Taller sobre metodologías de desarrollo de software

Danny Julián Perilla Mikán

Mayo, 2024

1 Introducción

En la industria del desarrollo de software se proponen un conjunto de marcos de trabajo y artefactos que el equipo de desarrollo de software usa para realizar un trabajo organizado que sea fácil hacerle seguimiento y de esta forma establecer planes de mejora en busca de una mejor calidad de los productos y servicios que se desarrollan. Existen dos grandes clasificaciones de marcos de trabajo de software que se agrupan generalmente como **marcos de trabajo tradicionales** o **marcos de trabajo ágiles**. La principal diferencia entre las dos, radica en que en las primeras el proceso es lineal y secuencial o en cascada, mientras que en las segundas el proceso es iterativo y es adaptable a cambios en los requerimientos. Algunas veces es común referirse a los marcos de trabajo tradicionales y ágiles como metodologías. Sin embargo, es importante distinguir la diferencia entre metodología y marco de trebajo: los marcos de trabajo describen qué hacer, pero permiten que quien lo hace decida cómo hacerlo. Una metodología es más precisa y especíca: qué hacer, cuándo hacerlo, cómo hacerlo y por qué.

Los marcos de trabajo tradicionales y ágiles también se pueden ver como paradigmas de modelos del ciclo de vida del software[1] ya que estos no solamente dan las directrices para trabajar en equipo de manera eficiente sino que también definen procesos en la etapa de análisis, en la ingeniería de requisitos, más exactamente cuando se hace el análisis y la especificación de requisitos[2].

2 Marcos de trabajo tradicionales

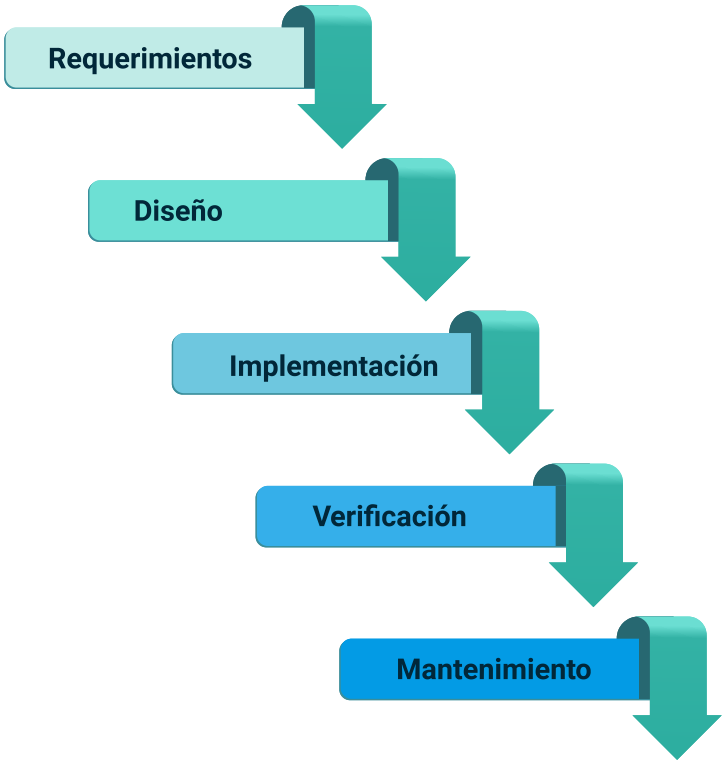
Los marcos tradicionales se identifican, fundamentalmente, por ser lineales, es decir se trata de completar cada proceso de principio a fin hasta que quede listo para avanzar a la segunda fase del ciclo del software. Se caracterizan por centrar la mayor parte de su esfuerzo en la planeación y control del proceso, lo que conlleva a una documentación exhaustiva y precisa de los artefactos que describen los requisitos y los modelos del sistema en las etapas iniciales del desarrollo del proyecto.

2.1 Waterfall

Este es uno de los modelos genéricos más ampliamente conocido en la ingeniería de software. Este modelo plantea un proceso lineal donde las actividades de desarrollo de un producto o servicios de software se agrupan en un conjunto de fases sucesivas donde estas son desarrolladas una única vez y los resultados de cada fase son la entrada requerida para cada fase subsiguiente. Se compone de cinco etapas principales que se asocian con actividades fundamentales en el proceso de desarrollo de

[1] Recordemos que el ciclo de vida del software posee: planeación, análisis, diseño, implementación, pruebas y mantenimiento.
[2] Recordemos que la ingeniería de requisitos se divide en 4 etapas: elicitación, análisis, especificación y evaluación.

software. El modelo en cascada define cuatro grupos de roles típicos: desarrolladores, testers, analísta de negocio y administrador del proyecto.



