Revisar el escenario normal**
3. **Revisar el escenario con excepciones.**
4. **Actualizar los diagramas de interacción.**
5. Actualizar y construir nuevos diagramas de estado.

3. Técnicas de diseño

Se pueden usar las notaciones gráficas de flujo, evento, proceso y ventana.

Una ventana al igual que una entidad estará integrada por atributos y estados además de texto fijo o imágenes. En cada estado, una ventana será sensible a unos u otros estímulos (eventos) que el usuario disparará mediante teclado o ratón.

NOTA: De este apartado existe mucha información adicional que he considerado no necesaria.

4. Prototipo visual

Realizar un prototipo dinámico que permita la simulación de los diálogos considerados como críticos podrá ayudar a establecer la descripción de los procesos del modelo de procesos y de las entidades del modelo de datos o en el caso de la *OTT* el correspondiente modelo de objetos.

Hay ciertas recomendaciones a seguir en la realización de este prototipo como:

- Determinar el punto de entrada a cada pantalla.
- Especificar datos asociados a cada pantalla.
- Evaluar la consistencia del diseño.
- Confirmar con el usuario la validez de los diseños de pantalla.

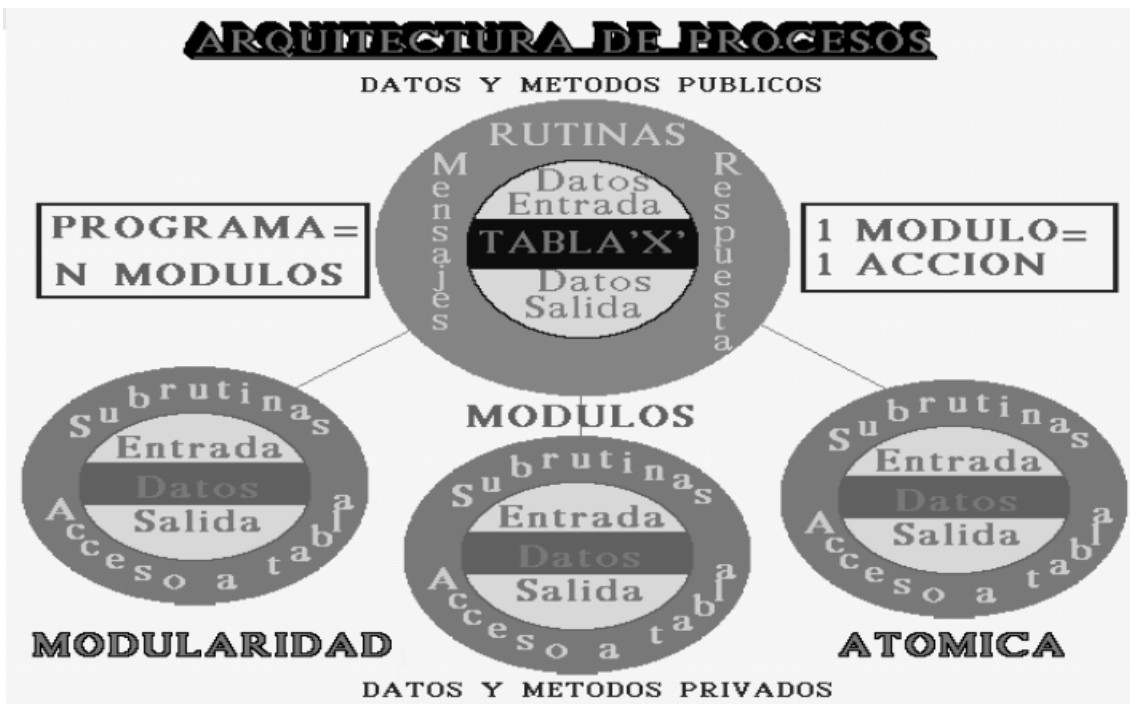
Los **productos a obtener** son formatos de pantallas, representación jerárquica de las pantallas del sistema, asignación de pantallas y diálogos a grupos de usuarios, diálogos críticos, etc.

5. Acoplamiento de los modelos

CICLO DE FABRICACION				
MODELOS	ANALISIS	DISEÑO	PROGRAM.	TEST
DATOS	*IDENTIFICAR CLASES ESTADOS SECUENCIA ATRIBUTOS *DIAGRAMA E-R	*DISEÑAR TABLAS INDICES LISTAS ESTRUCT. de Mens/ Resp.	Procedimientos: Definición Carga Copia Recuperación SQL+Explain	✓ ✓ ✓ ✓
PROCESOS	*IDENTIFICAR MACRO-ACC ACCIONES *VINCULAR ACC->ENT.	*Modularidad atómica. *Programas: 1 tabla(N A) n tablas:1xF	* Modular e independ. * Integración en Programa	Plataforma gral * Unit.
PRESENT.	*ENTIDAD: CLASE OOUI *IDENTIFICAR OOUI *DIAGRAMA O-R *IDENTIFICAR Subsistemas	*DISEÑO E/I Plataforma OOUI M. At. MENS/RES. Prototipo del Subsist.	*Externa *Lógica Estado *Mensajes y Respuestas *Clases de acciones.	*Plataf. *OOUI *OOUI+ Modulo *Macro *Subs.

Ciclo de fabricación

6. Modularidad atómica



Arquitectura de procesos

7. Reglas de usabilidad

10 reglas heurísticas de usabilidad:

1. **Visibilidad del estado del sistema.**
2. **Utilizar el lenguaje de los usuarios.**
3. **Control y libertad para el usuario.**
4. **Consistencia y estándares.**
5. **Prevención de errores.**
6. **Minimizar la carga de memoria del usuario.**
7. **Flexibilidad y eficiencia de uso.**
8. **Los diálogos estéticos y diseño minimalista.**
9. **Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de los errores.**
10. **Ayuda y documentación.**

8. El modelo vista en Métrica V3

Proceso ASI:

ASI 8: Definición de interfaces de usuario.

Proceso DSI:

DSI 3: Diseño de casos de uso reales.

DSI 3.3: Revisión de la interfaz de usuario.

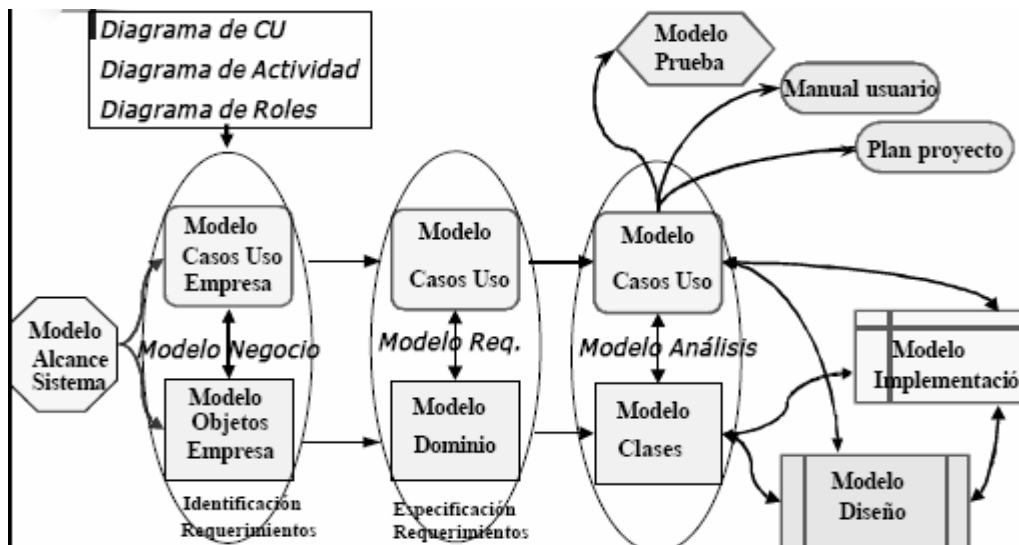
DSI 5: Diseño de la arquitectura de módulos del sistema.

