Introducción a la Ingeniería del Software

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Sevilla

Objetivos


Desarrollo
Tendencias
Chaos
Actuales
Configuración
Normas
Gestión
Industria
Mantenimiento
Software
Calidad
Estándares
Productos
Proyecto
Report
Requisitos
Ingeniería
Problemas
Características
Contenido

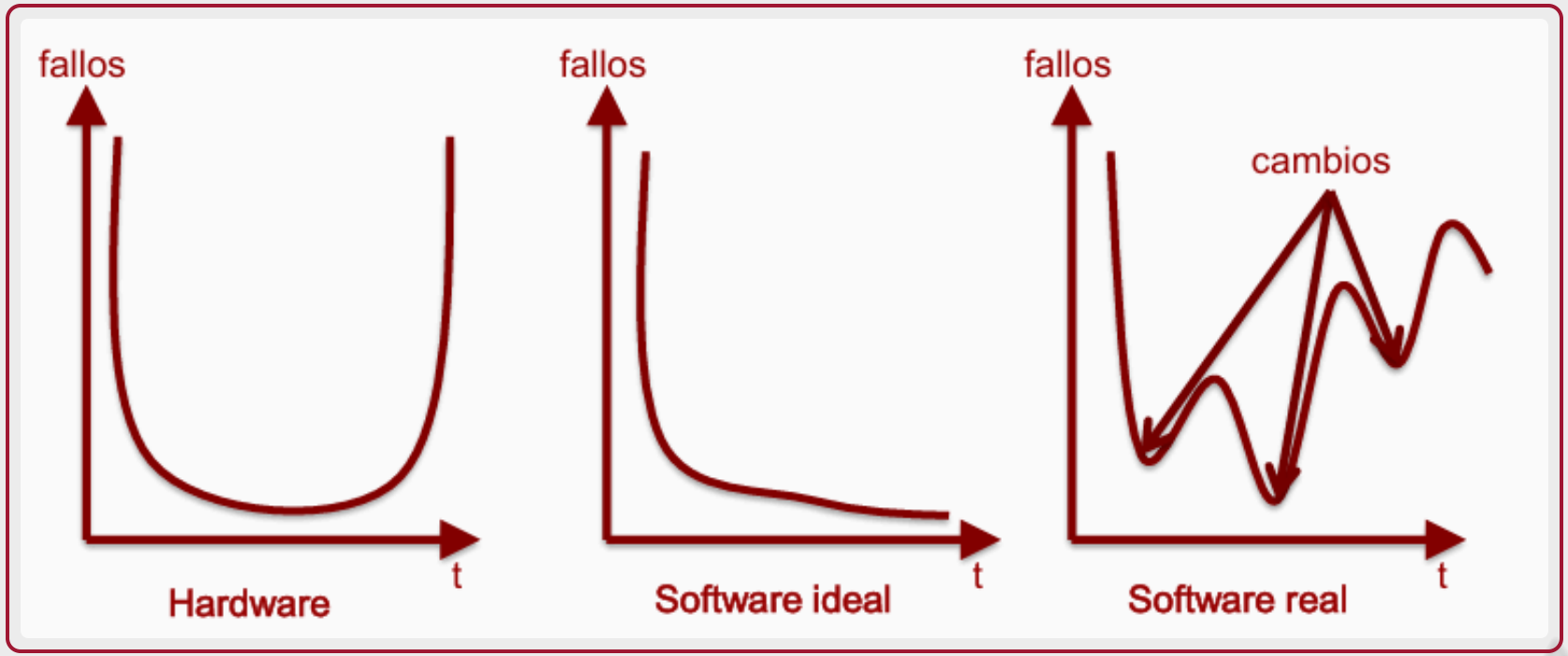
Contenido

- Características del software
- Problemas de la industria del software
- La necesidad de una ingeniería del software (IS)
- Normas y estándares
- Productos de la IS
- Mantenimiento del software
- Calidad del software

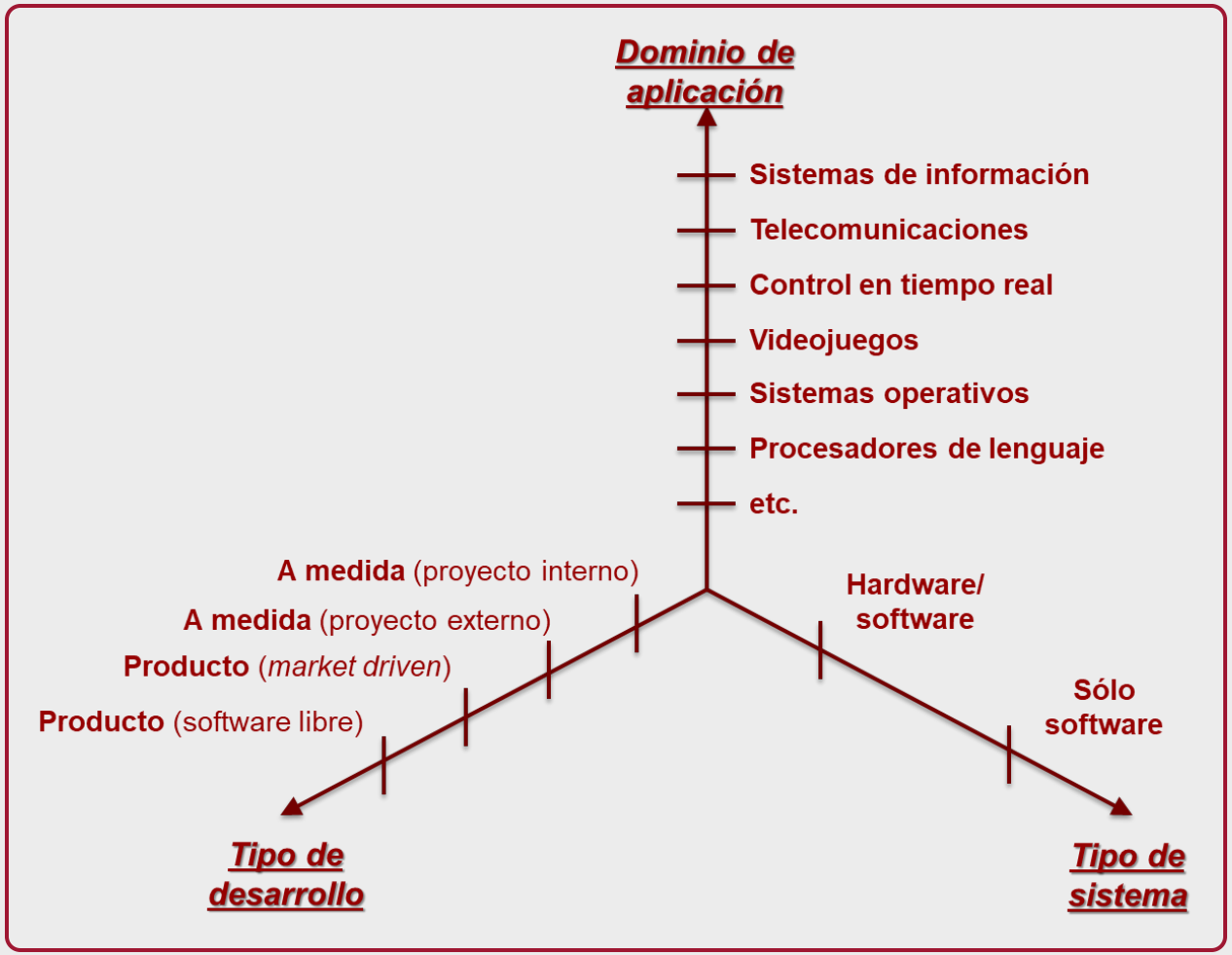
Contenido

El software…

- Es intangible.
- Se desarrolla, no se fabrica.
- No se estropea (se puede quedar obsoleto).



