Universidad Autónoma "Gabriel Rene Moreno"

INF513-SC: Tecnologia Web

Desarrollo Ágil

Docente:
Ing. Evans Balcazar Veizaga M.Sc.

Alumno:
Mauricio Elian Delgadillo Garcia (Cod. 217015689)

Santa Cruz - Bolivia

Índice

1. Programación Extrema (XP)

1.1. Origen de la Programación Extrema

Nace de la mano de Kent Beck en el verano de 1996, cuando trabajaba para Chrysler Corporation. Él tenía varias ideas de metodologías para la realización de programas que eran cruciales para el buen desarrollo de cualquier sistema. Las ideas primordiales de sus sistemas las comunico en las revistas C++ Magazine en una entrevista que esta le hizo el año 1999.

1.2. Que es la Programación Extrema?

Es una Metodología ligera de desarrollo de aplicaciones que se basa en la simplicidad, la comunicación y la realimentación del código desarrollado.

1.3. Objetivos de la Programación Extrema

- La Satisfacción del cliente.
- Potenciar el trabajo en grupo.
- Minimizar el riesgo actuando sobre las variables del proyecto: costo, tiempo, calidad y alcance.

1.4. Características

1. Metodología basada en prueba y error para obtener un software que funcione realmente.
2. Fundamentada en principios.
3. Esta orientada hacia quien produce y usa software (el cliente participa muy activamente).
4. Reduce el coste del cambio en todas las etapas del ciclo de vida del sistema.
5. Combina las que han demostrado ser las mejores practicas para desarrollar software y las lleva al extremo.
5. Cliente bien definido.
6. Los requisitos pueden cambiar.
7. Grupo pequeño y muy integrado (2-12 personas)
8. Equipo con formación elevada y capacidad de aprender.

1.5. Herramientas de la Metodología XP

1.5.1. Historia de Usuario

Las Historias de Usuario representan una breve descripción del comportamiento del sistema, se realizan por cada característica principal del sistema y son utilizadas para cumplir estimaciones de tiempo y el plan de lanzamientos, así mismo reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación.

Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarlas en unas semanas.

La Plantilla a utilizarse para la elaboración de las historias de usuario se muestra en la siguiente tabla y cada uno de sus componentes se explica a continuación.

HISTORIA DE USUARIO	
Número: Permite identificar a una historia de usuario.	**Usuario:** Persona que utilizará la funcionalidad del sistema descrita en la historia de usuario.
Nombre Historia: Describe de manera general a una historia de usuario.	
Prioridad en Negocio: Grado de importancia que el cliente asigna a una historia de usuario.	**Riesgo en Desarrollo:** Valor de complejidad que una historia de usuario representa al equipo de desarrollo.
Puntos Estimados: Número de semanas que se necesitará para el desarrollo de una historia de usuario.	**Iteración Asignada:** Número de iteración, en que el cliente desea que se implemente una historia de usuario.
Programador Responsable: Persona encargada de programar cada historia de usuario.	
Descripción: Información detallada de una historia de usuario.	
Observaciones: Campo opcional utilizado para aclarar, si es necesario, el requerimiento descrito de una historia de usuario.	

1.5.2. Tareas de Ingenierías (Task Card)

Una Historias de Usuario se descompone en varias tareas de ingeniería, las cuales describen las actividades que se realizarán en cada historia de usuario, así mismo las tareas de ingeniería se vinculan más al desarrollador, ya que permite tener un acercamiento con el código.

La Plantilla a utilizarse para la elaboración de las tareas de ingeniería se muestra en la siguiente tabla y cada uno de sus componentes.

TAREA DE INGENIERÍA	
Número de Tarea: Permite identificar a una tarea de ingeniería.	**Número de Historia:** Número asignado de la historia correspondiente.
Nombre de Tarea: Describe de manera general a una tarea de ingeniería.	
Tipo de Tarea: Tipo al que corresponde la tarea de ingeniería.	**Puntos Estimados:** Número de días que se necesitará para el desarrollo de una tarea de ingeniería.
Fecha Inicio: Fecha inicial de la creación de la tarea de ingeniería.	**Fecha Fin:** Final concluida de la tarea de ingeniería.
Programador Responsable: Persona encargada de programar la tarea de ingeniería.	
Descripción: Información detallada de la tarea de ingeniería.	

