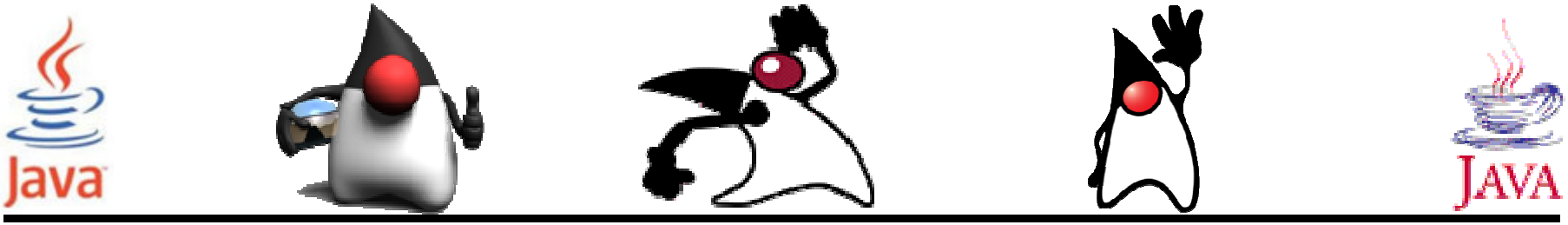


Programación orientada a objetos con Java

Pedro Corcuera

Dpto. Matemática Aplicada y Ciencias de la Computación

Universidad de Cantabria

corcuerp@unican.es

Objetivos

- Presentar los conceptos de la programación orientada a objetos
- Conocer la metodología UML

Índice

- Conceptos básicos de la ingeniería de software
- Metodología RUP-UML

¿ Qué es software ?

- Programas de ordenador y la documentación asociada tales como los requerimientos, modelos de diseño y manuales de usuario.
- Los productos software pueden ser desarrollados para un cliente particular (a medida) o puede ser desarrollado para el mercado general (genérico).
- El software nuevo puede ser creado por el desarrollo de nuevos programas, configurando programas genéricos o reusando software existente.

Impacto del software

- **Economía**: la industria del software representa una parte importante del PIB y emplea a millones de personas en el mundo.

- **Social**: existe una gran dependencia de la sociedad respecto de los productos informáticos y de comunicación.

¿ Qué es ingeniería de software ?

- La ingeniería de software es una disciplina de la ingeniería que se ocupa de todos los aspectos de la producción del software.
- Los ingenieros de software deben adoptar un enfoque sistemático y organizado en su trabajo y usar herramientas y técnicas apropiadas en función del problema a resolver, los problemas de desarrollo y los recursos disponibles.

¿ Qué es un proceso software ?

- Un conjunto de actividades cuyo objetivo es el desarrollo o la evolución del software.
- Las actividades genéricas en todos los procesos de software son:
  - Especificación - lo que debería hacer el sistema y las restricciones de desarrollo
  - Desarrollo - la producción de un sistema informático
  - Validación - comprobar que el software es lo que quiere el cliente
  - Evolución - modificar el software en respuesta a las cambiantes demandas.

¿ Qué es un modelo de proceso software ?

- Una representación simplificada de un proceso presentado desde una perspectiva específica.
- Ejemplos de perspectivas de proceso son:
  - Flujo de trabajo - secuencia de las actividades;
  - Flujo de datos - el flujo de información;
  - Rol / acción - quién hace qué.
- Modelos de procesos genéricos:
  - Cascada;
  - Desarrollo iterativo;
  - Ingeniería de software basada en componentes.

