UNIVERSIDAD AUTONOMA DE SAN LUIS POTOSI.

FACULTAD DE INGENIERIA.

AREA: CIENCIAS DE LA COMPUTACION.

INVESTIGACION: Metodologías ágiles para el desarrollo de software.

Alumnos: Daniel Omar Torres Carbajal.

Raúl González González.

Mauricio Alemán Páez.

Metodologías ágiles para el desarrollo de software.

Introducción:

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte, tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos.

Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo, pero manteniendo una alta calidad.

PROGRAMACIÓN EXTREMA (EXTREME PROGRAMMING, XP).

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.

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck, el padre de XP, describe la filosofía de XP.

A continuación, se presentarán las características esenciales de XP organizadas en los tres apartados siguientes: historias de usuario, roles, proceso y prácticas.

Historias de usuario:

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

No hay que preocuparse si en un principio no se identifican todas las historias de usuario. Al comienzo de cada iteración estarán registrados los cambios en las historias de usuario y según eso se planificará la siguiente iteración. Las historias de usuario son descompuestas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración.

Roles XP:

propuesta original de Beck.

Programador

El programador escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.

Cliente

El cliente escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio. El cliente es sólo uno dentro del proyecto, pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema.

Encargado de pruebas (*Tester*)

El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

Encargado de seguimiento (*Tracker*)

El encargado de seguimiento proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones. También realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes.

Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.

Entrenador (*Coach*)

Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

Consultor

Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico.

Gestor (*Big boss*)

Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

Proceso XP:

Un proyecto XP tiene éxito cuando el cliente selecciona el valor de negocio a implementar basado en la habilidad del equipo para medir la funcionalidad que puede entregar a través del tiempo. El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

El ciclo de vida ideal de XP consiste en seis fases: Exploración, planificación de la entrega (release), Iteraciones, Producción, mantenimiento y muerte del proyecto.

Fase I: Exploración

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

Fase II: Planificación de la Entrega

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

Fase III: Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores. Wake en [18] proporciona algunas guías útiles para realizar la planificación de la entrega y de cada iteración.

Fase IV: Producción

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente.

Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).

Fase V: Mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

Fase VI: Muerte del Proyecto

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

Metodología Scrum

Scrum al ser una metodología de desarrollo ágil tiene como base la idea de creación de ciclos breves para el desarrollo, que comúnmente se llaman iteraciones y que en Scrum se llamarán “Sprints”.

Scrum gestiona las iteraciones en el ciclo de desarrollo ágil (concepto, especulación, exploración, revisión, cierre)a través de reuniones diarias,uno de los elementos fundamentales de esta metodología.

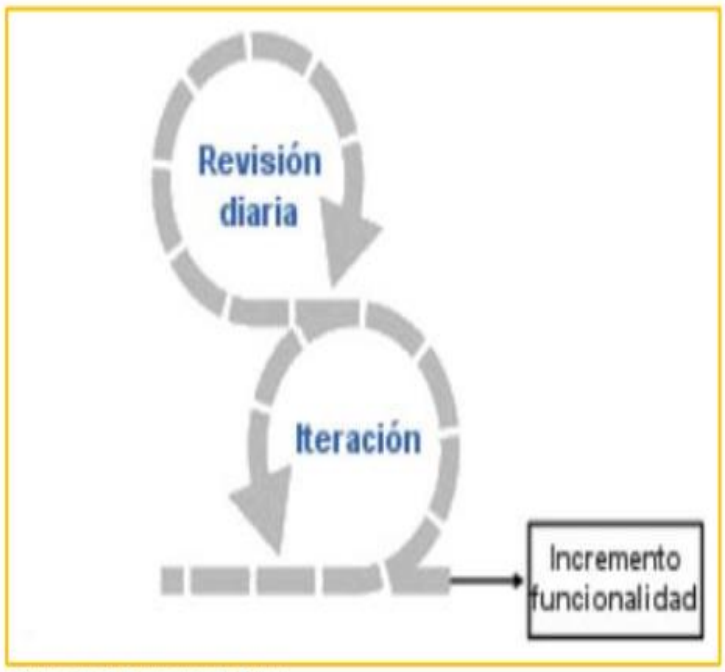


Figura.- 21: Ciclo principal de Scrum

Scrum se puede dividir de forma general en 3 fases, que podemos entender como reuniones. Las reuniones forman parte de los artefactos de esta metodología junto con los roles y los elementos que lo forman.