1.5.3. Pruebas de Aceptación

Las Pruebas de aceptación son de vital importancia para el éxito de una iteración y el comienzo de la siguiente, con lo cual el cliente puede conocer el avance en el desarrollo del sistema y a los programadores lo que les resta por hacer.Además permite una retroalimentación para el desarrollo de las próximas historias de usuarios a ser entregadas. Estas son comúnmente llamadas pruebas del cliente, por lo que son realizadas por el encargado de verificar si las historias de usuarios de cada iteración cumplen con la funcionalidad esperada.

La Plantilla a utilizarse para la elaboración de las pruebas de aceptación se muestra en la siguiente tabla y a continuación se definen cada uno de los componentes.

PRUEBAS DE ACEPTACIÓN	
Código: Nº Único, permite identificar la prueba de aceptación.	**Nº Historia de Usuario:** Número único que identifica a la historia de usuario.
Historia de Usuario: Nombre que indica de manera general la descripción de la historia de usuario.	
Condiciones de Ejecución: Condiciones previas que deben cumplirse para realizar la prueba de aceptación.	
Entrada/Pasos de Ejecución: Pasos que siguen los usuarios para probar la funcionalidad de la historia de usuario.	
Resultado Esperado: Respuesta del sistema que el cliente espera, después de haber ejecutado una funcionalidad	
Evaluación de la Prueba: Nivel de satisfacción del cliente sobre la respuesta del sistema. Los niveles son: Aprobada y No Aprobada.	

1.5.4. Tarjetas CRC (Clase - Responsabilidades - Colaboradores)

Las Tarjetas CRC (Clase-Responsabilidades-Colaboradores), permiten conocer que clases componen el sistema y cuales interactúan entre sí. Se dividen en tres secciones:

Nombre de la Clase, Responsabilidades y Colaboradores.

La Plantilla a utilizarse para la elaboración de las Tarjetas CRC se muestra en la siguiente Tabla y a continuación se describen cada uno de los componentes.

TARJETAS CRC	
Nombre de la Clase: Nombre de la clase al cual hace referencia la tarjeta.	
Responsabilidades: Atributos y operaciones de la clase.	**Colaboradores**: Clases que colaboran con la clase citada en la tarjeta.

2. SCRUM

2.1. Qué es SCRUM?

SCRUM es una metodología ágil para el desarrollo de software o la gestión de proyectos. Antes de la definición de SCRUM, es imprescindible entender el concepto de ágil.

Generalmente, el desarrollo de software es una actividad caótica, a menudo se caracteriza por la frase "código y arreglar". El problema del desarrollo de software reside en que el código se escribe muchas veces sin seguir un plan subyacente, y el propio diseño del sistema se improvisa a partir de muchas decisiones a corto plazo. De hecho, esto funciona bastante bien cuando el sistema es pequeño,pero conforme el sistema va creciendo, se va haciendo más difícil añadir nuevas funcionalidades. Además, también se hacen predominantes los bugs y aumenta la dificultad de arreglarlos. Para evitarlos, es necesario una larga fase de pruebas una vez que se han definido todas las funciones del sistema. Esta fase causa estragos en los cronogramas, ya que la realización de pruebas y la depuración es imposible de planificar.

El movimiento original de intentar cambiar esto, introdujo la noción de metodología. Estas metodologías imponen un proceso disciplinado sobre el desarrollo de software con el objetivo de elaborar un software más predecible y eficiente. Esto se consigue desarrollando un proceso detallado con especial énfasis en la planificación, inspirada en otras disciplinas de la ingeniería -también llamadas metodologías de la ingeniería.

Actualmente hay muchas metodologías ágiles en uso, pero ha sido necesario el uso de esta metodología para el desarrollo del presente proyecto. Por lo tanto, es imprescindible hablar de Scrum antes de meterse en la implementación del mismo.

