novática
Revista de la Asociación de Técnicos de Informática

Novática, revista fundada en 1975 y decana de la prensa informática española, es el órgano oficial de expresión y formación continua de **ATI** (Asociación de Técnicos de Informática), organización que edita también la revista **REICIS** (Revista Española de Innovación, Calidad e Ingeniería del Software).

<http://www.ati.es/novatica/>
<http://www.ati.es/reicis/>

ATI es miembro fundador de **CEPIS** *(Council of European Professional Informatics Societies)* y es representante de España en **IFIP** *(International Federation for Information Processing);* tiene un acuerdo de colaboración con **ACM** *(Association for Computing Machinery),* así como acuerdos de vinculación o colaboración con **AdaSpain**, **AI2**, **ASTIC**, **RITSI** e **Hispalinux**, junto a la que participa en **ProInnova**.

Coordinación Editorial, Redacción Central y Redacción ATI Madrid
Padilla 66, 3º, dcha., 28006 Madrid
Tlfn.914029391; fax.913093685 <novatica@ati.es>
Composición, Edición y Redacción ATI Valencia
Av. del Reino de Valencia 23, 46005 Valencia
Tlfn. 963740173 <novatica_prod@ati.es>
Administración y Redacción ATI Cataluña
Via Laietana 46, ppal. 1ª, 08003 Barcelona
Tlfn.934125235; fax 934127713 <secregen@ati.es>
Redacción ATI Aragón
Lagasca 9, 3-B, 50006 Zaragoza.
Tlfn./fax 976235181 <secreara@ati.es>
Redacción ATI Andalucía <secreand@ati.es>
Redacción ATI Galicia<secregal@ati.es>
Suscripción y Ventas <http://www.ati.es/novatica/interes.html>, ATI Cataluña, ATI Madrid
Publicidad Padilla 66, 3º, dcha., 28006 Madrid.
Tlnf.914029391; fax.913093685 <novatica@ati.es>
Imprenta: Derra S.A., Juan de Austria 66, 08005 Barcelona.
Depósito legal: B 15.154-1975 -- ISSN: 0211-2124; CODEN NOVAEC
Portada: La casa danzante - Concha Arias Pérez / © ATI
Diseño: Fernando Agresta / © ATI 2003

sumario

Nº 217, mayo-junio 2012, año XXXVIII

Tema del próximo número: "Sistemas multiagente"
(Número distribuido solamente en versión digital)

Métodos ágiles hoy

Alonso Alvarez García[1,2], Carmen Lasa Gómez[1], Rafael de la Heras del Dedo[1]
[1]Telefónica Digital, Product Development and Innovation. Autores del libro "Métodos ágiles y Scrum"; [2]Coordinador de la sección técnica "Tendencias tecnológicas" de Novática

<{aag,lasa,rafaelh}@tid.es>

Resumen: *Los métodos ágiles son una forma de trabajo en auge que trata de dar una nueva respuesta a los problemas históricos del desarrollo de software. A diferencia de otros métodos tradicionales, los ágiles se adaptan al cambio y a la incertidumbre, reconociendo la realidad de los proyectos y no luchando contra ella. Antes de considerar la adopción de uno u otro método, es conveniente conocer su filosofía, principios y prácticas. Este artículo hace una revisión rápida de los métodos más habituales hoy en día para ayudar a elegir el más conveniente a la realidad de la organización que se plantee su adopción.*

Palabras clave: *Agile, desarrollo software, Extreme Programming, JIT, Kanban, Lean, metodología, métodos ágiles, métodos de desarrollo, Scrum, XP.*

1. Introducción

En sus más de sesenta años de historia la industria del software ha alcanzado la madurez en muchos aspectos. Sin embargo, algunos de los problemas que arrastra desde sus orígenes persisten: los proyectos siguen teniendo retrasos, problemas de calidad y costes, y continúan siendo esencialmente impredecibles. Todas las técnicas que hemos ideado en estos años no han acabado de proporcionar una solución satisfactoria a estos problemas.

En otros campos de la industria de fabricación (y notablemente en la de vehículos) se han desarrollado métodos para mejorar la productividad entendida no sólo como producir más en menos tiempo o por unidad de recursos, sino también con más calidad.

