Prácticas Ágiles - Desarrollo de software con un enfoque ágil

Por Rohit Sinha, PMP

La palabra ágil ya no es más un murmullo. Los retornos inmediatos son lo que la han hecho popular. La filosofía básica del enfoque ágil busca acomodar los cambios. Sabemos que las ideas comienzan a verter cuando de hecho vemos un producto funcionando. Como dice mucha gente, una de las principales dificultades del modelo en cascada tradicional (Figura I) ha sido que gestionar los cambios (por ejemplo, pensar todo de antemano) es un desafío.

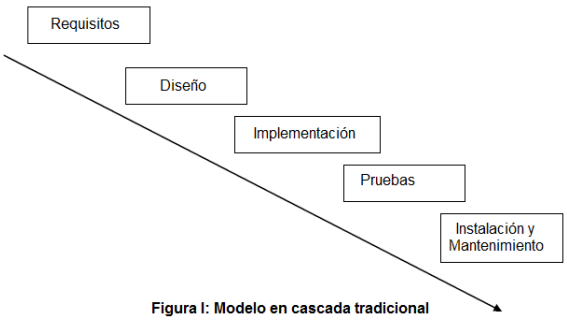

Figura I: Modelo en cascada tradicional

Este artículo habla brevemente sobre la metodología ágil y sus características, y da detalles para gestionar proyectos de software con prácticas ágiles.

Historia

En el modelo tradicional de cascada, el proyecto fluye de arriba hacia abajo, justo como una cascada, por ejemplo el equipo del proyecto debe seguir una secuencia. Los pasos que se siguen uno detrás del otro son: los requisitos, el diseño, la implementación, las pruebas, la instalación, y el mantenimiento (Figura I).

El equipo comienza con los requerimientos, luego va a la fase de diseño y crea un diseño técnico detallado, y así sucesivamente. Este modelo tiene muchas restricciones y una de ellas es no poder mantenerse al ritmo de los requisitos cambiantes. El congelar los requisitos antes de comenzar la implementación ha sido un desafío continuo. Una documentación completa es uno de los aspectos claves de este modelo.

Ha habido una gran cantidad de sesiones de tormenta de ideas sobre varios enfoques prácticos alternativos para las necesidades actuales. El Manifiesto Ágil (bosquejado en febrero del 2001) dice: "E*stamos descubriendo mejores formas de desarrollar software al hacerlo [Ágil] y al ayudar a otros a que lo hagan. A través de este trabajo hemos llegado a valorar:*"

- *a los individuos y a las interacciones sobre los procesos y las herramientas,*
- *al software funcionando sobre la documentación completa,*
- *a la colaboración con el cliente sobre la negociación del contrato,*
- *el responder al cambio sobre el seguir un plan"*

Metodología

Ágil es un enfoque adaptativo que se basa en la filosofía de que los cambios no se pueden evitar. Promueve un ciclo corto de entrega, un análisis justo a tiempo, una colaboración cercana, y una alta visibilidad. Los proyectos se dividen en

pequeños períodos de tiempo conocidos como iteraciones. Una iteración se divide en *scrums* y *sprints*. Una iteración en general abarca de dos a cuatro semanas y generalmente se completa con la una entrega. En general, la primera iteración se usa para realizar el alcance y plan preliminar, y un diseño inicial. En las siguientes iteraciones se hace el desarrollo. Luego de completar una iteración de desarrollo, se hace una demo y se obtiene retroalimentación del cliente. Cualquier cambio necesario en el software que está funcionando se implementará en las iteraciones siguientes. Uno de los beneficios de este modelo es que las modificaciones necesarias se incorporan en el software sin sorpresas de último minuto.

> "No se necesita cambiar. No es obligatorio sobrevivir
> – W. Edwards Deming"

La Figura 2 presenta una visión a alto nivel del proceso general de ágil con sus componentes principales.

