

UD3: Desarrollo de Software

1º DAW – ENTORNOS DE
DESARROLLO

IES Macia Abela

Índice

- Ingeniería del software
- Equipo de desarrollo
- Calidad del software
- Fases del desarrollo
- Contratos en desarrollo
- Metodologías de desarrollo
 - Clásicas: Cascada / V / Prototipos / Espiral
 - Ágiles: Scrum / Kanban / XP

Ingeniería del software

- Disciplina que estudia los principios y metodologías para el desarrollo y mantenimiento de sistemas software.
- Algunos autores consideran que "desarrollo de software" es un término más apropiado que "ingeniería de software"
 - IS implica niveles de rigor y prueba de procesos que no son apropiados para todo tipo de desarrollo de software.

Desarrollo de software



El software es único e intangible.

Desarrollo de software

- No es una ingeniería tradicional
 - En el software la parte de construcción es muy barata.
 - En el software todo el esfuerzo es diseño y requiere por tanto, de personas creativas y con talento.
 - Los procesos creativos no se se pueden planificar fácilmente, por lo que se trata de un proceso **no predecible**.
 - Solución: iteraciones cortas.

¿Los requisitos son predecibles?



Desarrollo de software

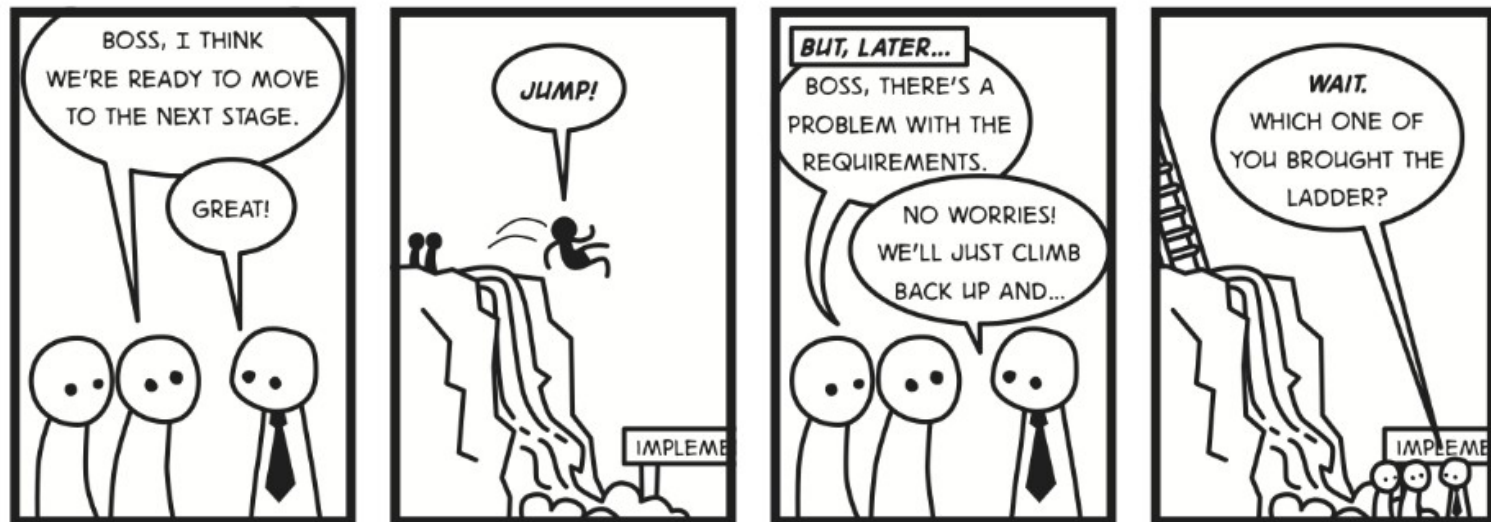
- Incertidumbres y riesgos que provocan que sea un proceso no predecible:
 - **Valor del producto:** ¿Se está construyendo el producto adecuado? ¿Las funcionalidades son las que realmente se necesitan? ¿Resolverán los usuarios sus problemas?
 - **Tecnología empleada:** ¿El equipo conoce bien la tecnología? ¿Va a poder escalar correctamente? ¿Tiene buena documentación?
 - **Incertidumbres sociales:** nuevos factores que afecten al negocio en el futuro y que modifiquen la forma de trabajar con ese software.

Desarrollo de software

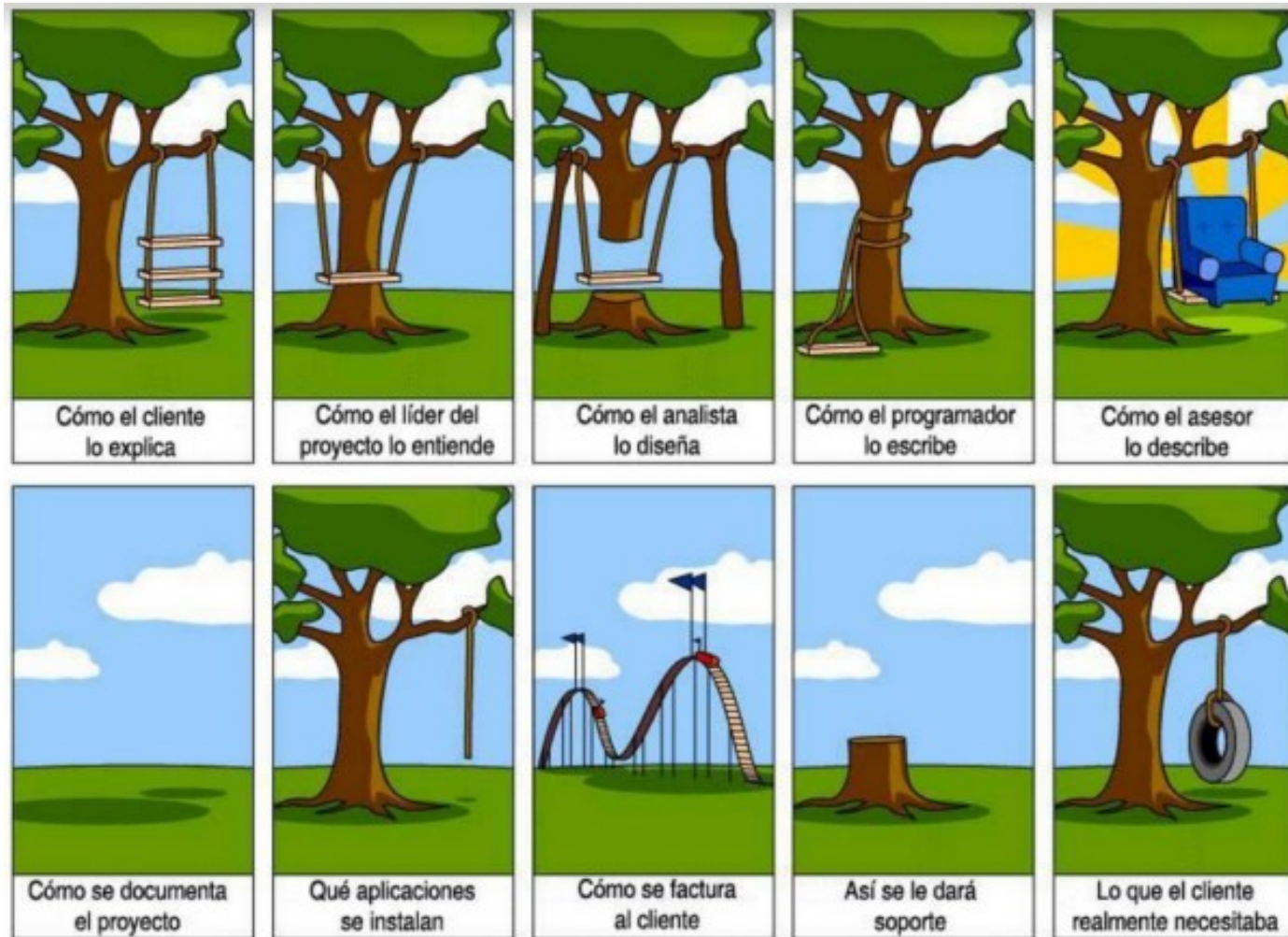
- Lo que nos gustaría...
 - Clientes que saben lo que quieren.
 - El equipo sabe cómo construirlo.
 - Nada cambiará en el camino.
 - Tenemos mucho tiempo y dinero para hacerlo.
- Lo que nos encontramos realmente...
 - Los clientes van descubriendo lo que necesitan.
 - Los desarrolladores descubren cómo hacerlo.
 - Muchas cosas cambian en el camino.
 - Siempre hay más cosas que hacer que tiempo y dinero disponible.

Desarrollo de software

- Solucionar problemas
- ¡Factor humano!
 - Entender el problema a resolver
 - Gestionar la comunicación y colaboración
 - Empatía



Comunicación



Equipo de desarrollo

- Una persona puede adoptar más de un rol, puede ir adoptando distintos roles con el paso del tiempo, o rotar de rol a lo largo del día.
- **Dueño del producto:** Pide lo que necesita (no el cómo sino el qué), y acepta/solicita correcciones sobre lo que se entrega.
- **Cliente:** Es el dueño del producto y el usuario final.
- **Analista de negocio:** Dueño del producto dentro del equipo.
 - Enlace con el cliente
 - Traduce los requisitos en tests de aceptación.
 - Explica a los desarrolladores qué hay que hacer, resolviendo las dudas.

