# Conocimientos generales para la puesta en marcha de un equipo de desarrollo ágil de forma práctica.

Carlos-Helder García Escobar

26 de enero de 2013

carloshelder.garciaescobar@alum.uca.es
http://estiloasertivo.blogspot.com.es/
http://estiloasertivo.16mb.com/

**Resumen**

Artículo de Investigación para la asignatura de Ingeniería de Requisitos donde se describe de forma general qué es el manifiesto ágil y cuáles son las metodologías ágiles más populares en la actualidad. No obstante, el esfuerzo de este artículo es explicar, de forma práctica, cómo se puede poner en marcha un equipo de desarrollo ágil de software, basándonos en el libro Do It Yourself Agile, Quick Start [Poole, 2008-2012].

## Introducción

A pesar de de haber cursado cuatro asignaturas sobre ingeniería del software a lo largo de las titulaciones de Ingeniería Técnica en Informática de Gestión e Ingeniería Técnica, el desarrollo ágil no formaba parte del temario de ninguna de ellas.

Al autor (yo) siempre le ha parecido que el prototipado evolutivo es la mejor forma de trabajar en la muy amplia mayoría de los casos. Esta investigación surge tras descubrir que hay un mundo detrás de los llamados prototipos evolutivos, que sí se explican de forma breve en la asignatura de ingeniería de requisitos. Y ese mundo es el desarrollo ágil.

Esta investigación ha sido pensada para ser presentada a los compañeros de clase de ingeniería de requisitos. Este documento se ha escrito después de la presentación y constituye un apoyo a la misma en caso de que quiera ser consultada en el futuro.

Este documento también puede leerse independientemente de la presentación y puede constituir una buena fuente para conseguir un conocimiento general útil sobre esta temática.

## Manifiesto Ágil

El manifiesto ágil se crea en febrero de 2001 tras una reunión de 17 expertos de la industria del software. Su objetivo fue esbozar los valores y principios que deberían permitir a los

equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

Tras esta reunión se creó The Agile Alliance, una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida es fue el Manifiesto Ágil, un documento que resume la filosofía ágil.

De las doce afirmaciones que constituyen el manifiesto ágil [Canós et al, 2003], se destacaron cinco en la presentación a los compañeros:

- I) La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
- V) Construir el proyecto entorno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
- VI) El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
- VII) El software que funciona es la medida principal de progreso.
- XII) En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

## Metodologías ágiles

De la gran cantidad de metodologías existentes, se han destacado tres: Scrum, Extreme Programming (XP) y Kanban. La primera, Scrum, por ser la clásica, la primera gran metodología y cuya terminología se usa en metodologías posteriores. La segunda, XP, por ser la más popular. Y la última, Kanban, por ser una de las más emergentes y que está dando que hablar.

### Scrum

Scrum es una metodología de desarrollo ágil, una manera de hacer las cosas procurando centrarse en el producto y no en toda la engorrosa documentación asociada. Scrum es flexible, nos permite ir adaptando el producto conforme vamos descubriendo cosas que no habíamos entendido bien o que no estaban bien definidas.Scrum nos permite disponer de algo tangible casi desde el primer momento de manera que el cliente tendrá una referencia sobre la que indicarnos si vamos por buen camino [Introducción a Scrum].

En Scrum hay diferentes roles :

- Propietario del Producto. Esta persona representa a los interesados en el producto final, es decir al cliente. Su tarea principal es conocer los requisitos del sistema.
- Equipo. Es quien convierte los requisitos en cosas tangibles.

- Scrum Master, Scrum Manager o Gestor. Su tarea es eliminar impedimentos, procurar que no haya colisiones y conocer Scrum para organizarlo todo. El Scrum Master no es «un jefe» es uno mas que carga con mas tarea.

En Scrum se usan distintos términos como:

- Product Backlog o Pila de Producto. Serie de tareas ordenadas de mayor a menor importancia.
- Sprint Backlog o Pila de Sprint. Lista de las primeras tareas del Product Backlog.
- Sprint Planning o Planificación del sprint. Reunión para seleccionar las tareas que formarán el Sprint backlog. Esta reunión tiene lugar al principio de cada sprint y define las tareas que serán realizadas en el siguiente sprint. Usualmente los sprint duran un mes.

## Extreme Programming (XP)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equi po, 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 sol uciones 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 [Canós et al, 2003].

