PROCESO

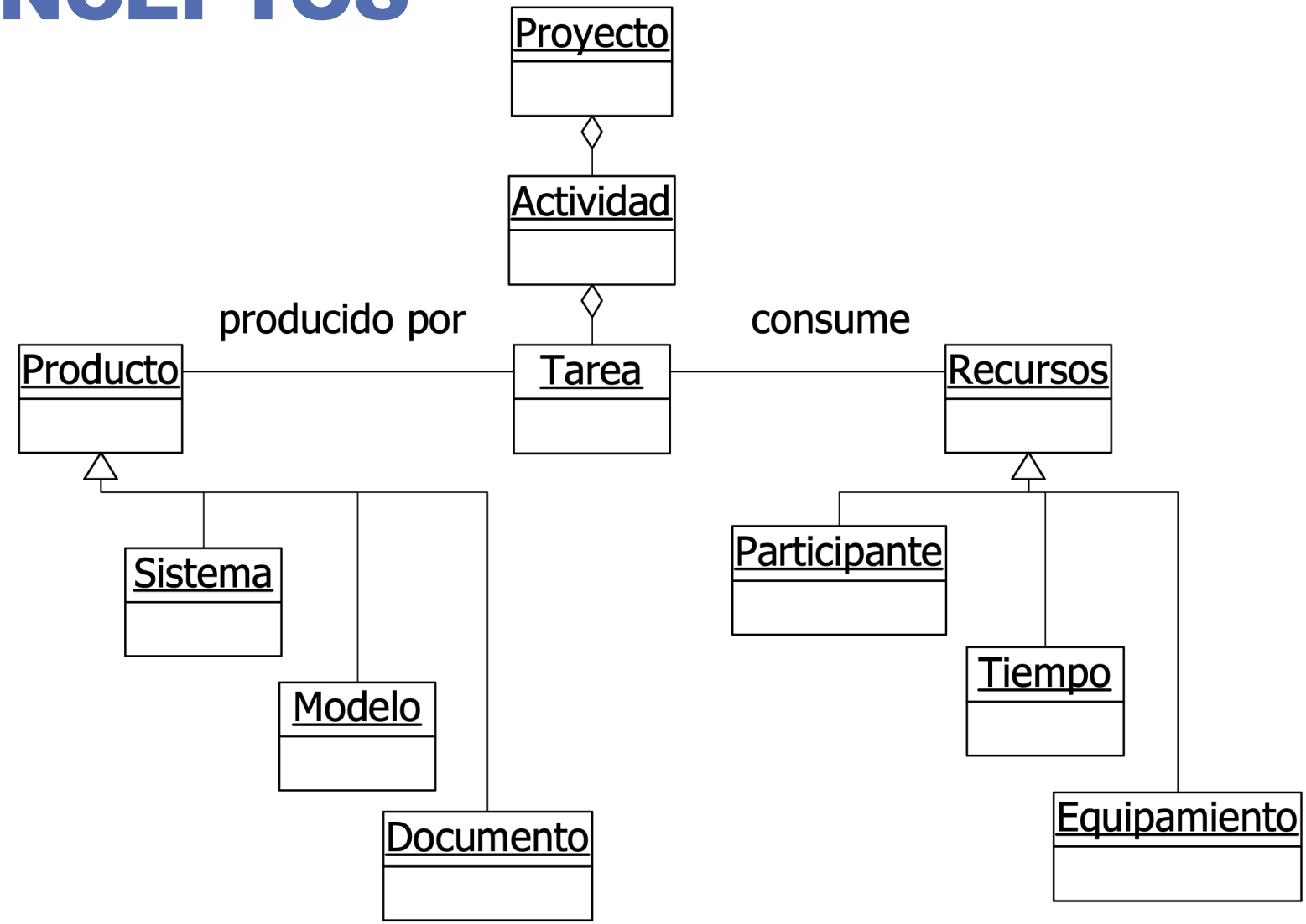
INGENIERÍA DE SOFTWARE I

2º DE GRADO EN INGENIERÍA INFORMÁTICA
CURSO 2020/2021

Francisco José García Peñalvo / fgarcia@usal.es
Alicia García Holgado / aliciagh@usal.es
Andrea Vázquez Ingelmo / andreavazquez@usal.es

Departamento de Informática y Automática
Universidad de Salamanca

CONCEPTOS


Proyecto
Actividad
producido por
consume
Producto
Tarea
Recursos
Sistema
Modelo
Documento
Participante
Tiempo
Equipamiento

CONCEPTOS

Proceso

- Define el marco de trabajo y permite un desarrollo racional y oportuno de la Ingeniería del *Software*

Método

- Indica cómo construir técnicamente el *software*. Se incluyen técnicas de modelado y otras técnicas descriptivas

Herramientas

- Proporcionan el soporte automático o semiautomático para el proceso y para los métodos

Notación

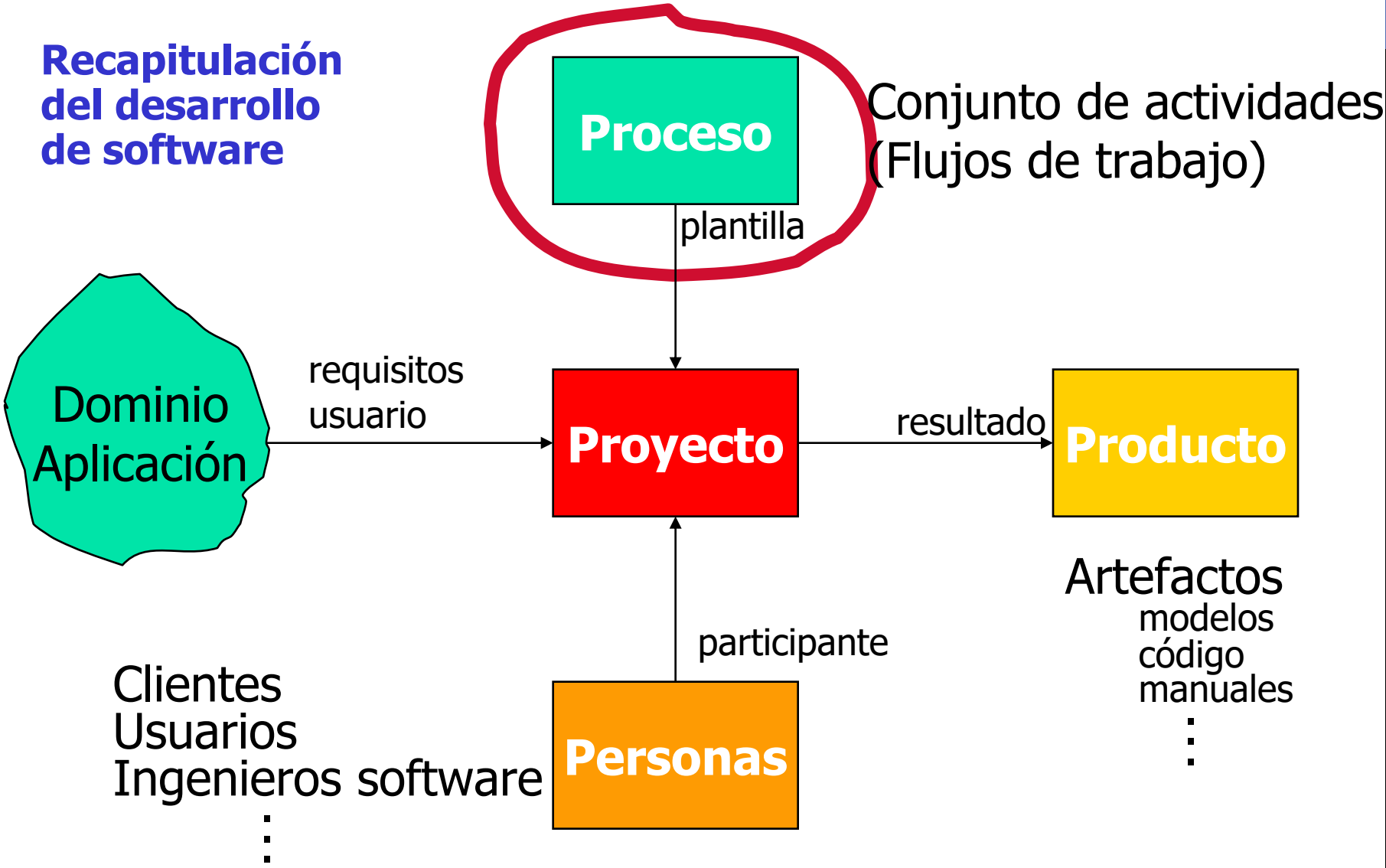
- Conjunto de reglas gráficas o textuales para la representación de un modelo

