METODOLOGÍAS EN EL DESARROLLO DE SOFTWARE

"El rugby no se trata solo del talento individual, sino de cómo quince individuos funcionan como una sola unidad."

(Clive Woodward)

¿QUÉ ES UN PROYECTO?

Un proyecto es una serie de tareas que deben llevarse a cabo para lograr un objetivo predefinido. Las organizaciones de todos los tipos y tamaños llevan a cabo proyectos que varían en complejidad, plazo y recursos.

Un proyecto tiene las siguientes tres características:

Crea un resultado nuevo y tangible: El objetivo principal de un proyecto es crear algo nuevo.

Tiene lugar por fuera de las rutinas comerciales normales: Un proyecto es una actividad distinta, más allá de las operaciones comerciales diarias.

Tiene límites: Un proyecto está limitado por varios factores, como el tiempo, el costo, los recursos y las entregas.

La gestión de proyectos (PM) se refiere a la forma en que organiza, supervisa y lleva a cabo su proyecto, y genera valor. La PM exitosa requiere conocimientos, habilidades, herramientas y técnicas específicas para ejecutar el proyecto de manera eficaz dentro de parámetros dados, como el cronograma y el presupuesto.

¿QUE ES UNA METODOLOGIA DE DESARROLLO DE SOFTWARE?

En el mundo del desarrollo de software, el éxito de un proyecto de software no sólo depende de las habilidades técnicas y el conocimiento del equipo, sino también de la metodología que se adopte durante el proceso de desarrollo. Hablar de metodologías de desarrollo de software es hablar de un conjunto de métodos, técnicas y prácticas que se emplean para llevar adelante el diseño de una solución de software desde la fase de planificación, diseño, construcción, pruebas y entrega final bajo estándares de calidad.

DIFERENTES METODOLOGÍAS DE DESARROLLO DE SOFTWARE

Al elegir una metodología de desarrollo de software, es importante evaluar las características y necesidades del proyecto, así como las capacidades y experiencia del equipo. No existe una metodología única que funcione para todos los proyectos; en su lugar, es fundamental seleccionar y adaptar la metodología que mejor se adapte a las demandas y desafíos específicos de cada proyecto de software. Existen diversas metodologías de desarrollo de software, que se pueden clasificar en dos categorías principales: tradicionales y ágiles.

METODOLOGÍAS TRADICIONALES

Dentro de las metodologías tradicionales podemos encontrar al Modelo en cascada, el Modelo en espiral y el Modelo en V. La metodología en cascada (del inglés Waterfall), es también llamado enfoque predictivo. En él, los equipos trabajan de manera más independiente y la gestión del proyecto es lineal: los requisitos se recopilan al inicio del proyecto y se crea un plan de proyecto según esos requisitos. Debido a que cada fase del proyecto desemboca en la siguiente, el nombre de la metodología representa lógicamente este movimiento en "cascada". La metodología en cascada, con su estructura lineal y secuencial, se ha mantenido firme en la industria durante décadas. Sus ventajas son claras: ofrece un plan de acción detallado y una visión clara del proyecto desde el principio. Sin embargo, su rigidez a menudo puede ser un desafío en un entorno de desarrollo de software donde los requisitos cambian con frecuencia. La inflexibilidad de este modelo con respecto a los cambios una vez que un proyecto ha comenzado puede causar retrasos e incrementos en los costos. La metodología Waterfall consta de cinco etapas principales:

Especificación de Requisitos: Es la recopilación de todos los requisitos del cliente antes del inicio del proyecto que permite la planificación de cada fase siguiente.