Tipos de software



Evolución del coste del software

Hardware

- Válvulas de vacío
- Transistores
- Circuitos integrados
- Microprocesador
- Ordenador personal

Software

- Interfaces gráficas de usuario
- Internet
- La nube

Evolución del Hardware / Software

1ª Generación (1945-56)

- **HW**: Electrónica con tubos de vacío. ENIAC-1946, primer ordenador. IBM 701, primer ordenador comercializado
- **SW**: Lenguaje máquina (ensamblador). Trabajo secuencial: perforación, ejecución, impresión. 10K instrucciones por segundo

2ª Generación (1957-63)

- **HW**: Electrónica con transistores, memoria con núcleos de ferrita
- **SW**: Lenguajes alto nivel: COBOL, FORTRAN, ALGOL.Procesamiento por lotes, maquinas dedicadas a E/S

3ª Generación (1964-71)

- **HW**: Electrónica con circuitos integrados, memoria en chips
- **HW**: Utilización de SSOO. 5M instrucciones por segundo

Evolución del Hardware / Software

4ª Generación (1972-81)

- **HW**:Microprocesadores y chips de memoria. Mayor capacidad de integración (VLSI)
- **SW**: Lenguaje C / Unix. Programas de amplio uso. 200M instrucciones por segundo

5ª Generación (1982-89)

- **HW**: Microprocesadores en paralelo. Redes de computadores. Componentes ópticos (CD/DVD)
- **SW**: Internet. Lenguajes simbólicos. Programas más complejos. 1G instrucción por segundo

6ª Generación (1990-Actualidad)

- **HW**: Arquitecturas paralelas y distribuidas. Procesadores especializados GPUs. Computación en la nube. Arquitecturas cuánticas
- **SW**: Programación funcional. Big Data. Desarrollo de la inteligencia artificial

Contenido

- Características del software
- **Problemas de la industria del software**
- La necesidad de una ingeniería del software (IS)
- Normas y estándares
- Productos de la IS
- Mantenimiento del software
- Calidad del software

Siempre es culpa del software…

Therac-25 (1985-1987)

Máquina de radioterapia que mató a seis pacientes por radiación excesiva.

Causa: error de programación en el sistema operativo.

Enlace

Siempre es culpa del software…

Ariane 5 (1996)

Cohete espacial de 370M$ que explotó 40seg después de su lanzamiento.

Causa: overflow de una variable por reutilización de código de un modelo anterior.

Enlace

Siempre es culpa del software…

Efecto 2000 (1999)

Miles de programas han de ser revisados para evitar que consideren el año 2000 como 1900.

Causa: almacenamiento del año en dos dígitos.

Enlace

Siempre es culpa del software…

Guerra del Golfo, 1991

Falló en interceptar un misil Scud, que impactó en un cuartel estadounidense.

Causa: Error de precisión acumulativa en el software del sistema antimisiles Patriot. 28 soldados murieron.

Enlace

Siempre es culpa del software…

Mars Climate Orbiter (1999)

La nave entró en la atmósfera de Marte con la trayectoria equivocada en 170km y se destruyó.

Causa: Un error de conversión de unidades (sistema inglés vs métrico). Pérdida de 327M$.

Enlace

Siempre es culpa del software…

Airbus Military (2015)

Un A400-M se estrella durante un vuelo de pruebas a 2 kilómetros del aeropuerto de Sevilla.

Causa: errores humanos y del software encargado de regular la potencia de los motores en función de las señales que le enviaba el piloto.

Enlace

Chaos Reports

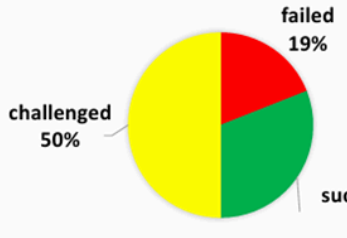
Intentan identificar los principales problemas del desarrollo de software.

- Realizado por la consultora Standish Group.
- Clasifica miles de proyectos reales como:
  - **Éxito**: finalizado dentro del plazo y presupuesto y cumpliendo todos los requisitos.
  - **Con problemas**: finalizado pero fuera de plazo, fuera de presupuesto y sin cumplir todos los requisitos.
  - **Fracaso**: cancelado durante el desarrollo.
- Periodos 2016-2020, 2011-2015, 1996-2012.

CHAOS report (2016 – 2020)

Project Success
Quick Reference Card

Based on CHAOS 2020: Beyond Infinity Overview. January'2021, QRC by Henny Portman



Modern measurement
(software projects)

Good Sponsor, Good Team, and Good Place are the only things we need to improve and build on to improve project performance.

The Good Place is where the sponsor and team work to create the product. It's made up of the people who support both sponsor and team. These people can be helpful or destructive. It's imperative that the organization work to improve their skills if a project is to succeed. This area is the hardest to mitigate, since each project is touched by so many people. Principles for a Good Place are:

- The Decision Latency Principle
- The Emotional Maturity Principle
- The Communication Principle
- The User Involvement Principle
- The Five Deadly Sins Principle
- The Negotiation Principle
- The Competency Principle
- The Optimization Principle
- The Rapid Execution Principle
- The Enterprise Architecture Principle

Successful project Resolution by Good Place Maturity Level:

highly mature	50%
mature	34%
moderately mature	23%
not mature	23%

The Good Team is the project's workhorse. They do the heavy lifting. The sponsor breathes life into the project, but the team takes that breath and uses it to create a viable product that the organization can use and from which it derives value. Since we recommend small teams, this is the second easiest area to improve. Principles for a Good Team are:

- The Influential Principle
- The Mindfulness Principle
- The Five Deadly Sins Principle
- The Problem-Solver Principle
- The Communication Principle
- The Acceptance Principle
- The Respectfulness Principle
- The Confrontationist Principle
- The Civility Principle
- The Driven Principle

