

# Un Mejor Scrum

Un conjunto no oficial de consejos e ideas sobre cómo implementar Scrum

Escrito por el Coach y Entrenador Certificado de Scrum Peter Hundermark

Traducción por Alan Cyment, Entrenador Certificado de Scrum



Leaders in facilitating lasting Agile change

[scrumsense.com](https://scrumsense.com)

# ¿Por qué esta guía?

Jim York, Entrenador Certificado de Scrum, dijo una vez: **Scrum es Simple. Hacer Scrum es difícil.** <sup>SM</sup>

Una gran parte de aquellos con los que he trabajado en distintas organizaciones dicen que les resulta muy difícil saber por dónde empezar a la hora de utilizar Scrum. Otras organizaciones tienen equipos que ya están siguiendo varias prácticas ágiles y, sin embargo, están muy lejos de convertirse en lo que Jeff Sutherland describió alguna vez como *hiper-productivos*.

Espero de corazón que este pequeño libro sea una fuente de inspiración para ayudarte a practicar un mejor Scrum. Por sobre todas las cosas aspiro a que te ayude a recorrer el duro camino que va de las viejas formas de trabajar que, simplemente, no funcionan hacia nuevas maneras de encarar la labor cotidiana, que te permitirán obtener más calidad, entregar antes y, sobre todas las cosas, pasarla un poco mejor.

Por favor hazme saber si ha sido de tu agrado y cómo puedo hacer para mejorarla.

¡Y ahora, simplemente ve y hazlo!

*Peter Hundermark*

*Ciudad del Cabo, noviembre de 2009*

*Segunda edición*

*Traducción: Alan Cyment*

*Buenos Aires, agosto de 2011*

# ¿Qué es Scrum?

## Orígenes

Scrum es un framework para el manejo de proyectos que tienen como fin el desarrollo de productos complejos. Scrum tiene sus orígenes en los campos del manejo del conocimiento, los sistemas adaptativos complejos y la teoría de control empírico de procesos. Ha sido influenciado también de patrones observados durante el desarrollo de software y la Teoría de las Limitaciones.

## Scrum y Agilidad

Scrum es el más popular de los métodos Ágiles. Se utiliza frecuentemente en conjunto con Extreme Programming (XP).

## ¿Cuál es el problema?

- ▶ Los releases toman demasiado tiempo
- ▶ La estabilización toma demasiado tiempo
- ▶ Los cambios son difíciles de llevar a cabo
- ▶ La calidad decae
- ▶ Las marchas mortuorias no dejan de herir la moral

Durante décadas los desarrolladores de software han intentado emplear métodos definidos de trabajo y administración de proyectos. Los métodos definidos son apropiados cuando las variables que ingresan al sistema se encuentran bien definidas y el método por el cual se las transforma en resultados es predecible. El desarrollo de software y otras formas de trabajo complejos no encajan con esa clase de métodos. El alto grado de proyectos fallidos e insatisfacción de los cliente ilustra esto con claridad.

## ¿Cómo ayuda Scrum a resolver esto?

Alistair Cockburn [Cockburn 2008] describe al desarrollo de software como un ‘juego cooperativo de invención y comunicación’.

Las metodologías de trabajo tradicionales se basan en documentos para plasmar y comunicar conocimiento de un especialista al siguiente. Los ciclos de retroalimentación son demasiado largos o incluso no existen. Décadas de proyectos con bajísima productividad demuestran que esta manera de trabajo lleva indefectiblemente al fracaso.

Scrum provee una plataforma para que la gente trabaje en conjunto de manera efectiva, a la vez que permite exponer de manera implacable cualquier problema que se ponga en su camino.

## La esencia de Scrum

La esencia de Scrum es:

- ▶ El equipo recibe objetivos claros
- ▶ El equipo se organiza en función del trabajo a realizar
- ▶ El equipo entrega con regularidad las funcionalidades más valiosas
- ▶ El equipo recibe retroalimentación de individuos que se encuentran fuera del equipo
- ▶ El equipo reflexiona sobre su manera de trabajar, con el objetivo de mejorar
- ▶ La organización completa posee visibilidad sobre el progreso del equipo
- ▶ El equipo y la gerencia se comunican entre sí de manera honesta, transparentando progreso y riesgos.

Esta forma de trabajar se basa en los siguientes valores: respeto por uno mismo y por los otros, confianza y coraje.

## ¿Por qué Scrum no hace mención de ninguna práctica de desarrollo de software?

Scrum no tiene como objetivo dar instrucciones a los equipos sobre la forma en la que deben llevar a cabo su trabajo. Scrum espera que los equipos hagan lo que sea necesario para entregar el producto esperado, brindándoles la potestad para ello. Las prácticas y herramientas de desarrollo cambian y mejoran de manera continua y los buenos equipos trabajarán constantemente en pos de obtener el mejor uso de las mismas.

## Aplicabilidad

Si bien es cierto que Scrum fue utilizado por primera vez para desarrollar productos de software, está diseñado para cualquier tipo de trabajo complejo. Hoy en día se utiliza para gerenciar desarrollo de software y hardware, publicidad, marketing, iglesias y organizaciones enteras.

## ¿Cómo se mapea Scrum con los métodos tradicionales?

La respuesta rápida es que no existe tal mapeo. La agilidad y Scrum están basados en un paradigma distinto. Los fundadores Jeff Sutherland y Ken Schwaber han

repetido en numerosas ocasiones que es inútil intentar mapear métodos definidos con un método empírico.

## ¿Tendrá éxito Scrum en mi organización?

¡Eso depende de ti! La implementación en tu organización puede llegar a fallar debido a la falta de voluntad de sus integrantes para resolver los problemas que Scrum expondrá. Sin embargo miles de equipos en todos los continentes y en decenas e industrias están convirtiendo día a día su mundo laboral en uno mejor que el que era ayer.

# El manifiesto ágil

En febrero de 2001 diecisiete expertos en metodologías “livianas” se encontraron para discutir y tratar de llegar a una definición en común de las maneras de trabajo que estaban utilizando. Entre ellos se encontraban Jeff Sutherland y Ken Schwaber, fundadores de Scrum, junto a Mike Beedle, quien trabajó en desarrollar los patrones iniciales y en la redacción del primer libro sobre Scrum. Este grupo se autodenominó “The Agile Alliance” (la Alianza Ágil) y definió de común acuerdo el Manifiesto para el Desarrollo Ágil de Software. Definió también un conjunto de doce principios que respaldan el manifiesto. Ambos se encuentran reproducidos en la página siguiente

## Comentario

El Manifiesto Ágil y sus doce principios han permanecido incólumes durante ya diez años. Continúan siendo hoy en día la mejor manera de juzgar si un método de trabajo es o no realmente ágil. La mayor crítica que ha recibido se debe al sesgo que presenta hacia el desarrollo de software, dado que los métodos ágiles pueden aplicarse en muchos más campos.

