

# *Fundamentos de Ingeniería del Software*




## *Capítulo 8. Introducción a los métodos de desarrollo de software*

# *Introducción a los métodos de desarrollo de software. Estructura*



1. Definición.
2. Beneficios.
3. Adaptación del método.
4. Características deseables.
5. Métodos ágiles.
6. Clasificación general.
7. Ejemplos de métodos.

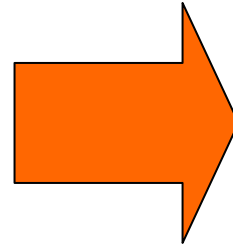
# *Introducción a los métodos de desarrollo de software. Bibliografía*



- (Piattini et al. 96) (Piattini et al. 04)
  - Capítulo 4.

# 1. Definición

- Es necesario establecer un enfoque disciplinado y sistemático para desarrollar un proyecto de software



Método  
(metodología)

Método  $\neq$  Notación

Método  $\neq$  Técnica

Método  $\neq$  Herramienta

# *Definición (II)*




- Conjunto de pasos y procedimientos que deben seguirse para el desarrollo de software
  - Cómo se debe dividir un proyecto en etapas.
  - Qué tareas se llevan a cabo en cada etapa.
  - Heurísticas para llevar a cabo dichas tareas.
  - Qué salidas se producen y cuándo se deben producir.
  - Qué restricciones se aplican.
  - Qué herramientas se van a utilizar.
  - Cómo se gestiona y controla un proyecto.

# *Definición (III)*




- “Conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los desarrolladores a producir nuevo software”
  - **Modelo de proceso** (fases y subfases, procesos, actividades, tareas)
  - **Procedimientos**, que definen la forma de ejecutar las tareas y dan lugar a **productos (artefactos)**
  - **Técnicas** (gráficas, textuales) (p.ej. DFDs, E/R)
  - **Herramientas** (p.ej. System Architect, Enterprise Architect)
- Puede acomodar varios ciclos de vida:
  - Ciclo de vida: qué hay que producir, no cómo
  - Método: qué y cómo

# *Definición (IV)* (Sommerville 2002)



- "Un método de ingeniería de software es un enfoque estructurado para el desarrollo de software cuyo propósito es facilitar la producción de software de alta calidad de una forma costeable."
- Todos los métodos se basan en la idea de modelos gráficos de desarrollo de un sistema y en el uso de estos modelos como un sistema de especificación o diseño.

# Definición (V) (Sommerville 2002)



Componentes	Descripción	Ejemplo
<b>Descripciones del modelo del sistema</b>	Descripciones de los modelos del sistema que se desarrollará y la notación utilizada para definir estos modelos	Modelos de objetos, de flujo de datos, de máquina de estado, etc.
<b>Reglas</b>	Restricciones que siempre aplican a los modelos de sistemas	Cada entidad de un modelo de sistema debe tener un nombre único
<b>Recomendaciones</b>	Heurística que caracteriza una buena práctica de diseño en este método. Seguir estas recomendaciones debe dar como resultado un modelo del sistema bien organizado.	Ningún objeto debe tener más de 7 subobjetos asociados a él.
<b>Guías en el proceso</b>	Descripciones de las actividades que deben seguirse para desarrollar los modelos del sistema y la organización de estas actividades.	Los atributos de los objetos deben documentarse antes de definir las operaciones asociadas a un objeto.



## 2. *Beneficios*

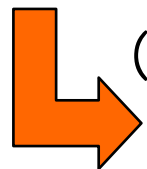


- Sistemas de mayor calidad
  - ¡pero el seguimiento de una metodología no basta!
- Proceso de desarrollo (modelo de procesos) definido
  - ⇒ productos intermedios en cada fase
  - ⇒ mejor planificación y gestión del proyecto
    - desarrollos más rápidos
    - recursos adecuados
- Proceso estándar en la organización
  - ⇒ facilidad de cambios de personal

### 3. *Adaptación del método*



- No existe un método “universal” o “ideal”
  - Métodos diferentes tienen distintas áreas donde son aplicables
    - P.ej., los métodos OO son adecuados para sistemas interactivos, pero no para sistemas en tiempo real con requisitos severos (Sommerville 2002)
- El método está condicionado por el tamaño y estructura de la organización, y el tipo de aplicaciones.
- “No es razonable pensar que dos organizaciones utilicen la misma metodología sin realizar cambios sobre ella”.



(entre otras razones)

**Proliferación de metodologías, técnicas, notaciones**

# 4. *Características deseables*

(Piattini et al. 04)



- Existencia de reglas predefinidas.
  - Procesos, actividades, tareas, productos intermedios, técnicas, herramientas, etc.
- Cobertura total del ciclo de desarrollo.
- Verificaciones intermedias.
- Planificación y control.
- Comunicación efectiva.
- Uso sobre un amplio abanico de proyectos.
- Fácil formación.

