FACULTAD DE TECNOLOGIAS DE INFORMACION Y COMUNICACION

ESCUELA DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

Tarea 1

BSI – 090-2 PROGRAMACION III

Profesor

Jorge Vásquez

Alumnos

Carlos Gonzalez Segura

San José, 21 Mayo 2018

Metodologías para el desarrollo de software

Metodología en cascada: Framework lineal.

Es la metodología más antigua. Básicamente no podrás avanzar a la siguiente fase, si la anterior no se encuentra totalmente terminada, pues no tiene por qué haber vuelta atrás. Se mantiene cada fase bien documentada, ayudando con esto a todo el equipo que interviene en el desarrollo.

Fases:

1. **Análisis de Requisitos**. Se realiza la documentación, las funciones del software al terminar su desarrollo.
2. **Diseño del Sistema.** Se crea la estructura y especificaciones del sistema.
3. **Diseño del Programa.** Se realiza los algoritmos.
4. **Codificación.** Inicia a escribir todo el código que será necesario para el desarrollo del software.
5. **Ejecución de Pruebas.** Se ejecutan todas las pruebas de funcionamiento del software, provocando fallas antes de que el usuario final lo utilice.
6. **Verificación.** Ejecución del Software por parte del usuario final.
7. **Mantenimiento.** Solucionan errores, se quitan algunos bugs, se añaden funcionalidades, todo después de que el usuario final ya lo ha probado y utilizado en su fase final.

Método de Prototipos

Uno de los métodos favoritos. En base a los requerimientos y necesidades del cliente se realiza un prototipo del software; una versión preliminar. El cliente se involucra más,

Fases:

1. **Planeación.** Rápida, ya que solo se debe presentar un prototipo.
2. **Modelado**. Al igual que la fase 1, es rápida ya que se debe presentar una breve solución.
3. **Elaboración del Prototipo.** Fase de desarrollo preliminar del prototipo.
4. **Desarrollo**. Continuación del prototipo, pero hacia un producto final.
5. **Entrega y Retroalimentación.** Entrega del producto final y capacitación sobre su uso.
6. **Comunicación con el Cliente.** Retroalimentación con el cliente sobre el software ya trabajando, para saber si requiere de mejoras.
7. **Entrega del Producto Final.**

Modelo Incremental o Iterativo y Creciente

Se obtiene un producto final mucho más completo y exitoso. Es una combinación de los modelos lineal e iterativo o bien, modelo de cascada y prototipos. Se crean mini modelos de desarrollo de software en cascada, permitiendo que el usuario vaya de la mano durante su desarrollo.

Fases:

1. **Inicialización**. Se obtienen algunas ideas y requisitos para iniciar.
2. **Periodos de Iteración.** Se inicia la interacción con el cliente, para poder llevar a un productos final exitoso.
3. **Lista de Control.** Control sobre cada interacción, llevando documentado cada cambio o nuevas versiones, por si se necesita regresar a versiones más estables o funcionales si el usuario así lo solicita.

Modelo en Espiral

Combinación entre el modelo lineal o de cascada y el modelo iterativo o basado en prototipos, con gestión de riesgos. Las fases se realizan en forma de espiral, pero no son obligatorios y no llevan un orden establecido. Como ejemplo, es el modelo utilizado para desarrollar un sistema operativo.

Fases:

1. **Determinar Objetivo.** Se debe iniciar con una planificación, sin embargo, es un proceso que se ira realizando constantemente según el avance.
2. **Análisis de Riesgo.** Lluvia de ideas de todo lo que pueda dañar tu sistema, manteniendo un Plan B para buscar una solución en caso de algún problema. Se crean prototipos, para mantener un control de versiones para respaldo.
3. **Desarrollar, Validar y Probar.** Dependerá del análisis de riesgos, se generan nuevas versiones derivadas del análisis de riegos, hasta que no se vea solucionado un riesgo no se continua con el desarrollo del producto final.
4. **Planificación**. Sirve para determinar el avance de nuestro proyecto y indicar hacia donde vamos a dirigirnos en la próxima iteración.

Modelo RAD: Desarrollo Rápido de Aplicaciones (Rapid Application Development)

No cuenta con una serie de fases ordenadas, aunque está basada en lo que es el modelo de cascada y la creación de prototipos, sin embargo el proceso es muy independiente a contar con ciertas fases estipuladas.

Basado en el uso de las iteraciones y principalmente en el manejo de prototipos. Hace uso de las Herramientas CASE, las cuales permitirán acelerar el proceso considerablemente.

Se subdivide en pequeñas secciones, las cuales se irán optimizando y de esta forma se irá avanzando mucho más rápido.

