Introducción a XP

XP Introduction

Daynel Marmol Lacal
Universidad de las Ciencias Informáticas
dmarmol@uci.cu

Resumen

En el presente trabajo se pretenden mencionar las características fundamentales de XP, así como algunos de sus inconvenientes a la hora de su aplicación, brindando así un punto de referencia a la hora de escoger alguna metodología de desarrollo de software.

Palabras clave: Desarrollo de software, metodologías, XP.

Abstract

This work aim at presenting the main XP features, as well as some inconvenient when applying and putting it into practice, so it offers a point of view when choosing a methodology of software development.

Key words*: methodologies, software development, XP.*

Introducción

XP (Programación Extrema o Extreme Programming) salió a la luz en el año 1996, en un proyecto desarrollado por Kent Beck. Desde ese momento hasta la actualidad ha generado un gran interés y controversia. Es una nueva disciplina que se ha sumado al creciente conjunto de métodos, técnicas y tecnologías existentes, y de la cual vamos a comentar en este trabajo por ser la más popular y usada de las de su tipo. Se clasifica como un método ágil (además de XP se pueden mencionar a Scrum, Cristal Clear, DSDM (Dynamic System Development Method), FDD (Feature Driven Development), ASD (Adaptative Software Development), XBreed y Extreme Modeling, entre otras), que no es más que una estrategia de software que promueve prácticas adaptativas en lugar de predictivas, centradas en las personas o en los equipos, iterativas, orientadas hacia prestaciones y hacia la entrega, de comunicación intensiva y que requieren que el negocio se involucre de forma directa.

Extreme Programming (XP)

La metodología XP está formada por un conjunto de valores y principios, con el objetivo fundamental de un rápido desarrollo y un software de alta calidad.

Principios de XP:

- **El juego del planeamiento**: rápidamente determinar el alcance de la próxima liberación (release) mediante la combinación de prioridades del negocio y estimaciones técnicas. A medida que la realidad va cambiando el plan, actualizar el mismo.
- **Pequeñas liberaciones**: poner un sistema simple en producción rápidamente, luego liberar nuevas versiones en ciclos muy cortos.
- **Metáfora (convención de nombres fácil de recordar)**: guiar todo el desarrollo con una historia simple y compartida de cómo funciona todo el sistema.
- **Diseño simple**: el sistema deberá ser diseñado tan simple como sea posible en cada momento. La complejidad extra es removida apenas es descubierta.

- **Pruebas**: los programadores constantemente escriben pruebas unitarias, las cuales deben correr sin problemas para que el desarrollo continúe. Los clientes escriben pruebas demostrando que las funcionalidades están terminadas.
- **Refactorización (refactoring)**: los programadores reestructuran el sistema sin cambiar su comportamiento para remover duplicación, mejorar la comunicación, simplificar, o añadir flexibilidad.
- **Programación en pares**: todo el código de producción es escrito por dos programadores en una máquina.
- **Propiedad colectiva del código**: cualquiera puede cambiar código en cualquier parte del sistema en cualquier momento.
- **Integración continua**: integrar y hacer construcciones (builds) del sistema varias veces por día, cada vez que una tarea se completa.
- **Semana de 40 horas**: trabajar no más de 40 horas semanales como regla. Nunca trabajar horas extras durante dos semanas consecutivas.
- **Cliente en el lugar de desarrollo**: incluir un cliente real en el equipo, disponible a tiempo completo para responder preguntas.
- **Estándares de codificación**: los programadores escriben todo el código de acuerdo con reglas que enfatizan la comunicación a través del mismo.