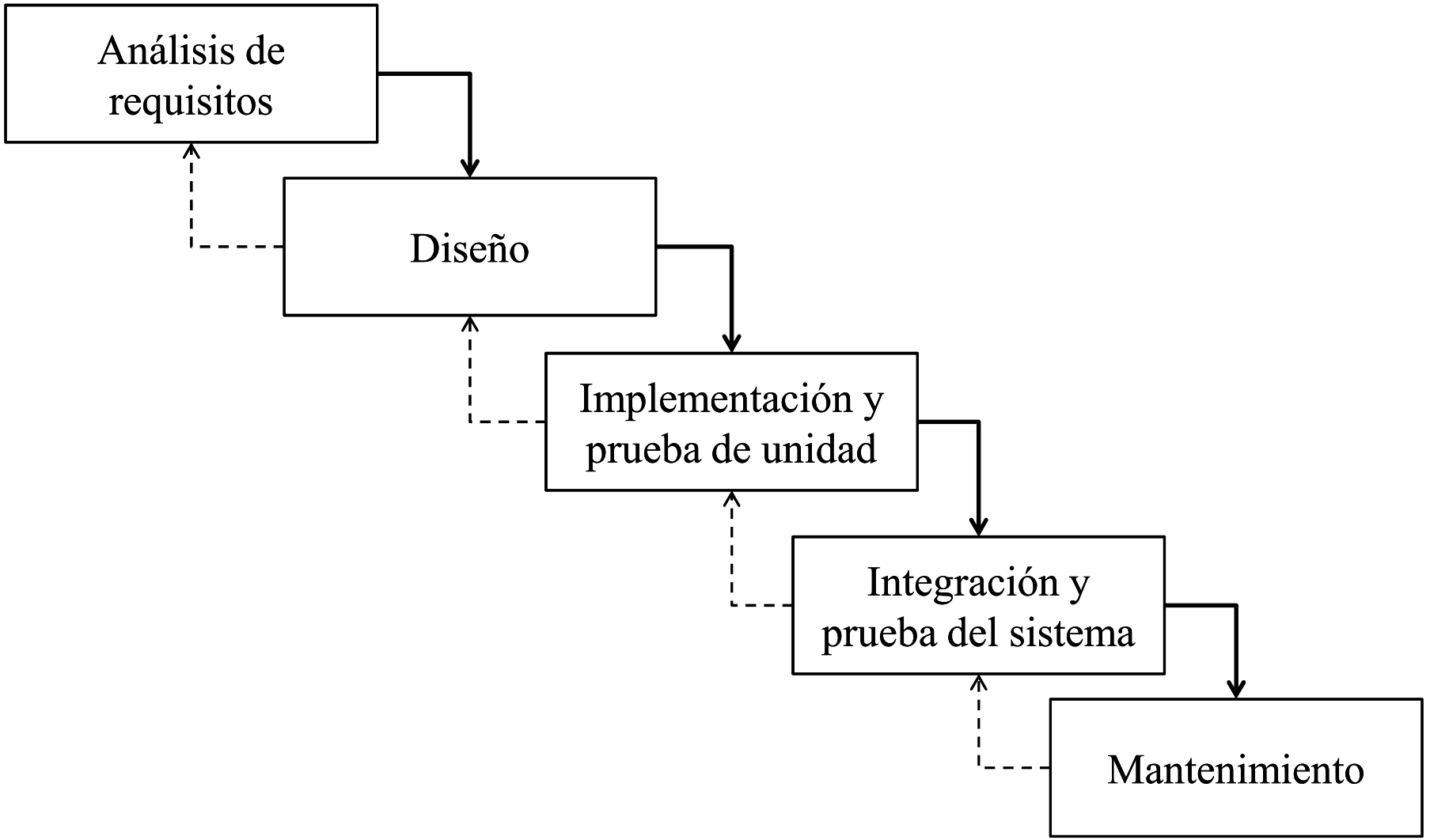
Atributos de un buen software

- El software debe ofrecer la funcionalidad y el rendimiento requerido por el usuario y debe ser mantenible, confiable y aceptable.
- *Mantenibilidad*: debe evolucionar para satisfacer las nuevas necesidades;
- *Confiabilidad*: debe ofrecer seguridad;
- *Eficiencia*: debe hacer uso adecuado de los recursos del sistema;
- *Aceptabilidad*: deben ser comprensibles, utilizables y compatibles con otros sistemas.

Modelos de proceso software genéricos

- El modelo en cascada
  - Fases separadas y distintas de especificación y desarrollo.
- Desarrollo evolutivo
  - Especificación, desarrollo y validación se entrelazan.
- Ingeniería de software basada en componentes
  - El sistema es ensamblado a partir de componentes existentes.
- Hay muchas variantes de estos modelos, p.e. desarrollo formal (cascada con especificación formal + refinamiento en etapas hasta un diseño factible).