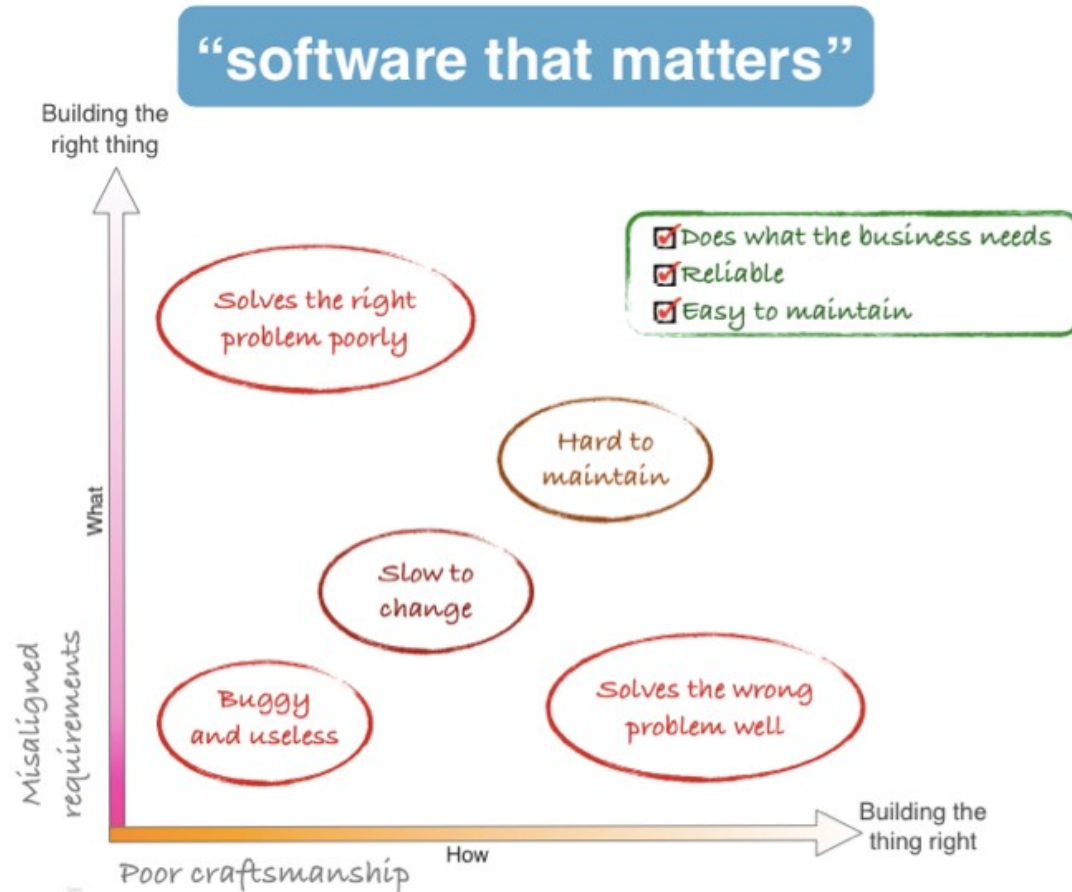
Equipo de desarrollo



- **Desarrolladores:** Toman la información del analista de negocio y deciden cómo lo van a resolver.
 - Aparte de escribir código, los desarrolladores deben tener conocimientos avanzados sobre usabilidad y diseño de interfaces de usuario.
- **Arquitectos:** Persona que adopta varios roles capaz de tomar decisiones de diseño.
 - Capacidad de poder hablar directamente con el cliente y entender los requisitos de negocio.
 - En el desarrollo ágil los desarrolladores hacen la función de arquitectos.
- **Administradores de sistemas:** Encargados de los servidores y servicios que necesitan los desarrolladores.

Qué vs Cómo

“Construir el
producto
correcto”



“Construir
correctamente
el producto”

Calidad en el software

- Calidad de producto vs calidad de desarrollo.
- **El producto no funciona** cuando:
 - Sólo cumple parcialmente lo que necesita el usuario
 - Faltan características y/o tiene errores.
- **El software está mal desarrollado** cuando:
 - Es lento, poco eficiente, poco robusto, tiene bugs.
 - Su código no es mantenible, es poco entendible y con deuda técnica (problemas de diseño que se van introduciendo en el software provocando que sea cada vez menos flexible y más rígido).

Desarrollo software - Cuello de botella



- **Complejidad**

Gran cantidad de entidades distintas que van creciendo conforme el proyecto crece, aumentando a su vez el número de estados de sus elementos.

- **Conformidad**

Debe adaptarse a interfaces ya existentes y no se puede simplificar o rediseñar por si solo.

- **Cambiabilidad**

El cambio es un constante con lo que tiene que coexistir el desarrollo.

- **Invisibilidad**

Es invisible e invisualizable, al no poder visualizarlo es muy difícil razonar sobre su estructura y analizarlo entre varias personas.

Triángulo de hierro

- Todo proyecto tiene 3 variables relacionadas:
 - Alcance: Características, funcionalidades.
 - Coste: Dinero, personas, etc..
 - Tiempo: Cuanto durará el proyecto.



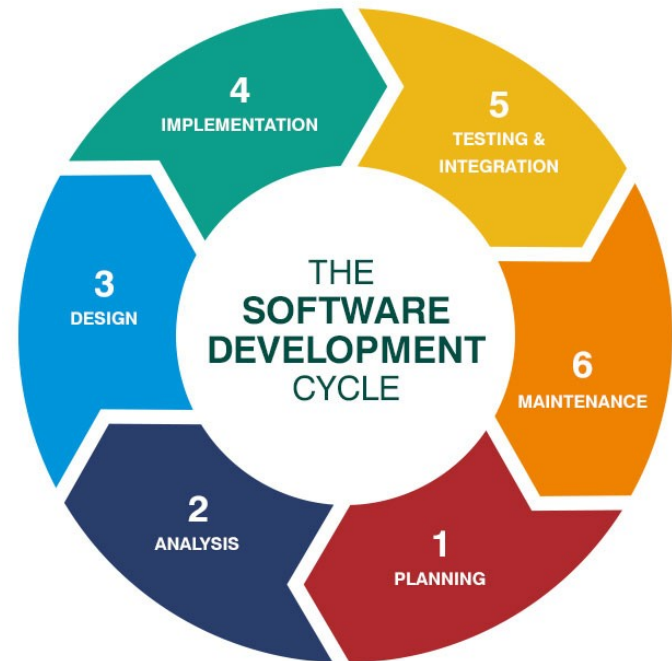
Al priorizar una variable, se deterioran el resto o se reduce la calidad del producto.

Las 4 dimensiones

- Personas
 - Composición, motivación, capacidad y formación
- Producto
 - Tamaño
 - Características
 - Funcionalidades
- Procesos
 - Metodología
 - QA
 - Gestión del riesgo
- Tecnología
 - Herramientas
 - Lenguajes
 - Frameworks

Fases del desarrollo

- 1) Planificación
- 2) Análisis
- 3) Diseño
- 4) Implementación
- 5) Pruebas e Integración
- 6) Mantenimiento



1. Planificación

- Determinación del ámbito del proyecto.
- Estudio de viabilidad del proyecto.
- Análisis de riesgos asociados.
- Estimación del coste del proyecto.
- Planificación temporal del proyecto.
- Asignación de recursos distintas etapas.

2.- Análisis



- Determina y define las necesidades del cliente
- Análisis de requisitos
 - Sistema
 - Software
- La especificación de requisitos debe:
 - Ser completa, concisa y sin ambigüedades.
 - Evitar detalles de diseño o implementación.
 - Ser entendible por el cliente.
 - Separar requisitos funcionales y no funcionales.
 - Fijar criterios de validación.

3.- Diseño



- Se descompone y organiza el sistema en componentes que pueden ser desarrollados por separado.
 - Se especifica su interrelación y funcionalidad.
- Resultados:
 - Diseño de arquitectura.
 - Diseño detallado de clases y colaboración entre objetos.
 - Diseño de datos.
 - Diseño de interfaz de usuario (prototipos).



4.- Implementación

- Se escribe el código fuente de cada componente.
- Pueden utilizarse distintos lenguajes informáticos:
 - Lenguajes de programación: C, C++, Java, Javascript, ...
 - Acceso a datos: SQL, JSON
 - Lenguajes de otro tipo: HTML, XML, ...
- Control de versiones y copias de seguridad.

5.- Pruebas e Integración

- Objetivo: conseguir que el programa funcione incorrectamente y que se descubran defectos.
- Deberemos someter al software al máximo número de situaciones diferentes.
- Pruebas automatizadas
 - Pruebas unitarias: internas.
 - Pruebas de integración.
 - Pruebas funcionales.
 - Pruebas de sistema.
- Integración continua de los diferentes componentes que conforman el sistema.



6.- Mantenimiento

- Durante la explotación del sistema software es necesario realizar cambios.
 - Rehacer parte del trabajo realizado en las fases previas.
- Tipos de mantenimiento:
 - Correctivo: se corrigen defectos.
 - Perfectivo: se mejoran funcionalidades existentes.
 - Evolutivo: se añade funcionalidades nuevas.
 - Adaptativo: se adapta a nuevos entornos.

Contratos en desarrollo software



- **Contrato tradicional:**

Precio fijo, el cliente paga al final por la entrega de unos requisitos en un plazo acordado. Si se incumple el plazo o los requisitos, el cliente puede no pagar y la empresa de software no entregar nada.

- **Contrato ágil:**

Presupuesto fijo, se paga a intervalos establecidos y la empresa de software entrega periódicamente el producto. Cualquiera de los dos puede cancelar el contrato en cualquier momento.