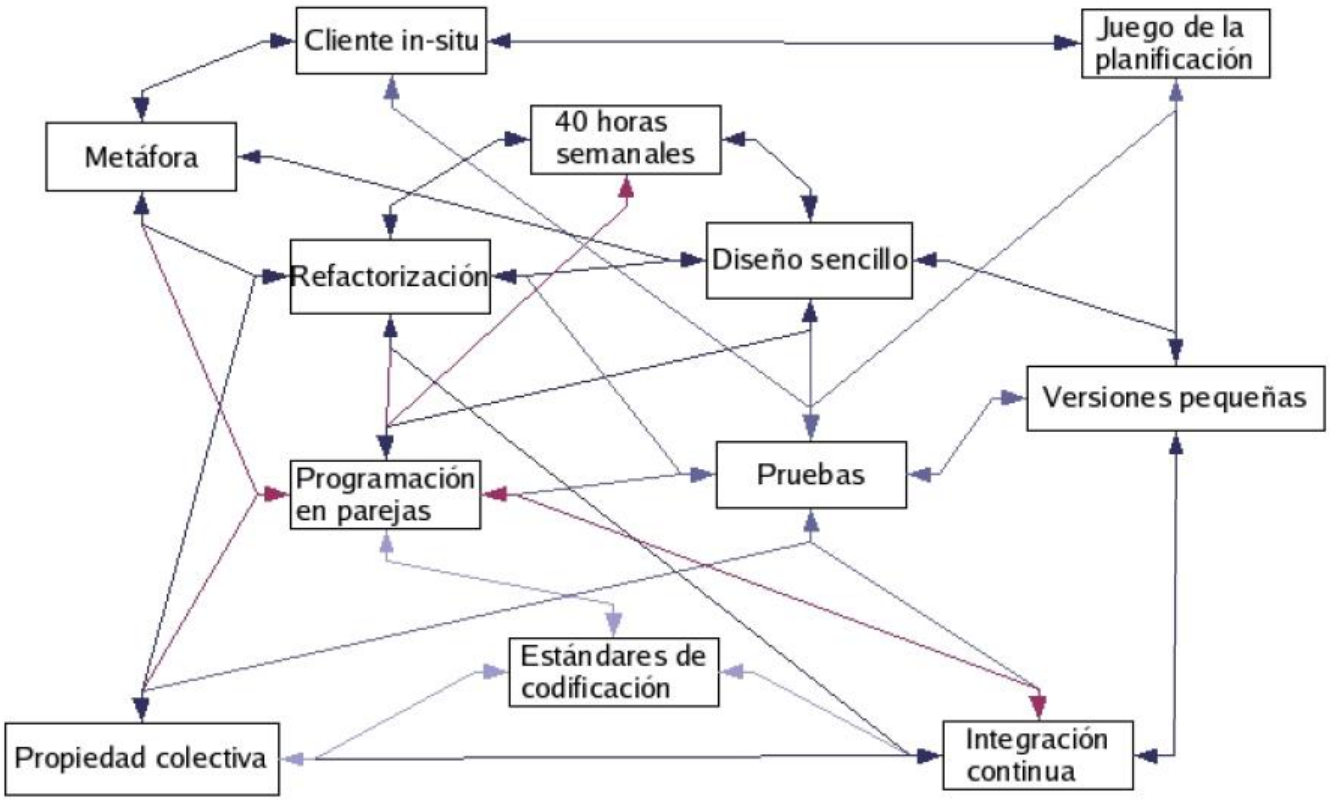

Fig. 1. Los principios se refuerzan entre sí (Letelier, 2003).

Valores de XP(Beck, 1999):

- **Comunicación**: es lo más importante dentro de un equipo de desarrollo de software, cuando aparecen problemas durante el desarrollo es muy probable que alguien del equipo conozca la solución.
- **Simplicidad**: hay que verla dentro del contexto del proyecto, mejorando la comunicación se ayuda a ganar en simplicidad eliminando requerimientos innecesarios.

