PROGRAMACIÓN EXTREMA (XP)
EXTREME PROGRAMMING (XP)

HISTORIA

La programación extrema o eXtreme Programming (XP) es un enfoque de la ingeniería de software formulado por Kent Beck, autor del primer libro sobre la materia, Extreme Programming Explained: Embrace Change (1999). Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

INTRODUCCION

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

¿QUÉ ES PROGRAMACIÓN EXTREMA O XP?

- ✓ Metodología liviana de desarrollo de software
- ✓ Conjunto de practicas y reglas empleadas para desarrollar software
- ✓ Basada en diferentes ideas acerca de cómo enfrentar ambientes muy cambiantes
- ✓ Originada en el proyecto C3 para Chrysler

- ✓ En vez de planificar, analizar y diseñar para el futuro distante, hacer todo esto un poco cada vez, a través de todo el proceso de desarrollo

OBJETIVOS.

- ✓ Establecer las mejores prácticas de Ingeniería de Software en los desarrollo de proyectos.
- ✓ Mejorar la productividad de los proyectos.
- ✓ Garantizar la Calidad del Software desarrollando, haciendo que este supere las expectativas del cliente.

CONTEXTO XP

- ✓ Cliente bien definido
- ✓ Los requisitos pueden (y van a) cambiar
- ✓ Grupo pequeño y muy integrado (máximo 12 personas
- ✓ Equipo con formación elevada y capacidad de aprender

CARACTERÍSTICAS XP

- ✓ Metodología basada en prueba y error
- ✓ Fundamentada en Valores y Prácticas
- ✓ Expresada en forma de 12 Prácticas–Conjunto completo–Se soportan unas a otras–Son conocidas desde hace tiempo. La novedad es juntarlas

VALORES XP

- ✓ Simplicidad XP propone el principio de hacer la cosa más simple que pueda funcionar, en relación al proceso y la codificación. Es mejor hacer hoy algo simple, que hacerlo complicado y probablemente nunca usarlo mañana.

- ✓ Comunicación Algunos problemas en los proyectos tienen origen en que alguien no dijo algo importante en algún momento. XP hace casi imposible la falta de comunicación.

- ✓ Realimentación Retralimentación concreta y frecuente del cliente, del equipo y de los usuarios finales da una mayor oportunidad de dirigir el esfuerzo eficientemente.

- ✓ Coraje El coraje (valor) existe en el contexto de los otros 3 valores.(si funciona…mejóralo)

EL ESTILO XP

- ✓ Esta orientada hacia quien produce y usa el software
- ✓ Reduce el costo del cambio en todas las etapas del ciclo de vida del sistema.
- ✓ Combina las que han demostrado ser las mejores practicas para desarrollar software, y las lleva al extremo.

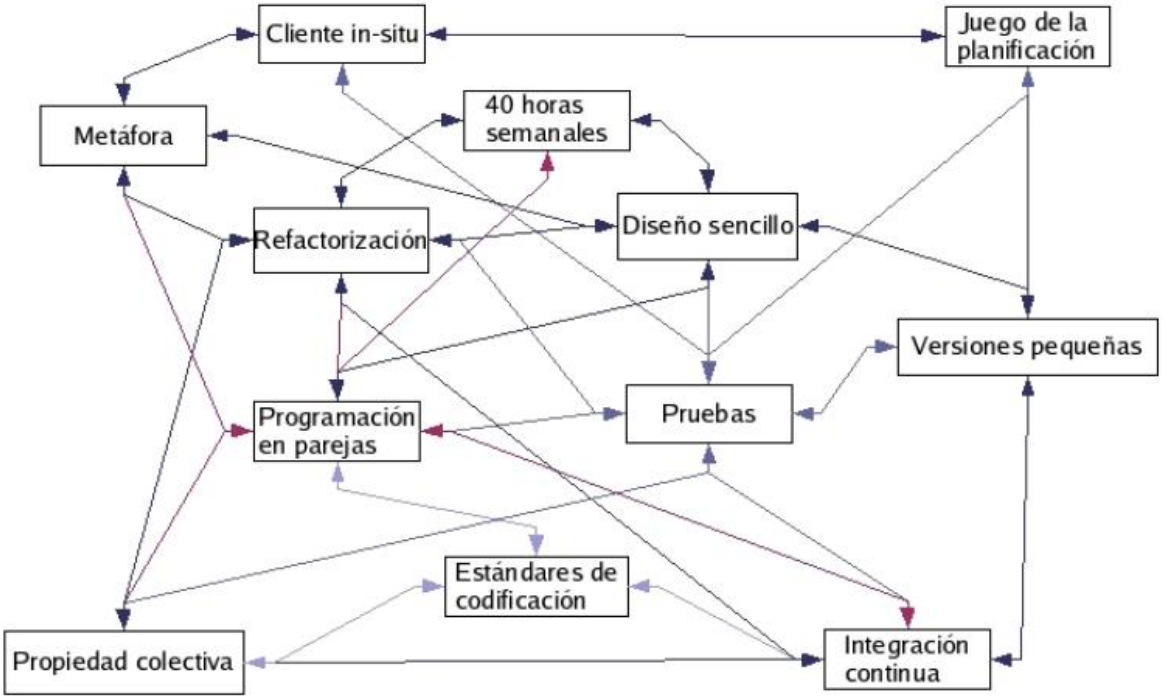
PRÁCTICAS BÁSICAS DE LA PROGRAMACIÓN EXTREMA