Metodología

- Colección de métodos para resolver un tipo de problemas
- Descompone el proceso de desarrollo en actividades y proporciona los métodos adecuados para llevar a cabo dichas actividades

CONCEPTOS

- **Recapitulación del desarrollo de software**



DEFINICIÓN DE PROCESO

Conjunto ordenado de actividades; una serie de pasos que involucran tareas, restricciones y recursos que producen una determinada salida esperada (Pfleeger, 2002)

Marco de trabajo de las tareas que se requieren para construir *software* de alta calidad (Pressman, 2010)

UN PROCESO SOFTWARE DEBE ESPECIFICAR

- La **secuencia de actividades** a realizar por el equipo de desarrollo
  - Flujo de actividades
- Los **productos** que deben crearse
  - Resultados del trabajo (modelos, documentos, datos informes...)
  - Qué y cuándo
- La **asignación de tareas** a cada miembro del equipo y al equipo como un todo
- Los **criterios** para controlar el proceso
  - Se establece el control de gestión de los proyectos *software*
  - Establecimiento de hitos
- Las posibles **heurísticas**

IMPORTANCIA DEL PROCESO

- Facilita la gestión del proyecto
- Establece una división del trabajo
- Facilita la comunicación de los miembros del equipo
- Permite la reasignación y la reutilización de personal especializado
  - Transferencia entre proyectos
- Mejora la productividad y el desarrollo
  - El desarrollo es reproducible
- Establece el contexto en el que se aplican los métodos técnicos
- Gestiona el cambio adecuadamente
- Asegura la calidad

CICLO DE VIDA DEL SOFTWARE

Cuando un proceso implica la construcción de algún producto, suele referirse al proceso como un **ciclo de vida**

- El proceso de desarrollo de *software* suele denominarse **ciclo de vida del *software***

La evolución del *software* representa el ciclo de actividades involucradas en el desarrollo, uso y mantenimiento de sistemas *software* (Scacchi, 1988)

Los **proyectos *software*** se desarrollan en una serie de **fases**

- Van desde la **concepción del *software*** y su desarrollo inicial hasta su **puesta en funcionamiento** y **posterior retirada** por otra nueva generación de *software*
- Estas fases pueden ser
  - Temporales
    - Forman una secuencia en el tiempo
  - Lógicas
    - Cuando representan pasos o etapas que no constituyen una secuencia temporal

CICLO DE VIDA DEL SOFTWARE

Ciclo de vida

Las distintas fases por las que pasa el *software* desde que nace una necesidad de mecanizar un proceso hasta que deja de utilizarse el *software* que sirvió para ese objetivo, pasando por las fases de desarrollo y explotación (Frakes et al., 1991)

Ciclo de desarrollo

El período de tiempo que comienza con la decisión de desarrollar un producto *software* y finaliza cuando se ha entregado este. Este ciclo incluye, en general, una fase de requisitos, una fase de diseño, una fase de implantación, una fase de pruebas, y a veces, una fase de instalación y aceptación (AECC, 1986)

MODELO DE PROCESO

Un modelo de proceso *software* es una representación abstracta de un proceso *software* (Sommerville, 2005)

Hay varios modelos de procesos definidos en la bibliografía de Ingeniería del *Software*

Cada modelo de proceso representa un proceso desde una perspectiva particular, por lo que solo ofrece una información parcial sobre dicho proceso

Los modelos de proceso genéricos, también llamados paradigmas de proceso

- Presentan un proceso desde una perspectiva arquitectónica, es decir, ofrecen un marco de definición para el proceso, pero no detallan las actividades específicas
- No son descripciones definitivas de los procesos *software*, más bien son abstracciones útiles que se utilizan para explicar diferentes aproximaciones al desarrollo del *software*

MODELO GENERAL DE PROCESO EN INGENIERÍA

Especificación
- Formulación de los requisitos y restricciones del sistema

Diseño
- Elaboración de un documento con el modelo del sistema

Fabricación
- Construcción del sistema

Prueba
- Comprobación de que el sistema cumple las especificaciones requeridas

Instalación
- Entrega del sistema al cliente y garantía de que es operativo

Mantenimiento
- Reparación de los fallos que aparecen en el sistema

MODELO GENERAL DE PROCESO EN INGENIERÍA DE SOFTWARE

En el proceso de construcción de sistemas informáticos se pueden distinguir tres fases genéricas

- La definición (Análisis de Sistemas; Análisis de Requisitos)
- El desarrollo (Diseño; Codificación; Prueba)
- El mantenimiento (Correctivo; Adaptativo; Perfectivo; Preventivo)

ESTÁNDAR ISO/IEC/IEEE 12207:2017

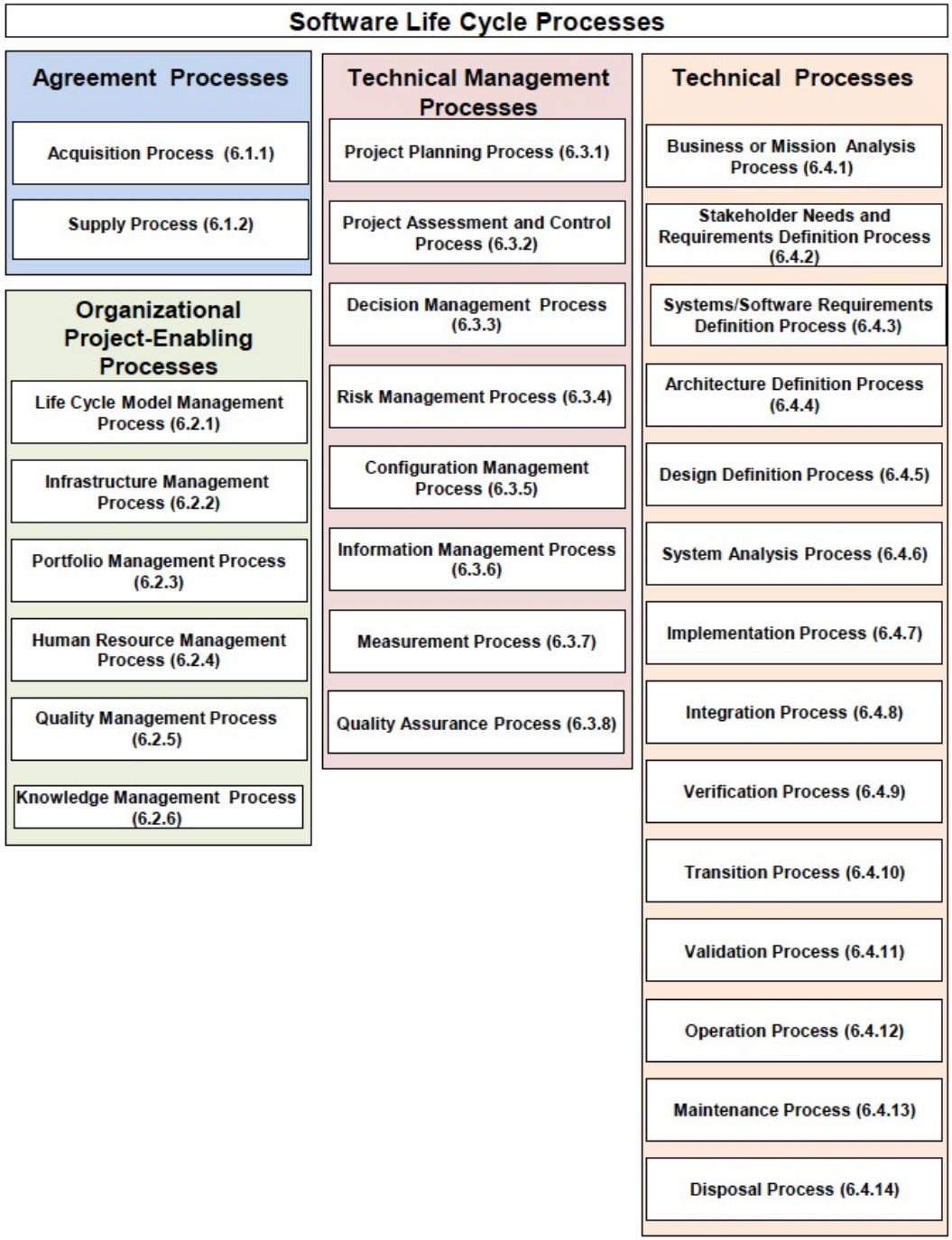
El estándar ISO/IEC/IEEE 12207:2017 (ISO/IEC/IEEE, 2017) relativo a los procesos del ciclo de vida del *software*

