Técnico en

DESARROLLO DE SOFTWARE

Metodología de Desarrollo de Software II

Metodología de Desarrollo de Software II

Unidad I

Extreme Programming

1. Extreme Programming

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.

¿Qué es programación extrema o XP?

Extreme Programming (XP) es una disciplina para el desarrollo de software basada en los valores de simpleza, comunicación, feedback y coraje. Reune al Equipo Completo junto a prácticas simples, con el feedback suficiente para permitirle al equipo ver en dónde está y ajustar las prácticas a su situación única.

Ventajas

El uso de la metodología de programación extrema nos brinda las siguientes ventajas:

- Programación organizada.
- Menor taza de errores.
- Satisfacción del programador.
- Apropiado para entornos volátiles
- Estar preparados para el cambio, significa reducir su coste.
- Planificación más transparente para nuestros clientes, conocen las fechas de entrega de funcionalidades. Vital para su negocio
- Permitirá definir en cada iteración cuales son los objetivos de la siguiente
- Permite tener realimentación de los usuarios muy útil.
- La presión está a lo largo de todo el proyecto y no en una entrega final.

Desventajas

- Es recomendable emplearlo solo en proyectos a corto plazo.

- Altas comisiones en caso de fallar.
- Delimitar el alcance del proyecto con nuestro cliente.

2. Valores De La Programación Extrema

XP se funda en cuatro valores los cuales se explican a continuación.

Simplicidad

"Desarrollaremos lo que sea solicitado y necesario, pero no más que eso. De esa forma, se maximiza el valor de la inversión realizada. Nos dirigiremos a nuestro objetivo a pasos simples y pequeños, mitigando las fallas a medida que ocurran. Crearemos algo de lo cual podamos sentirnos orgullos y que pueda mantenerse en el largo plazo a costos razonables." (Cita de extremeprogramming.org).

En XP se comienza desarrollando las soluciones más sencillas necesarias para solucionar los problemas (requerimientos) que se están viendo en ese momento, añadiendo funcionalidad extra más tarde, en la medida en que se obtiene más información de los requerimientos. La diferencia respecto a esquemas tradicionales es que se enfoca en las necesidades de hoy en lugar de las necesidades de mañana, la semana que viene o el mes que viene.

Retroalimentación (Feedback)

El valor de la retroalimentación establece: “Nos tomaremos seriamente los compromisos con el usuario establecidos en todas las iteraciones, entregando software en funcionamiento en cada una. Mostraremos al usuario nuestro software frecuentemente y de forma temprana, escuchando cuidadosamente sus observaciones y realizando los cambios que sean necesarios. Adaptaremos nuestros procesos al proyecto y no al contrario”.

Se recomienda el tener la retroalimentación necesaria que cumpla con los siguientes aspectos:

- Retroalimentación del sistema: Por medio de la ejecución de pruebas unitarias y de integración, los programadores reciben retroalimentación directa del estado del sistema.
- Retroalimentación del cliente (usuario): Las pruebas de aceptación, son diseñadas conjuntamente por el cliente y los analistas de pruebas, obteniendo en conjunto retroalimentación del estado actual del sistema. Esta revisión puede hacerse cada 2 o 3 semanas, permitiendo así que el cliente sea quien guíe el desarrollo del software.

- Retroalimentación del equipo: Cuando el cliente trae nuevos requerimientos, el equipo puede directamente proporcionar la estimación del tiempo que tomará implementarlos.

Bajo este esquema, las fallas de sistema se pueden comunicar fácilmente, pues existen pruebas unitarias que demuestran que el sistema fallará si es puesto en producción. Asimismo, un cliente puede probar el sistema periódicamente, contrastando el funcionamiento con sus requerimientos funcionales o "Historias de usuario".

Coraje

"En nuestros avances y estimados, no documentaremos excusas para el fracaso, pues planificamos para tener éxito. No tendremos miedo a nada pues sabemos que nadie trabaja solo. Nos adaptaremos a los cambios cuando sea que estos ocurran." (Cita de extremeprogramming.org).

