

# METODOLOGÍAS DE INGENIERÍA DE SOFTWARE

## INGENIERÍA DE SOFTWARE I

2º DE GRADO EN INGENIERÍA INFORMÁTICA  
CURSO 2017/2018

Francisco José García Peñalvo / fgarcia@usal.es

Alicia García Holgado / aliciagh@usal.es

Departamento de Informática y Automática

Universidad de Salamanca



UNIVERSIDAD  
DE SALAMANCA

CAMPUS OF INTERNATIONAL EXCELLENCE



# INTRODUCCIÓN

Desde una perspectiva de Ingeniería de *Software*, una metodología

- Describe cómo se organiza un proyecto
- Establece el orden en el que la mayoría de las actividades tienen que realizarse y los enlaces entre ellas
- Indica cómo tienen que realizarse algunas tareas proporcionando las herramientas concretas e intelectuales

Con una metodología se intentan cubrir las siguientes necesidades [Piattini et al., 2004]

- Mejores aplicaciones
- Mejor proceso de desarrollo
- Establecer un proceso estándar en una organización

# DEFINICIÓN DE METODOLOGÍA

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
  - 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
  - Se puede considerar a la metodología como un conjunto de métodos

# CARACTERÍSTICAS DESEABLES EN UNA METODOLOGÍA

Una metodología debe cubrir (Henderson-Sellers y Firesmith, 1999)

- Un proceso de ciclo de vida completo, que comprenda aspectos tanto del negocio como técnicos
- Un conjunto completo de conceptos y modelos que sean internamente consistentes
- Una colección de reglas y guías
- Una descripción completa de artefactos a desarrollar
- Una notación con la que trabajar, idealmente soportada por diversas herramientas CASE y diseñada para una usabilidad óptima
- Un conjunto de técnicas probadas
- Un conjunto de métricas, junto con asesoramiento sobre calidad, estándares y estrategias de prueba
- Identificación de los roles organizacionales
- Guías para la gestión de proyectos y aseguramiento de la calidad
- Asesoramiento para la gestión de bibliotecas y reutilización

# METODOLOGÍAS ESTRUCTURADAS

- Proponen la creación de modelos del sistema que representan los procesos, los flujos y la estructura de los datos de una manera descendente
- Se pasa de una visión general del problema, nivel de abstracción alto, a un nivel de abstracción sencillo
- Esta visión se puede enfocar
  - Hacia un punto de vista funcional del sistema
    - Metodologías orientadas a procesos
  - Hacia la estructura de datos
    - Metodologías orientadas a datos

# METODOLOGÍAS ESTRUCTURADAS ORIENTADAS A PROCESOS

- La Ingeniería del *Software* se fundamenta en el modelo básico **entrada/proceso/salida** de un sistema
- Estas metodologías se enfocan fundamentalmente en la parte de **proceso**
- Utilizan un enfoque de descomposición descendente para evaluar los procesos del espacio del problema y los flujos de datos con los que están conectados
- Este tipo de metodologías se desarrolló a lo largo de los años 70
- Representantes de este grupo son las metodologías de análisis y diseño estructurado como
  - **Merise** (Tardieu et al., 1986)
  - **YSM** (*Yourdon Systems Method*) (Yourdon Inc., 1993)
  - **SSADM** (*Structured Systems Analysis and Design Method*) (Ashworth y Goodland, 1990)
  - **METRICA v.2.1** (MAP, 1995)
  - **METRICA v3.0** (Parcialmente) (MAP, 2001)

# METODOLOGÍAS ORIENTADAS A OBJETOS

- Se fundamentan en la integración de los dos aspectos de los sistemas de información: **datos** y **procesos**
- En este paradigma un sistema se concibe como un conjunto de objetos que se comunican entre sí mediante mensajes
- El objeto encapsula datos y operaciones
  - Este enfoque permite un modelado más natural del mundo real y facilita enormemente la reutilización del *software*
- Las metodologías orientadas a objetos acortan la distancia existente entre el espacio de conceptos (lo que los expertos o usuarios conocen) y el espacio de diseño e implementación

# METODOLOGÍAS ORIENTADAS A OBJETOS

Gran cantidad de representantes

- 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)