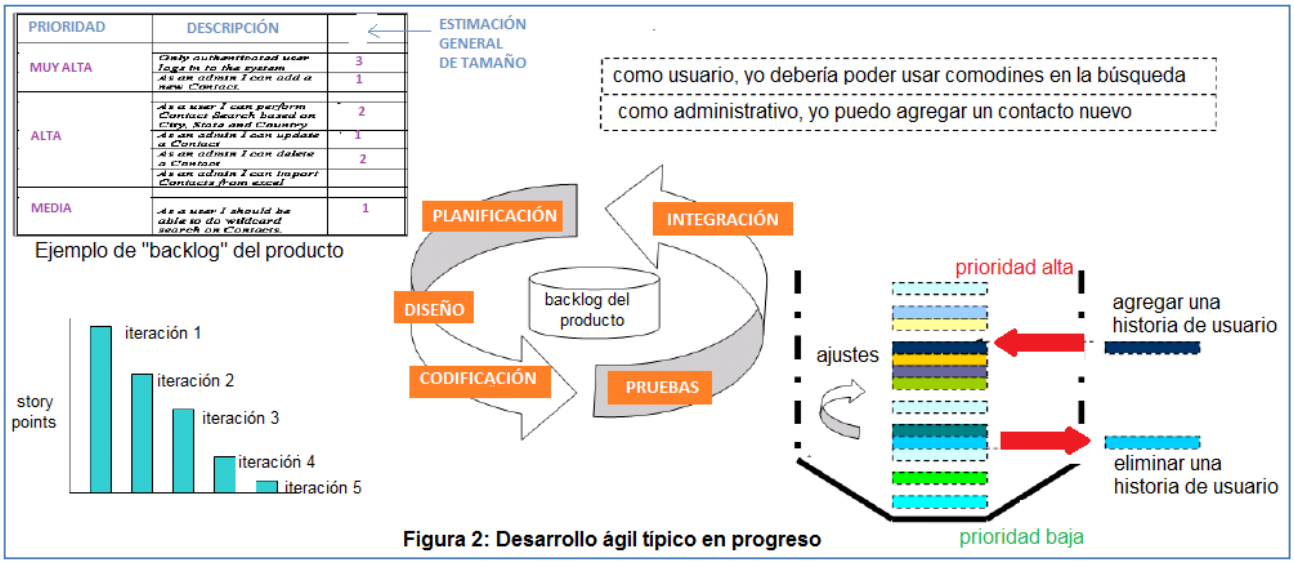

Figura 2: Desarrollo ágil típico en progreso

Los detalles de este proceso mostrado en la Figura 2 se discuten en las siguientes secciones. Este enfoque tiene las siguientes características:

Scrum

Scrum es un marco del proceso de desarrollo de software que contiene prácticas y roles predefinidos que permiten la creación de equipos que se organizan a sí mismos. Reconoce los cambios y se enfoca en tratar con los requisitos que surgen. Los roles principales de los equipos de Scrum son el scrum master, el dueño del producto, y el equipo:

- *Scrum master* es el rol del líder del proyecto. Es la persona que coordina y mantiene los procesos. Es quién facilita los procesos de Scrum y coordina con el dueño del producto y con el equipo de desarrollo.
- *Dueño del producto* es un interesado clave que representa al usuario final y sirve como un vínculo entre el equipo del proyecto y el cliente. Es quién prioriza los requisites. Responde a las preguntas del equipo y le da dirección al equipo. En mi opinión, algunas características valiosas del dueño del producto son tener buenas habilidades de comunicación, estar dispuestos a profundizar en su entendimiento del producto y del valor para el mercado, buenas habilidades con la interfaz de usuario, y algún antecedente técnico (le ayuda para comunicarse con el equipo de desarrollo).

- *El equipo* es un equipo de varios departamentos, de 5 a 9 miembros que hacen el análisis, el diseño, la implementación y las pruebas.

Sprint

Un sprint tiene una duración de dos a cuatro semanas, y contiene un producto en progreso con una lista de casos o historias de usuario tomadas del log del producto (o feature backlog en inglés). Cada día, en un sprint, ocurre una reunión diaria que se hace de pie con el equipo, conocida también como el Scrum diario. Esto ayuda a dar a conocer el estado actual, y a resolver cualquier cosa que bloquee el proyecto. Durante la reunión, cada miembro del equipo responde a tres preguntas:

- ¿Qué has hecho desde ayer?
- ¿Qué planificas hacer hoy?
- ¿Tienes algún problema que te detenga el cumplir con tu meta?

