Qué es la metodología XP (Extreme Programming)

El concepto Extreme Programming (XP) fue formalizado en 1999, cuando Kent Beck publicó el libro **Extreme Programming Explained**, conocido como '***The white book***'. Se considera como una metodología de desarrollo de software ágil, creada específicamente para promover la aplicación de prácticas de ingeniería apropiadas para la creación de software. Tiene como objetivo principal que un equipo de desarrollo pueda producir software de mejor calidad de forma constante y a su vez busca promover una buena calidad de vida para el equipo. *A principios de los años 2000, XP fue la metodología dominante en el mundo ágil, antes de que el framework Scrum se volviera tan popular*.

"No soy un gran programador; Solo soy un buen programador con buenos hábitos". - Kent Beck

A pesar de que XP promueve principalmente un enfoque de desarrollo de software basado en valores, principios y prácticas de ingeniería, cuando lo analizamos detenidamente (excluyendo las prácticas de ingeniería) podemos ver de es aplicable a cualquier equipo de trabajo, sea o no de software, porque su objetivo principal es promover buenas prácticas y herramientas para que un equipo pueda convertirse en un equipo de excelencia y alto rendimiento.

Valores, Principios y Prácticas de XP

Para poder aplicar XP en nuestro proceso de desarrollo es necesario entender los valores, principios y prácticas que lo componen. XP presenta 5 valores, 15 principios y 24 prácticas, en donde observamos que:

- *Los valores sin prácticas son difíciles de interiorizar* y es posible aplicarlos de diversas maneras, resultando difícil saber por dónde empezar.
- *Las prácticas sin valores son actividades de memoria sin un propósito* y hacerlas de forma individual sin ese propósito puede no brindar el resultado esperado.

Entre los valores y las prácticas, existe una gran distancia de comprensión y aplicación que es reducida por los principios. Los principios de XP tienen una relación directa con los valores, ya que tienen la intención de reflejarlos de maneras más concretas y práctica.

Valores de XP

- **Comunicación**: XP enfatiza la importancia del tipo apropiado de comunicación: discusión cara a cara con la ayuda de una pizarra blanca u otro mecanismo de dibujo.
- **Simplicidad**: La simplicidad busca responder la pregunta "¿qué es lo más simple que funcionará?" El propósito de esto es evitar el desperdicio y hacer solo cosas absolutamente necesarias, como hacer el trabajo lo más simple posible para que sea más fácil de mantener, respaldar y revisar.

- **Retroalimentación**: La retroalimentación continua permite un diseño simple. El equipo construye algo, recopila comentarios sobre su diseño e implementación, y ajusta el software en el futuro.
- **Coraje**: Kent Beck definió el coraje como "acción efectiva frente al miedo" (Libro Programación extrema explicada, página. 20). Se necesita valor para plantear problemas que reducen la eficacia de su equipo, se necesita valor para dejar de hacer algo que no funciona y probar otra cosa, etc.
- **Respeto**: Los miembros del equipo deben respetarse mutuamente, comunicarse entre sí, proporcionar y aceptar comentarios que beneficien su relación, y trabajar juntos para identificar diseños y soluciones simples.

"El desarrollo de software es un juego de comprensión, y la comprensión llega a la mente preparada, descansada y relajada". – Kent Beck

Principios de XP

De los 15 principios de XP, existen 5 llamados los principios core:

- **Realimentación rápida** (*Rapid feedback*): el equipo pide retroalimentación, la entiende y reacciona de inmediato de acuerdo a lo recibido.
- **Asumir simplicidad** (*Assume simplicity*): el equipo debe centrarse en el trabajo que es importante en el momento, lo que se ha planificado, lo que se ha comprometido como entregable. Tienen presente acrónimos como **YAGNI** = *You Ain't Gonna Need It* (No lo vas a necesitar) y **DRY** = *Don't Repeat Yourself* (No te repitas a ti mismo).
- **Cambio incremental** (*Incremental change*): aplicar pequeños cambios al software es mejor que aplicar grandes cambios hechos de una sola vez.
- **Abrazar el cambio** (*Embracing change*): Si un cliente piensa que un producto necesita ser cambiado, el equipo debe apoyar esta decisión y planificar cómo implementar los nuevos requisitos.
- **Trabajo de calidad** (*Quality work*): un equipo que trabaja bien hace un producto valioso, con calidad y se siente orgulloso de ello.

Los principios nos permiten tomar mejores decisiones cuando nos encontramos con distintas alternativas. Es mejor elegir una alternativa que cumpla con los principios de forma directa que una que no lo haga de la misma forma. Cada principio encarna uno o más valores. Un valor puede ser vago y difícil de aplicar, por ejemplo, algo que es "Simple" para una persona, para otra puede ser complejo "Complejo". Un principio es más concreto, u obtienes feedback rápidamente o no.

Los otros principios son:

- Enseñar a aprender
- Pequeña inversión inicial
- Juega a ganar
- Trabajar con los instintos de las personas, no en contra de ellos
- Responsabilidad aceptada

- Experimentos concretos
- Comunicación abierta y honesta
- Adaptación local
- Viaja con poco equipaje
- Medición honesta

En resumen, **los valores en conjunto con los principios promueven que la excelencia es un hábito, y que es algo que debemos llevar en nuestro ADN y cultura de equipo**.