Algunas prácticas del coraje son:

- Diseñar y programar para hoy y no para mañana, evitando así hacer énfasis en el diseño en detrimento de todo lo demás.
- Refactorizar el código siempre que sea necesario (No tener reservas al respecto).
- Inspeccionar constantemente el código y modificarlo (refactorizar), de tal manera que futuros cambios se puedan implementar más fácilmente (desarrollar rápido para atender las necesidades de hoy pero refactorizar después para facilitar el mantenimiento).

- Desechar componentes o piezas de código cuando sea necesario, sin preocuparse del tiempo invertido (y perdido) en su creación (Es mejor desechar algo que no es útil en lugar de tratar de repararlo).
- Ser persistente en la resolución de problemas.

Respeto

El valor del respeto en XP establece: “Todos en el equipo dan y reciben el respeto que merecen como integrantes del equipo y los aportes de cada integrante son valorados por todos. Todos contribuyen, así sea simplemente con entusiasmo. Los desarrolladores respetan la experticia de los clientes y viceversa. La Gerencia respeta el derecho del equipo de asumir responsabilidad y tener autoridad sobre su trabajo”. (Cita de extremeprogramming.org).

Respeto es tanto por el trabajo de los demás como por el trabajo de uno mismo, por ejemplo, los desarrolladores nunca deben subir cambios que impidan la compilación de la versión, que hagan fallar pruebas unitarias ya realizadas o que de alguna otra forma retrasen el trabajo de sus pares, esto significa tener respeto por el trabajo (y el tiempo) de los demás.

Asimismo, los desarrolladores respetan su propio trabajo por medio de su compromiso con una alta calidad y buscando el mejor diseño para la solución por medio de la refactorización constante.

En cuanto al trabajo en equipo, nadie debe sentirse poco apreciado o ignorado, todos deben colaborar en esto, tratando con respeto a sus compañeros y mostrando respeto por sus opiniones, esto asegura altos niveles de motivación y lealtad hacia el proyecto.

3. Actores y sus responsabilidades

Hay diferentes roles en XP para diferentes tareas y propósitos durante el proceso y sus prácticas.

- **Programador**. El programador escribe las pruebas y mantiene el código del programa tan simple y definido como sea posible.
- **Cliente**. Escribe las historias (especificaciones) y pruebas funcionales, decide cuándo es logrado cada requisito y determina la prioridad de la implementación de cada uno de ellos.
- **Verificadores (Testers).** Ayudan al cliente a escribir pruebas funcionales, las corren regularmente, comunican los resultados de las mismas y mantienen las herramientas de prueba.
- **Seguidor de rastros (Tracker).** El seguidor de rastros retroalimenta en XP. Sigue las estimaciones hechas por el equipo, es decir, la estimación del esfuerzo; y da retroalimentación en cómo están de acertados para mejorar las futuras estimaciones.
- **Facilitador**. Es la persona responsable del proceso como un todo. Un buen entendimiento de XP es importante en este rol para habilitar al entrenador para guiar a los otros miembros del equipo en el siguiente proceso.

- **Consultor técnico**. El consultor técnico es un miembro externo que procesa el conocimiento técnico específico necesitado y guía al equipo en resolver sus problemas específicos.
- **Administrador**. El administrador toma las decisiones. Para lograr esto, él se comunica con el equipo del proyecto para determinar la situación actual y distinguir cualquier dificultad o deficiencia en el proceso.

4. Ciclo de Vida de la Programación Extrema

El ciclo de vida de XP consiste de 6 fases:

Exploración

En la fase de Exploración los clientes escriben las historias de usuario (funcionalidades con que debe contar el sistema) de lo que ellos quisieran incluir para la primera entrega. Cada plantilla describe las características que deben ser adicionadas al programa. Al mismo tiempo el equipo del proyecto se familiariza con las herramientas, la tecnología y las prácticas que utilizarán en el proyecto. La tecnología a ser usada será probada y las posibles arquitectura para el sistema son exploradas construyendo un prototipo del sistema. La fase de exploración toma entre unas cuantas semanas a unos cuantos meses, dependiendo de que tanto los programadores conocen la tecnología.

