X P

Programación Extrema (XP: eXtreme Programming)

Ing. Ramón Roque Hernández, M.C.
ramonroque@yahoo.com
http://www.geocities.com/ramonroque

1

X P

Antecedentes: Cómo surgen las metodologías ágiles

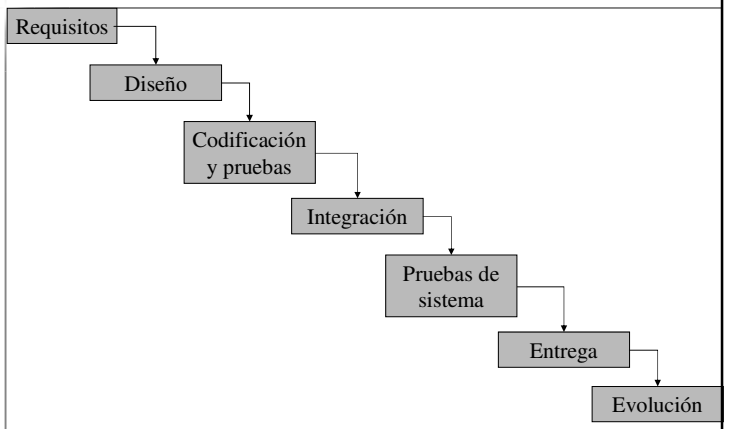
2

Concepto de Metodología de Desarrollo de Software

- Una metodología de desarrollo es una colección de procedimientos, técnicas, principios y herramientas que ayudan a construir sistemas computacionales.

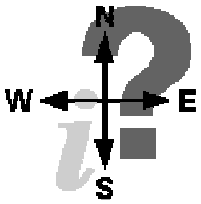
3

Metodología tradicional: Ciclo de desarrollo en cascada



Ciclo de desarrollo en cascada (Secuencial)

- Se sigue siempre una secuencia ordenada de fases.
- Los cambios después de las primeras fases vuelven el proceso difícil e incómodo.
- Entre mas tarde se detectan los cambios, mas caros resultan éstos.
- Produce demasiada documentación
- Requiere que una etapa termine antes de iniciar con la siguiente.
- Implica requerimientos fijos, conocidos claramente desde el principio.



5

Problemas con los modelos secuenciales

Potencializan los efectos negativos de los errores y los cambios provocando que su detección sea tardía.

Contribuyen a incrementar los costos de corrección y codificación.

Presentan procesos burocráticos que retrasan el ritmo del proyecto, y consumen tiempo y recursos en su extensa documentación.

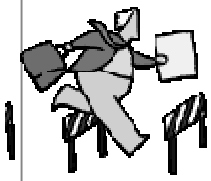
6

Situación real del entorno de desarrollo

- En la vida real, en un proyecto de software existe riesgo que aparece en forma de errores y de cambios, los cuales se deben a la evolución del entorno del usuario.
- En muchas ocasiones esa evolución sucede tan rápidamente que los cambios se producen durante el propio desarrollo del proyecto.
- Los modelos secuenciales **NO** proporcionan un soporte adecuado para estos entornos evolutivos.

7

Metodologías Agiles



- Nacen como una alternativa a los antiguos modelos.
- Buscan un punto medio entre "ningún proceso" y "demasiado proceso" proporcionando simplemente "suficiente proceso".
- Las metodologías ágiles están menos orientadas a la documentación, y mas centradas en la codificación; son completamente orientadas a la gente y no a los procesos; y son adaptables en lugar de predictivas.

8

Metodologías Agiles

- Están enfocadas al cliente.
- Son adaptables al cambio.
- No exigen muchos requerimientos de documentación.
- Comenzaron a emerger en la década de los noventa.
- Son propuestas para desarrollar sistemas rápidamente en ambientes cambiantes, manteniendo la calidad y el control de costos.
- Existen muchas metodologías ágiles.
- Cada una es diferente pero todas poseen principios comunes.

