DESARROLLO ÁGIL DE SOFTWARE APLICANDO PROGRAMACIÓN EXTREMA

Recepción:
Julio 23 de 2012

Aceptado:
Agosto 12 de 2012

Alveiro Rosado Gómez
Msc. en Gestión Aplicación y Desarrollo de Software
Investigador grupo GYTYD
Facultad de Ingenierías
Universidad Francisco de Paula Santander Ocaña
aarosadog@ufpso.edu.co

Alexander Quintero Duarte
Ingeniero de Sistemas
Facultad de Ingenierías
Universidad Francisco de Paula Santander Ocaña
aquinterod@ufpso.edu.co

Cesar Daniel Meneses Guevara
Ingeniero de Sistemas
Facultad de Ingenierías
Universidad Francisco de Paula Santander Ocaña
cdmenesesg@ufpso.edu.co

Resumen

La Programación Extrema o XP (Extreme Programming) pertenece a la familia de las metodologías agiles. XP propone cuatro prácticas esenciales; Entregas limitadas o pequeñas, semana de trabajo de 40 horas, Cliente en el sitio, Programación en Pareja. Este artículo describe la aplicación de XP, en la construcción de un Software para la captura y tabulación de encuestas para el proceso de Autoevaluación de los Programas Académicos de la Universidad Francisco de Paula Santander de Ocaña, para guiar el desarrollo del aplicativo se utilizó cada una de las etapas propuestas por XP, y dentro de ellas se describe como el equipo de trabajo aplicó los conceptos propuestos y los llevo a la práctica, con el objetivo de construir un Software mediante entregas frecuentes, funcionalmente completas, probadas, y con la documentación necesaria. Demostrando de esta forma que no todas las fases proporcionadas por los ciclos de vida tradicionales se adaptan a las necesidades, complejidad y magnitud de diferentes proyectos de desarrollo de Software.

Palabras Claves:

Ciclo de Vida, Manifiesto Agile, Programación Extrema, XP.

Abstract

Extreme Programming or XP belongs to an agile methodologies family. XP has a proposal concerning to four keys practices; small Deliveries, 40 hours working week, customer on-site and programming pair. This article describes the application of XP, in a building software for capture and tabulation surveys for self-evaluation process of Francisco de Paula Santander Ocaña University Academic Programs, to guide the application development one of the steps proposed by XP were used, and within them is described as the team applied the quoted concepts and take them to practice focused of building a quick software, with whole functionally, tested and with the necessary documentation. Thus, demonstrate that not all phases provided by traditional life cycles are adapted to the needs, complexity and magnitude of different software development projects.

Key Words:

Agile Manifiesto, Extreme Programming, life cycles, XP.

Introducción

Para el Instituto de Ingeniería de Software (2010) o SEI (Software Engineering Institute), los métodos agiles son tan robustos y pueden ser utilizados en soluciones tan complejas y dinámicas como las que necesita el Departamento de Defensa de los Estados Unidos. Hoy en día es cada vez más frecuente encontrar desarrollos de software realizados con Métodos Agiles, algunos de ellos buscando procesos más sencillos, rápidos y flexibles, y otros buscando una excusa para generar el mínimo de documentación posible. La Programación Extrema o XP (Extreme Programming) pertenece a la familia de las metodologías agiles, y proporciona una serie de reglas y principios, útiles y sencillos de implementar.

Este artículo trata de mostrar cómo se pueden aprovechar las bondades ofrecidas por XP, en el desarrollo de un aplicativo que permite almacenar y tabular los resultados de diferentes encuestas realizadas por el proceso de autoevaluación, que vienen llevando los programas académicos de la Universidad Francisco de Paula Santander de Ocaña. El documento se encuentra dividido en dos secciones; la primera expone los fundamentos teóricos sobre los cuales se rige XP y la segunda sección describe como esos fundamentos fueron aplicados en el desarrollo del aplicativo.