Planeación

La fase de planeación configura la prioridad para las historias de usuario, contenidas en las tarjetas CRC (Clase-Responsabilidad-Colaboración, una técnica que reemplaza a los

diagramas para la representación de modelos, en las que se escriben las responsabilidades) y se realiza un contrato del contenido para la primera entrega. Los programadores primero estiman cuánto esfuerzo requieren para cada historia y se hace una programación de acuerdo a esta estimación. El tiempo de la programación de la primera entrega normalmente no excede dos meses y el tiempo de la fase como tal toma un par de días.

Iteraciones

La fase de iteraciones hacia la entrega incluye varias iteraciones del sistema antes de la primera entrega. La programación que se determinó en la etapa de planeación es dividida en un número de iteraciones donde cada una tomará de una a cuatro semanas para ser implementada. La primera iteración crea la arquitectura de todo el sistema; esto es logrado seleccionando las historias que hacen cumplir la estructura para todo el sistema. El cliente decide las historias seleccionadas para cada iteración. Las pruebas funcionales creadas por los clientes son para correr al final de cada iteración. Al final de la última iteración, el sistema estará listo para ser entregado y llevarlo a producción.

Producción

La fase de producción requiere pruebas extras y chequeos de la ejecución del sistema antes de que sea entregado al cliente. En ésta fase, se pueden encontrar nuevos cambios y se toma la decisión si serán incluidos en la entrega actual. Durante esta fase, las iteraciones pueden necesitar ser recortadas de tres semanas a una semana. Después que la primera entrega es producida para el uso del cliente, el proyecto XP debe

mantener el sistema en producción corriendo mientras que también se estén produciendo nuevas iteraciones.

Mantenimiento

La fase de mantenimiento requiere también un esfuerzo para soportar las tareas de los clientes. Así, la velocidad del desarrollo puede desacelerarse después de que el sistema está en producción. La fase de mantenimiento puede requerir incorporar nuevas personas al equipo y cambiar la estructura del equipo. Dentro de esta fase se llega a un estado llamado “de muerte”, que sucede cuando el cliente no tiene más historias para ser implementadas. Esto requiere que el sistema satisfaga también las necesidades en otros aspectos, como por ejemplo lo concerniente a la ejecución y la confiabilidad. Éste es el momento en el proceso XP cuando la documentación necesaria del sistema es finalmente escrita porque no habrá más cambios en la arquitectura, diseño o código. La muerte puede ocurrir si el sistema no está entregando los artefactos deseados o si se está convirtiendo muy costoso implementarlo.

Muerte del Proyecto

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

5. Prácticas de Programación Extrema

1. **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.
2. **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.
3. **Test del cliente**: El cliente, con la ayuda de los desarrolladores, propone sus propias pruebas para validar las mini-versiones.
4. **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.
5. **Diseño simple**: Hacer siempre lo mínimo imprescindible de la forma más sencilla posible. Mantener siempre sencillo el código.
6. **Pareja de programadores**: Los programadores trabajan por parejas (dos delante del mismo ordenador) y se intercambian las parejas con frecuencia (un cambio diario).
7. **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.
8. **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.

9. **El código es de todos**: Cualquiera puede y debe tocar y conocer cualquier parte del código. Para eso se hacen las pruebas automáticas.
10. **Normas de codificación**: Debe haber un estilo común de codificación (no importa cuál), de forma que parezca que ha sido realizado por una única persona.
11. **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.
12. **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.

6. Referencias

- PMO Informática
  - http://www.pmoinformatica.com/2012/11/los-5-valores-de-la-programacion.html
- Extreme Programming
  - http://www.extremeprogramming.org/values.html
- Oness
  - http://oness.sourceforge.net/proyecto/html/ch05s02.html
- EcuRed

- http://www.ecured.cu/index.php/Extreme_Programming
- DanielzBlogspot
- http://danielzs75.blogspot.com/
- Dos Ideas
- http://www.dosideas.com/noticias/metodologias/822-una-introduccion-a-extreme-programming.html
- Ingeniería de Software
- http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html