

XP- EXTREME PROGRAMMING

RUBBY CASALLAS

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

FACULTAD DE INGENIERÍA

UNIVERSIDAD DE LOS ANDES

Agenda

- Qué es XP?
- 12 Prácticas
- Actividades Principales:
 - ▣ Planeación
 - ▣ Diseño
 - ▣ Codificación
 - ▣ Testing
- Conclusiones
- Referencias

¿Qué es XP?

- Extreme Programming, o XP, es un proceso de software liviano (lightweight process) (pocas reglas, pocas prácticas)
- Basado en los siguientes principios:
 - ▣ **Simplicidad**: buenos diseñadores (eXtreme good designers)
 - ▣ **Comunicación**: Trabajo en equipo (eXtreme teamwork)
 - ▣ **Feedback**: Participación del cliente (eXtreme customer participation)
 - ▣ **Coraje**: Iteraciones, Refactoring, Testing (eXtreme iterations, eXtreme refactoring, eXtreme testing)

¿Qué es XP? (2)

- XP está diseñado para ser usado en pequeños equipos de desarrollo.

Las 12 Practicas

1. El Proceso de Planeación:

- ▣ Permite a los clientes definir valor de negocio a los requerimientos.
- ▣ Se usan estimados de los programadores
- ▣ Se selecciona cuáles aspectos serán realizados y cuáles pospuestos.

2. Pequeños releases:

- ▣ Poner en producción un sistema simple desde el comienzo
- ▣ Actualizarlo frecuentemente en ciclos muy cortos

Las 12 Practicas(2)

3. Metáfora del sistema:

- ▣ Usar un sistema de nombres y una descripción común.
- ▣ Vocabulario común

Las 12 Practicas(2)

4. Diseño simple.

1. Un programa construido con XP debe ser el más simple que satisfaga los requerimientos.
2. No se desarrolla “para el futuro”. Se hace énfasis en lo que tiene valor para el cliente.

5. Testing.

- ▣ Programadores desarrollan software escribiendo primero las pruebas.
- ▣ Clientes proveen pruebas de aceptación para asegurarse que los aspectos que ellos requieren son provistos por el software.

Las 12 Practicas(3)

6. Refactoring.

- ▣ Equipos XP mejoran constantemente el diseño del sistema haciendo refactoring.
- ▣ Esfuerzo por mantener el sistema sin código duplicado, simple, cohesivo, etc.

Las 12 Practicas(4)

7. Pair Programming.

- ▣ Codificación en parejas, dos programadores en la misma máquina.

8. Propiedad colectiva del código

9. Integración continua

- ▣ Equipos XP integran y construyen el sistema múltiples veces por día.

Las 12 Practicas(5)

10. 40-horas por semana

- ▣ Programadores cansados cometen más errores.
- ▣ Programadores XP no trabajan tiempo excesivo durante largos periodos, ellos se mantienen frescos, saludables y efectivos.

11. El cliente en el sitio de trabajo de los desarrolladores

- ▣ EL cliente trabaja a la par con los desarrolladores determinando los requerimientos, prioridades y respondiendo preguntas.

Las 12 Practicas(5)

12. Estándar de codificación

- ▣ Todos los programadores deben escribir y documentar el código en la misma manera.

Planeación

- El proyecto está dividido en iteraciones
 - ▣ El resultado de una iteración es un pequeño release
- Cada iteración tiene su propio plan
- El calendario para el release está basado en:
 - ▣ Historias de usuario
 - ▣ Velocidad del proyecto

Planeación: Historias de usuario

- Son escritas por los clientes y usuarios como cosas que ellos necesitan que el sistema haga
- Propósito:
 - ▣ Crear estimados de tiempo de desarrollo
 - ▣ Conducir la creación de las pruebas de aceptación

Planeación: Reunión de la planeación del release

- El propósito es crear el plan del release
- El plan del release se usa para crear el plan de la iteración
- El plan del release tiene un conjunto de reglas para negociar el cronograma de cada individuo y negociar los compromisos.
- Los estimados para el desarrollo de cada historia, se hacen en términos de semanas ideales de trabajo

Planeación: Reunión de la planeación del release

- Hay cuatro variables para cuantificar el proyecto:
 - ▣ Alcance: cuánto será hecho
 - ▣ Recursos: cuánta gente hay disponible
 - ▣ Tiempo: cuanto tiempo se tiene disponible para el release
 - ▣ Calidad: qué tan bueno se requiere que sea el software y que tantas pruebas debe tener.

Planeación: Reunión de la planeación del release

- Hay cuatro variables para cuantificar el proyecto:
 - ▣ Alcance: cuánto será hecho
 - ▣ Recursos: cuánta gente hay disponible
 - ▣ Tiempo: cuanto tiempo se tiene disponible para el release
 - ▣ Calidad: qué tan bueno se requiere que sea el software y que tantas pruebas debe tener.

