Desarrollo Ágil

MATERIA: TECNOLOGIA WEB

ESTUDIANTE:

✓ OLIVA ROJAS GERSON 216036127

Contenido

PROGRAMACION EXTREMA XP 3
HISTORIA 3
INTRODUCCION 3
¿QUÉ ES PROGRAMACIÓN EXTREMA O XP? 3
OBJETIVOS. 3
CONTEXTO XP 4
CARACTERÍSTICAS XP 4
VALORES XP 4
EL ESTILO XP 4
PRÁCTICAS BÁSICAS DE LA PROGRAMACIÓN EXTREMA 4
VENTAJAS Y DESVENTAJAS DE EXTREME PROGRAMMING 6
VENTAJAS: 6
DESVENTAJAS: 6
SCRUM 7
HISTORIA 7
CARACTERÍSTICAS DE SCRUM 7
DESARROLLO MANEJADO POR RASGOS 8
METODOLOGÍA DE TRABAJO 9
EL FLUJO SCRUM 9

PROGRAMACION EXTREMA 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.

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.

SCRUM

HISTORIA

En 1986 Hirotaka Takeuchi e Ikujiro Nonaka describieron una nueva aproximación holística que incrementa la rapidez y la flexibilidad en el desarrollo de nuevos productos comerciales.[1] Takeuchi y Nonaka comparan esta nueva aproximación holística, en la cual las fases se traslapan de manera intensa y el proceso completo es realizado por un equipo con funciones transversales, con el rugby, donde el equipo entero «actúa como un solo hombre para intentar llegar al otro lado del campo, pasando el balón de uno a otro».[*cita requerida*]Los casos de estudio provienen de las industrias automovilísticas, así como de fabricación de máquinas fotográficas, computadoras e impresoras.

En 1991 Peter DeGrace y Leslie Stahl en su libro *Wicked Problems, Righteous Solutions (A problemas malvados, soluciones virtuosas)*,[2]se refirieron a esta aproximación como *scrum* (melé en inglés), un término propio del rugby mencionado en el artículo por Takeuchi y Nonaka.

A principios de los años 1990 Ken Schwaber empleó una aproximación que lo llevó a poner en práctica el *scrum*en su compañía, Advanced Development Methods.[*cita requerida*]Por aquel tiempo Jeff Sutherland desarrolló una aproximación similar en Easel Corporation y fue el primero en denominarla *scrum*.[3] En 1995 Sutherland y Schwaber, durante el OOPSLA '95 desarrollado en Austin, presentaron en paralelo una serie de artículos describiendo *scrum*, siendo ésta la primera aparición pública de la metodología.Durante los años siguientes, Schwaber y Sutherland, colaboraron para consolidar los artículos antes mencionados, así como sus experiencias y el conjunto de mejores prácticas de la industria que conforman a lo que ahora se le conoce como *scrum*.En 2001, Schwaber y Mike Beedle describieron la metodología en el libro *Agile Software Development with Scrum*.

CARACTERÍSTICAS DE SCRUM

Scrum es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Los roles principales en Scrum son el *ScrumMaster*, que mantiene los procesos y trabaja de forma similar al director de proyecto, el *ProductOwner*, que representa a los *stakeholders* (clientes externos o internos), y el *Team*que incluye a los desarrolladores.

Durante cada *sprint*, un periodo entre 15 y 30 días (la magnitud es definida por el equipo), el equipo crea un incremento de software *potencialmente entregable* (utilizable). El conjunto de características que forma parte de cada sprint viene del *Product Backlog*, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar. Los elementos del *Product Backlog* que forman parte del sprint se determinan durante la reunión de *Sprint Planning*. Durante esta reunión, el *Product Owner* identifica los elementos del *Product Backlog* que quiere ver completados y los hace del conocimiento del equipo. Entonces, el equipo determina la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente sprint.[4] Durante el sprint, nadie puede cambiar el Sprint Backlog, lo que significa que los requisitos están congelados durante el sprint.

Existen varias implementaciones de sistemas para gestionar el proceso de Scrum, que van desde notas amarillas "post-it" y pizarras hasta paquetes de software. Una de las mayores ventajas de Scrum es que es muy fácil de aprender, y requiere muy poco esfuerzo para comenzarse a utilizar.

Scrum ha estado durante algún tiempo en los círculos orientados a objetos, aunque confesaré que yo no estoy muy al tanto de su historia o desarrollo. De nuevo se enfoca en el hecho de que procesos definidos y repetibles sólo funcionan para atacar problemas definidos y repetibles con gente definida y repetible en ambientes definidos y repetibles.

Scrum divide un proyecto en iteraciones (que ellos llaman carreras cortas) de 30 días. Antes de que comience una carrera se define la funcionalidad requerida para esa carrera y entonces se deja al equipo para que la entregue. El punto es estabilizar los requisitos durante la carrera.

Sin embargo la gerencia no se desentiende durante la carrera corta. Todos los días el equipo sostiene una junta corta (quince minutos), llamada scrum, dónde el equipo discurre lo que hará al día siguiente. En particular muestran a los bloques de la gerencia: los impedimentos para progresar que se atraviesan y que la gerencia debe resolver. También informan lo que se ha hecho para que la gerencia tenga una actualización diaria de dónde va el proyecto.

La literatura de Scrum se enfoca principalmente en la planeación iterativa y el seguimiento del proceso. Es muy cercana a las otras metodologías agiles en muchos aspectos y debe funcionar bien con las prácticas de código de la XP.

DESARROLLO MANEJADO POR RASGOS

El Desarrollo Manejado por Rasgos (FDD por sus siglas en inglés) fue desarrollado por Jeff De Luca y el viejo gurú de la OO PeterCoad. Como las otras metodologías adaptables, se enfoca en iteraciones cortas que entregan funcionalidad tangible. En el caso del FDD las iteraciones duran dos semanas.

El FDD tiene cinco procesos. Los primeros tres se hacen al principio del proyecto.

- Desarrollar un Modelo Global
- Construir una Lista de los Rasgos
- Planear por Rasgo
- Diseñar por Rasgo
- Construir por Rasgo

Los últimos dos se hacen en cada iteración. Cada proceso se divide en tareas y se da un criterio de comprobación.

Los desarrolladores entran en dos tipos: dueños de clases y programadores jefe. Los programadores jefe son los desarrolladores más experimentados. A ellos se les asignan rasgos a construir. Sin embargo ellos no los construyen solos. Solo identifican qué clases se involucran en la implantación de un rasgo y juntan a los dueños de dichas clases para que formen un equipo para desarrollar ese rasgo. El programador jefe actúa como el coordinador, diseñador líder y mentor mientras los dueños de clases hacen gran parte de la codificación del rasgo.

Hasta recientemente, la documentación sobre FDD era muy escasa. Finalmente hay un libro completo sobre FDD. Jeff De Luca, el inventor primario, ya tiene un portal FDD con artículos, blogs y foros de discusión. La descripción original estaba en el libro UML in Color de Peter Coad et al. Su compañía, TogetherSoft, también da consultoría y entrenamiento en FDD.

METODOLOGÍA DE TRABAJO

Equipos de entre 6 y 10 personas revisan los requisitos, la tecnología disponible y evalúan los conocimientos para Colectivamente determinar como **incrementar la uncionalidad.**

* Reuniones diarias, antes de empezar a trabajar,con una duración máxima de 4 hrs.

* Se llevan a cabo hasta que el proyecto este listo para ser puesto en producción o ser lanzado al mercado.

EL FLUJO SCRUM