2.2 Otros marcos tradicionales

Aquí solamente haremos hincapié en RUP que por sus siglas en inglés equivalentes a Proceso Racional Unificado, el cual es un proceso de desarrollo de software tradicional basado en el modelo cascada y que fue desarrollado por la empresa Rational Software que es propiedad de IBM, esta metodología se centra en la arquitectura y es guía por casos de uso. RUP divide el proceso de desarrollo en cuatro grandes fases: incepción, elaboración, construcción y transición, dentro de las que se realizan algunas iteraciones donde se desglosan, en mayor o menor intensidad, un conjunto de actividades según la fase que se está abordando.

3 Marcos de trabajo Ágiles

El objetivo de los marcos ágiles es el desarrollo de proyectos en poco tiempo, se simplifican procesos tediosos, se agilizan las fases del desarrollo y las interacciones se hacen en corto tiempo. Una de las principales diferencias con los marcos tradicionales es que *el cliente se ve involucrado en el proyecto durante el desarrollo de este, así, el cliente sugiere mejoras, propone ideas y se mantiene al tanto del desarrollo del producto*, a diferencia del paradigma tradicional. El inicio de las metodologías ágiles nació en el año 2001 a partir del manifiesto ágil de software donde se establecen cuatro valores fundamentales (Manifiesto Ágil, 2001):

1. Individuos e interacciones sobre procesos y herramientas.

2. Software funcionando sobre documentación extensiva.
3. Colaboración con el cliente sobre negociación contractual.
4. Respuesta ante el cambio sobre seguir un plan.

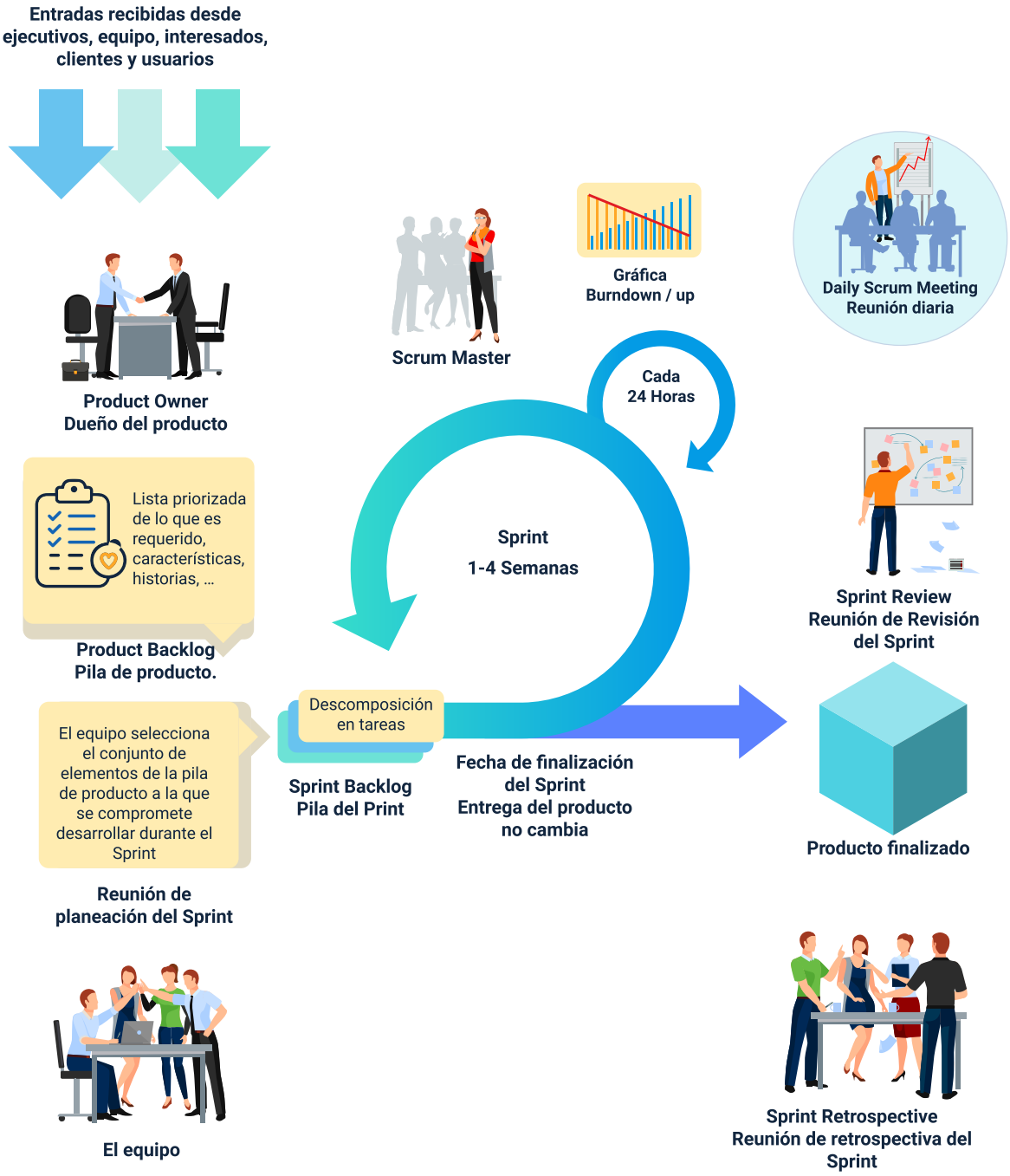
Adicionalmente, el manifiesto ágil establece 12 principios ágiles Kniberg [2015]:

N°	Principio
1	Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.
2	Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
3	Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los períodos breves.
4	Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
5	Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.
6	La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.
7	El software que funciona es la principal medida del progreso.
8	Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.
9	La atención continua a la excelencia técnica enaltece la agilidad.
10	La simplicidad como arte de maximizar la cantidad de trabajo que se hace, es esencial.
11	Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto organizan.
12	En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

"el cliente se ve involucrado en el proyecto durante el desarrollo de este, así, el cliente sugiere mejoras, propone ideas y se mantiene al tanto del desarrollo del producto."

3.1 SCRUM

Scrum es un marco de trabajo ágil de muy amplio uso en la industria del software que se fundamenta en los valores y principios ágiles definidos en (Manifiesto Ágil, 2001) y donde se definen tres pilares fundamentales: transparencia, inspección y adaptación. Adicionalmente, este marco de trabajo ágil está estructurado por un conjunto de roles: Product Owner, Scrum Master, Developer Team; eventos: sprint planning meeting, daily scrum meeting, sprint review y sprint retrospective; artefactos: Product Backlog, Sprint Backlog, Burndown Chart, que pueden ser observados a continuación.

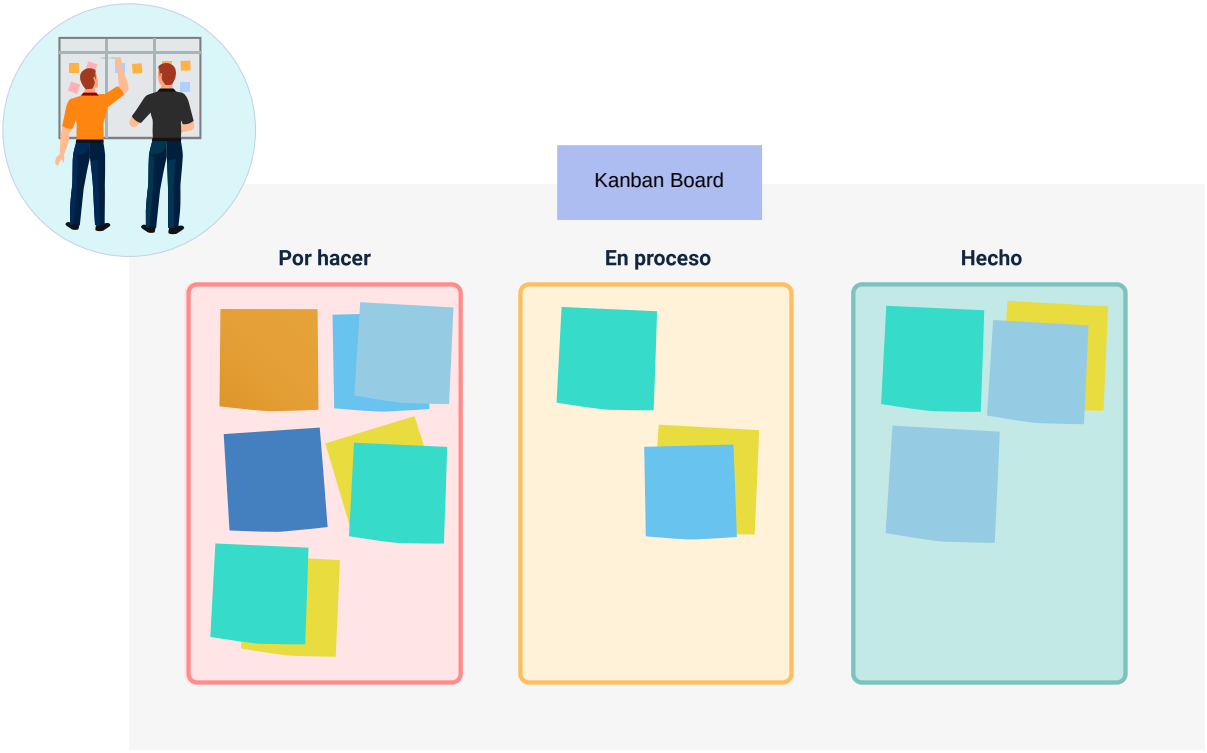


3.2 Kanban

Kanban es una metodología para gestionar el trabajo que surge del sistema de producción Toyota Production System (TPS) a finales de la década de 1940, el cual representaba un sistema de producción basado en la demanda de los clientes y no en la producción masiva, lo anterior sentó los fundamentos para los sistemas de producción ajustada que consisten en minimizar los desperdicios sin afectar la producción y en crear más valor a los clientes sin generar más gastos.

Tal vez la herramienta más conocida que permite implementar los principios de Kanban es el **tablero**

Kanban, el cual permite mapear y visualizar el flujo de trabajo. Este se divide en columnas a través de las cuales se pueden visualizar cada una de las fases del proceso; las filas del tablero representan los diferentes tipos de actividades específicas que se desarrollan en el marco del proyecto. Normalmente el tablero tiene tres secciones que representan el estado de cada una de las tareas: **por hacer**, **en proceso**, **hecho**. Cada equipo de trabajo puede realizar un mapeo más detallado de su proceso y agregar tanta sección como considere pertinente, como se muestra en la siguiente figura.