Una de las ventajas, es que el enfoque y las prioridades van hacia la fase de desarrollo, a diferencia de modelos como el espiral, que se enfoca en que los riesgos al momento sean mucho menores. Acá con el RAD, se hace lo contrario, si hay riesgos reducimos los requerimientos para reducir los riesgos, no como en el espiral, que entre más riesgos, más requisitos aportamos para que se incremente. Acá la idea es reducir tiempos y no riesgos, o si, tal vez también, pero la reducción de riesgos es totalmente inversa al modelo espiral.

Por supuesto, como en todos los modelos, vas a requerir de ciertos factores documentados, de preferencia todo lo que se pueda.

Metodologías ágiles de desarrollo de software

- Crear software de una forma más rápida, menos documentación. El enfoque del desarrollo ágil nos dice, que es mejor formar primero un buen equipo de trabajo y posteriormente entre ellos vayan creando su propio entorno. Existe una regla: “No producir documentación, al menos que sean sumamente necesario al momento para tomar una decisión”.
- Existen dos elementos fundamentales para que un nuevo miembro del equipo se ponga al día. El primero es el código fuente, pues suponiendo que es una persona conocedora, sabrá hacia donde va el curso del proyecto con tan solo leer el código. La segunda es que la interacción con el equipo de trabajo, será el complemento ideal para que se acople al proyecto. De este modo nos olvidamos de la extensa documentación para un software que al final del día debe estar totalmente funcional.
- Colaboración con el Cliente en lugar de hacer Contrato. Una de las cosas importantes dentro de las metodologías ágiles. Es que cambia el modo en que se trabajaba con el cliente anteriormente.
- • Posibilidad de Hacer cambios de planes a medio proyecto.

Metodología Scrum

Amigable y fomenta lo que es el trabajo en equipo.

Para que tengas una idea rápida, para que un proyecto ingrese al marco de lo que es el modelo Scrum, debe contar con las siguientes características:

• **Desarrollo Incremental.** Una metodología ágil sin desarrollo incremental, no puede ser considerada Scrum. Con incremental hago énfasis a olvidarnos de la planeación y de la ejecución de las líneas sin salirnos de lo pre establecido, pues con una metodología Scrum, el desarrollo se irá incrementando poco a poco, sin importar el orden en el cual se lleven a cabo los procesos.

• **Calidad de las personas.** La calidad dependerá de las personas, la auto organización y el conocimiento de los equipos de trabajo.

• **Adiós al Secuencial y Cascada**. No importa en que proceso te encuentres, si un proceso necesita ser trabajado, vuelves a el para realizar lo que tienes .

• **La comunicación es Fundamental.** Constante comunicación con los otros equipos de trabajo.

Procesos :

• **Product Backlog**. Lista de las funcionalidades del producto a desarrollar. Debe estar ordenado de acuerdo a las prioridades del sistema de mas a menos, con la idea de que las cosas con mayor prioridad sean las que se realicen antes de cualquier cosa. De forma concreta, digamos que el objetivo base del Product Owner, es que nos de respuesta a la pregunta "¿Qué hay que hacer?".

• **Sprint Backlog.** Seleccionar algunos de los puntos escritos en el Product Backlog, y marcar el tiempo en que se llevará a cabo el Sprint.

• **Sprint Planning Meeting.** Reunión que se realiza para definir plazos y procesos a efectuarse para el proyecto establecido en el Product Backlog. Algo importante que debes saber, es que cada Sprint, se compone de diversos features, que no son otra cosa más que procesos o subprocesos que se deben realizar.

• **Daily Scrum o Stand-up Meeting.** Reuniones diarias mientras se está llevando a cabo un Sprint, para responder las siguientes preguntas: ¿Que hice ayer?, ¿Qué voy a hacer hoy, ¿Qué ayuda necesito?. Aquí entra en función el Scrum Master (encargado de determinar la solución de los problemas y cada complicación que suceda).

• **Sprint Review.** Revisión del Sprint terminado y para este punto ya tendría que haber algo que mostrarle al cliente, algo realmente visual o tangible para que se pueda analizar un cierto avance.

• **Sprint Retrospective.** Permite al equipo analizar los objetivos cumplidos, si se cometieron errores, visualizarlos y tratar de no cometerlos nuevamente mas adelante. Básicamente también sirve este proceso para lo que son la implementación de mejoras.

Equipos que Componen los Procesos Scrum

• **Product Owner.** Líder de proyecto. Sera los ojos del cliente, será la persona encargada del proyecto y que se cumpla las expectativas de lo que se espera.

• **Scrum Master**. Lider de cada una de las reuniones. Debe conocer el lenguaje o lenguajes que se manejan en el proyecto.

• **Scrum Team.** Equipo de desarrollo, encargado de lo que es la codificación del software y de cumplir los objetivos o metas propuestas por el Product Owner.

