METODOLOGÍAS DE INGENIERÍA DE SOFTWARE

Imagen generada con Gemini Nano Banana Pro

INGENIERÍA DE SOFTWARE I

2º DE GRADO EN INGENIERÍA INFORMÁTICA
CURSO 2025/2026

Dr. Francisco José García-Peñalvo / fgarcia@usal.es
Dra. Alicia García-Holgado / aliciagh@usal.es
Dra. Andrea Vázquez-Ingelmo / andreavazquez@usal.es
Dr. Miguel Ángel Conde González / mconde@usal.es

Departamento de Informática y Automática
Universidad de Salamanca

MÁS INFORMACIÓN

Tema 1 – Introducción a la Ingeniería del Software [1]

Tema 3 – Modelos de proceso [2]

PÍLDORAS DE VÍDEO RELACIONADAS

Metodologías estructuradas y orientadas a objetos [3]

Metodologías ágiles [4]

Scrum [5]

https://bit.ly/3urdvyx

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 [6]

- 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 [7]

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 [8]

- 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** (1983) [9]
  - **YSM** (*Yourdon Systems Method*) (1993) [10]
  - **SSADM** (*Structured Systems Analysis and Design Method*) (1990) [11]
  - **METRICA v.2.1** (1995) [12]
  - **METRICA v3.0** (Parcialmente) (2001) [13]