Algunos de estos métodos, especialmente la fabricación JIT (*Just in Time*) y LEAN han resultado tener aplicación en el mundo del software bajo el nombre de "métodos ágiles" [1].

¿Qué hace que un método sea ágil?, es decir, ¿qué es lo que tienen en común estos métodos?

Todos ellos consideran la **colaboración** como un elemento clave. Tanto las personas que están construyendo el producto como el cliente deben trabajar en constante comunicación y sentirse miembros de un gran equipo con objetivos comunes. Con este enfoque, la **comunicación** constante y a todos los niveles es crucial para crear el producto con una **calidad** excelente y que cumpla exactamente las necesidades del cliente evitando sorpresas a todos los implicados.

Por otro lado, un método es ágil si permite construir un producto de forma **incremental**, es decir crear algo muy sencillo inicialmente y que vaya siendo enriquecido y completado de forma progresiva. No se construirán trozos de producto por separado que luego tendremos que hacer encajar al final como un rompecabezas, sino que se construye contemplando la totalidad desde el principio.

Otro factor común de estos métodos ágiles es su **sencillez**. Sus reglas son sencillas y de sentido común pero eso sí, es necesaria la experiencia y profesionalidad para obtener el máximo beneficio de estas reglas. Para que un método pueda considerarse ágil, debe ser **adaptativo**, es decir, se debe construir contemplando siempre la posibilidad de introducir modificaciones y cambios en cualquier etapa de la construcción.

No debe extrañar encontrarse al repasar los distintos métodos con conceptos, principios y prácticas comunes que se repiten una y otra vez. Lo cierto es que muchos de estos principios son aplicados intuitivamente por muchas personas y organizaciones. Esto no es más que una forma de constatar que los métodos ágiles son en realidad una forma de enunciar el sentido común en la construcción de productos.

El Manifiesto Ágil [2] recoge las ideas en las que se fundamentan estos métodos. Posiblemente una de las más relevantes sea dejar atrás la orientación inflexible y determinista de los proyectos convencionales. Ahora, la incertidumbre deja de ser una amenaza, y pasa a ser una corriente que se acepta y de la que se trata de sacar el máximo partido.

Bajo estas premisas, la agilidad como filosofía de trabajo ha ido abriéndose camino en equipos, departamentos y empresas hasta ser hoy en día una de las formas de organizar proyectos más en boga, incluso de *moda*.

Y esto no es casual. Los métodos tradicionales se han basado en dos premisas: la disponibilidad de unos requisitos completos, detallados y estables antes de iniciar el trabajo; y en que los proyectos tienen un alcance y duración definidos, un final. Sin embargo lo habitual (antes y ahora) es tener requisitos mutantes que se definen casi a la vez que se construye el software; y los proyectos tienden a alargarse indefinidamente en versiones y revisiones sucesivas.

En este entorno sólo los métodos ágiles proporcionan una forma de trabajar sacando partido de la incertidumbre hasta convertirla en una aliada, y de proporcionar un marco colaborativo y productivo para que los equipos de trabajo desarrollen todo su potencial.

La adopción de los distintos métodos ágiles se está acelerando y generalizando, por lo que es posible que muchas organizaciones ya los hayan aplicado o estén sopesando usarlos. Los métodos ágiles son mucho menos prescriptivos que sus equivalentes convencionales, y esto hace que no se apliquen de forma rígida sino adaptándolos al modo de trabajo de cada empresa. De hecho, los métodos ágiles ofrecen reglas generativas, es decir, favorecen el que se creen reglas nuevas en el caso de que fuera necesario.

Por todo ello, conocer (siquiera esquemáticamente) cómo operan y en qué se basan servirá de ayuda a la hora de elegir la modalidad y combinación más apropiada para las necesidades de cada organización. Éste es el ánimo con el que hemos escrito este artículo.

2. Scrum

Scrum propone un marco de trabajo basado en iteraciones y que se apoya en equipos autogestionados. Se trata de una de las metodologías ágiles más difundidas en la actualidad, y eso hace que sus conceptos, herramientas y prácticas aparezcan en muchos otros métodos.

