Revista de Arquitectura e Ingeniería
E-ISSN: 1990-8830
melena-torrensp@empai.co.cu
Empresa de Proyectos de Arquitectura e Ingeniería de Matanzas
Cuba

Delgado Expósito, Erly
Metodologías de desarrollo de software. ¿Cuál es el camino?
Revista de Arquitectura e Ingeniería, vol. 2, núm. 3, diciembre, 2008
Empresa de Proyectos de Arquitectura e Ingeniería de Matanzas
Matanzas, Cuba

Metodologías de desarrollo de software. ¿Cuál es el camino?

Ing. Erly Delgado Expósito
Profesor Ingeniería Informática
Universidad de Matanzas "Camilo Cienfuegos"
Telf: (45) 283498
Email: erly_delgado2003@yahoo.es

Recibido: 29-05-2008
Aceptado: 2-09-2008

RESUMEN :
El desarrollo de software no es una tarea fácil. Como resultado a este problema ha surgido una alternativa desde hace mucho: la Metodología. Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar, inspirado por otras disciplinas de la ingeniería.
Las metodologías ingenieriles han estado presentes durante mucho tiempo. No se han distinguido precisamente por ser muy exitosas, aún menos por su popularidad. La crítica más frecuente a estas metodologías es que son burocráticas. Hay tanto que hacer para seguir la metodología que el ritmo entero del desarrollo se retarda.
La razón de ser de este trabajo se basa en el análisis de algunos tipos de metodologías existentes, viendo sus principales características y los rasgos que las hacen superior a las otras.
Palabras claves: Metodologías, RUP, XP, 3P, Software, Procesos, Ingeniería.

ABSTRACT:
The software development is not an easy task. As a result to this problem an alternative has arisen for a lot: the Methodology. The methodologies impose a process disciplined on the software development with the purpose of making it more predictable and more efficient. They make it developing a detailed process with a strong emphasis in planning, inspired by other disciplines of the engineering.
The methodologies of engineerings have been present during a lot of time. They have not in fact been distinguished to be very successful, even less for their popularity. The most frequent critic to these methodologies is that they are bureaucratic. There is so much to make to follow the methodology that the whole rhythm of the development is slowed.
The reason of being of this work is based on the analysis of some types of existent methodologies, seeing its main ones characteristic and the features that make them superior to the other ones.
Key words: Methodologies, RUP, XP, 3P, Software, Processes, Engineering.

Introducción:

Hoy en día existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Un ejemplo de ellas son las propuestas tradicionales centradas específicamente en el control del proceso. Éstas han demostrado ser efectivas y necesarias en un gran número de proyectos, sobre todo aquellos proyectos de gran tamaño (respecto a tiempo y recursos). Sin embargo la experiencia ha demostrado que las metodologías tradicionales no ofrecen una buena solución para proyectos donde el entorno es volátil y donde los requisitos no se conocen con exactitud, porque no están pensadas para trabajar con incertidumbre.

Aplicar metodologías tradicionales nos obliga a forzar a nuestro cliente a que tome la mayoría de las decisiones al principio. Como respuesta a los problemas de las metodologías tradicionales surgieron otras metodologías. El encanto de estas metodologías ágiles es su reacción ante la burocracia de las metodologías monumentales. El resultado de todo esto es que los métodos ágiles cambian significativamente algunos de los énfasis de los métodos ingenieriles. La diferencia es que son menos orientados al documento, exigiendo una cantidad más pequeña de documentación para una tarea dada. De muchas maneras son más bien orientados al código: siguiendo un camino que dice que la parte importante de la documentación es el código fuente.

Desarrollo:

Problemas detectados en las metodologías ingenieriles:

Retrasos en la planificación: llegada la fecha de entregar el software, éste no esta disponible.
Sistemas deteriorados: el software se ha creado, pero después de un par de años el coste de su mantenimiento es tan complicado que definitivamente se abandona su producción.
Tasa de defectos: el software se pone en producción, pero los defectos son tantos que nadie lo usa.
Requisitos mal comprendidos: el software no resuelve los requisitos planificados inicialmente.
Cambios de negocio: el problema que resolvía nuestro software ha cambiado y nuestro software no se ha adaptado.
Falsa riqueza: el software hace muchas cosas técnicamente muy interesantes y divertidas, pero no resuelven el problema de nuestro cliente, ni hace que éste gane más dinero.
Cambios de personal: después de unos años de trabajo los programadores comienzan a odiar el proyecto y lo abandonan.

Metodologías pesadas. RUP

El proceso unificado de desarrollo (RUP) es una metodología para la ingeniería de software, que va más allá del mero análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software. El resultado es un proceso basado en componentes, dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental. [6].

