

UD 01

DESARROLLO DE SOFTWARE

ENTORNOS DE DESARROLLO 20/21
CFGS DAW

PARTE 2 DE 2: INGENIERÍA DEL SOFTWARE

Autor: Sergio Badal

sergio.badal@ceedcv.es

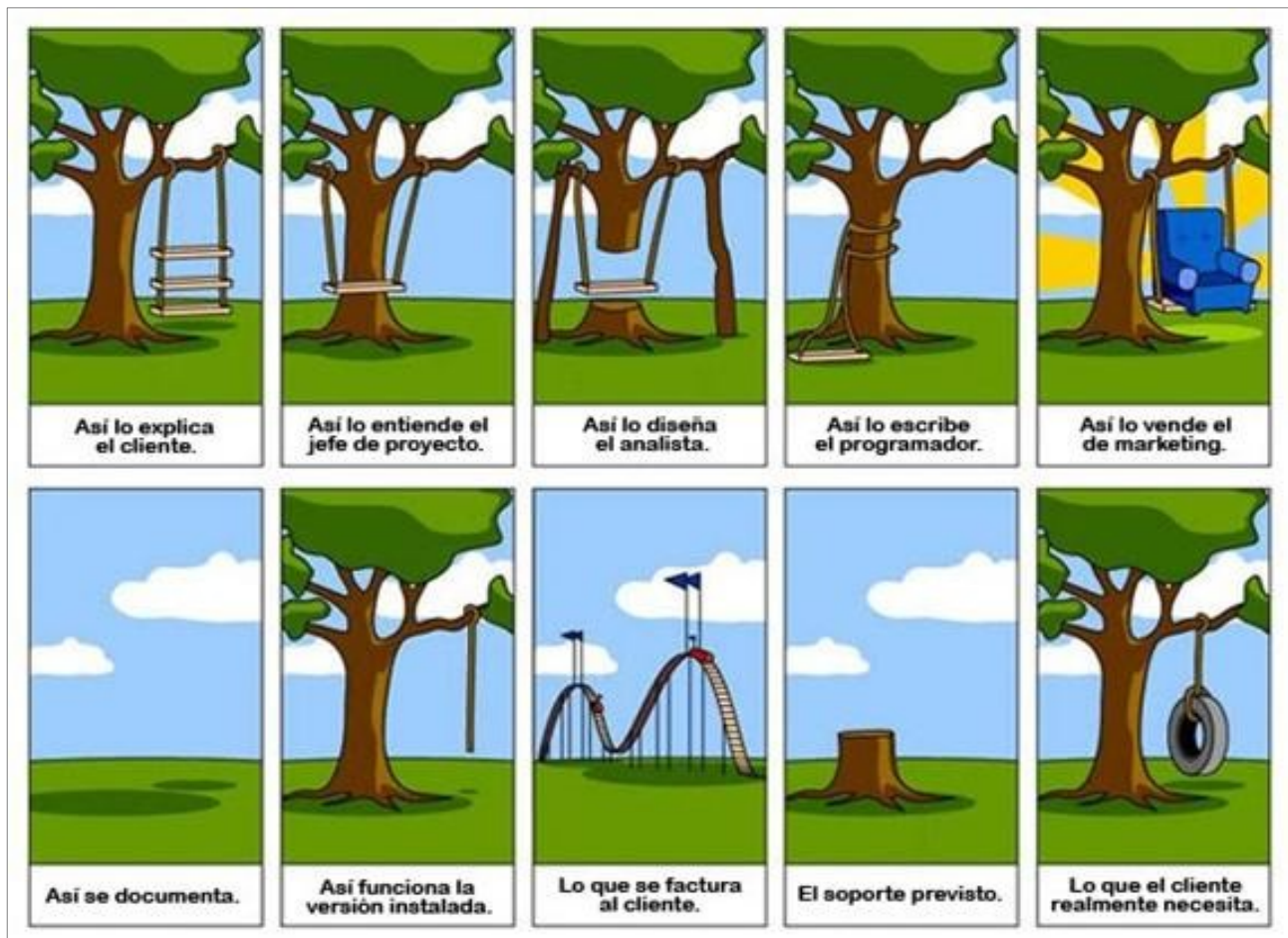
Fecha: 2-10-2020

Licencia Creative Commons

versión 2.0



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.