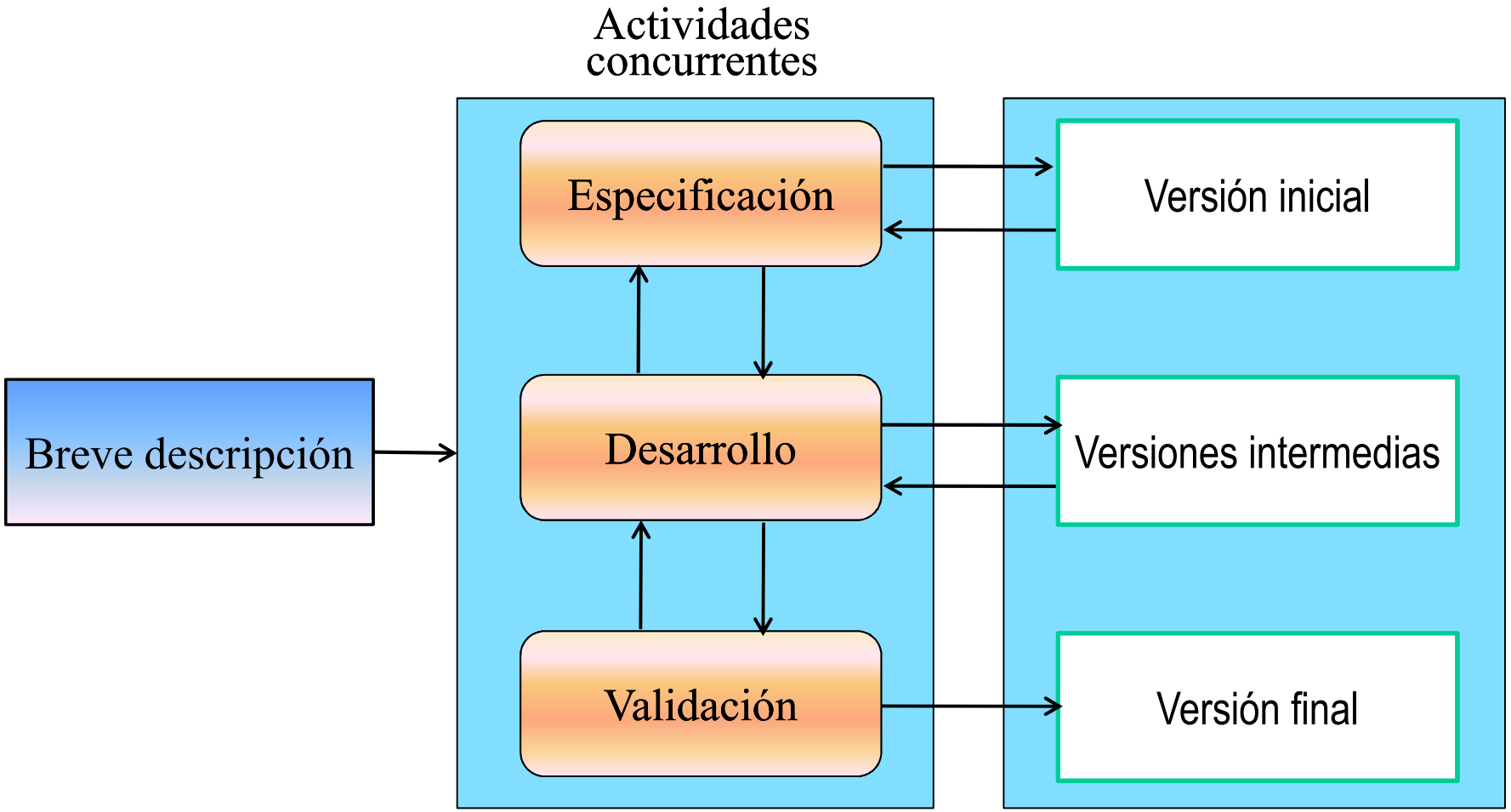
Modelo en cascada (waterfall model)



Fases del modelo en cascada

- Análisis de requerimientos y definición
- Diseño del sistema y software
- Implementación y pruebas unitarias
- Integración y prueba del sistema
- Operación y mantenimiento
- El principal inconveniente del modelo de cascada es la dificultad de incorporar los cambios después de que el proceso está en marcha. Una fase tiene que ser completada antes de pasar a la siguiente fase.