## Nota

Scrum es una manera particular de desarrollo ágil. Seguir el Manifiesto Ágil y sus principios no implica necesariamente que una organización o equipo esté utilizando Scrum. ¡Sin embargo, el incumplimiento de CUALQUIERA de estos principios implica que no se está utilizando Scrum!

## Manifiesto por el Desarrollo Ágil de Software

Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:

Individuos e interacciones sobre procesos y herramientas

Software funcionando sobre documentación extensiva

Colaboración con el cliente sobre negociación contractual

Respuesta ante el cambio sobre seguir un plan

Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda.

# Principios del Manifiesto Ágil

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

*Highsmith (2001)*

# Los roles en Scrum

## Introducción

No existe el rol de gerente de proyecto en Scrum. Las responsabilidades del clásico gerente de proyecto se encuentran divididas en los tres roles que conforman el Equipo Scrum:

- ▶ El Product Owner gestiona el *producto* (y el retorno de la inversión)
- ▶ El ScrumMaster gestiona el *proceso*
- ▶ El equipo se gestiona *a sí mismo*

Esto resulta desafiante para aquellos que hoy en día desempeñan ese rol y para los gerentes de las organizaciones en las que trabajan. Michele Sliger y Stacia Broderick han escrito una útil guía de transición de gerente de proyecto a coach ágil [Sliger y Broderick 2008].

No existen líderes designados en el Equipo Scrum más allá del Product Owner y el ScrumMaster; no es necesario. La necesidad de gerentes de línea se reduce dado que los equipos se administran a sí mismos en un gran abanico de responsabilidades. No es extraño encontrar 50 miembros de equipos que reportan directamente a un único gerente de línea en una organización que ha hecho la transición a Scrum.

## Auto-organización

La auto-organización no implica en absoluta que se siga un enfoque del tipo *laissez-faire*; muy por el contrario, los equipos auto-organizados son altamente disciplinados. Dado que se les otorga completa autonomía, se espera de ellos una mayor responsabilidad a la hora de cumplir con los compromisos que han tomado. Se fomenta que tomen riesgos razonables y que aprendan a través del error y la introspección. Un alto grado de confianza y compromiso es un resultado que se obtiene siempre que se forme un equipo verdaderamente auto-organizado.

Aquellos equipos que son nuevos en Scrum requieren el aliento necesario para explorar nuevos y mayores límites, así como también responder al desafío que implica una mayor responsabilidad. Deben superar a menudo las trampas que les imponen las antiguas y poco productivas formas en las que realizaron su trabajo durante años.

La auto-organización no es una opción en Scrum; es un principio básico. Sin ella nunca tendremos equipos de alta performance. *¡Caveat emptor!*



# Product Owner

Las responsabilidades del rol de Product Owner son las siguientes:

- ▶ Obtención de una visión compartida
- ▶ Recolección de requerimientos
- ▶ Administración y priorización del Product Backlog
- ▶ Aceptación de software al final de cada iteración
- ▶ Administración del plan de release
- ▶ Maximización del retorno de inversión del proyecto

Metáfora: El Product Owner es un CEO

# ScrumMaster

Las responsabilidades del rol del ScrumMaster son:

- ▶ Aseguramiento de un entorno de trabajo para el equipo, protegido de interferencias y directivas
- ▶ Remoción de impedimentos
- ▶ Fomento del uso y respeto al proceso
- ▶ Extensión del uso de Scrum a lo largo de la organización

Metáfora: ¡El ScrumMaster es un facilitador, coach, mentor y aplanadora!

# Equipo

Las responsabilidades del rol Equipo o Miembro de Equipo son:

- ▶ Estimación del tamaño de los ítems del backlog
- ▶ Compromiso de entregar incrementos de software con calidad de producción
- ▶ Seguimiento de su propio avance
- ▶ Auto-organización, aunque con la responsabilidad ante el Product Owner de entregar aquello que fue comprometido

- ✿ *Los miembros del equipo pueden ser desarrolladores, testers, analistas, arquitectos, redactores, diseñadores e incluso usuarios. El equipo es multidisciplinario, lo que significa que entre todos sus miembros poseen las habilidades suficientes para llevar a cabo el trabajo. No existe un liderazgo definido de antemano entre los miembros del equipo.*
- ✿ *El Equipo Scrum comprende los tres roles: un Product Owner, un ScrumMaster y entre cinco y nueve Miembros de Equipo.*

# Las reuniones durante el Sprint

El sprint es el latido del ciclo de Scrum. La *planificación* y la seguidilla de *revisión* y *retrospectiva* marcan respectivamente el comienzo y fin del sprint. La longitud del sprint se encuentra fija y jamás se extiende. La gran mayoría de los equipos Scrum eligen una duración de sprint de dos, tres o a lo sumo cuatro semanas. Durante el sprint el equipo lleva a cabo una *reunión diaria*. Cada reunión en Scrum tiene una duración máxima fijada a priori (concepto denominado *timeboxing* en la jerga de Scrum). Para un sprint de cuatro semanas el tiempo que suele dedicarse a la primer y segunda parte de la reunión planificación, así como también a la revisión y retrospectiva suele fijarse en cuatro horas cada una. Para sprints más cortos deberán ser ajustadas en proporción con la duración del sprint.

Algunos de los atributos clave de las distintas reuniones se describen en las secciones que siguen a continuación. Pero antes quisiera compartir con ustedes algunas experiencias personales que creo vale la pena relatar.