Planeación: Reunión de la planeación del release

- Los clientes definen las prioridades
 - ▣ Se define el alcance de la iteración
 - ▣ Se detalla el cronograma
 - ▣ Se estima el tiempo de desarrollo usando la métrica de velocidad del proyecto

Planeación : Rotar a la gente, cambiar de roles

- Un equipo es más flexible si todos saben lo suficiente de todas las partes del sistema y pueden trabajar sobre ellas
- La estrategia es rotar los desarrolladores para evitar perdidas de conocimiento y cuellos de botella
- Pair programming es una manera de hacerlo

Diseño

- ❑ Simplicidad.
- ❑ Usar cartas CRC para las sesiones de diseño
- ❑ Crear soluciones prototipo para reducir riesgo
- ❑ La funcionalidad se adiciona sólo cuando se necesite no pensando en el futuro
- ❑ Refactor cuando y donde sea posible

Diseño: Simplicidad

- Nunca adicionar funcionalidad antes de que sea requerida

Diseño: CRC

Class:

Collaborates with:

Responsibilities:

Diseño: Soluciones prototipo

- Programación simple para explorar soluciones.
- Propósito:
 - ▣ Reducir riesgos
 - ▣ Refinar estimaciones
- Estas soluciones pueden echar a la basura

Diseño: Refactor

- ❑ Remover redundancia,
- ❑ Eliminar funcionalidad no utilizada, y
- ❑ Rejuvenecer diseños obsoletos
- ❑ Mantener el código claro y conciso

Codificación

- ❑ Respetar estándares de codificación y de documentación del código.
- ❑ Escribir primero el código de las pruebas unitarias
- ❑ Todo el código producido en pares
- ❑ Integración constante
- ❑ Propiedad colectiva del código
- ❑ Dejar optimización para lo último
- ❑ No tiempo extra

Codificación: Pruebas primero

- Es más fácil y más rápido escribir el código si primero se han hecho las pruebas unitarias
- Esto ayuda al desarrollador a escribir el código que realmente se necesita
- Esto ayuda a tener inmediata retroalimentación

Codificación: Pair programming

- Todo el código que será incluido en producción debe ser producido por dos personas que trabajan juntas en un computador:
 - ▣ Una persona escribe u piensa tácticamente sobre el método que se está creando
 - ▣ Mientras que la otra persona piensa estratégicamente sobre cómo el método se integra con el resto de la clase y chequea que:
 - sea correcto
 - se entienda
 - uso de estándares

Codificación : Integración secuencial frecuente

- Problemas de integración:
 - ▣ solución XP: integración estrictamente secuencial realizada por los mismos desarrolladores
 - ▣ Sólo integra un par a la vez, prueba y libera el depósito del código

Testing

- Todo el código debe tener pruebas unitarias
- Todo el código debe pasar las pruebas unitarias antes de que sea liberado
- Se deben crear nuevas pruebas cuando un defecto es encontrado
- Las pruebas de aceptación se ejecutan frecuentemente

Testing: Unit Test Framework


- • <http://Xprogramming.com>
- • <http://www.junit.org/>

Testing: Pruebas de Aceptación

- ❑ Las pruebas de aceptación son pruebas de caja negra
- ❑ Cada prueba representa algún aspecto esperado del sistema
- ❑ Clientes son responsables por verificar si la prueba falló o no
- ❑ Las pruebas de aceptación también se deben ejecutar cuando se hacen pruebas de regresión antes de liberar un nuevo release

Conclusiones

- Principales suposiciones:
 - ▣ participación del cliente y negociación
 - ▣ Iteraciones
 - ▣ Pequeños incrementos
 - ▣ Testing
 - ▣ Integración
 - ▣ Buenos diseñadores (Simplicidad)
 - ▣ Trabajo en equipo

- 
- ❑ XP es un proceso: actividades, entregables, responsables, métodos, ...
 - ❑ Principales actividades: Planeación, diseño, codificación, pruebas
 - ❑ XP no es para hackers
 - ❑ XP no es sinónimo de “Programación Heroica”
 - ❑ XP requiere disciplina

Referencias

- Beck, Extreme Programming Explained, Addison Wesley, 1999, ISBN 0- 201- 61641- 6
- Beck & Fowler, Planning Extreme Programming, Addison Wesley, 2000, ISBN 0- 201- 71091- 9
- Fowler, Refactoring, Addison Wesley, 1999, ISBN 0- 20148567- 2
- Jeffries, Anderson & Hendrickson, Extreme Programming Installed, Addison Wesley, 2001, ISBN 0- 201- 70842- 6.31
- [http:// www. extremeprogramming. org/](http://www.extremeprogramming.org/)
- [http:// www. xprogramming. com/](http://www.xprogramming.com/)
- [http:// www. martinfowler. com/](http://www.martinfowler.com/)