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

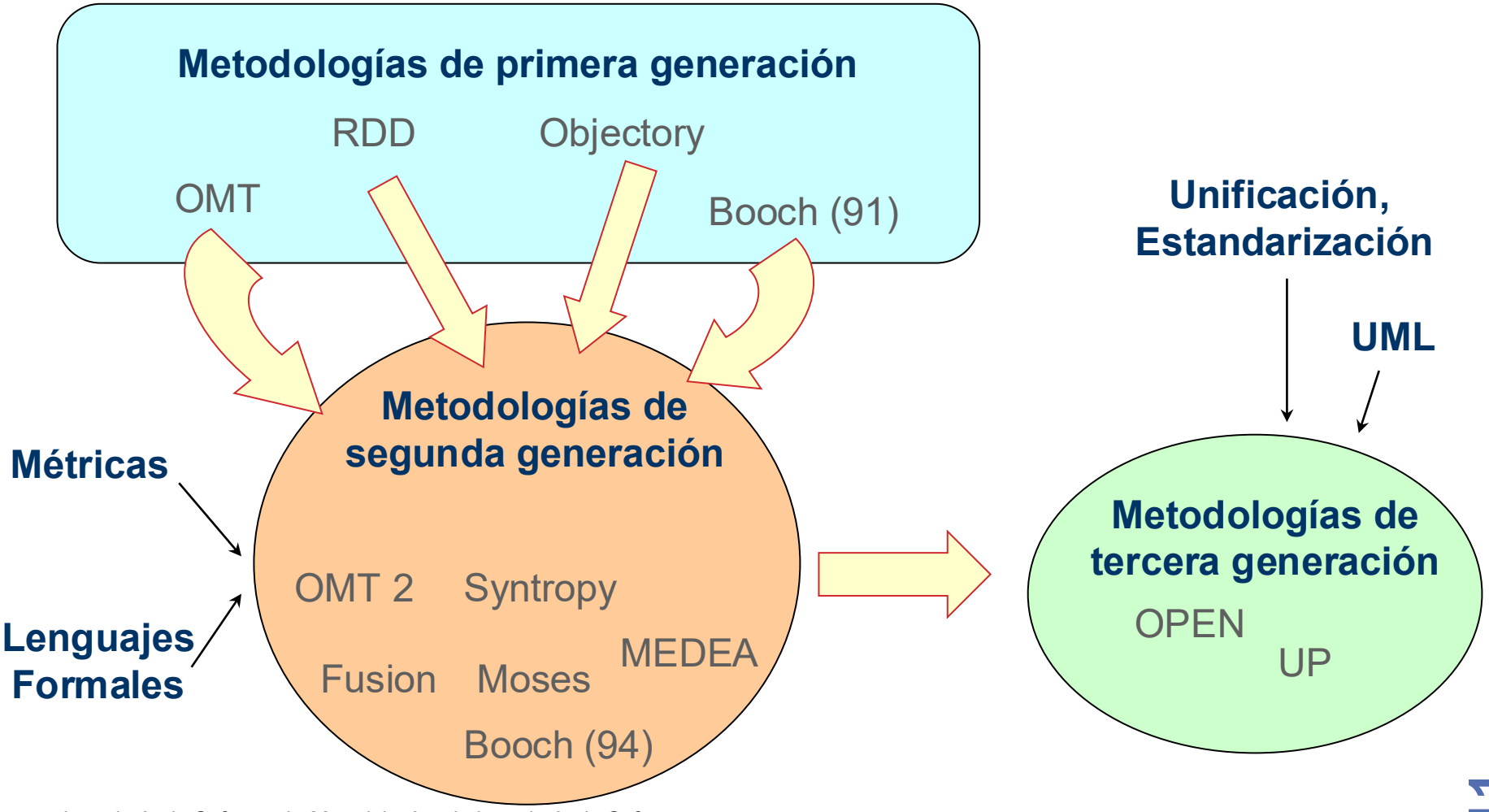
METODOLOGÍAS ORIENTADAS A OBJETOS

Gran cantidad de representantes

- Metodologías dirigidas por los datos
  - **OMT** (*Object Modeling Technique*) (1991) [7]
  - **Fusion** (1994) [14]
- Metodologías dirigidas por las responsabilidades
  - **RDD** (*Responsibility Driven Design*) (1990) [15]
  - **OBA** (*Object Behavior Analysis*) (1992) [16]
- Metodologías dirigidas por los casos de uso
  - **Objectory** (1992) [17]
  - **Proceso Unificado** (1999) [18]
- Metodologías dirigidas por estados
  - **Metodología de Shlaer y Mellor** (1992) [19]