ÍNDICE DE CONTENIDOS

1. INGENIERÍA DEL SOFTWARE.....	1
1.1 ¿Qué es la Ingeniería del Software?.....	1
1.2 ¿Por qué es tan importante y necesaria la ISW?.....	2
2. CICLO DE VIDA DEL SOFTWARE.....	3
2.1 Fases más comunes.....	3
2.2 Tipos de ciclos de vida.....	7
2.3 Técnicas de modelado.....	8
3. METODOLOGÍAS.....	9
4. CONTENIDO EXTRA PARA MENTES INQUIETAS.....	10
4.1 Nueva metodología experimental.....	10
4.2 Recursos sobre Ingeniería del Software.....	10
4.3 Actividad sobre metodologías ágiles.....	10
5. BIBLIOGRAFÍA.....	11

1. INGENIERÍA DEL SOFTWARE

1.1 ¿Qué es la Ingeniería del Software?

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).

Rogers no era un desarrollador de software, sino un ingeniero especializado en desarrollo de motores. Sin embargo, su definición capta la esencia de la ingeniería y tiene una amplia aplicabilidad que se extiende al software. Gran parte de nuestro esfuerzo en el desarrollo de aplicaciones informáticas es el diseño y la construcción de programas informáticos para satisfacer alguna necesidad - de las personas, las organizaciones o la sociedad en general - con un efecto tangible en el mundo real.

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.



Importante

*El desarrollo de una aplicación se entiende como un intercambio comercial de un **producto** (el software) entre un **desarrollador** y un **cliente** pudiendo ambas partes ser empresas, organizaciones o personas físicas y ser una o varias.*

Esta disciplina tiene la codificación o programación como pilar fundamental pero incluye otras tareas antes y después de esta que son tanto o más importantes que la propia redacción de instrucciones. El ingeniero de software, por tanto, se encarga de toda la gestión del proyecto para que éste se pueda desarrollar en **tiempo y forma**, es decir, en un plazo determinado y con presupuesto y requisitos previstos.

La ingeniería de software, en esencia, incluye el análisis previo de la situación (qué y cuándo lo haremos), el diseño del proyecto (cómo lo haremos), el desarrollo del software, las pruebas necesarias para su correcto funcionamiento y la documentación e implementación del sistema o puesta en marcha.

Podríamos resumir este proceso en dos preguntas que nos haremos antes y después de codificar:

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	

1.2 ¿Por qué es tan importante y necesaria la ISW?

El término **Crisis del Software**¹ 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. El 80% de los errores se producían en esta fase, con lo que se incrementaba el coste de corrección de errores conforme evolucionaba el proyecto. Con estos indicadores estaba claro que algo estaba fallando y que el proceso de desarrollo de software necesitaba un cambio radical.

El Informe CHAOS² de Standish sobre el 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.
- **El 19% fracasaron:** Se cancelaron antes de su finalización o se entregaron y nunca se utilizaron.

Cada año, desde 1994, el Grupo Standish crea una lista de 10 atributos y su peso relativo que ellos llaman los factores de éxito. Los tres primeros para 2018 son:

- **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.

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.