Características principales de RUP

Centrado en los modelos: Los diagramas son un vehículo de comunicación más expresivo que las descripciones en lenguaje natural. Se trata de minimizar el uso de descripciones y especificaciones textuales del sistema. [10]
Guiado por los Casos de Uso: Los Casos de Uso son el instrumento para validar la arquitectura del software y extraer los casos de prueba. [10]
Centrado en la arquitectura: Los modelos son proyecciones del análisis y el diseño constituye la arquitectura del producto a desarrollar. [10]
Iterativo e incremental: Durante todo el proceso de desarrollo se producen versiones incrementales (que se acercan al producto terminado) del producto en desarrollo. [10]
Beneficios que aporta RUP
Permite desarrollar aplicaciones sacando el máximo provecho de las nuevas tecnologías, mejorando la calidad, el rendimiento, la reutilización, la seguridad y el mantenimiento del software mediante una gestión sistemática de riesgos. [8]
Permite la producción de software que cumpla con las necesidades de los usuarios, a través de la especificación de los requisitos, con una agenda y costo predecible. [8]
Enriquece la productividad en equipo y proporciona prácticas óptimas de software a todos sus miembros. [8].

Permite llevar a cabo el proceso de desarrollo práctico, brindando amplias guías, plantillas y ejemplos para todas las actividades críticas. [8].

Se integra estrechamente con herramientas Rational, permitiendo a los equipos de desarrollo aprovechar todas las ventajas de las características de los productos Rational, el Lenguaje de Modelado Unificado (UML) y otras prácticas óptimas de la industria. [8].
Optimiza la productividad de cada miembro del equipo al poner al alcance la experiencia derivada de miles de proyectos y muchos líderes de la industria.
No solo garantiza que los proyectos abordados serán ejecutados íntegramente, sino que además evita desviaciones importantes respecto a los plazos. [8].

Metodologías ágiles.

XP
La Programación Extrema surge ideada por Kent Beck, como proceso de creación de software diferente al convencional. En palabras de Beck: “XP es una metodología ligera, eficiente, con bajo riesgo, flexible, predecible y divertida para desarrollar software”.

Objetivos de XP
Los objetivos de XP son muy simples: **la satisfacción del cliente**. Esta metodología trata de dar al cliente el software que él necesita y cuándo lo necesita. Por tanto, debemos responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final de ciclo de la programación.
El segundo objetivo es **potenciar al máximo el trabajo en grupo**. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software.

Bases de XP
La programación extrema se basa en la simplicidad, la comunicación y el reciclado continuo de código, para algunos no es más que aplicar una pura lógica. Lo que buscan en definitiva es la reducción de costes.

Valores XP
Una de las cosas que a los programadores nos tiene que quedar muy claro es que en el ciclo de vida del desarrollo de un proyecto software los cambios van a aparecer, cambiarán los requisitos, las reglas de negocio, el personal, la tecnología, todo va a cambiar. Por tanto el problema no es el cambio en si, ya que éste va a suceder, sino la incapacidad de enfrentarnos a estos cambios.
Como en otra cualquier actividad humana necesitamos valores para desarrollar nuestro trabajo y conseguir los planteamientos iniciales. Estos cuatro valores son: Comunicación, Sencillez, Retroalimentación, Valentía.
Variables XP
XP define cuatro variables para proyectos: Coste, Tiempo, Calidad, y Ámbito.
Actividades básicas XP
Ahora que tenemos nuestros cuatro valores estamos preparados para construir una disciplina de desarrollo de software. ¿Qué tareas debemos de llevar a cabo para desarrollar un buen software?
Codificar
Es la única actividad de la que no podremos prescindir. Sin código fuente no hay programa, por tanto necesitamos codificar y plasmar nuestras ideas a través del código. En una programación en XP en pareja el código expresa tu interpretación del problema, así podemos utilizar el código para comunicar, para hacer mías tus ideas, y por consiguiente para aprender y mejorar.

Hacer pruebas
Las características del software que no pueden ser demostradas mediante pruebas simplemente no existen. Las pruebas me dan la oportunidad de saber si lo que implementé es lo que en realidad yo pensaba que había implementado. Las pruebas nos indican que nuestro trabajo funciona, cuando no podemos pensar en ninguna prueba que pudiese originar un fallo en nuestro sistema, entonces has acabado por completo.
No debemos de escribir tan solo una prueba, ver qué funciona y salir corriendo, debemos de pensar en todas las posibles pruebas para nuestro código.
Programar y probar es más rápido que sólo programar. Puedes ganar media hora de productividad sin hacer pruebas, pero perderás mucho tiempo en la depuración. Tendrás menos errores, tendrás que volver menos veces sobre el código, te costará menos localizar los errores, perderás menos tiempo, escuchando como tus clientes te dicen que no funciona.
Las pruebas deben de ser sensatas y valientes. No podemos hacer pruebecillas que no testen a fondo el sistema, esos agujeros que vamos dejando nos esperan para cuando pasemos de nuevo por allí y volveremos a caer dentro.

Escuchar
Los programadores no lo conocemos todo y sobre todo muchas cosas que las personas de negocios piensan que son interesantes. Si ellos pudieran programarse su propio software ¿para qué nos querrían ?
Si vamos a hacer pruebas tenemos que preguntar si lo obtenido es lo deseado y tenemos que preguntar quién necesita la información. Tenemos que escuchar a nuestros clientes, cuáles son los problemas de su negocio, debemos de tener una escucha activa explicando lo que es fácil y difícil de obtener y la realimentación entre ambos nos ayudan a todos a entender los problemas.