- ☼ *Considero que una longitud de dos semanas para los sprints suele ser muy útil para comenzar. Luego de tres sprints lo mejor es dejar que el equipo decida el ritmo con el que desea trabajar.*
- ☼ *Los grupos necesitan tres sprints para comenzar a entender los nuevos conceptos, deshacerse de viejos hábitos y comenzar a moldearse como un verdadero equipo.*
- ☼ *Nunca realice la planificación de sprints los lunes a la mañana. El equipo no está con todas sus luces y suele ser el día en el que se producen la mayor cantidad de ausencias por enfermedad. Nunca realice revisiones o retrospectivas los viernes por la tarde. El equipo está cansado y pensando en el fin de semana. Por lo tanto es una buena idea considerar comienzo y fin de los sprints entre martes y viernes.*
- ☼ *Aquellos equipos que trabajan en sprints de dos semanas suelen pensar que es una buena idea tener las reuniones de comienzo y fin de sprint en un sólo día. En otras palabras, comenzar el día con la revisión, luego la retrospectiva; luego del mediodía partes 1 y 2 de la planificación. La idea es poder terminar con todas las reuniones lo antes posible, de modo de tener 9 días disponibles para trabajar. En mi experiencia este enfoque conlleva dos problemas:*
  - ✦ *El equipo no considera que estas reuniones son parte del trabajo—¡en los hechos, son la parte que más influye en el éxito del resto del sprint!*
  - ✦ *Durante la última parte del día —planificación de sprint parte 2—el equipo está mentalmente agotado.*

*Sin embargo, como siempre, ¡deja que el equipo prueba esa alternativa si así lo desean!*

## Planificación del Sprint - Parte 1

La Parte 1 de la Planificación del Sprint (PS1) es en realidad un *workshop de toma de requerimientos* detallado. El Product Owner presenta la serie de funcionalidades que desea que sean implementadas y el equipo realiza las preguntas necesarias para comprender los requerimientos con el detalle suficiente que le permita comprometerse a entregar dichas funcionalidades al final del sprint. El equipo decide por sí mismo cuánto puede entregar, considerando la duración del sprint, el tamaño del equipo y las habilidades y disponibilidad de sus miembros, la definición de LISTO y cualquier acción a tomar decidida durante la retrospectiva que precedió a esta reunión.

El Product Owner debe estar presente durante esta reunión para poder guiar al equipo en la dirección correcta y responder preguntas—que seguramente serán muchas. El ScrumMaster debe asegurarse que cualquier stakeholder cuya presencia sea necesaria para que el equipo comprenda los requerimientos esté en la reunión, al menos telefónicamente.

Cualquier ítem del backlog que se desea sea desarrollado durante el sprint que no haya sido estimado con anterioridad será estimado de manera inmediata durante la reunión. Esto no es, sin embargo, una excusa para evitar la preparación del backlog (también conocida como Backlog Grooming - ver más adelante).

Al finalizar PS1 el equipo se compromete ante el Product Owner a desarrollar aquello que consideran podrá ser entregado testeado y funcionando al finalizar el sprint. Un equipo experimentado podrá usar su velocidad histórica para predecir su capacidad (“el clima de ayer”). Esto es conocido como *planificación basada en velocidad* (*velocity-based planning*). Mi recomendación para la gran mayoría de los equipos es que lleven a cabo una *planificación basada en el compromiso* (*commitment-based planning*). El conjunto de los ítems del backlog que han sido comprometidos se denominará *product backlog seleccionado*.

## Planificación del Sprint- Parte 2

Si la parte 1 es un *workshop de requerimientos*, la parte 2 de la planificación del sprint (PS2) es un *workshop de diseño*. Durante esta sesión el equipo colabora de forma tal de crear un diseño a alto nivel de los ítems a los que se ha comprometido. Un resultado de la reunión será el backlog del sprint, o la lista de tareas que el equipo debe ejecutar de manera colectiva para poder entregar en forma de funcionalidades testeadas. Esta serie de tareas se denomina backlog del sprint (*sprint backlog*) y suele representarse en un tablero de tareas (*taskboard*) físico.

Durante PS2 el equipo podría tener preguntas adicionales con respecto a los requerimientos. El ScrumMaster debe asegurarse que el Product Owner y, si fuera necesario, otros stakeholders se encuentren disponibles para poder responder estas preguntas.

El diseño, como todo lo demás en el desarrollo ágil, es emergente. Además, la reunión está timeboxeada (ie. tiene un límite pre-establecido de tiempo). Por ende es normal que el equipo no arribe a un diseño completo durante esta sesión y que descubra nuevas tareas durante el sprint. Esto no es un signo de alarma. Simplemente tomarán un post-it, un marcador y crearán nuevas tareas cuando sea necesario.

☼ *Sabrás que PS2 está funcionando cuando el equipo se encuentre reunido alrededor de una pizarra discutiendo airadamente sobre “la mejor” o incluso “la manera correcta” de implementar una funcionalidad.*

## Reunión diaria

La reunión diaria (*daily meeting*) es uno de los tres puntos de *inspección y adaptación* en Scrum. El equipo se reúne para comunicar y sincronizar su trabajo. Dado que el equipo trabaja de manera colaborativa este momento es esencial para asegurar un progreso continuo y evitar bloqueos. Además el equipo medirá permanentemente su propio progreso en términos del *objetivo del sprint (sprint goal)*.

La reunión diaria NO tiene como objetivo reportar progreso al ScrumMaster, Product Owner o cualquier otro stakeholder. El Product Owner podrá participar de la misma siempre y cuando se mantenga en posición pasiva, hablando únicamente cuando se le realicen preguntas. El ScrumMaster debe asegurarse, utilizando su habilidades como facilitador, que todos los miembros del equipo se hayan comprometido a realizar cierto trabajo en las próximas 24 horas y que este trabajo tenga como objetivo exclusivo ayudar a que el equipo entregue el siguiente ítem del backlog. El ScrumMaster deberá, a su vez, asegurarse que cualquier impedimento que dificulte la tarea del equipo sea removido lo antes posible. El ScrumMaster tiene también la responsabilidad de restringir la reunión a un máximo de 15 minutos. Sorprendentemente las reuniones pueden ser efectivas en ese breve lapso de tiempo (incluso suelen durar menos tiempo).

Jason Yip [Yip 2006] ofrece una útil guía para ayudar a los ScrumMasters a llevar a cabo de manera óptima la reunión.

## Revisión del Sprint

La *revisión del sprint* (*sprint review*) es a veces denominada de forma incorrecta como *demo*. Si bien es cierto que se incluye en ella una demostración de las nuevas funcionalidades que el equipo desarrolló durante el sprint, su principal objetivo es *inspeccionar* lo entregado por el equipo y obtener feedback de los asistentes a la reunión para poder *adaptar* el plan para sprints subsiguientes. Es por ende mucho más que una mera demo.

Cuando se me pregunta quién debe ser invitado a la *revisión del sprint* suelo contestar “todo el mundo”. Mi intención es ayudar al ScrumMaster y a la organización completa a entender que es crucial maximizar el valor entregado por el equipo al concluir el sprint. Las *revisiones del sprint* tienen muchas posibles consecuencias, incluyendo la cancelación del proyecto. En la mayoría de las veces se autoriza al equipo a continuar trabajando en un siguiente sprint y se decide el objetivo para el mismo.