2.2. Qué caracteriza Scrum?

De todas las metodologías ágiles, Scrum es única porque introduce la idea del control empírico de los procesos. Esto significa que Scrum utiliza el progreso real de un proyecto para planificar y concertar los lanzamientos. En Scrum, los proyectos se dividen en ritmos de trabajo breves, conocidos como sprints. Normalmente, tienen una, dos o tres semanas de duración. Al final de cada sprint, el cliente y los miembros del equipo se reúnen para evaluar el progreso del proyecto y planear los siguientes pasos a seguir. Esto permite que la dirección del proyecto se ajuste o se reoriente una vez finalizado el trabajo, sin especulaciones ni predicciones

Filosóficamente, este énfasis continuo de evaluar las tareas finalizadas es el principal causante del éxito que tiene esta metodología entre los directores y desarrolladores. Pero lo que verdaderamente permite funcionar a la metodología Scrum es un conjunto de papeles, responsabilidades, y reuniones.

Cada uno de los puntos hacen que SCRUM sea utilizado de manera regular en un conjunto de buenas prácticas para el trabajo en equipo y de esa manera obtener resultados posibles

- Gestión regular de las expectativas del cliente, resultados anticipados, flexibilidad y adaptación, mitigación de riesgos, productividad y calidad, o equipo motivado
- Se hace uso de equipos y auto-organizados
- Se realiza a diario una reunión de Scrum, que es una reunión de avance diaria que no dura más de 15 minutos con el objetivo de obtener reglamentación sobre las tareas del equipo y los obstáculos que se presenten

2.3. Los papeles de Scrum

Scrum tiene tres papeles fundamentales: Product Owner (propietario del producto), Scrum Master (especialista en Scrum) y Team Member (miembros del equipo)

1. **Product Owner:** En Scrum, el Product Owner se encarga de comunicar la visión del producto al equipo de desarrollo. Él/ella también deben representar el interés del cliente por medio de los requisitos y la priorización. Como el Product Owner es el que más autoridad tiene de los tres papeles en Scrum, también es el que mayor responsabilidad recibe. En otras palabras, el Product Owner es el individuo que tiene afrontar las consecuencias cuando unproyecto va mal.
2. **Scrum Master:** El Scrum Master actúa como enlace entre el Product Ow-

ner y el equipo. El Scrum Master no dirige al equipo. Él/ella se encarga de evitar cualquier barrera que impida al equipo lograr sus objetivos de sprint. En resumen, este papel hace que el equipo sea creativo y productivo, a la vez que permite que los logros del equipo sean visibles ante el Product Owner. El Scrum Master también aconseja al Product Owner sobre cómo maximizar el ROI (Return Of Investment)para el equipo.

3. **Team Member:** En la metodología Scrum, el equipo es el responsable de terminar el trabajo. Idealmente, los equipos están formados por siete miembros multifuncionales, más/menos dos personas. Para proyectos de software, un equipo habitual sería una mezcla de ingenieros de software, arquitectos, programadores, analistas, testers, y diseñadores de UIs. En cada sprint, el equipo es responsable de determinar cómo va a lograr acabar el trabajo. Esto garantiza al equipo un grado de autonomía, pero, al igual que pasa con la situación del Product Owner, esta libertad viene acompañada por la responsabilidad de cumplir los objetivos del sprint.

2.4. Proyectos de Aplicación

No es necesario, ni siquiera conveniente, utilizar la metodología SCRUM en todo tipo de proyectos. Por eso debes saber qué requerimientos debe tener tu proyecto si quieres utilizarla con eficiencia.

* Equipos pequeños, cuando en tus proyectos los equipos de trabajo no superan las 8 personas
* Poca necesidad de documentación, Si el cliente te exige que todo el proyecto esté muy bien documentación desde el principio SCRUM no es la metodología adecuada.
* Proyectos con riesgos de cambios durante el proceso
* Confianza en la metodología, el SCRUM es el encargado de que se cumpla