Diseñar
El diseño crea una estructura que organiza la lógica del sistema, un buen diseño permite que el sistema crezca con cambios en un solo lugar. Los diseños deben de ser sencillos, si alguna parte del sistema es de desarrollo complejo, divídela en varias. Si hay fallos en el diseño o malos diseños, éstos deben de ser corregidos cuanto antes.
En resumen, tenemos que codificar porque sin código no hay programas, tenemos que hacer pruebas, porque sin pruebas no sabemos si hemos acabado de codificar, tenemos que escuchar, porque si no escuchamos no sabemos qué codificar ni probar y tenemos que diseñar para poder codificar, probar y escuchar indefinidamente.

3P [7]
Paradigma 3P es una metodología de desarrollo de software nacida al calor de la experiencia acumulada del grupo de investigación y desarrollo Atis, debido a la insuficiente capacidad de respuesta a los clientes utilizando las metodologías tradicionales.

Principios que sustentan el modelo
Los individuos y sus interacciones son más importantes que los procesos y las herramientas: El PERSONAL.
La comunicación con el cliente evita construir una elegante solución para un problema equivocado: El PROBLEMA.
El software que funciona es más importante que la documentación exhaustiva. El PROCESO.
Valores 3P
Comunicación: Sin comunicación todo proyecto estaría destinado a fracasar, comunicar no es escribir o hablar muchas palabras, sino utilizar solo las palabras necesarias para trasmitir una idea.
Sencillez: Nadie es mejor o peor que los demás miembros del grupo de desarrollo, todos tenemos fortalezas y debilidades, conocerlas hará que nuestras relaciones con los miembros del grupo sean mejores.

Retroalimentación: Saber cuándo se debe rehacer algo que no funciona, equivocarse es de humanos, encarar nuevamente la tarea con emprendimiento y optimismo.
Emprendimiento: Estar dispuesto siempre a acometer las tareas más complejas, encararla con esmero y con alegría hará que crezca nuestro prestigio entre los demás miembros, la convicción y el deseo del triunfo debe prevalecer.
Optimismo: Ser realista, pero tener siempre el pensamiento hacia el éxito.

Actividades básicas
Codificar, Probar, Comunicar, Idear, Escuchar, Diseñar.

Roles del proyecto
Jefe del Proyecto, Cliente, Consultor, Analista-Programador, Programador, Diseñador de Interfaces.

Resumen puntos clave

RUP
- Pesado
- Dividido en cuatro fases, que se dividen en iteraciones
- Los artefactos son el objetivo de cada actividad
- Se basa en roles
- UML
- Muy organizativo
- Mucha documentación

XP
- Ligero
- Cercano al desarrollo
- Se basa en UserStories
- Fuerte comunicación con el cliente
- El código fuente pertenece a todos
- Programación por parejas
- Solo el mínimo de organización
- Pobre en cuanto a documentación

3P

- Ágil
- Cercano al desarrollo, pero sin olvidar el diseño
- Se basa en 3 principios: Personal, Problema, Proceso
- Gran interacción con el cliente
- Logra alcanzar un control y organización del proceso
- Logra un equilibrio en cuanto a la generación de documentación

Conclusiones:
¿Qué debemos esperar entonces de un proceso de desarrollo?
¿Qué criterios debe cumplir para que aporte algo a la empresa?
Básicamente el proceso de desarrollo tiene que ayudarnos a escribir software, tiene que poner las reglas necesarias para alcanzar el éxito en nuestro proyecto, pero dejando la libertad suficiente para no sentirnos agobiados.
Esto no nos lo va a ofrecer ningún proceso estándar y como dice el refrán (aunque no se cumple exactamente en el mundo de la informática) todos los caminos conducen a Roma.

De forma que es tarea de cada empresa, casi para cada proyecto, decidir cuál es el mejor modo de llegar a nuestra meta.

Bibliografía:

Alianza Ágil. Disponible en: http://www.agilealliance.org
Alistair Desarrollo de Software Ágil. Disponible en: http://www.amazon.com/exec/obidos/ASIN/0201699699/programacione-20
Anónimo. Proceso Unificado de Rational para el desarrollo de software. Disponible en: http://www.dybox.cl/metodologia/rup.html. (Mayo 2 del 2005)
Anónimo. Rational Unified Process. Disponible en:http://www.itera.com.mx/itera/productos/fundamentos.asp. (Mayo 2 del 2005)
Anónimo. Seminario sobre RUP en un entorno empresarial de desarrollo. Disponible en: http://www-5.ibm.com/services/learning/es/tairis.nsf. (Mayo 2 del 2005)
Manifiesto para el Desarrollo de Software Ágil. Disponible en: http://www.agilemanifesto.org
Martín, F. La Nueva Metodología. Disponible en: http://www.programacion.net
Patricio, L. Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia. Disponible en: www.upv.es/index-es.html