Successful project Resolution by Good Team Maturity Level:

highly mature	66%
mature	46%
moderately mature	21%
not mature	1%

The Good Sponsor is the soul of the project. The sponsor breathes life into a project, and without the sponsor there is no project. Improving the skills of the project sponsor is the number-one factor of success – and also the easiest to improve upon, since each project has only one. Principles for a Good Sponsor are:

- The Decision Latency principle
- The Vision Principle
- The Work Smart Principle
- The Daydream Principle
- The Influence Principle
- The Passionate Principle
- The People Principle
- The Tension Principle
- The Torque Principle
- The Progress Principle

Successful project Resolution by Good Sponsor Maturity Level:

highly mature	67%
mature	33%
moderately mature	21%
not mature	18%

CHAOS report (2016 – 2020)

Identifica 3 factores críticos para el éxito:

- Good **Sponsor**: El patrocinador que impulsa el proyecto: tiene visión, influencia, pasión, capacidad de tensión, supervisión, etc.
- Good **Team**: que trabaja bien junto con el sponsor. Se destacan principios como comunicación, resolución de problemas, aceptación, respeto, etc. Se recomienda que los equipos sean pequeños.
- Good **Place**: entorno de proyecto favorable, incluye herramientas, cultura, personas que apoyan el equipo y sponsor. Se considera el más difícil de moldear ya que involucra muchos actores externos.

Problema raíz: latencia en la toma de decisiones

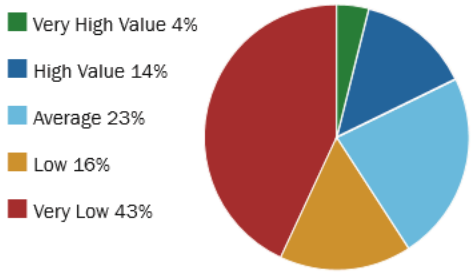
CHAOS Report (2011 - 2015)

MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

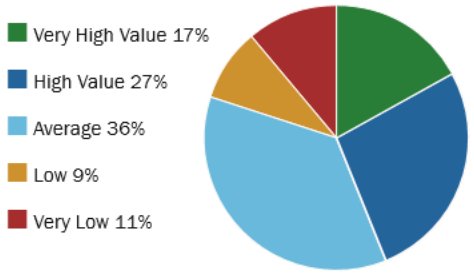
The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

VALUE FOR LARGE PROJECTS



The return of value for large projects from FY2011–to 2015 within the new CHAOS database.

VALUE FOR SMALL PROJECTS



The return of value for small projects from FY2011–2015 within the new CHAOS database.

CHAOS Report (2011 - 2015)

PROJECT SIZE BY CHAOS RESOLUTION

	SUCCESSFUL	CHALLENGED	FAILED	TOTAL
Grand	6%	51%	43%	100%
Large	11%	59%	30%	100%
Medium	12%	62%	26%	100%
Moderate	24%	64%	12%	100%
Small	61%	32%	7%	100%

The size of software projects by the Modern Resolution definition from FY2011–2015 within the new CHAOS database.

CHAOS RESOLUTION BY PROJECT SIZE

	SUCCESSFUL	CHALLENGED	FAILED
Grand	2%	7%	17%
Large	6%	17%	24%
Medium	9%	26%	31%
Moderate	21%	32%	17%
Small	62%	16%	11%
TOTAL	100%	100%	100%

The resolution of all software projects by size from FY2011–2015 within the new CHAOS database.

CHAOS RESOLUTION BY INDUSTRY

	SUCCESSFUL	CHALLENGED	FAILED
Banking	30%	55%	15%
Financial	29%	56%	15%
Government	21%	55%	24%
Healthcare	29%	53%	18%
Manufacturing	28%	53%	19%
Retail	35%	49%	16%
Services	29%	52%	19%
Telecom	24%	53%	23%
Other	29%	48%	23%

The resolution of all software projects by industry from FY2011–2015 within the new CHAOS database.

CHAOS Report (2011 - 2015)

CHAOS RESOLUTION BY AREA OF THE WORLD

	SUCCESSFUL	CHALLENGED	FAILED
North America	31%	51%	18%
Europe	25%	56%	19%
Asia	22%	58%	20%
Rest of World	24%	55%	21%

The resolution of all software projects from FY2011–2015 by the four major areas of the world.

CHAOS RESOLUTION BY AGILE VERSUS WATERFALL

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

The resolution of all software projects from FY2011–2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000.

CHAOS Report (2011 - 2015)

CHAOS RESOLUTION BY PROJECT TYPE

PROJECT TYPE	SUCCESSFUL	CHALLENGED	FAILED
Developed from scratch using traditional languages and methods	22%	61%	17%
Developed from scratch using modern methodologies	23%	54%	23%
Developed some components and purchased others	24%	59%	17%
Purchased components and assembled the application	25%	59%	16%
Purchased application and modified	42%	37%	21%
Purchased application and performed no modifications	57%	28%	15%
Modernization	53%	38%	9%
Other	28%	47%	25%

The resolution of all software projects by project type from FY2011–2015 within the new CHAOS database.

CHAOS FACTORS OF SUCCESS

FACTORS OF SUCCESS	POINTS	INVESTMENT
Executive Sponsorship	15	15%
Emotional Maturity	15	15%
User Involvement	15	15%
Optimization	15	15%
Skilled Resources	10	10%
Standard Architecture	8	8%
Agile Process	7	7%
Modest Execution	6	6%
Project Management Expertise	5	5%
Clear Business Objectives	4	4%

The 2015 Factors of Success. This chart reflects our opinion of the importance of each attribute and our recommendation of the amount of effort and investment that should be considered to improve project success.

CHAOS Report (1996 - 2012)

Factores de éxito

- Implicación de los usuarios
- Apoyo de los directivos
- Enunciado claro de los requisitos
- Planificación adecuada
- Expectativas realistas
- Hitos de proyecto pequeños
- Personal competente
- Sentimiento de propiedad
- Visión y objetivos claros
- Trabajo duro y personal concentrado

Causas de problemas

- Falta de información por parte de los usuarios
- Especificaciones y requisitos incompletos
- Especificaciones y requisitos cambiantes
- Falta de apoyo de los directivos
- Incompetencia tecnológica
- Falta de recursos
- Expectativas no realistas
- Objetivos poco claros
- Plazos temporales no realistas
- Nueva tecnología

Causas de fracasos

- Requisitos incompletos
- Falta de implicación de los usuarios
- Falta de recursos
- Expectativas no realistas
- Falta de apoyo de los directivos
- Especificaciones y requisitos cambiantes
- Falta de planificación
- Ya no lo necesito
- Falta de gestión de TIC
- Desconocimiento de la tecnología

¿Qué es una Ingeniería?

Actividades propias de los Ingenieros:

> Conciben, proyectan, construyen y gestionan la explotación eficiente de: obras públicas, máquinas, sistemas de control, redes de comunicaciones, centrales energéticas, sistemas de regadíos, …. Y el software

Científicos:

> Realizan actividad sistemática para adquirir nuevos conocimientos.