# METODOLOGÍAS ORIENTADAS A OBJETOS

## Evolución de las metodologías OO

### Metodologías de primera generación

OMT      RDD      Objectory      Booch (91)

### Metodologías de segunda generación

OMT 2      Syntropy  
Fusion      Moses      MEDEA  
Booch (94)

Métricas

Lenguajes Formales

Unificación,  
Estandarización

UML

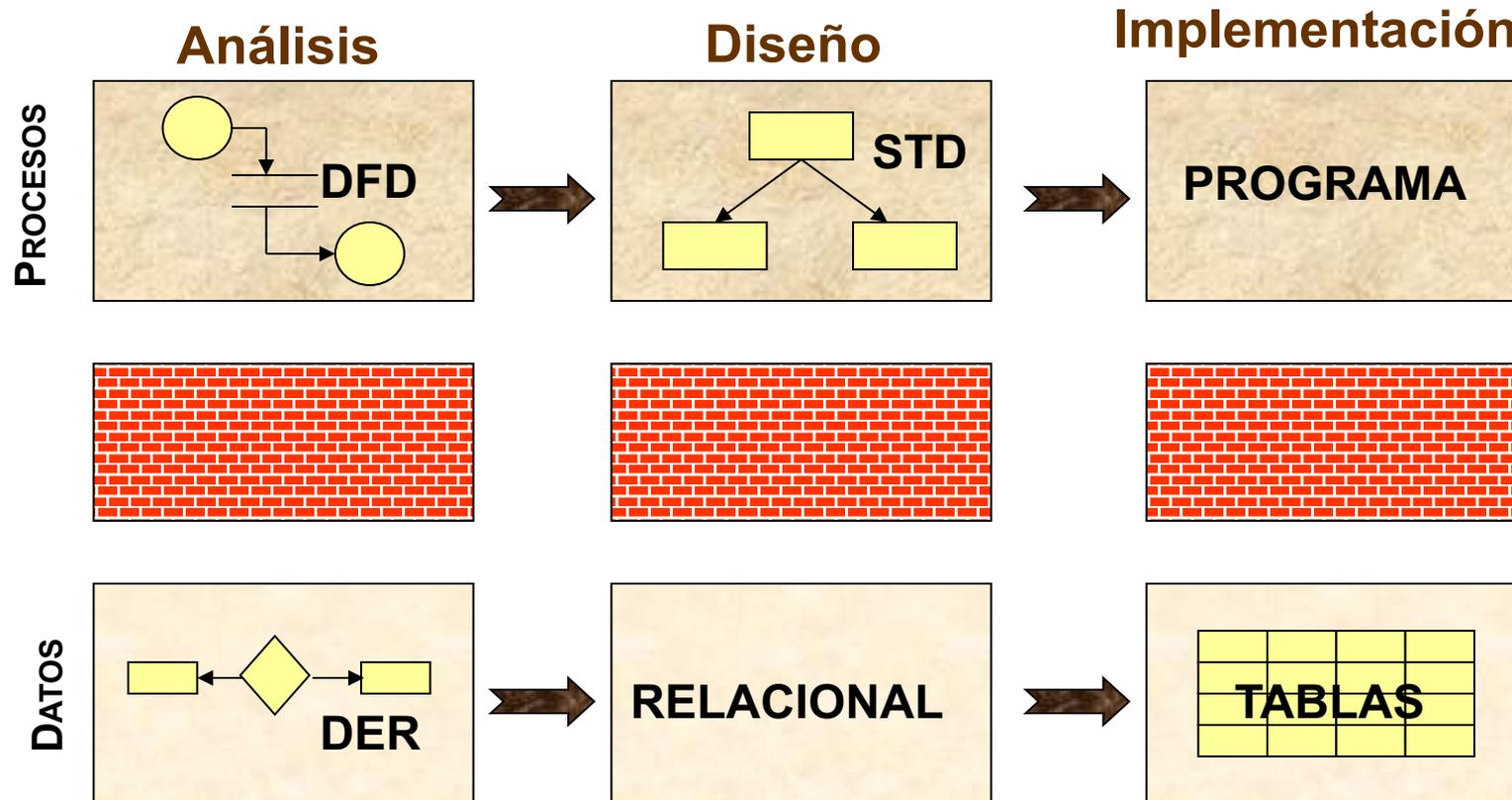
### Metodologías de tercera generación

OPEN  
UP

# METODOLOGÍAS ORIENTADAS A OBJETOS

Metodologías estructuradas vs. Metodologías Orientadas a Objetos

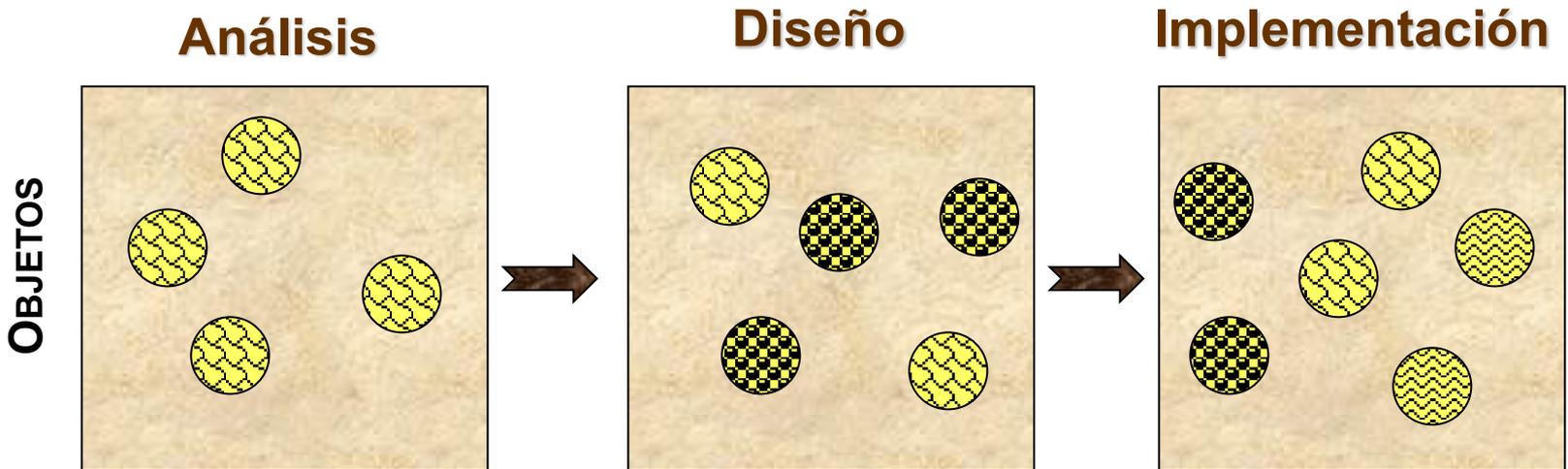
Metodologías estructuradas



# METODOLOGÍAS ORIENTADAS A OBJETOS

Metodologías estructuradas vs. Metodologías Orientadas a Objetos

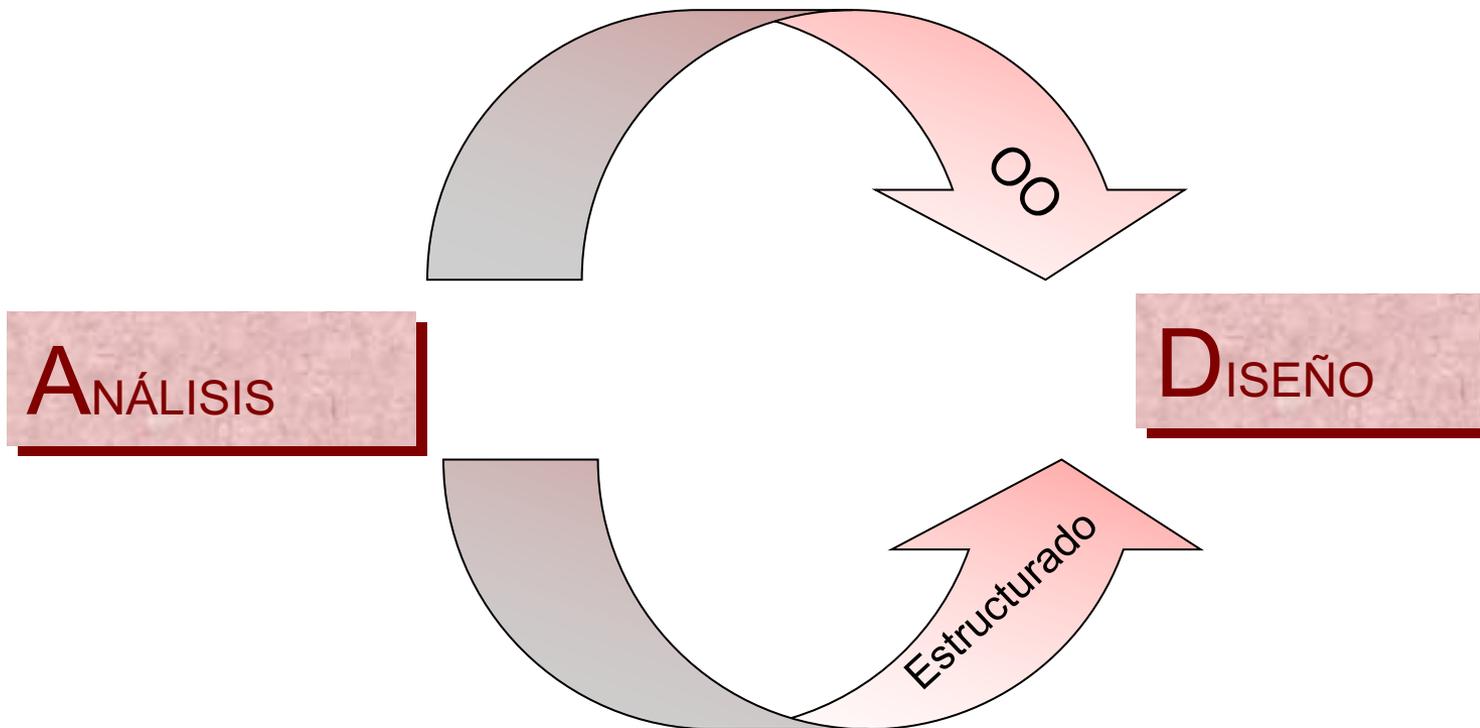
Metodologías Orientadas a Objetos



# METODOLOGÍAS ORIENTADAS A OBJETOS

Metodologías estructuradas vs. Metodologías Orientadas a Objetos