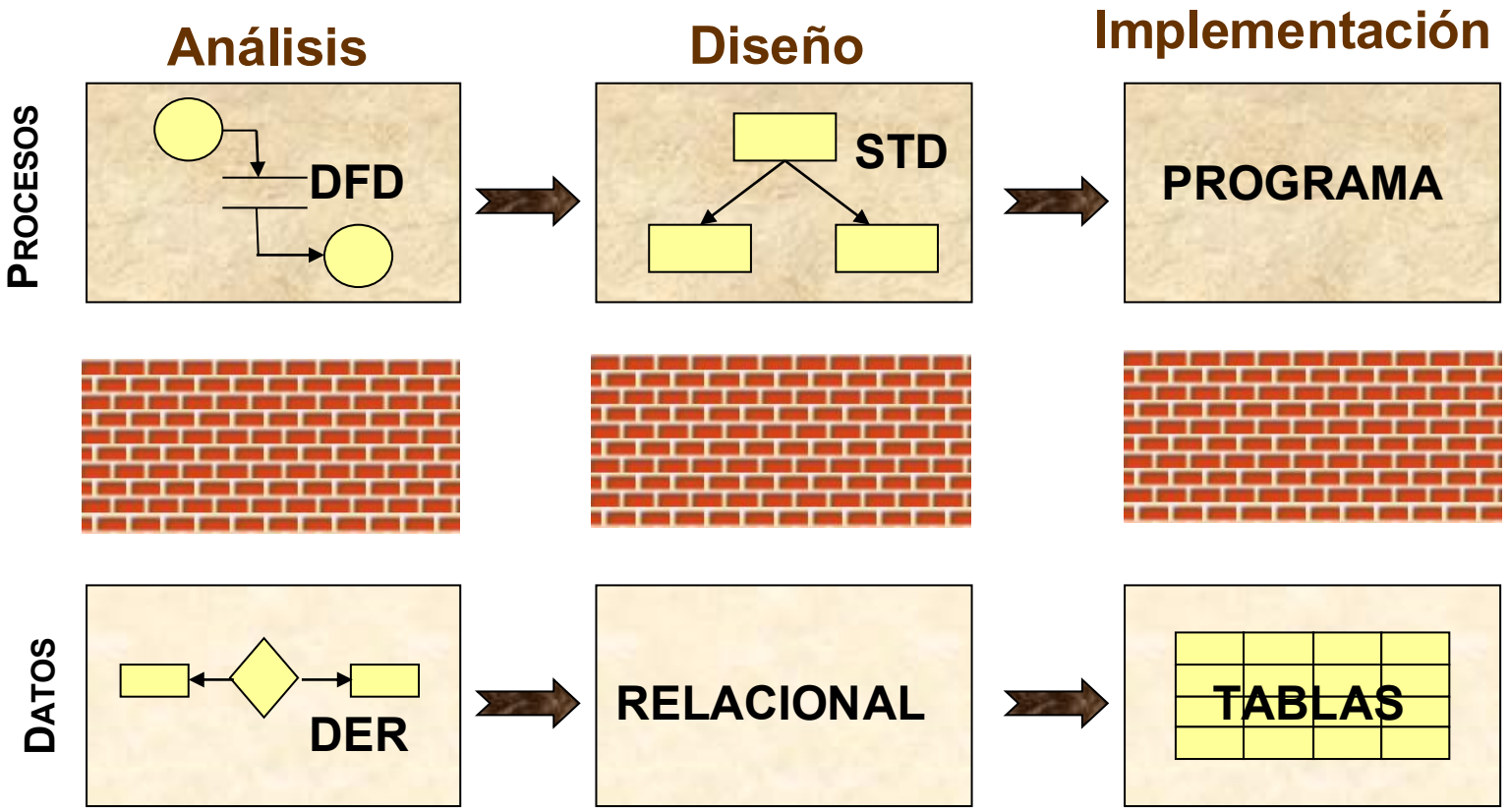
METODOLOGÍAS ORIENTADAS A OBJETOS

Evolución de las metodologías OO



METODOLOGÍAS ORIENTADAS A OBJETOS

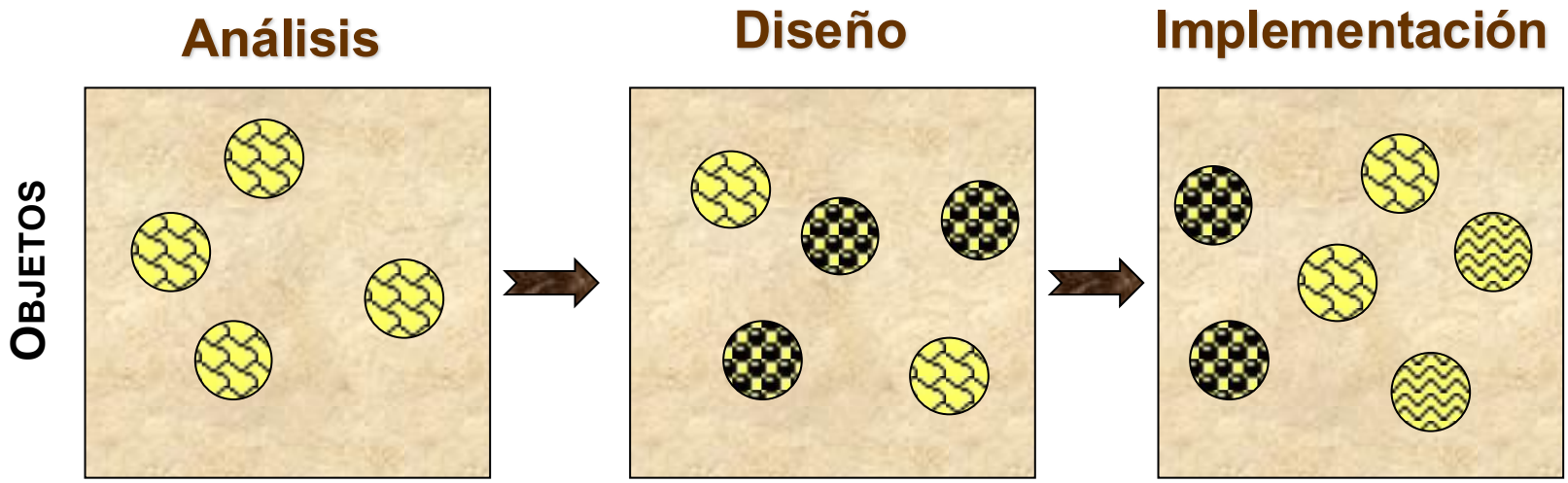
Metodologías estructuradas vs. Metodologías Orientadas a Objetos