Desarrollo

Características Generales de la Programación Extrema o XP (Extreme Programming)

El origen de los Procesos o Metodologías Agiles se remonta a Febrero del 2001, en las montañas de Utah, cuando 17 expertos en la industria del Software, cansados de tener problemas con algunas de las metodologías basadas en el Ciclo de Vida Tradicional, en donde el empleo de estrategias de Análisis y Diseño estaban altamente burocratizadas (Bennett, 2006), desicidieron establecer unos principios y valores, adecuados al desarrollo de software para medianas y pequeñas empresas, donde los procesos no se dan a gran escala, de manera que las metodologías ágiles ofrecen un proceso de desarrollo rápido e incremental que se convirtio en alternativa a los procesos de desarrollo de software tradicionales (Highsmith, 2001). Con respecto a esta separación de criterios, Pressman (2010) aconseja no ser excluyente entre procesos agiles e Ingenieria de Software, en vez de pensar de esta forma es mejor definir un enfoque de Ingenieria de Software que sea agil.

La Programación Extrema o XP (Extreme Programming) pertenece a la familia de las metodologías agiles. XP es un enfoque de desarrollo de sistemas que acepta lo que se conoce como buenas prácticas en esta área y las lleva al extremo (Kendall & Kendall, 2005, pág. 94). Al introducirnos en esta metodología cabe resaltar la importancia del cliente, las pruebas, la refactorización [1], la simplicidad, la propiedad colectiva del código que se ven reflejadas en las cuatro prácticas esenciales de XP:

Entregas limitadas o pequeñas: Consiste en realizar entregas parciales de módulos del sistema. Esto no quiere decir que las tareas se dejen inconclusas; las funcionalidades quedan probadas, estables y completas, esta práctica lo que busca es mantener al cliente satisfecho.

Semana de trabajo de 40 horas: Los equipos de desarrollo de XP trabajan de manera intensa durante una semana típica de 40 horas. No admite horas extras, ya que lo que se busca es utilizar al máximo la energía de los desarrolladores.

Cliente en el sitio: Esta práctica insiste en que el cliente debe hacer parte fundamental y activa del grupo de trabajo y debe estar presente durante todo el proceso de desarrollo.

Programación en Pareja: Con esto se busca aumentar la calidad del código, ahorrar tiempo, estimula la creatividad y la reducción de código fuente (Kendall & Kendall, 2005, pág. 170).

Etapas de Desarrollo de XP

La programación extrema engloba un conjunto de reglas (Wells, 1999) que se ejecutan dentro de cuatro actividades estructurales: planeación, diseño, codificación y pruebas. La Figura 1 muestra cada una de las actividades de XP, y resalta las tareas claves de cada una.

[1] Proceso por el cual se cambia un software de forma que mejore la estructura interna, pero que no altere el comportamiento externo del código (Pressman, 2010, pág. 63)

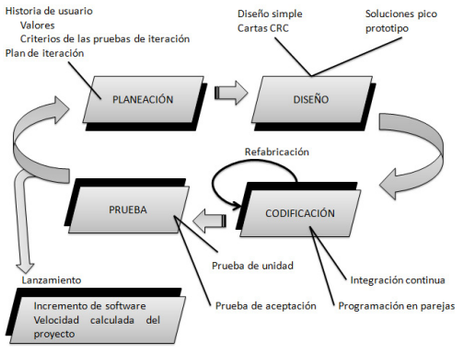

Figura 1. Etapas de Desarrollo de XP (Pressman, 2010).

Planeación

Esta actividad comienza escuchando a los clientes, para entender el contexto del negocio y definir las características principales y funcionalidad que se requiere (Pressman, 2010, pág. 62), estas características se transforman en requerimientos del negocio que se especifican mediante Historias de Usuario; las cuales recogen la interacción hablada entre desarrolladores y usuarios (Wells, user stories, 1999). Una vez hechas las Historias de Usuario, el equipo de desarrollo las divide en tareas, estima el esfuerzo, recursos requeridos para su implementación, se genera el plan de entregas, las iteraciones, la rotación de parejas y las reuniones diarias (SommerVille, 2005).

