Modelos de desarrollo Ágil

El agilismo es una respuesta a los fracasos y las frustraciones del modelo en cascada. Contempla un enfoque para la toma de decisiones y la forma de organización en los proyectos de software, basándose en los modelos de desarrollo iterativo e incremental, con iteraciones cortas (semanas) y sin que dentro de cada iteración tenga que haber fases lineales.

En el 2001, 17 representantes de las nuevas tecnologías y el desarrollo de software se juntaron para discutir sobre el desarrollo de software. De esa reunión surgio el Manifiesto Ágil:

"Estamos descubriendo mejores maneras de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de esta experiencia hemos aprendido a valorar:

- **Individuos e interacciones** sobre procesos y herramientas
- **Software funcionando** sobre documentación extensiva
- **Colaboración con el cliente** sobre negociación contractual
- **Respuesta ante el cambio** sobre seguir un plan

Esto es, aunque los elementos a la derecha tienen valor, nosotros valoramos por encima de ellos los que están a la izquierda."

Estas ideas se explican en los 12 principios para el desarrollo ágil de software.

¿En qué consiste el desarrollo ágil? Características

- La esencia del agilismo es la habilidad para adaptarse a los cambios. La superación de obstáculos imprevistos tiene prioridad sobre las reglas generales de trabajo preestablecidas. Es decir, se para la forma habitual de trabajo o la forma en la que se lanzan nuevas versiones, para resolver un problema.
- También existen las fases de análisis, desarrollo y pruebas pero, en lugar de ser consecutivas, están solapadas y se suelen repetir en cada iteración. Las iteraciones suelen durar de dos a seis semanas.
- En cada iteración se habla con el cliente para analizar requerimientos, se escriben pruebas automatizadas, se escriben líneas de código nuevas y se mejora código existente (Refactorización).
- Al cliente se le enseñan los resultados después de cada iteración para comprobar su aceptación e incidir sobre los detalles que se estimen oportunos.
- Todo el equipo trabaja unido, y el cliente es parte de él. No es un oponente. La estrategia de juego ya no es el control sino la colaboración y la confianza.
- La jerarquía clásica (director técnico, analista de negocio, arquitecto, programador senior, junior ...) pierde sentido y los roles se disponen sobre un eje horizontal, donde cada cual cumple su cometido pero sin estar por encima ni por debajo de los demás.
- En lugar de trabajar por horas, se trabaja por objetivos y se usa el tiempo como un recurso más. (Pero existen fechas de entrega para cada iteración).
- En cualquier método ágil, los equipos deben ser pequeños, típicamente menores de siete personas.
- Todo el equipo se reúne periódicamente, incluidos usuarios. Las reuniones tienen hora de comienzo y de final y son breves.
- La aplicación se ensambla y se despliega a diario, de forma automatizada. Las baterías de tests se ejecutan varias veces al día.
- Los desarrolladores envían sus cambios al repositorio de código fuente al menos una vez al día (commit).

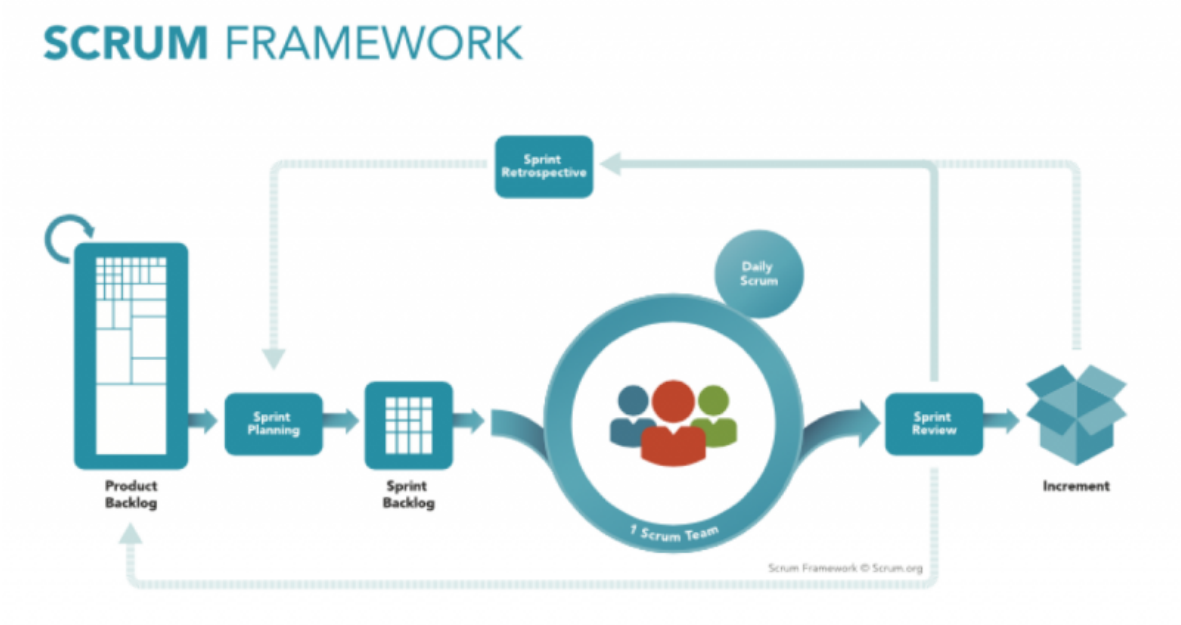
El abanico de metodologías ágiles es amplio, existiendo métodos para organizar equipos o proyectos, y técnicas para escribir y mantener el software. Tanto las técnicas de desarrollo de software como las de organización de proyectos se pueden combinar siendo de las más famosas la programación extrema junto con la gestión Scrum.