• **Cliente**. Tiene la capacidad para influir en el proceso, debido a que siempre estará empapado de el, ya sea que proponga nuevas ideas o bien haciendo algún tipo de comentario.

Metodología Kanban

Metodología Japonesa, la cual consiste en ir etiquetando con tarjetas cada uno de los procesos que se deben llevar a cabo, también se le ha denominado como "Un sistema de producción de alta efectividad y productividad".

Principios básicos:

• **Garantía de Calidad.** Promueve la calidad antes que la velocidad.

• **Desperdicios.** Basado en lo que es el principio YAGNI, ofrece una mayor calidad en el producto, optimizando tiempos y costos.

• **Mejora Continua.**

• **Es Flexible.**

¿Cómo configurar una estrategia Kanban?

1. Definir el Flujo de Trabajo.

2. Fases del Ciclo de Producción. Dividir los procesos en pequeños segmentos. Es por eso que en los post-it que vayas colocando con cada proceso, deberá ser colocado el número de horas requeridas para completarlo, al finalizar las horas se determinará el porque no se ha terminado o bien ya se habrá avanzado a otra fase.

3. Stop Starting, start finishing. "No se empieza una nueva tarea, hasta terminar la otra".

4. Tener un Control. Control del flujo de trabajo.

Metodología XP

La mas destacada de las metodologías ágiles y esto se debe a su gran capacidad de adaptación ante cualquier tipo de imprevisto que surja.

Valores de la Metodología XP

• **Comunicación**. El cliente tiene una gran intervención, pero obviamente la comunicación dependerá de mas factores. De igual forma, se deben documentar las cosas mas relevantes, independientemente de que sean comentadas en el código, pero es importante tener un documento extra para explicaciones extensas, de lo contrario el código se verá infestado de escrito.

• **Simplicidad**.

• **Retroalimentación**.

• **Valentía**.

• **Respeto**. Importante para que haya una buena comunión entre los programadores del equipo.

Características que componen la metodología XP

• **Tipo de Desarrollo Iterativo e incremental.** Basado en lo que son las mejoras continuas, a base de iteraciones y por supuesto un desarrollo incremental al estilo espiral.

• **Pruebas Unitarias.** Analiza el código y solucionan errores, antes de validarlo y darlo por terminado.

• **Trabajo en Equipo.**

• **Alguien del equipo trabaja con el cliente.**

• **Corrección de Errores.**

• **Reestructuración del Código.** Simplificar el código pero no las funciones.

• **El Código es de todos.** La idea es que si uno no detecta un error, otro lo podrá hacer, por eso el código fuente es compartido entre todos.

• **Código simple es la clave.**

Equipo de Trabajo dentro de una Metodología XP

• **Programador.** Encargado del código del sistema y además de hacer las pruebas unitarias.

• **Tester**. Encargado de las pruebas del desarrollo.

• **Tracker**. Encargado de realizar las comparaciones entre los tiempos estimados antes de empezar un desarrollo y los tiempos reales que se obtuvieron

• **Entrenador**. Responsable del proyecto básicamente y precisamente hace las funciones de un entrenador. Se encarga de guiar al equipo por el camino que deben seguir.

• **Consultor.** Es externo, pero que cuenta con conocimientos específicos y que será capaz de ayudar en la solución de problemas.

• **Gestor**. Encargado de vincular e interrelacionar al cliente con los programadores.

Comparativa MyISAM vs InnoDB

Características MyISAM:

Se establece por defecto cuando se crea una tabla, salvo que se indique lo contrario.

Soporta transacciones.

Realizar bloqueo de registros.

Soporta un gran número de consultas SQL, lo que se refleja en una velocidad de carga muy rápida para nuestra web.

Desventaja, no realiza bloqueo de tablas.

Características InnoDB:

Bloqueo de registros. Importante para accesos múltiples al mantenimiento de tablas, es decir, ejecuciones de sentencias tipo INSERT o UPTATE, éstas ejecuciones tienen una velocidad optimizada.

Capacidad para soportar transacciones e integridad de datos, es decir previene el alta de datos no adecuados.

Aplica las características propias de ACID (Atomicity, Consistency, Isolation and Durability), consistentes en garantizar la integridad de las tablas.

Desventaja, reduce el rendimiento en velocidad para desarrollo que requieren de un elevado número de consultas.

Recomendaciones o ejemplos:

Casos:

a. Un solo gestor de mantenimiento para una plataforma que requerirá muchas consultas o visitas: **MyISAM**
b. Velocidad y mínimo consumo de recursos en servidor, espacio, RAM, etc.: **MyISAM**
c. Varios o muchos gestores de mantenimiento: **InnoDB**
d. Desarrollo donde se prioriza el diseño relacional de bases de datos: **InnoDB**