Para que todo esto funcione, la programación extrema se basa en doce "prácticas básicas" que deben seguirse al pie de la letra. Dichas prácticas están definidas (en perfecto inglés) en www.xprogramming.com/xpmag/whatisxp.htm. Aquí tienes un pequeño resumen de ellas.

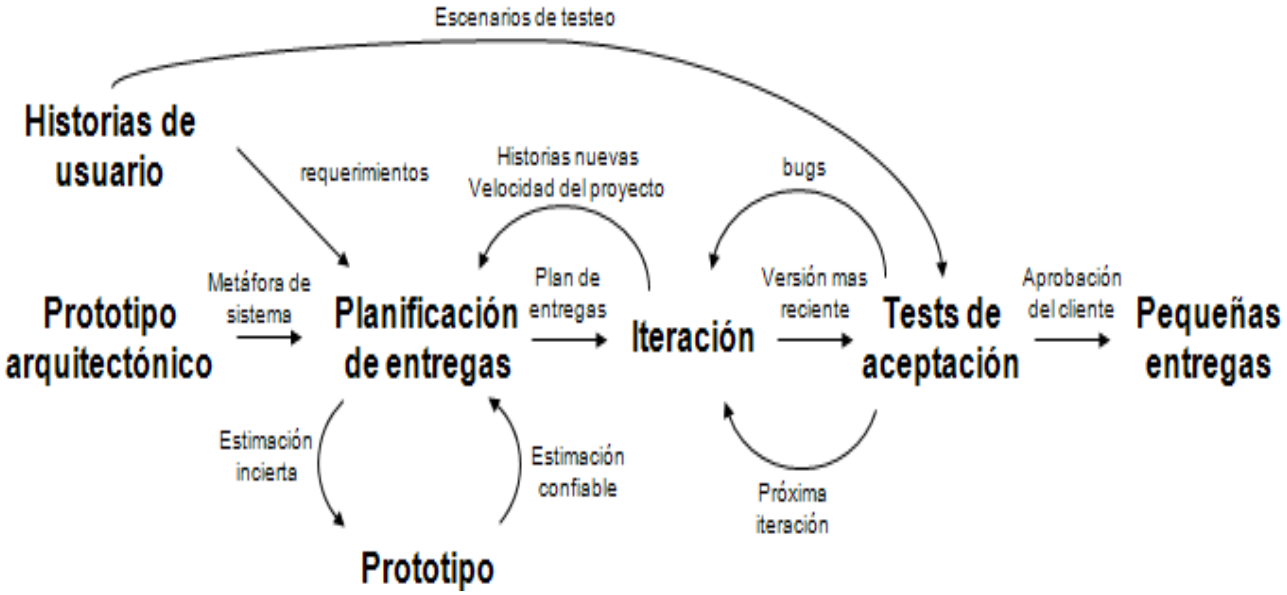
- ✓ **Equipo completo:** Forman parte del equipo todas las personas que tienen algo que ver con el proyecto, incluido el cliente y el responsable del proyecto.
- ✓ **Planificación:** Se hacen las historias de usuario y se planifica en qué orden se van a hacer y las mini-versiones. La planificación se revisa continuamente.
- ✓ **Test del cliente:** El cliente, con la ayuda de los desarrolladores, propone sus propias pruebas para validar las mini-versiones.
- ✓ **Versiones pequeñas:** Las mini-versiones deben ser lo suficientemente pequeñas como para poder hacer una cada pocas semanas. Deben ser versiones que ofrezcan algo útil al usuario final y no trozos de código que no pueda ver funcionando.
- ✓ **Diseño simple:** Hacer siempre lo mínimo imprescindible de la forma más sencilla posible. Mantener siempre sencillo el código.
- ✓ **Pareja de programadores:** Los programadores trabajan por parejas (dos delante del mismo ordenador) y se intercambian las parejas con frecuencia (un cambio diario).