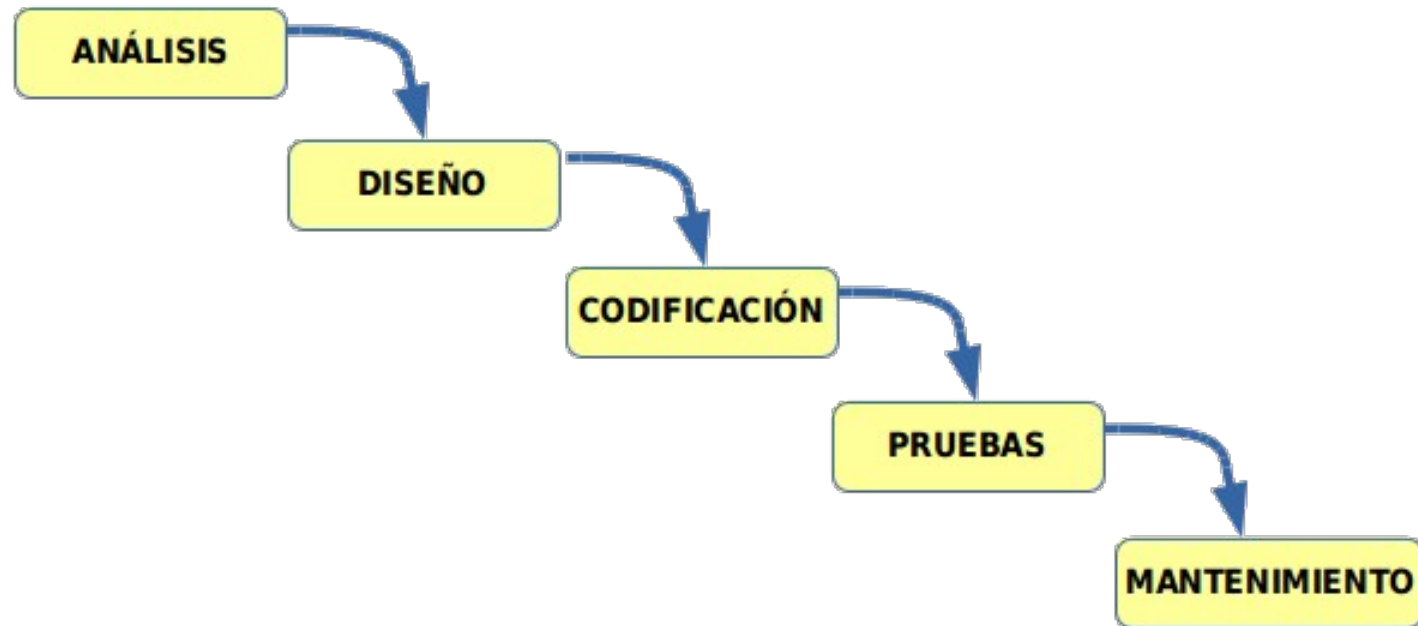
Metodologías de desarrollo

- Pasos a seguir
- Modelo de ciclo de vida del software
 - Valores
 - Principios
 - Buenas prácticas
- Roles y responsabilidades de los distintos actores que intervienen en el proceso

Modelos de Desarrollo

- Modelos clásicos (predictivos)
 - Modelo en cascada
 - Modelo en V
- Modelo de construcción de prototipos
- Modelos evolutivos o incrementales
 - Modelo en espiral (iterativos)
 - Metodologías ágiles (adaptativos)

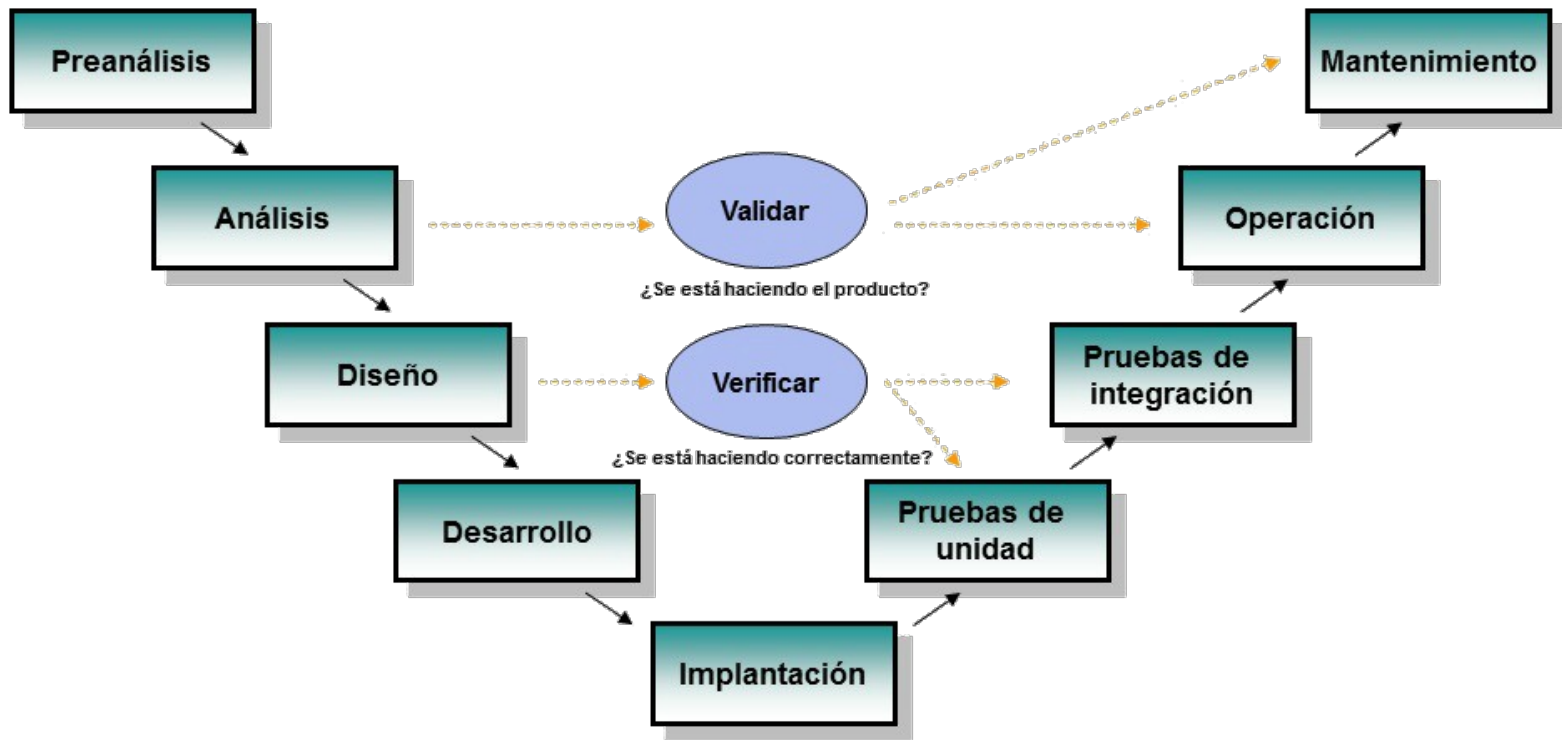
Modelo en cascada



Modelo en cascada

- Modelo de mayor antigüedad.
- Identifica las fases principales del desarrollo software.
 - Las fases han de realizarse en el orden indicado.
 - El resultado de una fase es la entrada de la siguiente fase.
- Rígido: se adapta mal al cambio continuo de especificaciones.
- No hay una versión operativa hasta el final.
 - Rectificar cualquier decisión errónea tomada en las fases iniciales supondrá un coste significativo (tiempo y dinero)
- Existen diferentes variantes con mayor o menor cantidad de actividades.

Modelo en V

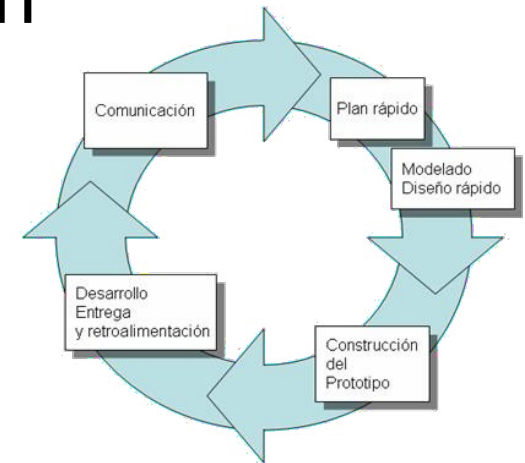


Modelo en V

- Similar al modelo en cascada.
- Visión jerarquizada con distintos niveles.
 - Los niveles superiores indican mayor abstracción.
 - Los niveles inferiores indican mayor nivel de detalle.
- El resultado de una fase es la entrada de la siguiente fase.
- Existen diferentes variantes con mayor o menor cantidad de actividades.
- Tampoco tolera bien los cambios. Rectificar supone un coste importante.

Modelo construcción de prototipos

- A menudo los requisitos no están especificados claramente:
 - por no existir experiencia previa.
 - por omisión o falta de concreción del usuario/cliente.
- Proceso:
 - Se crea un prototipo durante la fase de análisis y el usuario/cliente lo prueba para refinar los requisitos del software a desarrollar.
 - Se repite el paso anterior las veces necesarias.



Modelo construcción de prototipos



- Ventajas:
 - Facilidad la comunicación entre el analista y los usuarios finales (cliente).
 - Los sistemas se desarrollan más rápidamente.
 - Implantación del sistema más sencilla (los usuarios ya lo conocen).
- Inconvenientes:
 - Puede crear falsas expectativas en el usuario final al ver el prototipo.
 - Fuerte intromisión de los usuarios en la integración.
 - Inconsistencias entre el prototipo y el producto final.

Tipos de Prototipos

- **Prototipos rápidos**

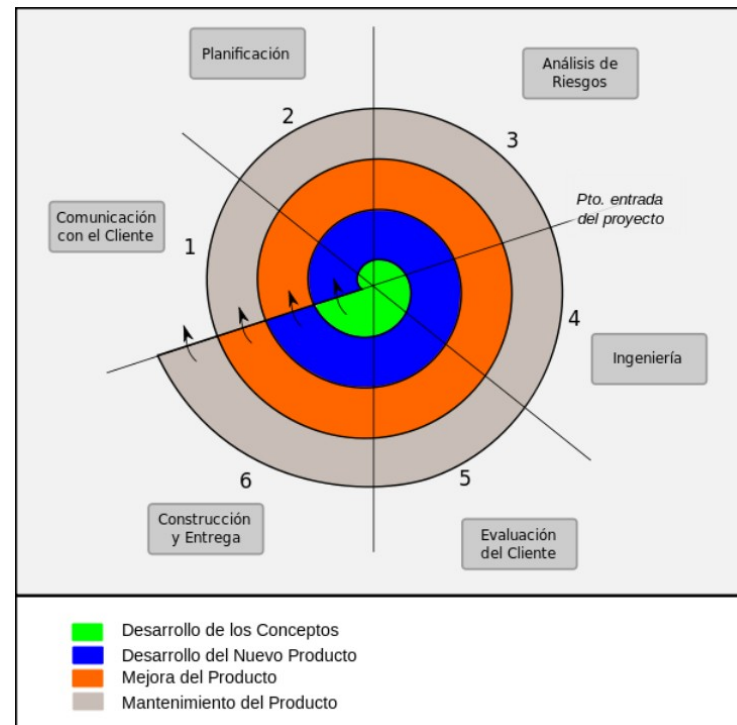
- El prototipo puede estar desarrollado usando otro lenguaje y/o herramientas.
- Se busca la rapidez y facilidad con el objetivo de reducir riesgos de construir un producto alejado de las necesidades del cliente.
- Finalmente el prototipo se desecha.

