

4.1. Metodologías de desarrollo SW



Indice

1. Introducción
2. Objetivos de las metodologías de desarrollo SW
3. Evolución de las metodologías de desarrollo SW
4. Clasificación
5. Metodologías estructuradas
6. Metodologías orientadas a objetos
7. Metodologías ágiles
8. Bibliografía

1.Introducción

- Es necesario establecer un enfoque sistemático y disciplinado para llevar a cabo un desarrollo de software
- Una **metodología** es el **conjunto de métodos** que se siguen en una investigación científica o en una exposición doctrinal [RAE, 2001]
- Una **metodología** representa el **camino a seguir** para **desarrollar software** de manera **sistemática**
- Se elaboran a partir del marco definido por uno o varios **ciclos de vida**

1.Introducción

- Se puede definir metodología de Ingeniería de Software como:
 - Un proceso para producir software de forma organizada empleando una colección de técnicas y convenciones de notación predefinidas [Rumbaugh et al. 1991]
- Confusión entre los términos **metodología**, **método** y **ciclo de vida** por abuso del lenguaje técnico:
 - Una **metodología** puede seguir **uno o varios modelos de ciclo de vida**, esto es, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto, pero no cómo. Esto sí lo debe indicar la metodología
 - Una metodología es un concepto más amplio que el de método. Así, se puede considerar a la **metodología** como un **conjunto de métodos**

2.Objetivos de las metodologías

- Establecer los **requisitos** de un sistema software de una forma acertada
- Proporcionar un **método sistemático de desarrollo** de forma que se pueda controlar su proceso
- Construir un sistema software dentro de un **tiempo apropiado** y unos **costes aceptables**
- Construir un sistema que esté **bien documentado** y que sea **fácil de mantener**
- Ayudar a **identificar, lo antes posible, cualquier cambio** que sea necesario realizar dentro del proceso de desarrollo
- Proporcionar un sistema que **satisfaga** a todas las **personas afectadas por el mismo**

3.Evolución de las metodologías

- Desarrollo convencional (años 50)
 - Desarrollo **artesanal y ausencia** de metodología
 - **Enfocado** en la tarea de **programación**
 - Inconvenientes:
 - Resultado final impredecible (resultados muy distintos para proyectos similares)
 - No se puede controlar lo que sucede en el proyecto
 - El éxito de los proyectos se basa mucho en la figura del “héroe”

3.Evolución de las metodologías

- Desarrollo estructurado (Años 60, mediados 70)
 - Programación estructurada
 - Normas para escribir código
 - Facilitar comprensión de programas
 - Normas para la aplicación de estructuras de datos y de control
- Desarrollo estructurado (mitad años 70)
 - Diseño estructurado
 - Independencia del lenguaje de programación
 - Elemento básico de diseño: Módulo
 - Modularidad. Medidas de calidad de programas

3.Evolución de las metodologías

- Desarrollo estructurado (finales años 70)
 - Análisis estructurado
 - Previamente los requisitos se describían de forma narrativa
 - Monolíticas, redundantes, ambiguos, imposibles de mantener
 - Se obtienen especificaciones funcionales
 - Gráficas, particionadas, mínimamente redundantes

3.Evolución de las metodologías

- Desarrollo orientado a objetos (Años 80)
 - Identificación y organización de conceptos del dominio de la aplicación y no tanto de su representación final en un lenguaje de programación
 - Trata funcionalidad y datos de forma conjunta
 - Principios:
 - Abstracción
 - Ocultación de información (Encapsulamiento)
 - Modularidad

3.Evolución de las metodologías

- Desarrollo ágil (Finales de los 90)
 - Métodos menos restrictivos
 - Útiles para aplicaciones de tamaño pequeño o medio con requisitos que cambian rápidamente durante el proceso de desarrollo

4. Clasificación

- Teniendo en cuenta las **notaciones** utilizadas para especificar artefactos:
 - Metodologías Estructuradas
 - Procesos, flujos y estructuras de los datos
 - Metodologías Orientadas a objetos
 - Conjunto de objetos que interactúan entre sí
- Teniendo en cuenta su **filosofía** de desarrollo:
 - Metodologías Tradicionales
 - Mayor énfasis en la planificación y control del proyecto, en la especificación precisa de requisitos y modelado
 - Metodologías Ágiles
 - orientadas a la generación de código con ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños e involucran activamente al cliente en el proceso

Elección de una metodología

- Factores que influyen:
 - Tamaño y estructura de la organización
 - Tipo de aplicaciones a desarrollar
- Se deben analizar y evaluar las metodologías existentes y seleccionar la que mejor se adapte a las necesidades

Características deseables de una metodología

- Existencia de reglas predefinidas
- Cobertura total del ciclo de desarrollo
- Verificaciones intermedias
- Planificación y control
- Comunicación efectiva
- Utilización sobre un abanico amplio de proyectos
- Fácil formación
- Herramientas CASE
- Actividades que mejoren el proceso de desarrollo
- Soporte al mantenimiento
- Soporte de la reutilización de software

