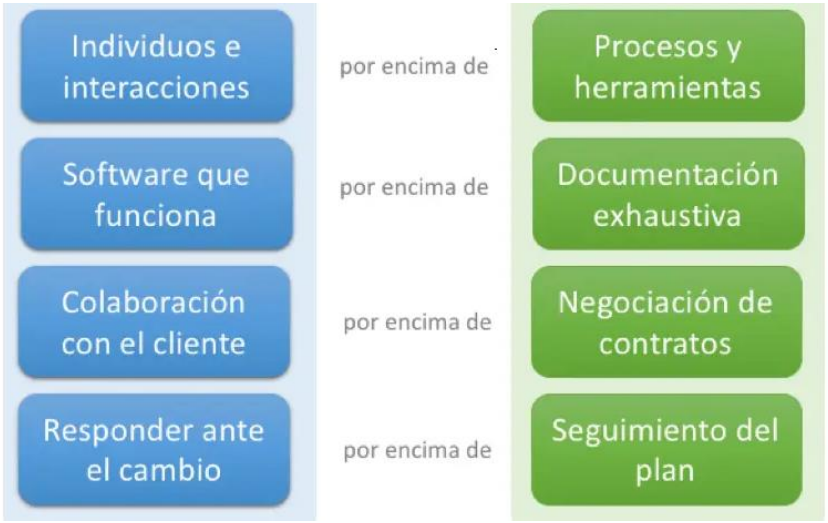
TEMA 37. METODOLOGÍAS ÁGILES PARA LA GESTIÓN DE PROYECTOS. METODOLOGÍAS LEAN

Actualizado a 05/05/2023

1. MANIFIESTO ÁGIL

La filosofía ágil tiene su origen en el **Manifiesto Ágil** (año 2001) y propone unos Valores y unos Principios.

VALORES:



Los 12 PRINCIPIOS

El segundo principio ("*se aceptan los cambios*") es uno de las grandes diferencias con respecto al modelo **Waterfall** que es una **estructura de trabajo muy rígida** en cuanto a cambios se refiere, donde h**ay poca tolerancia a la incertidumbre y puede ser perjudicial en proyectos de larga duración**, en los que la tecnología o las necesidades del cliente pueden variar.

La **metodología Agile o Ágil,** es un enfoque de **trabajo basado siempre en la satisfacción del cliente**. Busca **repartir el trabajo y las diferentes tareas del proyecto de forma flexible**, para poder trabajar rápidamente sin la rigidez de las secuencias establecidas en la Waterfall. Los cambios se aceptan como algo bueno para el producto final, así que las peticiones de cambio no sufren esa burocracia de Waterfall.

2. AGILIDAD Y LEAN

Lean es un término que se popularizó cuando se pretendía aligerar los procesos, principalmente en la industria manufacturera. Toyota fue su principal impulsor. Su objetivo es eliminar actividades que no aportan valor, para así poder obtener un producto o servicio de mayor calidad y que mejore la experiencia de los clientes.

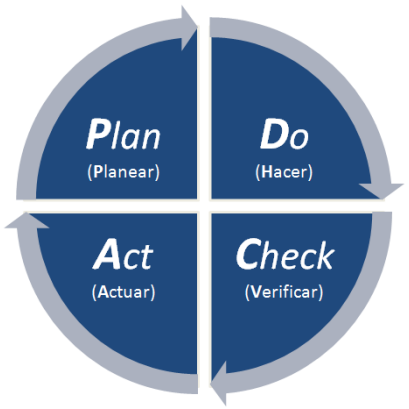
Podemos decir que Lean es el paraguas bajo el que se instala *Agile*, con parte de sus aprendizajes y prácticas integradas en su ADN.

3. AGILIDAD Y CICLO DE DEMING

Edward Deming era un estadista norteamericano que, tras la segunda guerra mundial, empezó a divulgar los conceptos relacionados con la mejora de los procesos y el aumento de la calidad.

Uno de los términos que popularizó fue el **Ciclo de Deming** que consiste en establecer un modelo de aprendizaje empírico, basado en cuatro fases (**PDCA**):

- *Plan* = Planificar
- *Do* = Hacer
- *Check* = Verificar los resultados.
- *Act* = Actuar en función de los resultados obtenidos



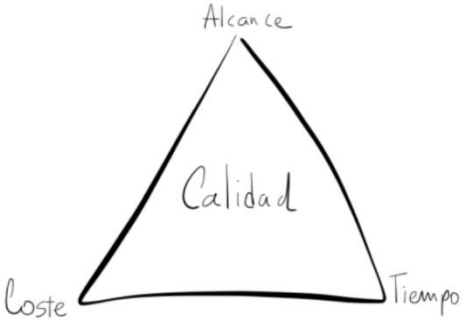
Es interesante esta aproximación ya que **la metodología ágil da especial importancia a la fase de feedback (Check) y a la de actuar (Act)**, lo que permitirá validar los resultados obtenidos y actuar en función del resultado continuamente, favoreciendo la mejora continua.

