

Unidad 1. Desarrollo de software

INGENIERÍA DEL SOFTWARE

ceedcv
CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

Entornos de desarrollo (ED)

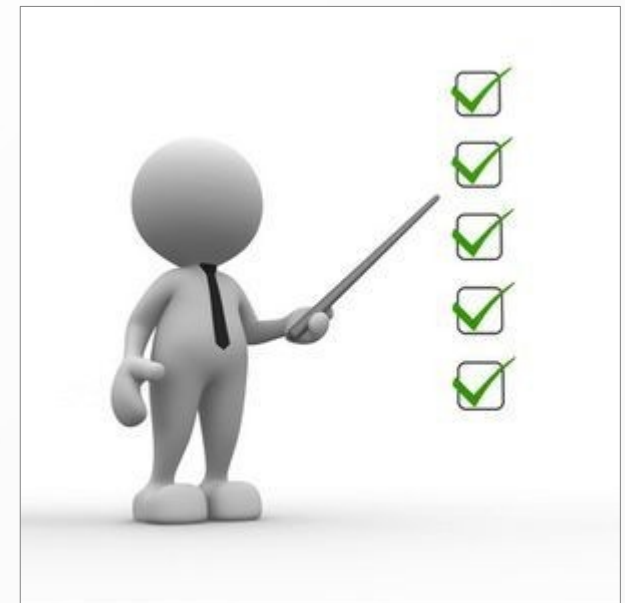
Sergio Badal
Raúl Palao

UD 01.Desarrollo de software

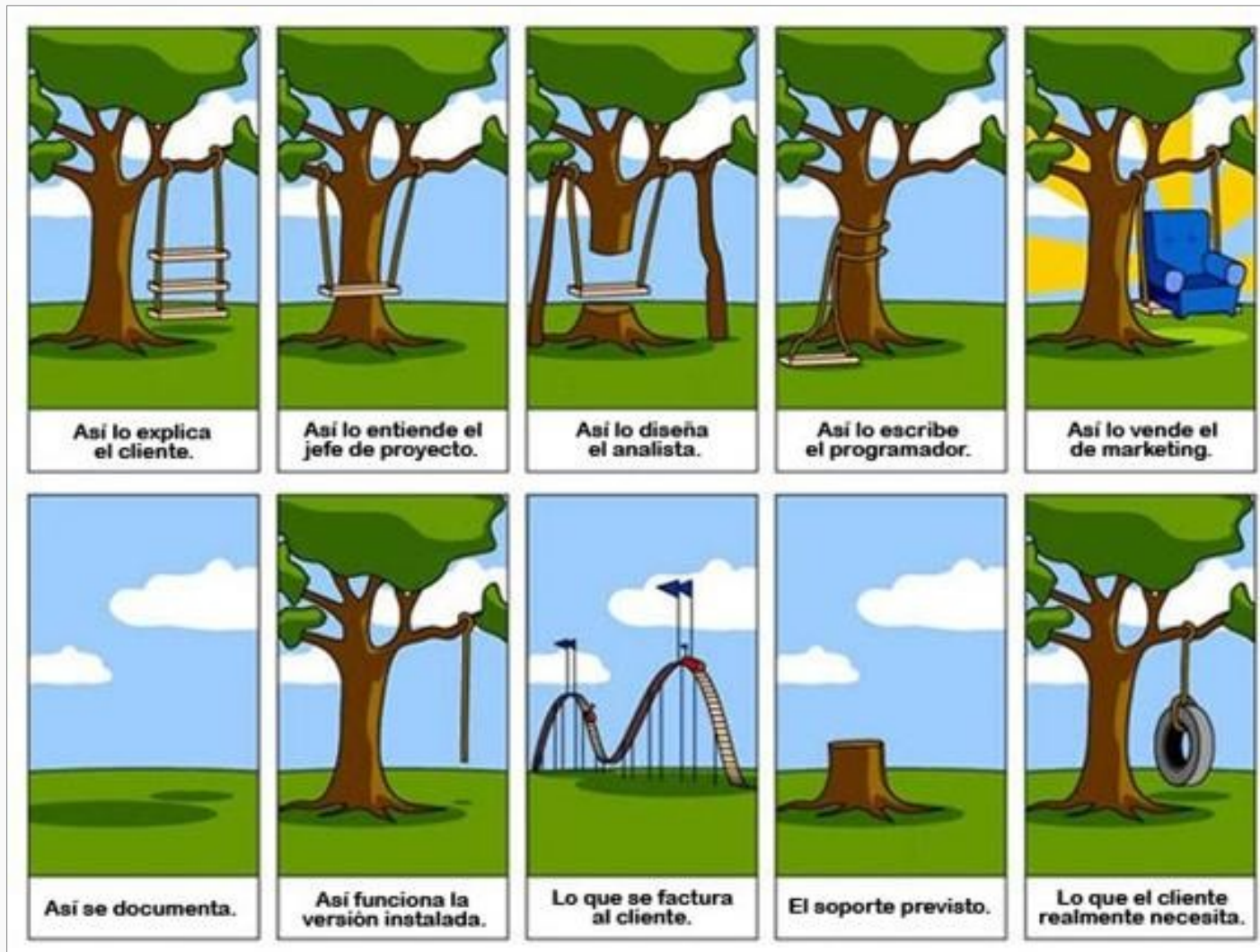
1. INGENIERÍA DEL SOFTWARE

2. CICLO DE VIDA DEL SOFTWARE

3. METODOLOGÍAS



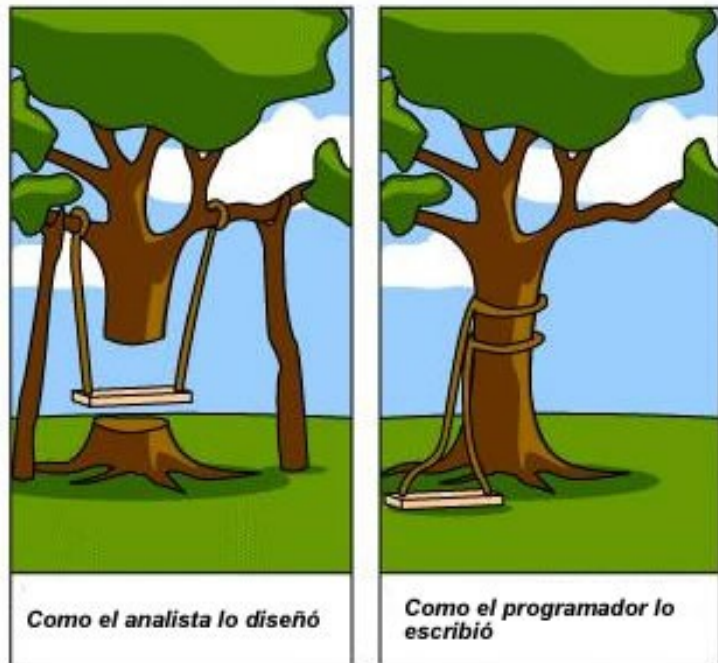
UD 01.Desarrollo de software



Bienvenidos/as al mundo del Desarrollo de Software.



1. INGENIERÍA DEL SOFTWARE (ISW)

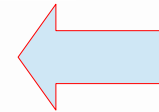
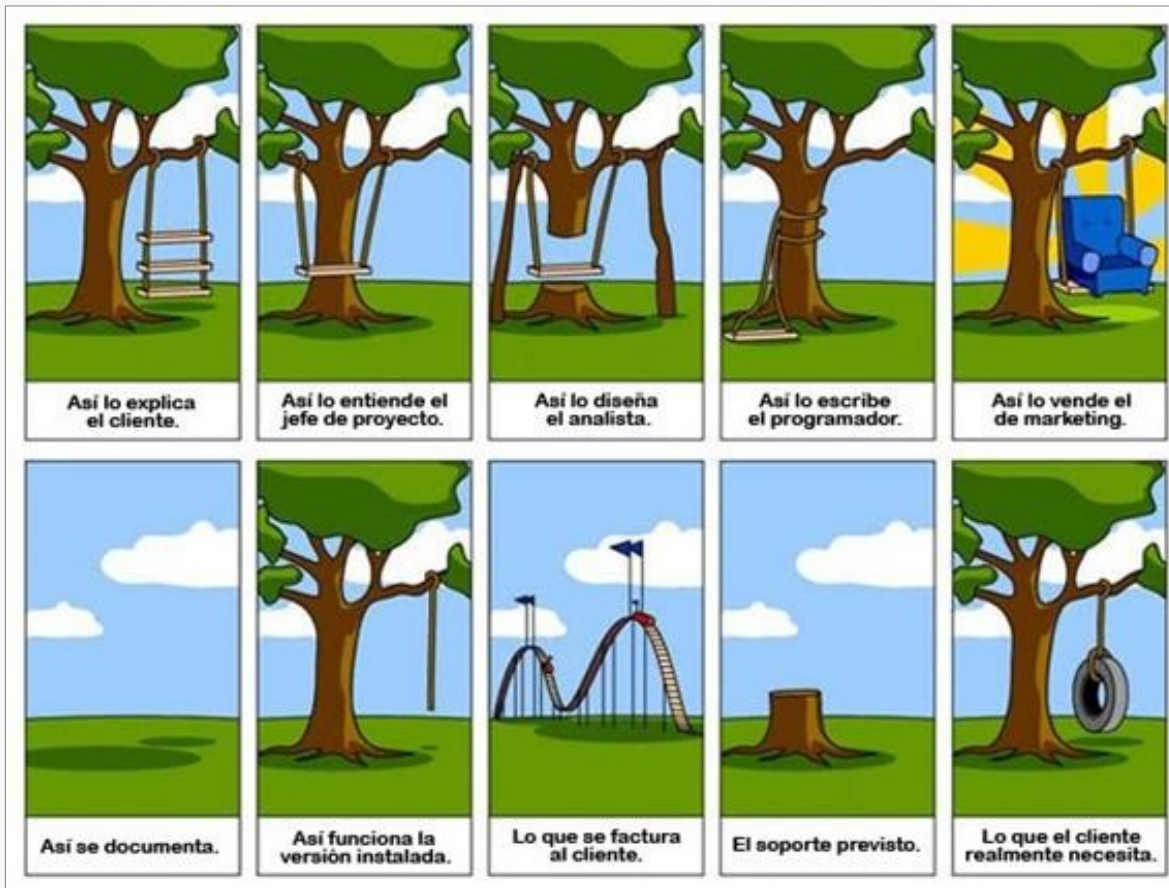