5. Metodologías estructuradas

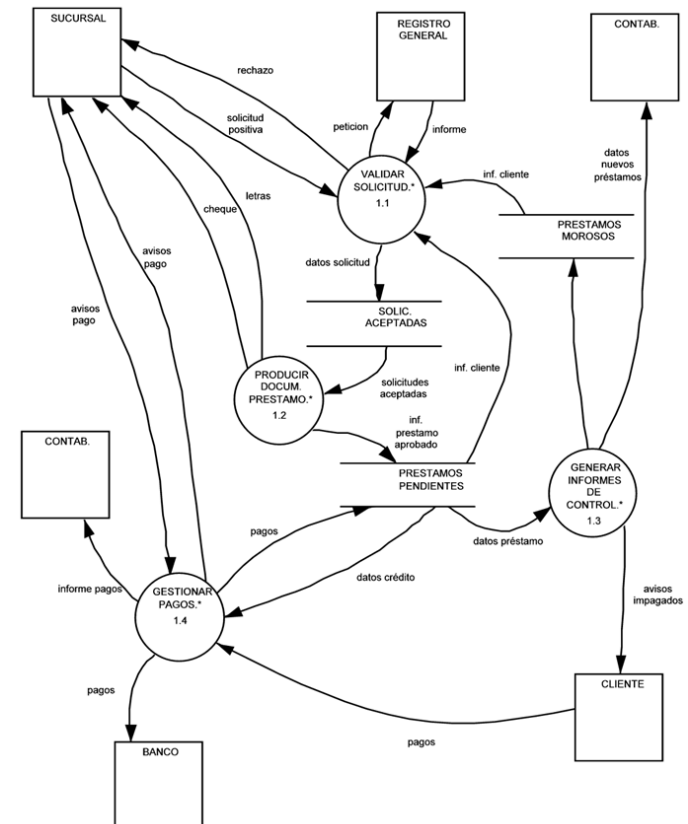
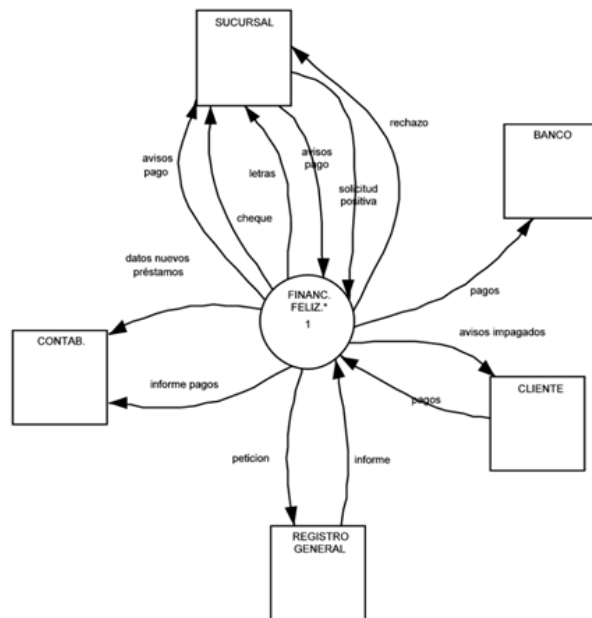
- Proponen la creación de modelos del sistema que representan:
 - Los procesos
 - Los flujos
 - Las estructuras de los datos
- Durante el análisis se identifican las funciones del sistema, mientras que durante el diseño se identifican los datos
- Enfoque Top-Down:
 - Desde una visión general hasta un nivel de abstracción más sencillo

5. Metodologías estructuradas

- Se basan en métodos descendentes de descomposición funcional para definir los requisitos del sistema
- Utilizan técnicas gráficas para obtener una especificación estructurada:
 - Modelo gráfico, particionado, descendente y jerárquico de los procesos del sistema y de los datos utilizados por éstos
 - Componentes:
 - Diagrama de flujo de datos
 - Diccionario de datos
 - Especificaciones de procesos
 - Diagramas Entidad-Relación

5. Metodologías estructuradas

- Diagrama de flujo de datos
 - Permite visualizar el sistema como una red de procesos funcionales, conectados entre sí por flujos de información y almacenes de datos

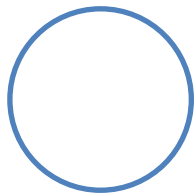


Notación: Yourdon/DeMarco

5. Metodologías estructuradas

- Diagrama de flujo de datos

- Componentes:



- **Proceso:** Equivalente a una función o transformación. Muestra una parte del sistema que convierte Entradas en Salidas. Se nombran mediante frases VERBO+OBJETO. Por ejemplo: Validar pedido



- **Flujo:** Describen el movimiento de paquetes o bloques de información de una parte del sistema a otra. Representan datos en movimiento



- **Almacén:** Modela una colección de datos en reposo



- **Entidad Externa:** Representan entidades externas con las que el sistema se comunica. Suelen ser personas, grupos u otros departamentos de la empresa. Son externos al sistema que se está modelando

5. Metodologías estructuradas

- Diccionario de datos
 - El diccionario de datos es un listado organizado de todos los datos existentes en el sistema
 - Contiene definiciones precisas para que tanto usuarios como analistas entiendan todas las entradas, salidas, componentes de almacenes y cálculos intermedios
 - Describe el significado de los flujos y almacenes que se muestran en los DFD
 - Describen la composición de los paquetes de datos en los almacenes

5. Metodologías estructuradas

- Notación del diccionario de datos
 - Existen muchos esquemas. Uno de los más comunes utiliza los siguientes símbolos:
 - = está compuesto de
 - + y
 - () optativo
 - { } iteración
 - [] seleccionar una de varias alternativas
 - ** comentario
 - @ identificador (campo clave) para un almacén
 - | separa opciones alternativas en la construcción

5. Metodologías estructuradas

- Ejemplos de entradas en un diccionario de datos:
nombre = título de cortesía + nombre de pila + (segundo nombre) + apellido
título de cortesía = [Sr. | Srta. | Sra. | Dr.]
nombre de pila = { carácter legal }
segundo nombre = { carácter legal }
apellido = { carácter legal }
carácter legal = [A - Z | a - z | 0 - 9 | ' | -]

5. Metodologías estructuradas

- Especificaciones de procesos
 - Descripción de qué sucede en cada burbuja primitiva del nivel más bajo de un DFD
 - Define lo que debe hacerse para transformar las entradas en salidas
 - Herramientas:
 - Tablas de decisiones
 - Lenguaje estructurado
 - Pre/post condiciones
 - Diagramas de flujo
 - ...