Diseño

Kendall & Kendall (2005) lo definen como la etapa en donde son evaluadas las historias de usuario por el equipo del proyecto para dividirlas en tareas, cada tarea representa una característica distinta del sistema y se puede diseñar una prueba de unidad que verifique cada tarea (SommerVille, 2005), estas tareas se representan por medio de las tarjetas CRC (Clase-Responsabilidad-Colaborador). Las tarjetas CRC identifican y organizan las clases bajo el paradigma orientado a objetos (lo que incluye asignación de responsabilidades), cada tarjeta contiene el nombre de la clase (que representa una o más historias de usuario), una descripción de las responsabilidades o métodos asociados con la clase, así como la lista de las clases con que se relaciona o que colaboran con ella. Las tarjetas CRC son el único trabajo de diseño que se genera como parte del proceso de XP (Holmes & T. Joyce, 2000).

Desarrollo

Se lleva a cabo la programación en pareja, la unidad de pruebas y la integración del código (Kendall & Kendall, 2005). Durante esta etapa se espera la disponibilidad del cliente para que éste pueda resolver cualquier duda que se presente durante una jornada de trabajo.

Pruebas

Cada tarea que se identificó con las historias de usuario, representa una característica distinta del sistema y se realiza una prueba de unidad por cada una de ellas, existen pruebas unitarias las cuales son diseñadas para probar cada uno de los métodos y clases, dichas pruebas son realizadas por los programadores (SommerVille, 2005).

Desarrollo Caso de Estudio

Dado el volumen de información que solicita el proceso de autoevaluación a cada programa académico de la Universidad; en donde se necesitaba aplicar y tabular los resultados de diferentes encuestas realizadas a 18 Programas Académicos, con una muestra de estudiantes activos de 700, 43 Administrativos, 126 Docentes y 97 Egresados. Los resultados obtenidos permiten generar un diagnóstico sobre la percepción que se tiene respecto a factores que fomentan el mejoramiento continuo, la calidad de los programas académicos. El proceso de autoevaluación es aplicado de conformidad con el modelo propuesto por el Consejo Nacional de Acreditación (CNA), el cual establece que una vez identificadas las falencias, se deben consolidar en un documento las acciones que eliminen o disminuyan las debilidades mediante la creación de un plan de mejora.

Según las características del software solicitado, dada la premura de tiempo en que se debería construir, según los requisitos de alto nivel suministrados y con el fin de aprovechar los beneficios que proporciona los procesos de construcción rápida de software, se tomó la decisión de utilizar un modelo de desarrollo de tipo ágil sobre los modelos tradicionales; dado que los métodos agiles permiten la implementación del software con el mínimo de documentación, en el menor tiempo posible y teniendo al cambio como una constante.

Para el Instituto de Ingeniería de Software (2010) o SEI (Software Engineering Institute), sugiere que dentro de la planeación de un proyecto de desarrollo de Software, se debe estimar los recursos humanos, tecnológicos y las herramientas a utilizar; no todas las aplicaciones necesitan el mismo modelo de desarrollo, depende de los requisitos que se determinen en la etapa inicial y de la rapidez y complejidad de las funcionalidades, es por eso que no todos los ciclos de vida de desarrollo de software se ajustan a todas las necesidades de proyectos diferentes, la metodología a seguir debe permitir realizar correcciones sobre el alcance, las necesidades, restricciones y costos del proyecto.

Aplicación de las Etapas de Desarrollo de XP

Planeación

Antes de comenzar a realizar las Historias de Usuario, es fundamental definir los roles que participaran en el desarrollo de la aplicación. Los roles que se definieron para la construcción del aplicativo son: Programador, Cliente, Probador, Entrenador, Consultor y Jefe (Baird, 2003, pág. 79).

El rol de Cliente fue tomado por la funcionara encargada de la Coordinación del proceso de Autoevaluación dentro de la Universidad, ella tiene como responsabilidades definir las características del sistema y el orden de las prioridades en que deben ser liberadas las funcionalidades; adicionalmente debe aportar en la redacción de Historias de Usuario.