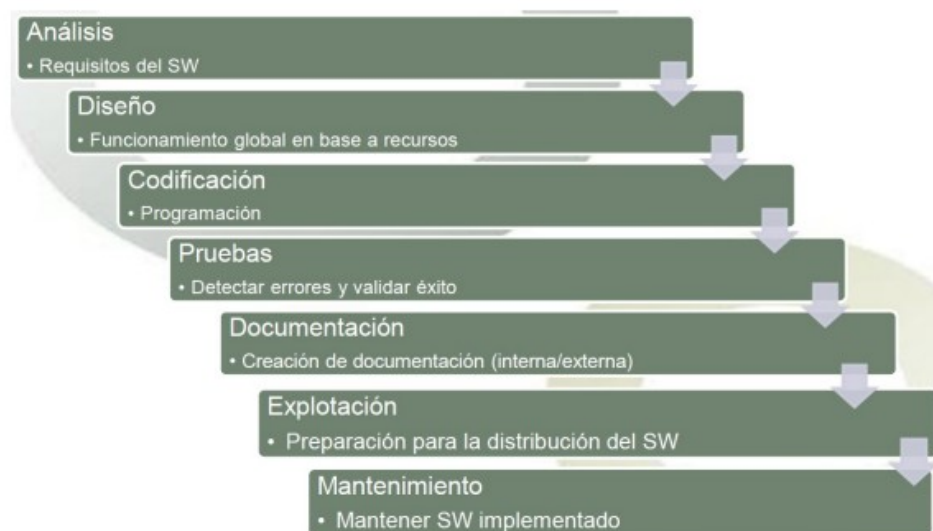
¹ <https://histinf.blogs.upv.es/2011/01/04/la-crisis-del-software/>

² <https://hennyporntman.wordpress.com/2020/01/03/review-chaos-report-2018/>

2. CICLO DE VIDA DEL SOFTWARE

2.1 Fases más comunes

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.



A) ANÁLISIS

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.		

El éxito de un proyecto software se basa en entender y comprender el problema que necesita resolver el cliente y, una vez comprendido, darle solución. La obtención de requisitos se plasma en estos documentos:

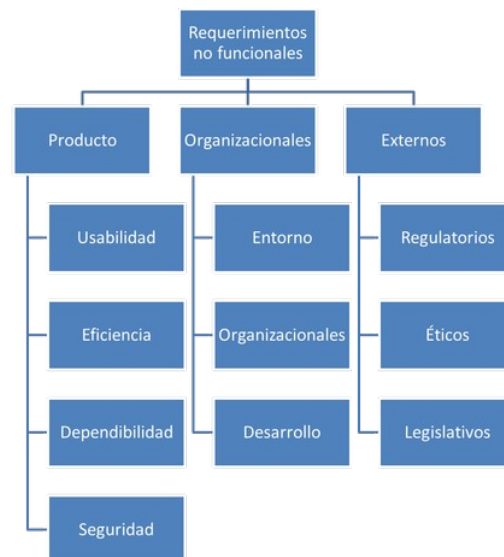
- Una **especificación de requisitos del sistema (ERS)**, con los requisitos funcionales y no funcionales.
 - Aquí un ejemplo: juntadeandalucia.es/servicios/madeja/contenido/recurso/407
- Uno o varios **prototipos**, que pueden ser desde un boceto en papel hasta una versión muy básica del aplicativo pasando por un esquema o wireframe que sirven para describir mejor el problema.
 - Aquí tienes un ejemplo: politicusapp.wordpress.com/imagenes

**Interesante**

Dependiendo del proyecto y del tipo de cliente, el análisis puede concluir con cero o varios ERS y cero o varios prototipos. Lo importante es que todo quede firmado por ambas partes para asegurarnos ante posibles (y frecuentes) cambios de opinión del cliente.

Respecto a los requisitos, se agrupan en:

- **Funcionales:** describen con detalle la función del sistema, es decir, qué debe hacer la aplicación.
- **No funcionales:** trata sobre características del sistema, como pueden ser unos plazos de entrega concretos, unos tiempos mínimos de ejecución o detalles más técnicos que exija por contrato el cliente como que esté desarrollado en determinado lenguaje de programación o que funcione en determinada *plataforma, sistema operativo, procesador* ...



Ian Somerville. Software Engineering. 9na edición

B) DISEÑO

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.		

En el **diseño (o modelado del software)** se plantea una solución para la aplicación y se realiza un modelo utilizando diferentes técnicas. Dicho diseño suele hacerse por niveles, comenzando con una idea general que se va dividiendo en componentes para luego modelar cada uno (diseño descendente/drop-down).