Reuniones

- **Planificación del Backlog:** se definirá un documento en el que se reflejarán los requisitos del sistema por prioridades.También se definirá la planificación del Sprint 0,en la que se decidirá cuales van a ser los objetivos y el trabajo que hay que realizar.Se obtendrá además un Sprint Backlog,que es la lista de tareas y que es el objetivo más importante del sprint.
- **Seguimiento del Sprint:**Se hacen reuniones diarias en las que se evalua el avance de las tareas a travésde 3 preguntas principales¿Qué trabajo se realizó desde la reunión anterior?¿que trabajo se hará hasta una nueva

reunión?inconvenientes que han surgido y qué hay que solucionar para poder continuar.
- **Revisión del Sprint:** Cuando se finaliza el Sprint se realizará una revisión del incremento que se ha generado.Se presentarán los resultados finales y una demo o versión,esto ayudará a mejorar el feedback con el cliente.

Roles

Los roles sedividen en 2 grupos:cerdos y gallinas.

- **Cerdos:**Personas que están comprometidas con el proyecto y el proceso de scrum.
    - **Product owner:**toma decisiones.
    - **ScrumMaster:**comprueba que el modelo y la metodología funciona.
    - **Equipo de Desarrollo**:equipo pequeño y tienen autoridad para organizar y tomar decisiones para conseguir su objetivo.

- **Gallinas.**
    - **Usuarios:**destinatario final del producto.
    - **Stakeholders:**personas a las que el proyecto les producirá un beneficio.
    - **Managers:**Toma las decisiones finales.

Elementos de Scrum.

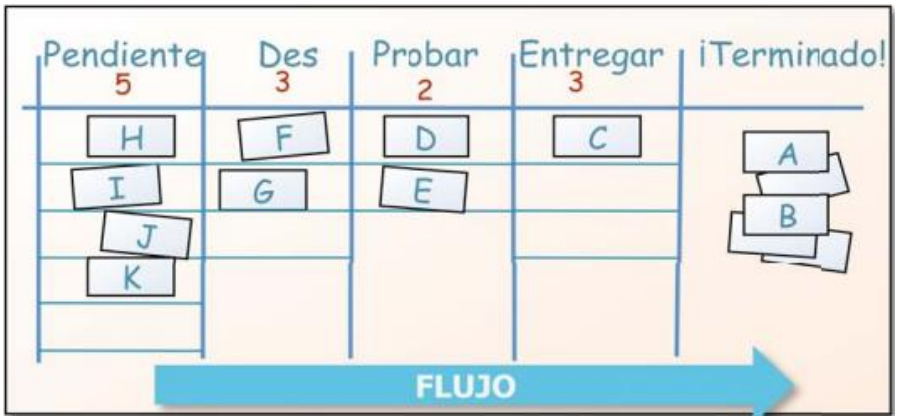
- Product backlog: lista de necesidades del cliente.
- Sprint backlog: lista de tareas que se realizan en un sprint.
- Incremento: parte añadida o desarrollada en un sprint,es una parte terminada y totalmente operativa.

KANBAN

Taichii Ohno fue un teórico de la organización industrial el cual, al trabajar para Toyota, escribió un libro en el año 1988 llamado ***Toyota Production System: Beyond Large-Scale Production*** donde ideo un sistema llamado Kanban donde el cual “Kan” significa visible o visual, y “ban” significaba una tarjeta o tablón***.***

Tomando como ejemplo un equipo de desarrollo, el sistema Kanban se organiza con un gran tablón dividido en columnas, las cuales pueden diferir depende de la decisión del equipo, pero estas pueden y normalmente son:

- Objetivos: se marcan objetivos a corto, mediano y largo plazo los cuales tienen como propósito que los miembros del equipo los mantengan en la mente, esta columna no siempre está presente.
- Pendiente: las tareas pendientes son las que se engloban aquí, son problemas que se afrontan de forma inmediata. El lugar mas alto de la columna debe de colocarse la tarea con la máxima prioridad, y cuando esta se comience a tratar, cambiara de columna.
- Preparación: Aquí se incluyen las tareas que necesitan algun tipo de discusión con el equipo antes de ser tratadas, una vez aclarado, se pasa a la siguiente columna.
- Desarrollo: Aquí se sitúan las tareas que se están tratando hasta que se terminen. Si durante este proceso algo falla o cambia, se necesita regresar a la columna anterior.
- Prueba: Se comprueba que todo funcione correctamente con distintos casos. Si las pruebas no salen bien, se regresa a la columna anterior. En caso contrario, se avanza a la siguiente.
- Aplicación: Aquí se sitúan las tareas las cuales dependen de actualizaciones o parches, por ejemplo, actualizar con una nueva versión una aplicación de un servidor.
- Terminado: Una vez que ya no haya porque preocuparse de una tarea específica, significa que está ya está terminada.



Kanban es un sistema que se debe de seguir sin poder modificar una vez definido, por ejemplo, al momento de escoger una tarea, debe de tratarse las de mayor prioridad primero antes que todas las demás y no escoger otra simplemente porque es mas fácil o mas corta para resolver. Además, puede aplicarse una regla donde se establezca un número máximo de tareas para cada columna, en función de la cantidad de personas en el equipo. Por ejemplo, si hay 10 programadores en el equipo, seria muy absurdo tener 15 tareas debido que algunas tendrían que esperar un buen tiempo en “pendientes”.

Ventajas y desventajas de la metodología Kanban

Ventajas	***Desventajas***
• Flexibilidad.	• No es útil para proyectos grandes y tiempos muy definidos.
• Fácil visualización y control del flujo de trabajo.	• No se controlan los tiempos dedicados a cada tarea.
• Fácil utilización.	• No existe feedback de las tareas realizadas.

¿Mejor que Scrum?

Scrum pone mas acento en la rapidez de cada proceso y en el control por parte del administrador del equipo. Este control exige tiempo en reuniones, planeación, verificaciones internas y discusión. Con Kanban, la labor de administrar al equipo es básicamente determinar las tareas que se deben de hacer y la prioridad que estas tienen en función de los acontecimientos. Todo el flujo del trabajo esta a la vista de todos y si hay imperfectos se tienen en claro donde se produjeron, así que por esa parte es mejor Kanban.

Scrumban

Scrumban es una combinación de las metodologías Kanban y Scrum. Se combinan las mejores características de ambos métodos. Reúne la naturaleza preceptiva de Scrum y la capacidad de mejorar procesos de Kanban.

Scrumban evolucionó a partir de una instancia de Scrum complementado con prácticas básicas Kanban. Estas son: visualización, límites del trabajo en progreso, gestión del flujo de trabajo, y manteniendo políticas explícitas.

- Los principios básicos de implementación de Scrumban incluyen:
    - Empieza con los tableros y labores que usas ahora.
    - Está de acuerdo con perseguir la mejora hacia un proceso más efectivo.
    - Respeta las labores y responsabilidades actuales mientras pretende mejorarlas fácilmente.

Bibliografía

Scrum Manager Gestión de Proyectos.Juan Palacio,Claudia Ruata.

https://s3.amazonaws.com/academia.edu.documents/39164786/mtrigasTFC0612memoria_1.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1552510195&Signature=gaMa5W5jIPWqRWK1gtDwrcAK3FQ%3D&response-content-disposition=inline%3B%20filename%3DMtrigas_TFC0612memoria_1.pdf

http://www.proyectosagiles.org

https://s3.amazonaws.com/academia.edu.documents/38825031/KanbanVsScrum_Castellano_FINAL-printed.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1552519142&Signature=4B16m6GFmu2pvzzo6mX3uGXPQkY%3D&response-content-disposition=inline%3B%20filename%3DKanban_Vs_Scrum_Castellano_FINAL-printed.pdf

https://www.yunbitsoftware.com/blog/2016/08/26/metodologias-agiles-desarrollo-software-kanban/

https://kanbantool.com/es/scrumban-scrum-y-kanban