Modelo de proceso evolutivo



Otros modelos de proceso

- Basado en componentes (orientado a la reusabilidad)
- Iterativo
  - Desarrollo espiral (Boehm, 1988)
  - Rational Unified Process (RUP, proceso orientado a objetos)
  - Programación extrema (XP)
  - Métodos ágiles

Actividades del proceso software

- Especificación
- Diseño e implementación
- Validación
- Evolución

Especificación software

- Proceso de establecer los servicios requeridos y las limitaciones en el funcionamiento del sistema y el desarrollo.
- Procesos en la ingeniería de requisitos:
  - Estudio de viabilidad (estudio de factibilidad);
  - Obtención y análisis de requisitos;
  - Especificación de requisitos (documento de requerimientos);
  - Validación de requerimientos.

Diseño e implementación software

- Proceso de convertir la especificación del sistema en un sistema ejecutable.
- Diseño de software:
  - Diseñar una estructura software que materialice la especificación;
- Implementación:
  - Traducir esta estructura en un programa ejecutable;
- Las actividades de diseño e implementación están estrechamente relacionados y pueden ser entrelazadas.

Validación del software

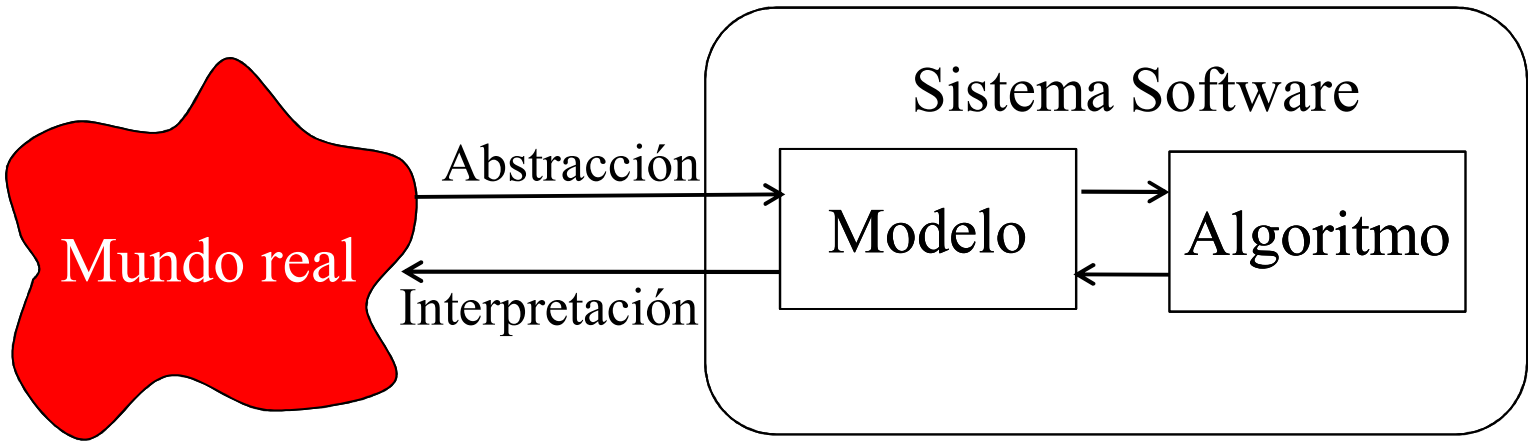
- La verificación y validación (V & V) sirve para demostrar que un sistema cumple con la especificación y los requerimientos establecidos.
- Comprende la comprobación y revisión de los procesos de revisión y las pruebas del sistema.
- Las pruebas del sistema consiste en la ejecución del sistema con casos de prueba que se derivan de la especificación de los datos reales para ser procesados por el sistema.

Evolución del software

- El software es inherentemente flexible y puede cambiar.
- Los requisitos cambian según las circunstancias cambiantes del negocio y por ello el software que lo soporta también debe evolucionar y cambiar.
- Aunque se puede establecer una demarcación entre el desarrollo y evolución (mantenimiento), este es irrelevante puesto muy pocos sistemas son completamente nuevos.

Modelado del mundo real

- El principal objetivo del desarrollo software es crear sistemas software que proporcionen servicios a los usuarios y mejoren sus habilidades para resolver problemas en el mundo real.



Actividades en el desarrollo orientado a objetos

- Conceptualización
  - establecer la visión y requerimientos nucleares del sistema software a desarrollar.
- Análisis y modelado orientado a objetos.
  - construir modelos del comportamiento del sistema usando notaciones como el Unified Modeling Language (UML).
- Diseño orientado a objetos.
  - crear la arquitectura en términos de objetos y clases y las relaciones entre ellos.