Pilares de una Ingeniería

- **Vocabulario**: conjunto de términos que se usan en un campo concreto
  > Interfaz, clase, objeto, variable, interface, requisito, …
- **Tecnología**: instrumentos, procedimientos o recursos usados en un campo
  > Java, Python, JS, HTML, Oracle, TCP/IP, …
- **Herramientas**: conjunto de instrumentos para desempeñar una trabajo concreto
  > Eclipse, Aptana, VSCode, SQLDeveloper, …
- **Buenas prácticas**: conjunto de acciones que dan buenos resultados en un campo
  > PMBOK, ITIL, CMMI, …
- **Metodologías**: conjunto de procedimientos bien definido para obtener buenos resultados.
  > SCRUM, RUP, XP, Kanban, FAFe, LeSS, DevOps, …

Orígenes de la ingeniería del software

Software Engineering Conference (SEC) OTAN, Garmisch, Alemania (1968).

- Enfoque ingenieril como la solución a lo que se denominó la **crisis del software**.
- El término se atribuye a **Fritz Bauer**.
- Se definió el concepto de *ciclo de vida* del software y se identificaron los principales **problemas** asociados al software:
  - Sobrecostes, retrasos, baja calidad, mantenimiento difícil, etc.

Definiciones de Ingeniería del Software

Según el glosario de IEEE (610.12)

> (a) la aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software; es decir, la aplicación de la ingeniería al software. (b) el estudio de los enfoques como los descritos en (a).*

Según A. Davis (201 Principles of Software Development)

> La aplicación inteligente de principios probados, técnicas, lenguajes y herramientas para la creación y mantenimiento, dentro de un coste razonable, de software que satisfaga las necesidades de los usuarios.*

Definición de Proyecto

PMI

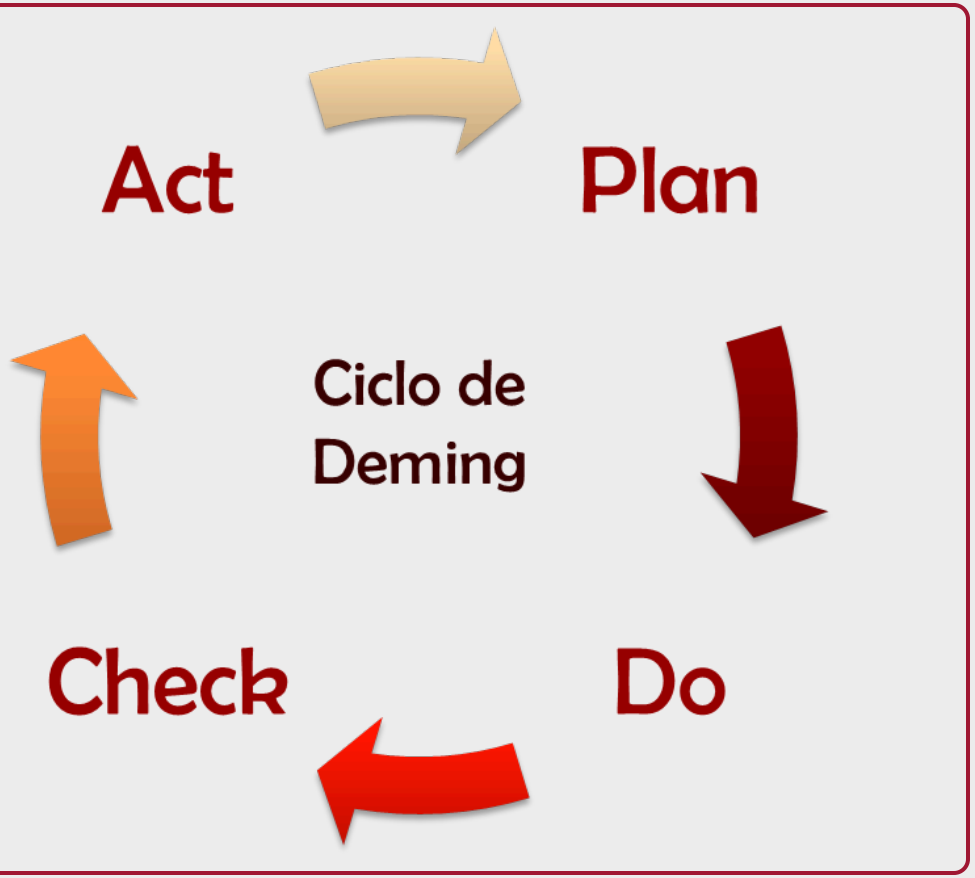
> Un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único

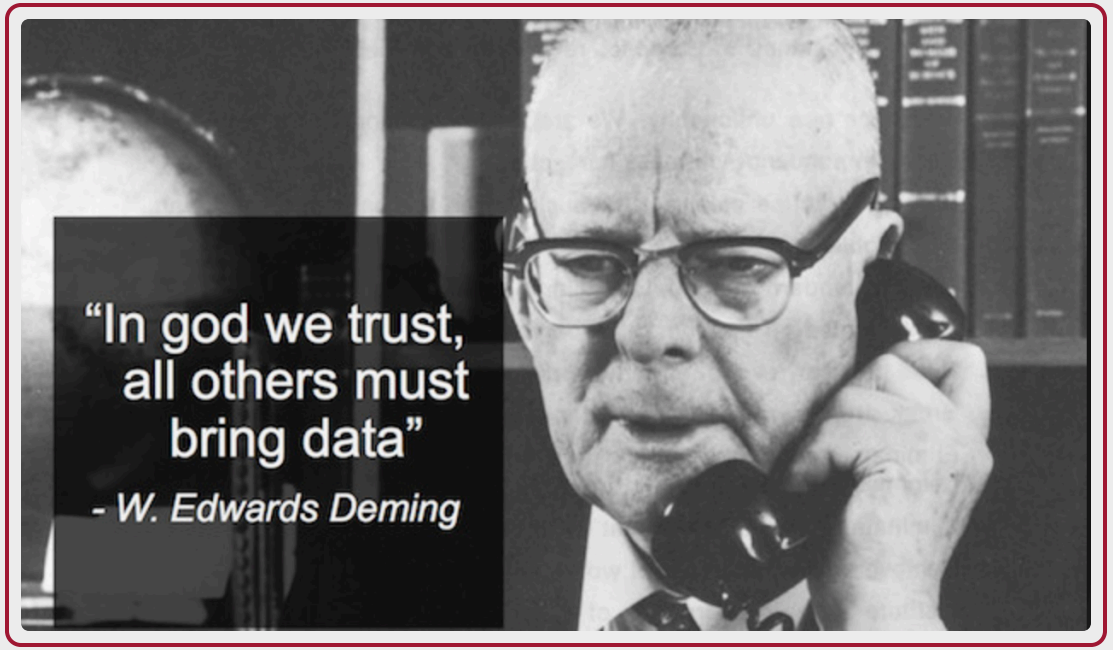
Tipos de Proyecto

> Productivo, Público, Social, De vida, Científico, ...