- El término “**Crisis del Software**” fue acuñado a principios de los años 70, cuando la planificación de un proyecto software era prácticamente inexistente.
- Seguían un proceso de desarrollo bastante **artesanal**, sin una metodología o un camino a seguir para su desarrollo.
- El análisis previo a la codificación abarcaba **el 8% del tiempo total** de desarrollo de software y el **80% de los errores se producían en esta fase**.
- Con estos indicadores estaba claro que algo estaba fallando y que el **proceso de desarrollo de software necesitaba un cambio radical**.

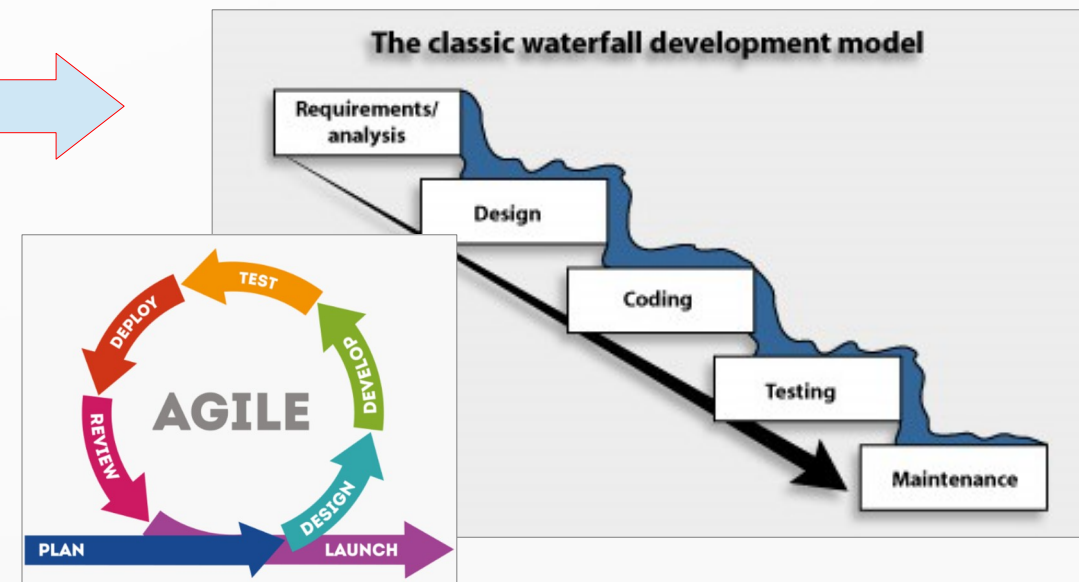
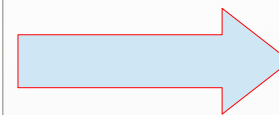
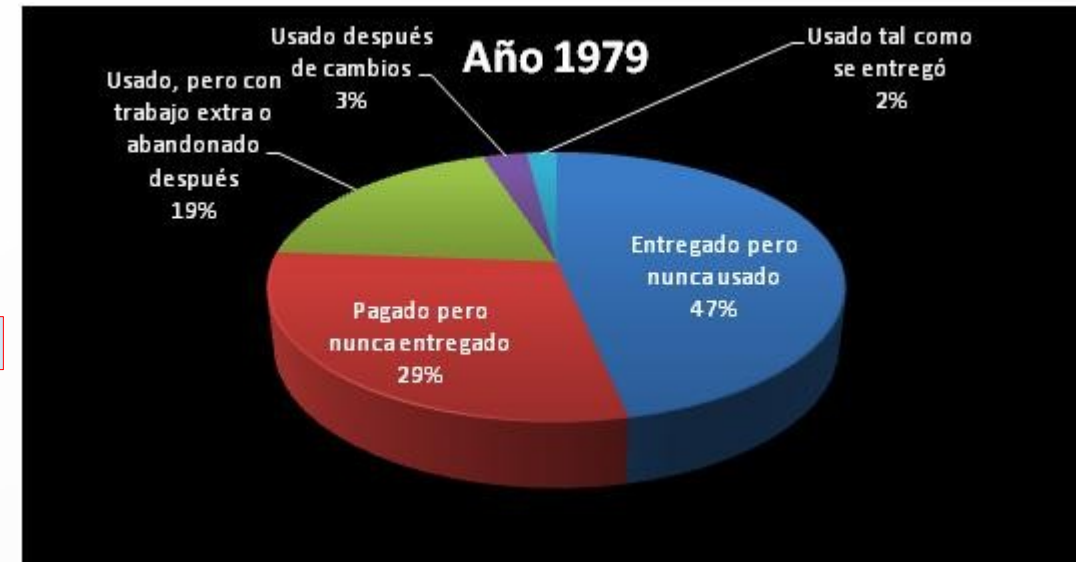
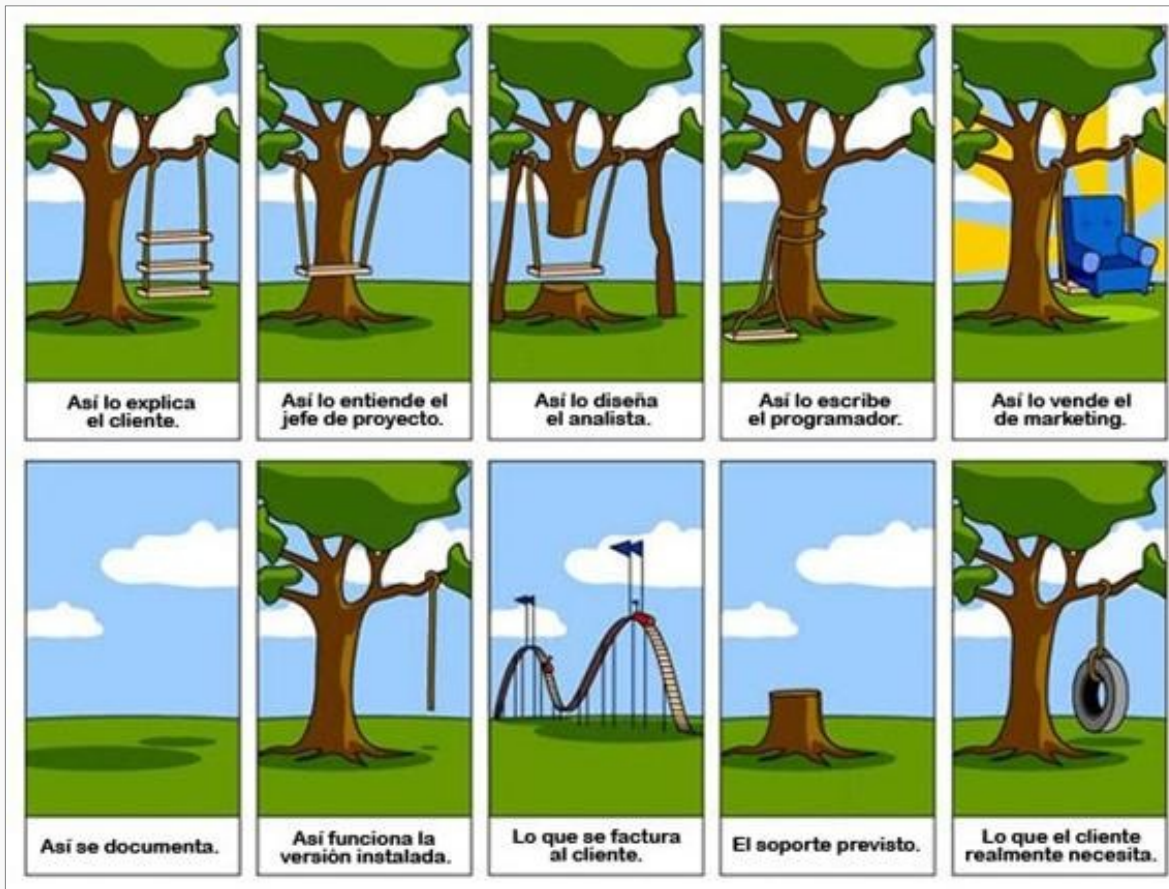
1. INGENIERÍA DEL SOFTWARE (ISW)