Actividades en el desarrollo orientado a objetos

- Implementación
  - desarrollar el diseño mediante el uso de un lenguaje de programación orientado a objetos (p.e. Java).
- Mantenimiento
  - gestionar la evolución después de la entrega del producto. Incluye eliminar errores, mejorar las funcionalidades y adaptar el sistema a nuevas necesidades y entornos.

Evolución de las metodologías de desarrollo software

- 1970s
  - Programación estructurada, 1969
  - Programación estructurada Jackson, 1975
- 1980s
  - Structured Systems Analysis and Design Methodology (SSADM), 1980
  - Structured Analysis and Design Technique (SADT), 1980
  - Ingeniería de la información (IE/IEM), 1981

Evolución de las metodologías de desarrollo software

- 1990s
  - Rapid application development (RAD), 1991.
  - Programación orientada a objetos (OOP), década de los 90's
  - Virtual finite state machine (VFSM), 1990s
  - Dynamic Systems Development Method, 1995.
  - Scrum (desarrollo), última parte de los 90's
  - Programación extrema (XP), 1999

Evolución de las metodologías de desarrollo software

- 2000
  - Enterprise Unified Process (EUP) extensiones RUP, 2002
  - Rational Unified Process (RUP), 2003.
  - Constructionist design methodology (CDM), 2004
  - Agile Unified Process (AUP), 2005

Proceso Unificado de Rational (RUP)

- Es uno de los procesos de desarrollo de sistemas orientados a objetos, junto con el Lenguaje Unificado de Modelado UML, más utilizado.
- Normalmente se describe desde 3 perspectivas:
  - Una perspectiva dinámica que muestra las fases en el tiempo;
  - Una perspectiva estática que muestra las actividades del proceso;
  - Una perspectiva práctica que sugiere buenas prácticas.
- Ejemplo: http://users.dsic.upv.es/asignaturas/facultad/lsi/ejemplorup/index.html

Fases RUP

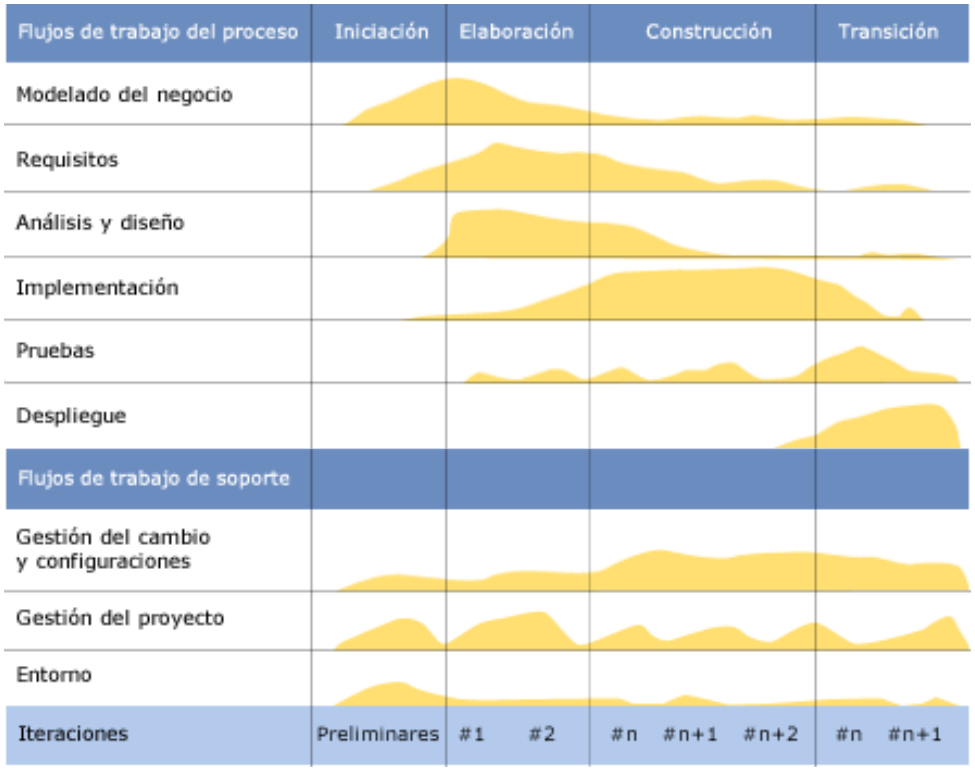
- Concepción
  - Establecer el modelo comercial para el sistema.
- Elaboración
  - Desarrollar un entendimiento del dominio del problema y la arquitectura del sistema.
- Construcción
  - Diseño de sistemas, programación y pruebas.
- Transición
  - Distribuir el sistema a los usuarios finales en su entorno operativo.