DSI 5.3: Revisión de la interfaz de usuario igual que la 3.3 pero ahora aplicado en desarrollo estructurado.

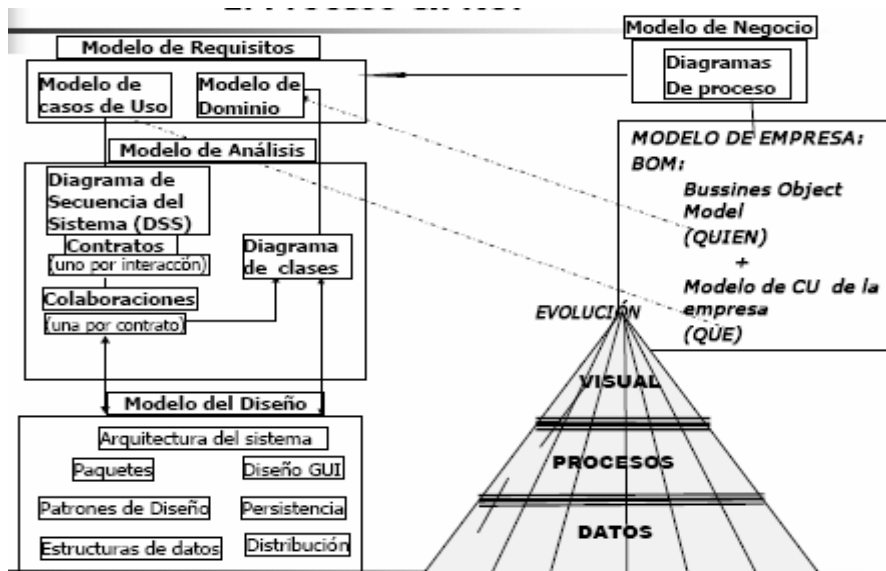
Tema 13. DSI con TOO. Modelado de requerimientos

1. Etapas del proceso (Aplicable a Métrica V3 y RUP)

- **Inicio:** Panificar proyecto (*pre*)
 - **Comprender procesos del negocio:** Modelo del negocio.
 - **Obtener y especificar requisitos:** Modelo de requisitos.
 - **Identificar y especificar clases y colaboraciones para objetos del dominio (*análisis*):** Modelo de análisis.
 - **Resolver problemas de diseño (*arquitectura, BD, etc...*):** Modelo de diseño.
- **Implementación:** Modelo de implementación.
- **Validación:** Modelo de validación.



Etapas del proceso



El proceso RUP

2. Etapa inicial (fase 1 en RUP)

Comprende el **estudio de necesidades de la empresa** (*ver si es viable, alternativas, análisis de riesgos*)... la definición de **objetivos** del proyecto, la estimación aproximada del **coste**, y la **duración** de la etapa inicial.

Realización de primeros talleres de **requisitos**, así como **casos de uso** escritos brevemente, e identificación de **riesgos**.

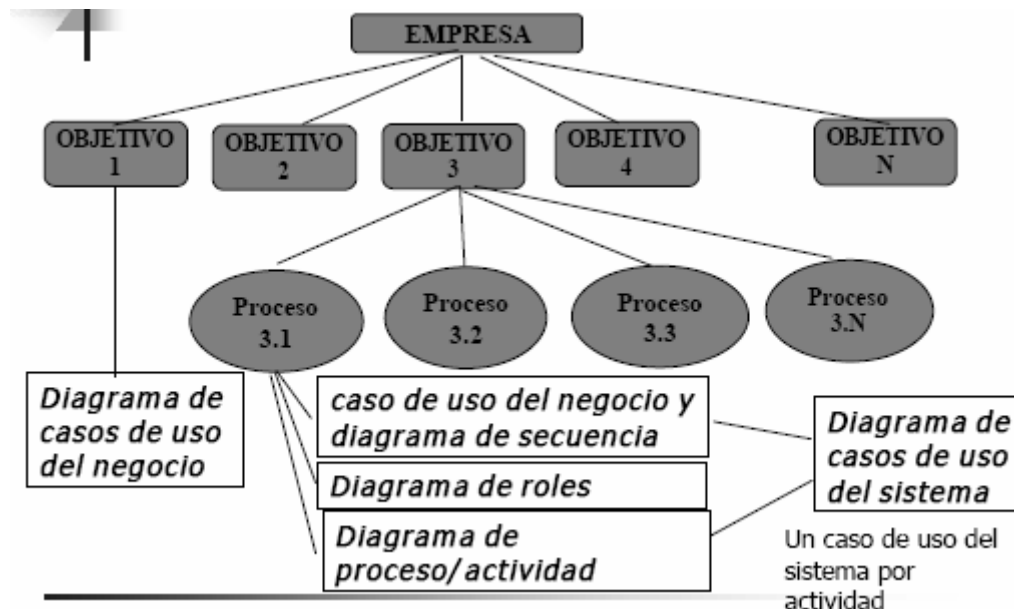
3. Modelado de negocio (primera disciplina en RUP)

Su objetivo es comprender los **procesos de negocio** que tienen lugar dentro de la empresa, así como la **estructura** de la organización y los **problemas actuales** y posibles mejoras.

3.1 Procesos de negocio

La organización tiene una serie de **objetivos** que satisface a través de **procesos de negocio**. Sus elementos son: *Flujo de tareas, agentes, información y reglas de negocio (estas regulan el funcionamiento de la empresa y describen restricciones y comportamientos aunque NO son requisitos pero influyen en ellos)*.

Las reglas del negocio determinan políticas y estructura de la información.



Estructura del modelo

3.2 Etapas del modelado de negocio

Identificar y definir los **procesos de negocio** según los objetivos.

Para cada objetivo habrá que realizar un **conjunto de procesos del negocio o subobjetivos**.

Cada **proceso de negocio** se realiza mediante un **caso de uso del negocio**.

Se debe definir un **caso de uso del negocio** para cada proceso del negocio (*diagrama de casos de uso del negocio*).

Se deben identificar los **roles** implicados en los diferentes procesos del negocio (*diagrama de roles*).

Por cada objetivo o subobjetivo haremos un **Diagrama de casos de uso del negocio**.

Se modela el **flujo de tareas asociado a cada proceso de negocio** mediante **escenarios (diagramas de secuencia)**, **diagramas de procesos (diagramas de actividades)** y **diagramas de roles**.

3.2.1 Casos de uso del negocio

Descripción: Existen dos opciones, **textual** y **visual** destacando de esta:

- **Diagrama de roles:** Aspecto estructural de colaboración entre roles.
- **Escenario:** Aspecto de comportamiento de la colaboración.
- **Diagrama de proceso:** Workflow que realiza el caso de uso del negocio.

3.2.2 Roles y escenarios

Identificamos los **agentes** o **roles** participantes (*Diagrama de roles*).

Creamos los **escenarios** para mostrar la colaboración entre agentes, distinguimos flujos básicos y alternativos (*Diagramas de secuencia, objetos son roles*).

3.2.3 Diagrama de proceso/actividad

Un proceso es un Workflow de tareas/actividades donde se aplican reglas de negocio. Se muestra el flujo del proceso mediante **diagramas de proceso**. Si la actividad necesita ser descrita mediante otro diagrama de actividad usaremos *objetivos y subobjetivos*. **Pueden existir procesos de negocio que no requieran interacciones entre agentes.**

Sobre las reglas de negocio, distinguimos dos tipos:

- **Reglas de restricción:** Especifican condiciones que **restringen la estructura y comportamiento** de las informaciones.
- **Reglas de derivación:** Especifican condiciones para **inferir nuevos hechos** a partir de otros.

4. Modelado de requisitos

El objetivo es establecer los **requisitos funcionales y no funcionales del sistema**. Los no funcionales son: *Usabilidad, Fiabilidad, Rendimiento, Adaptabilidad, Mantenimiento, Configurable, Implementación: lenguajes, herramientas, GUI, Legales*.