Los roles Programador y Probador fueron asumidos por dos estudiantes y un egresado del programa de Ingeniería de Sistemas, como Programadores ellos tienen destrezas para comprender diseños y codificación de lenguaje de Pre-procesador de hipertexto o PHP (Hypertext Preprocessor). Ellos están orientados a trabajar en compañía, con disposición al cambio y comprometidos con la realización pruebas unitarias al código. Desde el punto de vista de Probador, deben definir en conjunto con el cliente las pruebas funcionales necesarias para validar las entregas programadas en una iteración.

El Director del Departamento de Sistemas e Informática, fue el encargado de los roles Entrenador, Consultor y Jefe. Como Entrenador, debe de estar pendiente de los detalles y de motivar, controlar y acompañar a los programadores en su tarea. Como Consultor, debe estar disponible a resolver problemas de tipo técnico, a proponer mejoras de rendimiento dentro del código, mantener la calidad de las funcionalidades y hacer respetar el diseño planteado.

Después de varias reuniones, de dos (2) horas, durante una semana, entre el Cliente, el Jefe y los Programadores se definieron las Historias de Usuario, a ellas se les asigno su respectiva prioridad, en total se especificaron dieciocho (18) Historias de Usuario, las cuales se agruparon en los módulos de configuración, captura y reportes; dentro de configuración se encuesta la gestión de usuarios del sistema y de los procesos de autoevaluación, en el módulo de captura se realiza todo el proceso de registro de las encuestas realizadas (estudiante, docente, egresado y administrativo), y en reportes se puede observar los consolidados, por todos los programas, y por cada uno de los programas, agrupados en las cuatro categorías de encuestados; los valores son mostrados en cuatro tipos diferentes de reportes, promedio obtenido por ítem (la media obtenida por cada una de las preguntas), promedio obtenido por factor (la media de las respuesta de cada encuestado agrupados por factor), porcentaje por ítem (porcentaje por cada una de las respuestas), valor general (calificación total por promedio para cada encuestado).

La prioridad de entrega de cada conjunto de funcionalidades se organizó de la siguiente manera; la liberación inicial contemplo el módulo de captura de encuestas; de él depende la generación de los reportes, la segunda iteración adiciona el módulo de reportes, para que se puedan generar los consolidados y de ultimo se liberó el módulo de configuración, debido a que las responsabilidades que él tiene podían ser asumidas inicialmente por el Administrador de la

Base de Datos o DBA (Database Administrator), quien gestionaba los usuario y fechas del proceso de autoevaluación directamente sobre la Base de Datos.

Diseño

Las tarjetas CRC que se definieron para el desarrollo fueron trece (13), las cuales cumplen con los parámetros sugeridos por Wells (1999), donde cada tarjeta corresponde a una o más tareas de una historia de usuario, y adicionalmente se le integraron más clases que implementan los Patrones de Asignación de Responsabilidades o GRASP (General Responsibility Assignment Software Patterns), lo que permite mejorar los resultados de la asignación de alcance y nivel de colaboración a las clases (Larman, 2002).

Desarrollo

Para la implementación de la aplicación se utilizó una arquitectura en tres niveles; interfaz, lógica de aplicación y almacenamiento. Como interfaz gráfica de usuario se implementaron paginas en lenguaje de programación Web PHP, que solo se encargan de pintar las etiquetas HTML de las vistas, como lógica de la aplicación se utilizó PHP Orientado a Objetos, como mecanismo de almacenamiento se empleó el motor de base de datos PostgreSQL versión 8.4. Para garantizar que no exista un acoplamiento [2] directo entre la interfaz y la lógica de aplicación se implementó el principio de separación Modelo-Vista; el cual propone que los objetos del modelo (lógica de aplicación) no deben conocer directamente los objetos de la vista (interfaz de usuario). En otras palabras lo que se pretende es separar la lógica de la aplicación de las páginas con las que interactúa el usuario de los procesos de toma de decisiones, accesos y escrituras en la bases de datos (Larman, 2002, pág. 441).