**Importante**

No hay que confundir la fase de DISEÑO en términos de ISW con el DISEÑO GRÁFICO (aspecto de nuestra aplicación). Son cosas completamente diferentes.

C) CODIFICACIÓN

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.		

En muchos proyectos se comete el error de suponer que, por un lado, el programador es el “chico para todo” en el desarrollo del software y que, por otro, cualquier analista o consultor es también un buen programador. No todo profesional del software está capacitado para tratar directamente con el cliente.

D) PRUEBAS

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.		

Se pueden hacer **pruebas** de varios tipos: individuales, de integración y sobre todo de aceptación con el cliente, siempre con la especificación de requisitos (el análisis) delante.

E) DOCUMENTACIÓN

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.		

La fase de **documentación** consiste en dejar por escrito las decisiones técnicas que se han tomado. Desde por qué un framework y no otro hasta por qué se han creado 10 tablas en la base de datos y no 200.

F) EXPLOTACIÓN

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.		

Muy pocas personas en la organización tienen (o deben tener) acceso a los servidores dónde residen las aplicaciones que están publicadas (servidores de producción). Esta fase recae en los más **veteranos (senior)**.

G) MANTENIMIENTO

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.		

Una vez entregado el proyecto este pasa a una fase denominada **mantenimiento** en el que se tratan:

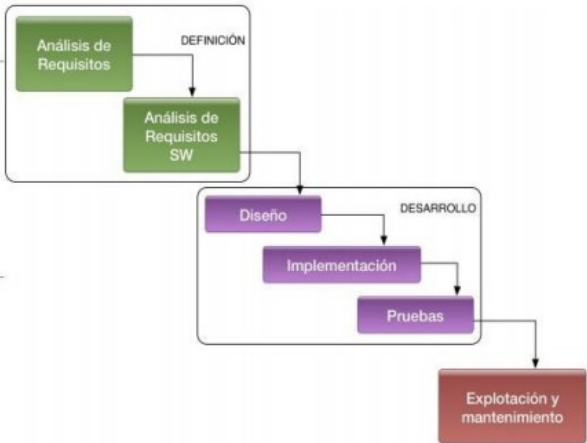
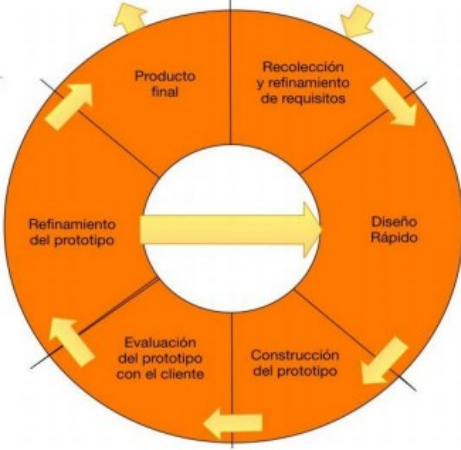
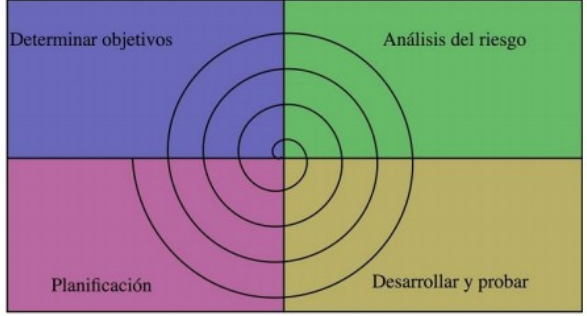
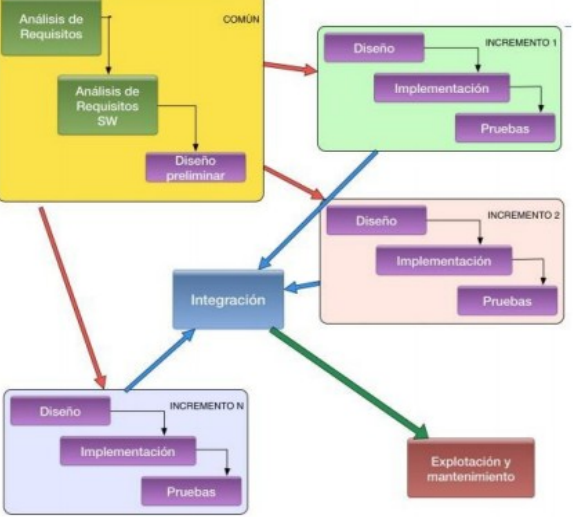
- El **evolutivo**: modificaciones y/o eliminaciones en el producto necesarias por un cambio en las necesidades del cliente o por cambios en requisitos externos (normativos, legales,...).
- El **correctivo**: acciones para mejorar la calidad del producto sistemas en cualquiera de sus aspectos: reestructuración del código, definición más clara del sistema y optimización del rendimiento.
- El **adaptativo**: modificaciones necesarias para adaptarse a variaciones del entorno (contexto) en los que el sistema opera, por ejemplo, cambios de hardware, de base de datos, comunicaciones, etc