Metodologías estructuradas

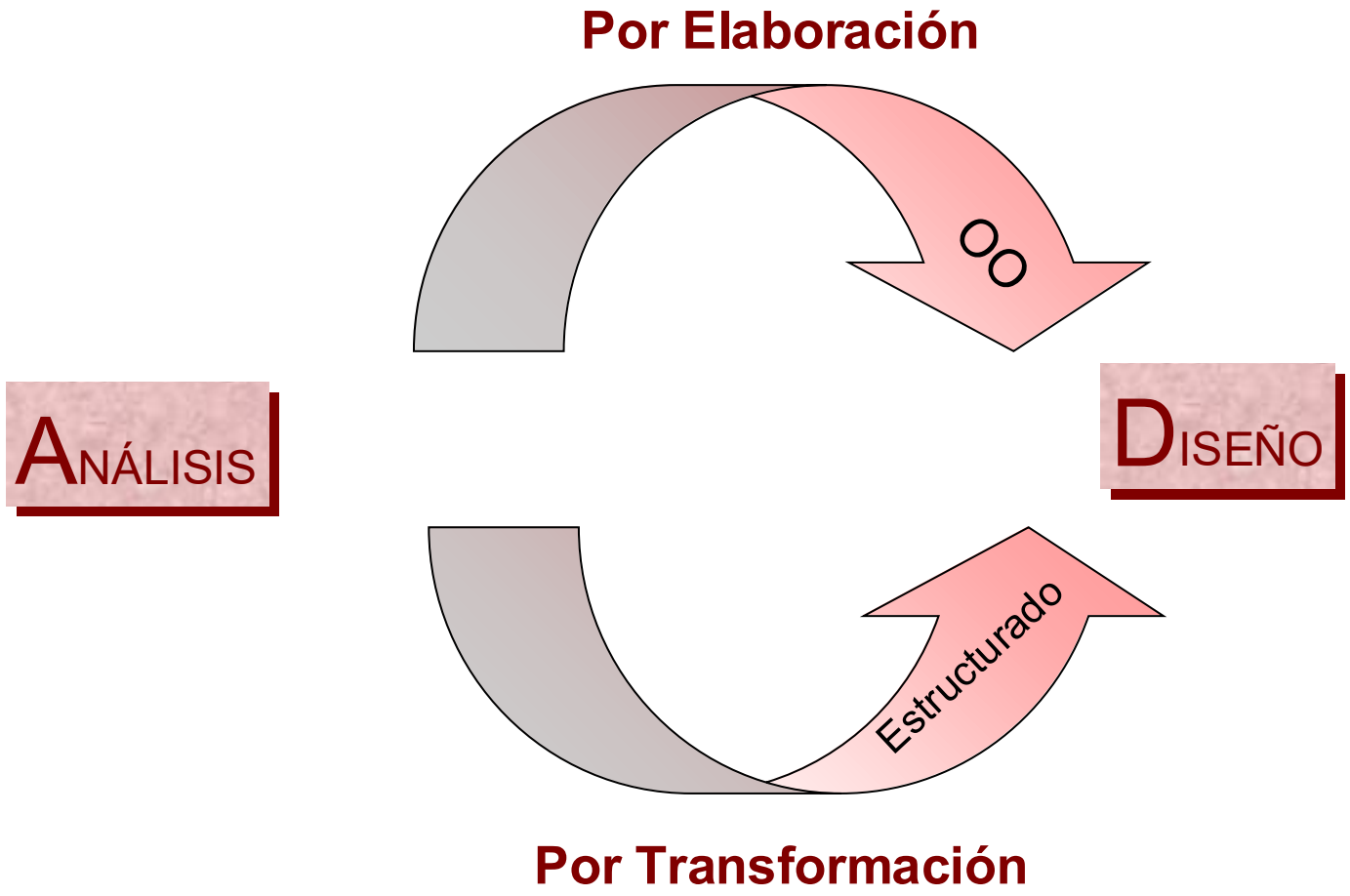
METODOLOGÍAS ORIENTADAS A OBJETOS

Metodologías estructuradas vs. Metodologías Orientadas a Objetos



METODOLOGÍAS ORIENTADAS A OBJETOS

Metodologías estructuradas vs. Metodologías Orientadas a Objetos



METODOLOGÍAS ÁGILES

Las aproximaciones ágiles emplean procesos técnicos y de gestión que continuamente se adaptan y se ajustan a [20]

- 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 [21]
https://agilemanifesto.org

METODOLOGÍAS ÁGILES

La agilidad significa respuesta efectiva al cambio

https://bit.ly/2LF8WNj

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 [22]