5. Metodologías estructuradas

- Lenguaje estructurado
 - Subconjunto de todo el idioma
 - Restricciones sobre el tipo de frases que se pueden utilizar y la forma en que pueden juntarse
 - Objetivo: hacer un **balance** razonable entre la **precisión del lenguaje formal** de programación y la **informalidad del lenguaje cotidiano**
 - Ejemplos:
 - $X = (Y * Z) / (Q + 14)$
 - CALCULAR $X = (Y * Z) / (Q + 14)$

5. Metodologías estructuradas

- Pre/post condiciones
 - Describen la función que debe realizar el proceso
 - No especifican el algoritmo que se utilizará
 - Utilidad:
 - El usuario tiene tendencia a explicar los procesos del sistema en términos de un algoritmo particular que ha utilizado durante mucho tiempo
 - El analista está seguro de que existen muchos algoritmos alternativos que podrían usarse

5. Metodologías estructuradas

Precondición 1

Ocurre DATOS-VENTA con TIPO-ITEM que
corresponde con CATEGORIA-ITEM en
CATEGORIAS-IMPUESTO

Postcondición 1

IMPUESTO-SOBRE-VENTA se hace igual a
 $\text{MONTO-VENTA} * \text{IMPUESTO}$

5. Metodologías estructuradas

- Tablas de decisión
 - Si el proceso produce alguna salida o debe realizar alguna acción basándose en decisiones complejas es preferible utilizar una tabla de decisiones
 - Se crean listando todas las variables relevantes o entradas y todas sus acciones relevantes en su lado izquierdo
 - En las filas, variables y acciones se separan mediante una línea horizontal
 - En las columnas se insertan las combinaciones posibles de valores de las variables (reglas)

5. Metodologías estructuradas

- Tabla de decisión

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
|--------------------|---|---|---|---|---|---|---|---|--|
| Edad > 21 | Y | Y | Y | Y | N | N | N | N | |
| Sexo | M | M | F | F | M | M | F | F | |
| Peso < 100 | Y | N | Y | N | Y | N | Y | N | |
| Medicamento1 | X | | | | X | | | X | |
| Medicamento2 | | X | | | X | | | | |
| Medicamento3 | | | X | | | X | | X | |
| Ningún medicamento | | | | X | | | X | | |

5. Metodologías estructuradas

- Diagramas Entidad-Relación
 - Es un modelo que describe con alto nivel de abstracción la distribución de datos almacenados en un sistema
 - Componentes:



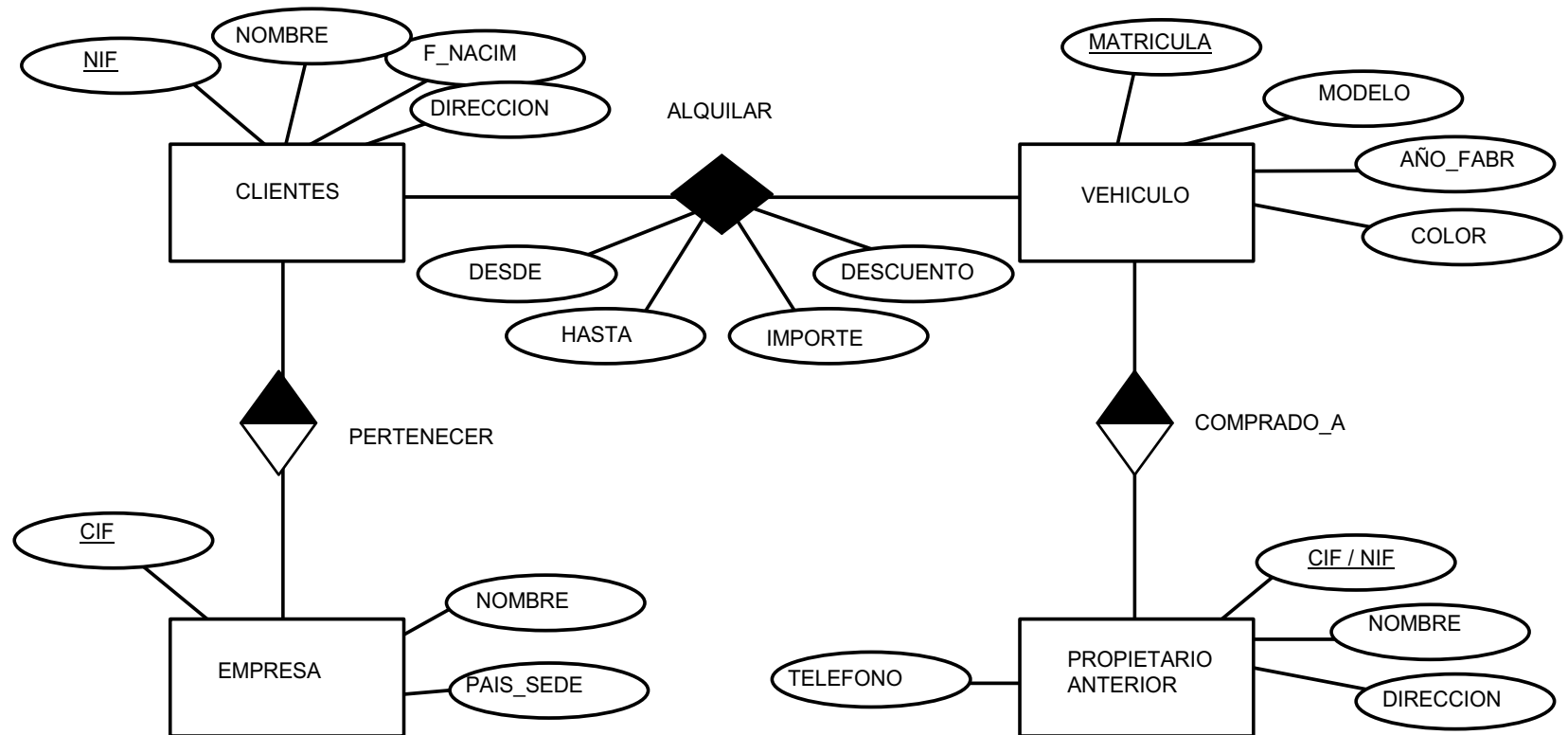
- Entidades: Representan una colección o conjunto de objetos