9

Manifiesto para desarrollo ágil de software

- Documento que reúne los principios de toda metodología "Agil". Algunos de ellos son:
  - Satisfacer al cliente entregándole software valioso de manera oportuna y continua (el periodo de entrega puede ser entre 2 semanas y 2 meses).
  - Aceptar los cambios tardíos en los requerimientos
  - Fomentar el trabajo conjunto del personal de desarrollo con el cliente.
  - Promover la motivación, comunicación, calidad y simplicidad.

http://agilemanifesto.org

10

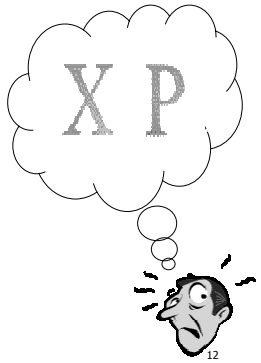
X P

Programación Extrema: Una metodología ágil

11

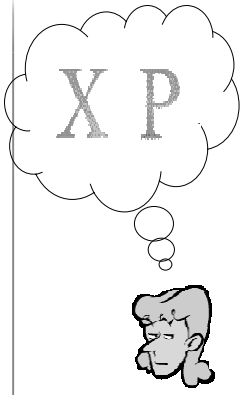
Programación Extrema (XP)

- La XP es una metodología ágil que ha tenido mucha aceptación entre algunos desarrolladores.
- Ha sido cuestionada por otros tantos.
- XP reúne elementos valiosos ya conocidos en ingeniería de software, administración y relaciones humanas.
- Solo los interrelaciona con el objetivo de desarrollar exitosamente sistemas en un entorno cambiante.



12

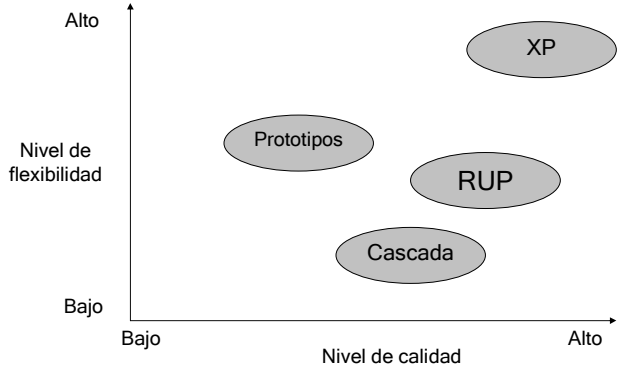
Programación Extrema (XP)



- Creada en 1996 por Kent Beck.
- Es una disciplina eficiente, de bajo riesgo, flexible, predecible, científica y divertida para desarrollar software.
- Es ideal para equipos pequeños o medianos que desarrollan software en un ambiente de requerimientos cambiantes.