Scrum de Scrums

El término *scrum de scrums* aplica normalmente a un proyecto que consiste de varios equipos, dado que ayuda a resolver situaciones y dependencias entre los distintos equipos. Involucra representantes de cada equipo, y normalmente el scrum master representa a su equipo durante la reunión. En general esto ocurre diariamente luego del Scrum diario.

Historias de usuario (User story)

En el mundo ágil, los requisitos se expresan como una historia de usuario. Es una forma rápida de representar los requisitos sin entrar en detalles. Se escribe en una o dos oraciones y en un lenguaje de negocios. Cada funcionalidad de la aplicación se representa mediante una historia de usuario. Hay ejemplos a continuación:

como usuario, yo debería poder usar comodines en la búsqueda

como administrativo, yo puedo agregar un contacto nuevo

En mi experiencia, esta es una gran manera de mantener el registro del producto (o *feature backlog* en inglés). Se presta atención a los detalles cuando se acerca la entrega de la historia. He encontrado que una ilustración a alto nivel de los requisitos le da flexibilidad al equipo de desarrollo para que piense más, y a menudo te sorprenden con su innovación.

Puntos de historia (Story Points)

Se debe estimar las historias de usuario, y la estimación se hace en puntos de historia. Un punto de historia es un número que dice qué tan fácil o difícil es implementar una historia. Representa el tamaño de la historia de usuario en relación a otras historias. Por ejemplo, se le da dos puntos a una historia si Ud. cree que es el doble más grande que una historia de un punto, y la mitad de una historia de cuatro puntos. Este método de estimación es más práctico, y por lo tanto, la estimación total del esfuerzo es apropiada.

Lista de requisitos del producto (Feature backlog)

El *backlog* del producto es una lista priorizada de historias de usuario. Es e l corazón de Scrum. El dueño del producto generalmente colabora con el cliente y vienen con ítems del backlog. Sus características son:

- Es dinámica. Cambia continuamente cuando cambian los requisitos.
- Cada ítem tiene una prioridad (Tabla I)
- En general no tiene el detalle del requisito. Los detalles se ven cuando se va a implementar el requisito.
- Los ítems de más arriba en la lista en general son más detallados comparados con los ítems de más abajo, para los cuales los requisitos en general son vagos.

PRIORIDA	DESCRIPCION	TAMAÑO
MUY ALTA	Solo ingresan al sistema los usuarios autenticados.	3
	Los administradores pueden agregar contactos nuevos	3
ALTA	Como usuario puedo buscar contactos según ciudad, esta	2
	Los administradores pueden actualizar un contacto	4
	Los administradores pueden borrar un contacto	1
	Los administradores pueden importar contactos de Excel	3
MEDIA	Los usuarios pueden usar comodines en la búsqueda	3

Tabla 1 - Ejemplo de backlog del producto

Dirigiendo proyectos ágiles

Como mencioné en el Manifiesto de Ágil, los proyectos ágiles recomiendan la comunicación cara a cara sobre la comunicación vía documentos, especialmente si el equipo está ubicado en el mismo lugar. En el caso de los equipos virtuales (dispersos geográficamente), se puede sustituir las comunicaciones personales mediante las conferencias por video, skype, wiki, y correo electrónico. En mi experiencia, la comunicación de este modo también funciona bien. En los proyectos ágiles los interesados están involucrados desde el inicio del proceso; he encontrado que esto es muy útil ya que permite que los incidentes se descubran temprano y ayuda a construir un producto amigable para el usuario. Los ejemplos se usan para describir las funcionalidades de una aplicación o una unidad de código. El software funcionando es la medida de progreso. Las prácticas de software como el desarrollo orientado a las pruebas, la integración continua, la restructuración del código fuente, y las pequeñas versiones, ayudan a mejorar la eficiencia general del proyecto.