- Relaciones: Representan el conjunto de conexiones entre entidades

5. Metodologías estructuradas

- Diagrama Entidad-Relación



5. Metodologías estructuradas

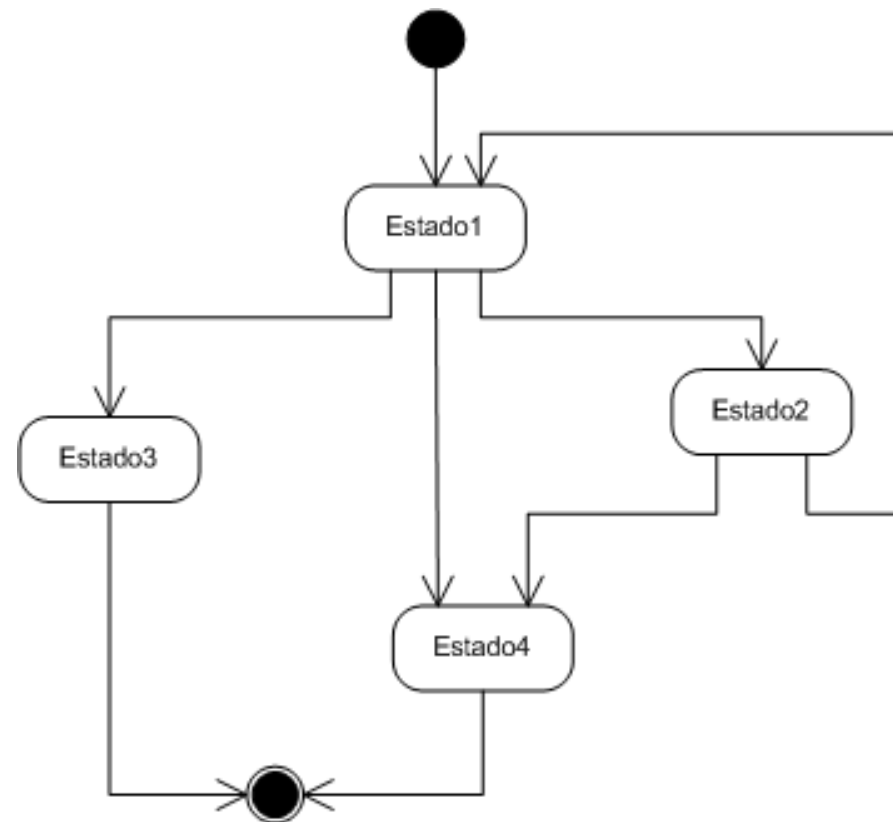
- Diagramas de transición de estados
 - Comportamiento dependiente del tiempo del sistema (sistemas de tiempo-real)
 - Componentes:
 - Estados 
 - Cambios de estado 
 - Cada rectángulo representa un estado en el que se puede encontrar el sistema
 - Un estado es un conjunto de circunstancias o atributos que caracterizan a una persona o cosa en un tiempo dado, forma de ser, condición.

5. Metodologías estructuradas

- Diagramas de transición de estados
 - Los cambios de estados válidos se muestran conectando pares relevantes de estados con una flecha
 - Cualquier estado puede llevar un número arbitrario de estados sucesores
 - La mayoría de los sistemas tienen un estado inicial reconocible y un estado final reconocible
 - El estado inicial típicamente suele ser el que se dibuja en la parte superior del diagrama (no conectado a otro estado anterior)
 - Un sistema puede tener un solo estado inicial pero puede tener múltiples estados finales

5. Metodologías estructuradas

- Diagrama de transición de estados



5. Metodologías estructuradas

- Metodología de DeMarco
 1. Construir el modelo físico actual (DFD físico actual)
 2. Construir el modelo lógico actual (DFD lógico actual)
 3. Derivación del nuevo modelo lógico
 4. Crear un conjunto de modelos físicos alternativos
 5. Estimar los costes y tiempos de cada opción
 6. Seleccionar un modelo
 7. Empaquetar la especificación

5. Metodologías estructuradas

- Metodología de Gane y Sarson
 1. Construir el modelo lógico actual (DFD lógico actual)
 2. Construir el modelo del nuevo sistema: elaborar una especificación estructurada y construir un modelo lógico de datos en tercera forma normal que exprese el contenido de los almacenes de datos
 3. Seleccionar un modelo lógico
 4. Crear el nuevo modelo físico del sistema
 5. Empaquetar la especificación

5. Metodologías estructuradas

- Metodología de Yourdon/Constantine
 1. Realizar el DFD del sistema
 2. Realizar el diagrama de estructuras
 3. Evaluar el diseño
 4. Preparar el diseño para la implantación

5. Metodologías estructuradas



5. Metodologías estructuradas

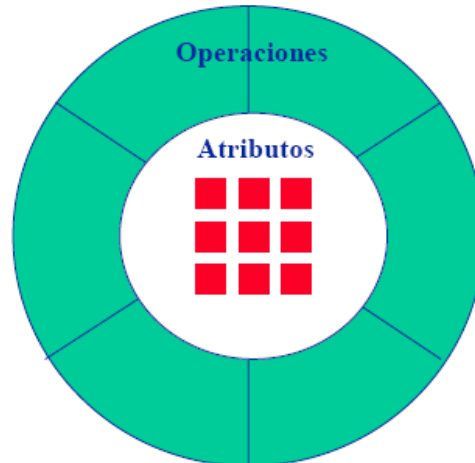
- Ejemplos de metodologías estructuradas gubernamentales: **MERISE** (Francia), **MÉTRICA 3** (España), **SSADM** (Reino Unido)
- Ejemplos de métodos estructurados en el ámbito académico: **Gane & Sarson**, **Ward & Mellor**, **Yourdon & DeMarco** e **Information Engineering**

6. Metodologías orientadas a objetos

- Cambian los principios de las metodologías estructuradas:
 - **Estructurado:** Examinar el sistema desde las funciones y tareas.
 - **OO:** Modelado del Sistema examinando el dominio del problema como un conjunto de **objetos** que **interactúan** entre sí.
 - **Objetos:** Encapsulan **Funciones + Datos**.