- Se aplica a la adquisición de sistemas de *software*, productos y servicios, al suministro, desarrollo, operación, mantenimiento y eliminación de productos de *software* o componentes de *software* de cualquier sistema, ya sea que se realice interna o externamente a una organización
- Se incluyen aquellos aspectos de la definición del sistema necesarios para proporcionar el contexto de los productos y servicios de *software*
- También proporciona procesos que pueden emplearse para definir, controlar y mejorar los procesos del ciclo de vida del *software* dentro de una organización o de un proyecto

ESTÁNDAR ISO/IEC/IEEE 12207:2017

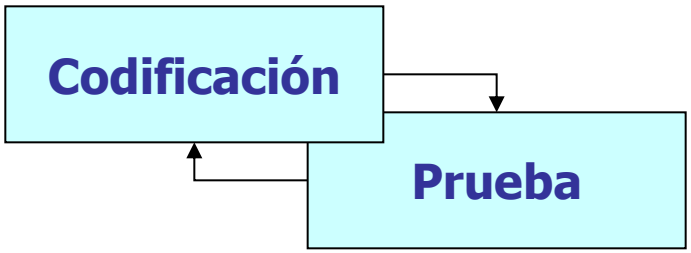
- Esta norma no fomenta o especifica ningún modelo concreto de ciclo de vida, gestión del *software* o método de ingeniería, ni prescribe cómo realizar ninguna de las actividades

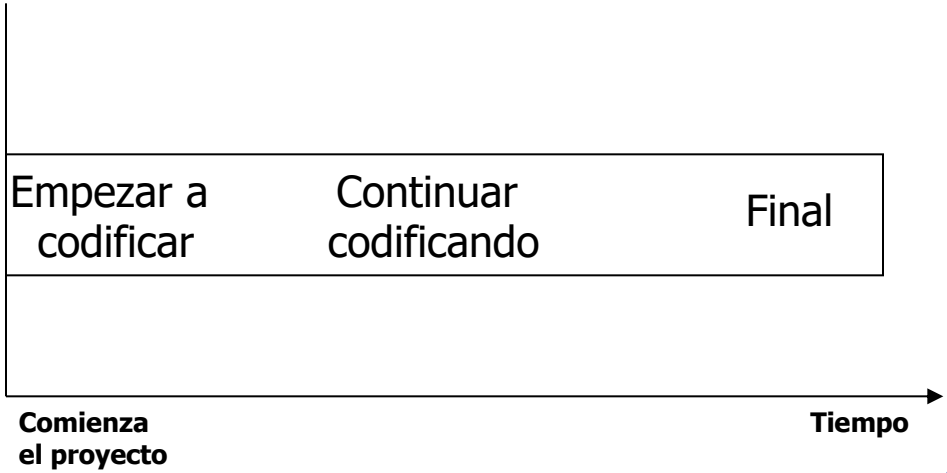
ESTÁNDAR ISO/IEC/IEEE 12207:2017



MODELO PRIMITIVO

- Se le conoce también con el nombre de **Modelo Prueba y Error** o **Modelo Codifica y Mejora**
- Proceso de desarrollo aplicado en las primeras experiencias de programación
- Supone una iteración de fases codificación-depuración sin ninguna planificación ni diseños previos





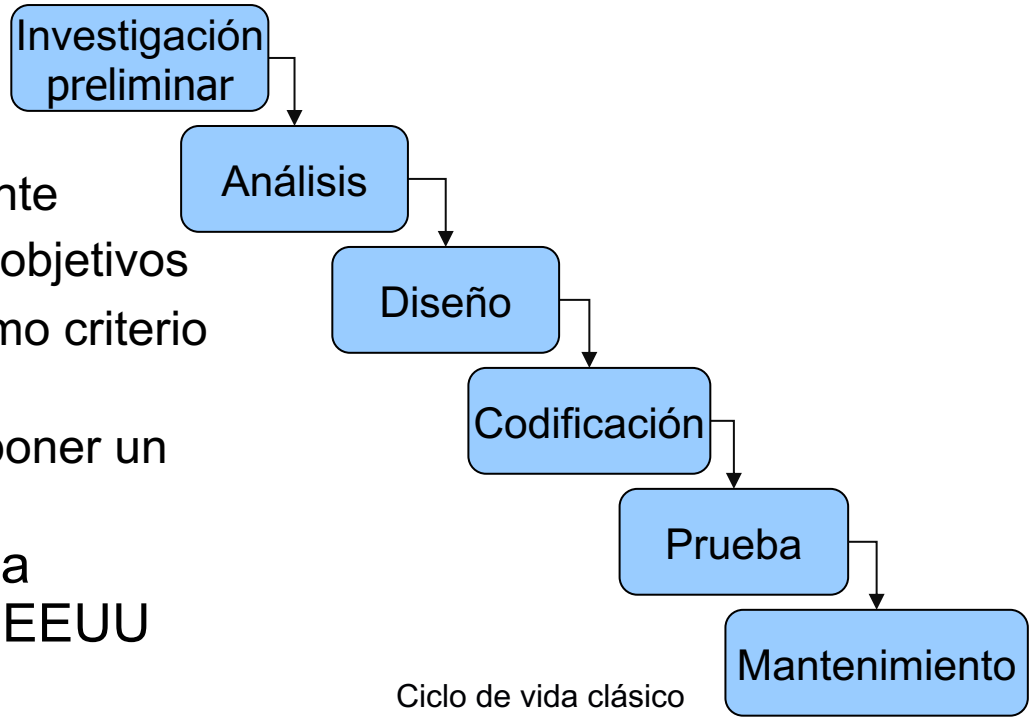
MODELO PRIMITIVO

Inconvenientes

- Código pobremente estructurado tras varias iteraciones
    - Código espagueti / Código pizza
- Caro de desarrollar por las numerosas recodificaciones
- Posible rechazo del usuario al no existir análisis de requisitos
- Caro de depurar por la falta de planificación
- Caro de mantener por la falta de estructura y documentación

MODELO CLÁSICO

- Conocido también como modelo lineal o “en cascada”
- Versión original se debe a W. Royce (Royce, 1970), pero aparecen después numerosos refinamientos
- Características
  - Está compuesto por una serie de fases que se ejecutan secuencialmente
  - Paso de fase al conseguir los objetivos
  - Obtención de documentos como criterio de finalización de fase
  - El final de una fase puede suponer un punto de revisión
- Se encuentra definido en la norma estándar 2167-A del DoD de EEUU