- ✓ **Desarrollo guiado por las pruebas automáticas:** Se deben realizar programas de prueba automática y deben ejecutarse con mucha frecuencia. Cuantas más pruebas se hagan, mejor.
- ✓ **Integración continua:** Deben tenerse siempre un ejecutable del proyecto que funcione y en cuanto se tenga una nueva pequeña funcionalidad, debe recompilarse y probarse. Es un error mantener una versión congelada dos meses mientras se hacen mejoras y luego integrarlas todas de golpe. Cuando falle algo, no se sabe qué es lo que falla de todo lo que hemos metido.
- ✓ **El código es de todos:** Cualquiera puede y debe tocar y conocer cualquier parte del código. Para eso se hacen las pruebas automáticas.
- ✓ **Normas de codificación:** Debe haber un estilo común de codificación (no importa cual), de forma que parezca que ha sido realizado por una única persona.
- ✓ **Metáforas:** Hay que buscar unas frases o nombres que definan cómo funcionan las distintas partes del programa, de forma que sólo con los nombres se pueda uno hacer una idea de qué es lo que hace cada parte del programa. Un ejemplo claro es el "recolector de basura" de java. Ayuda a que todos los programadores (y el cliente) sepan de qué estamos hablando y que no haya mal entendidos.
- ✓ **Ritmo sostenible:** Se debe trabajar a un ritmo que se pueda mantener indefinidamente. Esto quiere decir que no debe haber días muertos en que no se sabe qué hacer y que no se deben hacer un exceso de horas otros días. Al tener claro semana a semana lo que debe hacerse, hay que trabajar duro en ello para conseguir el objetivo cercano de terminar una historia de usuario o mini-versión.