6. Metodologías orientadas a objetos

- Encapsulación
 - **Agrupación de datos y operaciones** sobre dichos datos para constituir una unidad de modelado, programación y reutilización
 - En general, los **atributos** de los objetos **no son públicos** y sólo son **accesibles a través de las operaciones asociadas**

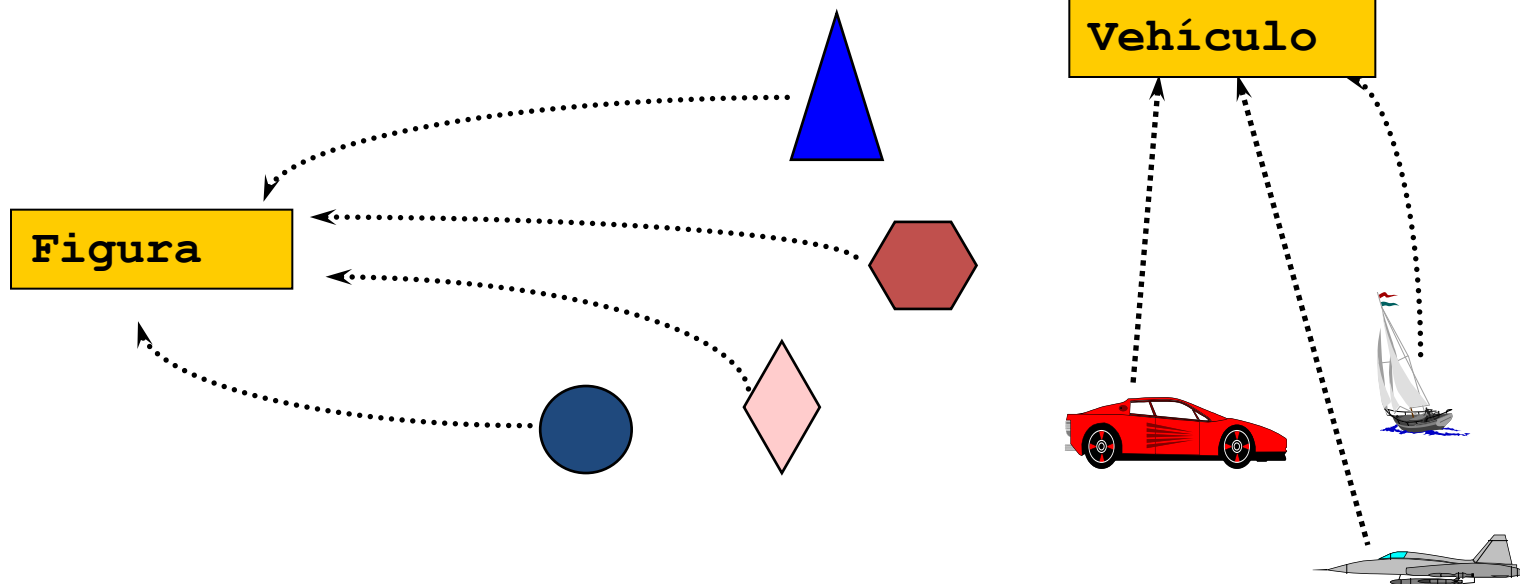


6. Metodologías orientadas a objetos

- Los métodos orientados a objetos basan su modelado en la **identificación de los objetos del problema**
- Los objetos, entidades del mundo real, son estudiados y descritos en función de sus **atributos (datos) y su comportamiento (procesos)**
- Los objetos que son encontrados, se analizan y refinan para ser programados, de forma natural, en **lenguajes orientados a objetos**
- Debido a que los objetos son la unidad durante todo el ciclo de vida del proyecto, estos **métodos** son bastante **uniformes y consistentes**

6. Metodologías orientadas a objetos

- Las clases y los objetos están en todas partes



6. Metodologías orientadas a objetos

- Su historia va unida a la evolución de los lenguajes de programación orientada a objeto
- Los más representativos: a finales de los 60's **SIMULA**, a finales de los 70's **Smalltalk-80**, la primera versión de **C++** por Bjarne Stroustrup en 1981 y actualmente **Java** o **C#**.
- A finales de los 80's comenzaron a consolidarse algunos métodos Orientadas a Objeto
- En 1995 aparece el Método del Proceso Unificado, que posteriormente se reorienta para dar lugar al Unified Modeling Language (UML), la notación OO más popular en la actualidad

6. Metodologías orientadas a objetos

- Hacia principios de los 90 coexistían distintas notaciones y enfoques para modelado de objetos
- Tres de los autores principales (Booch, Jacobson, Rumaugh) decidieron intentar estandarizar notaciones y reunir los mejores rasgos y características de modelos de proceso
- Así surgió el Proceso Unificado (UP) y el UML (Unified Modeling Language)
- El UP reconoce la importancia de la comunicación con el cliente y el importante papel de la arquitectura de software

6. Metodologías orientadas a objetos

- Enfoques OO:

- **“Revolucionarios” o “Puros”**

- La OO se entiende como un cambio profundo de las metodologías estructuradas que se ven como obsoletas.
 - OOD (Booch), CRC/RDD (Wirfs-Brock).

- **“Sintetistas” o “Evolutivos”**

- **Ánàlisis y Diseño Estructurado** se consideran como la base para el desarrollo OO.
 - OMT (Rumbaugh), RUP (Rational Software Corporation)

6. Metodologías orientadas a objetos

- Metodologías dirigidas **por los datos**
 - OMT (*Object Modeling Technique*) [Rumbaugh et al., 1991]
 - Fusion [Coleman et al., 1994]
- Metodologías dirigidas **por las responsabilidades**
 - RDD (*Responsibility Driven Design*) [Wirfs-Brock et al., 1990]
 - OBA (*Object Behavior Analysis*) [Rubin y Goldberg, 1992]
- Metodologías dirigidas **por los casos de uso**
 - Objectory [Jacobson et al., 1992]
 - Proceso Unificado [Jacobson et al., 1999]
- Metodologías dirigidas **por estados**
 - Metodología de Shlaer y Mellor [Shlaer y Mellor, 1992]

6. Metodologías orientadas a objetos

Proceso Unificado

- Es un marco de proceso (*process framework*)
 - Se puede adaptar a las características de un proyecto
- Está dirigido por casos de uso (funciones)
 - Presentes en todas las fases
- Se centra en la arquitectura (estructura)
 - Prioritaria de principio a fin
 - Se facilita su refinamiento progresivo
- Es iterativo e incremental
 - Los riesgos guían la construcción

6. Metodologías orientadas a objetos

Proceso Unificado

- Mejores prácticas del UP
 - Desarrollar iterativamente
 - Administrar los requerimientos
 - Usar arquitecturas de componentes
 - Modelar visualmente (UML)
 - Verificar la calidad
 - Controlar los cambios