**Interesante**

En este punto el software está entregado y facturado por lo que se le suele restar importancia pese a que muchas empresas de desarrollo de software pagan gran parte de las nóminas de sus empleados con las cuotas de mantenimiento de todos sus clientes. Esta fase suele recaer en los recién llegados (junior).

2.2 Tipos de ciclos de vida

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).
	
3 CICLO DE VIDA EN ESPIRAL	4 CICLO DE VIDA INCREMENTAL
Añade al iterativo la gestión de riesgos en cada iteración para poder parar el desarrollo si no es viable.	Como el iterativo pero con los prototipos funcionales, es decir, que se usan para componer el producto final.
	

2.3 Técnicas de modelado

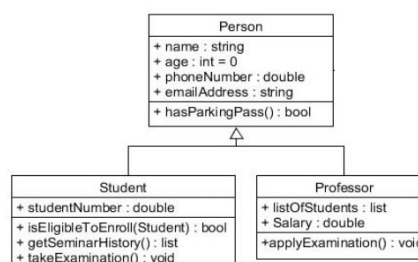
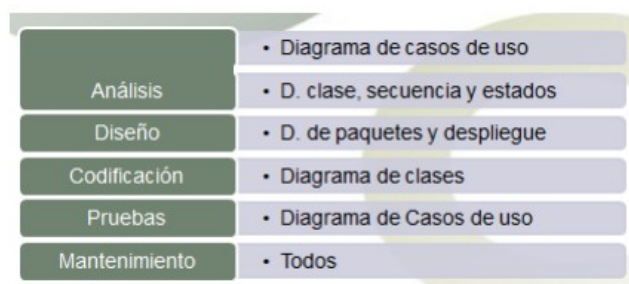
Durante todo el ciclo de vida del software usamos modelos para representar la realidad.

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 modelado³ 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

Las técnicas de modelización más usadas en el desarrollo de software se definen en el **Lenguaje de Modelización Unificado** (UML), uno de los estándares más populares y exitosos actualmente que incluye casi una veintena de diagramas de cuales veremos los más importantes en la segunda parte del módulo.



Puedes echar un vistazo rápido al diagrama UML más popular, el diagrama de clases, [en este vídeo](#).

³ <https://tallyfy.com/business-process-modeling-techniques/>

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 combinaciones de los ciclos de vida anteriores.