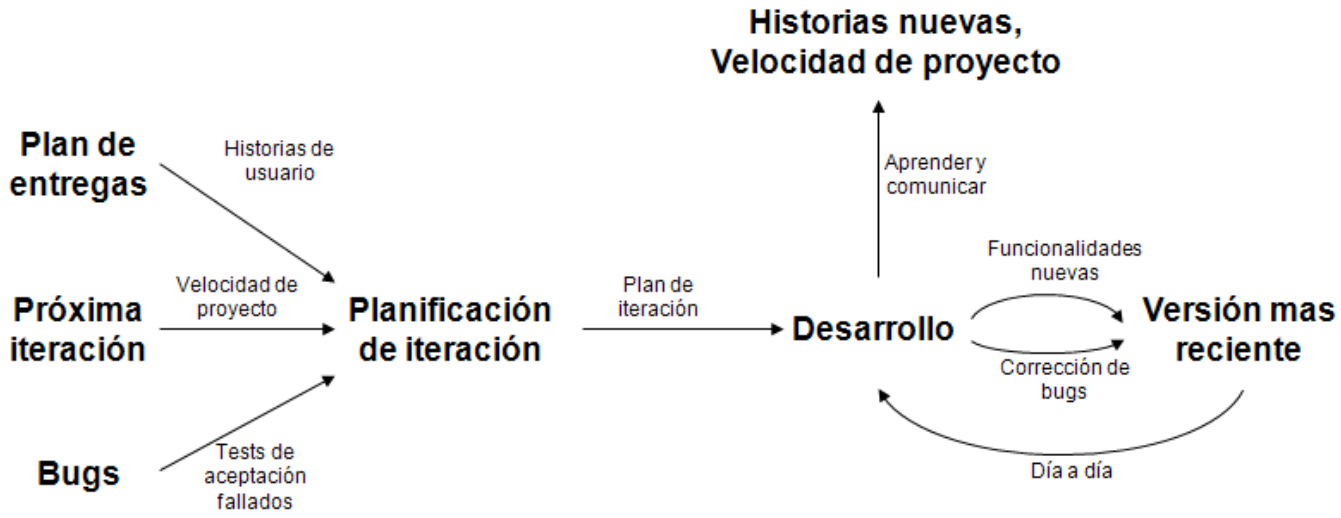
LAS PRACTICAS SE REFUERZAN



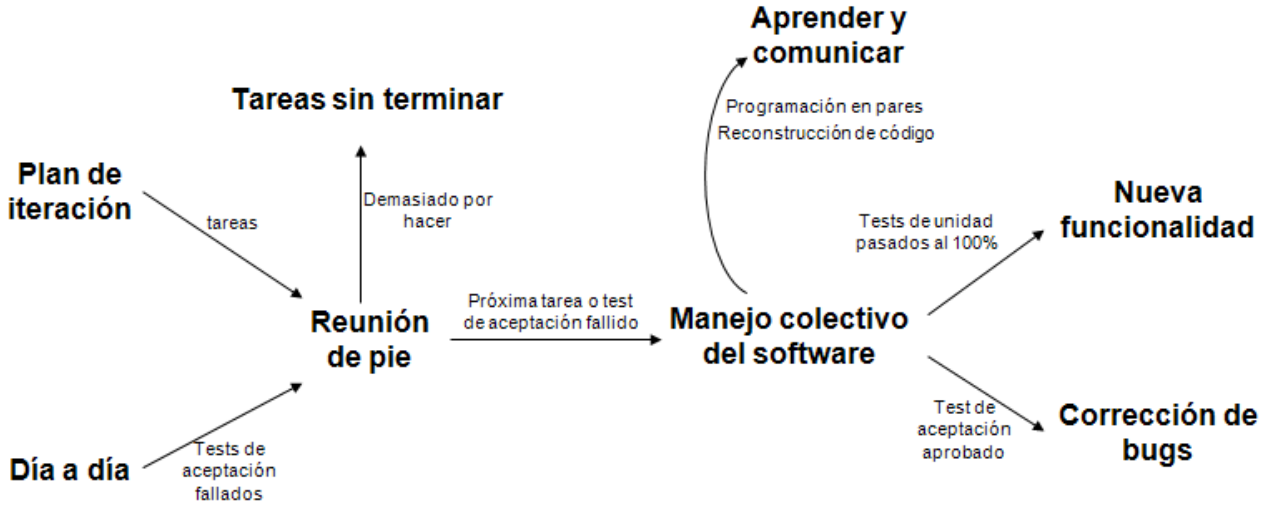
UN PROYECTO XP



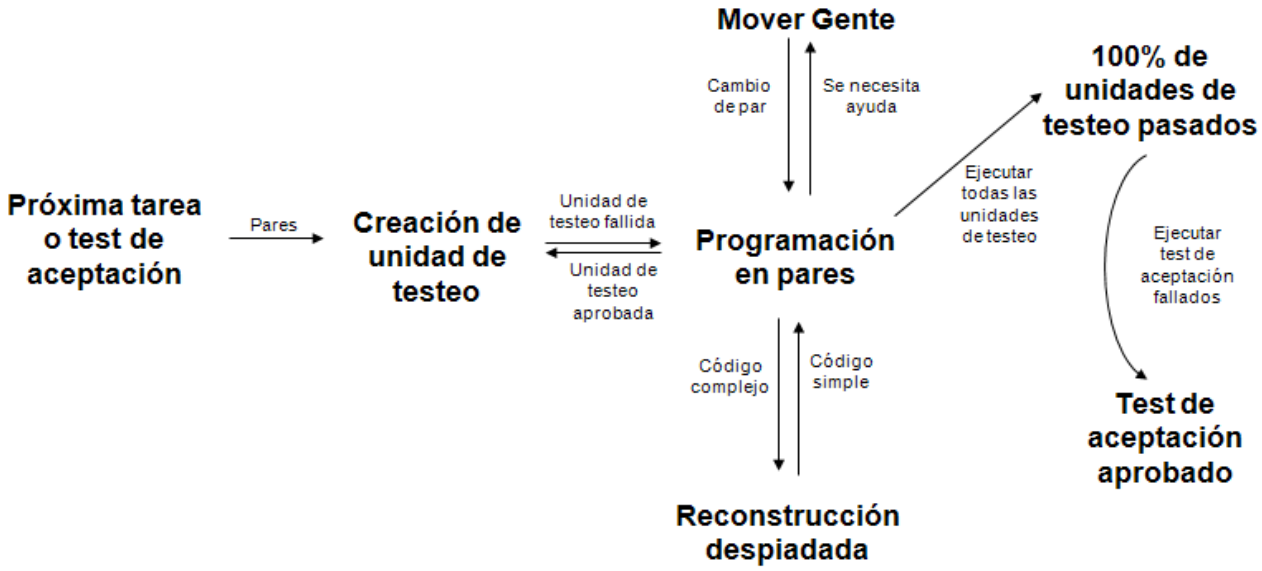
ITERACIÓN



DESARROLLO



MANEJO COLECTIVO DEL CÓDIGO



VENTAJAS Y DESVENTAJAS DE EXTREME PROGRAMMING

Ventajas:

- ✓ Programación organizada.
- ✓ Menor taza de errores.
- ✓ Satisfacción del programador.

Desventajas:

- ✓ Es recomendable emplearlo solo en proyectos a corto plazo.
- ✓ Altas comisiones en caso de fallar.

CONCLUSIONES

- ✓ Apostolado de metodologías exitosas
- ✓ Aporte de la experiencia práctica a los modelos teóricos
- ✓ Enfoque de conjunto de prácticas como rompecabezas
- ✓ Tecnología en expansión
- ✓ Importancia de revisitar las metodologías desde la experiencia práctica