4. TRIÁNGULO DE HIERRO

Las tres variables que se deben considerar al estimar un proyecto son:

- **TIEMPO** necesario para realizar el proyecto (planificación).
- **ALCANCE**, que se refiere a los requerimientos del proyecto.
- **COSTE** a invertir para poder ejecutar el proyecto, directamente relacionado con las personas y recursos necesarios.

El Triángulo de Hierro sigue un enfoque de cascada (Waterfall) para el desarrollo de productos en el cual el **alcance** es fijo y los **recursos** y el **tiempo** son variables.



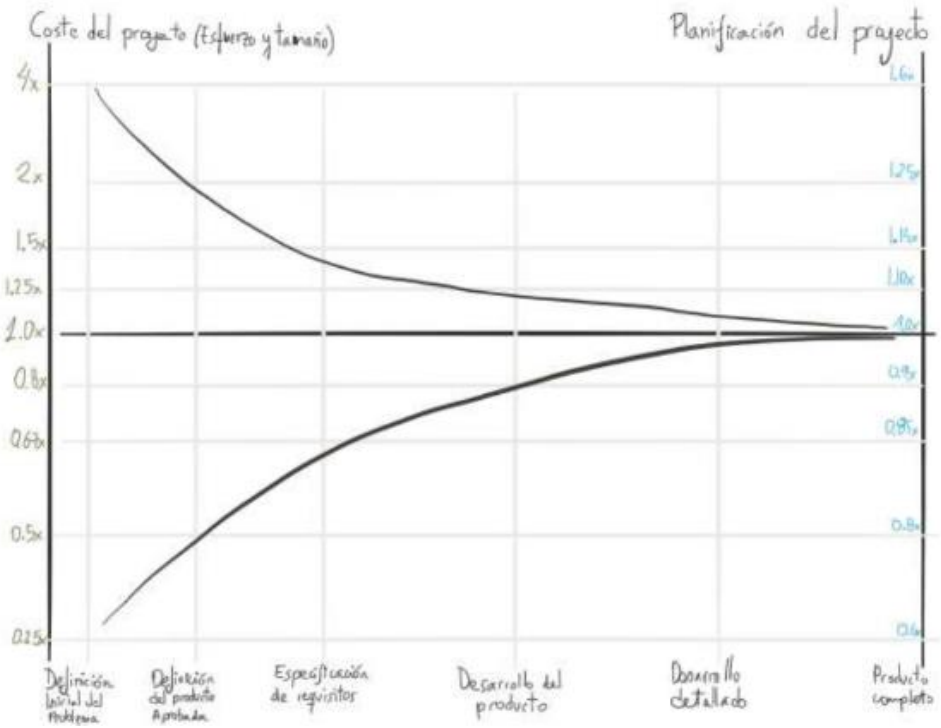
En cambio, en los **proyectos ágiles** lo que *se **fija es el tiempo y los recursos***, mientras que *lo que **varía es el alcance***. La idea de alcance en el desarrollo ágil es la misma: qué software construir y entregar. Sin embargo, Agile se enfoca en requisitos de alto nivel en lugar de tratar de presentar requisitos profundos y detallados por adelantado, lo que permite adaptarse al cambio en cualquier momento.

5. CONO DE INCERTIDUMBRE

El cono de incertidumbre describe la medida de incertidumbre de un proyecto.

Nos dice que al inicio de un proyecto tenemos mayor probabilidad de confundirnos en nuestras estimaciones ya que es la fase inicial, cuando menos información y conocimiento tenemos sobre la resolución del problema.