Prácticas de XP

La programación extrema, además de los valores y principios, propone buenas prácticas de planificación, organización, comunicación y de ingeniería de software que, sumado a los valores y principios, permiten crear una cultura de equipo de excelencia. *Las prácticas de Extreme Programming (XP) más populares son:*

- **El Juego de Planificación** (*The Planning Game*): La idea principal de esta práctica es compartir las responsabilidades de planificación entre el equipo y el cliente, es decir que requiere una fuerte participación del cliente en el proceso de planificación. Se basa en un principio simple: los clientes generalmente tienen toda la información sobre el valor de los proyectos, y los equipos saben todo sobre su costo.
- **Pequeños Releases** (*Small Releases*): El equipo de desarrollo debe ser capaz de liberar versiones iterativas del sistema a los clientes con frecuencia.
- **Metáforas** (*Metaphor*): Debemos tener la capacidad de explicar el diseño del sistema a personas nuevas a través de metáforas en vez de pedirles que lean una gran cantidad de documentos.
- **Diseño Simple** (*Simple Design*): La regla es mantener las cosas, como su nombre indica, simples.
- **Desarrollo guiado por pruebas** (*TDD, Test Driven Development*): Es un enfoque evolutivo en la ingeniería de software que combina 2 prácticas que permiten crear código de calidad, pensar en la arquitectura del software que queremos desarrollar, escribiendo la prueba primero y luego mejorarla a través de la refactorización.
- **Refactorización** (*Refactoring*): Mejorar el diseño del código existente sin cambiar su comportamiento.
- **Programación en parejas** (*Pair Programming*): Dos programadores trabajando en pareja en una sola máquina, resolviendo el mismo problema.
- **Propiedad Colectiva del código** (*Collective Code Ownership*): Ningún miembro de todo el equipo posee una parte específica del código fuente, el código es de todos.
- **Integración Continua** (*Continuous Integration*): Práctica especialmente diseñada para construir o integrar todas las etapas de desarrollo, identificar errores y eliminarlos durante el proceso de desarrollo, reduciendo así el tiempo de respuesta y mejorando la calidad del software que se lanza como resultado.

- **Historias de Usuario** (*User Stories*): La historia es un pequeño documento de texto que escribe el cliente donde especifica en palabras simples lo que necesita que haga el software.
- **Estándares de Codificación** (*Coding Standard*): Su propósito es producir software que tenga un estilo consistente, independientemente del autor, lo que da como resultado un software que es más fácil de entender y mantener.

Roles de XP

Actualmente, los roles de XP no son tan populares como las prácticas, sin embargo, vale la pena conocerlos y entenderlos para comprender la metodología de programación extrema (XP) en su contexto general.

Un equipo de XP incluye seis roles:

- **El cliente** (*Customer*) es la persona responsable de escribir historias de usuarios, establecer prioridades y formular la cartera de productos.
- **El programador** (*Developer*) es un desarrollador normal, que escribe el código y realiza la totalidad de las tareas del proyecto.
- **El entrenador** (*Coach*) es la persona que vigila el trabajo del equipo, lo controla y enseña a sus miembros a implementar las prácticas más efectivas.
- **El rastreador** (*Tracker*) es la persona cuya tarea principal es monitorear el progreso del desarrollo del software y detectar todos los problemas en él.
- **El probador** (*Tester*) es el miembro del equipo responsable de la prueba del producto. La calidad del producto final depende en gran medida de su trabajo.
- **El pronosticador** (*Doomsayer*) es la persona que rastrea los riesgos del proyecto y advierte al equipo sobre ellos.

Aplicaciones de XP en la actualidad

Actualmente, XP es visto como un marco de trabajo, en vez de como una metodología cerrada, en la que hay que aplicar todo lo que se explica en el libro blanco. Hoy se reconocen sus valores, principios, prácticas y roles como un compendio de buenas prácticas de las que podemos utilizar solamente las que más ayudarán a nuestro equipo durante un proyecto específico.

Algunas prácticas de XP siguen evolucionando y son promovidas por las comunidades IT para la creación de productos de software exitosos, ya que consideran que son prácticas creadas por desarrolladores para mejorar la profesión y el arte del desarrollo de software.

Por ejemplo, es muy común ver que el equipo se organice basado en el marco de trabajo Scrum, siguiendo las reglas y valores que promueve este marco de trabajo y que a su vez apliquen prácticas de Ingeniería de Software que promueve XP en su proceso de desarrollo, como programación en parejas, integración continua, el uso de historias de usuario, etc.

Además, prácticas como la Integración continua y la entrega continua, han sido precursoras de movimientos como el de **DevOps** y el **desarrollo basado en arquitectura de Microservicios** que son tendencia en la actualidad y que en sus procesos integran diversas prácticas de XP como parte de ellas.

Es muy común que se hagan comparativas entre XP y otros marcos de trabajo ágiles, pero principalmente encontrarás "*extreme programming vs scrum*" o "*extreme programming vs kanban*", aunque en la realidad no son competencia el uno del otro, más bien se complementan muy bien cuando sabemos aplicar lo mejor de ambos mundos en nuestro proceso.

Conclusión

XP es un marco de desarrollo que persigue el feedback continuo, la excelencia técnica y la entrega de valor constante promovido a través de:

- *Valores que representan la mentalidad del equipo, incentivan el trabajo en equipo y el enfoque hacia un objetivo común.*
- *Principios que reflejan los valores de maneras más concretas.*
- *Prácticas que nos ayudan a conseguir una cultura de equipo y una buena organización a través de la excelencia técnica y la creación de software de valor.*

Cuando creamos una cultura de equipo basándonos en los principios y valores de XP podremos crear un entorno competitivo, pero a la vez motivacional, donde cada miembro aprecia el trabajo de cada uno de sus compañeros, entregan trabajo valioso de forma constante y rápidamente porque pueden distinguir las tareas relevantes de las que no son necesarias. Reaccionan rápidamente al feedback dándose cuenta de que es una crítica razonable que tiene el objetivo de hacer un mejor producto, trabajo y equipo, lo que promueve el espíritu kaizen, la mejora continua.