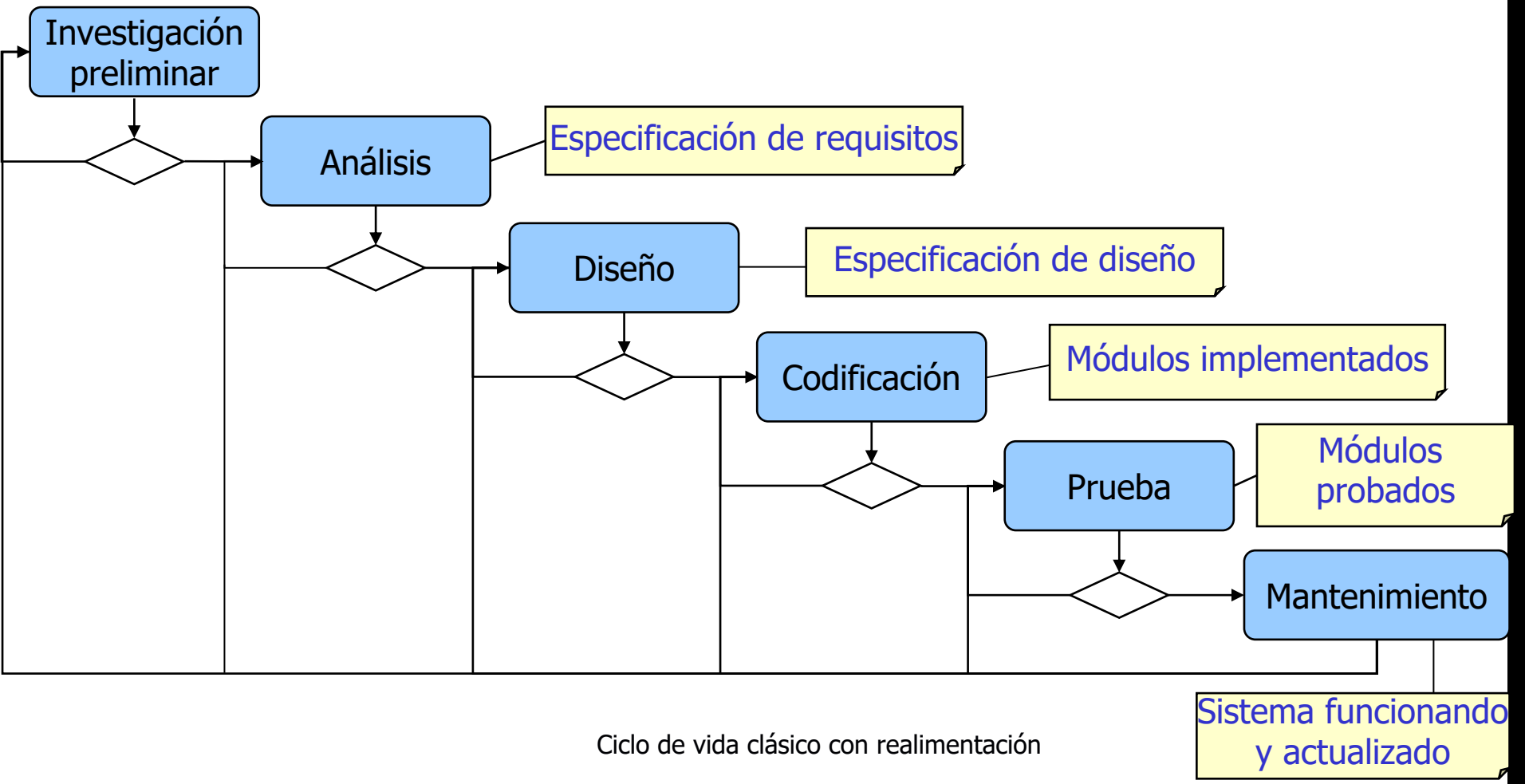


Ciclo de vida clásico

MODELO CLÁSICO

- Apoyo a los gestores
- Distintas configuraciones
    - Muchos modelos más complejos son variaciones del modelo en cascada que incorporan lazos de realimentación y fases adicionales
- Modelo satisfactorio solo en desarrollos conocidos y estables
- El desconocimiento y el riesgo suele ser alto en el desarrollo del *software*
    - Desconocimiento de las necesidades por parte del cliente
    - Incomprensión de las necesidades por parte del proveedor
    - Inestabilidad de las necesidades
    - Opciones tecnológicas
    - Movimientos de personal
- La linealidad no se corresponde con la realidad
    - Los retornos de información entre las fases se hacen necesarios para incorporar correcciones hacia arriba, en función de los descubrimientos realizados hacia abajo

MODELO CLÁSICO



Ciclo de vida clásico con realimentación

MODELO CLÁSICO

- Inconvenientes
  - Su progresión secuencial o lineal no refleja la manera en que realmente se desarrolla el *software* (Pfleeger, 2002; Pressman, 2006)
  - Es un modelo que adolece de rigidez (Pressman, 2010)
    - Exige al usuario que exponga explícitamente todos los requisitos al principio, presentando problemas para gestionar la incertidumbre natural propia del comienzo de la mayoría de los proyectos
  - Se tarda mucho tiempo en pasar por todo el ciclo (Piattini et al., 2004)
  - Es un modelo monolítico (Pressman, 2010)
    - Hasta llegar a las etapas finales del desarrollo no habrá una versión operativa del programa, lo que influye negativamente en el descubrimiento a tiempo de errores o incongruencias en los requisitos
  - Impone una estructura de gestión de proyecto al desarrollo del sistema (McCracken y Jackson, 1981)
  - No trata al *software* como un proceso de resolución de problemas (Curtis et al., 1987)

MODELO CLÁSICO

- Consideraciones finales
  - Tiene un lugar destacado en la Ingeniería del *Software*
  - Proporciona una plantilla para adecuar los métodos
  - Es muy utilizado
  - Tiene problemas pero es mejor que desarrollar sin guías

MODELOS BASADOS EN PROTOTIPOS

Un prototipo es un modelo experimental de un sistema o de un componente de un sistema que tiene los suficientes elementos que permiten su uso

Objetivos

- Son un medio eficaz para aclarar los requisitos de los usuarios e identificar las características de un sistema que deben cambiarse o añadirse
- Mediante el prototipo se puede verificar la viabilidad del diseño de un sistema

Características

- Es una aplicación que funciona
- Su finalidad es probar varias suposiciones con respecto a las características requeridas por el sistema
- Se crean con rapidez
- Evolucionan a través de un proceso iterativo
- Tienen un costo bajo de desarrollo

MODELOS BASADOS EN PROTOTIPOS

- **Enfoques de desarrollo**
  - **Desechable**: El prototipo es una versión rudimentaria del sistema que posteriormente es desechada
  - **Evolutivo**: El prototipo debe convertirse, eventualmente, en el sistema final usado (alternativa al ciclo de vida) (Basili y Turner, 1975)
  - **Mixto** (prototipado operativo) (Davis, 1993)
    - Se aplican técnicas convencionales para los requisitos bien conocidos
    - Combinación de prototipos desechables y evolutivos para los requisitos poco conocidos

	DESECHABLE	EVOLUTIVO
Enfoque de desarrollo	Rápido y sin rigor	Riguroso
Qué construir	Sólo las partes problemáticas	Primero las partes bien entendidas. Sobre una base sólida.
Directrices del diseño	Optimizar el tiempo de desarrollo	Optimizar la modificabilidad
Objetivo último	Desecharlo	Incluirlo en el sistema