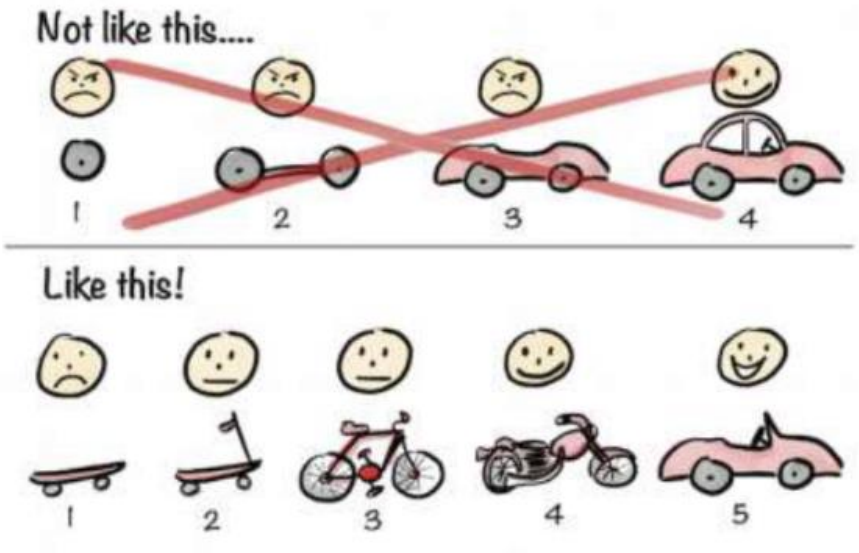
Las metodologías ágiles promueven el inicio de la fase de "hacer" lo antes posible para tratar que aparezcan estos impedimentos tan pronto como sea posible.



6. ITERATIVO E INCREMENTAL

En los proyectos en cascada (Waterfall) se divide el proyecto en fases secuenciales (análisis, diseño, construcción, pruebas y validación, etc), dejando la fase de validación al final del proyecto.

Los enfoques ágiles promueven un **desarrollo iterativo e incremental,** ya que están pensados para **proyectos con alta incertidumbre y una gran variabilidad en los requisitos**, es decir, muchos cambios. Por ello, es muy importante obtener feedback lo antes posible de nuestros clientes, con el objetivo de reducir la incertidumbre y saber si lo que estamos construyendo es realmente lo que quieren.



ITERATIVO: se divide el proyecto en pequeñas fases (o iteraciones) con el objetivo de entregar al final de cada iteración un pequeño entregable de nuestro producto que añada (incremente) valor al anterior.

INCREMENTAL: parte de lo sencillo y va dotando de complejidad al producto final.

7. METODOLOGÍAS ÁGILES MÁS POPULARES

Alguna de estas metodologías se trata en mayor profundidad en el tema 87: ANÁLISIS FUNCIONAL DE SISTEMAS, CASOS DE USO E HISTORIAS DE USUARIO. METODOLOGÍAS DE DESARROLLO DE SISTEMAS. METODOLOGÍAS ÁGILES: SCRUM Y KANBAN. A continuación, se listan las metodologías ágiles más populares indicando sus principales características:

- **SCRUM**: es un conjunto de prácticas y recomendaciones para el desarrollo de un producto, basada en un proceso iterativo e incremental y que promueven la comunicación, el trabajo en equipo y el feedback rápido sobre el producto construido. Sus objetivos principales son:
    - Obtener el feedback del usuario mediante entregas tempranas y planificadas, que permitan entregar un producto de calidad y ajustado a las necesidades del usuario.
    - Conseguir equipos de alto rendimiento
    - Roles principales: **Product Owner, Scrum Master, Scrum Team.**
    - Reuniones: **planificación (sprint planning), reunión diaria (daily), revisión de sprint (sprint review), retrospectiva del sprint (sprint retrospective)**.

- **KANBAN**: Método usado para gestionar la creación de productos con especial atención a la entrega continuada y a tiempo, sin sobrecargar al equipo de desarrollo (*ver detalle tema 87*). Se basa en 3 principios:
    - **Visualizar** lo que se hace (tablero Kanban público para fomentar la visibilidad).
    - **Limitar la cantidad de trabajo en curso** (WIP: *work in progress*).
    - **Mejorar el flujo de trabajo, eliminando cuellos de botella**.

- **Extreme Programming (XP)**: Es un marco de desarrollo de software ágil que tiene como objetivo producir un software de mayor calidad para mejorar la eficiencia del equipo de desarrollo. Está diseñada para ofrecer el software que los usuarios necesitan en el momento adecuado. En este

sentido, ayuda a los desarrolladores a **ajustarse a los requerimientos cambiantes de los clientes**. Las características fundamentales de esta metodología son:

- **Pruebas unitarias** continuas.
- **Planificación flexible** y abierta
- **Respuesta rápida y eficaz** ante posibles cambios.
- Interacción continua entre desarrolladores y cliente. Las **historias de usuario** sirven para crear test de aceptación y para estimar.
- **Refactorización del código**.
- Propiedad del **código compartido y rotación interna del equipo.**

- **Feature Driven Development (FDD):** desarrollo basado en funcionalidades. **Iteraciones cortas con funcionalidades tangibles**. Se diseña y construye a partir de una lista de funcionalidades (*features*). Se planifica, diseña y construye por cada una de las funcionalidades.