De forma muy resumida [3], Scrum:

- Promueve la organización en torno a pequeños equipos autoorganizados e interdisciplinares.
- Divide el trabajo en pequeños entregables, priorizados y estimados en esfuerzo.
- Distribuye el tiempo de trabajo en iteraciones fijas que generan un producto enseñable a su fin.

Figura 1. Métodos Ágiles.

■ Optimiza el plan de entregas y prioridades de acuerdo con el cliente, y aprendiendo de las iteraciones.
■ Y optimiza el proceso por medio de revisiones (retrospectivas) al final de cada iteración.

Una de las principales características de Scrum es que en cada iteración todas las etapas de la creación de un producto se solapan. En cada *Sprint* [1] se realiza la planificación, análisis, creación y comprobación de lo que se va a entregar al final del mismo.

El marco de trabajo general de Scrum está compuesto por una serie de roles, reuniones y de paneles de información o artefactos. Estas entidades se utilizan también en la aplicación práctica que muchas organizaciones hacen de otros métodos ágiles.

Los roles en el equipo Scrum representan una responsabilidad en el proceso y no la posición dentro de la organización**:**
■ El *Product Owner* o dueño del producto. Es el responsable desde el punto de vista del negocio.
■ El *Scrum Master*. Responsable de que el equipo sea productivo, ayudándole en todo momento a conseguir el objetivo acordado y de asegurar que los principios de Scrum se están respetando.
■ El equipo. Es el responsable de la construcción del producto.

Los artefactos de Scrum:
■ El *Product Backlog*. Es el lugar que contiene los requisitos del cliente priorizados y estimados. El *Product Backlog* está gestionado por el *Product Owner* y usa lenguaje de negocio.
■ El *Sprint Backlog*. Es la selección de requisitos del *Product Backlog* negociados para el *Sprint* y que se ha descompuesto en tareas por el equipo para expresar los requisitos del cliente en un lenguaje técnico.
■ El *Burndown Chart:* En esta gráfica se representa el trabajo pendiente del equipo, tanto dentro el Sprint como en la totalidad del proyecto.

Las reuniones en Scrum: Dado que uno de los principios ágiles más destacados es el llamado *time-boxing*, en Scrum se busca contener estas reuniones en el tiempo para hacerlas más efectivas y productivas.
■ Planificación del Sprint (*Sprint Planning*). En ella se busca obtener un objetivo claro y compartido del trabajo que hay que realizar para la iteración siguiente. El equipo selecciona los items del *Product Backlog* con los que considera que puede comprometerse a realizar durante el *Sprint* y los dividirá de forma colaborativa en tareas.
■ Reunión diaria (*Daily Meeting*). Es el momento de la sincronización del equipo en el que cada miembro comparte con el resto en qué estado se encuentra el trabajo que está realizando con qué piensa continuar, y qué impedimentos hay para continuar.
■ Revisión del *Sprint* (*Sprint Review*). Al finalizar el *Sprint*, el equipo analiza el estado de su trabajo con el *Product Owner* y con cualquier otra persona que pueda aportar información valiosa. Este es el momento de analizar para mejorar el "qué" se está construyendo.
■ Retrospectiva del equipo (*Sprint Retrospective*). Después de la *Review*, el equipo se reunirá para buscar la mejora continua en su trabajo y analizar los aspectos que le impiden ser más productivo. Es este el momento de analizar para mejorar "cómo" se está trabajando.

Scrum es poco prescriptivo, por lo que se pueden añadir los roles, artefactos o reuniones que sean necesarios, aunque siempre basándose en la filosofía general *Agile* de hacer las cosas de la manera más sencilla posible.

Más que una metodología de trabajo, Scrum ayuda a que se planteen las preguntas correctas y a hacer que sean las personas las últimas responsables de encontrar las respuestas acertadas. Scrum ayuda enormemente a que salga a la luz todo aquello que nos impide ser más productivos para poder darle una solución temprana.

3. Kanban

Kanban es una palabra de origen japonés que significa literalmente "tarjetas".