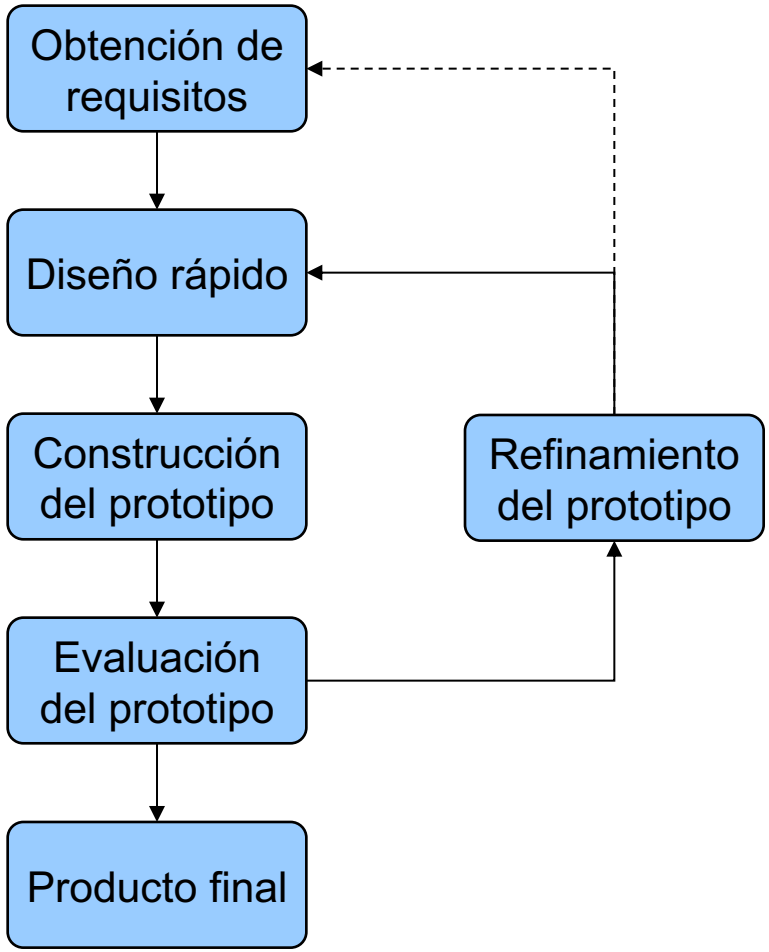
Diferencias entre los prototipos desechables y evolutivos

PROTOTIPOS DESECHABLES

- Características
  - Se desarrolla código para explorar factores críticos para el éxito del sistema
  - La implementación usa lenguajes y/o métodos de desarrollo más rápidos que los definitivos
  - Se usa como herramienta auxiliar de la especificación de requisitos y el diseño
    - Determinar la viabilidad de los requisitos
    - Validar la funcionalidad del sistema
    - Encontrar requisitos ocultos
    - Determinar la viabilidad de la interfaz de usuario
    - Examinar alternativas de diseño
    - Validar una arquitectura de diseño particular
  - Este enfoque suele derivar en un modelo lineal una vez que el prototipo ha cumplido su misión

PROTOTIPOS DESECHABLES

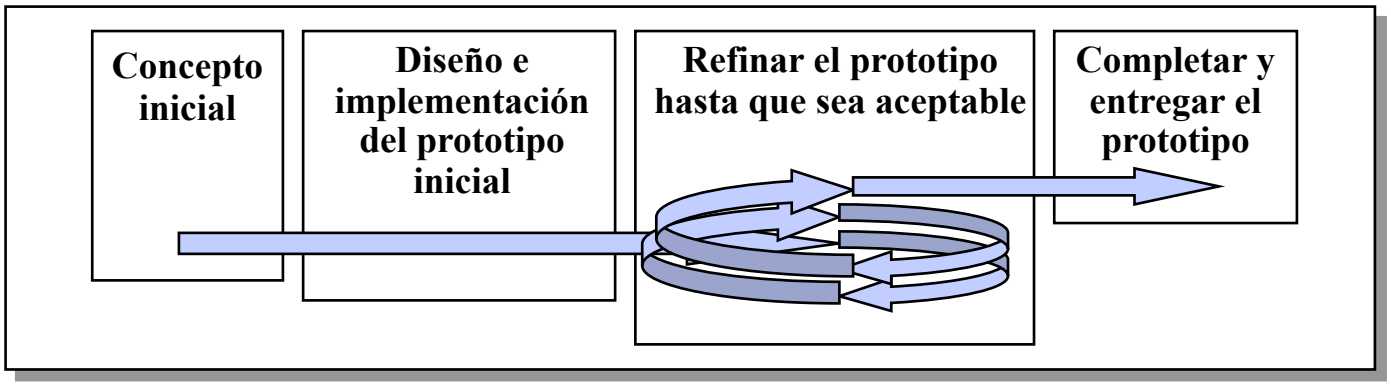
- Características
  - Idea del *software* en líneas generales desde el punto de vista del usuario
  - Idealmente sirve para identificar los requisitos del *software*
    - Introduce cierta flexibilidad en la introducción de requisitos
  - Proceso iterativo
    - La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo a la vez que el desarrollador comprenda mejor lo que necesita hacer



Modelo de prototipos

PROTOTIPADO EVOLUTIVO (CICLO DE VIDA ITERATIVO)

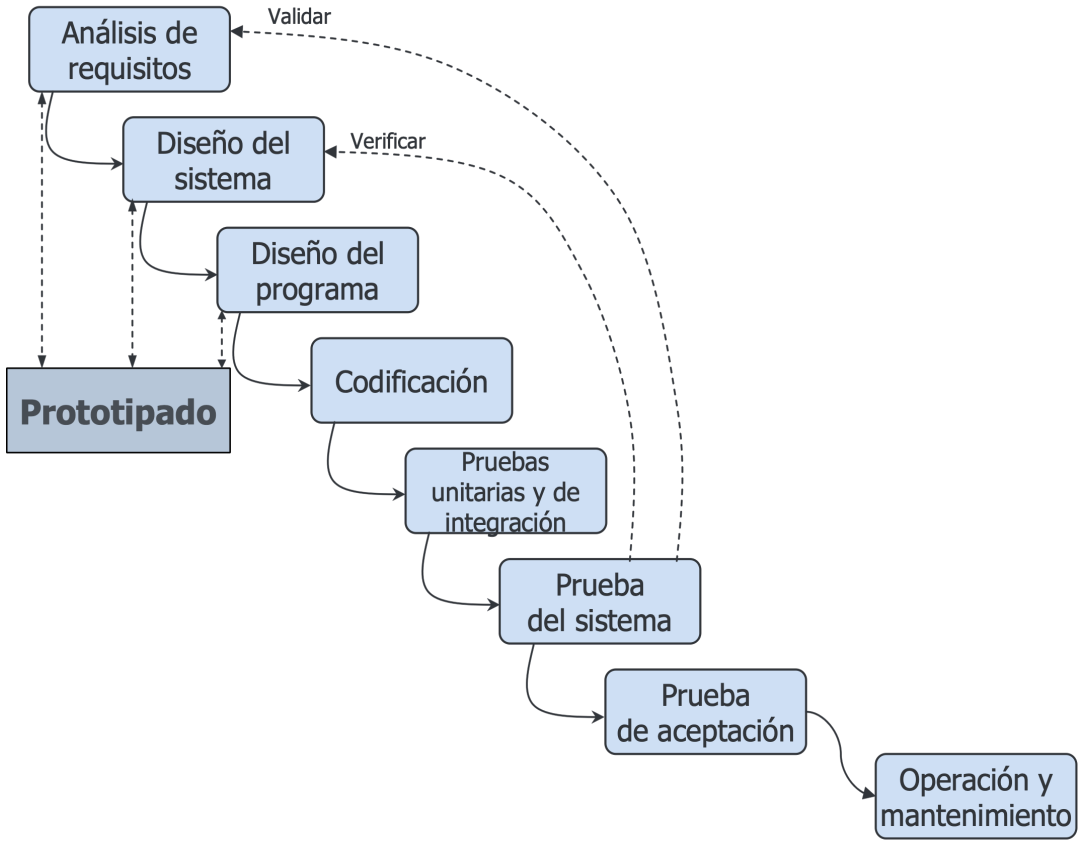
- Características
  - Enfoque de desarrollo que se utiliza cuando no se conoce con seguridad lo que se quiere construir
  - Se comienza diseñando e implementando las partes más destacadas del sistema
  - La evaluación del prototipo proporciona la realimentación necesaria para aumentar y refinar el prototipo
  - El prototipo evoluciona y se transforma en el sistema final