Nota

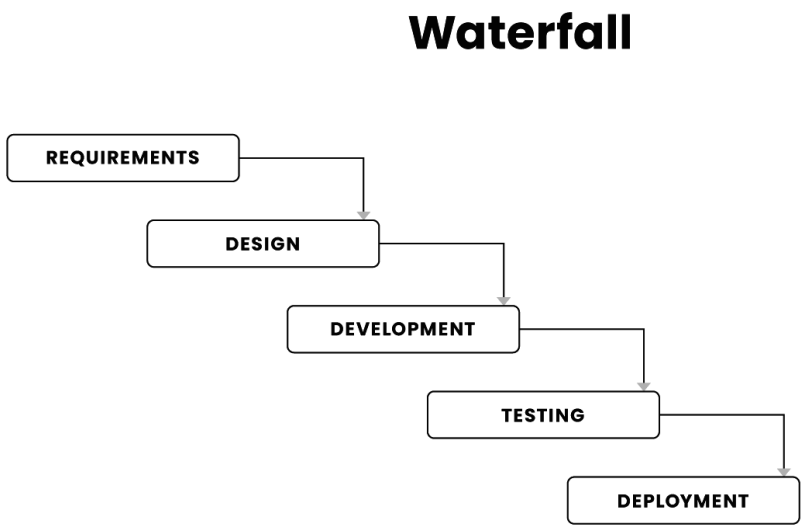
Un requisito es una declaración que describe una característica, función o restricción que debe cumplir un sistema o software. Estos requisitos pueden ser de dos tipos: funcionales, que describen lo que el sistema debe hacer (por ejemplo, registrar un usuario), y no funcionales, que describen cómo el sistema debe ser (por ejemplo, ser seguro o rápido).

Diseño : Esto incluye el diseño lógico y el diseño físico: generar ideas sobre soluciones y convertirlas en especificaciones.

Implementación: Cuando los desarrolladores toman todos los requisitos y especificaciones y escriben el código para la solución.

Prueba: Las pruebas de software son necesarias para detectar los errores y para probar si el software cumple con los requisitos. Esto ayuda al equipo de desarrollo a corregir los errores y entregar un producto de buena calidad.

Lanzamiento o despliegue: Esta etapa incluye la implementación del software en el entorno de producción, la capacitación de los usuarios y la realización de cualquier migración de datos necesaria.



Con Waterfall, los equipos trabajan de forma más independiente y no necesitan estar en comunicación continua ni dar informes continuamente. Los equipos se agrupan por función y cuando cada equipo termina su parte, la "entregan" al siguiente equipo que necesita continuar con el desarrollo. Este tipo de metodología es fácil de gestionar y ejecutar, pero necesita de una intensa planificación inicial incluyendo un presupuesto preciso.

METODOLOGÍAS AGILES

La idea básica detrás de cualquier método de desarrollo Agile es que la implementación del software ocurre de forma incremental. Dentro de este tipo de metodología encontraremos a Scrum, Kanban y programación extrema (XP). La metodología Scrum es un subconjunto de las metodologías Agiles y es el marco de proceso más popular que implementa Agile. Fue creado en 1993 por Jeff Sutherland, quien adaptó la formación Scrum utilizada por los equipos de rugby para el desarrollo de software. En lugar de desarrollar el software de una sola vez, Scrum divide el proyecto en varias iteraciones, cada una con una duración de entre 1 y 4 semanas denominadas sprint. Al dividir el proyecto en iteraciones manejables, el equipo recopila más requisitos y reduce el riesgo de no satisfacer las necesidades de los usuarios. Durante la iteración, el equipo diseña, construye y prueba funciones completas que pueden lanzarse como un producto funcional. La aplicación resulta inmediatamente valiosa para los usuarios y puede mejorarse en las siguientes iteraciones. Comparando este enfoque con el modelo de desarrollo en Cascada, existe un valor que se entrega constantemente en intervalos regulares, mientras que el modelo en Cascada entrega todo el valor al final del proyecto.

¿QUÉ ES EL MANIFIESTO ÁGIL?