Buenas prácticas RUP

- Desarrollar software iterativamente
- Obtener, organizar y gestionar los cambios de requisitos
- Uso de arquitecturas basadas en componentes
- Modelar visualmente el software usando UML
- Verificar de forma continua la calidad del software
- Control de cambios en el software

Flujos de Trabajos RUP

- Proceso:
  - Modelado de negocio
  - Requisitos
  - Análisis y Diseño
  - Implementación
  - Pruebas
  - Despliegue
- Soporte:
  - Gestión del cambio y configuraciones
  - Gestión del proyecto
  - Entorno



Objetos y clases

- Son dos conceptos fundamentales en el desarrollo orientado a objetos (OO).
- Se pueden ver desde dos perspectivas:
  - Su representación en los modelos OO.
  - Su interpretación en el mundo real.
- Cada objeto tiene una identidad única definida por:
  - El estado del objeto: conjunto de *campos* o *atributos* (*fields* o *attributes*) que tiene un nombre, tipo y valor.
  - El comportamiento del objeto: conjunto de *métodos* u *operaciones* que puede realizar el objeto.

Objetos y clases

	Interpretación en el Mundo Real	Representación en el Modelo
Objeto	Representa algo del mundo real que puede identificarse unívocamente.	Tiene una identidad única, un estado y comportamiento.
Clase	Representa un conjunto de objetos con características y comportamientos similares. Los objetos son llamados instancias de la clase.	Caracteriza la estructura de estados y comportamientos que son compartidos por todas las instancias.

Principios del diseño OO - Modularidad

- Un sistema software complejo debe ser descompuesto en un conjunto de ***módulos*** muy cohesionados y débilmente acoplados.
  - *Cohesión* se refiere a la relación funcional de las entidades dentro de un módulo.
  - *Acoplamiento* se refiere a la interdependencia entre diferentes módulos.
- En el enfoque OO los módulos toman la forma de clases y paquetes.

Principios del diseño OO - Abstracción

- Los comportamientos, o funcionalidades, de un módulo debe ser caracterizado en una descripción breve y precisa conocida como *interfaz contractual* del módulo.
- Un módulo se puede ver como un *proveedor de servicios* y otros módulos que lo usan como *clientes*.
- La interfaz contractual se puede ver como un *contrato de servicio* que describe qué servicios se puede ofrecer, no cómo se realiza.

Principios del diseño OO - Encapsulación

- Es un principio complementario a la encapsulación que estipula que el cliente no necesita conocer más que el contrato de servicio mientras use es servicio.
- La implementación de un módulo debe estar separado de su interfaz contractual y oculta a los clientes del módulo.
- Este principio también es conocido como ocultación de la información.

Principios del diseño OO - Polimorfismo

- Varios proveedores de servicios pueden aceptar la misma interfaz contractual y ser intercambiados sin afectar los clientes.
- El polimorfismo es la habilidad para intercambiar módulos dinámicamente sin afectar los clientes.

Lenguaje Unificado de Modelado (UML)

- Unified Modeling Language (UML) es un lenguaje estándar de modelado orientado a objetos de sistemas software, respaldado por el OMG (Object Management Group http://www.uml.org/).
- Combina notaciones provenientes de varios tipos de modelado:
  - Orientado a Objetos
  - Datos
  - Componentes
  - Flujos de Trabajo (Workflows)

Lenguaje Unificado de Modelado (UML)

- UML es un lenguaje de modelado visual que sirve para
  - visualizar,
  - especificar,
  - construir y
  - documentar

  sistemas
- Independientemente de la metodología de análisis y diseño.

UML - Elementos

- Los *elementos de UML* son abstracciones que constituyen los bloques básicos de construcción orientada a objetos en un modelo.
- Se utilizan para construir modelos bien formados.
- Hay cuatro tipos de elementos:
  - Estructurales
  - de Comportamiento
  - de Agrupamiento.
  - de Anotación.