**Por Elaboración**



**Por Transformación**

# METODOLOGÍAS ÁGILES

Las aproximaciones ágiles emplean procesos técnicos y de gestión que continuamente se adaptan y se ajustan a (Turk et al., 2002)

- Cambios derivados de las experiencias ganadas durante el desarrollo
- Cambios en los requisitos
- Cambios en el entorno de desarrollo

La agilidad en el desarrollo del *software* no significa únicamente poner en el mercado o en explotación los productos *software* más rápidamente

- Esto choca frontalmente con los modelos de procesos tradicionales que son monolíticos y lentos, centrados en una única iteración o ciclo de larga duración

# AGILE MANIFESTO

*We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*

- *Individuals and interactions over processes and tools*
- *Working software over comprehensive documentation*
- *Customer collaboration over contract negotiation*
- *Responding to change over following a plan*

*While there is value in the items on the right, we value the items on the left more.*

## **Agile Manifesto**

# METODOLOGÍAS ÁGILES

La agilidad significa respuesta efectiva al cambio



pero incluye

- Estructuras de equipo y actitudes para facilitar la comunicación entre los miembros
- Énfasis en la entrega rápida de *software* funcional, para lo que resta importancia a los productos intermedios (lo cual siempre no es bueno)
- Adopción del cliente como parte del equipo de desarrollo
- Planificación incierta y, por tanto, flexible

# EXTREME PROGRAMMING

Controvertido enfoque de desarrollo de *software* basado en el modelo incremental

Está indicado para

- Equipos de tamaño mediano o pequeño
- Requisitos imprecisos y cambiantes

Características (Beck, 2000)

- El juego de la planificación
- Versiones pequeñas
- Metáfora
- Diseño sencillo
- Hacer pruebas
- Refactorización
- Programación en parejas
- Propiedad colectiva
- Integración continua
- Cliente in-situ
- Estándares de codificación