Aplicando este método se consigue mostrar permanentemente y de forma muy visual el estado del proyecto a todos los implicados. Métodos similares al que propone Kanban llevan aplicándose con éxito en cadenas de producción y talleres desde hace más de un siglo. La aplicación de Kanban al desarrollo de software es relativamente reciente (en torno a 2004) y está en auge.

Kanban es un método muy útil para gestionar productos cuyos requisitos cambian constantemente, bien porque aparezcan nuevas necesidades, o bien porque su prioridad varíe. Este método también es útil en los casos en los que las planificaciones o estimaciones de un equipo se alarguen demasiado y dejen de ser productivas, así como cuando no se pueda comprometer un equipo a trabajar con iteraciones de duración fija y predeterminada por el motivo que sea (interrupciones, cambios, dependencias, etc.).

De forma simplificada, los pasos que se deben seguir para trabajar con Kanban son los siguientes:
■ Visualizar el flujo de todo el trabajo. En un panel debe estar representado todo el flujo del trabajo que hay que realizar en el proyecto, desde el principio hasta el último momento. Este panel debe estar accesible y visible para todos los miembros del equipo.
■ El panel cuenta con columnas que permiten mostrar en qué estado del flujo está cada ítem (donde la primera representa el *Backlog* del producto).
■ División del trabajo en ítems pequeños que se describen en una tarjeta. Se priorizan

ordenados en la primera columna del tablero. Una buena práctica es tratar de dividir los ítems de forma que la carga de trabajo sea similar entre ellos, y de esa forma poder estimar visualmente el trabajo asociado a cada estado.

- Limitación del trabajo en curso. Esta es una de las claves para que Kanban funcione. Es imprescindible poner un límite al número de ítems permitidos en cada columna y de esta forma evitar colapsos, cuellos de botella y eliminar cuanto antes los impedimentos que surjan y que impidan trabajar con un ritmo sostenible. Este número (WIP) que indica el límite permitido en cada columna debe estar visible en la parte superior de la misma.
- Medición del tiempo empleado en completar un ciclo. Se calcula el tiempo que se emplea desde que se empieza a trabajar con un ítem o tarea hasta que se da por cerrado o terminado y se busca la manera de hacer disminuir este tiempo. Esta práctica ayuda también a ser predecible y a poder hacer una estimación previa de cuánto tiempo se empleará en completar cada ítem.

Con Kanban los implicados en la creación de un producto tienen acceso a toda la información del mismo y al estado de cada una de sus partes en cada momento. El grado de compromiso aumenta notablemente ya que todos pueden participar en la mejora directa e inmediata del proceso.

Trabajando de esta forma tan visual, se facilita identificar los problemas, detectar cuellos de botella y eliminar impedimentos según se producen, reduciendo costes y aumentando la productividad y la calidad. Por otro lado, todo el equipo participa en la mejora de la totalidad y no se centra solo en su parte.

Las características de Kanban hacen que pueda utilizarse para organizar y gestionar el trabajo en cualquier campo y no exclusivamente para proyectos de desarrollo de software. Por ejemplo, Kanban es especialmente valioso para los equipos de soporte y mantenimiento. También es habitual hacer una combinación de prácticas de Scrum y Kanban que suelen conocerse informalmente como "*Scrumban*".

4. Lean Software Development

El método ágil *Lean Software Development* es un método ágil cuyo origen está en la fabricación industrial, pero que posteriormente fue adaptado al desarrollo de software [4]. Lean exige una gran madurez del equipo de trabajo para evitar que una mala aplicación degenere en una forma de trabajar sin método ni dirección.

Los principios en los que se basa este método son los siguientes:

- Eliminar todo lo que no aporte valor para no perder foco. Esto se traduce en simplificar burocracia y gestión, optimizar los procesos y evitar la falta de información así como las interrupciones al equipo de trabajo.
- Optimizar el todo. La orientación buscada es global y a largo plazo. Se busca evitar que cambios locales a pequeña escala afecten negativamente al conjunto del desarrollo.
- Construir con calidad y desde el primer momento.
- Aprender constantemente.
- Reaccionar rápido. Para ello hay que trabajar con iteraciones cortas de forma que los comentarios del cliente sean frecuentes y se cubran sus expectativas y necesidades en cada momento.
- La mejora continua. El foco de la mejora debe centrarse en las personas y en los procesos que hacen posible construir un producto y no en mejorar exclusiva y directamente el producto en sí. De esta forma se mejorará el producto actual y el sistema estará listo para poder crear otros productos con éxito en el futuro.
- Cuidar al equipo de trabajo. Un equipo de trabajo debe estar motivado y esto se consigue proporcionándole cierto grado de autonomía para poder tomar decisiones con sentido, ofreciendo a cada persona la posibilidad de aprender y mejorar de manera permanente, y por último, procurar que sientan que su trabajo es valioso en todo momento.