Este proyecto es importantísimo, pero no tiene presupuesto, ni documentación y además es para mañana. Por fin, esta es tu oportunidad para impresionar de verdad a todos.

Etapas de un proyecto





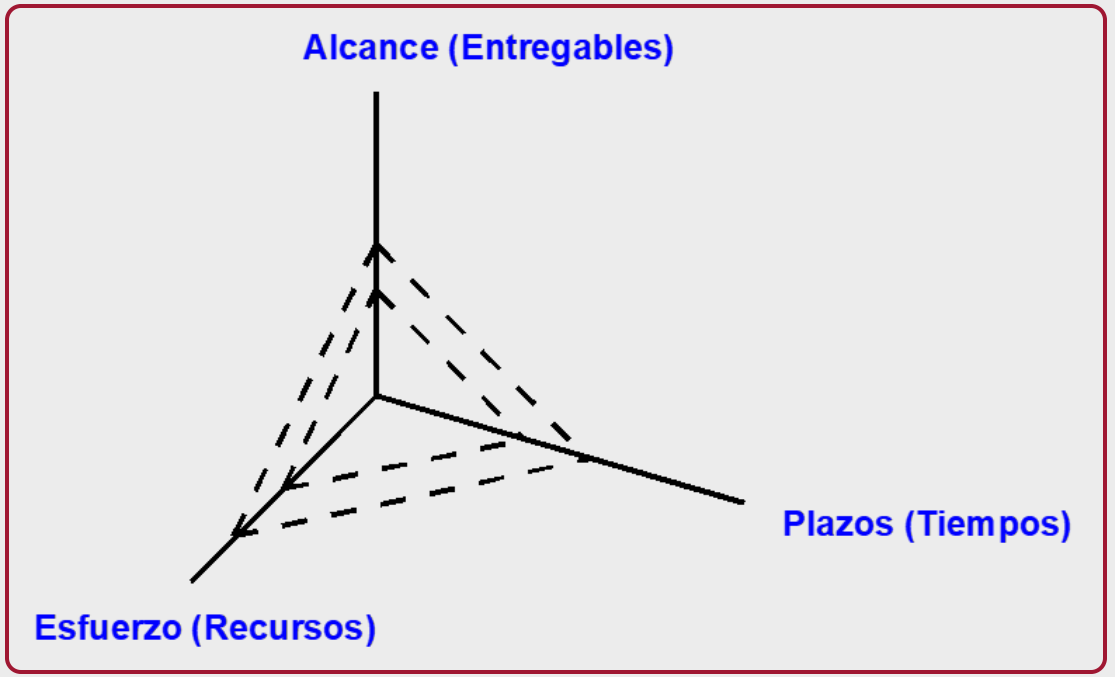
Proyecto software

Esfuerzo **temporal** acometido para crear un único producto o servicio software.

Es realizado por **personas**.

Debe ser limitado en **tiempo y coste**.

Debe ser **planificado**, **ejecutado** y **controlado**.



Roles en un proyecto software

Director de proyecto

Responsable de la ejecución del proyecto con capacidad ejecutiva para tomar **decisiones** sobre el mismo de acuerdo con el cliente.

Ingeniero de requisitos

También denominado **analista**. Responsable de interactuar con clientes y usuarios para obtener sus necesidades y de desarrollar y gestionar los requisitos.

Equipo de desarrollo

Conjunto de personas implicadas en el **desarrollo** del software: arquitecto software, diseñador de IU, programador, responsable de pruebas, administrador de BD, etc.

Roles en un proyecto software

Equipo de calidad

> Personas responsables de la **calidad** de los productos (documentación como software). Se ocupan también de la calidad de los procesos.

Cliente

> Responsable de la **financiación** del proyecto con capacidad ejecutiva para tomar decisiones. Suele tener una visión global del modelo de negocio.

Usuario

> Usuario potencial del software a desarrollar en el proyecto con una **visión detallada**, aunque puede que parcial, del modelo de negocio.

Responsable TIC del Cliente

> Responsable del entorno tecnológico del cliente, sobre el que se debe **integrar** el sistema a desarrollar.

Contenido

- Características del software
- Problemas de la industria del software
- La necesidad de una ingeniería del software (IS)
- **Normas y estándares**
- Productos de la IS
- Mantenimiento del software
- Calidad del software

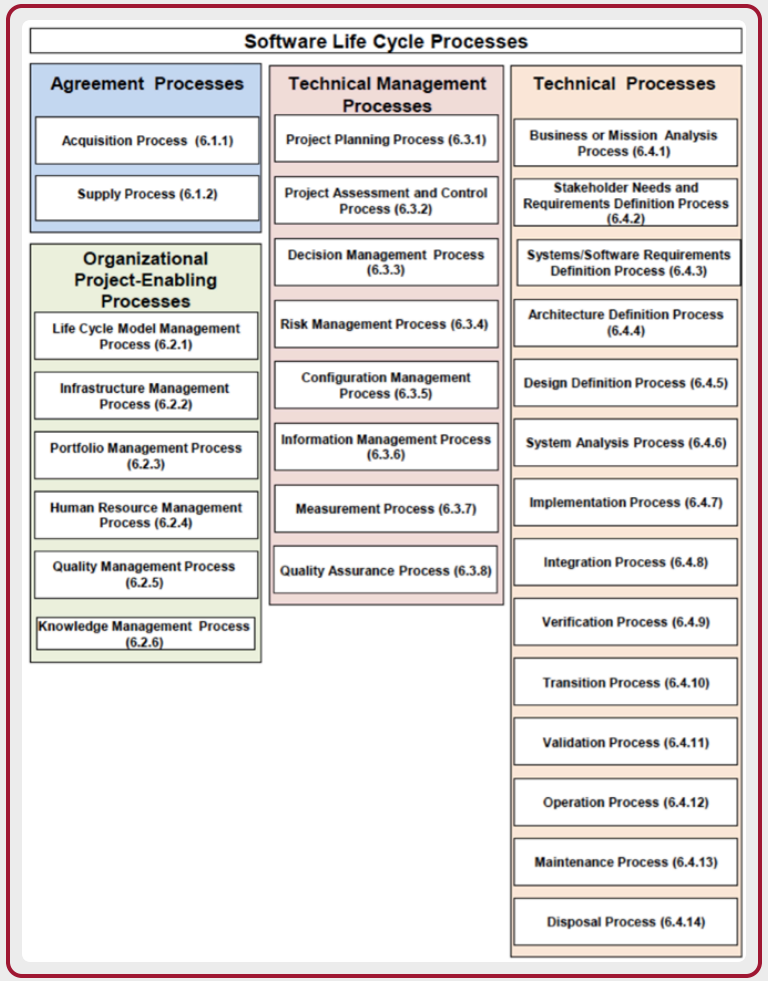
ISO/IEC/IEEE 12207:2017 (Enlace)

Estándar 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

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.