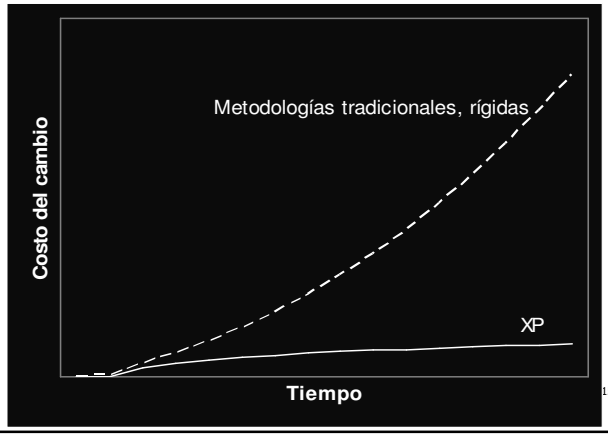
13

Comparación de XP con otras metodologías



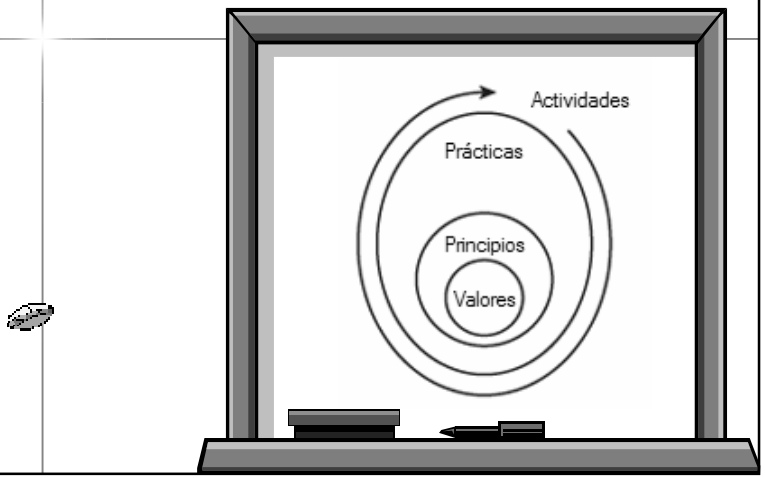
14

XP vs. Metodologías rígidas



15

Elementos de la XP



Valores de la XP

- Comunicación ← Principal elemento
- Sencillez
- Retroalimentación
- Coraje (Valor)

17

Principios básicos de la XP

- Obtener retroalimentación rápida acerca del programa.
- Adoptar SIEMPRE soluciones sencillas, pensando en el HOY.
- Preferir SIEMPRE pequeños cambios graduales a un solo cambio abrupto.
- Aceptar el cambio.
- Hacer siempre Trabajo de Calidad.

1

Principios secundarios de la XP

- Enseñar a aprender
- Iniciar con una pequeña inversión
- Jugar para ganar, en lugar de Jugar para NO perder
- Realizar experimentos concretos
- Comunicación abierta y sincera
- Trabajar con los instintos de las personas no contra ellos
- Responsabilidad Aceptada
- Adaptar XP a las necesidades particulares
- Viajar con poco equipaje
- Utilizar métricas sinceras

2

Actividades que fomenta la XP

Codificar

Diseñar

Escuchar

Probar

20

Prácticas de XP

- Juego de planeación
- Versiones pequeñas
- Metáfora
- Diseño simple
- Pruebas
- Refactorización
- Programación por pares
- Propiedad colectiva de código
- Integración continua
- 40 horas de trabajo por semana
- Cliente en el lugar de desarrollo
- Usar estándares de codificación

21

Prácticas de XP: Juego de planeación con el cliente

- Es una reunión entre clientes y desarrolladores.
- El cliente escribe "Historias de usuario" [Cosas que el cliente quiere que el sistema haga].
- Se usan tarjetas blancas de archivo.
- Se dividen las historias grandes en varias pequeñas.
- Los programadores estiman tiempos de desarrollo.
- El cliente ordena las historias por su valor de negocios
- Los programadores las ordenan por su claridad para estimarles tiempo.
- Se acuerda cuales historias se incluirán en el desarrollo y su orden de implementación.

22

Prácticas de XP: Juego de planeación entre programadores

- También se utilizan tarjetas de archivo
- Los programadores toman una "Historia de usuario" y la dividen en tareas.
- Se dividen tareas grandes en varias mas pequeñas.
- Cada programador se "apunta" para sus tareas y estima tiempo para cada una.

23

Prácticas de XP: Versiones Pequeñas

- Se trata de implementar la menor cantidad de historias de usuario a la vez.

- Cada versión debe tener sentido propio y debe entregar valor de negocios para el cliente.
- En teoría, usando XP se obtienen sistemas funcionales desde la primera versión.

24

Prácticas de XP: Metáfora

- Permite establecer un conjunto común de términos para la comunicación en el proyecto.
- Unifica el lenguaje técnico y de negocios, y ayuda a entender los elementos básicos del proyecto y las relaciones entre ellos.

- Una metáfora hace que la tecnología sea mejor comprendida por el cliente.

Prácticas de XP: Diseño Simple

- El diseño debe ser:
  - El mas simple posible.
  - Consistente con las metas del proyecto.
  - Fácil de comunicar y entender.
  - Sujeto a cambios en todo momento.
  - Completo pero contener solo lo necesario y nada mas.
  - Realizado evitando la complejidad innecesaria.
- Un programa con diseño simple:
  - Pasa todas las pruebas.
  - No contiene código duplicado.
  - Refleja claramente lo que el programador planeó hacer .
  - Contiene la menor cantidad de clases y métodos.