1. INGENIERÍA DEL SOFTWARE (ISW)

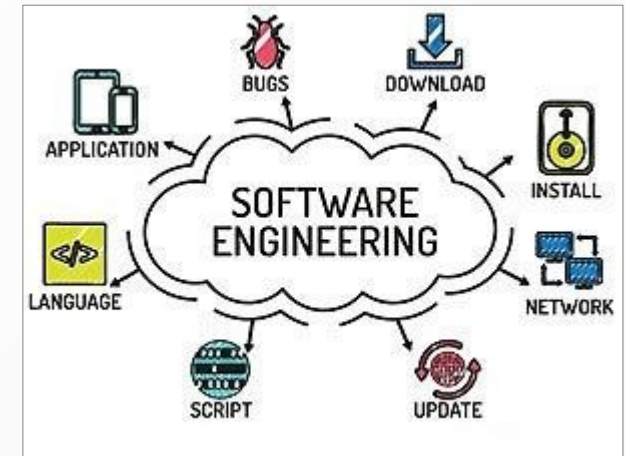


1. INGENIERÍA DEL SOFTWARE (ISW)



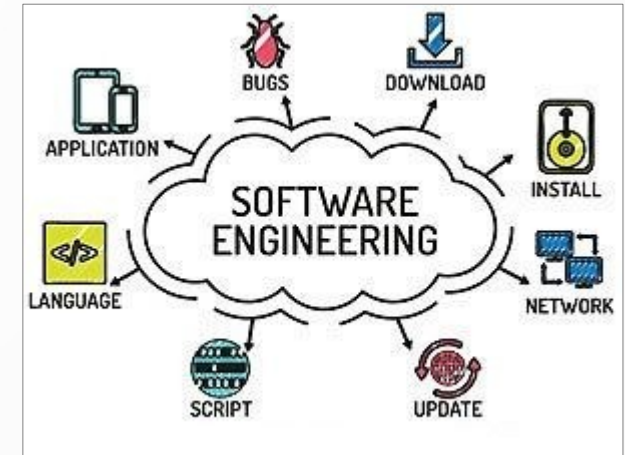
1. INGENIERÍA DEL SOFTWARE (ISW)

- Podríamos definir la **ingeniería** como "la práctica de organizar el diseño y la construcción de cualquier artefacto que transforme el mundo que nos rodea para satisfacer alguna necesidad" (Rogers, 1983).
- El objetivo de toda aplicación informática es satisfacer alguna necesidad - de las personas, las organizaciones o la sociedad en general - con un efecto tangible en el mundo real.
- Por tanto, podríamos definir la **ingeniería del software** es una disciplina que engloba herramientas, técnicas, recomendaciones y métodos que se utilizan en la creación de una aplicación/sistema/aplicativo/plataforma informática o, más genéricamente, en el desarrollo de un proyecto software que resuelve un problema o suple una necesidad.



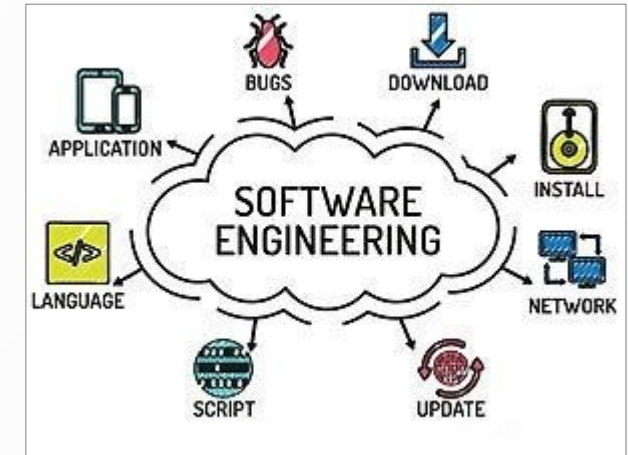
1. INGENIERÍA DEL SOFTWARE (ISW)

- Objetivos de la **ingeniería del software**:
 - Resolver un problema o cubrir una necesidad.
 - En tiempo y forma (en plazo, cumpliendo requisitos y en presupuesto)
- Estadios de la **ingeniería del software**:
 - **Validación**: ¿Estamos resolviendo el problema correcto?
 - **Codificación**: Resolución del problema
 - **Verificación**: ¿Estamos resolviendo el problema correctamente?



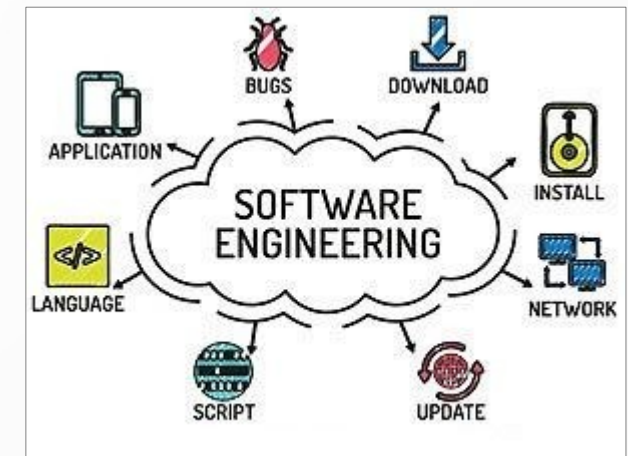
1. INGENIERÍA DEL SOFTWARE (ISW)

- Objetivos de la **ingeniería del software**:
 - Resolver un problema o cubrir una necesidad.
 - En tiempo y forma (en plazo, cumpliendo requisitos y en presupuesto)
- Estadios de la **ingeniería del software**:
 - **Validación**: ¿Estamos resolviendo el problema correcto?
 - (1) Análisis + (2) Diseño
 - (3) **Codificación**: Resolución del problema
 - **Verificación**: ¿Estamos resolviendo el problema correctamente?
 - (4) Pruebas + (5) Documentación + (6) Mantenimiento



1. INGENIERÍA DEL SOFTWARE (ISW)

- Objetivos de la **ingeniería del software**:
 - Resolver un problema o cubrir una necesidad.
 - En tiempo y forma (en plazo, cumpliendo requisitos y en presupuesto)
- Estadios de la **ingeniería del software**:
 - **Validación**: ¿Estamos resolviendo el problema correcto?
 - (1) Análisis + (2) Diseño
 - (3) **Codificación**: Resolución del problema
 - **Verificación**: ¿Estamos resolviendo el problema correctamente?
 - (4) Pruebas + (5) Documentación + (6) Mantenimiento



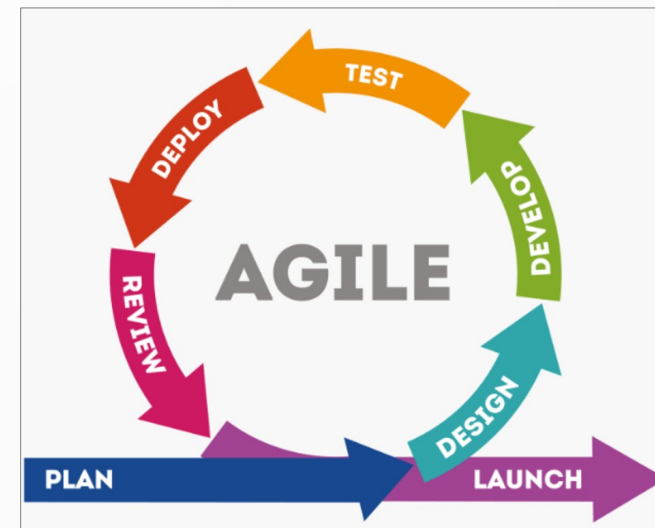
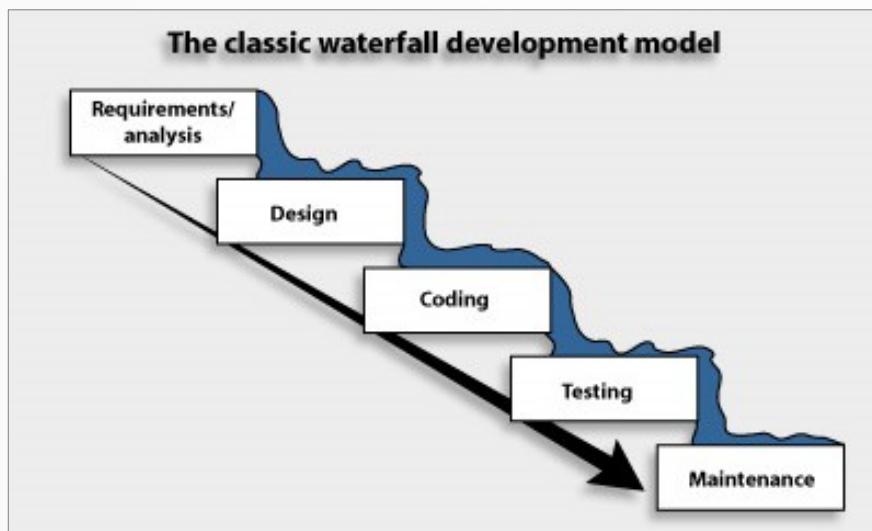
INGENIERÍA DEL SOFTWARE	
ANTES DE CODIFICAR/PROGRAMAR	DESPUÉS DE CODIFICAR/PROGRAMAR
¿Estamos resolviendo el problema correcto?	¿Estamos resolviendo el problema correctamente?
validación => codificación => verificación análisis => diseño => codificación => pruebas => [...] => mantenimiento	