ISO/IEC/IEEE 12207:2017 (Enlace)

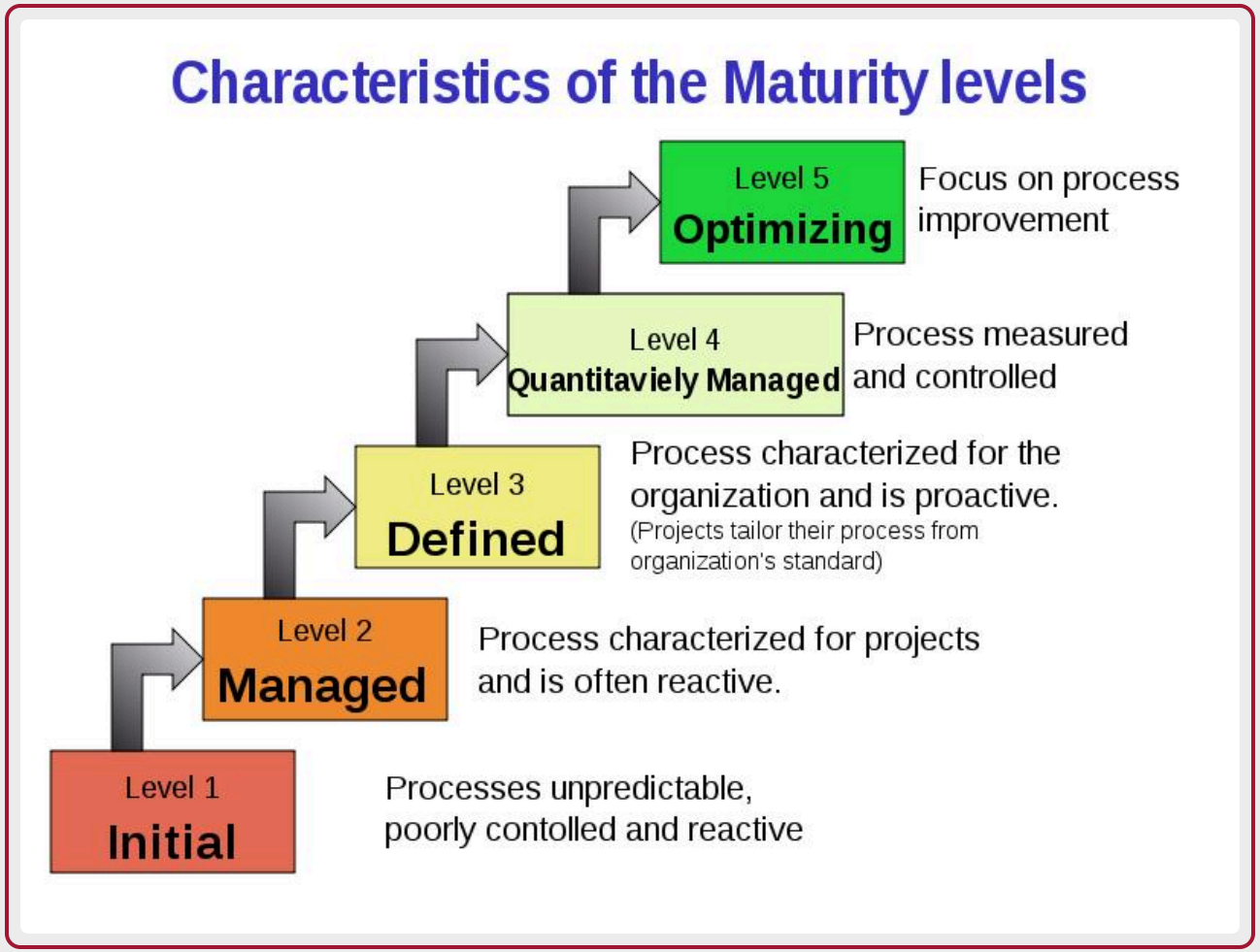


CMMI-DEV (2010)

Capability Maturity Model Integration for Development.

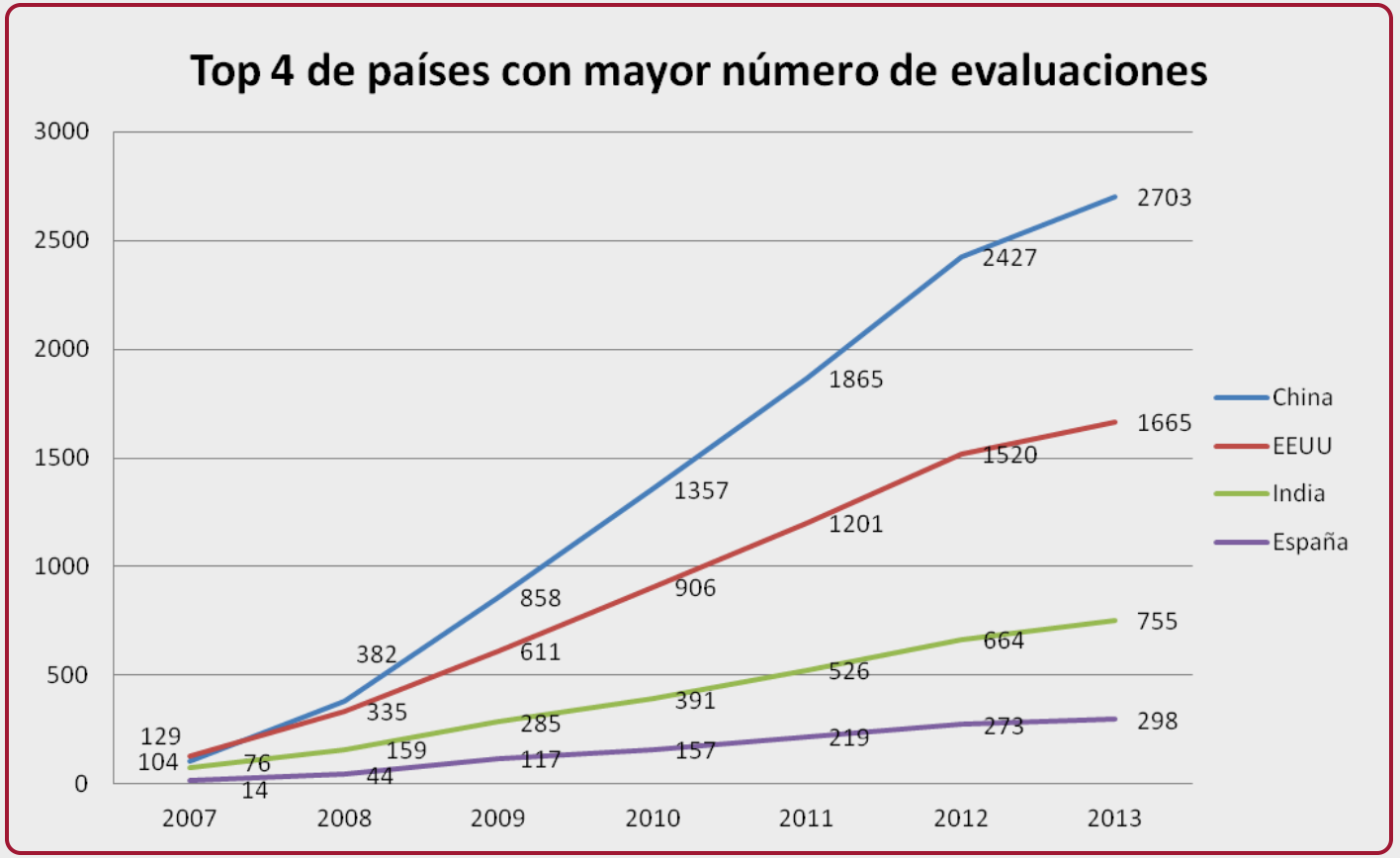
- Modelo para la mejora y **evaluación** de procesos para el desarrollo, mantenimiento y operación de sistemas software.
- Desarrollado por el Software Engineering Institute (SEI) de la Universidad Carnegie-Mellon para el Departamento de Defensa de EE.UU.
- Muchas administraciones públicas exigen un **nivel** mínimo de **certificación** en CMMI para contratar (entre 3 y 5 normalmente).