- **Prototipos evolutivos**

- El prototipo está diseñado en el mismo lenguaje y herramientas del proyecto.
- El prototipo se usa como base para desarrollar el proyecto.
- Aumenta las posibilidades de éxito del proyecto.

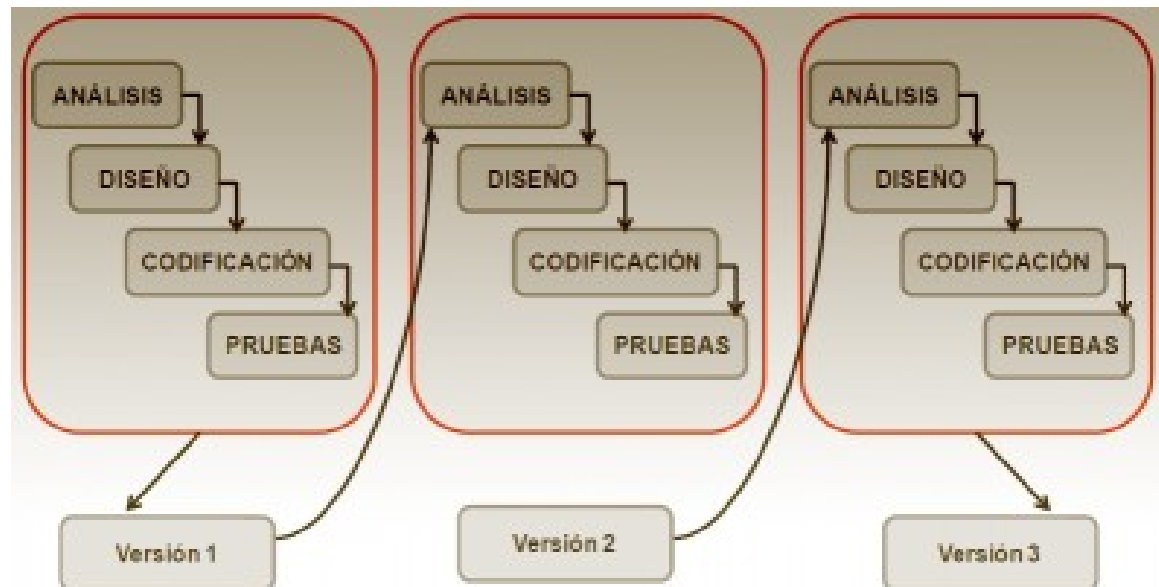
Modelo en espiral

- Desarrollado por Boehm en 1988.
- La actividad de ingeniería corresponde a las fases de los modelos clásicos: planificación, análisis, diseño, implementación, evaluación...



Desarrollo iterativo / incremental

- Cada iteración se centra en el desarrollo de una pequeña parte → incremento
 - En cada iteración se vuelven a realizar todas las fases para desarrollar una parte mayor.



Modelo en espiral

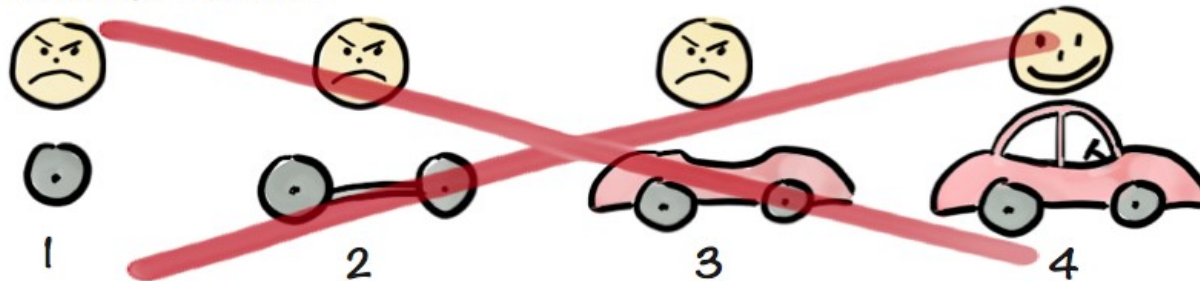
- Ventajas:
 - Orientado al riesgo, si se aplica adecuadamente elimina muchas de las posibles dificultades.
 - Modelo flexible y genérico.
 - Apropiado para entornos técnicos novedosos.
 - Máximo control sobre costes, recursos y calidad.
- Inconvenientes:
 - Requiere experiencia en la identificación de riesgos.
 - No se aconseja utilizarlo en pequeños sistemas.
 - Las decisiones periódicas pueden dilatar el proceso de desarrollo.
 - Modelo costoso, con gran esfuerzo de gestión.

Metodologías ágiles

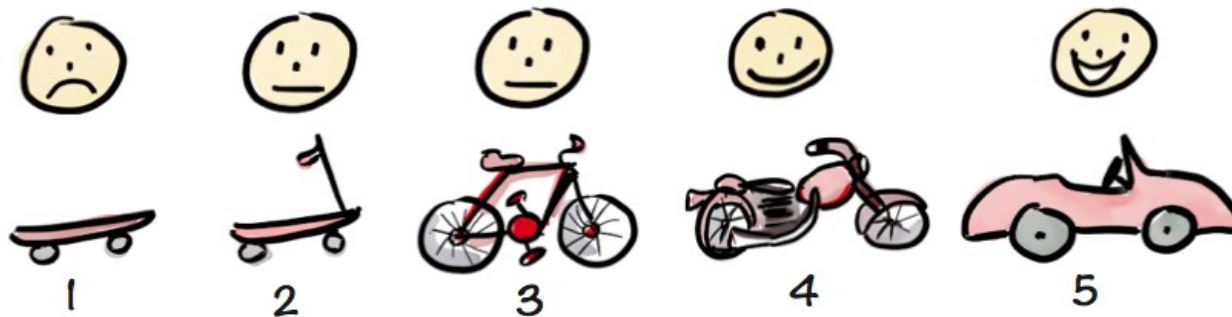
- Promueven un desarrollo evolutivo mediante una planificación dinámica, realizando entregas incrementales.
- Incluyen prácticas que incentivan la agilidad (respuesta rápida y flexible al cambio).
- Los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto.
- El trabajo es realizado mediante la colaboración de equipos auto-organizados y multidisciplinarios, inmersos en un proceso cooperativo de toma de decisiones a corto plazo, fomentado la comunicación y cooperación.

Metodologías ágiles

Not like this....



Like this!



Henrik Kniberg

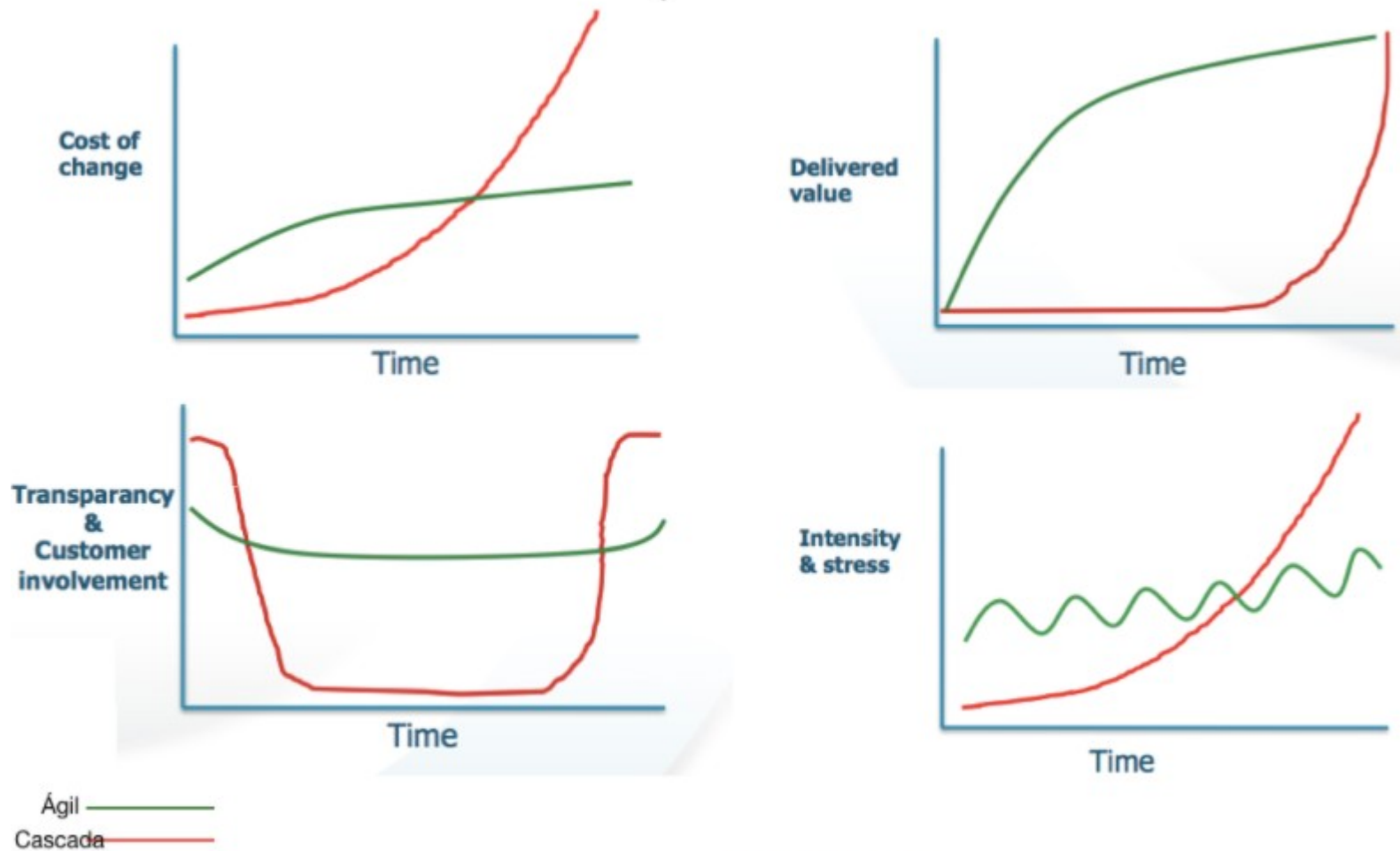
Manifiesto ágil (2001)