## Retrospectiva

La retrospectiva es la última reunión del sprint. Sigue inmediatamente después de la revisión y nunca debe ser omitida.

Mientras que la revisión está centrada en el producto, la retrospectiva se encuentra enfocada en el *proceso*—la manera en la que el equipo Scrum trabaja de manera conjunta, incluyendo habilidades técnicas y prácticas y herramientas de desarrollo.

Si bien la revisión se encuentra abierta a quien desee asistir, la retrospectiva se restringe a los miembros del equipo Scrum—esto es el Product Owner, los miembros del equipo de desarrollo y el ScrumMaster. Cualquier persona ajena, incluyendo gerentes de cualquier nivel jerárquico, están excluidos, a menos que sean invitados por el equipo.

Esta regla debe ser comprendida en el marco del objetivo de la reunión, que es inspeccionar en profundidad cuán colaborativo y productivo es el equipo y cómo hacer para mejorar en ese sentido. Esto suele requerir de una profunda introspección, lo que a su vez requiere un ambiente seguro, donde nadie se sienta intimidado. La directiva básica para las retrospectivas según Norman Kerth acentúa esta necesidad: ‘Sin importar lo que sea descubierto, entendemos y creemos que todos hicieron lo mejor que pudieron, dado lo que sabían en ese momento, sus habilidades, los recursos disponibles y la situación en la que se encontraban.’ [Kerth 2001].

Boris Gloger [Gloger 2008] ofrece un patrón muy simple llamado *Retrospectivas de Latido de Corazón* (*Heartbeat Retrospectives*) para que nuevos equipos aprenden rápidamente a llevar a cabo retrospectivas. Esther Derby y Diana Larsen [Derby y Larsen 2006] proveen muchas actividades para que los facilitadores utilicen durante la reunión.

## Reunión de Estimación

Esta reunión no se menciona generalmente en la literatura sobre Scrum, pero es esencial si deseas obtener un flujo continuo de entregas por parte del equipo de las funcionalidades más valiosas que cumplan con el *criterio de hecho*.

Durante cada sprint el Product Owner organiza una o dos reuniones de las que debe participar todo el equipo Scrum y, si es necesario, otros stakeholders. Estos se reúnen para estimar el costo de nuevos ítems del backlog o recalcular el tamaño de ítems de gran costo que deberán ser subdivididos en otros más pequeños, de forma tal que puedan ser desarrollados en los próximos sprints.

- ✿ *Los equipos necesitan dedicar entre un 5 a 10% de su tiempo durante el sprint para poder preparar los sprints subsiguientes. La reunión de estimación descrita más arriba es un buen ejemplo. Otros ejemplos pueden ser workshops de redacción de historias de usuario de planificación de release. Es importante evitar el efecto arranque-freno al comienzo y fin del sprint. ¡La otra consecuencia es, por supuesto, que los equipos deberían utilizar entre el 90-95% de su tiempo en realizar trabajo propio del sprint actual!*

# Artefactos

Scrum define solamente cuatro artefactos:

- Product Backlog
- Sprint Backlog
- Burndown de tareas
- Backlog de impedimentos

Scrum no menciona adrede cualquier otra documentación y/o artefactos. Esto a menudo lleva al malentendido de que los equipos ágiles no necesitan ninguna documentación. Suelo sugerirle a los equipos que utilicen solamente aquellos artefactos que son realmente *muy* valiosos para ellos u otros, a corto y/o largo plazo. ¡Esto elimina trabajo y una innecesaria masacre de bosques enteros!

## Product Backlog

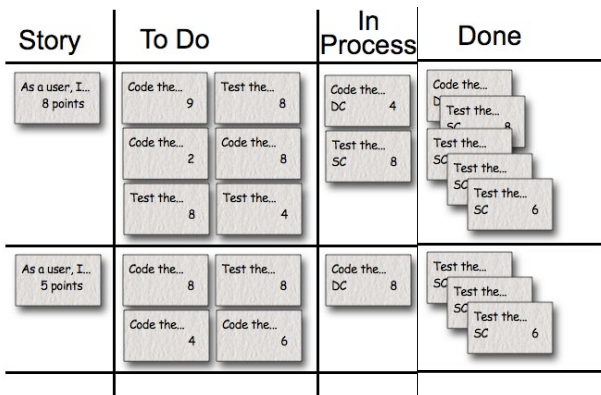
El product backlog es simplemente una lista de ítems que representan trabajo pendiente. Cualquiera puede agregar ítems al backlog, pero sólo el Product Owner tiene el derecho de determinar el orden en el que serán desarrollados por el equipo. Un buen Product Owner, por supuesto, negociará esto con los stakeholders y el equipo.

Los requerimientos son *emergentes*, lo que significa que no conocemos ni podemos conocer de antemano todas y cada una de las características que queremos que contenga el producto. Es por ello que Product Backlog es un documento viviente, que requiere una constante *preparación (grooming)* para mantenerlo actualizado y útil con el paso del tiempo. Se agregarán nuevos ítems; ítems existentes serán desagregados en múltiples ítems más pequeños; algunos ítems serán removidos al darnos cuenta que ya no son necesarios. Más aún, los ítems deben ser estimados para conocer la relación costo beneficio de los mismo, lo que influirá de manera directa en la ubicación que el mismo tendrá en el backlog.

En prácticamente todos los casos es suficiente, y generalmente preferible, crear y mantener un Product Backlog compuesto exclusivamente de un conjunto de historias de usuario escritas sobre fichas de 150 x 100 mm . Ron Jeffries [Jeffries 2005] acuñó la frase Tarjeta, Confirmación, Conversación (*Card, Confirmation, Conversation - las 3Cs*) para describir cómo debe trabajarse con las historias. Las historias suelen ser escritas desde la perspectiva de un usuario del producto. El libro de Mike Cohn sobre historias de usuario [Cohn 2004] te dirá todo lo que necesites aprender sobre las mismas.

## Sprint Backlog

La mayoría de los equipos conoce al *sprint backlog* como el *tablero de tareas*, que es simplemente una representación física del trabajo al que se han comprometido para lo que resta del sprint. El tablero de tareas es un ejemplo de un *kanban*, una palabra japonesa que significa “señal visual”. El mismo comunica al equipo y a cualquiera que desee saberlo qué tareas planificó el equipo y su estado actual.