UML – Elementos estructurales

- Son los nombres (sujetos) de los modelos UML. En su mayoría, son las partes *estáticas* de un modelo.
- En UML 2 existen los siguientes tipos:

Clase	Clase activa
Interfaz	Componente
Colaboración	Artefacto
Caso de Uso	Nodo

- Representan cosas conceptuales o lógicas (seis primeros) o elementos físicos (dos últimos).

UML – Elementos de Comportamiento

- Son las partes *dinámicas* de los modelos UML y equivalen a los *verbos* de un modelo.
- Representan comportamiento en el tiempo y el espacio.
- Suelen estar conectados semánticamente a elementos estructurales.
- Hay tres tipos:
  - Interacción
  - Máquina de estados
  - Actividad

UML – Elementos de Agrupación

- Son las partes *organizativas* de los modelos UML.
- Son las “cajas” en las que puede dividirse un modelo.
- Hay un tipo principal:
  - Paquete
- Y varias especializaciones (subtipos de paquetes):
  - Frameworks, Modelos, Subsistemas.

UML – Elementos de Anotación

- Son las partes *explicativas* de los modelos UML.
- Son comentarios que se añaden para describir, clarificar y hacer observaciones.
- Hay un tipo principal:
  - Nota
- Y una especialización (subtipo de nota):
  - Requisito.

UML - Relaciones

- Una relación es una conexión entre elementos estructurales.
- Hay cuatro tipos de relaciones en UML2:
  - Dependencia
  - Asociación
  - Generalización
  - Realización

  } Herencia
- A su vez, hay dos tipos especiales de asociación:
  - Agregación
  - Composición

UML – Relaciones

- Dependencia, relación semántica en la cual un cambio a un elemento (independiente) puede afectar a la semántica del otro.
- Asociación, representa relaciones binarias generales entre clases.
- Herencia, es una de las relaciones más importantes en el modelado OO. Define una relación entre clases e interfaces.

UML – Relaciones

- Generalización
  - Relación de especialización/generalización en la que el elemento especializado (hijo) se basa en la especificación del elemento generalizado (padre).
  - Las subclases (hijos) comparten la estructura y el comportamiento de la superclase (padre).
- Realización, relación semántica entre clasificadores, donde un clasificador especifica un contrato que otro clasificador garantiza que cumplirá.

UML – Relaciones

- Agregación
  - relación estructural que distingue el todo (clase agregada) de la parte (clase componente). El objeto agregado utiliza al incluido para su funcionamiento sin implicar una relación en el tiempo de vida del objeto agregado y los componentes.
- Composición
  - forma de agregación fuerte que implica propiedad exclusiva de la clase componente por la agregada.
- Permiten modelar objetos complejos en base a relaciones todo – parte.

UML - Diagramas

- Un diagrama es la representación gráfica de un conjunto de elementos de modelado (parte de un modelo).
- Se dibujan como un grafo conexo de nodos (elementos) y aristas (relaciones).
- Sirven para visualizar un sistema desde diferentes perspectivas.
- Un diagrama es una proyección gráfica de un sistema y puede contener cualquier combinación de elementos y relaciones.

UML – Tipos de Diagramas

- UML 2.0 tiene 13 tipos diferentes de diagramas.
- Se pueden categorizar jerárquicamente:
  - *Diagramas de Estructura*, enfatizan en los elementos de especificación del sistema modelado sin factores temporales.
  - *Diagramas de Comportamiento*, enfatizan en el comportamiento (dinámica) del sistema/proceso de negocio modelado.
  - *Diagramas de Interacción*, son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado.

UML –Diagramas Estructurales

- Sirven para visualizar, especificar, construir y documentar los *aspectos estáticos* de un sistema.
- Se organizan según los grupos de elementos que aparecen en el modelo:
  - Diagrama de clases
  - Diagrama de componentes
  - Diagrama de objetos
  - Diagrama de estructura compuesta (UML 2.0)
  - Diagrama de despliegue
  - Diagrama de paquetes