SCRUM

Es una técnica de *gestión de proyectos* englobada dentro del paradigma de desarrollo ágil. No se aplica únicamente al desarrollo de software, sino a la gestión de cualquier tipo de proyectos. El nombre de Scrum, se traduce por Melé, palabra del argot del Rugby usada para designar la unión de los jugadores en bloque.

Como las demás, se basa en el desarrollo **iterativo e incremental** en el que el proyecto se planifica en diversos bloques temporales (un mes, normalmente) llamados **Sprints**. Cumple con las características del desarrollo ágil.

La metodología Scrum se organiza de la siguiente forma:



Fases

- Reunión inicial

El cliente nos indica qué es lo que quiere. De esta reunión entre el cliente y el *Scrum Master* surge la lista de requisito del producto (*Product Backlog*). Se realiza una sola vez al inicio del proyecto, a diferencia de las demás fases que se realizan en cada Sprint.

- Planificación de la iteración (**Sprint Planning**)

Es una reunión realizada antes de cada Sprint en la que el cliente indica los requisitos prioritarios de la iteración (Sprint), se plantean las dudas y se traducen dichos requisitos a tareas (Sprint Backlog), que deben ser realizadas por cada uno de los miembros del equipo de desarrollo durante el Sprint. Estas tareas se incorporan a un panel que controlará su estado de realización (por hacer, en progreso, terminado, pendiente, etc).

- Ejecución de la iteración (**Sprint**)

Es el bloque de tiempo, iteración, en el cuál se realizan la tareas (2-4 semanas), siendo recomendable que siempre tengan la misma duración y la duración esté definida por el grupo en base a su experiencia.

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

Una vez comenzado el Sprint, cada día se celebra una reunión rápida (no más de 10 minutos, siempre puntual y suele ser de pié) en la que se habla del estado de las tareas del Sprint (Sprint Backlog). Se revisan las tareas en proceso del día anterior, y se marcan como terminadas. Se asignan nuevas tareas a miembros del equipos y se ponen en proceso. Solo se habla del estado de las tareas y solo los miembros del equipo pueden hablar.

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

Es una reunion que se produce al final del Sprint en la que se presenta al cliente el incremento (parte del producto pactada) realizado durante el Sprint. En ella el cliente puede ver cómo han sido desarrollados los requisitos que indicó para el Sprint (Sprint Backlog), si se cumplen las expectativas, y entender mejor qué es lo que necesita.

- Retrospectiva (**Sprint Retrospective**)

Cuando se ha terminado un *Sprint*, se comprobarán, medirán y discutirán los resultados obtenidos en el periodo anterior a través de las impresiones de todos los miembros del equipo. El propósito de la retrospectiva es realizar una mejora contínua del proceso.

Roles

- Cliente (**Product Owner**)

Es quién plantea los objetivos y requisitos y ayuda al usuario a escribir las **historias de usuario** que se incorporarán al *Sprint Backlog*, definiendo las prioridades. Representa a todas las personas interesadas en el proyecto (usuarios finales, promotores, etc). Es quién indica qué quiere en total y qué quiere en cada Sprint.