Principios manifiesto ágil



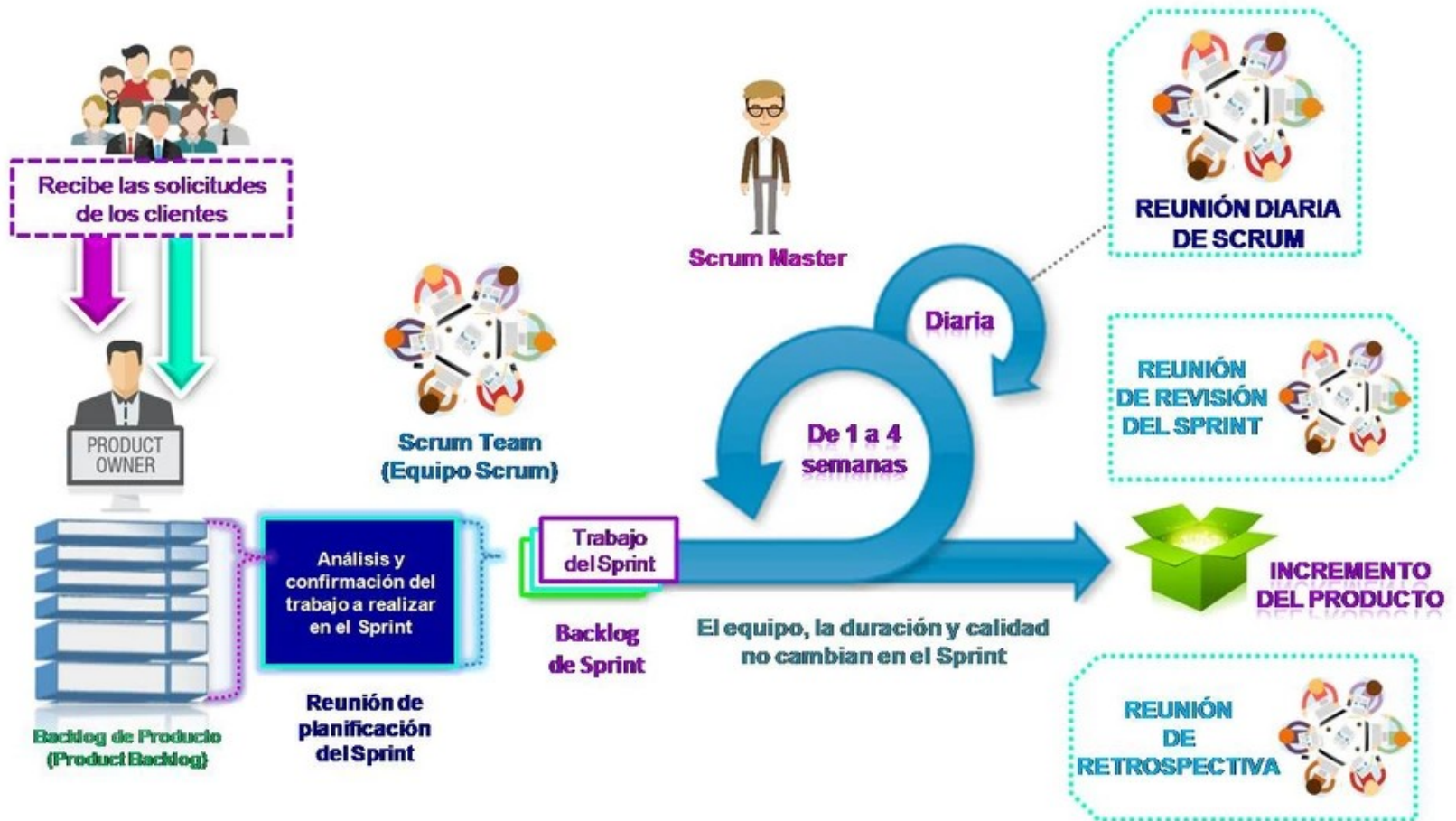
Modelos Ágiles vs Cascada



Scrum

- Framework de desarrollo incremental.
- Iteraciones (*Sprint*) regulares cada 2 a 4 semanas.
- Al principio de cada iteración se establecen sus objetivos priorizados (*Sprint Backlog*).
- Al finalizar cada iteración se obtiene una entrega parcial utilizable por el cliente → incremento
- Existen reuniones diarias para tratar la marcha del sprint.

Scrum



Roles

- Cliente (*Product Owner*)
 - Todas las personas interesadas en el proyecto (usuarios finales, promotores, etc).
 - Plantea los objetivos y requisitos: a nivel global y de Sprint.
- Facilitador (*Scrum Master*)
 - Asegura de que se sigan los valores y principios ágiles
 - Facilita las reuniones, quita los impedimentos que se van encontrando en el camino y protege/aisla al equipo de interrupciones.
- Equipo (*Team*)
 - Autoorganizados, objetivo común, comparten la responsabilidad y sus confían entre ellos.
 - Tamaño \pm 5-9 personas. Si hay más, se crean más equipos.

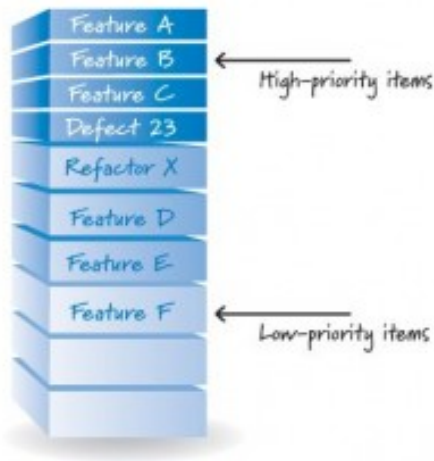
Fases I

1) Reunión inicial

- Cliente + Scrum Master → **Product Backlog**

2) Planificación de la iteración (*Sprint planning*)

- A partir de requisitos prioritarios, se traducen a tareas y se define el **Sprint Backlog**.



Product backlog

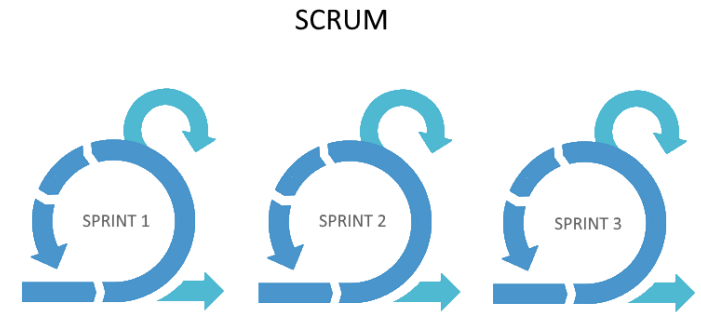
ID	Status	Task	Task Type	Priority Lvl.	Est. Time (hours)	Actual Time (hours)	Owner of Task
1	Complete	Finish project proposal and decide on a topic	Planning	Very High	1	1.5	Jennifer
2	Complete	Set up MySQL and PhpMyAdmin	Content	Very High	0.5	1	Jennifer
3	Complete	Create online portfolio using Wordpress	Content	Very High	0.5	0.5	Jennifer
4	Complete	Work on website wireframe and layout	Design	Very High	2	0.5	Jennifer
5	Complete	Code HTML/CSS for the pages	Feature	Very High	4	3	Jennifer
6	Complete	Decide what pages must be created for the website	Planning	High	1	0.5	Jennifer
7	Complete	Create and begin formatting the other pages	Feature	High	8	3	Jennifer
8	Complete	Code some sort of navigation so users can navigate through the different pages	Feature	High	2	1	Jennifer
9	In Progress	Do research on my topic of clean drinking water	Content	High	10	TBA	Jennifer
10	In Progress	Add information that I've gathered in my research to my website	Content	Medium	2	TBA	Jennifer
11	In Progress	Integrate a 3rd party service such as Google Maps	Feature	Medium	1.5	TBA	Jennifer
12	In Progress	Find / take own pictures to use	Design	Low	1	TBA	Jennifer
13	In Progress	Find / create video(s) to incorporate	Feature	Low	1	TBA	Jennifer
14	In Progress	Create graphics and visuals using Photoshop	Design	Low	1.5	TBA	Jennifer
15	In Progress	Look into incorporating audio to the website	Feature	Low	1	TBA	Jennifer

Sprint backlog

Fases II

3) Ejecución de la iteración (*Sprint*)

- 2-4 semanas. Siempre la misma duración



4) Reunión diaria del equipo (*Daily Scrum Meeting*)

- 10 minutos, de pie y en círculo
- Revisión de las tareas: actualización del Sprint backlog



Fases III

5) Revisión de la entrega (*Sprint Review*)

- Al final del Sprint. Se muestra al cliente el incremento.

6) Retrospectiva (*Sprint Retrospective*)

- Se comprueban, miden y evalúan los resultados.
- Mejora continua del proceso



Kanban

- También se denomina "sistema de tarjetas".
- Desarrollado inicialmente por Toyota para la industria de fabricación de productos.
 - Controla por demanda la fabricación de los productos necesarios en la cantidad y tiempo necesarios.
 - Enfocado a entregar el máximo valor para los clientes, utilizando los recursos justos.
- Lean manufacturing
 - Organización del trabajo centrada en la continua mejora y optimización del sistema de producción, mediante la eliminación de actividades que no suman ningún valor al proceso.

Motivaciones Kanban

- Conseguir un ritmo de trabajo sostenible en el desarrollo de software.
- Visualizar en todo momento la carga de trabajo del equipo de desarrollo (WIP: Work in progress).
- Visualizar y estandarizar el flujo de trabajo de las historias de usuario.
- Políticas explícitas.