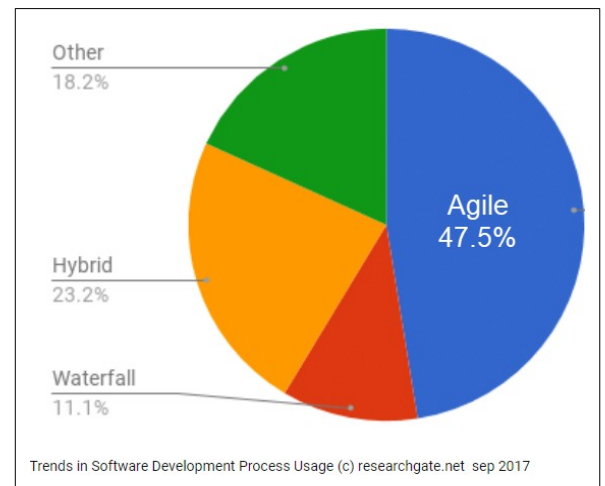
Fases demasiado largas, feedback insuficiente.

- Ejemplos: METRICA3, RUP, MERISE

Metodologías ágiles

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



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.

4. CONTENIDO EXTRA PARA MENTES INQUIETAS

4.1 Nueva metodología experimental

TDD o Test-Driven Development (desarrollo dirigido por tests) es una NUEVA METODOLOGÍA 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. Lo veremos más adelante, cuando hablemos de REFACTORIZACIÓN, pero te animamos a que busques información sobre esta metodología y comentes en el foro cuáles son tus impresiones.

A Eric Eliott le cambió la vida:


- Web: <https://medium.com/javascript-scene/tdd-changed-my-life-5af0ce099f80>

4.2 Recursos sobre Ingeniería del Software

Existen numerosos recursos sobre ISW en la red de los que te recomendamos estos:

- **The Open University.** Curso express de Ingeniería del Software (en inglés británico) (6 horas)
 - Web: open.edu/openlearn/science-maths-technology/introduction-software-development/content-section-2
- **Platzi.** Aprende diseño, desarrollo web, marketing y negocios digitales.
 - Youtube: youtube.com/user/mejorandolaweb
- **Informe CHAOS 2015** decenas de variables en el informe más completo del sector de la ISW
 - PDF: standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf
- **Javier Garzás.** Mentor Ágil, Ágil Coach. Gestión de proyectos y equipos.
 - ¿Qué es AGILIDAD? youtube.com/watch?v=BWAmq_WhTkM
 - Web: javiergarzas.com Youtube: youtube.com/channel/UCSdjrn9u1AiXQdopOOQvI6kg

4.3 Actividad sobre metodologías ágiles

	<p><i>Por muy raro que te suene existe una técnica llamada Planning Poker, englobada dentro de las técnicas ágiles. ¿Podrías comentar en qué consiste?</i></p>
---	---

RESPUESTA: Aquí tienes toda la información y un vídeo muy interesante:

- <https://samuelcasanova.com/2016/01/estimacion-agil-con-la-tecnica-planning-poker/>

5. BIBLIOGRAFÍA

- i. Cuesta Vicente, Sergio, S. C. V. (2011, 17 octubre). *Tema 1 - Desarrollo de Software*. Apuntes de Sergio Cuesta Vicente. docs.google.com/viewer?a=vπd=sites&srcid=ZG9tZW5pY29zY2FybGF0dGkuZXN8c2N1ZXN0YXxneDoyZGQ2MjBjNDcwODIyYmFj
- ii. The Open University, (2019, 6 noviembre). *An introduction to software development*. The Open University - England. open.edu/openlearn/science-maths-technology/introduction-software-development/content-section-0?active-tab=description-tab
- iii. Javier Garzás, J. G. (consultado online 2020, 2 octubre). *Javier Garzás - Mentor Ágil, Ágil Coach. Gestión de proyectos y equipos*. Blog oficial de Javier Garzás. javiergarzas.com
- iv. *CHAOS REPORT*. (2015). The Standish Group International, Inc. standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf
- v. *CHAOS REPORT*. (1994). The Standish Group International, Inc. standishgroup.com/sample_research_files/chaos_report_1994.pdf
- vi. Iglesias, C. C. (2020). *Entornos de Desarrollo (GRADO SUPERIOR)*. RA-MA S.A. Editorial y Publicaciones.
- vii. Aldarias, F. (2012): *Apuntes de Entornos de Desarrollo*, CEEDCV