5. Feature Driven Development (FDD)

El desarrollo orientado a funcionalidad (FDD) es un método ágil que no cubre todo el proceso de desarrollo de un producto sino que se centra en las fases de diseño y construcción. Sus puntos clave son el trabajo en iteraciones, control continuo, la calidad de lo creado y entregas frecuentes para poder realizar un seguimiento continuo en colaboración con el cliente y poder así incorporar sus necesidades en el producto con frecuencia.

FDD [5] propone seguir unos pasos secuenciales para realizar el diseño y la implementación del sistema. Estos pasos son:

- Crear un modelo global. Un primer paso será tener un conocimiento del alcance, de los

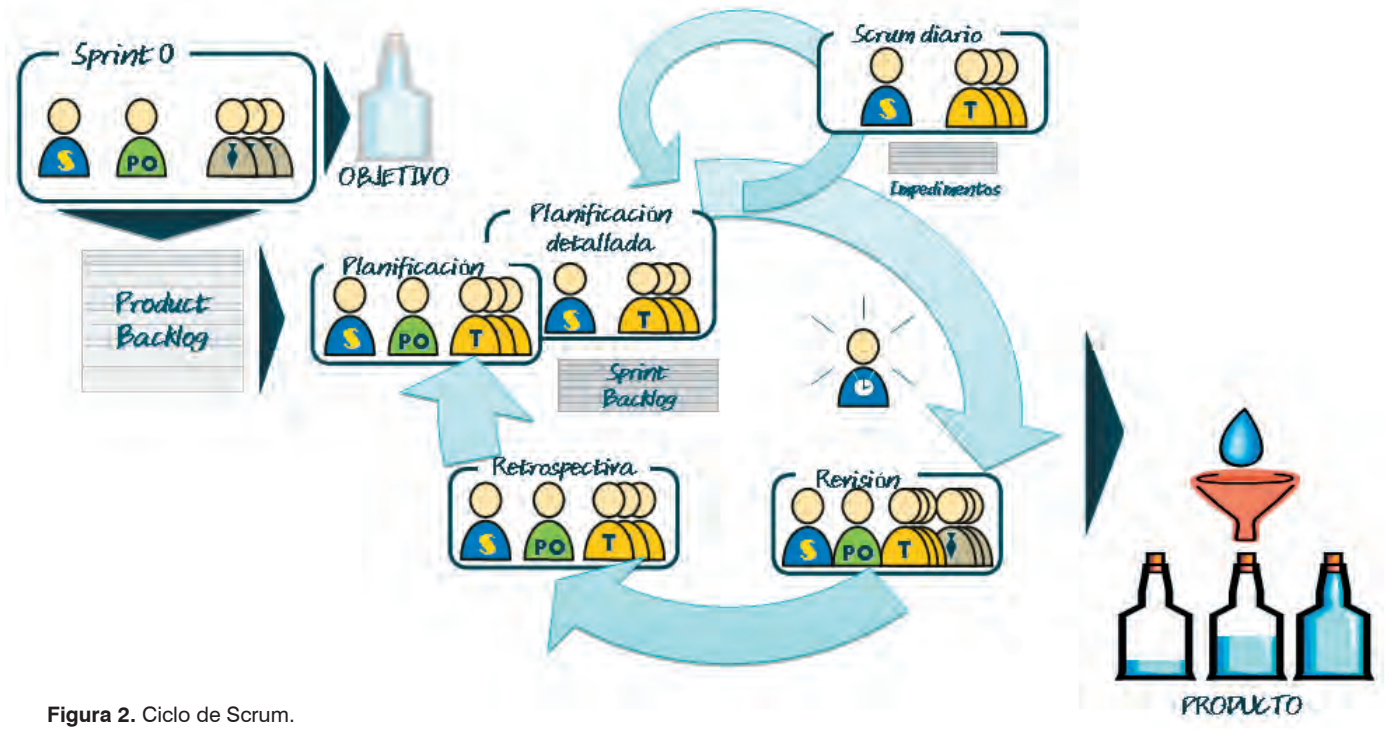

Figura 2. Ciclo de Scrum.