1. INGENIERÍA DEL SOFTWARE (ISW)

INGENIERÍA DEL SOFTWARE	
ANTES DE CODIFICAR/PROGRAMAR	DESPUÉS DE CODIFICAR/PROGRAMAR
¿Estamos resolviendo el problema correcto?	¿Estamos resolviendo el problema correctamente?
validación => codificación => verificación análisis => diseño => codificación => pruebas => [...] => mantenimiento	

MISMAS FASES => DOS ENFOQUES

Cascada vs Ágil



1. INGENIERÍA DEL SOFTWARE (ISW)

- **¿Por qué es tan importante y necesaria la ISW?**
 - En 2004 el coste de los fallos de software en la Unión Europea se estimó en 142.000 millones de euros.
 - El Informe CHAOS sobre las tasas de éxito de los proyectos de software concluyó en **2015** que:
 - **36% tuvieron éxito:** Se entregaron a tiempo, en presupuesto, con las características requeridas.
 - **45% fueron modificados:** Con retraso, por encima del presupuesto, y/o con menos características.
 - **19% fracasaron:** Se cancelaron antes de su finalización o se entregaron y nunca se utilizaron.

TRADITIONAL RESOLUTION FOR ALL PROJECTS					
	1994	2012	2013	2014	2015
SUCCESSFUL	16%	37%	41%	36%	36%
CHALLENGED	53%	46%	40%	47%	45%
FAILED	31%	17%	19%	17%	19%

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

1. INGENIERÍA DEL SOFTWARE (ISW)

- **¿Por qué es tan importante y necesaria la ISW?**
 - En 2004 el coste de los fallos de software en la Unión Europea se estimó en 142.000 millones de euros.
 - El Informe CHAOS sobre las tasas de éxito de los proyectos de software concluyó en **2015** que:
 - **36% tuvieron éxito:** Se entregaron a tiempo, en presupuesto, con las características requeridas.
 - **45% fueron modificados:** Con retraso, por encima del presupuesto, y/o con menos características.
 - **19% fracasaron:** Se cancelaron antes de su finalización o se entregaron y nunca se utilizaron.

TRADITIONAL RESOLUTION FOR ALL PROJECTS					
	1994	2012	2013	2014	2015
SUCCESSFUL	16%	37%	41%	36%	36%
CHALLENGED	53%	46%	40%	47%	45%
FAILED	31%	17%	19%	17%	19%

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



1. INGENIERÍA DEL SOFTWARE (ISW)

TRADITIONAL RESOLUTION FOR ALL PROJECTS

	1994	2012	2013	2014	2015
SUCCESSFUL	16%	37%	41%	36%	36%
CHALLENGED	53%	46%	40%	47%	45%
FAILED	31%	17%	19%	17%	19%

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

PROJECT SUCCESS RATES AGILE VS WATERFALL

METHOD	SUCCESSFUL	CHALLENGED	FAILED
AGILE	42%	50%	8%
WATERFALL	26%	53%	21%

WWW.VITALITYCHICAGO.COM

Source: Standish Group Chaos Studies 2013-2017

¿Te parecen aceptables estos porcentajes?



1. INGENIERÍA DEL SOFTWARE (ISW)

TRADITIONAL RESOLUTION FOR ALL PROJECTS

	1994	2012	2013	2014	2015
SUCCESSFUL	16%	37%	41%	36%	36%
CHALLENGED	53%	46%	40%	47%	45%
FAILED	31%	17%	19%	17%	19%

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

PROJECT SUCCESS RATES AGILE VS WATERFALL

METHOD	SUCCESSFUL	CHALLENGED	FAILED
AGILE	42%	50%	8%
WATERFALL	26%	53%	21%

WWW.VITALITYCHICAGO.COM

Source: Standish Group Chaos Studies 2013-2017

La ISW sigue trabajando para mejorar esos porcentajes y tiene varias propuestas:



1. INGENIERÍA DEL SOFTWARE (ISW)

TRADITIONAL RESOLUTION FOR ALL PROJECTS

	1994	2012	2013	2014	2015
SUCCESSFUL	16%	37%	41%	36%	36%
CHALLENGED	53%	46%	40%	47%	45%
FAILED	31%	17%	19%	17%	19%

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

PROJECT SUCCESS RATES AGILE VS WATERFALL

METHOD	SUCCESSFUL	CHALLENGED	FAILED
AGILE	42%	50%	8%
WATERFALL	26%	53%	21%

WWW.VITALITYCHICAGO.COM

Source: Standish Group Chaos Studies 2013-2017

La ISW sigue trabajando para mejorar esos porcentajes y tiene varias propuestas:

- **Latencia de decisión:** Las decisiones deben tomarse lo antes posible.
- **Alcance mínimo:** A mayor tamaño del proyecto mayor índice de fracaso.
- **Responsables del proyecto:** Mejor una única persona que no una Junta Directiva, por ejemplo.

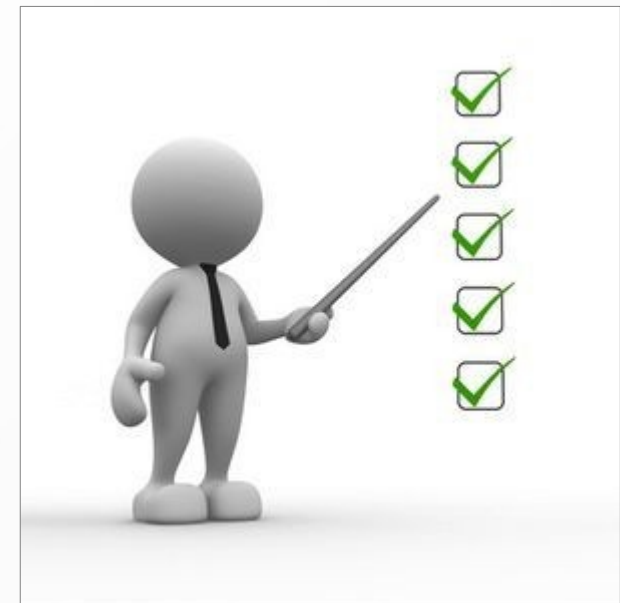


UD 01.Desarrollo de software

1. INGENIERÍA DEL SOFTWARE

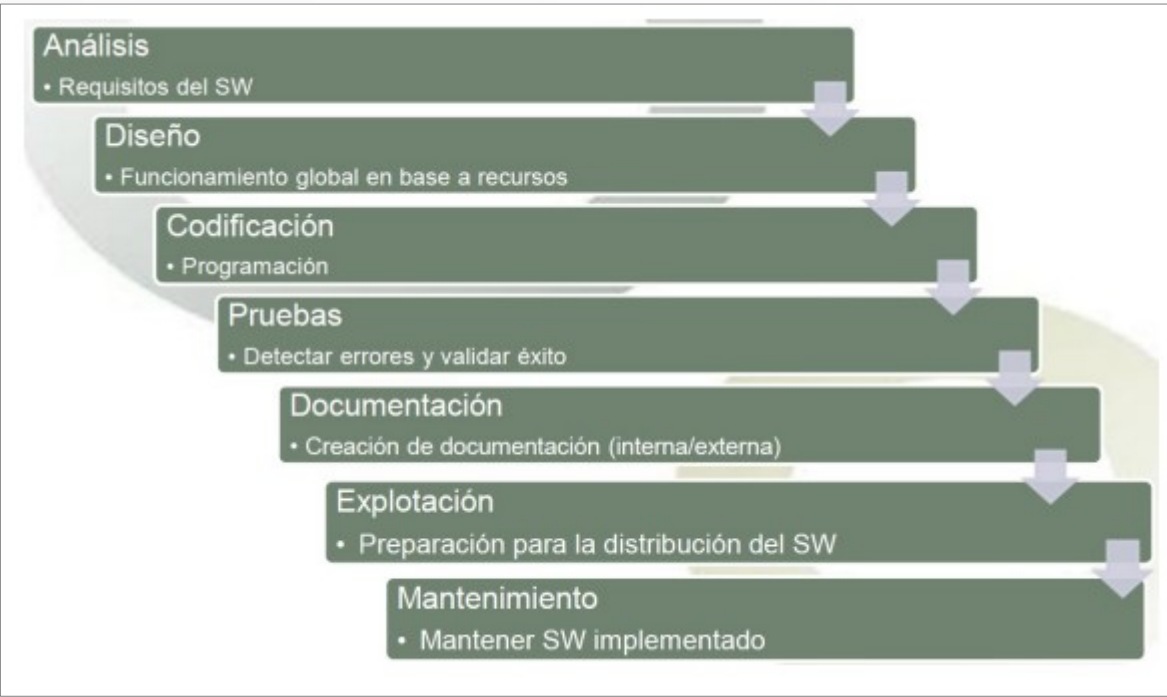
2. CICLO DE VIDA DEL SOFTWARE

3. METODOLOGÍAS



2. CICLO DE VIDA DEL SOFTWARE

- Las fases del desarrollo de software, conocidas como **ciclo de vida del software**, representan las etapas por las que debe pasar una aplicación para ser entregada a su destinatario con todas las garantías.