A partir del modelo de negocio (*si se hace*) se construye el **modelo de casos de uso** y el **modelo conceptual/dominio**. A partir de las actividades que aparecen en los diagramas de actividades se extraen los casos de uso del sistema y se establece el **modelo conceptual o del dominio** a partir de las informaciones incluidas en los diagramas de actividades.

En este modelado **se define un actor** para cada rol del diagrama de proceso que interacciona con el sistema, llamado **actor primario** y se crea **un caso de uso por cada actividad soportada por el sistema**.

4.1 Modelo conceptual/dominio

El **modelo de casos de uso** y el **conceptual** conforman el **modelo de requisitos**.

Este modelo representa **el vocabulario** del dominio. No incluyen operaciones, **solo atributos**, y existen **asociaciones** entre clases.

4.2 Modelo de casos de uso

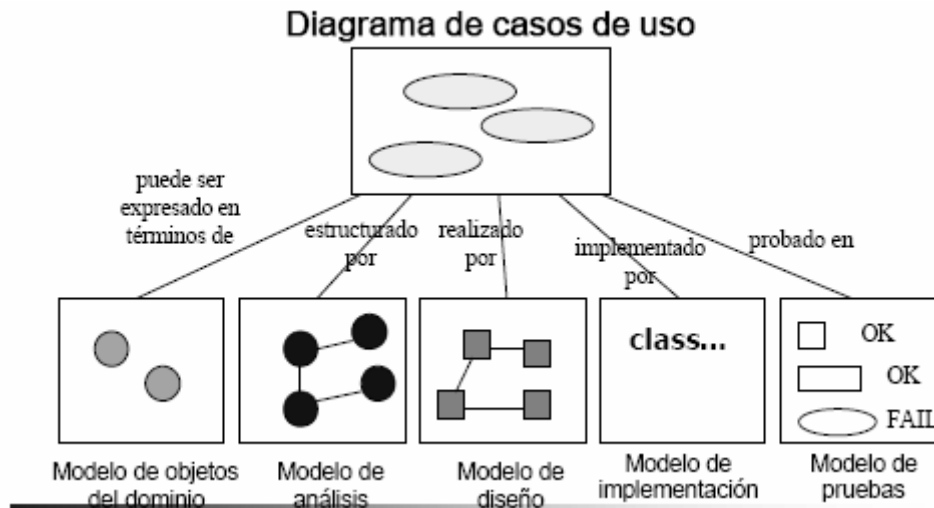


Diagrama de casos de uso

5. Casos de uso

Es una **técnica** de recolección y especificación de requisitos. Son **fáciles** de comprender/validar por los usuarios y **guían** todo el proceso de desarrollo además de **ayudar a la planificación/desarrollo incremental**. Describen **QUE hace el sistema, no COMO**. Es una técnica para registrar **requerimientos funcionales**.

Los casos de uso permiten definir los **límites** del sistema y las **relaciones** entre el sistema y el entorno. Son descripciones de la **funcionalidad del sistema** y comparándolo con los *DFD's* podrían definirse como los Diagramas de flujo de datos del enfoque estructurado.

Además, los casos de uso **particional** el conjunto de necesidades atendiendo a la **categoría de usuarios** del sistema y están basados en el **lenguaje natural**.

6. Diagramas de casos de uso

Sus elementos son:

- **Actores:** Roles que juegan los usuarios con respecto al sistema.
- **Casos de uso:** Interacciones típicas entre usuarios y el sistema.

6.1 Actores. Características

Un actor representa un **conjunto coherente de roles**. **No forman** parte del sistema e **inician** la ejecución de los casos de uso. **Un usuario** puede jugar diferentes roles y pueden intervenir **diferentes actores**, y **un actor** puede intervenir en varios casos de uso.

Los casos de uso son iniciados por un actor con un **objetivo**. Puede incluir secuencias **alternativas** que llevan al **éxito o fracaso**.

El conjunto **completo** de casos de uso especifica todas las posibles formas de usar el sistema, esto es el **comportamiento requerido**.

6.2 Casos de uso. Características

Capturan una **función visible** para el usuario. Por cada caso de uso puede haber un **camino básico** o **caminos alternativos**.

6.3 Relaciones entre casos de uso

Extends: El caso de uso que extiende realiza una acción en un punto del caso de uso extendido, si se cumple una condición. Denota un "punto de extensión".

Un **CDU (Caso de uso) base** incorpora implícitamente el comportamiento de otro **CDU** en el lugar especificado indirectamente por este otro **CDU**.

Sirve para representar las **condiciones de error y poco usuales** que podrían oscurecer el caso de uso base.

Uses: Se factorizan acciones que se usan en más de un caso de uso.

Un **CDU base** incorpora explícitamente el comportamiento de otro en algún lugar de su secuencia.

En UML

- también existe la generalización
- "uses" se denota <<include>>
- "extends" se denota <<extend>>



Relaciones entre casos de uso

6.4 Casos de uso: Construcción

Un caso de uso debe ser simple, inteligible, claro y conciso. **Generalmente** hay pocos actores asociados a cada caso de uso. Existen ciertas preguntas clave como ¿cuales son **las tareas** del actor?, o ¿**que información** crea, guarda, etc. el actor?

La descripción del caso de uso comprende:

- **El inicio:** ¿Cuándo y que actor lo produce?
- **El fin:** ¿Cuándo se produce y que valor devuelve?
- **La interacción** actor-caso de uso: ¿Que mensajes intercambian ambos?
- **Objetivo** del caso de uso: ¿Que lleva a cabo o intenta?
- **Cronología** y origen de las interacciones
- **Repeticiones** de comportamiento: ¿Que operaciones son iteradas?
- Situaciones **opcionales:** ¿Que ejecuciones alternativas se presentan en el caso de uso?
- Situaciones **excepcionales.**

7. Escenarios

Interacción típica entre el usuario y el sistema.

Tiene distintas acepciones:

- Escenario = Caso de uso (a veces)
- En *UML*, escenario = *camino dentro de un caso de uso*.

Un caso de uso describe un conjunto de secuencias de interacciones o **escenarios**: flujos principales y alternativos o excepcionales.

Un escenario es una instancia de un caso de uso.

7.1 Especificación de requisitos

El *SRS* (*Software Requirements Specification*) puede estar formado por:

- Diagrama de casos de uso
- Modelo del dominio

8. Casos de uso en Métrica V3

Proceso *EVS*

EVS 4: Estudio de alternativas de solución

EVS 4.2: Descripción de alternativas de solución (*Se describe el modelo de negocio*).

EVS 6: Selección de la solución

EVS 6.2: Evaluación de las alternativas de solución (*Incluye los modelos de negocio de cada una*).

Tema 14. DSI con TOO. Modelado Estructural

1. Introducción al modelado estructural

Se describen **los tipos de objetos** de un sistema y las **relaciones estáticas** que existen entre ellos. Se expresa mediante los **diagramas de clase** los cuales normalmente contienen **clases, interfaces, y relaciones de dependencia, realización, generalización y asociación**.

Los diagramas de clase se usan para modelar el **vocabulario del sistema, colaboraciones y esquema lógico de base de datos**.

Fundamentalmente se usan para reflejar aspectos **estructurales** del sistema.

1.1 Uso en métrica

Proceso EVS:

EVS 2.3: Descripción de los SI existentes: Diagrama de clases.

EVS 4.2: Descripción de alternativas: Modelo de negocio/dominio.

Proceso ASI:

En el proceso **ASI** se usan para estudiar y documentar la estructura de clases del sistema (*sobre todo para ASI 5*).

Proceso DSI:

En el proceso **DSI** se usan para obtener un diagrama de clases de diseño a partir del diagrama de análisis (*DSI 4*).

2. Clases

2.1 Notación

Una **Clase** es una descripción de un conjunto de objetos que comparten los mismos **atributos, operaciones, relaciones y semántica**.