Duración de la iteración

Una iteración ágil típica dura dos semanas, debido a que una semana es demasiado corto y un mes puede ser muy largo. Dos semanas es suficiente para obtener algo significativo funcionando dado que hay alguna sobrecarga de la planificación y el cierre de la iteración. Sin embargo, a veces, los equipos más grandes (de 8 a 9 personas), tienen iteraciones más largas para mostrar progreso. En mi opinión, como regla general se usan dos semanas para una iteración, pero en el caso de un proyecto o de un equipo más grande, se puede considerar un ciclo de cuatro semanas.

Planificación de la iteración

El principal objetivo de la planificación es definir el alcance de la iteración. Sobre la base del proyecto actual o de la situación del negocio, se agregan o se eliminan funcionalidades del backlog. La planificación de iteraciones no debería llevar más de dos a cuatro horas. Esto se hace justo antes de comenzar con la iteración. Se seleccionan las funcionalidades de mayor prioridad; puede haber también algunos pocos ítems de prioridad más baja si encajan con la meta general de la iteración. Estas funcionalidades se parten en tareas. Cada tara en general lleva de cuatro horas a dos días. Si una tarea necesita más de dos días, se parte en subtareas. Durante el plan de la iteración, si se descubre que las funcionalidades se completarán antes de tiempo, el dueño del producto ajusta su alcance agregándole más funcionalidades. Del mismo modo, si el tiempo de la iteración no es suficiente para todos los requisitos de la iteración, el dueño del producto ajusta la iteración eliminando algunos requisitos y asignándolos a otras iteraciones.

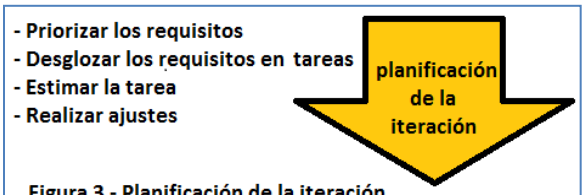

Figura 3 - Planificación de la iteración

El desarrollo

En la mayoría de las implementaciones ágiles se comunican cara a cara diariamente entre los miembros del equipo. Esto también incluye al dueño del producto como representante del cliente y a algunos interesados como observadores. Aquí el software funcionando es la medida principal de avance. Cuando un equipo trabaja en diferentes lugares, mantiene contacto diario a través de conferencias por video, voz, correo electrónico, etc. La generación automatizada del build y las pruebas automatizadas en general son parte del desarrollo y ayudan a maximizar la eficiencia del equipo. He encontrado que vale la pena pasar un tiempo inicial para poner esos procesos a funcionar. Hace más fácil la vida de todos y el desarrollo general más eficiente. Los proyectos se enfocan en la calidad, la cual incluye la calidad del código. La restructuración del código (code refactoring) y las pruebas son atributos de la calidad. La práctica de restructuración del código es donde Ud. hace cambios simples al código que

mejoran la calidad sin cambiar la semántica. El equipo prefiere probar a menudo y temprano, y los más disciplinados aún toman un enfoque orientado a las pruebas donde escriben una prueba y el código suficiente para satisfacer esa prueba, y luego vuelven a iterar. La Figura 4 muestra un ciclo de iteración típico:

Figura 4 - El ciclo del desarrollo ágil gira alrededor del backlog del producto

Seguimiento y ajustes

Se registra el avance del proyecto en la iteración y se hacen ajustes (Figura 5).

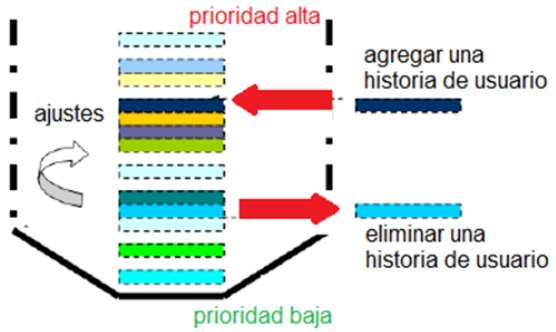

Figura 5 - El backlog del producto muestra la gestión de cambios

A medida que el proyecto avanza, los interesados logran un mejor entendimiento al ver el software funcionando, y sus necesidades pueden cambiar. Además, también son factores comunes los cambios en la situación del negocio o los cambios de prioridades a nivel de la organización. El backlog del producto se actualiza basado en las necesidades actuales, y se vuelven a planificar las siguientes iteraciones.