# *Características deseables (II)*



- Herramientas CASE.
- Debe contener actividades que mejoren el proceso de desarrollo.
- Soporte al mantenimiento.
  - p.ej. reingeniería
- Soporte de la reutilización del software
  - no sólo reutilización de código
- En muchos dominios, no debería ser muy "burocrático"
  - ⇒ Métodos "ágiles"

## 5. *Métodos ágiles* (Sommerville 2004)



- En los 80 y principios de los 90, existía un acuerdo generalizado en que la mejor forma de desarrollar buen software era a través de:
  - Cuidadosa planificación del proyecto
  - Formalizar el aseguramiento de calidad
  - Métodos de análisis y diseño soportados por herramientas CASE
  - Proceso de desarrollo controlado y riguroso
- Procedía de la experiencia de desarrollo de sistemas software grandes, de larga vida, que estaban compuestos de un gran número de programas
  - Muchas veces estos sistemas eran críticos
  - Grandes equipos de desarrollo, a veces geográficamente dispersos, a veces trabajando incluso para empresas distintas
    - P.ej., el software para un sistema de control de un avión
      - A veces lleva diez años desde la especificación inicial al despliegue
  - En estos sistemas era necesario una sobrecarga en planificación, diseño y documentación del sistema

# Métodos ágiles (II)

- ¿Qué ocurre cuando se aplican estos métodos burocráticos (*heavyweight*) a una aplicación de gestión de tamaño medio o pequeño?
  - La sobrecarga de trabajo (requisitos, diseño, documentación) domina el proceso de desarrollo
  - Se dedica más tiempo a cómo el sistema será desarrollado que a programación y prueba
  - Cada vez que cambia un requisito hay que hacer mucho trabajo de rediseño y redocumentación
- ⇒ Insatisfacción del cliente y del equipo de desarrollo
- ⇒ A finales de los 90, aparecen “métodos ágiles” (*lightweight*) como *extreme programming* (Beck 2000)
  - Atención al software más que al diseño y la documentación
  - Iterativos e incrementales

# *Métodos ágiles (III)* (Sommerville 2004)

Principios de los métodos ágiles	Descripción
<b>Implicación del cliente</b>	Los clientes deberían estar muy involucrados en el proceso de desarrollo. Deben proporcionar y otorgar prioridades a los nuevos requisitos del sistema y evaluar las iteraciones del sistema.
<b>Entrega incremental</b>	El software es desarrollado en incrementos, y es el cliente el que especifica los requisitos que se deben incluir en cada incremento.
<b>Personas, no proceso</b>	Las destrezas del equipo de desarrollo deben ser reconocidas, y se les debe sacar partido. Los miembros del equipo deberían poder usar sus propias formas de trabajo (creatividad) en lugar de trabajar con procesos prescriptivos.
<b>Asumir el cambio</b>	Ser consciente de que los requisitos del sistema cambiarán, de manera que se debe diseñar el sistema para acomodar esos cambios.
<b>Mantener la simplicidad</b>	El enfoque debe estar en la simplicidad, tanto en el software como en el proceso de desarrollo. Donde sea posible, trabajar activamente para eliminar la complejidad del sistema.

# Métodos ágiles (IV)

- ¿Cuándo son útiles? (Sommerville 2004)
  - Útiles para aplicaciones de gestión y productos software de tamaño pequeño o medio con requisitos que cambian rápidamente durante el proceso de desarrollo
  - No son útiles en desarrollo de software de gran escala con equipos de desarrollo en lugares distintos y con interacciones complejas con otros sistemas software y hardware.
  - Tampoco son útiles en sistemas críticos en los que es necesario un análisis detallado de los requisitos del sistema para comprender las implicaciones de seguridad (*security* y *safety*)

⇒ *Con los métodos ágiles,  
¿se puede articular el SRS como contrato con los clientes?*



# 6. Clasificación de métodos

*Adaptado de (Piattini et al.)*

ENFOQUE	TIPO DE SISTEMA	FORMALIDAD	FILOSOFÍA
<b>ESTRUCTURADOS</b> Orientados a procesos Orientados a datos Jerárquicos No jerárquicos Mixtos	GESTIÓN	NO FORMAL	ÁGIL
OO	TIEMPO REAL	FORMAL	CONVENCIONAL

# *Clasificación de métodos (II)*



Estructurados: representan los procesos, flujos y estructuras de datos, de una manera jerárquica, descendente

- Ven el sistema como entradas-proceso-salidas
- Orientados a procesos:
  - se centran en la parte proceso
  - constan de (fundamentalmente) DFDs, DD, miniespecificaciones de proceso, E-R/DED
- Orientados a datos:
  - se orientaban más a las entradas y salidas
  - primero se definían los datos
  - a partir de ellos, los componentes procedimentales
  - "Los datos son más estables"

# 7. Ejemplos de métodos

## ■ Orientados a procesos:

- Análisis estructurado - De Marco 79, Gane & Sarson 79
- Análisis estructurado moderno - Yourdon 89
- SSADM – Reino Unido
- Merise - Francia
- MÉTRICA 2.1 – España

## ■ Orientados a datos:

- JSP/JSD Jackson
- Warnier 74

## ■ Ágiles:

- XP, Extreme Programming - Beck 99
- SCRUM
- Cristal
- FDD, Feature Driven Development - Palmer y Fesling 02
- Modelado Ágil - Ambler 02

## ■ Orientados a objetos:

- OMT - Rumbaugh et al. 91
- Booch 94
- Objectory/OOSE - Jacobson et al. 93
- FUSION - Coleman 94
- OOram - Reenskaug 96
- Proceso Unificado - Jacobson et al. 99
- Rational Unified Process (RUP) - Krutchen et al. 99

## ■ Tiempo real, ambos orientados a procesos:

- Ward & Mellor 85
- Hatley & Pirbhay 87