Sprint Backlog

Adaptado de <http://epf.eclipse.org>

## Burndown del Sprint

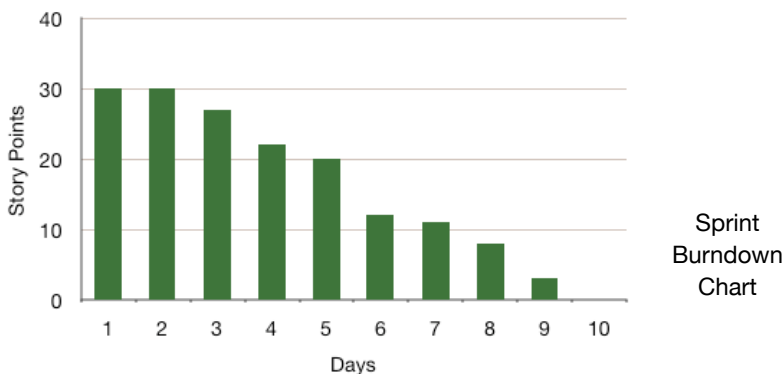
El gráfico de burndown de tareas del sprint está diseñado para ayudar al equipo en la monitorización de su progreso y para ser el indicador principal que informará sobre sus posibilidades de alcanzar su compromiso al finalizar el sprint. El formato clásico requiere que el equipo estime la duración de cada tarea en horas de forma diaria. El burndown deberá completarse de forma tal que grafique cuántas horas de trabajo restan para concluir el sprint

Yo recomiendo a los equipos que en su lugar construyan un gráfico que muestre cuántos puntos de historia resta hacer durante el sprint actual. La lógica detrás de esto es:

- ▶ La estimación de nuevas tareas y la reestimación de tareas que se encuentran en progreso requiere de un esfuerzo considerable
- ▶ La estimación de tareas es muy imprecisa
- ▶ La estimación en unidades de tiempo nos recuerda los viejos hábitos laborales
- ▶ La finalización de tareas no entrega valor alguno; sólo la historias completadas entregan valor

Por ende en mi trabajo como consultor el burndown del sprint es similar al burndown de release o producto, excepto que el eje X representa días en vez de sprints.



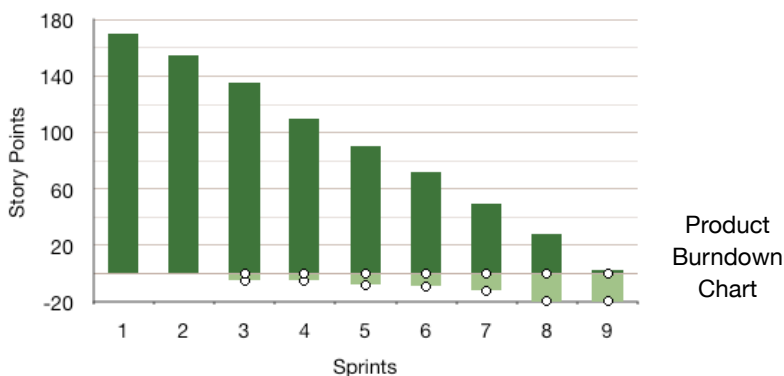


## Burndown de Release o Producto

El burndown de producto, también conocido como el burndown de release, mide el ritmo de entrega de funcionalidades testeadas a lo largo del tiempo. Este ritmo es conocido como la *velocidad del equipo*. Dado que las funcionalidades varían en complejidad y, por ende, en esfuerzo y tiempo, usamos una escala para poder comparar su tamaño. La unidad más conocida es la de *puntos de historia*. Una vez que los miembros del equipo han trabajado juntos por algunos sprints, generalmente puede decirse que han establecido una velocidad que se encuentra dentro de un cierto rango. Los Product Owners utilizan esta velocidad para predecir el ritmo con el cual el equipo va a entregar funcionalidad en el futuro, lo que lleva a tener planes de release cada vez más predecibles.

Suelo aconsejar a los equipos que utilicen una forma alternativa del burndown de producto que permite al Product Owner tener conciencia de los cambios realizados al product backlog. Esto es esencial, dada la naturaleza dinámica de esta lista.

Utilizando este gráfico los Product Owners podrán reportar progreso, determinar fechas de release y predecir el alcance del mismo.



## Backlog de Impedimentos

El backlog de impedimentos es simplemente la lista de situaciones que están impidiendo que el equipo progrese. Estas son situaciones que el ScrumMaster debe quitar del camino en su interminable gesta a través del cual ayudará al equipo para que trabaje de la mejor manera posible. ¡Los impedimentos van desde la imperiosa necesidad de que se arregle la máquina de café hasta reemplazar al CEO! Un buen ScrumMaster intentará remover los impedimentos dentro de las 24 horas posteriores a su identificación (bueno, tal vez no el CEO.)

# Comenzando con Scrum

Ken Schwaber [Schwaber 2007] dijo alguna vez que no se necesita hacer nada antes de comenzar a utilizar Scrum. Yo interpreto esto como un llamado a no personalizar el proceso antes de comenzar. Sin embargo prácticamente no existe literatura que nos ayude a la hora de ayudar a nuestro primer equipo Scrum a salir adelante sin ayuda del exterior.

Lo mejor que puedes hacer es contratar a un coach externo. Si eso no es posible, puedes tratar de utilizar un patrón que me ha funcionado con docenas de equipos..

Obviamente (espero) necesitas un equipo Scrum. Esto significa tener un Product Owner, un ScrumMaster y entre cinco y nueve miembros de equipo. Luego sigue esta serie de pasos, que será detallada en las próximas páginas.

1. Enseña al equipo las bases de Scrum
2. Establece la visión
3. Escribe historias de usuario para conformar el backlog
4. Ordena los ítem del backlog por valor de negocio
5. Estima el tamaño de los ítems del backlog
6. Reordena el backlog, según sea necesario, en base a factores adicionales
7. Crea el plan de release inicial
8. Planifica el primer sprint
9. ¡Comienza con tus sprints!

- ✿ *Comienzo mi trabajo con nuevos equipos con un entrenamiento de un día sobre lo esencial que hay que saber sobre Agilidad y Scrum. Esto es suficiente para comenzar a trabajar con un equipo que posee un ScrumMaster/Coach experimentado para brindarle apoyo durante sus primeros sprints.*
- ✿ *Los pasos 2 al 7 pueden ser fácilmente completados durante un workshop de definición de producto del que debería participar el equipo Scrum completo. Los mejores workshops son aquellos de los que participan otros stakeholders, como arquitectos, gerentes y otra gente del negocio.*

# Entrenamiento

Durante los entrenamientos que llevo a cabo utilizo mucho ejercicios grupales y juegos para ilustrar los principios. Cubro todos o la gran mayoría de los siguientes temas:

- ▶ El poder de la auto-organización
- ▶ Procesos empíricos versus procesos definidos
- ▶ El valor de trabajar en el mismo espacio físico
- ▶ La importancia de la confianza
- ▶ Principios de desarrollo ágil (El manifiesto ágil)
- ▶ El flujo de Scrum (ciclo de reuniones)
- ▶ Roles y responsabilidades (Los 3 roles de Scrum y más)
- ▶ Utilización de historias de usuario para representar requerimientos
- ▶ Estimación ágil utilizando planning poker
- ▶ Conceptos de tamaño y velocidad
- ▶ ¡Hecho!
- ▶ Utilizando un tablero de tareas
- ▶ Monitoreo de progreso (gráficos de burndown)
- ▶ Simulación de un sprint completo

## Definición de la Visión

Katzenberg y Smith [2002] han confirmado que tener metas claras es esencial para la conformación de equipos de alto rendimiento.