FASE	Perfil profesional	Entregables
Análisis ¿QUÉ VOY A HACER? <i>Descripción del problema</i>	Jefe de Proyecto Arquitecto Software Consultor	Especificación de requisitos Prototipos Diagrama de casos de uso
El análisis de una aplicación pretende determinar las necesidades que debe cubrir en directo contacto con el cliente. En esta fase se especifican los requisitos funcionales y no funcionales del proyecto.		

2. CICLO DE VIDA DEL SOFTWARE

- Las fases del desarrollo de software, conocidas como **ciclo de vida del software**, representan las etapas por las que debe pasar una aplicación para ser entregada a su destinatario con todas las garantías.

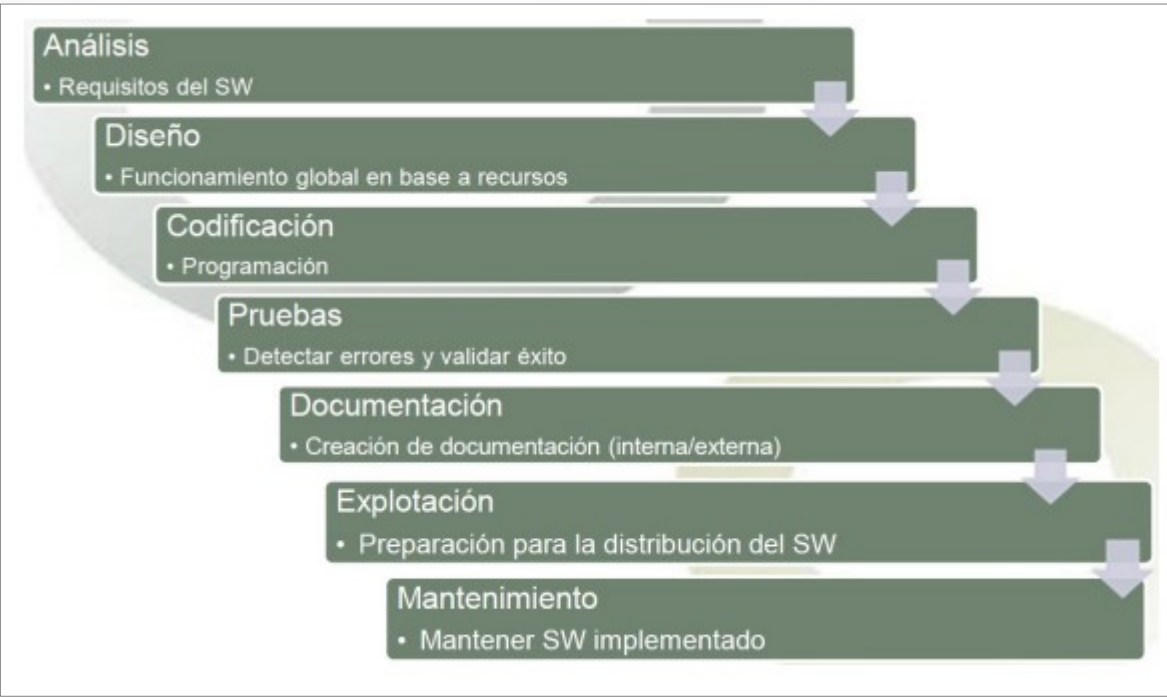


FASE	Perfil profesional	Entregables
Análisis ¿QUÉ VOY A HACER? <i>Descripción del problema</i>	Jefe de Proyecto Arquitecto Software Consultor	Especificación de requisitos Prototipos Diagrama de casos de uso
El análisis de una aplicación pretende determinar las necesidades que debe cubrir en directo contacto con el cliente. En esta fase se especifican los requisitos funcionales y no funcionales del proyecto.		

FASE	Perfil profesional	Entregables
Diseño ¿CÓMO LO VOY A HACER? <i>Descripción de la solución</i>	Analista Funcional Arquitecto Software Analista Programador (AP)	Diagramas de estructura Diagramas de comportamiento
Decidimos en fase de diseño cómo abordar la solución (tablas, clases, métodos...). Una vez sabemos cuál es el problema tenemos que ver cuál es la mejor solución antes de escribir una sola línea de código.		

2. CICLO DE VIDA DEL SOFTWARE

- Las fases del desarrollo de software, conocidas como **ciclo de vida del software**, representan las etapas por las que debe pasar una aplicación para ser entregada a su destinatario con todas las garantías.



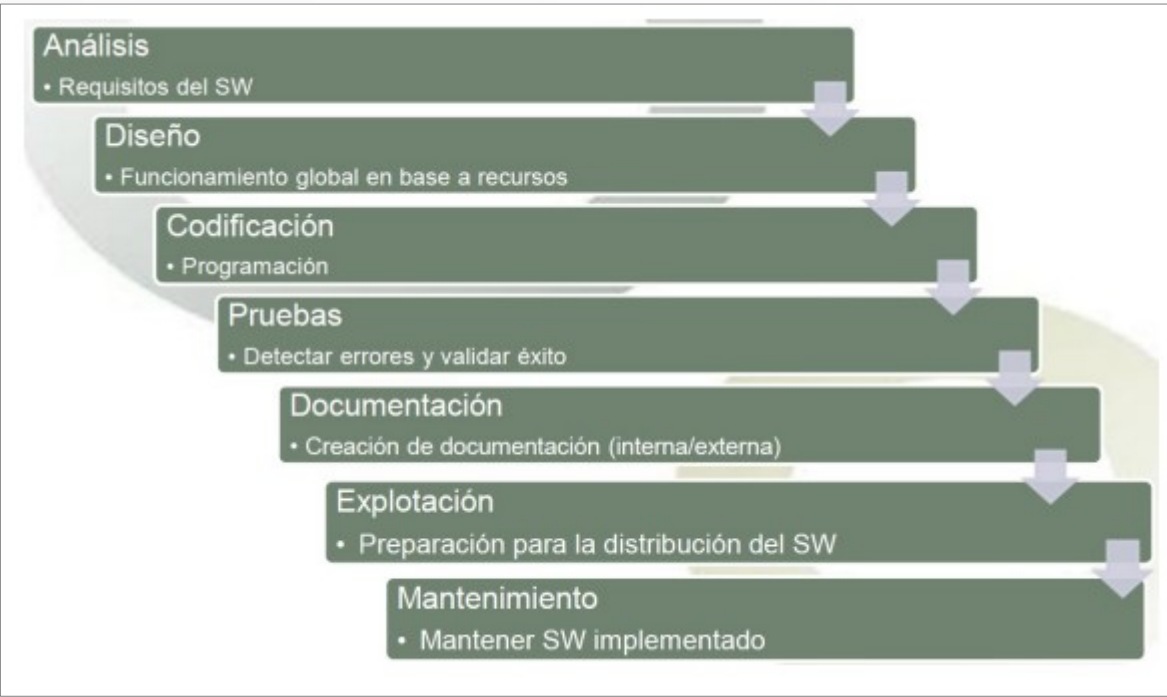
FASE	Perfil profesional	Entregables
Análisis ¿QUÉ VOY A HACER? <i>Descripción del problema</i>	Jefe de Proyecto Arquitecto Software Consultor	Especificación de requisitos Prototipos Diagrama de casos de uso
El análisis de una aplicación pretende determinar las necesidades que debe cubrir en directo contacto con el cliente. En esta fase se especifican los requisitos funcionales y no funcionales del proyecto.		

FASE	Perfil profesional	Entregables
Diseño ¿CÓMO LO VOY A HACER? <i>Descripción de la solución</i>	Analista Funcional Arquitecto Software Analista Programador (AP)	Diagramas de estructura Diagramas de comportamiento
Decidimos en fase de diseño cómo abordar la solución (tablas, clases, métodos...). Una vez sabemos cuál es el problema tenemos que ver cuál es la mejor solución antes de escribir una sola línea de código.		

FASE	Perfil profesional	Entregables
Codificación o implementación	AP / Programador Administrador de BBDD	Código fuente / Librerías Ejecutable(s) Bases de datos (sistemas de información)
En la codificación se realiza el programa atendiendo a todos sus componentes; esto incluye elementos como la base de datos, servidores o comunicaciones.		