Prácticas de XP: Pruebas

Los programadores escriben las pruebas **antes** de realizar el código a probar.

Los usuarios realizan pruebas **funcionales** para el programa.

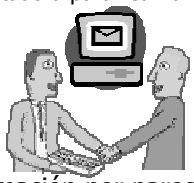
- Las pruebas:
  - Deben ser escritas y conservadas para todas aquellas situaciones que puedan desencadenar un error.
  - No deben interactuar con otras.
  - Deben realizarse con un entorno automatizado especial.

Prácticas de XP: Refactorización

Es la técnica de mejorar el código sin cambiar su funcionalidad con el objetivo de eliminar duplicidad, hacerlo mas entendible, simplificarlo o hacerlo mas flexible.

Prácticas de XP: Programación por Pares

Dos programadores comparten
UNA computadora para realizar su trabajo.



La Programación por pares fomenta:

- Que todas las decisiones de diseño sean tomadas por dos personas.
- Que por lo menos dos personas estén siempre involucradas con esa parte del sistema.
- El intercambio de integrantes.
- La expansión del conocimiento con otras parejas en el equipo.

29

Con la Programación por pares...

- Se logra que la productividad sea mas alta que en el supuesto escenario donde cada programador hiciera su trabajo individualmente y luego se integraran ambos.
- El código resultante posee una mayor calidad

30

Prácticas de XP: Propiedad colectiva de código

Cualquier persona puede cambiar el código en cualquier momento.

Si una persona puede cambiar todo el código, entonces también es responsable de todo el código.

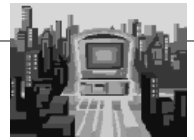
El código complejo es simplificado en corto tiempo por otros programadores.

Para evitar el caos, es necesario usarla en conjunto con otras prácticas como la programación por pares, pruebas y estándares de codificación.

31

Prácticas de XP: Integración continua

- Se deben integrar los componentes de software por lo menos una vez al día.
- Al integrar, se deben ejecutar nuevamente las pruebas y asegurarse que todas fueron superadas.
- Existen herramientas que facilitan este proceso (ej: ANT, NANT).



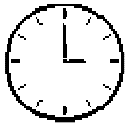
Existe una computadora especial para integrar el trabajo de todo el equipo de desarrollo.

Los conflictos entre el código de los diferentes programadores son visibles en corto tiempo.

32

Prácticas de XP: 40 horas de trabajo por semana

- Las pesadas cargas de trabajo:
  - Indican serios problemas en el proyecto.
  - Disminuyen la calidad de trabajo
- Se debe evitar trabajar Horas Extras por mas de una semana seguida.
- 40 Horas: Medida flexible a juicio de los programadores (35 hrs., 45 hrs., etc.) siempre y cuando se trabaje con calidad.



33

Prácticas de XP: Cliente en el lugar de desarrollo

- El cliente se encuentra en el mismo sitio donde el desarrollo se está llevando a cabo y debe permanecer ahí durante todo el ciclo de vida del proyecto

  - Esto garantiza su participación y retroalimentación en todas las actividades.
- El cliente puede realizar sus actividades de negocio cotidianas, siempre y cuando se encuentre separado de su lugar de trabajo habitual.

34

Prácticas de XP: Usar estándares de codificación

- Son las reglas, convenciones, recomendaciones y buenas prácticas de programación que deben ser adoptadas.
- Garantizan la producción de software con un estilo uniforme.

Los programadores decidirán cuales estándares adquirir y se comprometerán a aplicarlos en todo momento.

Con un poco de práctica, debe ser imposible saber quien del equipo programó cualquier código.