La técnica utilizada para especificar los requisitos del software son las historias de usuario. Las historias de usuario son tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales.

En XP se diferencian los siguientes roles:

- Programador. El programador escribe las pruebas unitarias y produce el código del sistema .
- Cliente. Escribe las historias de usuario y las pru ebas 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.
- Encargado de pruebas (Tester). Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, di funde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- Encargado de seguimiento (Tracker). Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento d e l progreso de cada iteración.
- Entrenador (Coach). Es responsable del proceso global. Debe proveer guías al 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 pa ra el proyecto, en el que p uedan surgir problemas.

- 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.

### Kanban

Kanban no es nuevo para nada, surge como todas estas herramientas y disciplinas en el seno del sistema de producción de Toyota, "la máquina que cambió el mundo". Consiste en la generación de una serie de tarjetas que representan los productos que se están desarrollando y que actuan como testigos de una carrera de relevos a lo largo de la línea de producción, disparando los eventos de fabricación conforme van siendo necesarios. La ventaja que pregonan los partidarios de Kanban para el desarrollo de software es la reducción del número de reglas y obligaciones respecto a otras metodologías: nada de planificación de producto, nada de scrum diario, nada de retrospectivas; sólo flujo productivo, priorización y limitación del número de tareas en curso en un momento dado.

La ventaja que pregonan los partidarios de Kanban para el desarrollo de software es la reducción del número de reglas y obligaciones respecto a otras metodologías: nada de planificación de producto, nada de scrum diario, nada de retrospectivas; sólo flujo productivo, priorización y limitación del número de tareas en curso en un momento dado.

El problema estiba en que equipos sin experiencia previa en Ágile se vean atraidos por la simplicidad de Kanban y entonces no lleguen a interiorizar los importantes conceptos contenidos en los roles, procesos y artefactos de Scrum [Medinilla, 2010].

## Puesta en práctica desde cero de un equipo de desarrollo ágil.

Tras la lectura del libro Do It Yourself Agile, Quick Start [Poole, 2008-2012] y debido a la genialidad del mismo se decidió basar la presentación en este libro. El autor afirma que se trata de una mezcla de las tres metodologías que se han descrito en este artículo de investigación. El autor permite la descarga gratuita desde su blog de este libro [Poole, 2008-2012]. También podemos entrar en su página web[1] con otros propósitos, como preguntarle alguna duda o contratar sus servicios para que la instauración del sistema ágil de desarrollo software se haga de forma más eficiente.

A continuación se describen los pasos para crear un nuevo equipo ágil desde cero siguiendo el orden del libro:

### 1) Put Together The Team

Un equipo de desarrollo ágil debe estar formado por nueve personas como máximo y debe ser autosuficiente, esto es, no hay un equipo de desarrollo, otro de marketing, otro de

[1]Página web de la empresa de Damon Poole: http://www.valtivity.com

comunicación, etc. El equipo no debe depender de ningun otro departamento y debe ser capaz por si solo de crear un producto final. Así se evitan conflictos entre departamentos que retrasen o empeoren la calidad del producto.

Todo el equipo debe trabajar de cerca unos de otros y así cumplir una de las máximas del manifiesto ágil: el diálogo cara a cara es la mejor forma de comunicación.

Cada uno tendrá sus responsabilidades particulares, pero se "premiará" al equipo en su conjunto, fomentando la ayuda entre unos y otros, hasta que no acaben todos no hay premio.

Por último y si se quiere seguir esta forma de trabajar recomendada en este apartado, todos los integrantes del grupo deberían haberse leído previamente el libro en que se basa este apartado [Poole, 2008-2012].

## 2) Get Planning Poker Cards For Your Whole Team

Para estimar la duración en implementar y dejar completamente listo cada una de las característica a desarrollar se utilizan "planning poker cards."Esta baraja contiene cartas numeradas como sigue: $\frac{1}{2}$, 1, 2, 3, 5, 8, 13, 21 y ya he tenido suficiente. Todos los integrantes del equipo deben poseer una baraja completa para estimar.