2. CICLO DE VIDA DEL SOFTWARE

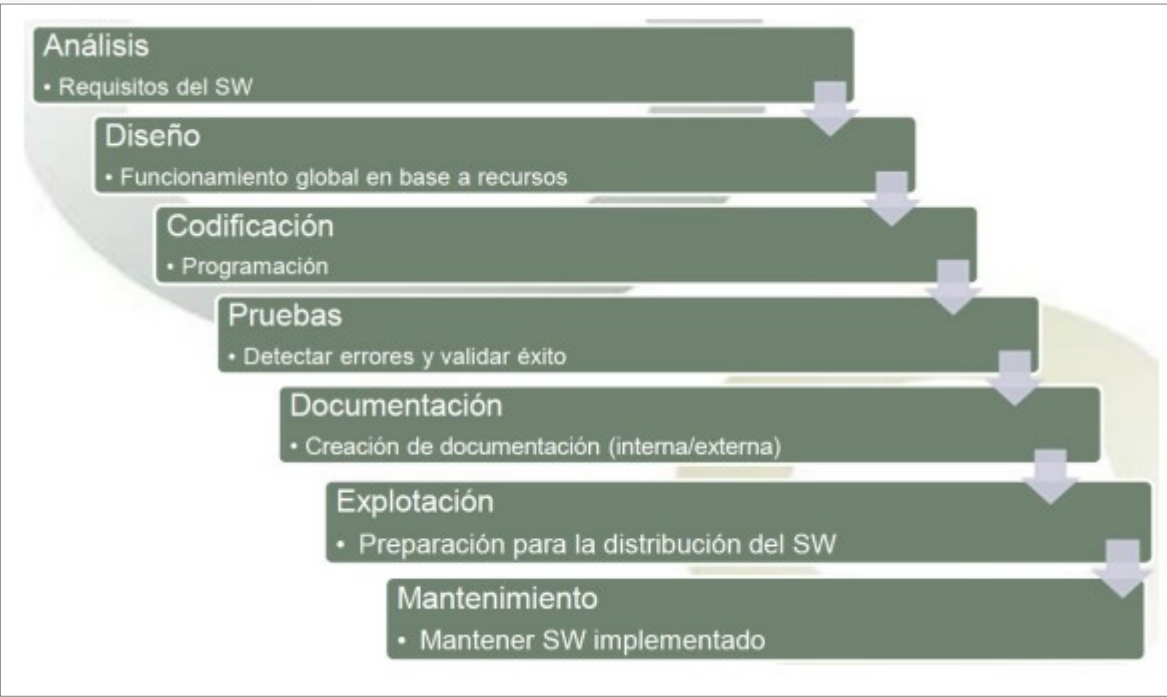
- Las fases del desarrollo de software, conocidas como **ciclo de vida del software**, representan las etapas por las que debe pasar una aplicación para ser entregada a su destinatario con todas las garantías.



FASE	Perfil profesional	Entregables
Pruebas	Tester AP / Programador	Proyecto VERIFICADO
Las pruebas son revisiones ADICIONALES que deben realizar personas distintas a las que codificaron el aplicativo para detectar errores de usabilidad o errores no detectados en la fase anterior.		

2. CICLO DE VIDA DEL SOFTWARE

- Las fases del desarrollo de software, conocidas como **ciclo de vida del software**, representan las etapas por las que debe pasar una aplicación para ser entregada a su destinatario con todas las garantías.

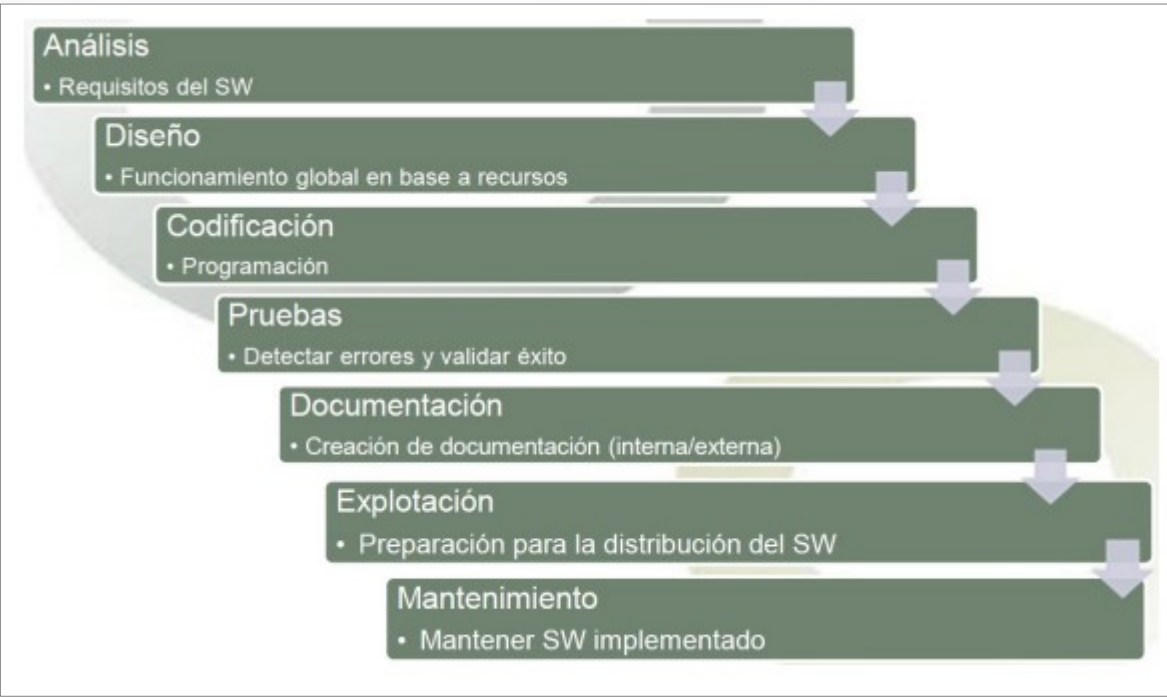


FASE	Perfil profesional	Entregables
Pruebas	Tester AP / Programador	Proyecto VERIFICADO
Las pruebas son revisiones ADICIONALES que deben realizar personas distintas a las que codificaron el aplicativo para detectar errores de usabilidad o errores no detectados en la fase anterior.		

FASE	Perfil profesional	Entregables
Documentación	Administrativo AP / Programador	Documentación
La documentación es la fase que primero se descarta cuando el proyecto está fuera de plazo o de presupuesto y es vital para que, si algún miembro del equipo debe ser reemplazado, se amplía el equipo o se retoma el proyecto tras un periodo de tiempo poder continuar/comenzar lo antes posible.		

2. CICLO DE VIDA DEL SOFTWARE

- Las fases del desarrollo de software, conocidas como **ciclo de vida del software**, representan las etapas por las que debe pasar una aplicación para ser entregada a su destinatario con todas las garantías.



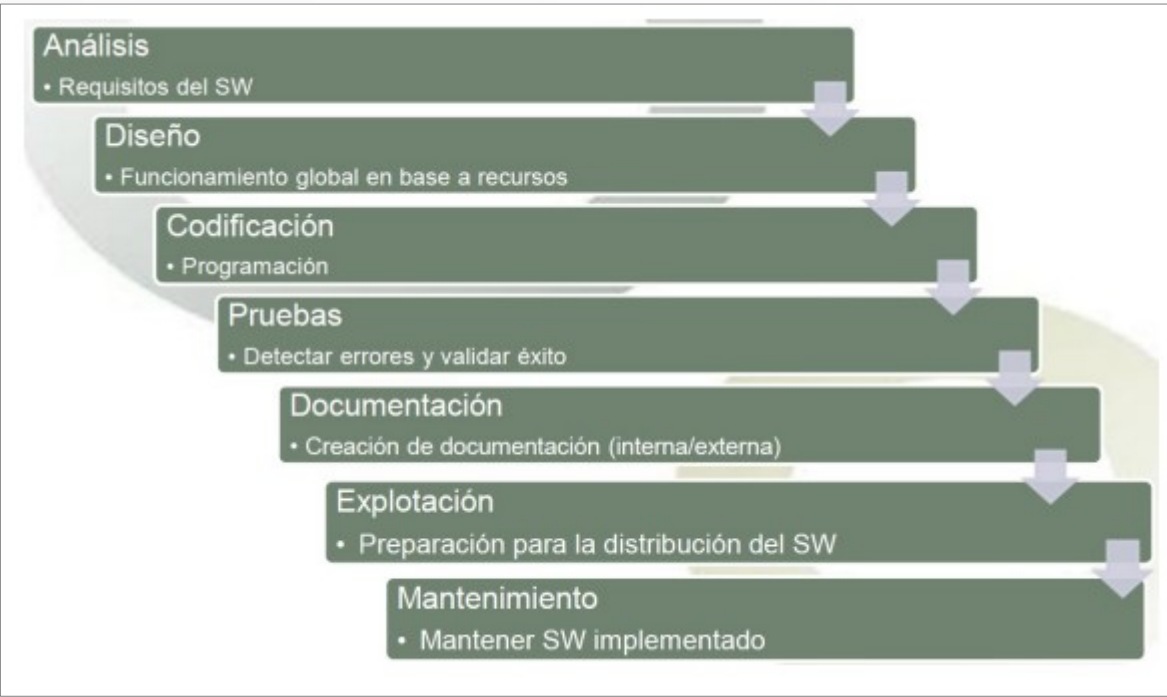
FASE	Perfil profesional	Entregables
Pruebas	Tester AP / Programador	Proyecto VERIFICADO
Las pruebas son revisiones ADICIONALES que deben realizar personas distintas a las que codificaron el aplicativo para detectar errores de usabilidad o errores no detectados en la fase anterior.		