- Facilitador (**Scrum Master**)

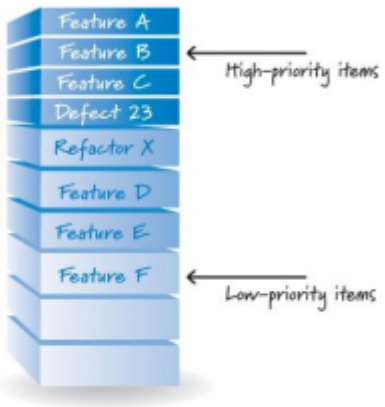
Se asegura de que se sigan los valores y principios ágiles, de la existencia de las listas de requisitos para cada iteracion, facilita las reuniones, quitar los impedimentos que se van encontrando en el camino y proteger y aislar al equipo de interrupciones.

- Equipo (**Team**)

Tienen un objetivo común, se autoorganizan, comparten la responsabilidad del trabajo que realizan, y cuyos miembros confian entre ellos. 5-9 personas es lo ideal, aunque si hay más se deben hacer más equipos

Herramientas

- Lista de requisitos priorizada (**Product Backlog**)



Es un documento abierto que representa la lista de objetivos o requisitos, y expectativas del cliente respecto a los objetivos y entregas del producto. Se concreta en la reunión inicial, antes de entrar en la planificación de los Sprints. Solo puede ser modificado por el *product owner*(cliente) aunque con ayuda del equipo y el "scrum

master", quienes proporcionan el coste estimado tanto del valor de negocio como del esfuerzo de desarrollo. Representa **el qué** va a ser contruido en su totalidad.

- Lista de tareas pendientes del Sprint (**Sprint Backlog**)

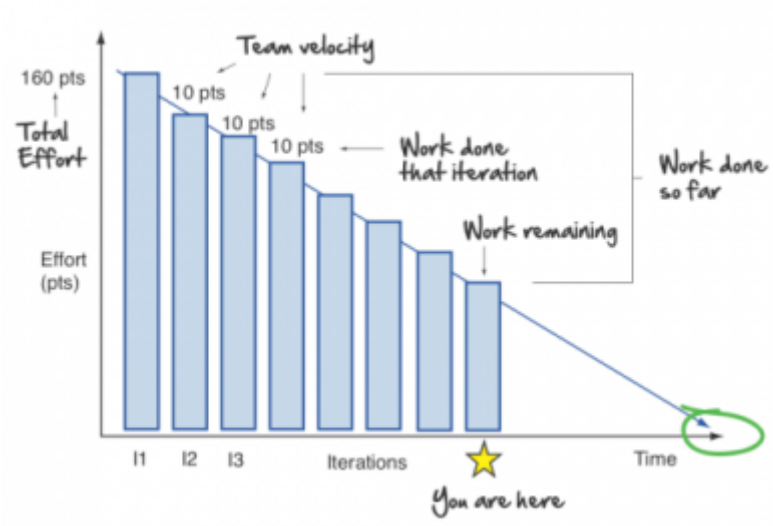
Lista de tareas que el equipo elabora en la reunión al inicio del Sprint,*Sprint Planning*, como plan para completar los requisitos para la iteración. Los requisitos tienen que estar claramente detallados en tareas (historias de usuario), a diferencia del *Product Backlog*. No se deben agregar requisitos al *Sprint Backlog* a menos que su falta amenace al éxito del proyecto. Representa **el cómo** el equipo va a implementar los requisitos.

La forma de definir las tareas o los requisitos de cada Sprint se suele hacer en terminos de historias de usuario (user stories). Suelen ser frases cortas definiendo en qué consiste la tarea. Algunos requisitos puede que no sean traducibles a historias de usuario.

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 formating 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

- Gráficos de trabajo pendiente (**Burndown Chart**)

Es un gráfico de trabajo pendiente a lo largo del tiempo, en el que se muestra la velocidad a la que se están completando los requisitos. También ayuda a intuir si el equipo terminará en el tiempo estimado. Lo normal es que la linea que une todos los sprints completados sea descendente, pero si se añaden nuevos requisitos durante el proceso, puede ser ascendente.



En resumen, Scrum permite conocer en un vistazo rápido cual es el estado general de las cosas, detectando rápidamente qué tareas se han quedado atascadas o qué equipos no están rindiendo al nivel que se esperaba.Es un método de desarrollo ágil ideal para entornos con mucha incertidumbre en cuanto al trabajo a realizar, en los que las tareas cambian muy rápidamente y son susceptibles de olvidarse.

En el siguiente enlace podéis ver un video sobre la herramienta Trello y Scrum

https://player.vimeo.com/video/240464329