El Product Owner usualmente compartirá su visión del producto. Una técnica que utilizo es pedir a cada miembro del equipo que escriba su versión de la visión. Luego los formo en pares y les pido que lleguen a una visión que combine la de ambos. El proceso continúa con dos parejas trabajando del mismo modo y así, hasta que el equipo completo concluye formulando una única visión.

Este ejercicio utiliza el poder del grupo y obtiene como resultado un mayor compromiso con la visión obtenida. El equipo deberá visualizar la visión de manera prominente en su entorno de trabajo. El ejercicio de definición de la visión puede tomar hasta unas tres horas.

## Creando el Product Backlog

La siguiente etapa consiste en llevar a cabo un workshop de redacción de historias de usuario. Es conveniente contar con stakeholders del negocio en esta etapa. Sin lugar a dudas el equipo de Scrum completo debe participar. Demás está decir que el ScrumMaster deberá facilitar esta reunión

Escribir buenas historias de usuario es simplemente una cuestión de práctica.

Mike Cohn [Cohn 2004] ha escrito una excelente guía sobre cómo redactar buenas historias de usuario. ¡Recomiendo a todos los Product Owner tener siempre una copia del libro a mano!

Bill Wake sugiere utilizar el acrónimo INVEST (invertir) para recordar los atributos que debe tener una buena historia de usuario [Wake 2003].

El backlog debe contener como mínimo la suficiente cantidad de ítems que permitan

Template: Como **rol**, quiero **funcionalidad** para qué **razón**.

Ejemplo: Como **programador**, quiero **café** para poder **estar despierto**.

planificar el primer sprint. Comúnmente el backlog contiene todos los ítems que conforman el próximo release del producto.

**Independiente**—idealmente puede desarrollarse en cualquier orden

**Negociable**—y negociada

**Valiosa**—para el cliente

**Estimable**—lo suficiente para ordenarla y planificar su entrega

**Pequeña (Small)**—y con descripciones breves

**Testeable**—Puedo escribir un test que pruebe su correcto funcionamiento

Como referencia, el backlog debería estar completo en un 80% (pero ni ordenado ni estimado) para el final del día uno del workshop. La primer parte del segundo día puede utilizarse para agregar el 20% restante y para resolver preguntas pendientes. La pausa entre ambos días es muy importante, dado que suele ayudar a que se disparen ideas novedosas.

## Ordenando el backlog

El backlog debe ordenarse por valor de negocio. Esto es mucho más fácil de decir que hacer. Mike Cohn [Cohn 2006] describe dos métodos para priorizar: el modelo Kano de satisfacción de clientes y el enfoque de peso relativo de Wiegers. Lo que me agrada de ambos es que no sólo consideran el beneficio esperado al disponer de la funcionalidad, sino que también toman en cuenta la penalidad por no tenerlo. Esto es particularmente útil cuando el backlog contiene ítems técnicos cuyo valor de negocio no resulta obvio de inmediato.

Como mínimo necesitaremos el juicio subjetivo del Product Owner para poder valorar una funcionalidad frente a las otras. Es preferible un enfoque cuantitativo utilizando una técnica similar al planning poker.

Sin importar cómo se lleve a cabo, ordenar el backlog es una de las principales responsabilidades del Product Owner.

## Estimando el Backlog

La estimación ha sido desde siempre la clave para llevar a cabo una efectiva planificación de proyecto. Gestores de proyecto como yo hemos utilizado semanas calibrando complejos modelos con datos históricos.

La cruda realidad es que las mejores de estas técnicas no entregan mejores resultados que técnicas mucho más simples como el planning poker o la estimación por afinidad. El planning poker funciona en parte dado que tiene una sólida base teórica, pero sobre todo porque quienes estiman son aquellos que en definitiva llevarán a cabo el trabajo. ¿Quién habría pensado eso?

El planning poker es rápido. Un equipo con práctica estimará en promedio un ítem cada 3 minutos. El planning poker es preciso. Las estimaciones obtenidas utilizando planning poker son tan buenas como las obtenidas usando métodos tradicionales. Y, sobre todo, el planning poker es divertido. Quita parte del aburrimiento asociado con este tema.

La estimación por afinidad es aún más rápida que el planning poker. Es genial para equipos que tienen un backlog completo para estimar y el tiempo es más importante que una gran precisión y distribución del conocimiento.

Sin embargo, todavía necesitamos comprender algunos conceptos clave sobre la estimación ágil:

- ▶ Estimamos los ítems de forma relativa entre ellos. ¿Por qué? Porque como humanos nos resulta más natural y por ende los resultados son más confiables. Resulta fácil acordar que 'esta historia tiene el doble de complejidad que esta' aunque no sepamos aún cuánto tiempo tomará implementarla.
- ▶ Estimamos los ítems usando unidades de complejidad en vez de unidades de tiempo. ¿Por qué? Porque nos permite separar el ritmo con el que trabaja un equipo de la complejidad del trabajo en si. Esto nos escuda frente a la necesidad de ajustar las estimaciones dependiendo de quién realiza el trabajo o cuando las habilidad y capacidad del equipo cambian con el correr del tiempo. Utilizamos 'puntos de historia' como unidades.
- ▶ Usamos una escala no lineal porque la diferencia entre '1' y '2' es obviamente más significativa que la que hay entre '20' y '21'. Prefiero siempre utilizar una escala basada en la serie de Fibonacci: 1, 2, 3, 5, 8, 13, 20, 40 and 100. Luego defino el rango que va del 1 al 8 (o incluso al 13) como los posibles tamaños de los ítems que un equipo puede desarrollar durante un sprint. Los números más grandes los reservo para historias más grandes ('épicas'), que deberán ser subdivididas en pequeñas historias antes de ingresar a un sprint.