2.2 Atributos

Derivados: Son aquellos que se obtienen como consecuencia de cálculos entre otros atributos de la clase. Se les antepone el carácter "/".

Comunes a todos los objetos: Se subraya su nombre.

2.3 Responsabilidades/Operaciones/Métodos

Es el servicio que puede realizar una instancia de una clase. Tiene:

- **Nombre:** Nombre de la operación:
- **Lista de parámetros:** Lista de parámetros del método separados por comas.
- **Tipo de retorno:** El tipo del valor devuelto por el método.
- **Propiedades:** *{isQuery}, {sequential}, {concurrent}*

Para indicar que un **método es abstracto** se usa el valor etiquetado *abstract=true*.

2.4 Encapsulación

Presenta tres ventajas básicas:

- Se **protegen** los datos de accesos indebidos.
- El **acoplamiento** entre las clases se disminuye.
- Favorece la **modularidad** y el mantenimiento.

Existen ciertos mecanismos como son establecer los atributos como **privados, protegidos y públicos**.

2.5 Clases parametrizadas y otras

No son realmente clases, son el resultado de generalizar clases total o parcialmente. Actúan como una función que recibe clases y devuelve clases.

Otras propiedades:

- **Metaclases:** Sus instancias son clases en tiempo de ejecución.
- **Plantillas:** Sus instancias son clases en tiempo de compilación.
- **Clase abstracta:** Se indica usando la propiedad `{abstract}`. No puede instanciarse.

3. Diagramas de clase

Existen diferentes perspectivas. **Conceptualmente** se representan los conceptos del **dominio** estudiado. Hablando de **especificación** se representan los tipos que representa una interface la cual puede tener cualquier implementación. Y hablando de **implementación** se representan las clases tal y como serán implementadas.

3.1 Relación dependencia

Un cambio en la especificación de un elemento (*destino*) puede afectar a otro (*origen*).

UML dispone de una serie de dependencias predefinidas: *bind, friend, refine, use, import, extend, include..*

3.2 Generalización

Es un tipo de..

La especialización es una técnica muy eficaz para la extensión y reutilización.

3.2.1 Disjunta - No disjunta (*Solapada*)

Determina si un objeto puede ser o no a la vez instancia de dos o más subclases en ese nivel de herencia.

Se define el **overlapping** cuando una **nueva clase** puede ser subclase de más de una subclase y **una instancia** puede ser instancia directa o indirecta de dos o más subclases.

3.2.2 Total (*Completa*) - Parcial (*Incompleta*)

Determina si todas las instancias de la clase padre son a la vez instancias de alguna de las clases hija.

Se define el **disjoint** cuando una nueva clase no puede ser subclase de más de una subclase. Instancias de una única clase.

3.3 Asociación

A nivel **conceptual** muestra la relación conceptual entre dos clases. A nivel de **especificación** representa operaciones y detectamos los mensajes del protocolo de una clase con respecto a la otra, mientras que hablando de **implementación** establece atributos (*navegabilidad*).

3.4 Agregación

En *UML* solo se distingue entre **agregación y composición** siendo esta última disjunta y estricta.

La **composición** es un caso particular de agregación: **exclusiva y dependiente**.

4. Clases asociación

Una clase asociación añade una restricción:

Solo puede existir una instancia de la asociación entre cualquier par de objetos participantes.

En las **asociaciones derivadas** un elemento derivado es aquel que se obtiene a partir de otro. Este elemento puede ser un atributo o una relación. Se expresa anteponiendo un "/" a su nombre.

5. Clases abstractas e interfaces

Un **interface** es una especificación del conjunto de operaciones de una clase o paquete que son visibles desde otras clases o paquetes.

Se define como **clase abstracta** aquella que no se puede instanciar de forma directa.

6. Polimorfismo

El termino polimorfismo se refiere a que **una característica de una clase puede tomar varias formas**. En nuestro caso representa la posibilidad de desencadenar operaciones distintas en respuesta a un mismo mensaje. Cada subclase hereda las operaciones pero tiene la posibilidad de modificar localmente el comportamiento de estas operaciones. La búsqueda automática del código que en cada momento se va a ejecutar es fruto del enlace dinámico.

7. Paquetes

Son un elemento organizativo que permite dividir un modelo en partes manejables mediante la agrupación de clases u otros paquetes. Puede contener elementos de cualquier tipo, y un elemento es exclusivo a un paquete. Existe la posibilidad de anidar paquetes.

Importación/Exportación en paquetes:

Los paquetes permiten controlar la complejidad del manejo de un gran número de abstracciones, controlando los accesos mediante la importación.

8. Sistema, modelo, vista, diagrama

- Un **sistema** es aquello que se está desarrollando y para lo que se crean modelos.
- Un **subsistema** es parte de un sistema.
- Un **modelo** es una abstracción de un sistema que ayuda a comprenderlo.

- Una **vista** es una proyección de la estructura y organización de un modelo del sistema, centrada en algún aspecto.
- Un **diagrama** es una representación de un conjunto de elementos.

9. Diagrama de clases en Métrica V3

Proceso *PSI*:

PSI 4: Identificación de requisitos.

PSI 4.2: Análisis de las necesidades de información.

Proceso *EVS*:

EVS 2: Estudio de la situación actual.

EVS 2.3: Descripción de los sistemas de información existentes.

EVS 4: Estudio de alternativas de solución.

EVS 4.2: Descripción de las alternativas de solución.

Proceso *ASI*:

ASI 1: Definición del sistema.

ASI 1.1: Determinación del alcance del sistema.

ASI 4: Análisis de casos de uso.

ASI 4.1: Identificación de clases asociadas a un caso de uso.

ASI 5: Análisis de clases.

ASI 5.1: Identificación de responsabilidades y atributos.

ASI 5.2: Identificación de asociaciones y agregaciones.

ASI 5.3: Identificación de generalizaciones.

Proceso *DSI*:

DSI 2: Diseño de la arquitectura de soporte.

DSI 2.1: Diseño de subsistemas de soporte.

DSI 2.2: Identificación de mecanismos genéricos de diseño.

DSI 4: Diseño de clases.

DSI 4.1: Identificación de clases adicionales.

DSI 4.2: Diseño de asociaciones y agregaciones.

DSI 4.3: Identificación de atributos de las clases.

DSI 4.4: Identificación de operaciones de las clases.

DSI 4.5: Diseño de jerarquía.

DSI 4.6: Descripción de métodos de las operaciones.

10. Diagrama de paquetes en Métrica V3

Proceso *ASI*:

ASI 3: Identificación de subsistemas de análisis.

ASI 3.1: Determinación de subsistemas de análisis.

ASI 3.2: Integración de subsistemas de análisis.

Proceso *DSI*:

DSI 1: Definición de la arquitectura del sistema.

DSI 1.5: Identificación de subsistemas de diseño.

Tema 15. DSI con TOO: Modelado de comportamiento

1. Introducción

El modelado de comportamiento describe como los objetos colaboran entre sí para realizar cierta actividad. Se expresan mediante los **diagramas de interacción** (*Diagramas de secuencia y de colaboración/comunicación*).

También se describe las **máquinas de estado** que caracterizan a los objetos.

2. Diagramas de interacción

Es un diagrama que muestra en detalle un caso de uso del negocio, del sistema o un determinado **escenario** para un caso de uso.

Los objetos interactúan para realizar colectivamente los servicios ofrecidos por las aplicaciones.

Los diagramas de interacción muestran cómo se **comunican los objetos** (*operaciones públicas*) en una interacción. Sus objetivos son describir **el comportamiento dinámico** del sistema y permitir **verificar la coherencia** del sistema validándolo con el modelo de clases.