- **Test Driven Development (TDD)**: es una práctica de programación en la que se comienzan escribiendo las pruebas en primer lugar, escribiendo el código fuente a continuación, pasando correctamente la prueba y terminando con la refactorización del código escrito.

- **Behavior Driven Development (BDD)**: Se puede considerar una extensión del TDD. Pretende definir el comportamiento que ha de tener la aplicación, y no sólo verificar el funcionamiento del código. En BDD, las pruebas de aceptación se escriben usando **historias de usuario:**

  "Como [rol] quiero [característica] para que [los beneficios]."

  El ciclo básico de BDD sería el siguiente:

  - Escribir las historias de usuario (en un lenguaje entendible por el negocio).
  - Realizar la implementación.
  - Automatizar las pruebas de aceptación.

- **Dynamic Systems Development Method (DSDM)**: Metodología que se originó en la década de los 90 con el objetivo de crear un marco común para la entrega rápida de aplicaciones (RAD). Se basa en los siguientes principios básicos:
  - Centrarse en la **necesidad y valor para el negocio**.
  - Involucración del **usuario**.
  - **Entregas frecuentes**.
  - Desarrollo **iterativo e incremental**.
  - **Pruebas integradas**.
  - **Colaboración** entre las partes.

- **Crystal**: Una de las metodologías más sencillas, ligeras y adaptables. Familia de metodologías que incluye entre otras Crystal Clear, Crystal Yellow o Crystal Orange. Crystal resalta el **valor del trabajo en equipo, la comunicación y la reflexión** sobre el proceso, para mejorarlo y adaptarlo.

- **Lean**: Ver capítulo final de este resumen.

8. AGILE A GRAN ESCALA

Scrum puede funcionar bien para un único equipo que trabaja sobre un producto. Pero la guía Scrum no especifica cómo se trabaja cuándo hay varios equipos scrum sobre el mismo producto (escalado horizontal) o cuando queremos llevar el agilismo más allá del Software (*agile beyond software*) (escalado vertical).

En este apartado se detallan las peculiaridades de aplicar metodologías ágiles en proyectos o programas grandes. En este caso se necesitará coordinar numerosos equipos que trabajen de forma ágil. Para eso se usan metodologías ágiles a gran escala entre las que destacan:

- **SAFe (Scaled agile framework):** Estructurado en cuatro niveles (equipo, programa, gran solución y portfolio). A nivel de equipo se aplican principios Scrum, XP, Kanban y Lean. A otros niveles existen conceptos como *Program Increment* o *Agile Release Train* para coordinar múltiples equipos que trabajan sobre una misma solución.
- **Scrum de scrums**: Se escala la metodología Scrum de manera más pura.
- **LeSS** (Large-Scale Scrum)

Los principios de organización de los equipos se mantienen y se escalan, para obtener:

- Un equipo de equipos ágiles autoorganizados y autogestionados.
- Entrega de incrementos de funcionalidad cada poco tiempo.
- Operan con visión y arquitectura común.
- Duración de iteraciones común.
- Colaboración cara a cara.
- Entrega de valor de forma incremental.

SAFe

Para aplicar metodología ágil a nivel de programa, se recomienda dividir los equipos por área de funcionalidad y no por capa tecnológica. Se habla típicamente de unos 5 a 12 equipos (50 a 120 personas) que planifican, se comprometen y ejecutan de forma conjunta y coordinada. Para sincronizar los equipos, se definen los incrementos de programa o PI (**Program Increment**): periodo de tiempo predefinido donde las iteraciones de cada equipo se deben acomodar. Suelen ser de unas 10 semanas. Los diferentes equipos se reúnen al principio del Program Increment y acuerdan el trabajo a realizar, que después cada equipo organizará en sus sprints en función de dependencias con otros equipos.

La misión común se plasma en el **program backlog** y se desarrolla de forma coordinada mediante los **product backlog** de cada uno de los equipos.

SAFe se organiza en dos niveles:

1) **Programa**: en este nivel participan diferentes roles que forman parte del equipo llamado Release **Magnament Team**, y su objetivo es que el PI salga adelante y se entregue completo y a tiempo. Los miembros que forman parte de este equipo son: Release Train Engineer, Product Management, System Architect/Engineer y Bussiness Owners.

2) **Equipo (Team):** este nivel es la base sobre la que se apoya SAFe. Los equipos son los encargados de definir, desarrollar y probar las User Stories de los que se componen los PI que se utilizarán en niveles superiores para componer la solución ofertada en el nivel de porfolio. Existen 4 roles importantes: Scrum Master, Product Owner, Development team (5 a 9 personas) y Agile Team (trabajan con cualquier metodología ágil: Scrum, Extreme Programing, etc).