- Relacionamos nuestras estimaciones de tamaño con el tiempo utilizando el concepto de *velocidad*, que no es sino el ritmo con el cual un equipo es capaz de entrega funcionalidades testeadas al Product Owner. Decimos que un equipo tiene una velocidad de '25' cuando al finalizar un sprint son sistemáticamente capaces de entregar historias 'hechas' cuyos tamaños suman, en promedio, 25 puntos.

☀ *No es una buena idea comparar las estimaciones entre equipos. Las mismas son inútiles, a menos que éstos se encuentren trabajando sobre el mismo backlog. Este es un problema que surge al escalar Scrum a proyectos más grandes, pero no es un tema a ser tratado en esta guía.*

## Reordenando el Backlog

Luego de estimar los ítems del backlog podría suceder que necesitamos reordenar algunos de ellos. Debemos tomar los siguientes factores en cuenta, además del valor de negocio:

- Tamaño: naturalmente elegiremos implementar una historia simple (pequeña) antes que una compleja (grande) si poseen un valor de negocio similar.
- Aprendizaje: podremos elegir implementar una historia antes si el equipo considera que desarrollarla ayudará a que éste comprenda mejor el dominio de negocios o una nueva tecnología.
- Riesgo: tal vezelijamos implementar una historia de manera temprana porque de esta forma estaremos mitigando un riesgo que hemos identificado. Ejemplos obvios son ítems que nos ayudarán a establecer límites de rendimiento y escalabilidad.

Nunca debemos olvidar que el Product Owner siempre tendrá la última palabra en lo que se refiere al ordenamiento del backlog.

## Planificación del release

Una vez que hemos ordenado y estimados el backlog, el siguiente paso es crear la primer versión del plan de release. Para ello necesitamos una estimación de la velocidad de nuestro equipo. Sin embargo, no hemos hecho ningún trabajo aún, por lo que utilizaremos una técnica muy simple denominada *planificación basada en el compromiso (commitment based planning)*.

Primero, por supuesto, debemos conocer el tamaño que tendrá el equipo durante el sprint—es importante saber si algún miembro estará ausente. Además debemos elegir una longitud de sprint—usualmente recomendando dos semanas para un equipo nuevo. Por último es necesario crear una definición de HECHO para el equipo—qué significa cuando el equipo dice que ha terminado con una historia.

A continuación el ScrumMaster toma el primer ítem del backlog y pregunta al equipo '¿Pueden completar este ítem durante el sprint?'. Esta dinámica continúa hasta que el equipo siente que ya no puede comprometerse a más ítems. Sumando los tamaños del backlog comprometido ya tenemos una estimación inicial de velocidad.

Este valor de velocidad es utilizado para dividir y ubicar los ítems restantes (al menos aquellos que han sido estimados) en los sprints subsiguientes. Esta lista de ítems asociados a sprints es la versión inicial de nuestro plan de release. ¿Es precisa la estimación? Tal vez no, pero es seguramente más creíble que cualquier otra a la que podría haber llegado un gestor de proyecto por si solo en una etapa tan temprana del proyecto. Una vez que comenzamos el trabajo, al finalizar un sprint tras otro, podremos graficar nuestra verdadera velocidad y utilizar este dato para predecir el ritmo futuro de entregas. Por ello decimos que plan de release es un ente vivo que se hace más y más confiable en la medida en la que progresamos.

☼ *No permitas que el workshop de producto se alargue innecesariamente. Es posible preparar suficiente backlog para los primeros sprints en sólo dos días para casi cualquier combinación de equipo y producto.*



# Ubicación física de los equipos

Alistair Cockburn [Cockburn 2007] define al desarrollo de software como un *juego cooperativo de invención y comunicación*. El manifiesto ágil dice que desarrolladores y representantes del negocio deben trabajar juntos de manera cotidiana y que las conversaciones cara a cara son la mejor manera de transferir información.

Sin duda el elemento más importante para que un equipo pueda colaborar de manera efectiva es que compartan el espacio físico. Un estudio sobre equipos de desarrollo de software con equipos que compartían el espacio físico [Teasley, Covi, Krishnan, & Olson, 2000] mostró que fueron el doble de productivos que otros equipos cuyos miembros se encontraban separados entre sí.

La definición que utilizo para ‘compartir el espacio físico’ es que los miembros del equipo estén sentados a como mucho 6 metros del miembro más lejano. Se deduce fácilmente el límite de personas que pueden entrar en ese espacio. En la práctica el número es entre 8 y 10, lo que felizmente concuerda con el tamaño máximo de un equipo Scrum.

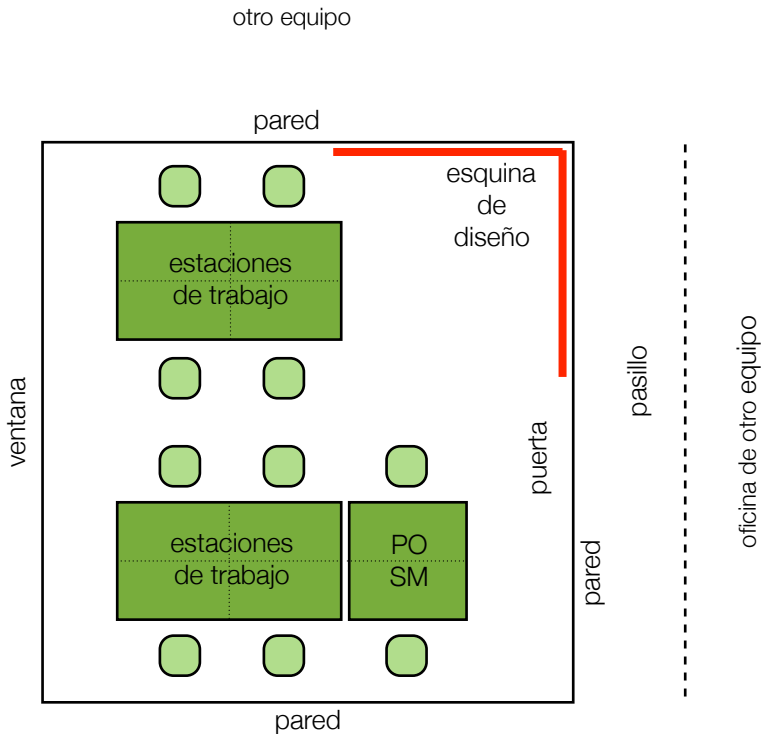
Aún más, cada miembro del equipo debe ser capaz de ver a cualquier otro miembro sin hacer más que un giro de su cabeza o su silla. Eso significa que no es posible que haya divisores entre escritorios—¡Adiós Dilbertville!