Modelo de prototipado evolutivo

VENTAJAS DEL PROTOTIPADO

- Permite solventar objeciones del usuario
- Sirve para formalizar la aceptación previa
- Introduce flexibilidad en la captura de requisitos
- Es útil cuando el área de aplicación no está definida, cuando el riesgo de rechazo el alto, o como forma de evaluar el impacto de una aplicación
- El prototipado es un subproceso que puede incluirse como parte de otros modelos de proceso
    - Por ejemplo puede combinarse con un ciclo en cascada para intentar solventar ciertas carencias de este



Modelo en cascada con prototipado (Pfleeger, 2002)

INCONVENIENTES DEL PROTOTIPADO

- El sistema se puede llegar a deteriorar tendiendo hacia el modelo primitivo
- Se suele refinar el prototipo hacia el sistema final en lugar de desecharlo y empezar desde el principio
- El cliente puede encontrar atractivo el prototipo y quedarse con el prototipo como sistema final
- Relajación de los desarrolladores
- No disminuye el tiempo entre la definición de los requisitos y la entrega del producto
- Al usuario le desagrada que se deseche código

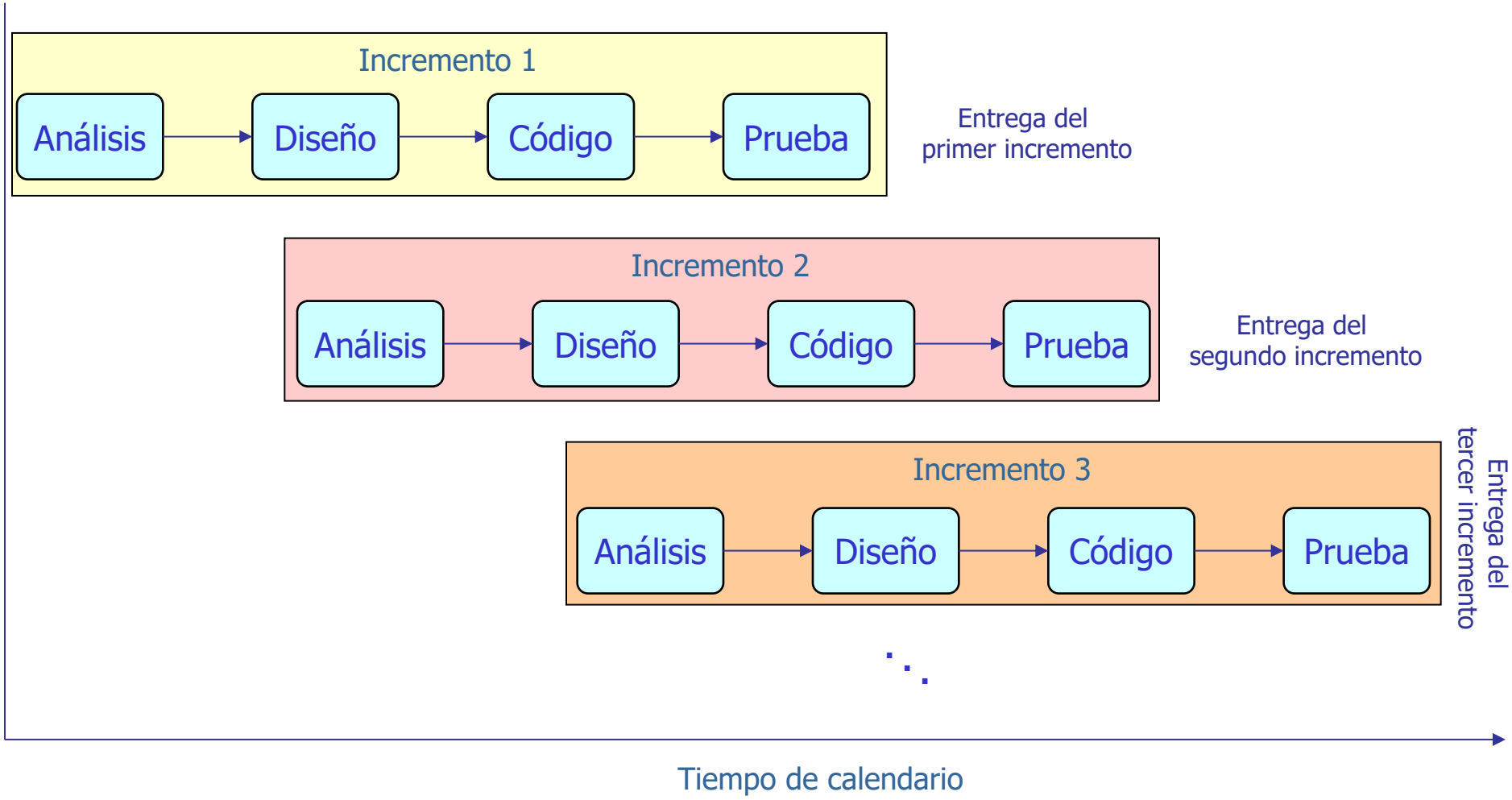
MODELOS EVOLUTIVOS

- El *software*, al igual que todos los sistemas complejos, evolucionan con el tiempo (Gilb, 1988)
- Se caracterizan porque permiten desarrollar versiones cada vez más completas del *software*, teniendo en cuenta la naturaleza evolutiva del *software*
- Presentan la filosofía de poner un producto en explotación cuanto antes
  - Están muy ligados a la idea de prototipado evolutivo
- Existen muchos modelos de proceso evolutivos
- Los modelos evolutivos son iterativos (Pressman, 2010)
  - Se caracterizan por la forma en que permiten a los ingenieros de *software* desarrollar versiones cada vez más completas del producto *software*, que puede ir entregándose al cliente en forma de incrementos