- **Retroalimentación**: cuando se habla de desarrollo de software o de los requerimientos de un sistema, los cambios son inevitables (creando la necesidad de retroalimentación). Es una parte crítica de la comunicación, y contribuye a la simplicidad.
- **Coraje**: en conjunto con los otros valores es poderoso, el coraje para descartar soluciones fallidas y buscar nuevas lleva a la simplicidad, el coraje para buscar respuestas concretas y reales crea retroalimentación.
- **Respeto**: los cuatro valores anteriores apuntan a este, que a su vez yace en la superficie de los anteriores. Si a los miembros de un equipo no les importa los otros y lo que estos hacen, XP no funciona.

Escalando XP:

Muchos se preguntan como escalar XP. Un equipo de 100 personas no puede hacer un planeamiento en detalle de su trabajo en una reunión por semana, pero si pueden trabajar juntos con un espíritu de comunicación, retroalimentación, simplicidad, coraje y respeto. Crear un equipo de esa magnitud no es igual a crear un equipo pequeño, pero es hecho todo el tiempo.

Aunque hay que resaltar que la cantidad de personas en un equipo de trabajo no es la única medida a escalar durante el desarrollo de un software. Las dimensiones a escalar pueden ser variadas, entre ellas:

- **número de personas**: al enfrentarse a un problema grande con un equipo de desarrollo pequeño, para resolver el problema hay que dividirlo en pequeños subproblemas, cada uno soluble por equipos pequeños. Esta partición introduce el riesgo de que los pedazos no se puedan unir a la hora de la integración, para mitigarlo es necesario integrar frecuentemente y conciliar las supuestas diferencias entre los equipos.
- **tamaño de la organización**: en esta área un equipo puede beneficiarse de las habilidades del jefe de proyecto, quien presenta la información de forma tal que la organización pueda asimilarla (siempre con la verdad), asegurándose que se cumplan las expectativas, esto es un gran desafío debido a las características de XP, respetar a los demás y no actuar en beneficio propio.
- **tiempo**: el caso más simple se da cuando el equipo mantiene la continuidad a través del proyecto. Los proyectos que inician y paran frecuentemente con el equipo disperso, son más difíciles de mantener con el transcurso del tiempo.
- **complejidad del problema**: XP es ideal para proyectos que requieren la estrecha colaboración de los especialistas, un desafío es lograr que cada cual se concentre en su trabajo mientras aprende un poco del otro especialista.
- **complejidad de la solución**: la estrategia de XP para el trabajo con exceso de complejidad es siempre la misma, dividir la complejidad mientras se continúan las entregas. Si se están arreglando los defectos en un área determinada, eliminarlos todos mientras se trabaje allí, esto reduce la sobrecarga de trabajo.
- **consecuencia de los fallos**: un sistema no está certificado como seguro si no fue hecho con un conjunto de principios en mente o auditado por un experto en seguridad. Mientras sean compatibles con XP estas prácticas pueden incorporarse dentro del trabajo diario de un equipo.

Roles

En XP se identifican 7 roles:

- **Programador (programmer)**: estima las historias, define las tareas de las historias y estimados e implementa las historias y las pruebas de unidad.
- **Cliente (customer)**: escribe las historias, especifica las pruebas funcionales, es quien determina las prioridades y puede no ser el usuario final.

- **Encargado de pruebas (tester)**: implementa y ejecuta las pruebas funcionales, y se asegura de dar a conocer cuando los resultados de las pruebas disminuyen.
- **Encargado de seguimiento (tracker)**: monitorea el avance del programador y actúa si las cosas parecen irse de su curso normal. Entre las acciones a tomar están: reunirse con el cliente, preguntarle al jefe o a otro programador para obtener ayuda.
- **Jefe (coach)**: lo mira todo, se asegura que el proyecto se mantenga en su curso y ayuda con cualquier cosa.
- **Consultor (consultant)**: miembro externo al equipo con conocimiento en algún tema necesario al proyecto, guía al equipo a resolver un problema específico.
- **Gestor (manager o big boss)**: programa las reuniones, se asegura que el proceso es continuo, guarda los resultados de las reuniones para reportes en el futuro, es el vínculo entre los clientes y los programadores.