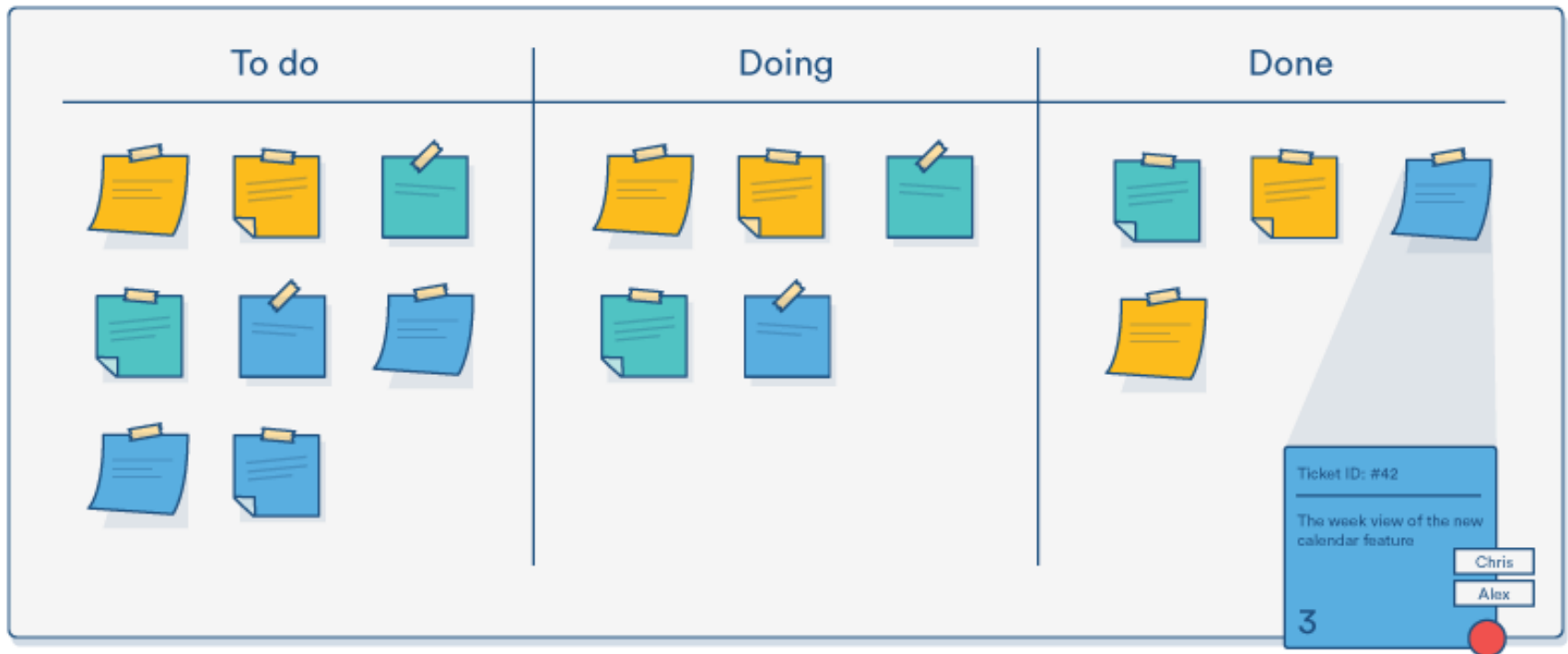
Principios Kanban

- Facilitan la gestión del cambio y mejoran la prestación de servicios.

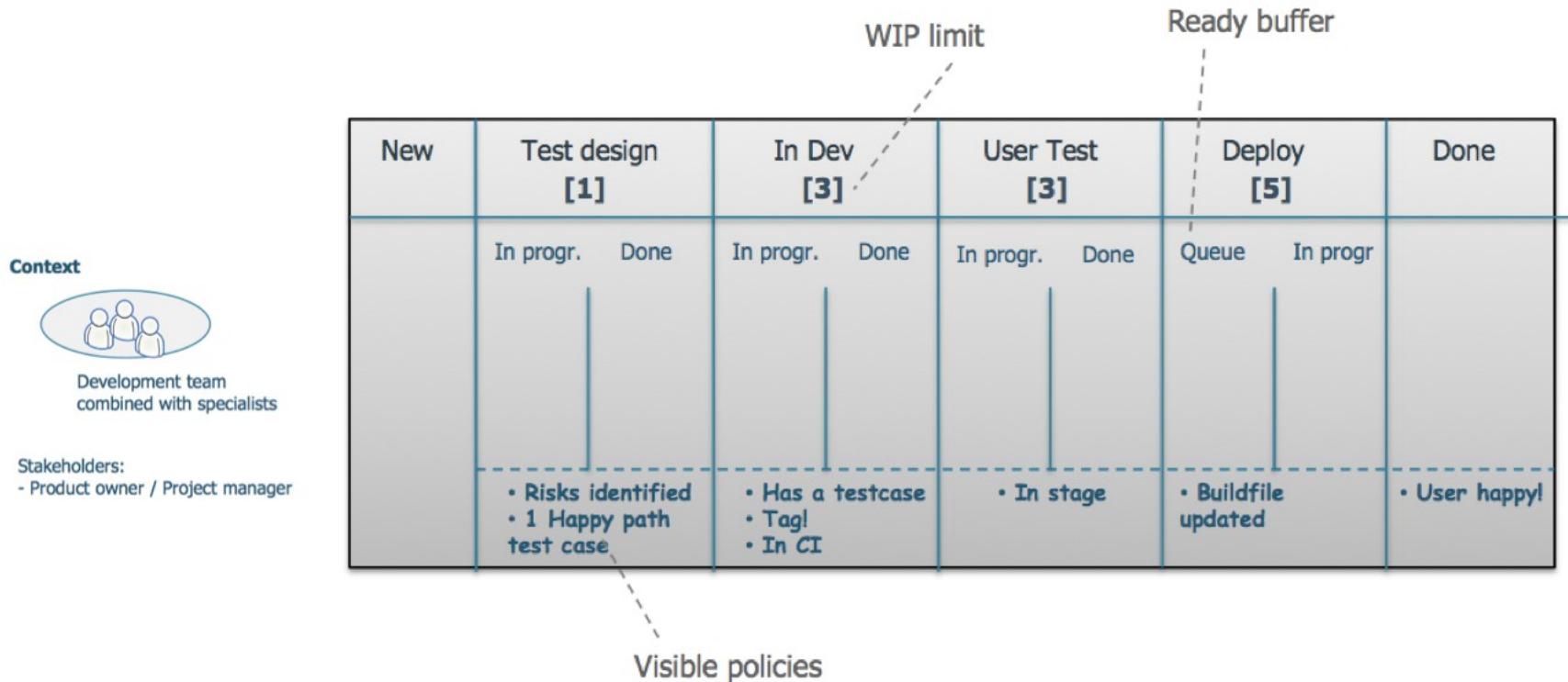
	GESTIÓN DEL CAMBIO	ENTREGA DE SERVICIOS
PRINCIPIOS DEL MÉTODO KANBAN	<ol style="list-style-type: none">1. Empezar con lo que se hace ahora.2. Acordar la ejecución de la mejora a través de un cambio evolutivo.3. Fomentar actos de liderazgo a todos los niveles.	<ol style="list-style-type: none">1. Entender y enfocarse a las necesidades y expectativas de los clientes.2. Gestiona el trabajo y deja a las personas autoorganizarse a su alrededor.3. Evolucionar las políticas.

<https://itnove.com/blog/kanban/equipos/los-principios-de-kanban/>


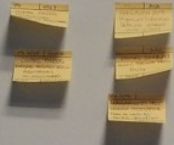
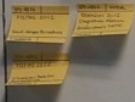



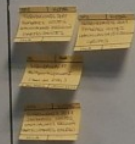



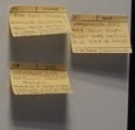
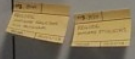


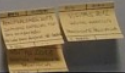
Un sistema Kanban sencillo



Un sistema Kanban más complejo

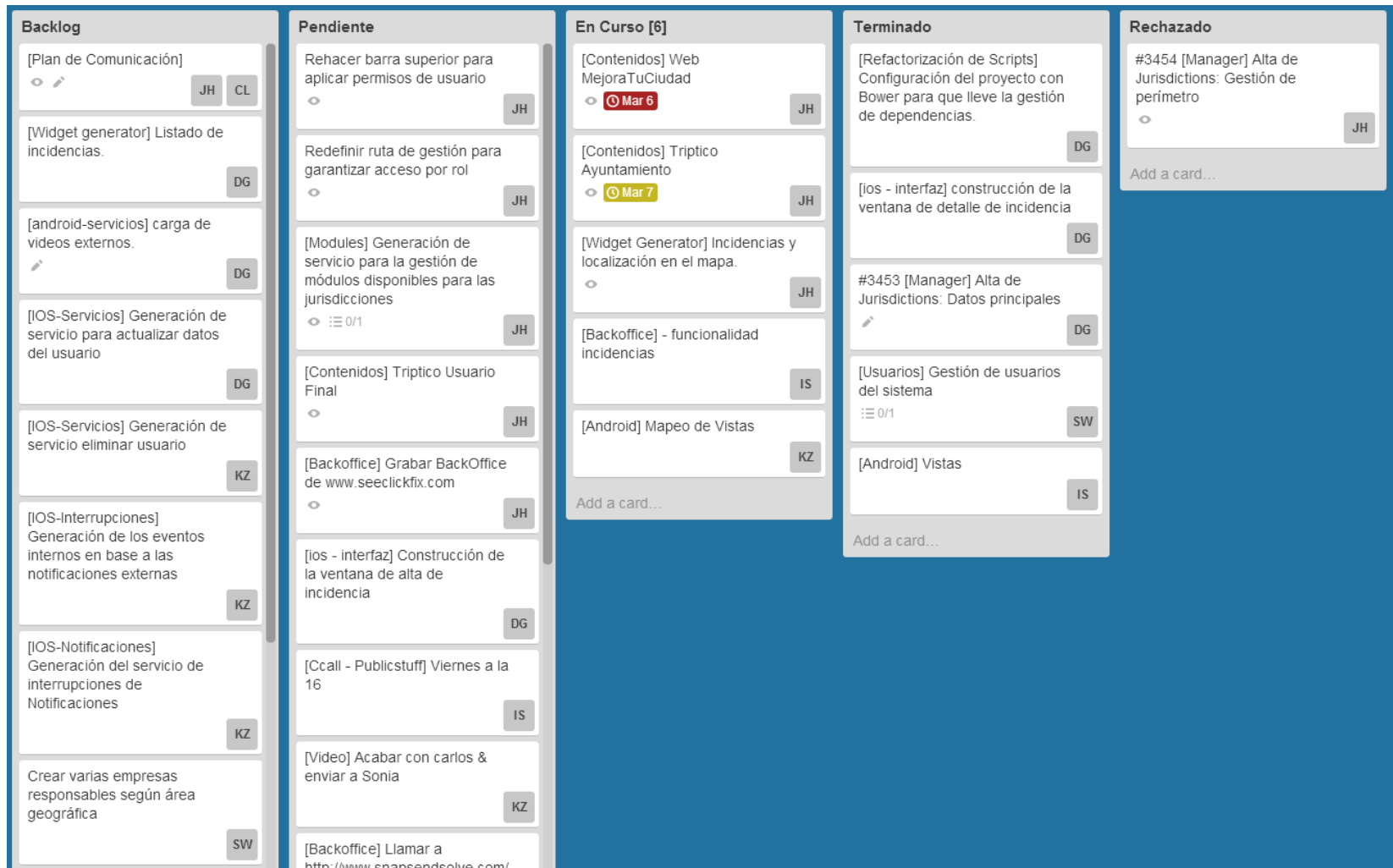


Ejemplos reales

PILA DE TAREAS		DESARROLLO		PRUEBAS		PRE/PRO
		EN CURSO (6)	HECHO (30)	EN CURSO (30)	HECHO	
BONIF.						
SUBV 1						
SUBV 2						
TRANS						
M.P.						