MODELOS EVOLUTIVOS

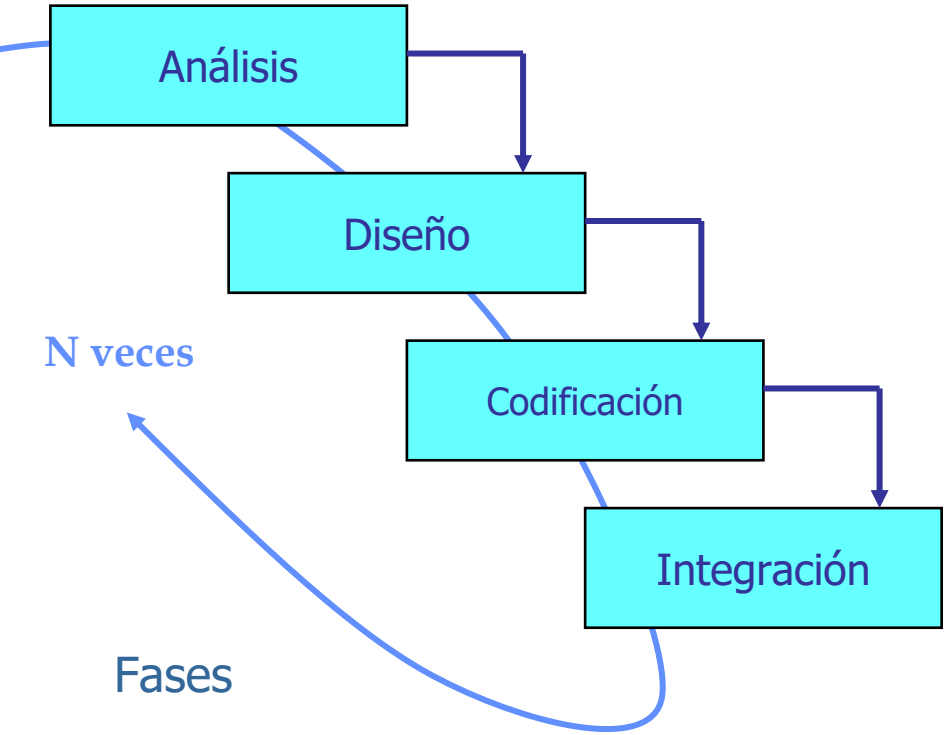
- Los desarrollos orientados a objetos se ajustan a un modelo de proceso iterativo e incremental
- Se puede argumentar lo mismo para los desarrollos basados en componentes
- Esto es así porque
  - Las tareas de cada fase se llevan a cabo de una forma iterativa
  - A la vez que existe un ciclo de desarrollo **análisis-diseño-implementación-análisis** que permite hacer evolucionar al sistema
  - En el desarrollo incremental el sistema se divide en un conjunto de particiones
    - Cada una se desarrolla de forma completa hasta que se finaliza el sistema
- Esta idea de iteratividad máxima propia de la orientación a objetos ha sido equiparada por autores como James Rumbaugh (1992) o L. B. S. Raccoon (1995) a las fractales o la teoría del caos

DIMENSIÓN INCREMENTAL

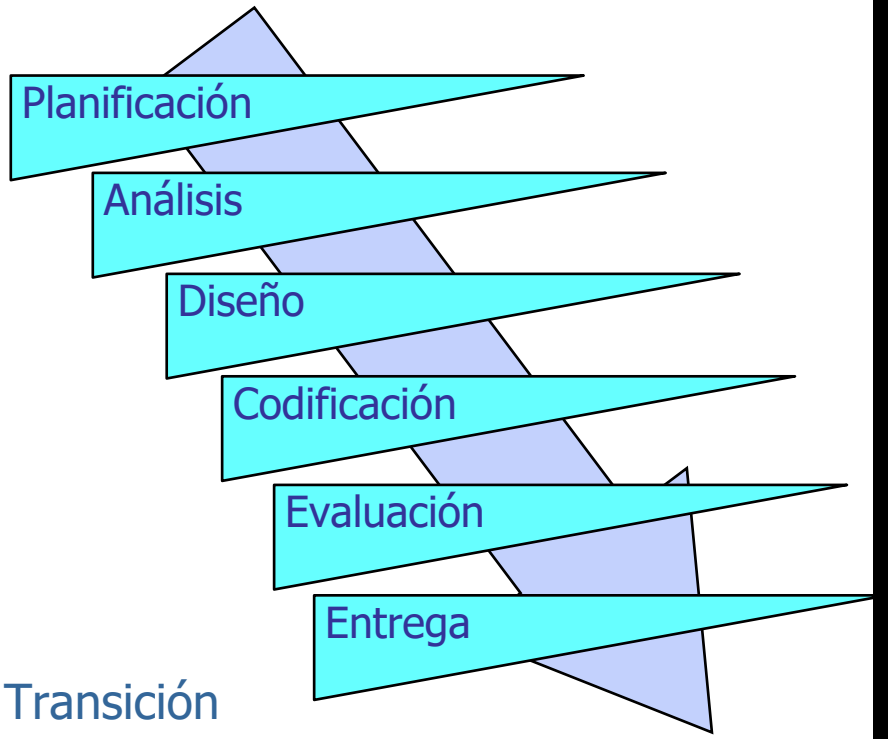


Modelo incremental (Pressman, 2010)

DIMENSIÓN ITERATIVA



Fases

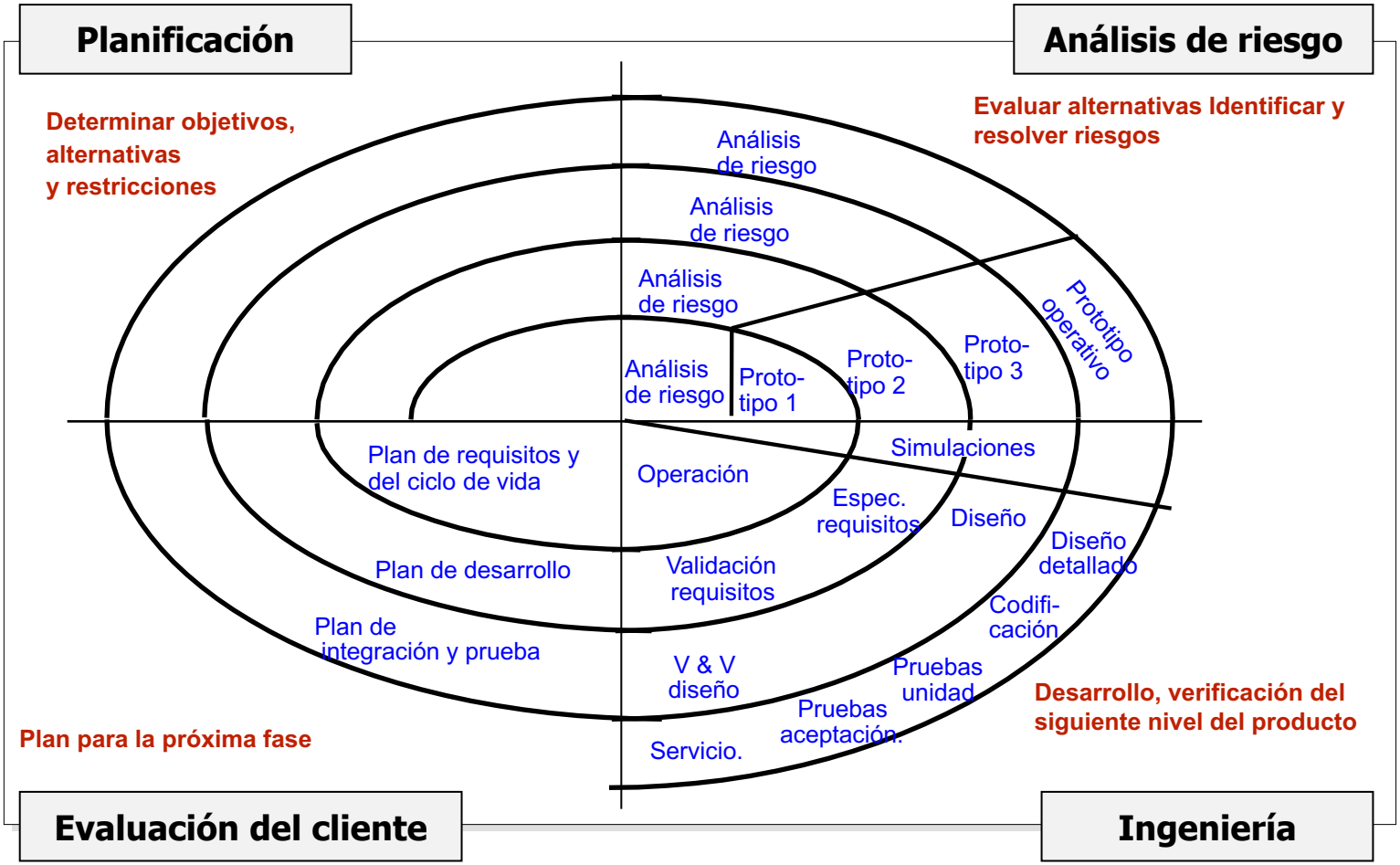


Transición progresiva

MODELO EN ESPIRAL

- Fue propuesto inicialmente por **B. Boehm** (1988)
- Es un modelo de proceso de *software* evolutivo, que proporciona el potencial para el desarrollo rápido de versiones incrementales del *software*
- Características
  - Puede considerarse como un metamodelo de proceso
    - Reúne características del modelo clásico y de prototipos
  - Aparece el análisis de riesgo
  - Se divide en un número de actividades estructurales, también denominadas regiones de tareas. En el modelo original de **Boehm** aparecen cuatro regiones de tareas
    - Planificación, Análisis de riesgos, Ingeniería, Evaluación del cliente
  - El avance se realiza desde el centro de la espiral hacia el exterior