La forma de estimar es la siguiente. Todos los miembros del equipo se sientan cara a cara en una mesa preferentemente redonda y se lee la característica en voz alta. A continuación todos sacan una carta que correponde lo que creen que será la duración que se tardará en dicha característica. Probablemente las cartas de los distintos miembros no concida con lo que se discutirá porqué uno a sacado una estimación u otro otra. Un valor bajo probablemente alguien se haya olvidado de algo, mientras que un valor alto implica que seguramente alguien se ha dado cuenta de algo que los demás no. Tras la discusión se vuelven a sacar las cartas y se continúa repitiendo este proceso hasta que todos coinciden en la misma estimación.

Una vez más entra en juego otras de las máximas del manifiesto ágil, se debe motivar al equipo. Esta forma de estimar no ha sido diseñada únicamente por ser buena en estimar, sino también con una función motivadora. Jugando a las planning poker cards todos todos opinan, todos participan, y la decisión se toma entre todos, fomentando el espíritu de equipo.

Por otra parte, en la presentación que se realizó a los compañeros se usaron las mano para simplificar el proceso. De forma que un dedo significaba que estimaba 1, etc.

## 3) Select Your Backlog and User Story Management Tool

El autor de libro sugiere varios sistemas para ayudar a la gestión del equipo y proyecto:

- De propósito general: Excel, Defect Tracking System.
- Específicos para desarrollo ágil: Jira, Rally, Version One.
- Post-it

El autor del libro sugiere empezar usando post-it y este también ha sido el método usado en la presentación.

Pero previamente puede ser necesario explicar qué es un backlog y qué es un user story. En Scrum, el backlog es la lista de requisitos que se están desarrollando, ordenada según las preferencias del cliente. Por otra parte, una historia de usuario es una descripción muy breve de una característica que el cliente quiere que se implemente. Hay que recalcar que no se trata de un requisito; es el ingeniero quien tiene libertad de crear el requisito que crea conveniente a partir de la historia de usuario del cliente.

En la presentación se propusieron dos características ejemplo a estimar en el diseño de una aplicación que facilitase la organización de competiciones de fúbol sala: administrador puede dar de alta los equipos y equipo puede añadir jugadores.

## 4) Create the Initial Backlog, Velocity, and Iteration Length

Crear el primer backlog es difícil por la razón que explicaré a continuación y que es vital para conseguir una correcta estimación. Los números que se estiman con las planning poker cards, story points, no son días ni cualquier unidad de tiempo. Se trata de otros proyectos conocidos por todos. Por ejemplo, 1 podría ser modificar nombre de empleado y 2 podría ser crear nuevo empleado (puesto que creemos que se tardará más), si todos hubiesen trabajado anteriormente en un proyecto en el que hubiesen tenido que desarrollar estas características. Entonces, estimar con 1 significaría que se estima que se tardará lo mismo que se tardó entonces en desarrollar dicha característica. Calcular el número de días es muy difícil y esta forma de estimar soluciona esa problemática con honores. Volviendo atrás, la dificultad de este primer backlog es que posiblemente todavía no han trabajado juntos en ningún proyecto similar y, por tanto, no es fácil elegir las historias de ejemplo.

En la presentación se eligieron como historias ejemplo asuntos relacionados con la asignatura, Ingeniería de Requisitos: estudiar, realizar el trabajo de la asignatura, preparase la presentación, ir a clase).

Por otra parte, velocity son la suma story points multiplicados por los puntos de historias finalizadas. Esta medida ayuda a planificar mejor en futuras ocasiones pero no se le ve utilidad en esta primera iteración.

Por último, iteration length se explica justo a continuación.

## 5) Select Your Iteration Length

La duración de la iteración es lo que queremos que dure una iteración. El objetivo de la iteración es tener algo útil y funcional para el cliente al finalizarla.

Una vez comenzada la iteración no se permite la inclusión de nuevas características salvo rara excepción.

La duración de la iteración debería estar entre una a seis semanas. Para empezar, el autor de libro recomienda treinta días para posteriormente modificarse para ajustarse al rendimiento del equipo si así se considera necesario.

Es importante resaltar que la primera iteración debe verse como una meta y no como un requisito estricto. Hacerla como un requisito estricto probablemente provocaría que no se haga bien y más bien se haga una chapuza durante los últimos días para poder terminar a

tiempo. Por la misma razón, no se recomienda hacer release tras ésta. Del release se habla a continuación.