- 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 Shwaber y Beedle [23]
- El conjunto de buenas prácticas de Scrum se aplica esencialmente a la gestión del 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

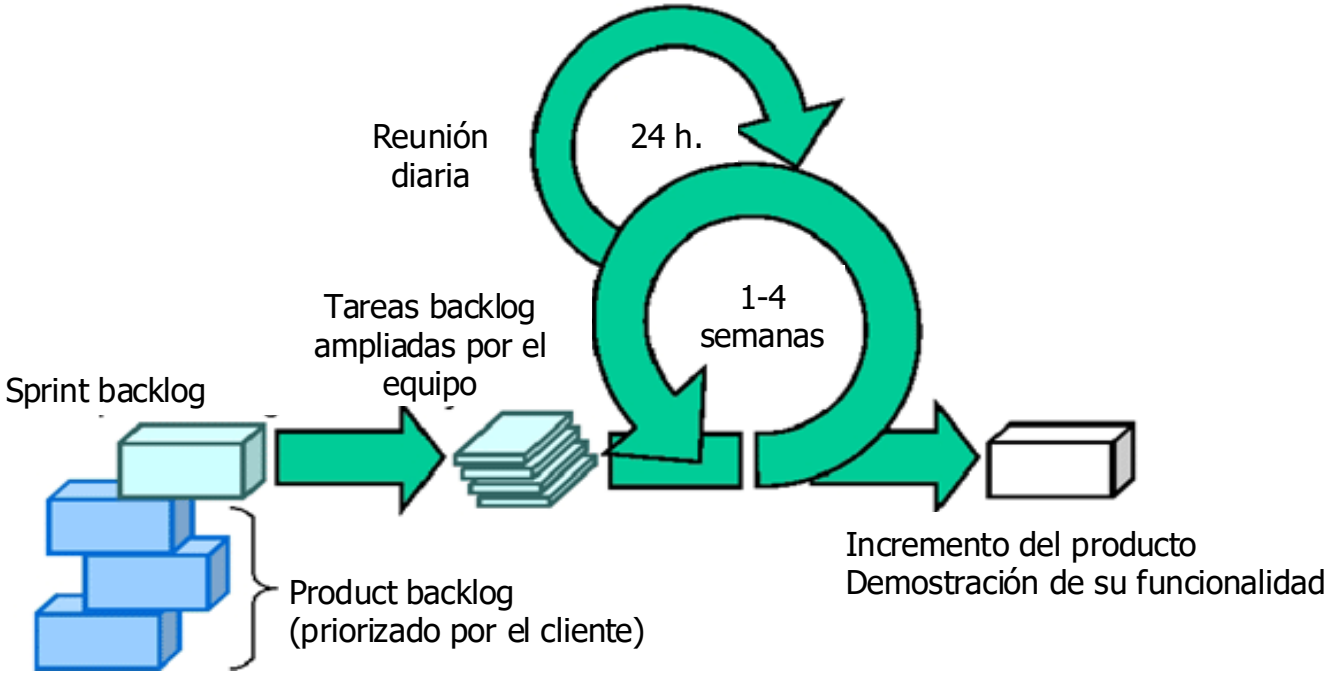
- *Scrum Master*. Es el responsable de asegurar que el equipo Scrum siga las prácticas de Scrum. Sus funciones
  - Ayudar a que el equipo y la organización adopten Scrum
  - Liderar el equipo Scrum para buscar la mejora en la productividad y la calidad de los entregables
  - Ayudar a la autogestión del 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