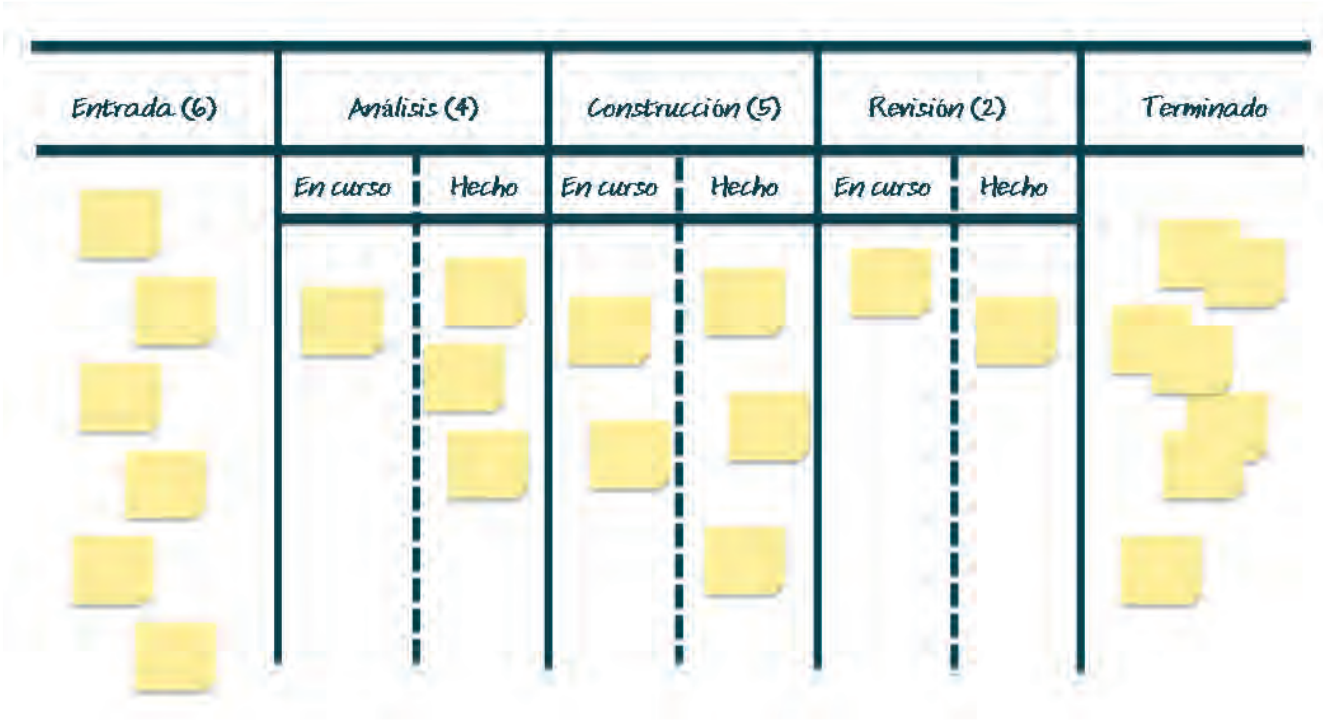

Figura 3. Panel de Kanban.

requisitos y del contexto del sistema en el que se va a construir el producto.

- Crear una lista de funcionalidades. Es necesario plasmar este modelo global, junto con los demás requisitos del conjunto del sistema, en una única lista de necesidades o funcionalidades que hay que cubrir.
- Planificar por funcionalidades. Es decir, a la hora de hacer un plan a alto nivel, debe siempre tenerse en cuenta la prioridad de las funcionalidades y las dependencias entre ellas, lo que afecta también a los hitos de entregas.
- Diseñar y construir por funcionalidad. Lo que se hace de forma iterativa en ciclos que oscilan entre algunos días y dos semanas. Esta etapa de diseño y desarrollo incluye las actividades de pruebas unitarias, revisión de código e integración.