# SCRUM

- Propuesto por Jeff Sutherland y desarrollado por Schwaber y Beedle (Schwaber, 1995)
- El conjunto de buenas prácticas de Scrum se aplica esencialmente a la gestión de proyecto
- Es un *framework*, por lo que es necesaria su adaptación en cada organización o incluso en cada equipo
- El objetivo es obtener resultados rápidos con adaptación a los cambios de las necesidades de los clientes
- Las principales características de Scrum
  - El desarrollo *software* mediante iteraciones incrementales
  - Las reuniones a lo largo del proyecto

# SCRUM

**Scrum se basa en entregas parciales priorizadas por el beneficio que aporta al receptor final del producto**

# SCRUM

- Actividades estructurales
  - Requisitos
  - Análisis
  - Diseño
  - Evolución
  - Entrega
- Dentro de cada actividad las tareas se organizan con un patrón de proceso denominado **sprint**
- El trabajo del *sprint* se adapta al problema y se define y modifica en tiempo real por el *equipo Scrum*
- Uso de patrones de proceso de demostrada eficacia en proyectos críticos, con plazos cortos y requisitos cambiantes

# SCRUM

- Scrum define un ciclo de vida iterativo e incremental, que mejora la gestión del riesgo y aumenta la comunicación
- Se basa en tres pilares
  - Transparencia
    - Todos los aspectos del proceso que afectan al resultado son visibles para todos aquellos que administran dicho resultado
  - Inspección
    - Se debe controlar con la suficiente frecuencia los diversos aspectos del proceso para que puedan detectarse variaciones inaceptables en el mismo
  - Revisión
    - El producto debe estar dentro de los límites aceptables
    - En caso de desviación se procederá a una adaptación del proceso y del material procesado
    - Mecanismo de mejora continua, esto es, de control, para adaptarse y mejorar

# SCRUM

## Cada equipo Scrum tiene tres roles

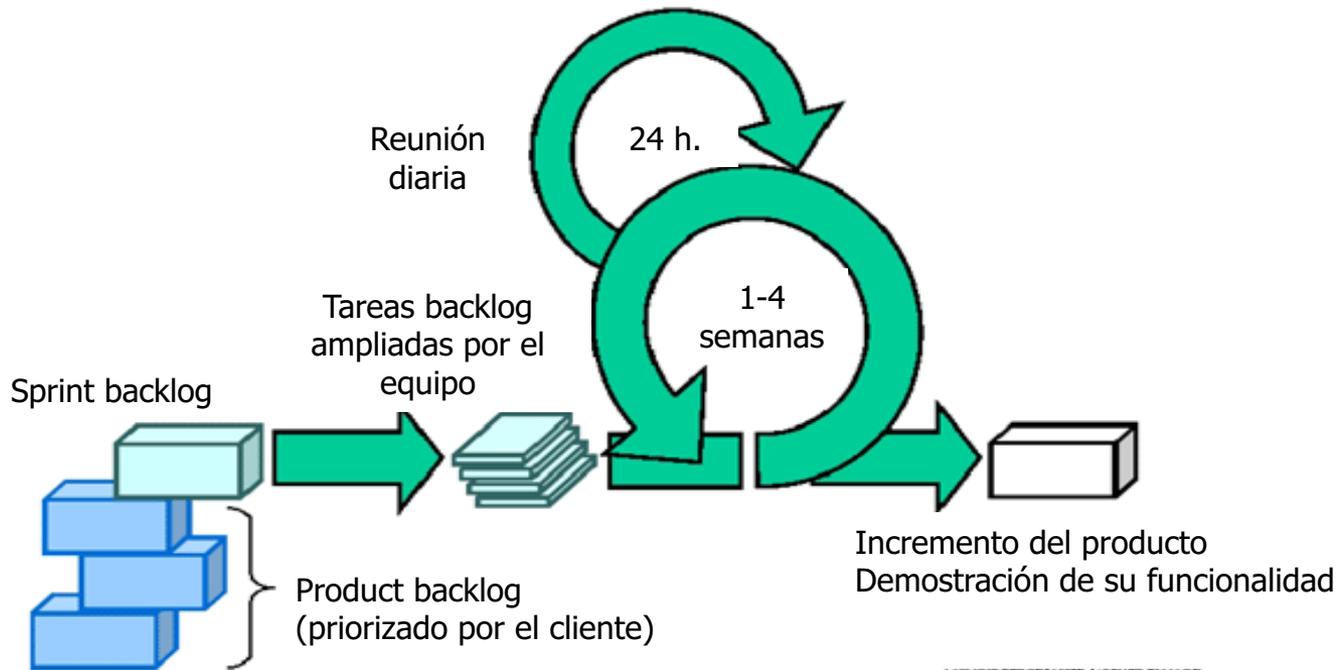
- *Scrum Master*. Es el responsable de asegurar que el equipo Scrum siga las prácticas de Scrum. Sus funciones
  - Ayudar a que el equipo y la organización adopten Scrum
  - Liderar el equipo Scrum para buscar la mejora en la productividad y la calidad de los entregables
  - Ayudar a la autogestión del equipo
  - Gestionar e intentar resolver los impedimentos con los que el equipo se encuentra para cumplir las tareas del proyecto
- *Product owner*. Es la persona responsable de gestionar las necesidades que se quieren satisfacer mediante el desarrollo del proyecto. Sus funciones
  - Recolectar las necesidades (historias de usuario)
  - Gestionar y ordenar las necesidades
  - Aceptar el producto *software* al finalizar cada iteración
  - Maximizar el retorno de la inversión del proyecto
- Equipo de desarrollo. Tiene las siguientes características
  - Autogestionado. El mismo equipo supervisa su trabajo (no existe el rol clásico de jefe de proyecto)
  - Multifuncional. Cada integrante del equipo debe ser capaz de realizar cualquier función
  - No distribuidos. Es conveniente que el equipo se encuentre en el mismo lugar físico
  - Tamaño óptimo. Al menos tres personas, máximo nueve, sin contar al *scrum master* ni al *product owner*