LeSS vs SAFe

Buscan un mismo objetivo y son relativamente similares, nombraremos algunas de las diferencias entre SAFe y LeSS para que se entienda a alto nivel las complejidades de escalar la agilidad:

	LeSS	SAFe
Número de Scrum Masters	1 full time por cada 2 ó 3 proyectos	No especifica el nº de Scrum Masters
Rol Scrum Master "Supremo"	No existe, todos los ScrumMasters son iguales	Nuevo rol: Release Train Engineer
Rol Product Master "Supremo"	Product owner	Nuevo rol: Product Manager
Metodología a nivel de equipo	Scrum	Scrum o XP o Kanban
Cuántos Product Backlogs	1 por grupo de *equipo*	1 por cada *equipo* + Backlog a nivel de *Programa*

9. LEAN

El origen se encuentra en el sistema de producción de Toyota. Mary Poppendieck y Tom Poppendieck lo "trasladan" al software. Proceso sistemático para eliminar desechos o desperdicios en los procesos (elementos que no aportan valor), reduciendo los costes y aumentando la calidad del producto o del servicio para entregar productos más rápido y con mejor calidad. La metodología Lean aplicada al SW no define un proceso concreto, lo importante es demostrar que los procesos están alineados con los principios.

PRINCIPIOS

1. Eliminar desperdicios. Se define desperdicio como cualquier cosa que no añade valor a un producto, desde el punto de vista del cliente. Ejemplo: es un desperdicio hacer más cosas de lo necesario o que no aportan valor al producto final.

2. Amplificar el aprendizaje. Como norma general no se obtiene una versión perfecta de nada a la primera, sino que se llevan a cabo varias versiones como parte del proceso de aprendizaje.

3. Decidir lo más tarde posible. Se debe de retrasar las decisiones a cuando se tenga suficiente información.

4. Entregar tan rápido como sea posible. Se realizan pequeñas entregas desde el inicio, recogiendo el feedback del cliente para ir mejorando.

5. Empoderar al equipo, involucrando a los desarrolladores en la toma de decisiones.

6. Embeber la calidad. Adquirir prácticas que aseguren la calidad lo antes posible.

7. Ver el todo. Uno de los grandes problemas de hoy en día de los proyectos software es que los expertos en cualquier área tienen tendencia a maximizar la importancia de aquella parte en la que son expertos. De tal manera que hay que evitar esto, y ver el sistema como un todo.

PRÁCTICAS Y TÉCNICAS

Existen una serie de técnicas y prácticas que se adoptan habitualmente dentro del ámbito lean para el desarrollo del SW:

- **Value Stream Mapping (VSM):** Mediante esta herramienta se representa la secuencia de actividades que se llevan a cabo, el promedio de duración y los tiempos de espera. Se usa para analizar el flujo de recursos y de información necesarios desde que se solicita un producto hasta que se entrega al cliente, representando la posición actual y localizando puntos de mejora.
- **5S**: Herramienta que ayuda a organizar el lugar de trabajo (ámbito lógico en ingeniería SW) y a conseguir procesos más eficientes. Consta de los siguientes elementos:
    - Sort (Clasificar)
    - Set in Order (Ordenar): Elementos (carpetas, ficheros) ubicados siguiendo la lógica que facilite su localización.
    - Shine (Limpiar): Eliminar del código todo lo que no es necesario.
    - Standardize (Estandarizar): Establecer procedimientos formales para todos los pasos.
    - Sustain (Sostener): Mantener el proceso a través de formación y comunicación.
- **Evento KAIZEN** (mejora continua): Evento en que se incorporan cambios al proceso para que mejore. Se reúnen los representantes de las áreas involucradas y deciden qué mejora afrontar y durante un espacio de tiempo no mayor a una semana trabajan en realizar las modificaciones en el proceso para lograr la mejora pactada.
- **Método Kanban**: Práctica que de forma simple y visual controla el flujo de estados de las tareas de un proceso. Permite limitar la cantidad de trabajo sin terminar en cada una de las etapas del flujo de trabajo.
- **Poka-yoke** (a prueba de errores): Técnica de calidad que se aplica con el fin de evitar errores en la operación de un sistema. Se busca eliminar errores ya sean humanas o automatizadas. Ejemplo: sistema de gestión que impide que un usuario modifique las tareas cogidas por otro miembro del equipo.
- **Jidoka**: Concepto por el que los procesos automáticos tienen su propio autocontrol de calidad. Si existe una anomalía en el proceso este se detendrá impidiendo que afecten al resto.