6. Dynamic Systems Development Method (DSDM)

DSDM [6] nace en 1994 en el Reino Unido donde continúa como uno de los marcos de trabajo más extendidos para el desarrollo rápido de aplicaciones.

La forma de trabajar que propone este método para el ciclo de vida de un proyecto, está estructurada en cinco fases, de las cuales las dos primeras se realizan una sola vez y las tres últimas se realizan de forma iterativa e incremental.

Estas etapas son: estudio de la viabilidad del proyecto, estudio del negocio, iteraciones del modelo funcional, iteraciones para la creación del diseño y desarrollo del producto y, finalmente, iteraciones para la implementación.

La filosofía de DSDM [7] es sencilla:

- El desarrollo de un producto debe entenderse como un trabajo en equipo ya que para tener éxito debe combinarse el conocimiento de las necesidades del negocio que tienen los clientes con el perfil técnico de los desarrolladores.
- La calidad debe contemplarse desde dos puntos de vista: la solidez técnica y la facilidad de uso.
- El desarrollo debe hacerse de forma incremental. Es mejor entregar una parte útil a tiempo que entregar todo el producto pero demasiado tarde.
- Debe trabajarse inicialmente en las funcionalidades que aporten mayor valor al negocio y por tanto, los recursos deben invertirse en el desarrollo de las mismas.

Uno de los principales principios en los que se basa DSDM es la creencia de que un requisito no se puede prefijar completamente al inicio del desarrollo, lo que supone trabajar en su prioridad y alcance repetidamente.

En el momento de su aparición en 1994, DSDM propuso una aproximación alternativa y revolucionaria al enfoque de los métodos más tradicionales. Frente a la fijación de requisitos y, en función de ellos, estimar recursos y fecha de entrega, DSDM propone fijar los recursos destinados a un producto y la fecha de entrega y hacer una estimación de la funcionalidad que se entregará. En definitiva, se sabrá cuando se va a entregar algo valioso al cliente pero a priori no se sabrá qué es exactamente lo que se va a entregar.

De manera resumida, algunas de las prácticas básicas de DSDM comunes a otros métodos son: mantener un latido (ritmo constante), requisitos priorizados y clasificados, fomentar el prototipado para refinar requisitos, y poner la calidad por delante en todo el proceso.

7. Programación eXtrema o *eXtreme Programming* (XP)

Tal y como lo define Kent Beck [8], XP es un método ágil para el desarrollo de software que resulta muy útil a la hora abordar proyectos con requisitos vagos o cambiantes. XP se aplica muchas veces en combinación con otros métodos como Scrum.

XP es un método adaptativo que se ajusta muy bien a los cambios. XP facilita desarrollar código de forma que su diseño, arquitectura y codificación permitan incorporar modificaciones y añadir funcionalidad nueva sin demasiado impacto en la calidad del mismo.

Por otro lado, como los demás métodos ágiles, XP está muy orientado hacia las personas, tanto a las que están creando el producto como a los clientes y usuarios finales.

El desarrollo con XP proporciona rápidamente resultados. Al trabajar con pequeñas iteraciones, se pueden obtener con frecuencia comentarios del cliente para que el producto final cubra ampliamente sus expectativas y necesidades. Como en otros métodos ágiles, la forma de crear el producto es incremental.

Para XP las pruebas son la base de la construcción y propone que sean los desarrolladores

los que escriban las pruebas a medida que van construyendo el código.

También favorece la integración continua con objeto de estabilizar el software. Las pruebas automáticas se realizan de forma constante para poder detectar los fallos rápidamente. Cuanto antes se detecte un problema, antes podrá resolverse sin que las consecuencias y el impacto sean mayores.

Antes de cada iteración se planifica el trabajo que va a realizarse y a continuación se realizan de forma simultánea el análisis, el desarrollo, el diseño y las pruebas del código.