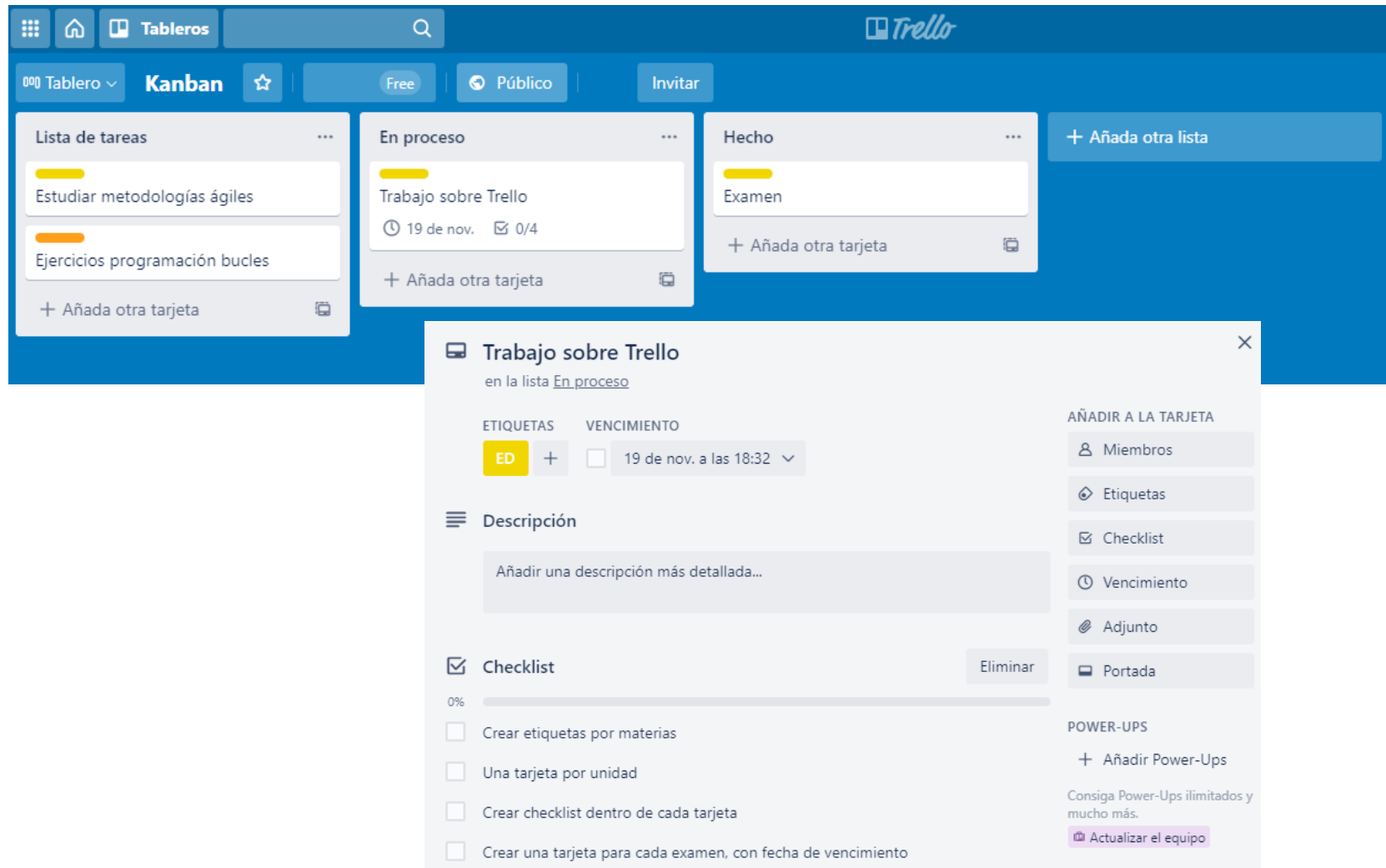
<https://www.javiergarzas.com/2014/03/tableros.html>

Ejemplos reales



Backlog	Pendiente	En Curso [6]	Terminado	Rechazado
[Plan de Comunicación] JH CL	Rehacer barra superior para aplicar permisos de usuario JH	[Contenidos] Web MejoraTuCiudad Mar 6 JH	[Refactorización de Scripts] Configuración del proyecto con Bower para que lleve la gestión de dependencias. DG	#3454 [Manager] Alta de Jurisdiccions: Gestión de perímetro JH
[Widget generator] Listado de incidencias. DG	Redefinir ruta de gestión para garantizar acceso por rol JH	[Contenidos] Triptico Ayuntamiento Mar 7 JH	[ios - interfaz] construcción de la ventana de detalle de incidencia DG	Add a card...
[android-servicios] carga de videos externos. DG	[Modules] Generación de servicio para la gestión de módulos disponibles para las jurisdicciones 0/1 JH	[Widget Generator] Incidencias y localización en el mapa. JH	#3453 [Manager] Alta de Jurisdiccions: Datos principales DG	
[IOS-Servicios] Generación de servicio para actualizar datos del usuario DG	[Contenidos] Triptico Usuario Final JH	[Backoffice] - funcionalidad incidencias IS	[Usuarios] Gestión de usuarios del sistema 0/1 SW	
[IOS-Servicios] Generación de servicio eliminar usuario KZ	[Backoffice] Grabar BackOffice de www.seeclickfix.com JH	[Android] Mapeo de Vistas KZ	[Android] Vistas IS	
[IOS-Interrupciones] Generación de los eventos internos en base a las notificaciones externas KZ	[ios - interfaz] Construcción de la ventana de alta de incidencia DG	Add a card...	Add a card...	
[IOS-Notificaciones] Generación del servicio de interrupciones de Notificaciones KZ	[Ccall - Publicstuff] Viernes a la 16 IS			
Crear varias empresas responsables según área geográfica SW	[Video] Acabar con carlos & enviar a Sonia KZ			
	[Backoffice] Llamar a http://www.snapsendsolve.com/			

Trello

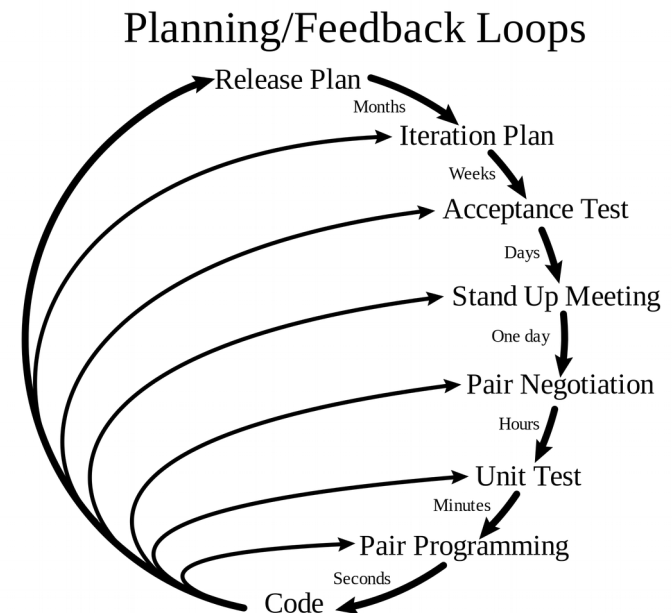


The screenshot displays the Trello Kanban board interface. The top navigation bar includes icons for a grid, home, and boards, with the text 'Tableros' and a search icon. The board is titled 'Kanban' and has a 'Free' label. The board is divided into three columns: 'Lista de tareas', 'En proceso', and 'Hecho'. The 'Lista de tareas' column contains two cards: 'Estudiar metodologías ágiles' and 'Ejercicios programación bucles'. The 'En proceso' column contains one card: 'Trabajo sobre Trello'. The 'Hecho' column contains one card: 'Examen'. A modal window is open for the card 'Trabajo sobre Trello', showing its details. The modal includes a title 'Trabajo sobre Trello', a description 'en la lista En proceso', a label 'ED', a due date '19 de nov. a las 18:32', and a checklist with four items: 'Crear etiquetas por materias', 'Una tarjeta por unidad', 'Crear checklist dentro de cada tarjeta', and 'Crear una tarjeta para cada examen, con fecha de vencimiento'. The modal also has a 'Checklist' section with a progress bar and a 'POWER-UPS' section with a button 'Actualizar el equipo'.

<https://trello.com/b/ehVSmDWY/kanban-template-plantilla-para-tablero-kanban>

Programación eXtrema

- *Kent Beck*
- Define unos valores y un conjunto de prácticas llevadas al extremo.
- Adaptación continua al cambio
- Plantea entregas continuas en ciclos cortos de desarrollo
- Las pruebas son el elemento fundamental



Valores de XP

- Eficaz entre los ingenieros de software y otros participantes, XP pone el énfasis en la colaboración estrecha pero informal (verbal) entre los clientes y los desarrolladores.

Comunicación



- XP restringe a los desarrolladores para que diseñen sólo para las necesidades inmediatas, en lugar de considerar las del futuro.

Simplicidad



- Se obtiene de tres fuentes: el software implementado, el cliente y otros miembros del equipo de software.

Retroalimentación



- Un término más apropiado sería disciplina. Por ejemplo, es frecuente que haya mucha presión para diseñar requerimientos futuros.