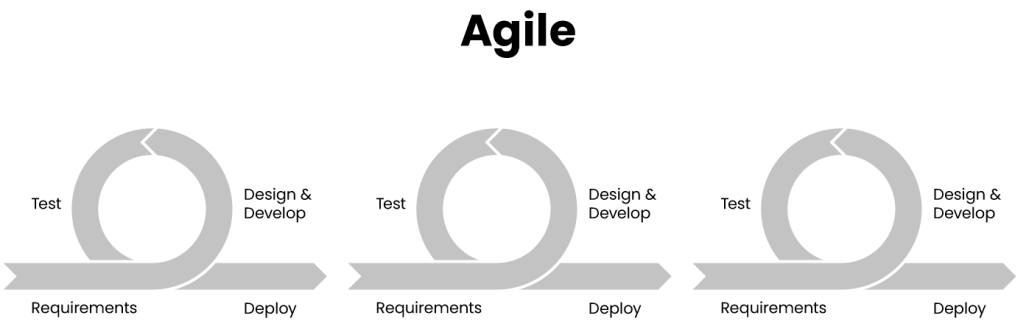
El Agile Manifesto, Manifiesto ágil o Manifiesto para el desarrollo de software ágil es una declaración de valores y principios sobre nuevas formas de desarrollar software que surgió en 2001, como reacción a los tradicionales métodos formales con los que se trabajaba entonces en la industria en ese manifiesto se describieron 4 prioridades fundamentales:

1. Las personas e interacciones antes que los procesos y las herramientas
2. El software funcional antes que la documentación completa
3. La colaboración con los clientes antes que la negociación del contrato
4. Responder a los cambios antes que seguir un plan

METODOLOGIA SCRUM

Scrum se divide en tres roles principales: el Product Owner, el Scrum Master y el equipo de desarrollo. El Product Owner es la persona encargada de representar lo que quiere el cliente. Es decir, establece la prioridad de las tareas para cumplir con los objetivos en base a las necesidades del cliente final. Además, es la persona que mantiene contacto directo con el cliente. El Scrum Master ayuda a todo el equipo para que se cumplan las metas. Motiva y estimula a todos los profesionales, y es la figura que sirve de ayuda ante cualquier problema que pueda surgir. Cumple la función de guía para todos los miembros del equipo. El equipo de desarrollo es responsable de realizar el trabajo necesario para entregar los productos y servicios de alta calidad en el menor tiempo posible suele estar compuesto por un grupo de no más de 10 personas. Realizan las tareas necesarias y se organizan y gestionan ellos mismos. Cada uno tiene su rol y sabe cuál es la responsabilidad de su trabajo dentro del proyecto. No hay jerarquías, trabajan horizontalmente. La metodología Scrum se puede aplicar en cualquier empresa, adaptando sus características. No obstante, lo más adecuado es que la empresa y el proyecto cumpla las siguientes premisas:

- Está pensado para equipos pequeños. Es decir, que tengan no más de 10 miembros. Aunque se puede aplicar en equipos más grandes, su eficacia no es la misma ya que no permiten cambiar de dirección rápidamente.
- Cuando se busca el control y la rapidez en las entregas. Lo que importa es alcanzar el objetivo en el plazo de tiempo acordado.
- Que exista flexibilidad a la hora de realizar el proyecto. Es decir, posibilidad de ir adaptándose a cada paso con cambios en mitad del proceso. Con posibilidad de reestructurar tareas.