Reunión
diaria
24 h.
1-4
semanas
Tareas backlog
ampliadas por el
equipo
Sprint backlog
Product backlog
(priorizado por el cliente)
Incremento del producto
Demostración de su funcionalidad

REFERENCIAS

1. F. J. García-Peñalvo, A. García-Holgado, A. Vázquez-Ingelmo y M. Á. Conde-González, "Introducción a la Ingeniería del Software," Recursos docentes de la asignatura Ingeniería de Software I. Grado en Ingeniería Informática. Curso 2025-2026, F. J. García-Peñalvo, A. García-Holgado y M. Á. Conde-González, Eds., Salamanca, España: Grupo GRIAL, Universidad de Salamanca, 2026. [Online]. Disponible en. doi: 10.5281/zenodo.18544463.
2. F. J. García-Peñalvo, A. García-Holgado, A. Vázquez-Ingelmo y M. Á. Conde-González, "Modelos de proceso," Recursos docentes de la asignatura Ingeniería de Software I. Grado en Ingeniería Informática. Curso 2025-2026, F. J. García-Peñalvo, A. García-Holgado y M. Á. Conde-González, Eds., Salamanca, España: Grupo GRIAL, Universidad de Salamanca, 2026. [Online]. Disponible en. doi: 10.5281/zenodo.18546800.
3. F. J. García-Peñalvo, A. García-Holgado y A. Vázquez-Ingelmo, "Metodologías estructuradas y orientadas a objetos," Recursos docentes de la asignatura Ingeniería de Software I. Grado en Ingeniería Informática. Curso 2020-2021, F. J. García-Peñalvo, A. García-Holgado y A. Vázquez-Ingelmo, Eds., Salamanca, España: Grupo GRIAL, Universidad de Salamanca, 2021. [Online]. Disponible en: https://bit.ly/33k3CHo. doi: 10.5281/zenodo.5781241.
4. F. J. García-Peñalvo, A. García-Holgado y A. Vázquez-Ingelmo, "Metodologías ágiles," Recursos docentes de la asignatura Ingeniería de Software I. Grado en Ingeniería Informática. Curso 2020-2021, F. J. García-Peñalvo, A. García-Holgado y A. Vázquez-Ingelmo, Eds., Salamanca, España: Grupo GRIAL, Universidad de Salamanca, 2021. [Online]. Disponible en: https://bit.ly/3yofrbf. doi: 10.5281/zenodo.5781297.
5. F. J. García-Peñalvo, A. García-Holgado y A. Vázquez-Ingelmo, "Scrum," Recursos docentes de la asignatura Ingeniería de Software I. Grado en Ingeniería Informática. Curso 2020-2021, F. J. García-Peñalvo, A. García-Holgado y A. Vázquez-Ingelmo, Eds., Salamanca, España: Grupo GRIAL, Universidad de Salamanca, 2021. [Online]. Disponible en: https://bit.ly/3J1rnEP. doi: 10.5281/zenodo.5781344.
6. M. G. Piattini Velthius, J. A. Calvo-Manzano, J. Cervera Bravo y L. Fernández Sanz, *Análisis y Diseño de Aplicaciones Informáticas de Gestión. Una perspectiva de Ingeniería del Software*. Madrid, España: Ra-ma, 2004.