Pruebas

En el desarrollo de la aplicación se utilizó una herramienta de pruebas unitarias llamada PHPUnit, allí se realizan casos de prueba, estos consisten en hacer pruebas a una clase relacionada con alguna funcionalidad (Team, 2011). Ya que la realización de pruebas puede tardar mucho tiempo, solo se realizaron pruebas a las funcionalidades de la clase Experto de Información (Larman, 2002), considerada relevante dentro de la aplicación.

La realización de las pruebas resulto de forma satisfactoria; aunque algunas pruebas no resultaron adecuadas, es decir, de alguna manera se evidenciaron algunos errores, su corrección ayudó a la depuración del código, haciéndolo sostenible para futuros cambios en las funcionalidades del sistema.

Conclusiones

Seleccionar un ciclo de vida para desarrollar software, es una tarea que afecta directamente el éxito de un proyecto, no todas las fases proporcionadas por los ciclos de vida se adaptan a las necesidades del desarrollo; dependiendo del conocimiento que se tenga sobre los requisitos de los usuarios y los tiempos de entrega de las funcionalidades, se debe evaluar y seleccionar cual es el ciclo de vida que mejor se adapta a las características del proyecto.

Los procesos Agiles deben utilizarse preferiblemente solo en proyectos en donde se necesiten resultados rápidos, se cuente con poco personal, sea soportado con la mínima documentación y se esté dispuesto a seguir las fases del ciclo de vida que la metodología propone.

Aplicar XP en la creación del software de captura de información; permitió la organización de las entregar las funcionalidades, de forma incremental. De manera que primero se liberó y utilizo el modulo de captura de información sin tener el software completamente terminado.

Referencias Bibliográficas

Baird, S. (2003). Sams teach yourself extreme programming in 24 hours. United States of America: Sams Publishing.

[2] Una dependencia entre elementos (como clases, paquetes, subsistemas), típicamente resultado de la colaboración entre los elementos para proporcionar un servicio (Larman, 2002).

Bennett, S., McRobb, S., & Farmer, R. (2006). Analisis y Diseno Orientado a Objetos de Sistemas. Madrid: McGraw-Hill.

Highsmith, J. (11 de Febrero de 2001). History: The Agile Manifesto. Recuperado el 15 de Septiembre de 2011, de http://www.agilemanifesto.org/history.html

Holmes, B., & T. Joyce, D. (2000). Object-oriented programming with Java. Sudbury: Jones and Bartlett Publishers.

Kendall, K. E., & Kendall, J. E. (2005). Análisis y diseño de sistemas. Sexta edición. México: Pearson Educación.

Lapham, M. A., Williams, R., Hammons, C., Burton, D., & Schenker, A. (Abril de 2010). Software Engineering Institute. Recuperado el 2012 de Enero de 20, de Considerations for Using Agile in DoD Acquisition: http://www.sei.cmu.edu/reports/10tn002.pdf

Larman, C. (2002). UML y Patrones. Madrid: Pearson Educacion, S.A.

Pressman, R. (2010). Ingenieria del Software un Enfoque Practico. Mexico, D.F: McGraw-Hill.

Program, S. E. (2010). CMMI for Development, Version 1.3. CMU/SEI-2010-TR-033.

SommerVille, I. (2005). Ingenieras de Software Séptima edición. Madrid: Pearson Educación.

Team, P. W. (2011). pear. Recuperado el 3 de Septiembre de 2011, de PHPUnit: http://pear.php.net/package/PHPUnit/redirected

Wells, D. (1999). CRC Cards. Recuperado el 3 de Septiembre de 2011, de CRC Cards: http://www.extremeprogramming.org/rules/crccards.html

Wells, D. (1999). The Rules of Extreme Programming . Recuperado el 2 de Septiembre de 2011, de The Rules of Extreme Programming : http://www.extremeprogramming.org/rules.html

Wells, D. (1999). user stories. Recuperado el 3 de Septiembre de 2011, de user stories: http://www.extremeprogramming.org/rules/userstories.html