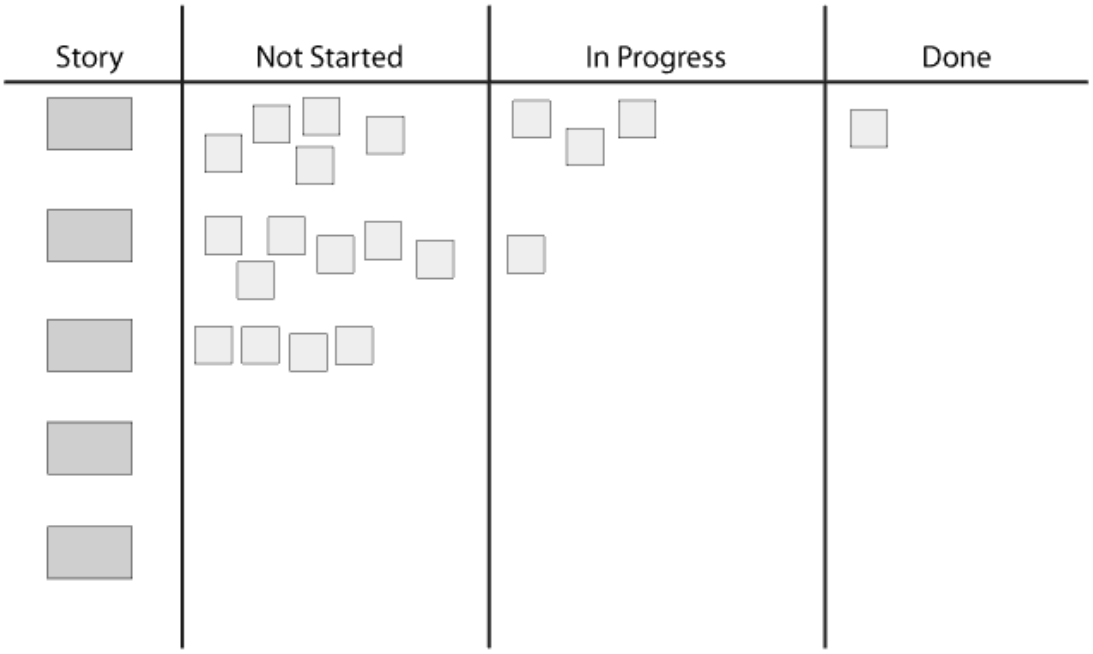
¿QUÉ ES UN SPRINT?

Un sprint consiste en un conjunto de tareas que dura un cierto tiempo. Las tareas que se incorporan al sprint están definidas por el backlog del producto (Es responsabilidad del Product Owner mantener y priorizar este backlog). Antes del sprint, el equipo se reúne para planificarlo y fija un backlog del sprint. Al final del sprint, se realizará una reunión/análisis junto con una demostración del trabajo completado. Aquí las mejoras son analizadas y se planifica el trabajo para el siguiente sprint. El objetivo de cada sprint es un incremento del producto. Cada uno de ellos es un producto parcialmente terminado que funciona. El sprint está basado en dos pilares: la priorización y la reevaluación. Esto implica que un sprint se adapta a las novedades y aprende de los resultados obtenidos en otros sprints, para responder de forma más flexible en los siguientes sprints. El tablero Scrum es un elemento imprescindible en este proceso de trabajo.