Una de las herramientas que ayuda a registrar el progreso es el cuadro llamado *burn down* que es una representación gráfica del trabajo en representación del tiempo. En general, el eje vertical representa la cantidad de trabajo restante y el eje horizontal representa el tiempo. Es una de las formas preferidas de mostrar el progreso de un sprint. La Figura 6, muestra un cuadro *burn down* de una versión a liberar, donde el trabajo se visualiza como puntos de historia y tiempo en las iteraciones:

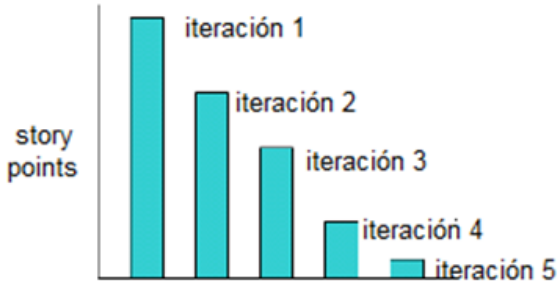

Figura 6 - Ejemplo de cuadro *burn down*

Asuntos

Algunas personas aún tienen dudas sobre este enfoque. Algunas preguntas o preocupaciones comunes son: "¿necesito adoptarlo?" "No parece ser un proceso organizado o fácil de vendérselo a mis clientes," "¿Qué pasará con el aseguramiento de la calidad?" y "Suena demasiado riesgoso." Todos estos son puntos válidos para considerar antes de adoptar este enfoque. Además, no siempre es ágil es bueno necesariamente para todos los proyectos. Hay otros enfoques de desarrollo que vale la pena considerar si se da lo siguiente en un proyecto particular:

1. Es un equipo grande
2. No es posible un ciclo de iteración pequeño
3. Los requisitos están claramente enunciados y detallados
4. Los requisitos no cambian.
5. Los interesados no están disponibles durante el desarrollo.
6. Se necesita la documentación completa para tener éxito.

En situaciones del mundo real, los puntos mencionados no aplican la mayoría de las veces, y por lo tanto, ágil es una buena solución.

Conclusión

Si bien aún hay preocupaciones sobre el enfoque ágil, sus beneficios—tales como el acomodar los requisitos cambiantes del cliente, retornos inmediatos, coordinación cercana, y flexibilidad—facilitan que aquellos que están desarrollo de software lo acepten rápidamente. La filosofía de las "personas son más importantes que los procesos" se está haciendo más exitosa con cada día que pasa. Es un enfoque muy bien pensado para proyectos que se ejecutan con éxito. En un mundo como este que funciona tan rápidamente, muchas compañías están adoptando las prácticas ágiles para ser más exitosos.

Sobre la autora

Rohit Sinha, PMP, es gerente de productos con 11 años de experiencia en proyectos de Tecnología de Información. Es un profesional innovador con gran experiencia en liderazgo y una habilidad comprobada para identificar, analizar, y resolver problemas para aumentar la satisfacción del cliente y controlar los costos. Es responsable por la ejecución y dirección de varios proyectos, y se enfoca en construir la confianza y las relaciones necesarias para que los proyectos avancen. Tiene expertise en el dominio de seguros, contabilidad, y gestión documenta. Tiene una maestría en ciencias de la computación. Puede contactarse a rohit_s25@hotmail.com

Artículo traducido del original en inglés titulado "Software Development with Agile Approach" en la Biblioteca Virtual del PMI (PMI Virtual Library) de www.PMI.org.

Si tiene alguna sugerencia de mejora de esta traducción al español puede enviarla a LASpanishNews@pmi.org