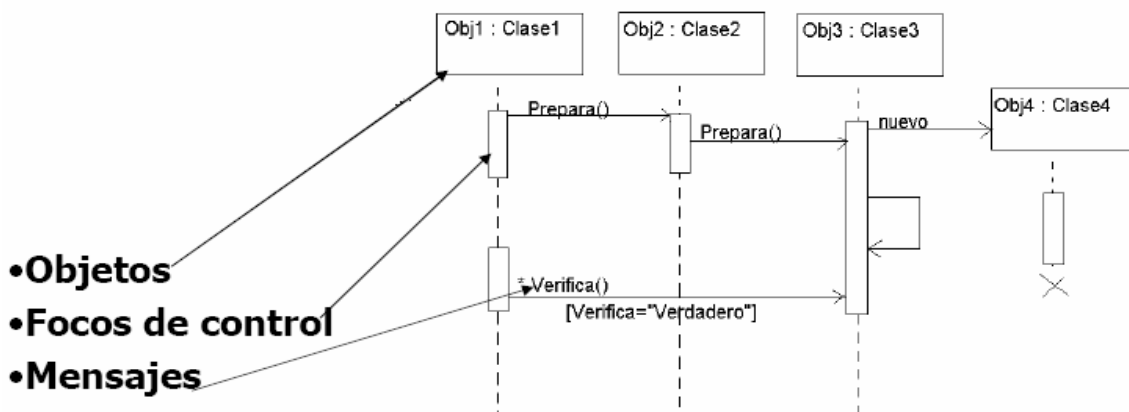
Estos diagramas representan objetos y mensajes entre estos. Existen dos tipos de diagramas:

- **De secuencia:** Es más adecuado para observar la perspectiva **cronológica** de las interacciones.
- **De colaboración:** Ofrece una mejor **visión espacial** mostrando los enlaces de comunicación entre objetos.

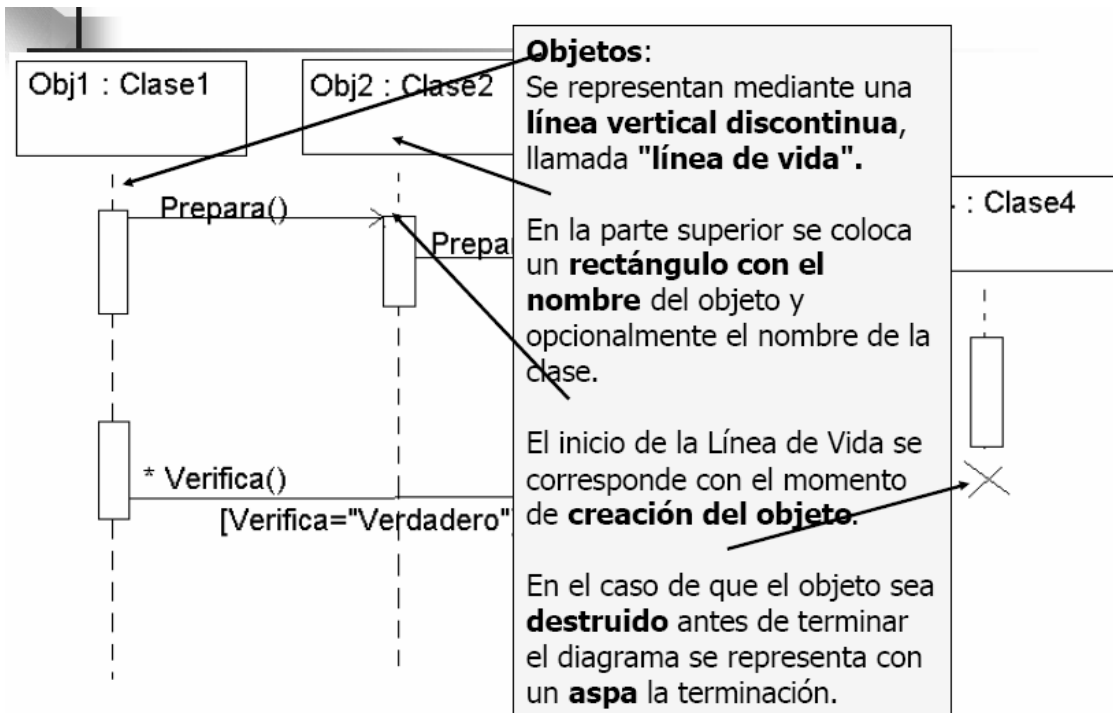
Cada diagrama puede ser obtenido a partir del otro.

2.1 Diagrama de secuencia

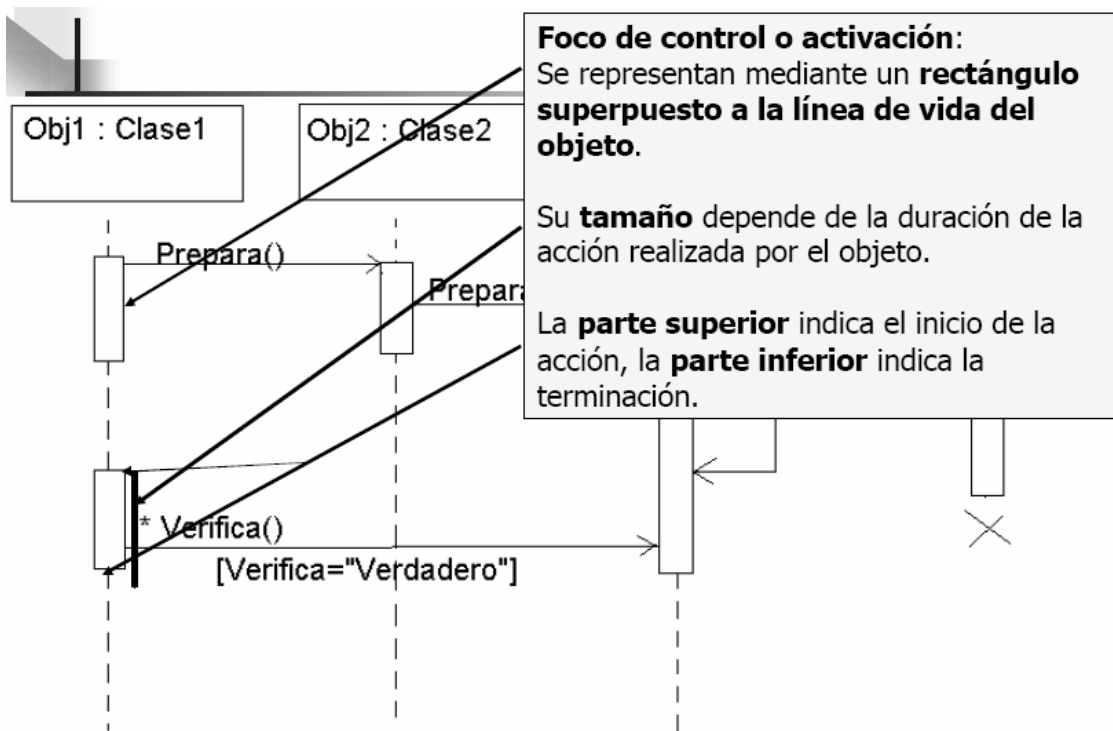
Es un diagrama de interacción cuyo objetivo es describir el **comportamiento dinámico** del sistema. Hace énfasis en los **objetos, mensajes entre objetos y su secuencia**. Existen tres tipos de elementos:



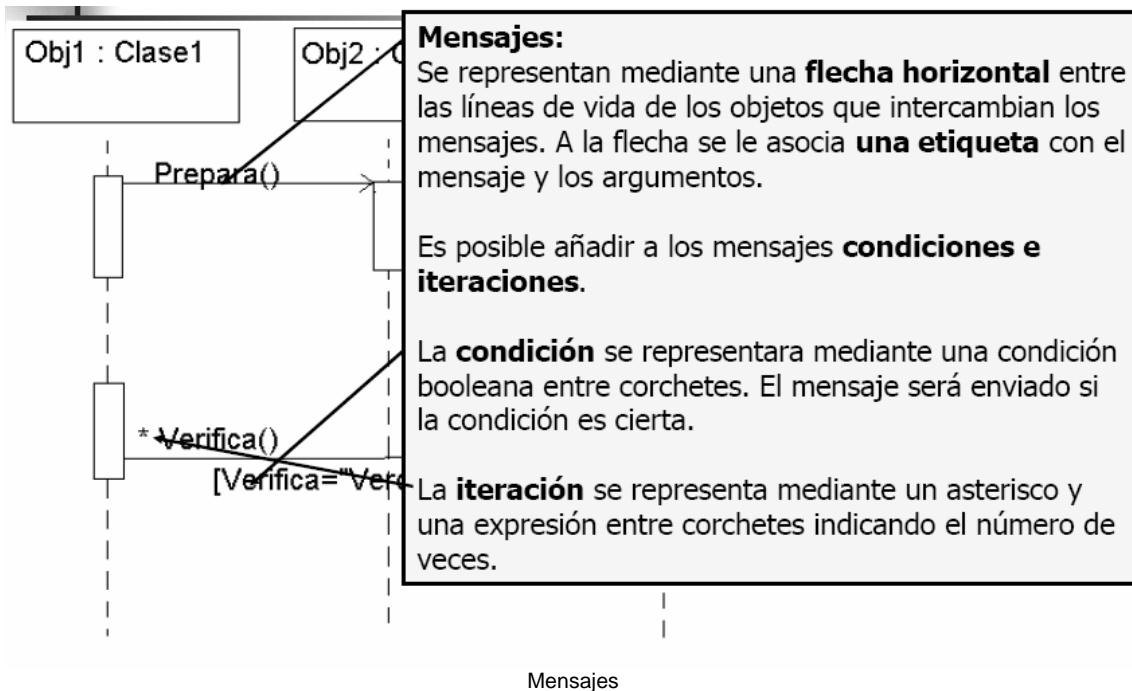
Elementos de un diagrama de secuencia







Objetos



Focos de control



Tipos de mensajes:

- 
 Síncrono o Llamada (El originario del mensaje **pierde el control hasta recibir la respuesta**)
- 
 Retorno (**No es un mensaje**, sino la respuesta de un mensaje previo)
- 
 Directo (Flat) (El originario **no espera respuesta**, pero se **pasa el control** al que recibe el mensaje, el fin del foco de control devuelve el control)
- 
 Asíncrono (El originario **no espera respuesta** y permanece activo pudiendo enviar mensajes)

Tipos de mensajes

Mensajes de un objeto a si mismo:

Un objeto recibe el control cuando recibe un mensaje, sin embargo, cuando un objeto se manda a si mismo un mensaje ya tenía el control. Es posible representar esto mediante anidamiento.

Diagramas de secuencias / UML 2.0

Sirven para representar fragmentos, bucles e instancias de interacciones. Dos formatos de diagramas de secuencia:

- **Formato de instancia:** Describen un escenario concreto en detalle sin bifurcaciones, condiciones y bucles. Se usa para representar un escenario de un caso de uso.

- **Formato genérico:** Describe todas las posibles alternativas de escenarios de un caso de uso en un único escenario de un diagrama de secuencia.

2.2 Diagrama de colaboración/comunicación

Es un diagrama de interacción cuyo objetivo es describir el **comportamiento dinámico** del sistema. Hace énfasis en la **interacción entre objetos** y como están vinculados.

La distribución de los objetos en el diagrama permite observar la interacción de **un objeto con respecto de los demás**.

En un diagrama de colaboración existen tres tipos de elementos:

Objetos:

Se representan mediante un rectángulo en cuyo interior se coloca el nombre del objeto y si se desea el nombre de la clase a la que pertenece.

Vínculo:

Se representa mediante una línea continua que une dos objetos y el cual puede contener uno o varios **mensajes**. Es posible indicar la **navegabilidad** mediante el uso de cabezas de **flecha**.

Mensajes:

Se representan mediante una pequeña flecha asociada a un vínculo. La dirección de la flecha indica el sentido del mensaje. Es necesario indicar delante del mensaje su número dentro de la secuencia de mensajes que se producen. También **pueden contener condiciones e interacciones** al igual que en los diagramas de secuencia.

Diagramas de Secuencia vs. Diagramas de Colaboración/Comunicación

- Equivalencia semántica.
- Simples para comportamientos simples.
- Si hay mucho comportamiento condicional, usar diferentes escenarios.
- **Diagramas de secuencia** muestran el mejor orden en que se ejecutan los mensajes.
- **Diagramas de colaboración/Comunicación** muestran los objetos que interactúa con un determinado objeto.

3. Diagrama de estados

Evento:

Un evento es un **acontecimiento** que ocupa un lugar en el **tiempo y espacio**, un estímulo que **dispara una transición** en una máquina de estados.

Tipos de eventos por su procedencia:

- Eventos externos (*entre actores y el sistema*)
- Eventos internos (*entre objetos*)

Tipos de eventos por su naturaleza:

- Señales (*excepciones*)
- Llamadas
- Paso de tiempo
- Cambio de estado

Estado:

Un estado es una **situación** en la vida de un objeto en la que satisface **cierta condición**, realiza alguna actividad o espera algún evento. Elementos:

- Nombre
- Acciones entrada/salida respecto al estado
- Transiciones internas (*acciones que no afectan al estado*)
- Subestados

Transición:

Una transición de un estado *A* a un estado *B*, se produce cuando se origina el evento asociado y se satisface la condición especificada, en cuyo caso se ejecuta la acción de salida de *A*, la acción de entrada de *B* y la acción asociada a la transición.

Elementos de una transición:

- Estados de origen y destino
- Evento de disparo
- Condición de guarda
- Acción

Acciones:

Podemos especificarlo como la solicitud de un servicio a otro objeto como consecuencia de la transición.

Se puede especificar el ejecutar una acción como consecuencia de entrar, salir, estar en un estado, o por la ocurrencia de un evento.

Diagrama de estados:

Representan autómatas de estados finitos, desde el punto de vista de los estados y las transiciones. Son útiles solo para los objetos con comportamiento significativo, porque el **estado** en el que se encuentra un objeto **determina su comportamiento**.

Los diagramas de estados y de escenario son complementarios.

Generalización de estados

Podemos reducir la complejidad de estos diagramas usando generalización de estados, distinguiendo así entre superestado y subestados. Un estado puede contener varios subestados disjuntos, y estos heredan las variables de estado y las transiciones externas.

4. Diagrama de actividad

El diagrama de actividad ha sido **hasta UML 2** una **especialización del diagrama de estado**, organizado respecto de las acciones y usado para especificar un método, clase, caso de uso o proceso de negocio (*Workflow*).

A partir de *UML* el diagrama de actividad **ya no tiene ninguna connotación de estado**.

Se podría definir como una **secuencia coordinada de unidades subordinadas cuyos elementos individuales son las acciones**. Incluye estados actividad y estados acción, transiciones y objetos.

Una **actividad produce alguna acción** que produce algún cambio en el sistema o retorna un valor. Se realiza un modelado de flujos de trabajo (*Workflow*) como son los **procesos de negocio** (*business process*). Un diagrama de actividad se puede asociar a cualquier elemento de modelado, pero lo más normal es **asociarlo a una operación: diagrama de flujo de las acciones**.

Una **acción** no se puede descomponer, representa una computación atómica.
Una **actividad** no es atómica, **se compone de otras acciones y/o actividades**.

UML 2.0

Object Nodes:

Los diagramas de actividad en *UML 2.0* también soportan los nodos objeto.

Un nodo objeto, es un nodo actividad que sirve para indicar la necesidad de que un clasificador en un estado concreto este disponible en un punto de la actividad (*como entrada o salida de una acción*).

Actúa como un contenedor de objetos de un tipo específico, desde el cual fluyen. Una nueva notación, denominada *pin*, ha sido incorporada.