XP se basa en un conjunto de valores**:** comunicación, simplicidad, *feedback*, valentía, y respeto. Como los valores de XP son demasiado abstractos, es necesario concretar para ponerlos en práctica. Para ello XP propone algunos principios útiles para un mejor desarrollo como: economía, búsqueda del beneficio mutuo, mejora continua, reflexión, simultaneidad de fases o flujo, redundancia buscando soluciones, aprender de los fallos, búsqueda constante de la calidad, o aceptar la responsabilidad de todos los implicados en el desarrollo de un producto.

La aplicación de estos valores y principios a un proyecto específico de desarrollo de software se hará aplicando las prácticas que XP propone.

Kent Beck redefinió XP más recientemente (2004) basando el desarrollo en trece prácticas primarias y otras añadidas a modo de corolario. La aplicación de las prácticas primarias aporta un beneficio directo en la creación de software. Las prácticas-corolario deben usarse siempre y cuando las primarias ya se apliquen porque se asume que existe una madurez y experiencia en el desarrollo.

Las prácticas primarias son las siguientes:
- Trabajar con historias de usuario.
- Realizar ciclos semanales de desarrollo.
- Organizar revisiones trimestrales.
- Trabajar con holgura.
- El equipo debe estar próximo físicamente.
- El equipo debe ser completo, es decir, compuesto por todas las personas necesarias para llevar a cabo el producto con éxito.
- Tener información sobre el proyecto en el puesto de trabajo.
- Mantener la energía en el trabajo ó un ritmo sostenible.
- Realizar con frecuencia la programación en parejas.
- Diseño incremental.
- Realizar las pruebas antes de programar.
- Construir en diez minutos.
- Realizar integración continua.

En cuanto a las prácticas corolario se puede destacar el contar con la participación real de los clientes, mantener la continuidad de los equipos, o compartir código.

8. Conclusiones

Los métodos ágiles son una forma de trabajo nacida en la fabricación industrial que se está extendiendo rápidamente al mundo del desarrollo de software. Implicación del cliente, foco en la calidad, colaboración, jugar con la incertidumbre, usar ciclos iterativos, flexibilidad… son algunas de sus características más destacadas.

En este artículo se presentan de forma muy resumida algunos de los métodos más aplicados actualmente, en especial Scrum, Kanban y XP.

Antes de plantearnos la adopción de uno u otro método es muy conveniente conocer sus fundamentos y premisas básicos. Este conocimiento servirá para ayudar a inclinarnos por uno u otro y a hacer las adaptaciones necesarias para aplicar los métodos ágiles dentro de nuestra organización.

Referencias

[1] Alonso Álvarez García, Rafael de las Heras del Dedo, Carmen Lasa Gómez. *Métodos Ágiles y Scrum*. Anaya Multimedia, 2012. ISBN: 978-84-415-3104-8.
[2] Kent Beck et al. *Manifiesto por el Desarrollo Ágil de Software*. <http://agilemanifesto.org/iso/es/manifesto.html>, 2001.
[3] Henrik Kniberg, Mattias Skarin. *Kanban and Scrum - making the most of both. Minibook* descargable desde <http://www.infoq.com/minibooks/kanban-scrum-minibook/ >, 2009.
[4] Mary Poppendieck, Tom Poppendieck. *Lean Software Development. An Agile Toolkit*. Alistair Cockburn and Jim Highsmith, Series Editors, 2003. ISBN: 0-321-15078-3.
[5] Stephen Palmer, Mac Felsing. *A Practical Guide to Feature-Driven Development*. Upper Saddle River, NJ, Prentice Hall. 2002. ISBN: 0130676152.
[6] DSDM Consortium. <http://www.dsdm.org/>.
[7] Dean Leffingwell. *Scaling Software Agility. Best Practices for Large Enterprises*. Addison-Wesley Professional, 2007. ISBN: 0321458192.
[8] Kent Beck. *eXtreme Programming Explained*. Reading, Mass., Addison-Wesley. 1999. ISBN: 0201616416.

Nota

[1] *El Sprint es el período en el cual se lleva a cabo el trabajo en sí. Es recomendado que la duración de los sprints sea constante y definida por el equipo con base en su propia experiencia*. <http://es.wikipedia.org/wiki/Scrum>.