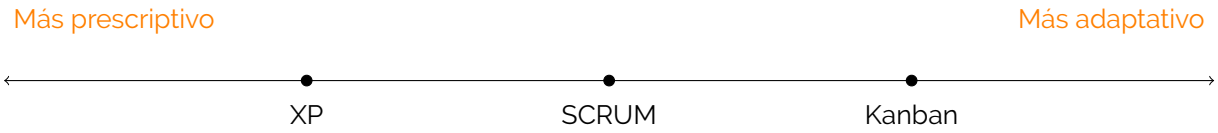
Dependiendo del marco de trabajo, varía la forma en la que se describen cada una de las tareas del tablero Kanban, por ejemplo, dentro de un marco de trabajo como Scrum cada una de las tareas se podría describir en el formato de historias de usuario.

3.3 Porgramación Extrema (XP)

XP es la abreviación comúnmente utilizada para referirse a Extreme Programming, que es un marco de desarrollo de software ágil que busca producir software de alta calidad en contextos con requisitos altamente cambiantes, riesgos que involucran tiempos fijo con tecnologías nuevas y equipos de trabajo pequeños ubicados en un mismo sitio.

Anunque aqui no especifiquemos en detaller XP podemos compararlo con respecto a SCRUM y Kanban observando cuántas reglas proporcionan. En el siguiente diagrama, **prescriptivo** significa *"más reglas que seguir"* y **adaptativo** significa *"menos reglas que seguir"*. Scrum y Kanban son ambos altamente adapt-

ables, pero en términos relativos, Scrum es más prescriptivo que Kanban y a su vez Scrum es más adaptativo que XP, esto significa que XP posse más procesos y artefactor que Scrum. Por su parte, Scrum proporciona más restricciones que Kanban, y por lo tanto deja menos opciones abiertas. Por ejemplo, Scrum prescribe el uso de iteraciones con límite de tiempo, mientras que Kanban no lo hace Kniberg and Skarin [2010].



3.4 Otros marcos ágiles

RAD (Desarrollo Rápido de Aplicaciones) es otra metodología de desarrollo de software ágil que se centra en el desarrollo rápido de aplicaciones mediante la realización de iteraciones frecuentes y realimentación constante y fue inventado por James Martin en 1991. El desarrollo guiado por pruebas (TDD) implica escribir una prueba automatizada primero, luego escribir solo la cantidad de código necesaria para que esa prueba pase, y posteriormente refactorizar el código principalmente para mejorar la legibilidad y eliminar la duplicación. Este proceso se repite una y otra vez.

4 Conclusiones

Es importante resaltar que cada marco es valioso y su adecuación depende del proyecto que se vaya a desarrollar. Ninguno es intrínsecamente malo; debemos escoger el que mejor se adapte a la dinámica del proyecto.

References

Henrik Kniberg. *Scrum and XP from the Trenches: How We Do Scrum*. InfoQ, 2015.

Henrik Kniberg and Mattias Skarin. *Kanban and Scrum: Making the Most of Both*. InfoQ, 2010.