## 6) Plan Your Release

A diferencia de Scrum y otras metodologías, el autor de este libro diferencia las iteraciones y los release del producto. El release o lanzamiento del producto se decide entre alguno de estas opciones:

- Release by Date. Si lo importante es entregar el producto en una determinada fecha, el release se hará cuando sea dicha fecha, haya lo que haya hecho.
- Release by Content. Si lo importante es que haya una característica o una serie de características acabada, se calcula el tiempo que se tardará y se planea el lanzamiento para entonces.
- Release at Will. Se hace el relase cuando se crea conveniente. Es la mejor forma pero también la más difícil de llevar a cabo en la práctica.

## 7) Schedule All of the Agile Meetings

Se deben realizar cuatro tipos de reuniones:

1. Iteration planning. Al principio de la iteración se decide quién va a hacer la característica (no hay un "jefe" que asigna tareas) y qué y si faltan o sobran características a desarrollar para rellenar la iteración.
2. Daily Standup. Todos los días a la misma hora se hace esta reunión que no debe durar más de quince minutos y que debe hacerse de pie. Durante esta reunión cada uno responde a las siguientes preguntas:
   *a*) Qué ha hecho desde la última reunión
   *b*) Qué es lo siguiente que van a realizar
   *c*) Dificultades encontradas. Aquí no se discuten las dificultades, sólo se nombran. Posteriormente se podrán discutir gracias a que todos estarán sentados cerca unos de otros.
3. Iteration review. Al acabar la iteración se hace una demostración de lo realizado al equipo, stakeholders (cualquiera que pueda estar interesado) y clientes, si es posible.
4. Retrospective. Al empezar la siguiente iteración se realiza esta reunión en la que se discuten los distintos aspectos que se podrían mejorar y se eligen uno o dos para la iteración que empieza.

## 8) Go!

Y eso es todo. ¡El equipo está listo para crear! ;)

## Conclusiones

Esta investigación no ha hecho más que afianzar la idea de desarrollo ágil para todo. Los escritos sobre desarrollo ágil suelen hablar de que se trata de una metodología recomendada para proyectos pequeños y de requisitos cambiantes. Pero tras el estudio la conclusión es distinta. Ahora más que nunca se cree que debería seguirse siempre esta forma de trabajo en todos los proyectos salvo en un pequeño grupo, como son aquellos en los que los requisitos son fijos desde el principio. Ejemplo de esa minoría puede ser el diseño de un compilador de lenguajes de programación. No obstante, la muy amplia mayoria de proyectos están ligados a un cliente y esta forma de trabajo es la mejor que se conoce para conseguir de la forma más precisa posible lo que el cliente quiere.

La importancia explícita que se le da a la motivación del equipo en el manifiesto ágil y en las metodologías ha sorprendido gratamente. Un equipo motivado es un equipo feliz. Y un equipo feliz y motivado trabaja mejor.

Pero esta investigación acaba con un sabor agridulce. Dulce por todo lo aprendido y haberse podido mostrar a los compañeros de clase. Agrio por saber que el conocimiento de la forma de trabajo ágil sigue sin ser parte obligatoria de los temarios de Ingeniería Informática. Quizás en el futuro todos trabajemos de esta manera y seamos más felices.

Como recomendación práctica, no se quiere terminar este artículo sin recomendar la lectura del libro en que se basa [Poole, 2008-2012], es un gran libro. Y por supuesto, se recomienda la divulgación de este artículo o cualquier otro que ayude a la difusión de esta gran forma de trabajo.

## Referencias

[Poole, 2008-2012] Damon B. Poole. 2008-2012. Do It Yourself Agile, Quick Start. Disponible en: http://www.valtivity.com/uploads/DIYAgileKickstart.pdf

[Canós et al, 2003] José H. Canós, Patricio Letelier y Mª Carmen Penadés. 2003. Métodologías Ágiles en el Desarrollo de Software. Disponible en: http://www.willydev.net/descargas/prev/TodoAgil.Pdf

[Introducción a Scrum] Autor desconocido. Año desconocido. Introducción a Scrum. Libro suministrado por la empresa Codeko Informática. Disponible en: http://codeko.com/descargas/Scrum.pdf

[Medinilla, 2010] Ángel Medinilla. 28 de enero de 2010. Kanban vs Scrum en castellano [Blog Internet]. Disponible en: http://www.presionblogosferica.com/2010/01/28/kanban-vs-scrum-en-castellano/