Valentía

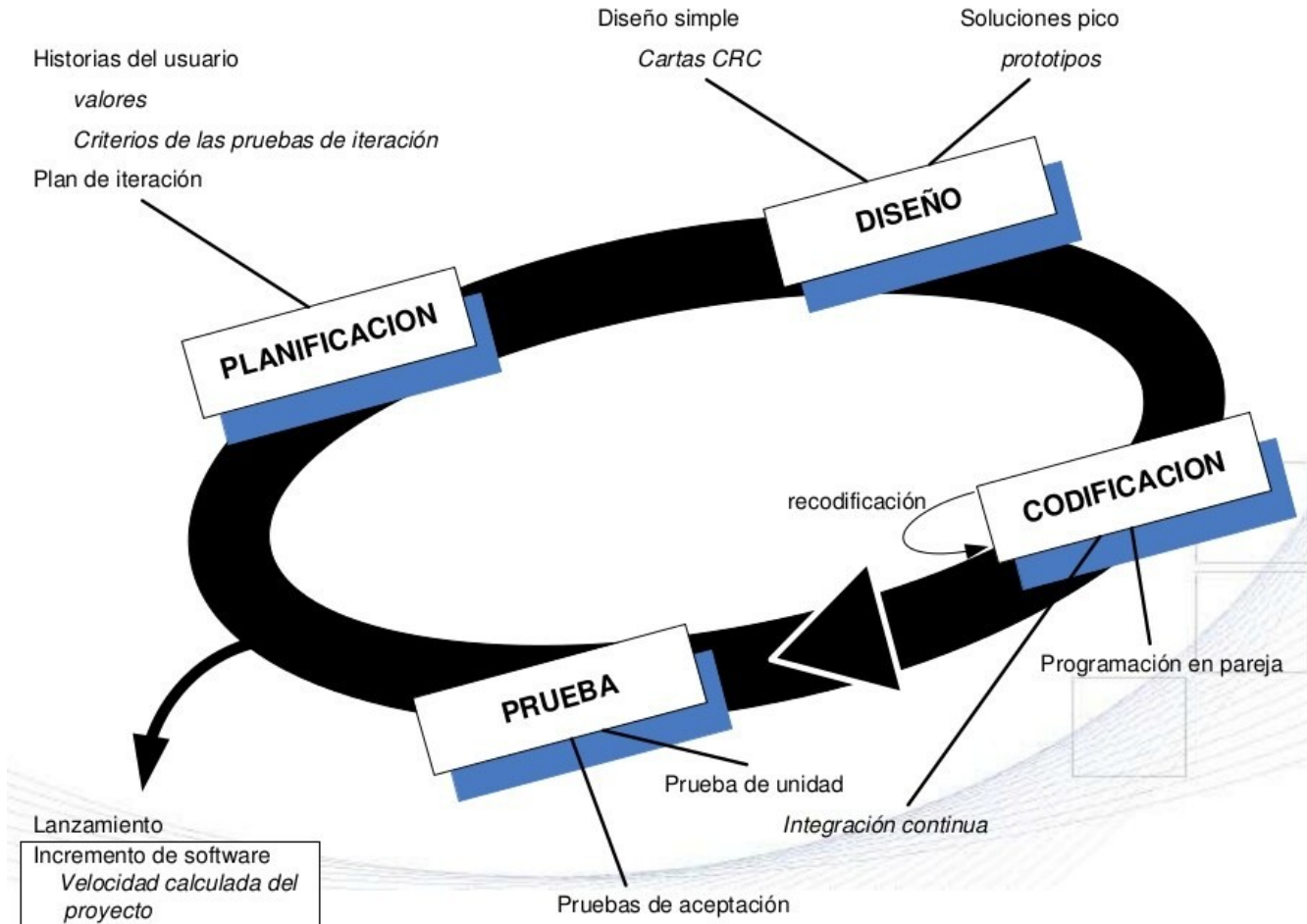


- Entre sus miembros, entre otros participantes y los integrantes del equipo, e indirectamente para el software en sí mismo.

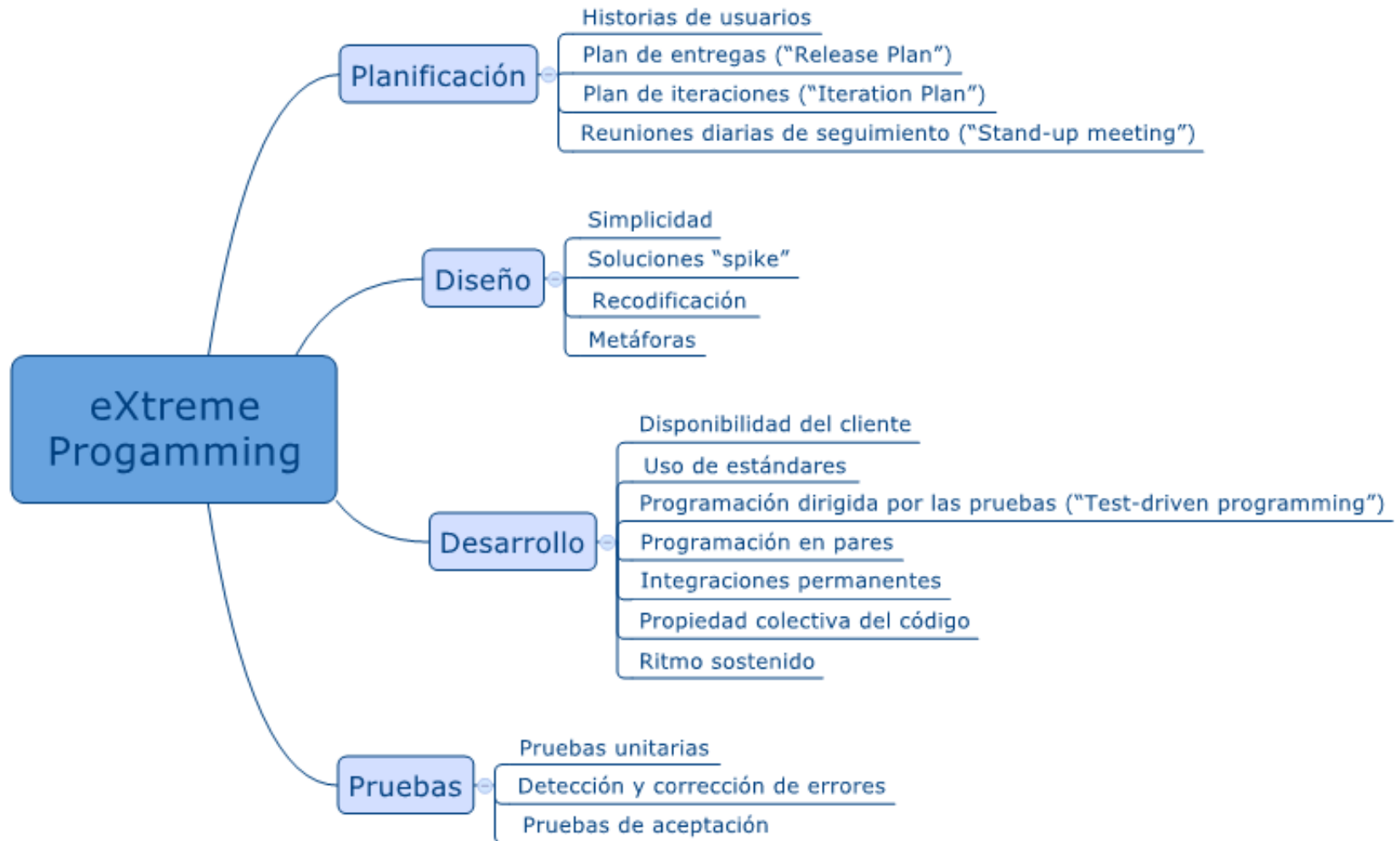
Respeto



Proceso XP

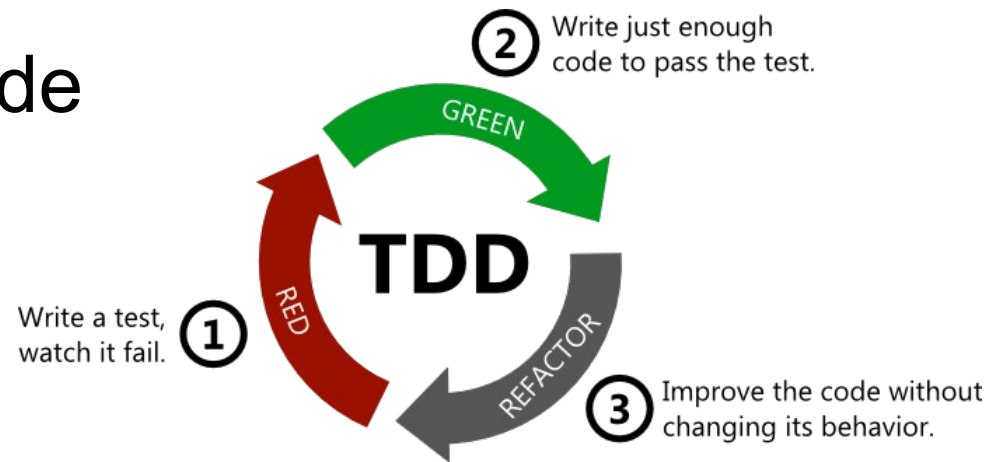


Prácticas XP



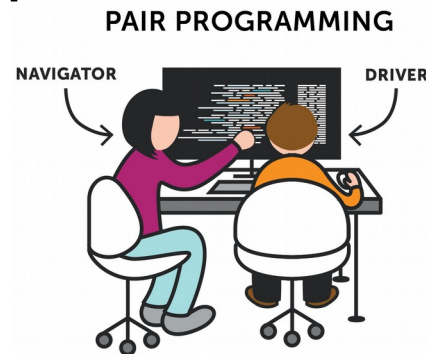
Test Driven Development

- Primero se escriben las pruebas
- Las pruebas sirven de especificación
- Dan seguridad al desarrollador
- *Make it green*
- *Test > Code > Refactor*



Pair Programming

- Programación en parejas
- 2 roles:
 - **controlador/codificador**
 - **observador**
- Resultado:
 - Calidad del código
 - Reducción del coste: menos errores
 - Aprendizaje y formación cruzada



¿Preguntas?