6. Metodologías orientadas a objetos

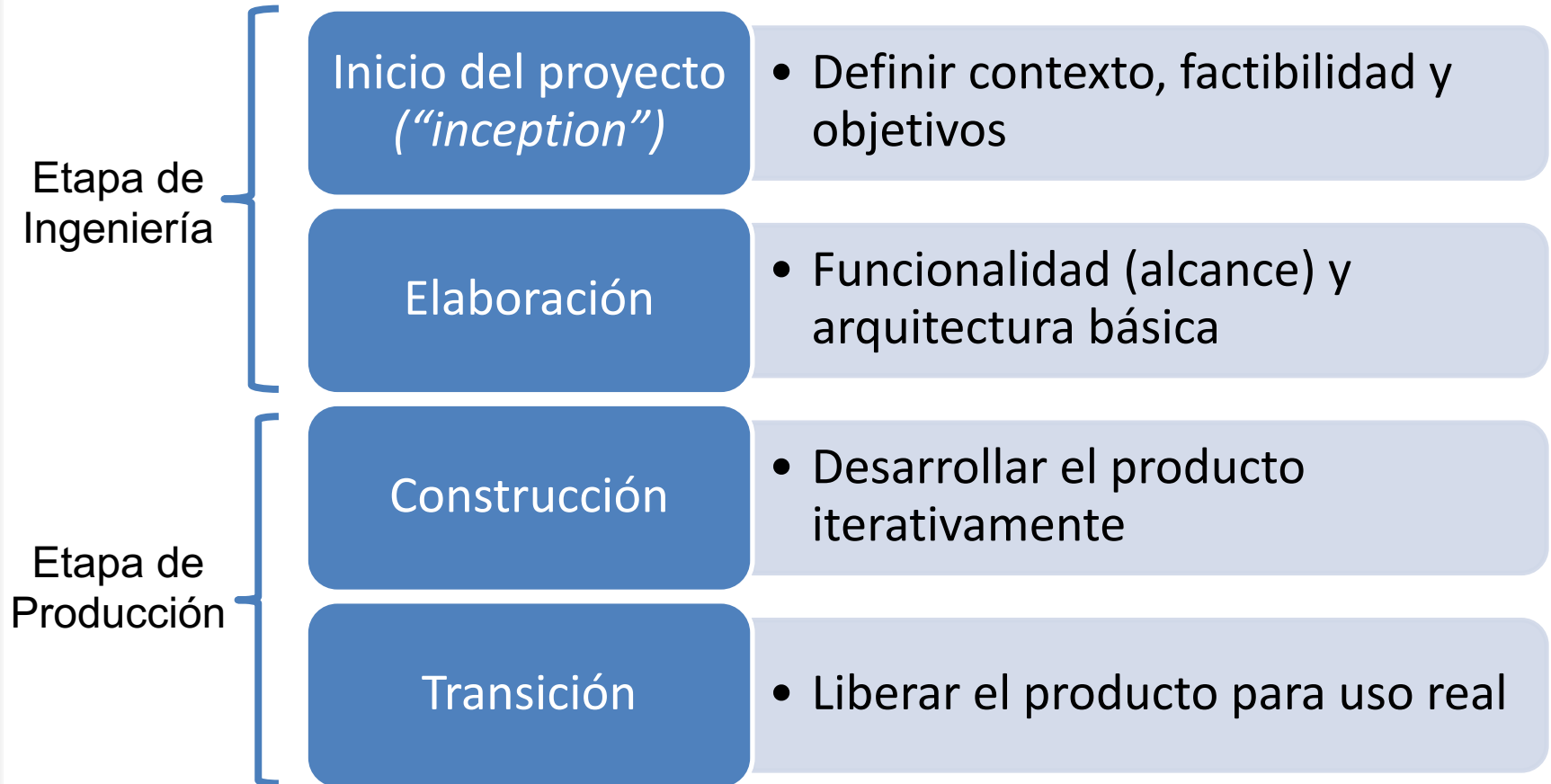
Proceso Unificado

- El resultado de cada iteración es un sistema funcionando, no necesariamente listo para su distribución
- Hay aprendizaje entre iteraciones
- Time boxing (2-12 semanas). Si no se llega, se recorta funcionalidad
- Se busca el *feedback* del usuario
- Los riesgos guían la elección de funcionalidad