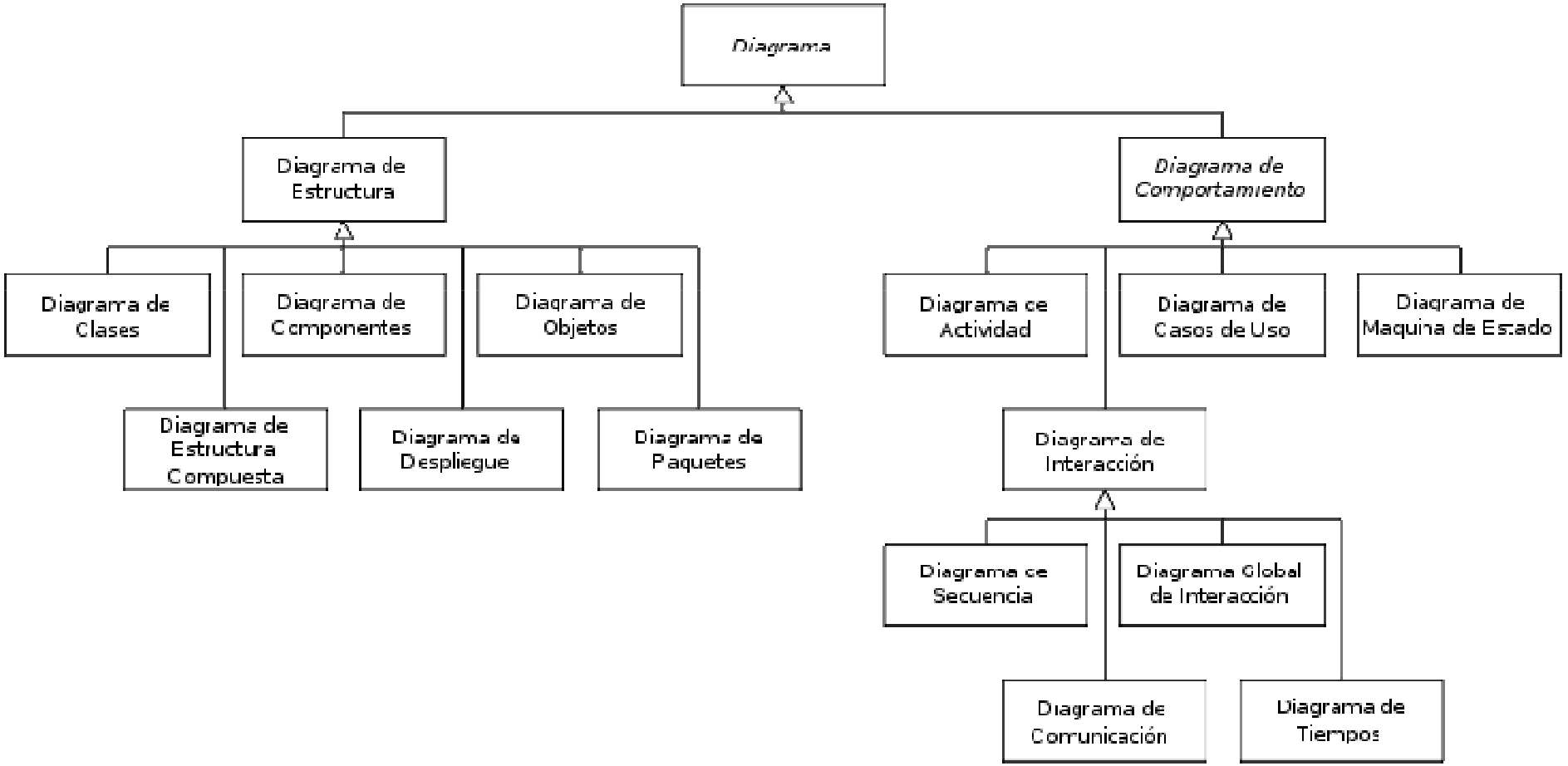
UML –Diagramas de Comportamiento

- Sirven para visualizar, especificar, construir y documentar los *aspectos dinámicos* de un sistema.
- Se organizan en base a las formas en que se puede modelar la dinámica de un sistema:
  - Diagrama de actividades
  - Diagrama de casos de uso
  - Diagrama de estados

UML –Diagramas de Interacción

- Sirven para visualizar, especificar, construir y documentar los *aspectos dinámicos* de un sistema, enfatizando la interacción entre objetos.
  - Diagrama de secuencia
  - Diagrama de comunicación, versión simplificada del Diagrama de colaboración (UML 1.x)
  - Diagrama de tiempos (UML 2.0)
  - Diagrama global de interacciones o Diagrama de vista de interacción (UML 2.0)

UML – Jerarquía de los Diagramas



UML - Diagramas

- Un diagrama es la representación gráfica de un conjunto de elementos de modelado (parte de un modelo).
- Sirven para visualizar un sistema desde diferentes perspectivas.