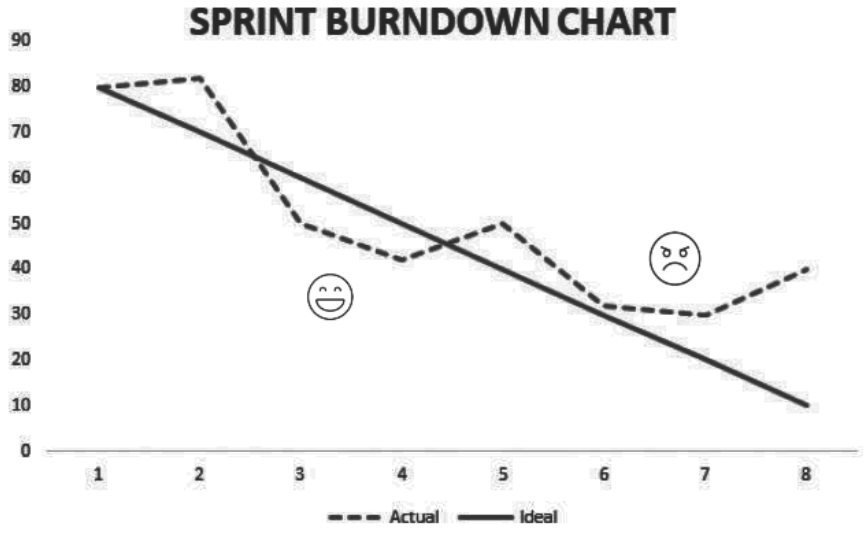
TABLERO SCRUM

Un tablero Scrum (también llamado Scrum Board) es una herramienta visual que permite al equipo hacer seguimiento del progreso del trabajo durante un sprint. La mayoría de los equipos Scrum utilizan un tablero Scrum para que puedan aislar y organizar las tareas, asignar tareas específicas a los miembros del equipo y dar seguimiento a cada tarea a través de su ciclo de vida. Un tablero Scrum siempre incluirá columnas para Historia, Tareas pendientes (To Do) , Trabajo en curso (Doing) y Trabajo realizado (Done). Los equipos que deseen obtener más detalles sobre el estado de una tarea pueden agregar columnas adicionales, como, por ejemplo: No iniciado o Para verificar. Independientemente de las columnas que elija, es importante definir claramente "hecho" para cada historia, ya que el objetivo es completar cada historia al final del Sprint. Los miembros del equipo se asignan una tarea durante la sesión de planificación de Sprint y asumen la propiedad total del trabajo asignado.

Al comienzo de cada Sprint, todas las tareas se colocan en la columna más a la izquierda detallando cada actividad, la persona responsable y las fechas de inicio y limite. Las tarjetas se desplazan de izquierda a derecha a través de las otras columnas a medida que avanzan hacia su finalización. Este es un ejemplo de cómo se vería un tablero de Scrum en medio de un Sprint.



Al final de un sprint, el Scrum Master realiza una retrospectiva de Sprint para evaluar el progreso general, un Sprint tiene éxito si todas las historias están en la columna "Hecho". Luego, el tablero se limpia de tareas completadas y se prepara para el próximo Sprint. Una de las métricas más utilizadas en Scrum es el Burndown Chart, un gráfico que muestra de forma visual en tiempo real el trabajo que queda por hacer en contraste con el tiempo que queda en el sprint. Gracias a este grafico el Scrum Master identifica no solo los retrasos, sino que también proporcionando datos esenciales para las reuniones de retrospectiva. El eje de coordenadas Y (vertical) representa la cantidad de trabajo por realizar o pendiente. Sobre el eje X (horizontal) se representa el tiempo (días del sprint).



Si su equipo decide implementar Scrum, el primer paso es elegir si usará un tablero Scrum físico o en línea. Ambos tienen sus beneficios y desventajas, pero es importante evaluar las necesidades de su equipo antes de comprometerse con un estilo determinado.

Nota

Jira desarrollado por Atlassian, es una poderosa herramienta de gestión de proyectos ampliamente utilizada para el desarrollo ágil de software. Cada persona del proyecto crea tickets para diferentes tareas. Cada ticket tiene un estado para que quede claro qué tareas se han completado y cuáles están pendientes. El uso exacto de Jira depende del equipo y de los requisitos respectivos. El software es muy flexible y puede optimizarse para diferentes formas de trabajo.

ÉPICAS e HISTORIA DE USUARIOS

Para poder avanzar, es necesario la correcta identificación de las necesidades planteadas por el cliente. Para esto deberíamos entender el concepto de Épica e Historia de usuario. En Scrum, una épica es un término utilizado para describir un tipo de elemento de trabajo o requisito de alto nivel que es demasiado grande para ser completado en una sola iteración o sprint. Una épica es esencialmente una historia de usuario muy grande y compleja que se divide en varias historias de usuario más pequeñas y manejables para que puedan ser planificadas y entregadas dentro de un sprint. Las épicas se utilizan para describir las funcionalidades o características de alto nivel que el equipo de Scrum debe desarrollar. Por ejemplo, una épica de una app de ciclista podría ser el desarrollo de un sistema de Navegación y Seguimiento en Tiempo Real. Las épicas se dividen en historias de usuario más pequeñas y manejables durante el proceso de planificación de Sprint, y cada historia de usuario se puede estimar y desarrollar de forma independiente. La descomposición de una épica en historias de usuario permite que el equipo de Scrum tenga una mejor comprensión de los detalles del trabajo y de los requisitos específicos, lo que a su vez ayuda a planificar y gestionar el trabajo de forma más eficiente. Dado que una épica comprende varias historias de usuario, normalmente se necesitan varios sprints para completarlas todas. Una historia de usuario es una descripción concisa de un requisito de software desde el punto de vista de un usuario final. El propósito de una historia de usuario es articular cómo una característica particular proporcionará valor al cliente. En otras palabras, ¿cómo usarán el producto para resolver problemas o abordar puntos débiles en su vida?. Normalmente, una historia de usuario sigue el formato de las 3W: Who? (¿Quién?), What? (¿Qué?) y Why? (¿y por qué?). Para ello usaremos la siguiente plantilla:

"Como [tipo de usuario], quiero [meta u objetivo] para que [beneficio o resultado]".