CMMI-DEV (2010)



CMMI-DEV (2010)

Gracias al plan Avanza, España es el cuarto país en certificaciones CMMI.



Contenido

- Características del software
- Problemas de la industria del software
- La necesidad de una ingeniería del software (IS)
- Normas y estándares
- **Productos de la IS**
- Mantenimiento del software
- Calidad del software

Software como producto de ingeniería

El conjunto de productos que deben desarrollarse y entregarse al cliente durante un proyecto se denominan entregables.

Productos previos al comienzo

1. Petición de Propuestas (Request for Proposals)
2. Pliego de Prescripciones Técnicas (AAPP)
3. Oferta
4. Contrato

Deben dejar claro…

- Las necesidades a satisfacer por el sistema.
- Los entregables del proyecto.
- El presupuesto y plazo de ejecución.
- Restricciones técnicas.
- Penalizaciones por retrasos.
- …

Entregables habituales

1. Plan de proyecto
2. Informes de seguimiento
3. Especificación de requisitos
4. Documento de diseño
5. Plan de pruebas
6. Código fuente
7. Software ejecutable
8. Manuales de usuario

Contenido

- Características del software
- Problemas de la industria del software
- La necesidad de una ingeniería del software (IS)
- Normas y estándares
- Productos de la IS
- **Mantenimiento del software**
- Calidad del software

Mantenimiento del software

Una vez entregado se debe proporcionar un servicio de **mantenimiento** y de **gestión de incidencias**.

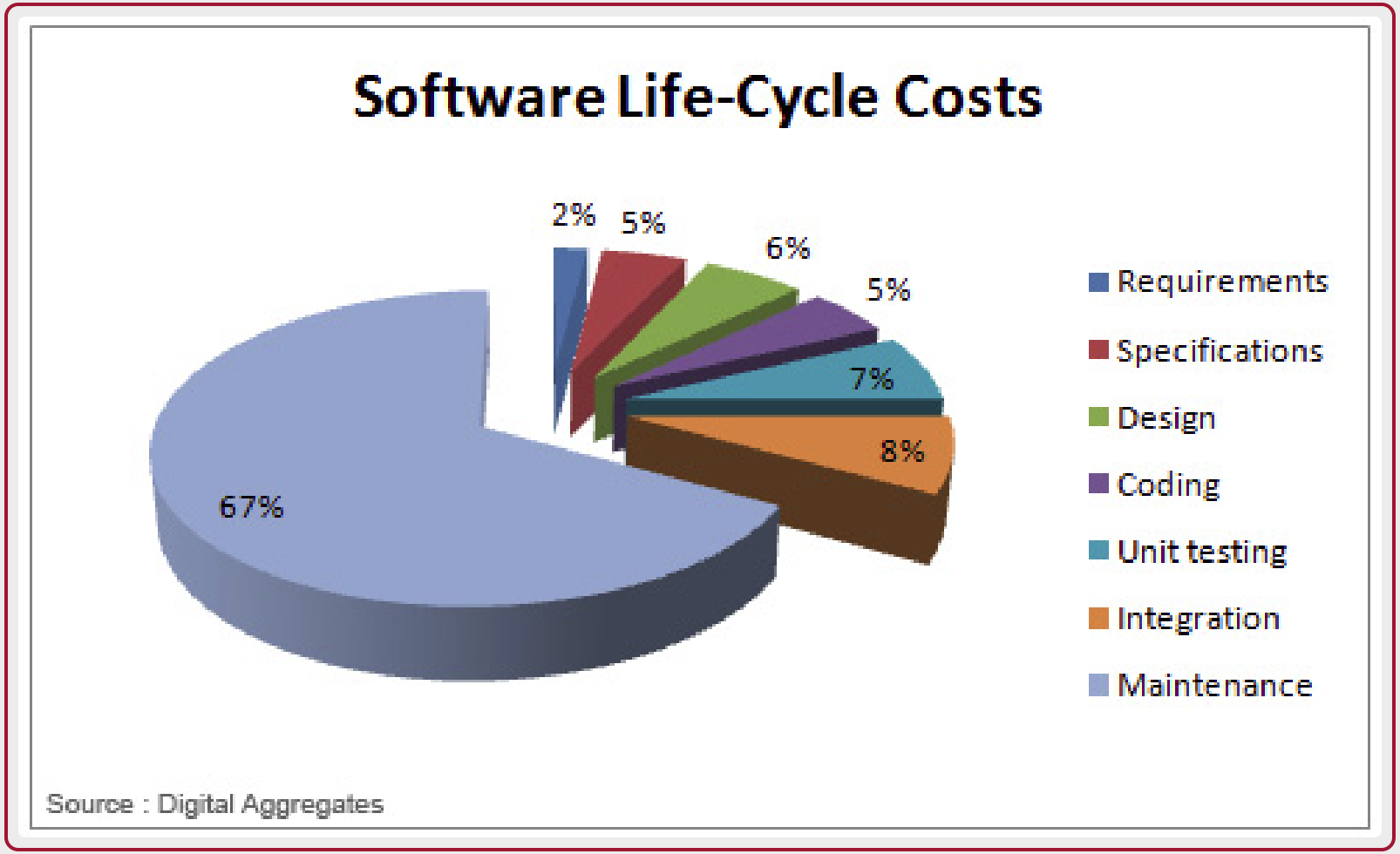
Mantenimiento

> Se encarga de mejorar, adaptar o corregir el software en explotación.
> Su coste es el más alto de todo el ciclo de vida.