FASE	Perfil profesional	Entregables
Documentación	Administrativo AP / Programador	Documentación
La documentación es la fase que primero se descarta cuando el proyecto está fuera de plazo o de presupuesto y es vital para que, si algún miembro del equipo debe ser reemplazado, se amplía el equipo o se retoma el proyecto tras un periodo de tiempo poder continuar/comenzar lo antes posible.		

FASE	Perfil profesional	Entregables
Explotación o implantación	Miembros Senior	Proyecto publicado / entregado
La explotación (no confundir con implementación) consiste en publicar la solución final en la plataforma destino o entregar al cliente el producto final en el formato acordado.		

2. CICLO DE VIDA DEL SOFTWARE

- Las fases del desarrollo de software, conocidas como **ciclo de vida del software**, representan las etapas por las que debe pasar una aplicación para ser entregada a su destinatario con todas las garantías.



FASE	Perfil profesional	Entregables
Mantenimiento	Técnicos de soporte Miembros Junior	Evolutivo, correctivo y adaptativo
Esta fase de mantenimiento suele tener una sección específica en los contratos de desarrollo software en los que se detalla cómo se va a facturar este servicio y en qué términos.		



2. CICLO DE VIDA DEL SOFTWARE

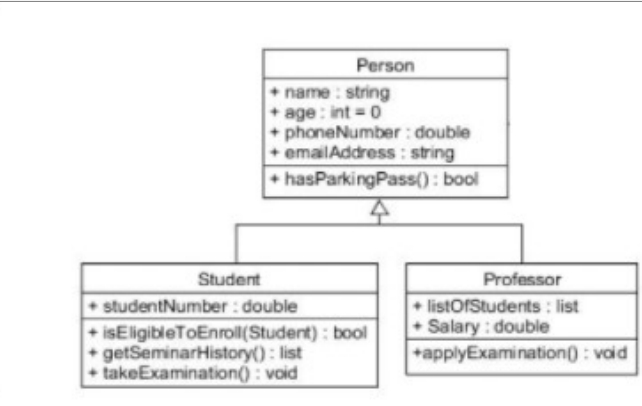
- Durante todo el ciclo de vida del software usamos modelos para representar la realidad.
- La **modelización** se utiliza en otras formas de diseño e ingeniería. Por ejemplo, los arquitectos desarrollan diferentes modelos de edificios - algunos abordan las estructuras, otros los materiales ... Lo mismo ocurre con la modelización del software, donde cada modelo es una representación abstracta y dinámica de alguna visión del sistema, que puede cambiar durante el desarrollo.
- Existen muchas técnicas de modelado que quizás veas en otros módulos de este ciclo:

Técnicas de diseño/modelado del software		
<ul style="list-style-type: none">• Modelo y Notación de Procesos de Negocio (BPMN)	<ul style="list-style-type: none">• Diagramas de Gantt	<ul style="list-style-type: none">• Diagramas UML
<ul style="list-style-type: none">• Flujogramas (flowcharts)	<ul style="list-style-type: none">• Integración para la Modelización de Funciones (IDEF)	<ul style="list-style-type: none">• Diagramas de PERT
<ul style="list-style-type: none">• Diagramas Flujo de Datos de Yourdon-DeMarco (DFDs)	<ul style="list-style-type: none">• Diagramas de Flujo Funcional	<ul style="list-style-type: none">• Redes de Petri

2. CICLO DE VIDA DEL SOFTWARE

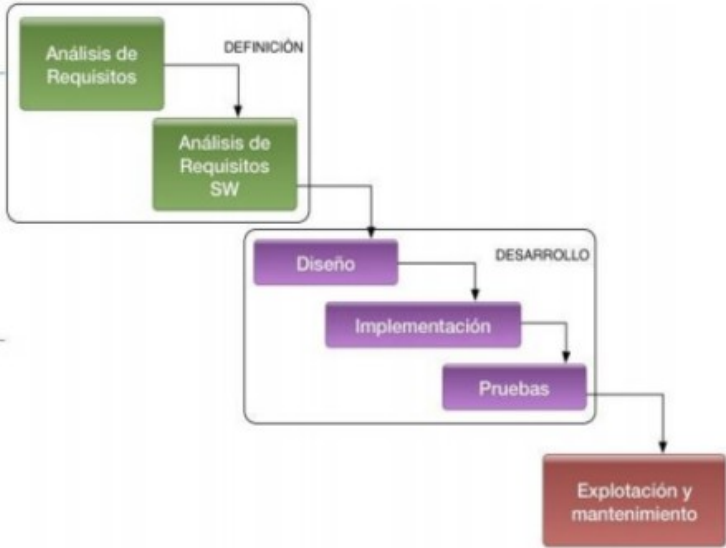
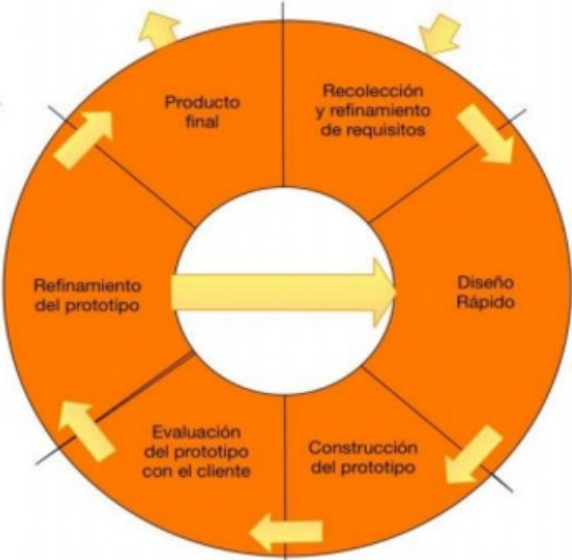
Técnicas de diseño/modelado del software		
<ul style="list-style-type: none">Modelo y Notación de Procesos de Negocio (BPMN)	<ul style="list-style-type: none">Diagramas de Gantt	<ul style="list-style-type: none">Diagramas UML
<ul style="list-style-type: none">Flujogramas (flowcharts)	<ul style="list-style-type: none">Integración para la Modelización de Funciones (IDEF)	<ul style="list-style-type: none">Diagramas de PERT
<ul style="list-style-type: none">Diagramas Flujo de Datos de Yourdon-DeMarco (DFDs)	<ul style="list-style-type: none">Diagramas de Flujo Funcional	<ul style="list-style-type: none">Redes de Petri

Análisis	<ul style="list-style-type: none">Diagrama de casos de uso
Diseño	<ul style="list-style-type: none">D. clase, secuencia y estados
Codificación	<ul style="list-style-type: none">D. de paquetes y despliegue
Pruebas	<ul style="list-style-type: none">Diagrama de clases
Mantenimiento	<ul style="list-style-type: none">Diagrama de Casos de uso
	<ul style="list-style-type: none">Todos