# SCRUM

## Acciones de los patrones de proceso

- **Retraso (pila de producto o product backlog)**: priorización de requisitos. Debe estar detallado de manera adecuada, estimado, emergente y priorizado
- **Sprints**: unidades de trabajo requeridas para alcanzar un requisito. Es cada iteración. Se recomiendan iteraciones cortas (1-4 semanas) y cuyo resultado será un producto *software* potencialmente entregable. El equipo de desarrollo selecciona las historias de usuario que se van a desarrollar en el *sprint* para conformar así la pila de *sprint* (*sprint Backlog*). La definición de cómo descomponer, analizar o desarrollar este *sprint backlog* queda a criterio del equipo de desarrollo. Además, la lista de tareas se mantendrá inamovible durante toda la iteración
- **Reuniones Scrum**: reuniones breves dirigidas por el *maestro Scrum*
- **Demostraciones preliminares**: entrega de un incremento al cliente

# SCRUM



# BIBLIOGRAFÍA

- F. J. García-Peñalvo y A. García-Holgado, "Introducción a la Ingeniería del Software," Recursos docentes de la asignatura Ingeniería de Software I. Grado en Ingeniería Informática. Curso 2017-2018, F. J. García-Peñalvo y A. García-Holgado, Eds., Salamanca, España: Grupo GRIAL, Universidad de Salamanca, 2018. [Online]. Disponible en: <https://goo.gl/9bVcWo>. doi: 10.5281/zenodo.1182457. (pp. 65-90)
- F. J. García-Peñalvo y A. García-Holgado, "Modelos de proceso," Recursos docentes de la asignatura Ingeniería de Software I. Grado en Ingeniería Informática. Curso 2017-2018, F. J. García-Peñalvo y A. García-Holgado, Eds., Salamanca, España: Grupo GRIAL, Universidad de Salamanca, 2018. [Online]. Disponible en: <https://goo.gl/xYYJGM>. doi: 10.5281/zenodo.1179286. (pp. 63-71)

# METODOLOGÍAS DE INGENIERÍA DE SOFTWARE

## INGENIERÍA DE SOFTWARE I

2º DE GRADO EN INGENIERÍA INFORMÁTICA  
CURSO 2017/2018

Francisco José García Peñalvo / fgarcia@usal.es

Alicia García Holgado / aliciagh@usal.es

Departamento de Informática y Automática

Universidad de Salamanca



UNIVERSIDAD  
DE SALAMANCA

CAMPUS OF INTERNATIONAL EXCELLENCE