6. Metodologías orientadas a objetos

Proceso Unificado

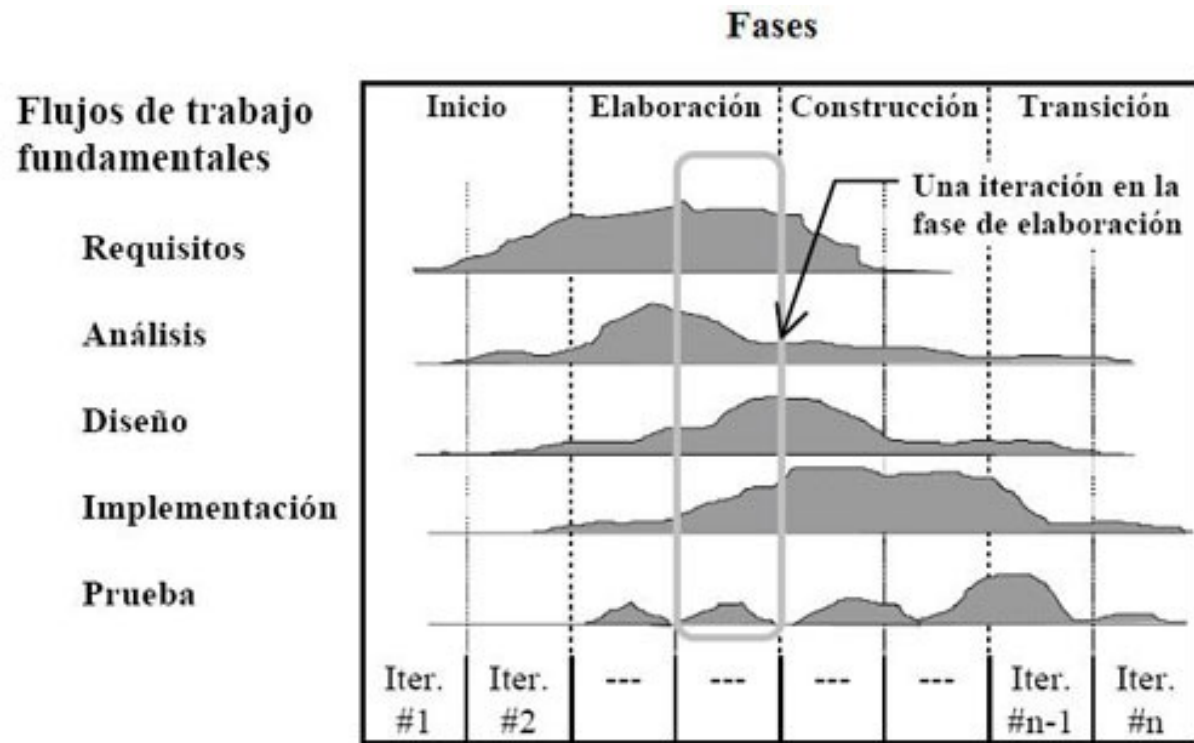
Fases del UP



6. Metodologías orientadas a objetos

Proceso Unificado

- Visión general del proceso



*"El Proceso Unificado de Desarrollo de Software",
I. Jacobson, G. Booch, J. Rumbaugh, de Addison-Wesley*

6. Metodologías orientadas a objetos

Proceso Unificado

- **Fase de inicio**

- ¿Cuáles son las principales funciones del sistema para sus usuarios más importantes?
 - Casos de uso críticos
- ¿Cómo podría ser la arquitectura del sistema?
 - Esbozo mostrando los subsistemas más importantes
- ¿Cuál es el plan de proyecto y cuánto costará desarrollar el producto?
 - Identificación y priorización de riesgos, se planifica la fase de elaboración y se estima el proyecto de manera aproximada

6. Metodologías orientadas a objetos

Proceso Unificado

- **Fase de elaboración**

- Se especifican en detalle la mayoría de los casos de uso
- Se diseña la arquitectura del sistema
- La arquitectura se expresa en forma de vistas de todos los modelos del sistema
- Vistas arquitectónicas del modelo de casos de uso, del modelo de análisis, del modelo de diseño, del modelo de implementación y del modelo de despliegue
- ¿Son suficientemente estables los casos de uso, la arquitectura y el plan, y están los riesgos suficientemente controlados para que seamos capaces de comprometernos al desarrollo entero mediante un contrato?

6. Metodologías orientadas a objetos

Proceso Unificado

- **Fase de construcción**

- Se crea el producto
- La descripción evoluciona hasta convertirse en un producto preparado para ser entregado
- Al final de esta fase, el producto contiene todos los casos de uso que la dirección y el cliente han acordado para el desarrollo de esta versión
- Puede no estar completamente libre de defectos
- ¿Cubre el producto las necesidades de algunos usuarios de manera suficiente como para hacer una primera entrega?

6. Metodologías orientadas a objetos

Proceso Unificado

- **Fase de transición**

- El producto se convierte en versión beta
- Un número reducido de usuarios con experiencia prueba el producto e informa de defectos y deficiencias
- Los desarrolladores corrigen problemas e incorporan algunas de las mejoras sugeridas
- Esta fase conlleva actividades como: la fabricación, formación del cliente, proporcionar una línea de ayuda y asistencia y la corrección de los defectos tras la entrega

6. Metodologías orientadas a objetos

Proceso Unificado

- Hitos en UP
 - Puntos de control para revisar el avance
 - Sincronizar expectativas
 - Identificar riesgos
 - Tienen entregables asociados para la evaluación
 - Dos tipos de hitos
 - Principales al finalizar una fase (visión, arquitectura básica, capacidad inicial, producto final)
 - Secundarios al finalizar una iteración