Gestión de incidencias

> Restaurar cuanto antes la operativa normal del servicio minimizando el impacto negativo en las operaciones de negocio.
> Detección, registro, categorización, priorización, diagnóstico, escalado, resolución y cierre

Coste del mantenimiento



Tipos de mantenimiento (Métrica 3)

- **Evolutivo** (60%): incorporar nuevos requisitos o cambios en los ya existentes.
- **Correctivo** (17%): corregir errores del producto software no detectados durante el desarrollo.
- **Adaptativo** (18%): adaptar a cambios en el entorno tecnológico (hardware, sistema operativo, base de datos, comunicaciones, etc.).
- **Perfectivo** (5%): mejorar la calidad interna de los sistemas (refactorizar código, mejorar rendimiento, etc.)

Contenido

- Características del software
- Problemas de la industria del software
- La necesidad de una ingeniería del software (IS)
- Normas y estándares
- Productos de la IS
- Mantenimiento del software
- **Calidad del software**

Aseguramiento de la calidad del software

Se encarga de asegurar un determinado nivel de calidad del software.

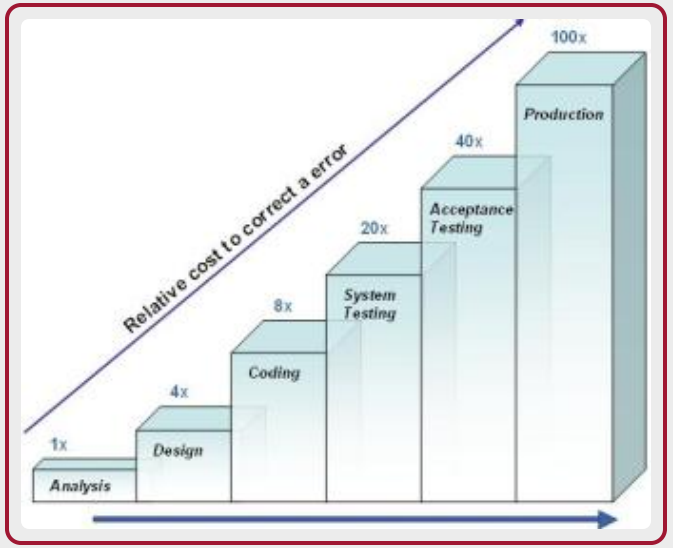
Por calidad del software se entiende:

- Cumplir los requisitos establecidos explícitamente.
- Cumplir con los estándares de desarrollo necesarios.
- Tener las características implícitas que se espera de todo software desarrollado profesionalmente.

Aseguramiento de la calidad del software

Los costes de aseguramiento de la calidad se compensan con el ahorro en mantenimiento.

> "El error es usualmente 100 veces más caro de corregir en la fase de mantenimiento que en la fase de requisitos."
> (Barry Boehm, Software Engineering Economics, 1981, p. 40.)



El equipo de SQA es responsable de:

Establecer el plan de SQA del proyecto.

Participar en la definición del plan del proyecto.

Auditar los productos del desarrollo.

Documentar e informar de las desviaciones o no conformidades que se vayan detectando en las revisiones técnicas formales (RTF).

Contenido

- Características del software
- Problemas de la industria del software
- La necesidad de una ingeniería del software (IS)
- Normas y estándares
- Productos de la IS
- Mantenimiento del software
- Calidad del software

Gestión de la configuración

Git/GitHub, SVN, Mercurial, GitLab, Bitbucket, …

Se encarga de identificar, controlar e informar de los cambios en los productos del desarrollo de software.

Dentro de sus actividades se identifican:

- Determinar los productos bajo control de configuración.
- Control de versiones.
- Control de cambios.
- Auditoría de la configuración.
- Generación de informes del estado de la configuración.

Gestión de la configuración

Línea base (baseline)

> Versión cerrada de algún elemento (o conjunto de elementos) de configuración a partir de la cual es necesario aplicar la política de control de cambios del proyecto antes de modificarlo.

CMS/CMDB (Configuration Management System / Configuration Management Database)

> Modelo lógico coherente de la infraestructura de la organización de TI, típicamente compuesto de varias Bases de Datos que operan como sub-sistemas físicos. Se usa para almacenar información acerca de los elementos software bajo control de versiones

Contenido

- Características del software
- Problemas de la industria del software
- La necesidad de una ingeniería del software (IS)
- Normas y estándares
- Productos de la IS
- Mantenimiento del software
- Calidad del software

Tendencias actuales

IA generativa: modelos capaces de acelerar la creatividad y reducir tiempos de entrega

- Texto (ChatGPT, Claude, Gemini)
- Código (Copilot, Amazon CodeWhisperer)
- Imágenes (DALL-E, MidJourney)
- Música (Suno, AIVA, Mubert)

Programación asistida por IA (Copilot)

- Mejora productividad y calidad, facilita el aprendizaje de nuevas tecnologías.

Desarrollo **low-code/no-code** (PowerApps, Mendix, …)

- Rapidez en la entrega, democratización del desarrollo, reducción de costos iniciales.
- Software sostenible (Green IT, consumo energético, algoritmos eficientes, CDCs verdes, …)
- Menor huella de carbono, ahorro energético, alineación con objetivos de sostenibilidad.

Conclusiones

La ingeniería del software es la aplicación de principios sistemáticos, disciplinados y cuantificables al desarrollo, operación y mantenimiento del software. Su surgimiento fue necesario ante la denominada "crisis del software": proyectos con sobrecostes, retrasos y baja calidad.

Un proyecto software es un esfuerzo temporal dirigido a crear un único producto o servicio, realizado por personas, limitado en tiempo y coste, que debe ser planificado, ejecutado y controlado. Requiere roles bien definidos: director, ingeniero de requisitos, equipo de desarrollo, equipo de calidad, cliente y usuario.

Las normas internacionales como ISO/IEC/IEEE 12207 y modelos como CMMI-DEV proporcionan marcos de referencia para mejorar y evaluar los procesos del ciclo de vida. Las administraciones públicas frecuentemente exigen certificaciones CMMI mínimas para contratar.

Los entregables de un proyecto incluyen planes, especificaciones de requisitos, documentos de diseño, código fuente, software ejecutable y manuales. El mantenimiento es la fase más costosa del ciclo de vida, representando más del 60% de los costes totales.

La calidad del software requiere cumplir requisitos explícitos, seguir estándares de desarrollo y poseer las características esperadas de un software profesional. La inversión en calidad durante desarrollo reduce significativamente los costes de mantenimiento posterior.

Gracias

Universidad de Sevilla

Departamento de Lenguajes y Sistemas Informáticos