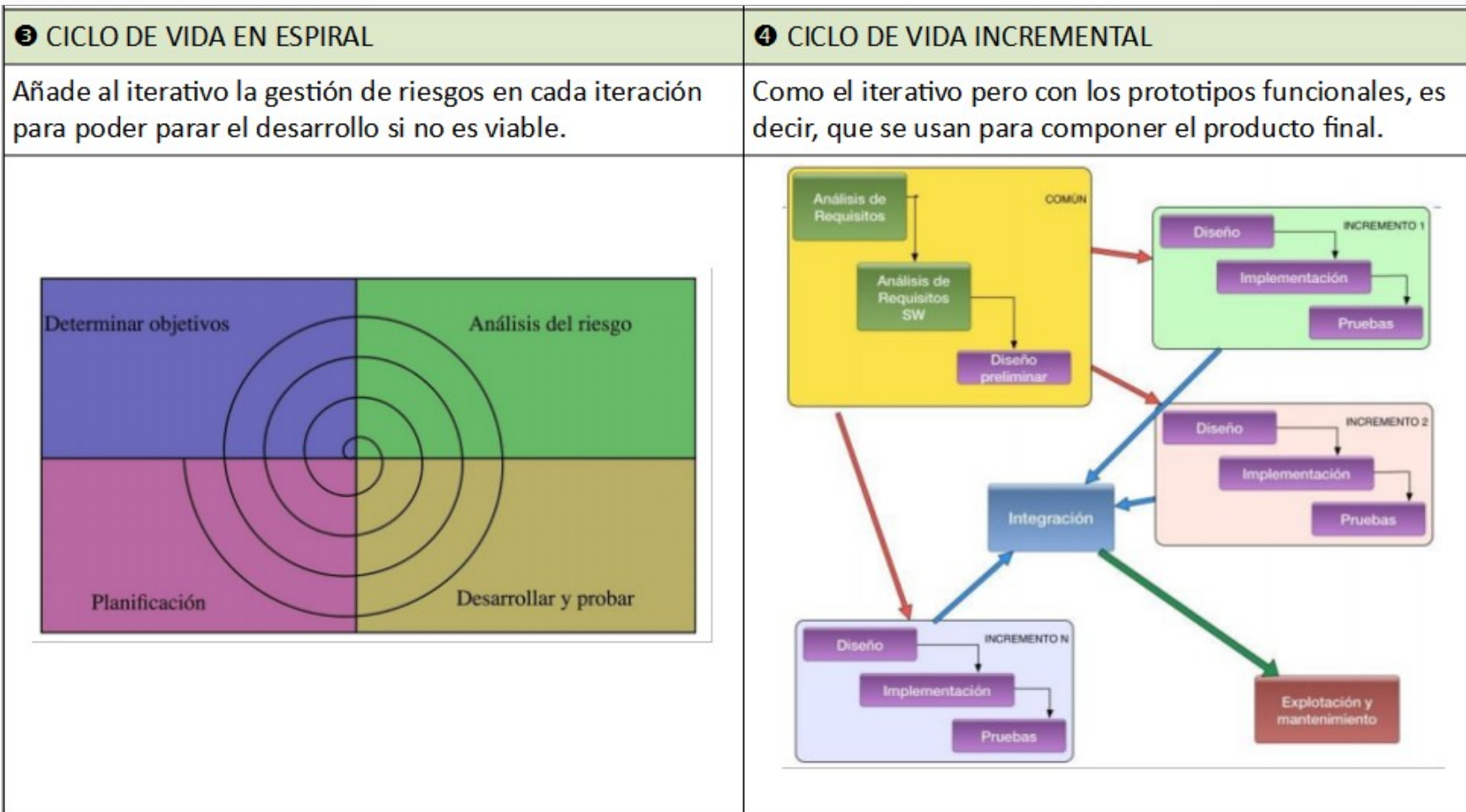
2. CICLO DE VIDA DEL SOFTWARE

En función de cómo se secuencien las fases que hemos visto antes, tendremos estos tipos:

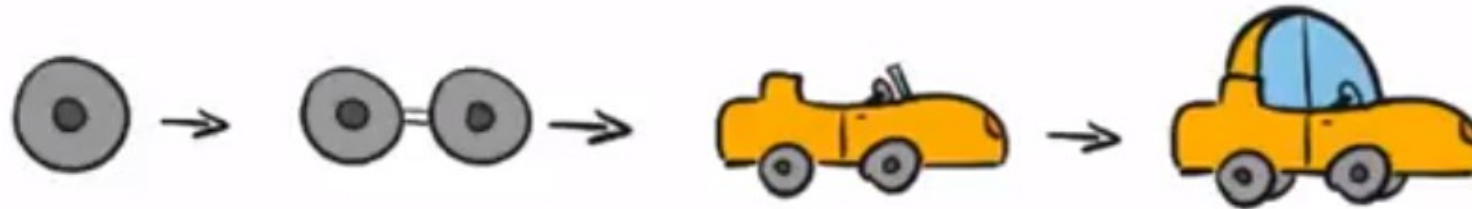
1 CICLO DE VIDA CLÁSICO O EN CASCADA	2 CICLO DE VIDA ITERATIVO
Sin retroalimentación (lineal). Solo válido en proyectos grandes con requisitos muy claros.	Se repiten las fases en bucle hasta tener el producto final. Se entregan maquetas con cada iteración que se desechan tras entregarlas al cliente (no funcionales).
 <pre>graph TD subgraph DEFINICIÓN A[Análisis de Requisitos] --> B[Análisis de Requisitos SW] end B --> subgraph DESARROLLO C[Diseño] --> D[Implementación] D --> E[Pruebas] end E --> F[Explotación y mantenimiento]</pre>	 <pre>graph TD A[Recolección y refinamiento de requisitos] --> B[Diseño Rápido] B --> C[Construcción del prototipo] C --> D[Evaluación del prototipo con el cliente] D --> E[Refinamiento del prototipo] E --> A E --> F[Producto final]</pre>

2. CICLO DE VIDA DEL SOFTWARE

En función de cómo se secuencien las fases que hemos visto antes, tendremos estos tipos:



2. CICLO DE VIDA DEL SOFTWARE



MÓDELO ITERATIVO

MÓDELO ITERATIVO E INCREMENTAL



UD 01.Desarrollo de software

1. INGENIERÍA DEL SOFTWARE
2. CICLO DE VIDA DEL SOFTWARE
- 3. METODOLOGÍAS**



3. METODOLOGÍAS

Los ciclos de vida se **combinan y explotan** para crear metodologías que ofrecen técnicas concretas para cada etapa o fase de los ciclos de vida escogidos. Existen tres enfoques:

- Metodologías tradicionales
- Metodologías ágiles (2001)
- Metodologías guiadas por pruebas o TestDriven (TDD) (2003)

3. METODOLOGÍAS

Los ciclos de vida se **combinan y explotan** para crear metodologías que ofrecen técnicas concretas para cada etapa o fase de los ciclos de vida escogidos. Existen tres enfoques:

- **Metodologías tradicionales**
 - Basadas en los ciclos de vida anteriores. Fases demasiado largas y feedback insuficiente.
 - Ejemplos: METRICA3, RUP, MERISE
- Metodologías ágiles (2001)
- Metodologías guiadas por pruebas o TestDriven (TDD) (2003)

3. METODOLOGÍAS

Los ciclos de vida se **combinan y explotan** para crear metodologías que ofrecen técnicas concretas para cada etapa o fase de los ciclos de vida escogidos. Existen tres enfoques:

- Metodologías tradicionales
- **Metodologías ágiles (2001)**
 - Fomentan el reajuste continuo de los objetivos de desarrollo con las necesidades y expectativas del cliente, y proporcionan procesos de desarrollo de software "más ligeros", más rápidos y más "ágiles" que pueden adaptarse a los inevitables cambios en los requisitos del cliente.
 - Ejemplos: SCRUM, KANBAN, XP PROGRAMMING
- Metodologías guiadas por pruebas o TestDriven (TDD) (2003)

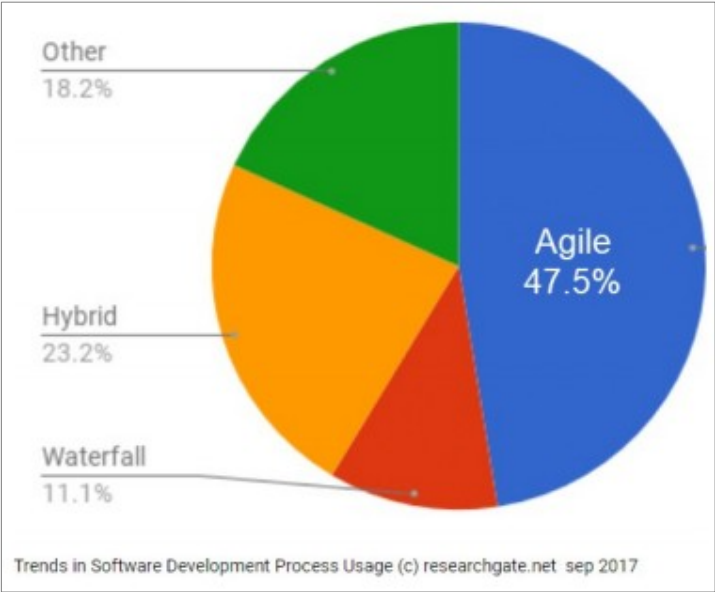
3. METODOLOGÍAS

Los ciclos de vida se **combinan y explotan** para crear metodologías que ofrecen técnicas concretas para cada etapa o fase de los ciclos de vida escogidos. Existen tres enfoques:

- Metodologías tradicionales
- Metodologías ágiles (2001)
- **Metodologías guiadas por pruebas o TestDriven (TDD) (2003)**
 - TDD o Test-Driven Development (desarrollo dirigido por tests) es una práctica de programación que consiste en escribir primero las pruebas (generalmente unitarias), después escribir el código fuente que pase la prueba satisfactoriamente y, por último, refactorizar el código escrito.

3. METODOLOGÍAS

Los ciclos de vida se **combinan y explotan** para crear metodologías que ofrecen técnicas concretas para cada etapa o fase de los ciclos de vida escogidos.



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.

UD 01.Desarrollo de software

RESUMEN:

- => CRISIS DEL SW (70's)
- => M. TRADICIONALES (80's)
- => M. ÁGILES (90's)
- => M. TDD (00's)