Pins, representan entradas a una acción o salidas y se dibujan como pequeños rectángulos unidos a los rectángulos acción.

Structured Activity Nodes:

Un nodo de actividad estructurada es un nodo de actividad que se expande en nodos de actividades subordinadas. Para identificarlas se usa un rectángulo de esquinas redondeadas y la etiqueta `<<structured>>`.

5. Diagramas de Interacción en Métrica V3

Proceso EVS:

EVS 2: Estudio de la situación actual.

EVS 2.3: Descripción de los sistemas de información existentes.

Proceso ASI:

ASI 4: Análisis de los casos de uso.

ASI 4.2: Descripción de la interacción de objetos.

ASI 8: Definición de interfaces de usuario.

ASI 8.4: Especificación del comportamiento dinámico de la interfaz.

Proceso DSI:

DSI 1: Definición de la arquitectura del sistema.

DSI 1.5: Identificación de subsistemas de diseño.

DSI 2: Diseño de la arquitectura de soporte.

DSI 2.1: Diseño de subsistemas de soporte.

DSI 2.2: Identificación de mecanismos genéricos de diseño.

DSI 3: Diseño de casos de uso reales.

DSI 3.1: Identificación de clases asociadas a un caso de uso.

DSI 3.2: Diseño de la realización de los casos de uso.

DSI 3.3: Revisión de la interfaz de usuario.

DSI 3.4: Revisión de subsistemas de diseño e interfaces.

6. Diagramas de estados en Métrica V3

Proceso *ASi*:

ASi 5: Análisis de clases.

ASi 5.1: Identificación de responsabilidades y atributos.

ASi 8: Definición de interfaces de usuario.

ASi 8.4: Especificación del comportamiento dinámico de la interfaz.

Proceso *DSi*:

DSi 3: Diseño de casos de uso reales.

DSi 3.3: Revisión de la interfaz de usuario.

DSi 4: Diseño de clases.

DSi 4.4: Identificación de las operaciones de las clases.

DSi 5: Diseño de la arquitectura de módulos del sistema.

DSi 5.3: Revisión de la interfaz de usuario.

7. Diagramas de actividad en Métrica V3

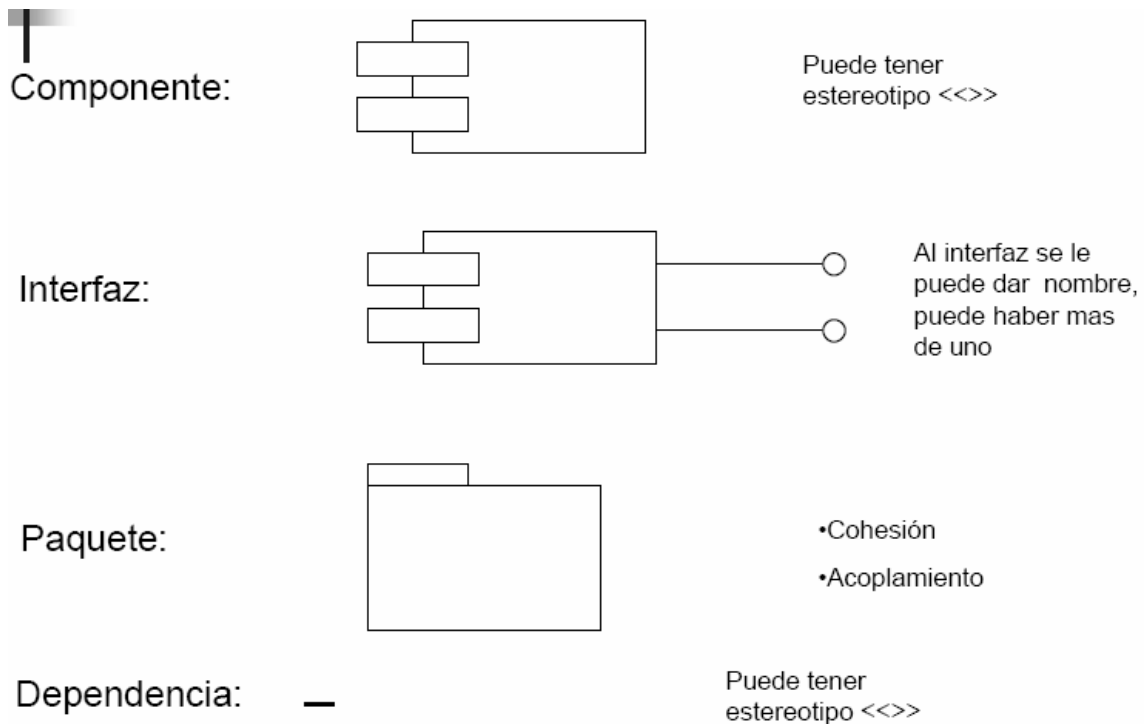
No se propone su uso.

Tema 16. DSI con TOO: Modelado de implementación

1. Componentes

El modelado de implementación en *RUP* es una **colección de componentes** y la implementación de **los subsistemas** que los contienen.

Un componente es una parte **física y reemplazable** de un sistema y proporciona su implementación. **Modela artefactos** tales como: ejecutables, bibliotecas, tablas, ficheros, documentos... y representa el **empaquetamiento físico de elementos lógicos** como clases, interfaces... Estos componentes **residirán en los nodos** del sistema.



Notación componentes

1.1 Tipos de componentes

De despliegue: Los necesarios y suficientes para formar un sistema ejecutable. Ej: Ejecutables y bibliotecas dinámicas (*DLLs*)

Producto del trabajo: Productos finales del proceso de desarrollo. Archivos fuente y de datos a partir de los cuales se crean los de despliegue.

De ejecución: Se crean como consecuencia de un sistema de ejecución. Proceso que se crea a partir de un ejecutable.

Todos los mecanismos de **extensibilidad** se pueden aplicar a los componentes (incluidos estereotipos). UML define cinco estereotipos estándar:

- **executable**: Componente que se puede ejecutar en un nodo.
- **library**: Biblioteca de objetos dinámica o estática.
- **table**: Componente que representa una tabla de una base de datos.
- **file**: Documento con código fuente o datos.
- **document**: Componente que representa un documento.

2. Diagramas de componentes

Representan una **visión física de la organización del SI desarrollado y de sus relaciones a través de sus componentes**.

Muestra la organización de los **componentes software, sus interfaces y las dependencias entre ellos**.

Un **componente** es un módulo de software que puede ser código fuente, binario, ejecutable o una librería con una interfaz definida.

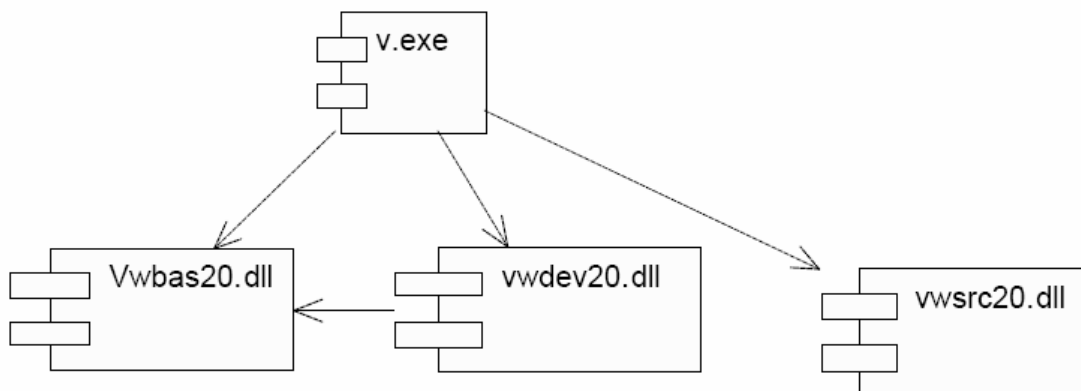
Una **interfaz** establece las operaciones externas de un componente.