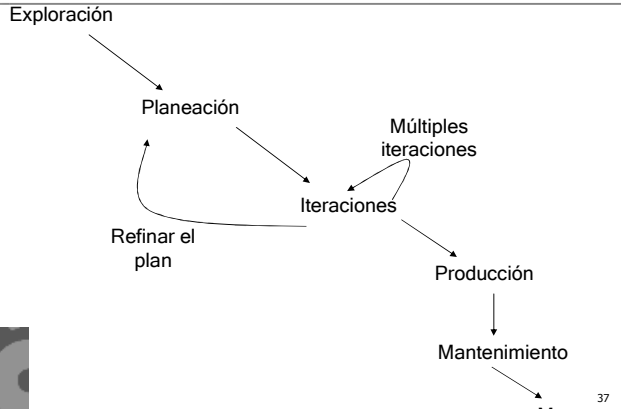
35

XP

Ciclo de vida de un proyecto en XP

36

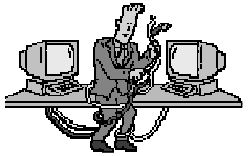
Ciclo de vida de un proyecto en XP



37

Ciclo de vida XP: Fase de Exploración

- **Propósito:** Descubrir los requerimientos del sistema
- **Resultado:** Metáfora y lista preliminar de "Historias de usuario".



Los programadores:
- Experimentan y aprenden toda la tecnología que usará el sistema.
- Exploran diferentes opciones para la arquitectura del sistema.

El cliente:
- Comunica de manera general lo que el sistema debe hacer.
- Practica las "Historias de usuario".

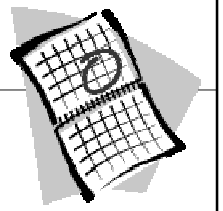
38

Ciclo de vida XP: Fase de Planeación

Definir las historias de usuario.	Estimar el tiempo de construcción de cada historia de usuario.
Decidir el valor de negocios de cada historia de usuario.	Prevenir al cliente de riesgos técnicos.
Decidir cuales historias serán incluidas en la entrega del sistema.	Medir el progreso del equipo.

39

Ciclo de vida XP: Fase de Iteraciones



Consiste en dividir el alcance total del sistema en iteraciones cortas (ciclos).

Cada iteración tiene una duración entre una y cuatro semanas.

La división en iteraciones se realiza en una junta a la que asisten desarrolladores y clientes.

Se debe celebrar con el equipo el fin de cada iteración. Tal vez con un brindis, o pizza, etc.

Después se divide la iteración en tareas de programación para implementar cada historia de usuario.

40

Ciclo de vida XP: Fase de Producción

Llegar a esta fase es motivo de celebración para el equipo.

El producto se verifica para ser utilizado en el ambiente de producción real (o simulado) del cliente.

Los usuarios realizan pruebas de aceptación. Si se detectan errores, se deben corregir. Hasta entonces la entrega se considerará completa.

En este punto, la meta NO es continuar con la evolución funcional, sino solo estabilizar el sistema realizando los ajustes necesarios.

Cualquier documentación realizada "Debe ser suficiente pero no mas".

41

Ciclo de vida XP: Fase de Mantenimiento

Es una fase progresiva con un plan para realizar las actualizaciones y cambios al sistema.

- Considerar:
  - Cambios que se realizan al código.
  - Los datos que se modifican.
  - Los usuarios conectados.
  - Tiempo y el personal dedicado al mantenimiento.

Mantenimiento es el estado normal de un proyecto en XP.

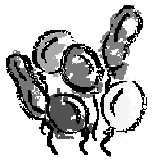
Debe existir un ambiente de confianza para realizar de manera segura y sencilla las modificaciones.

42

Ciclo de vida XP: Fase de Muerte del proyecto

- "Morir bien es tan importante como vivir bien. Esto se aplica tanto a la XP como a las personas"

Esta fase deja aprendizaje al equipo de desarrollo y debe celebrarse.



Mejor razón

La muerte del proyecto puede darse debido a que:

- El usuario ya no tiene mas historias para implementar.
- El sistema ya no es factible.
- No hay presupuesto para continuar.
- Los defectos son tantos que no conviene seguir adelante.

43