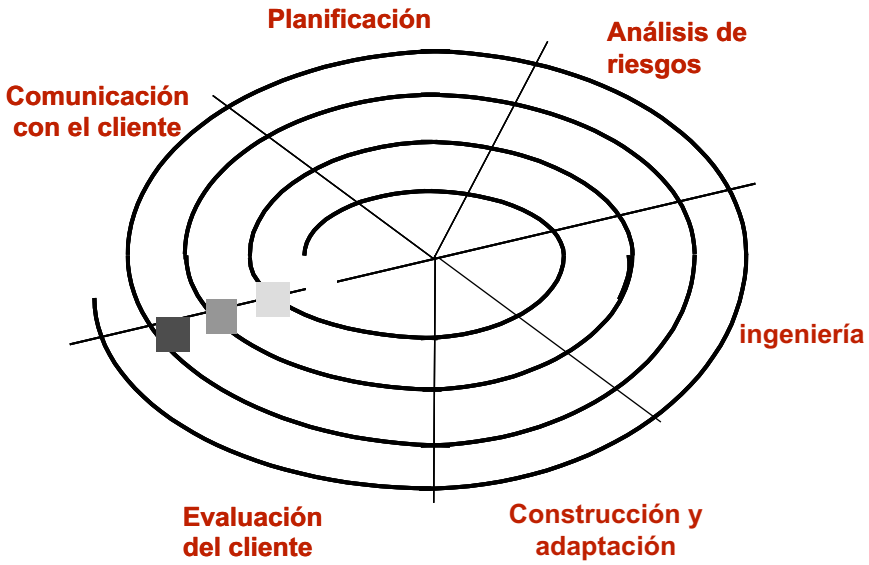
MODELO EN ESPIRAL



Ciclo de vida en espiral (Boehm, 1988)

MODELO EN ESPIRAL

- Variante del modelo de Boehm con 6 regiones de tareas (Pressman, 2000)
  - Se define un eje con diferentes puntos de entrada para diferentes tipos de proyectos



Puntos de entrada al proyecto

- Proyecto de mantenimiento de productos
- Proyecto de mejora de productos
- Proyecto de desarrollo de productos nuevos
- Proyecto de desarrollo de conceptos

Modelo en espiral de Pressman

MODELO EN ESPIRAL

Ventajas

- Refleja de forma más realista la idiosincrasia del desarrollo de *software*
- Toma lo mejor y evita lo peor de los demás modelos, según la situación en cada momento
- Las opciones de reutilización se tienen en cuenta desde el primer momento
- Proporciona una preparación para la evolución, crecimiento y cambio
- Proporciona un mecanismo para incorporar objetivos de calidad en el desarrollo
- Se centra en la eliminación de errores y opciones no atractivas desde el principio
- Determina el nivel de esfuerzo de cada fase en cada proyecto
- Se sigue el mismo procedimiento para el desarrollo que para el mantenimiento, con lo que se evitan los problemas de las “mejoras rutinarias” de alto riesgo
- Permite una gran flexibilidad
- Se adapta bien al diseño y programación orientado a objetos

BIBLIOGRAFÍA

- A. M. Davis, *Software requirements. Objects, functions and states*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall International, 1993.
- Asociación Española para la Calidad. *Glosario de Términos de Calidad e Ingeniería del Software*. AECC, 1986
- B. Curtis, H. Krasner, V. Shen y N. Iscoe, "On Building Software Process Models under the Lamppost," en *Proceedings of the 9th International Conference on Software Engineering* pp. 96-103, USA: IEEE CS Press, 1987.
- B. W. Boehm, "A spiral model of software development and enhancement," *Computer,* vol. 21, no. 5, pp. 61-72, 1988. doi: 10.1109/2.59.
- D. D. McCracken y M. A. Jackson, "A Minority Dissenting Opinion," en *Systems Analysis and Design: A Foundation for the 1980s*, W. W. Cotterman, J. D. Couger, N. L. Enger y F. Harold, Eds. pp. 551-553, New York, USA, 1981.
- F. J. García-Peñalvo, A. García-Holgado y A. Vázquez-Ingelmo, "Introducción a la Ingeniería del Software," 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/2WZIfWt. doi: 10.5281/zenodo.4399270. (pp. 42-64).
- F. J. García-Peñalvo, A. García-Holgado y A. Vázquez-Ingelmo, "Modelos de proceso," 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/3pSgw5o. doi: 10.5281/zenodo.4420089.
- I. Sommerville, *Ingeniería del Software*, 7ª ed. Madrid, España: Pearson Educación, 2005.
- ISO/IEC/IEEE, *ISO/IEC/IEEE International Standard - Systems and software engineering -- Software life cycle processes* (ISO/IEC/IEEE 12207:2017(E)). USA: IEEE, 2017. doi: 10.1109/IEEESTD.2017.8100771.
- J. Rumbaugh, "Over the waterfall and into the whirlpool," *JOOP,* pp. 23-26, 1992.
- L. B. S. Raccoon, "Fifty years of progress in software engineering," *ACM SIGSOFT Software Engineering Notes,* vol. 22, no. 1, pp. 88-104, 1997. doi: 10.1145/251759.251878.
- 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.

BIBLIOGRAFÍA

- R. S. Pressman, *Software Engineering: A Practitioner's Approach - European Adaptation*, 5th ed. London, England: McGraw-Hill, 2000.
- R. S. Pressman, *Ingeniería del Software: Un Enfoque Práctico*, 6ª ed. México D. F., México: McGraw-Hill, 2006.
- R. S. Pressman, *Ingeniería del Software: Un Enfoque Práctico*, 7ª ed. México D. F., México: McGraw-Hill, 2010.
- S. L. Pfleeger, *Ingeniería del Software. Teoría y Práctica*. Argentina: Prentice Hall, 2002.
- T. Gilb, *Principles of Software Engineering Management*. Wokingham, UK: Addison-Wesley Professional, 1988.
- V. Basili y A. Turner, "Iterative Enhancement: A practical Technique for Software Development," *IEEE Transactions on Software Engineering,* vol. 1, no. 4, pp. 390-396, 1975. doi: 10.1109/TSE.1975.6312870.
- W. Scacchi, "Modelling software evolution: A knowledge-based approach," *ACM SIGSOFT Software Engineering Notes,* vol. 14, no. 4, pp. 153–155, 1988. doi: 10.1145/75111.75139.
- W. B. Frakes, C. Fox y B. A. Nejmeh, *Software Engineering in the UNIX/C Environment*. Englewood Cliffs, NJ, USA: Prentice Hall, 1991.
- W. W. Royce, "Managing the development of large software systems: Concepts and techniques," presentado en Western Electronic Show and Convention (WesCon) August 25–28, 1970, Los Angeles, CA, USA, 1970.

PROCESO

INGENIERÍA DE SOFTWARE I

2º DE GRADO EN INGENIERÍA INFORMÁTICA
CURSO 2020/2021

Francisco José García Peñalvo / fgarcia@usal.es
Alicia García Holgado / aliciagh@usal.es
Andrea Vázquez Ingelmo / andreavazquez@usal.es

Departamento de Informática y Automática
Universidad de Salamanca