Proceso XP:

El ciclo de vida ideal de XP consta de 6 fases (Beck, 1999):

- **Exploración**: los clientes plantean a grandes rasgos las historias de usuario que les son de interés para la primera entrega del producto, a la vez que los programadores se familiarizan con las herramientas, tecnologías y prácticas que se utilizarán. Demora de pocas semanas a pocos meses, en dependencia de la magnitud y la familiaridad de los programadores con la tecnología.
- **Planificación de la entrega**: el cliente establece la prioridad de cada historia de usuario, y en correspondencia los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega (en no más de tres meses) y se determina un cronograma en conjunto con el cliente. Esta fase dura unos pocos días.
- **Iteraciones**: esta fase incluye varias iteraciones sobre el sistema antes de ser entregado, en la primera iteración se puede intentar establecer una arquitectura del sistema a ser utilizada durante el proyecto. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas, al final de la última iteración el sistema estará listo para entrar en producción.
- **Producción**: requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente, además de la toma de decisiones sobre la inclusión de nuevas características a la versión actual debido a cambios durante esta fase. Las ideas propuestas y sugerencias son documentadas para su posterior implementación.
- **Mantenimiento**: mientras se encuentra en producción la primera versión, el proyecto debe mantener el sistema en funcionamiento a la vez que desarrolla nuevas iteraciones, para esto se requiere de tareas de soporte para el cliente. De esta forma la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. Esta fase puede requerir nuevo personal dentro del equipo.
- **Muerte del proyecto**: cuando el cliente no tiene más historias para ser incluidas en el 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.

Cuando no usar XP

XP es inefectivo en organizaciones cuyos valores actuales son incompatibles con los valores de XP, donde se tiene un conjunto de prácticas que pretenden expresar y reforzar un grupo de valores. Si una organización tiene como premisas el secreto, aislamiento, complejidad, timidez o irrespeto, estaría expresando lo contrario a XP, por lo que su aplicación causaría problemas en lugar de mejoras.

Entre las principales críticas que se le encuentran a XP tenemos:

- poco diseño previo.
- no hay documentación.
- las pruebas son realizadas por los mismos programadores.
- el usuario representativo puede no serlo.

Conclusiones

A pesar de todas las técnicas, métodos y metodologías existentes, aun no existe nada preciso para que un proyecto de desarrollo de software sea exitoso, sino que deben ser adaptados al contexto del proyecto. Una de las características de las metodologías ágiles es la adaptabilidad y la flexibilidad ante determinadas situaciones, de allí su gran uso y aplicación, entre ellas destacando a XP que es la más popular. Aunque cabe señalar la serie de inconvenientes y restricciones para su aplicación que no se deben pasar por alto si no queremos fracasar a la hora de la gestión de un proyecto de desarrollo de software.

XP favorece el trabajo en equipo y ayuda a la comunicación entre sus miembros y al entendimiento con el cliente al tenerlo como un miembro más dentro del equipo de desarrollo con un rol bien definido.

Referencias Bibliográficas

(Acebal, 2002) Acebal CF, Lovelle JMC. A New Method of Software Development: eXtreme Programming. 2002.

(Beck, 1999) Beck K. Extreme Programming Explained. Embrace Change: Pearson Education, 1999.

(Emery, 2002) Emery P. The Dangers of Extreme Programming. 2002.

(Hernán, 2004) Hernán SM. Diseño de una Metodología Ágil de Desarrollo de Software. Facultad de Ingeniería: Universidad de Buenos Aires, 2004.

(Letelier, 2003) Letelier P, Penadés MC. Metodologías Ágiles para el Desarrollo de Software: eXtreme Programming (XP). Valencia: Universidad Politécnica de Valencia, 2003.

(Sanchez, 2004) Sanchez MAM. Metodologías de Desarrollo de Software. 2004.

(Smith, 2001) Smith J. A comparison of RUP and XP. 2001.