Ejemplos de historias de usuarios para la épica mencionada anteriormente:

Ejemplo1:

Como ciclista **quiero** seguir mi recorrido en tiempo real en Google Maps **para que** pueda orientarme durante el recorrido.

Ejemplo2:

Como ciclista **quiero** guardar mis recorridos completados **para que** pueda repetirlos en el futuro o compartirlos con otros ciclistas.

Después de que las historias de usuario estén definidas, el equipo dividirá cada historia en pedazos más pequeños denominadas tareas y asignará cada tarea a un solo miembro del equipo que sea responsable de completarla al final de un período de trabajo de duración fija, llamado Sprint. De esta manera, los trabajadores no dividen su energía entre múltiples tareas durante el Sprint: centrarse en una sola tarea aumenta la probabilidad de que se complete a tiempo. Con ellas, el equipo de desarrollo entra en detalles técnicos sobre cómo se implementarán las piezas más pequeñas. A diferencia de las historias, las tareas son más directas y con lenguaje técnico, no siguiendo un formato textual definido. Las tareas identifican algo que debe hacerse como parte de una Historia. Como ejemplos de tareas para nuestra aplicación de ciclistas podemos indicar: Crear un mensaje emergente en la aplicación informando la distancia recorrida cuando se llega a destino en el recorrido trazado en Google Maps. Las tareas suelen ser definidas por quienes realizan el trabajo (desarrolladores, testers, etc.), mientras que las historias y las epopeyas suelen ser creadas por el cliente o el propietario del producto en nombre del cliente. Por lo tanto, las tareas ya no necesitan ser comprensibles para los usuarios de negocio y, por lo tanto, pueden ser muy técnicas. El proceso de descomponer una historia en tareas también ayuda al equipo de desarrollo a comprender mejor qué se debe hacer.

PRODUCT BACKLOG Y SPRINT BACKLOG

En scrum los artefactos representan valor o trabajo y su objetivo es maximizar la transparencia de la información. En scrum hay tres artefactos: El Product Backlog, el Sprint Backlog y el incremento. El Backlog del Producto es una lista de cosas por hacer para completar un producto scrum. El Product Backlog se modifica a lo largo de todo el proyecto. Si es necesario, se agregan nuevos requerimientos y los requerimientos existentes pueden modificarse, definirse con más detalle o incluso eliminarse. Las entradas en el Product Backlog son a menudo escritas en forma de Historias de usuarios. El backlog del sprint es una lista de elementos del backlog del producto que el equipo puede completar durante cada sprint.

¿CUÁLES SON LOS TIPOS DE REUNIONES SCRUM?

Para que un proceso Scrum tenga éxito hace falta no sólo que se terminen las tareas de un sprint, sino que se celebren las reuniones con todos los participantes. Hay cuatro tipos de reuniones:

- En la planificación del sprint (Sprint Planning): el equipo Scrum decide qué tareas del backlog de producto van a acometerse en el siguiente sprint. El equipo de desarrollo lleva a cabo la planificación de forma libre.
- La reunión Scrum diaria (Daily Scrum): Es una reunión corta que tiene lugar a la misma hora todos los días, donde los miembros del equipo informan sobre lo que hicieron el día anterior, lo que planean hacer ese día y si tienen algún obstáculo.
- En la revisión del sprint (Sprint Review): El equipo Scrum presenta los resultados de un sprint terminado al propietario del producto. Esta información le sirve al propietario para modificar o añadir tareas en el backlog del producto.

