CONTENIDO

XP (Programación Extrema)

XP (del inglés "eXtremeProgramming") es una metodología ágil exclusiva para el desarrollo de software. Al igual que Scrum, considera que los cambios durante el proyecto serán frecuentes, tanto, que se puede llegar a trabajar en iteraciones de 1 sólo día, con entregas y despliegues de los resultados a diario, incluso en períodos más breves de tiempo. Contempla prácticas de ingeniería, valores, actividades y roles.

13 prácticas de ingeniería:

1. Equipo compacto
2. Juegos de planificación
3. Pruebas de usuario
4. Entregas (releases) pequeñas
5. Diseño simple
6. Programación por parejas
7. Refactorización
8. Desarrollo dirigido por las pruebas
9. Integración continua
10. Propiedad colectiva del código
11. Estándares de codificación
12. Metáfora del sistema
13. Ritmo sostenible

5 valores:

1. Simplicidad
2. Comunicación
3. Feedback
4. Motivación
5. Respeto

4 actividades:

1. Codificar
2. Probar
3. Escuchar

4. Diseñar

4 roles

1. Líder ágil o coach
2. Cliente
3. Programador (desarrollador)
4. Tester

Prácticas de ingeniería

Equipo compacto: Dentro de XP, el "cliente" no es el que paga la factura, sino el que realmente utiliza el sistema. XP dice que el cliente debe estar accesible en todo momento y disponible para preguntas. Por ejemplo, el equipo que desarrolla un sistema de administración financiera debe incluir o tener accesible un administrador financiero.

Juegos de planificación: El proceso principal de planificación dentro de la programación extrema se llama el Juego de Planificación. El proceso de planificación se divide en dos partes:

♣ Planificación de la reléase: establece que las historias de usuarios serán agrupadas para conformar una entrega y el orden de las mismas. Este cronograma será el resultado de una reunión entre todos los actores del proyecto.

♣ Planificación de la iteración: las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido.

Pruebas de usuario: Las pruebas de aceptación son propiedad del Cliente y definidos por dicha figura. Debemos incluir un punto como "pasar las pruebas del cliente" en la definición de "hecho".

Entregas (releases) pequeñas: La entrega se realiza a través de frecuentes lanzamientos de funcionalidad en producción creando valor real de negocio. Los pequeños lanzamientos ayudan al cliente a ganar confianza en el progreso del proyecto. El cliente ahora puede llegar a sus propuestas futuras sobre el proyecto basado en la experiencia real, sobre un producto tangible, usable (no sobre documentación, como ocurre en los proyectos no ágiles).

Diseño simple: Los programadores deben tomar un enfoque al diseño del software del tipo "lo simple siempre es lo mejor". Cada vez que se escribe un nuevo código, el autor debería preguntarse si existe

una manera más sencilla de introducir la misma funcionalidad. Si la respuesta es sí, la solución más simple debe ser elegida.

Programación por parejas: También llamada "programación por pares". Todo el código es producido por dos personas que programan en una estación de trabajo (un PC). Un programador tiene control sobre la estación de trabajo y está pensando sobre todo en la codificación en detalle. El otro programador está más centrado en el panorama general y está continuamente revisando el código que está siendo producido por el primer programador. Los programadores intercambian estos papeles varias veces al día.

Refactorización: Refactorización es el proceso de cambiar un sistema de software de tal manera que no altera el comportamiento externo del código, pero mejora su estructura interna.

Desarrollo dirigido por las pruebas: Dentro de XP, las pruebas unitarias se escriben antes de codificar el código final. Este enfoque pretende estimular al programador a pensar en las condiciones en las que su código podría fallar. Una sección de código fuente está terminada ("done", o hecha) cuando el programador no puede llegar a cualquier otra condición en la que el código pueda fallar.

Integración continua: El equipo de desarrollo siempre debe estar trabajando en la última versión del software. Los miembros del equipo deben intentar cargar su versión actual en el repositorio de código cada pocas horas. La integración continua evitará retrasos más adelante en el ciclo del proyecto, causado por problemas de integración.