REFERENCIAS

7. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy y W. Lorensen, *Object-oriented modeling and design*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1991.
8. B. Henderson-Sellers y D. G. Firesmith, "Comparing OPEN and UML: The two third-generation OO development approaches," *Information and Software Technology,* vol. 41, no. 3, pp. 139-156, 1999. doi: 10.1016/S0950-5849(98)00127-X.
9. H. Tardieu, A. Rochfeld y R. Coletti, *La Méthode Merise. Tome 1. Principes et outils*. Paris: Editions d'Organisation, 1983.
10. Yourdon Inc., *Yourdon™ Systems Method. Model-driven systems development* (Yourdon press Computing Series). Englewood Cliffs, NJ, USA: Prentice Hall, 1993.
11. C. Ashworth y M. Goodland, *SSADM: A Practical Approach*. London, UK: McGraw-Hill, 1990.
12. Ministerio de las Administraciones Públicas, *Metodología Métrica 2.1. Volúmenes 1-3*. Madrid, España: Tecnos, 1995.
13. Ministerio de las Administraciones Públicas, *Métrica v3*, Madrid, España: Ministerio de las Administraciones Públicas, 2001. [Online]. Disponible en: https://d66z.short.gy/NhJWCJ.
14. D. Coleman, P. Arnold, S. Bodoff, C. Dolin, F. Hayes y P. Jeremaes, *Object-Oriented Development: The Fusion Method*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1994.
15. R. Wirfs-Brock, B. Wilkerson y L. Wiener, *Designing Object-Oriented Software*. Upper Saddle River, NJ, USA: Prentice-Hall, 1990.
16. K. S. Rubin y A. Goldberg, "Object Behavior Analysis," *Communications of the ACM,* vol. 35, no. 9, pp. 48-62, 1992. doi: 10.1145/130994.130996.
17. I. Jacobson, M. Christerson, P. Jonsson y G. Övergaad, *Object oriented software engineering: A use case driven approach*. Reading, MA, USA: Addison-Wesley, 1992.
18. I. Jacobson, G. Booch y J. Rumbaugh, *The Unified Software Development Process* (Object Technology Series). Reading, Massachusetts, USA: Addison Wesley, 1999.

REFERENCIAS

19. S. Shlaer y S. Mellor, *Object Life Cycles: Modeling the World in States*. Upper Saddle River, NJ, USA: Prentice-Hall, 1992.
20. D. Turk, R. France y B. Rumpe, "Limitations of Agile Software Processes," en *Proceedings of 4th International Conference on eXtreme Programming and Agile Processes in Software Engineering, XP2002. (Alghero, Sardinia, Italy, April 2002)* pp. 43-46, 2002.
21. K. Beck *et al.*, "Agile Manifesto," Agile Alliance, 2001, Disponible en: https://agilemanifesto.org/.
22. K. Beck, *Extreme Programming Explained. Embrace Change*. Boston, MA, USA: Addison-Wesley, 2000.
23. K. Schwaber, "SCRUM Development Process," en *Business Object Design and Implementation. OOPSLA '95 Workshop Proceedings 16 October 1995, Austin, Texas*, J. Sutherland, C. Casanave, J. Miller, P. Patel y G. Hollowell, Eds. pp. 117-134, London, UK: Springer London, 1997. doi: 10.1007/978-1-4471-0947-1_11.

METODOLOGÍAS DE INGENIERÍA DE SOFTWARE

Imagen generada con Gemini Nano Banana Pro

INGENIERÍA DE SOFTWARE I

2º DE GRADO EN INGENIERÍA INFORMÁTICA
CURSO 2025/2026

Dr. Francisco José García-Peñalvo / fgarcia@usal.es
Dra. Alicia García-Holgado / aliciagh@usal.es
Dra. Andrea Vázquez-Ingelmo / andreavazquez@usal.es
Dr. Miguel Ángel Conde González / mconde@usal.es

Departamento de Informática y Automática
Universidad de Salamanca