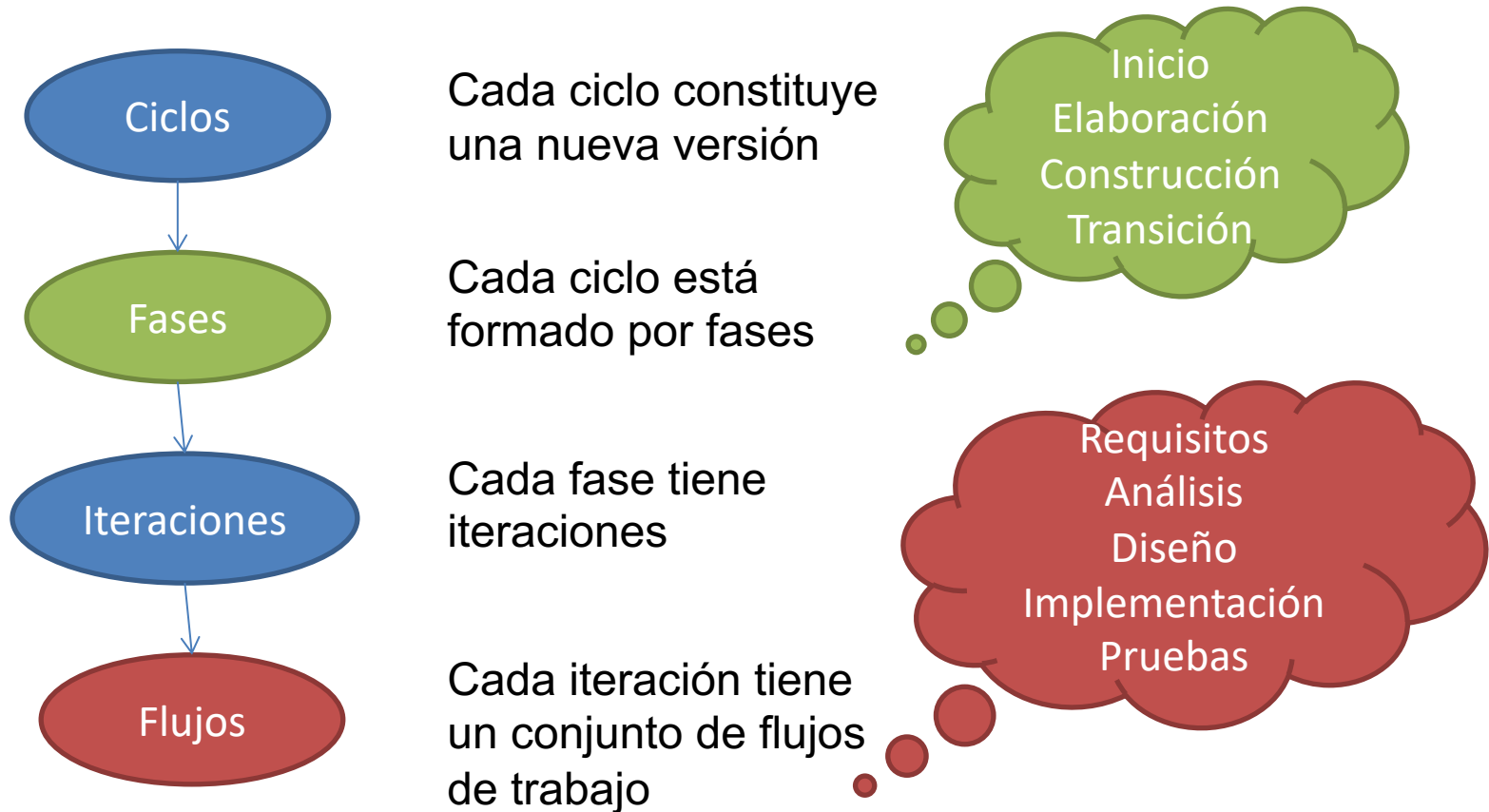
6. Metodologías orientadas a objetos

Proceso Unificado

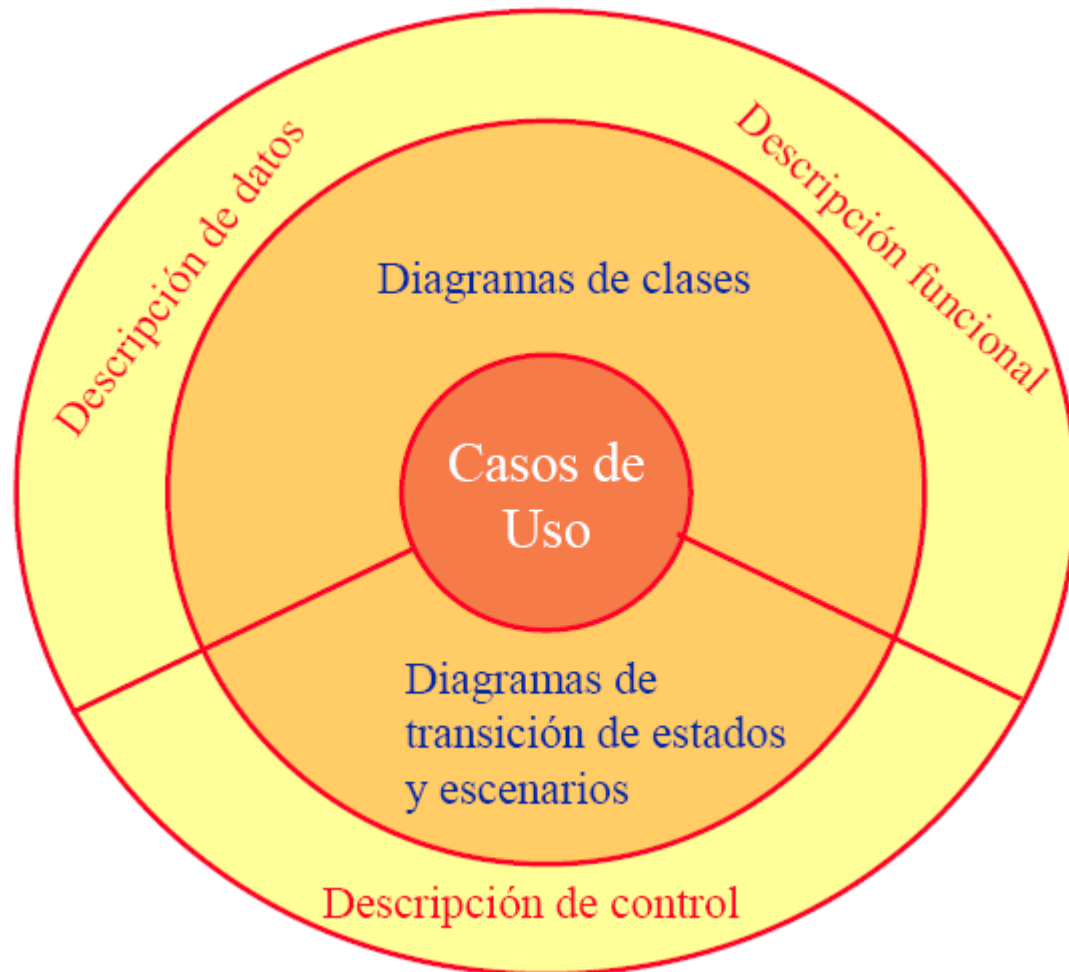
- **Disciplinas:** organizan las actividades del proyecto
 - De desarrollo: requerimientos, análisis, arquitectura, diseño, implementación, pruebas
 - De gestión: administración de riesgos, planificación y seguimiento, etc
 - Cada disciplina de desarrollo genera modelos de UML (casos de uso, análisis, diseño, etc.). Los modelos incluyen diagramas
- **Artefactos:** Cualquier tipo de información generada por el equipo de desarrollo

6. Metodologías orientadas a objetos

Proceso Unificado



6. Metodologías orientadas a objetos



7. Metodologías ágiles

- Simplifican la complejidad de otras metodologías haciendo que la carga de gestión y control sea más liviana.
 - La Agilidad es un aspecto que se puede incorporar a las metodologías estructuradas u OO.
- Entre las más conocidas se encuentran:
 - XP (eXtreme Programming).
 - SCRUM.
 - RAD (Rapid Application Development).

Bibliografía

- Ingeniería del software. Un enfoque práctico. Sexta edición. Roger S. Pressman
- Análisis y diseño de sistemas. Sexta edición. Kendall & Kendall
- Ingeniería del software. Séptima edición. Ian Sommerville
- El proceso unificado de desarrollo de software. Ivar Jacobson, Grady Booch y James Rumbaugh.