- La retrospectiva del sprint (Sprint Retrospective): Se realiza al final para que el equipo pueda mirar hacia atrás en su trabajo e identificar elementos que podrían mejorarse. Recolectar información, Generar ideas y Decidir qué hacer. ¿Se han terminado todos los tickets a tiempo? ¿Se han producido novedades no previstas? Las respuestas a estas preguntas ayudan en la planificación de los siguientes sprints.

SCRUM VS. WATERFALL: UNA COMPARACIÓN

Si tuviéramos que señalar la principal diferencia entre Scrum y Waterfall como metodologías para el desarrollo de software, sería que Scrum se basa en valores con iteraciones más cortas y Waterfall se basa en cronogramas con costos y planes claramente estimados. Aquí hay una comparación de algunas de las diferencias básicas entre los dos.

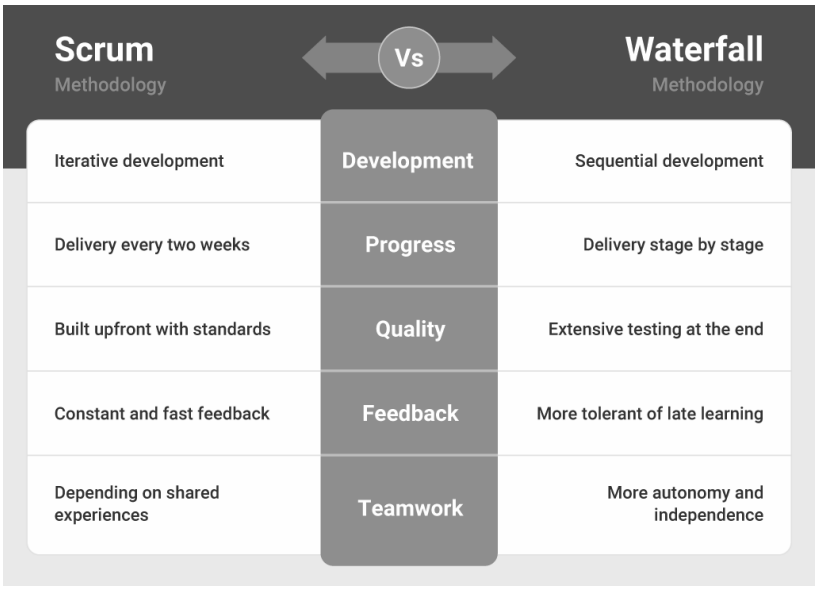
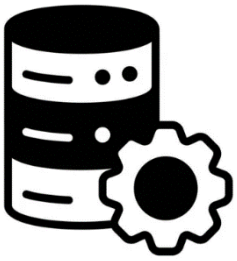


Figura 5.4. Diferencias entre waterfall y scrum.

Desarrollo: Con Scrum al tener un enfoque con interacciones y tener un producto mínimo viable que asegure la usabilidad, se va generando cambios y adaptaciones para tener la mejor versión posible. En Waterfall ese modelo es secuencial.
Progreso: Con Scrum vemos el progreso mediante la entrega de características valiosas cada dos semanas y con Waterfall, hay informes sobre cada etapa y actividad.
Calidad: Con Scrum, la calidad se construye desde el principio con estándares mientras que con Waterfall, la calidad del producto se define con pruebas exhaustivas al final.



Nota

En general en el modelo Waterfall muchas veces el presupuesto y el tiempo dedicado al proyecto se agota en la fase de prueba, para entregar el producto más rápido se dedica a realizar menos prueba y la calidad del producto final se ve afectada.

Comentarios: Scrum se organiza en torno a comentarios constantes y rápidos para el equipo; Waterfall es más tolerante con el aprendizaje tardío.

Nota

En el modelo ágil requiere una participación mucho mayor del cliente. Esto puede ser una desventaja ya que algunos clientes no tienen tiempo para hacer un seguimiento del producto.

Trabajo en equipo: Scrum utiliza equipos multifuncionales con conocimiento común del producto y experiencias compartidas y Waterfall trabaja con conocimientos almacenados en documentos y mucha más autonomía e independencia.