En los diagramas de componentes se representan las **relaciones de dependencia** entre componentes o entre un componente y la interfaz de otro. Estos diagramas pueden incluir **paquetes** que permiten organizar la construcción del sistema de información en **subsistemas**.

Muestran las opciones de realización incluyendo código fuente, binario y ejecutable.

Modelado de ejecutables:

Es importante que **cada componente** sea equivalente a un **paquete de clases**. De esta forma, las dependencias entre componentes son las mismas que las existentes entre los paquetes.



Ejemplo

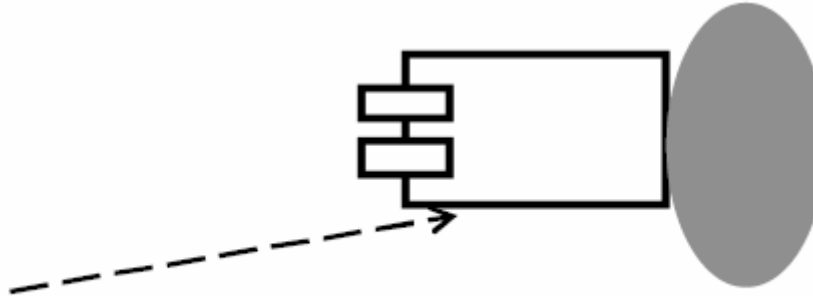
Notación:

Componente: Se representa con un rectángulo con dos pequeños rectángulos superpuestos perpendicularmente en el lado izquierdo. Para distinguir tipos de componentes se les puede asignar un estereotipo cuyo nombre estará dentro del símbolo << >>.

Interfaz: Se representa como un pequeño círculo situado junto al componente que lo implementa y unido a él por una línea continua. La interfaz puede tener un nombre que se escribe junto al círculo. Un componente **puede proporcionar más de una interfaz**.

Relación de dependencia:

Se representa mediante una línea discontinua con una flecha que apunta al componente o interfaz que provee del servicio o facilidad al otro. Puede tener un estereotipo que se coloca junto a la línea.



Componente e interfaz y dependencia

3. Diagramas de despliegue

Definen los elementos de las configuraciones físicas de la red, incluyendo los elementos usados por los **usuarios finales** y también los usados en el **desarrollo** y las **pruebas**.

El objetivo es mostrar la disposición de las **particiones físicas** del *SI* y la **asignación de los componentes** software a estas particiones. En estos diagramas se representan dos tipos de elementos, **nodos y conexiones**, así como la distribución de componentes del sistema de información con respecto a la partición física del sistema.

Los componentes se ejecutan en nodos.

Un **nodo** es un elemento físico que existe en tiempo de ejecución y representa un **recurso computacional** que puede tener memoria y capacidad de procesamiento. Los nodos representan el despliegue físico de los componentes. Respecto a la notación:

- **Nodos:** Como cubos
- **Conexiones:** Como líneas de **comunicaciones**.
- Se reflejan los **protocolos** de comunicaciones.
- Se refleja la **cantidad de nodos** existentes.

El diagrama de despliegue muestra la **configuración de los nodos** que participan en la ejecución y de los componentes que residen en los nodos. Incluye **nodos y arcos** que representa conexiones físicas entre nodos.

La descripción el texto de un diagrama de despliegue se compone de:

Introducción: Breve descripción del modelo.

Nodos: Deben describirse mediante:

- Nombre.
- Describir el procesador, memoria, etc.
- Lista de procesos y threads que se ejecutan en el procesador.

Dispositivos: Que no tienen capacidad de proceso, que soportan los nodos. Cada nodo se describe:

- Nombre.
- Descripción, capacidades.

Conectores: Conexiones entre nodos y entre nodos y dispositivos.

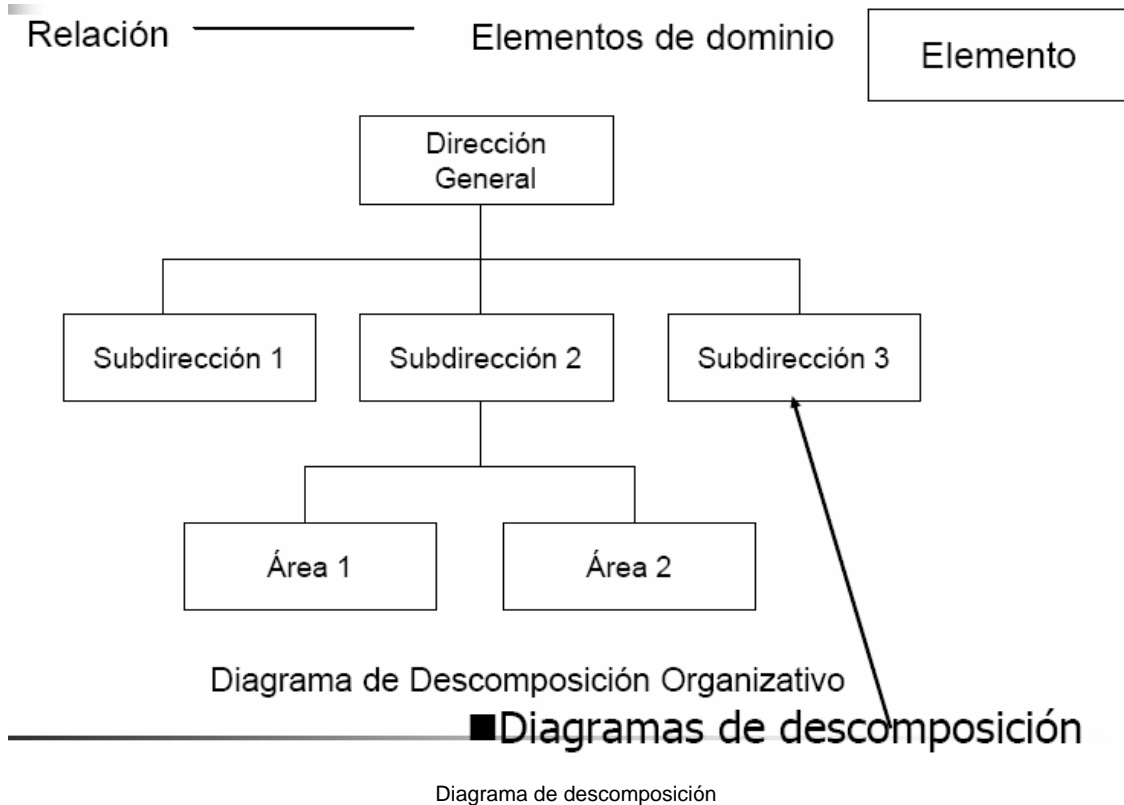
Diagramas: Los diagramas del modelo propiedad de los paquetes.

4. Diagramas de descomposición

Es una **representación de la estructura jerárquica de un dominio**. Se elabora una **estructura por niveles** en donde cada elemento puede ser descompuesto a su vez en otros. Según el lugar de Métrica en el que se use puede recibir el nombre de:

- Diag. de descomposición funcional.
- Diag. de descomposición organizativo.
- Diag. de descomposición en diálogos.

Notación:



5. Diagramas de componentes en Métrica v3

DSI 8: Generación de especificaciones de construcción

DSI 8.2: Definición de componentes y subsistemas de construcción.

DSI 8.3: Elaboración de especificaciones de construcción.

6. Diagramas de despliegue en Métrica v3

ASI 3: Identificación de subsistemas de análisis.

ASI 3.1: Determinación de subsistemas de análisis.

ASI 3.2: Integración de subsistemas de análisis.

DSI 1: Definición de la arquitectura del sistema.

DSI 1.1: Definición de niveles de arquitectura.

DSI 1.5: Identificación de subsistemas de diseño.

DSI 8: Generación de especificaciones de construcción

DSI 8.2: Definición de componentes y subsistemas de construcción.