Propiedad colectiva del código: Todo el mundo es responsable de todo el código; esto, a su vez, significa que todo el mundo está autorizado a cambiar cualquier parte del código. Acelera el proceso de desarrollo, porque si se produce un error en el código cualquier programador puede arreglarlo, pero existe el riesgo de errores introducidos por los programadores que no prevén ciertas dependencias. Las pruebas unitarias bien definidas solucionan este problema: si las dependencias imprevistas crean errores, cuando se ejecuten pruebas unitarias, se mostrarán fallos.

Estándares de codificación: El estándar de codificación es un conjunto acordado de reglas que todo el equipo de desarrollo acuerda seguir durante todo el proyecto. El estándar especifica un estilo y un formato consistentes para el código fuente, dentro del lenguaje de programación escogido, así como

varias construcciones y patrones de programación que se deben evitar para reducir la probabilidad de defectos.

Metáfora del sistema: La metáfora del sistema es una historia que todo el mundo - clientes, programadores y administradores - puede decir acerca de cómo funciona el sistema. Es un concepto de nomenclatura para las clases y los métodos que deberían facilitar a un miembro del equipo adivinar la funcionalidad de una clase / método particular, sólo a partir de su nombre. Para cada clase u operación, la funcionalidad es obvia para todo el equipo.

Ritmo sostenible: El concepto es que los programadores o desarrolladores de software no deben trabajar más de 40 horas por semana, y si hay horas extras una semana, que la próxima semana no debe incluir más horas extras. Este concepto incluye la idea de que las personas realizan mejor y más creativamente su trabajo si están descansados.

Valores en XP

Simplicidad: Recordemos que el diseño simple es una de las premisas que se deben seguir. También simplicidad en el código, puesto que se aplica la refactorización. Y simplicidad en la documentación: si el código está bien comentado, el diseño se mantiene simple, y está refactorizado, no será necesario escribir documentación adicional para su comprensión.

Comunicación: El código simple y bien documento comunica a los programadores muy bien qué funcionalidad persigue. Las pruebas unitarias comunican por qué se ha elegido ese diseño para el código, y desde luego si funciona correctamente. Y la comunicación con el cliente es fluida, porque éste forma parte del equipo de trabajo.

Feedback: La retroalimentación por parte del cliente es inmediata, puesto que forma parte del equipo de trabajo. Esto se refuerza porque las entregas se realizan continuamente en periodos muy cortos de tiempo. Las pruebas dan feedback en tiempo real sobre el estado de salud del código.

Motivación: Motivación y valentía para centrarse sólo en lo necesario para la entrega de hoy, y no para la de mañana. Hay que cumplir entregas todo el tiempo, aunque sean pequeñas. Hay que estar motivado para dar la mejor solución simple para la siguiente entrega. Y ser persistente para mantener el ritmo sostenible pero necesario para cumplir con las entregas.

Respeto: Los programadores se respetan mutuamente porque nadie puede escribir código que haga fallar código o pruebas de otra persona. El cliente respeta a los programadores, y los programadores al cliente, puesto que su comunicación es fluida, constante y debe haber entendimiento entre todos.

Actividades en XP

Codificar: El código se construye siguiendo las prácticas de ingeniería recomendadas (programación por parejas, refactorización, estándar de codificación, etc.)

Probar: Es fundamental que el desarrollo sea dirigido por las pruebas, por lo tanto codificar y probar son actividades que van de la mano.

Escuchar: Puesto que se trabaja por parejas, y también conjuntamente con el cliente, las personas del equipo tienen que aprender a escuchar a los demás y a compartir su información y opinión profesional.

Diseñar: Se contempla en todo momento el diseño simple, la refactorización, el respeto al estándar de codificación, la documentación en el código... estas actividades forman parte del diseño del software.

Roles en XP

Líder ágil o coach: Responsable del proceso global, conocedor de la metodología, y facilitador del trabajo. Guía a los miembros del equipo para seguir el proceso correctamente.

Cliente: Escribe los requisitos (las historias de usuario) y comprueba los criterios de aceptación de los mismos. Decide la prioridad, según criterios de negocio, de los requisitos para su implementación y entrega.

Programador (desarrollador): Construye el código y las pruebas unitarias que resuelven los requisitos. En este rol se incluirían todos los perfiles necesarios para ello (programador, analistas programadores, diseñador, maquetador.... excepto pruebas funcionales).

Tester: Ayuda al cliente a escribir las pruebas funcionales. Gestiona y utiliza las herramientas para las pruebas funcionales.