Sin ninguna buena razón más allá del hábito, las organizaciones —no los equipos— se resisten a cambiar el ambiente de trabajo para que éste facilite la colaboración. Esto ocurre a pesar de la evidencia que muestra que, en promedio, ubicar a un equipo en un espacio común para todos ellos no requiere más superficie que con la configuración actual.

Cualquier propuesta que considere una reestructuración del espacio físico será criticada por la gerencia como una amenaza a la flexibilidad. Esto sencillamente no es cierto—¡La flexibilidad es necesaria en la estructura organizacional, no en las oficinas!

Mi experiencia me ha mostrado que el 90% de los equipos Scrum en organizaciones donde éste jamás ha sido implementada deben luchar durante al menos un año para poder modificar su paupérrimo espacio de trabajo antes de lograr persuadir a sus gerentes de llevar a cabo los cambios necesarios. Eso sí: una vez que los cambios se llevaron a cabo, todos acuerdan que la mejora fue inmediata e incluso medible. ¿¿Por qué ocurre esto??

Es con la esperanza de que esta absurda resistencia disminuirá con el tiempo que ofrezco aquí un simple esquema de trabajo para un equipo. Escapa al objetivo de esta pequeña guía el razonamiento que existe detrás del diseño—¡Tendrás que contratarme si quieres eso!



☼ Algunos parámetros sobre el armado:

- ♦ Los escritorios deben ser rectangulares para facilitar el trabajo en pares
- ♦ 1.5-1.8 m x 0.8 m es buen tamaño de escritorio
- ♦ 1.2 m de espacio entre un escritorio y la pared
- ♦ Esquina de diseño 3 x 3 m
- ♦ Tamaño típico de la oficina 7 x 8 m = 50 to 60 m<sup>2</sup>
- ♦ El ScrumMaster puede ver a todo el equipo
- ♦ El Product Owner tiene un asiento flotante (no permanente)
- ♦ Las paredes de cartón yeso con ventanas internas crean una barrera contra el ruido y suficiente espacio para los irradiadores de información sin quitar luz

- ☼ *Tener una oficina propia del equipo reduce la necesidad de salas de reunión —los equipos pueden realizar la planificación, las reuniones diarias y las revisiones en su oficina*
- ☼ *Sabe de antemano que el diseñador o arquitecto que trabaja para tu empresa no te será de gran ayuda. Usualmente no han estudiado para diseñar espacios para trabajo en equipo.*
- ☼ *El costo de los cambios (paredes de cartón y muebles) se recupera usualmente en menos de un mes. ¿Suenan increíbles, no?*

## Métricas

Es razonable que cualquier gerente desee, dentro del rango de lo posible, poder medir los resultados de la transición de un equipo u organización hacia métodos ágiles.

Por suerte existen métricas que pueden implementarse de manera simple y rápida. Basado en mi propia experiencia y la de otros practicantes, he descrito un conjunto de métricas que tu equipo puede y debería implementar [Hundermark 2009]. En cuanto el equipo ha comprendido las reglas básicas del framework Scrum y tiene alguna idea sobre su velocidad—usualmente para el final del tercer sprint—probablemente es el momento de comenzar a medir. ¡De hecho puedes comenzar con la encuestas de satisfacción de equipo y clientes antes de empezar a trabajar con Scrum!

A modo de resumen, este es el conjunto de métricas que recomiendo se utilicen en al comienzo de una implementación:

- Encuestas de satisfacción de equipo y clientes
- Gráfico de velocidad
- Gráficos de burndown
- Cuántos tests están automatizados
- Deuda técnica
- Trabajo en proceso
- Tiempo promedio de finalización de historias

Una vez que ya has sido capaz de poner algún tipo de medida concreta al valor de negocios, agrega las siguientes:

- Costo por sprint o punto de historia
- Verdadero valor entregado
- Retorno de Inversión

# Coaching

## ¿Qué es lo que hace un coach?

*‘El coaching de Scrum se define como el trabajo realizado en una o más organizaciones durante el cual el coach actúa como mentor o facilitador de esos equipos para mejorar su entendimiento y aplicación de Scrum a fin de obtener su objetivos. El trabajo incluye mentoring a individuos y equipos, facilitación del proceso, desarrollo organizacional e interacción con todos los niveles de liderazgo de la organización. Tal vez incluya además mentoring en métodos de trabajo relacionados con el uso efectivo de Scrum, principios descritos en el Manifiesto Ágil, principios Lean y prácticas de Extreme Programming..’*

*Aplicación para convertirse en Certified Scrum Coach, Scrum Alliance*

Un coach guía a los individuos que conforman una organización de las siguientes formas:

- Consejo y consultoría para mejorar y acelerar el proceso de auto-descubrimiento
- Facilitación de la adopción, implementación y aprendizaje de Scrum
- Liderazgo ágil, basado en un estilo de liderazgo servil
- Desarrollo organizacional que mejore las habilidades, recursos y creatividad existentes en el cliente.

## ¿Por qué mi organización necesita coaching?

El desarrollo de software y otras clases de trabajo llevado a cabo por trabajadores del conocimiento depende en el conocimiento *tácito*, en contraste con el más tradicional conocimiento *formal* o *explícito*. El conocimiento *tácito* es difícil de transmitir de persona a persona y suele ser obtenido a través de la experiencia personal. Un ejemplo puede ser el aprender a andar en bicicleta.

El rol de ScrumMaster debe liderar la adopción y definición del proceso tanto para el equipo Scrum como para el resto de la organización. La capacitación, como por ejemplo mediante el curso de Certified ScrumMaster, es un medio insuficiente para poder adquirir el conocimiento *tácito* requerido para poder utilizar de manera efectiva las prácticas ágiles y el framework Scrum. El equipo y la organización necesitan las habilidades de un practicante experimentado, que los guíe a través del laberinto de desafíos que deberán enfrentar al realizar la transición a un modo de trabajo más ágil.

El comentario que más suelo escuchar cuando se les pregunta a gerentes en organizaciones que realizaron efectivamente una transición al trabajo ágil con Scrum qué habrían deseado tener es **‘¡Más coaching!’**

# References

1. Cockburn, Alistair (2007). *Agile Software Development: The Cooperative Game (2nd edition)*. Addison Wesley
2. Cohn, Mike (2004). *User Stories Applied*. Addison Wesley.
3. Cohn, Mike (2006). *Agile Estimating and Planning*. Prentice Hall.
4. Gloger, Boris (2008). *Heartbeat Retrospectives*. <http://borisgloger.com/2008/04/24/heart-beat-retrospectives-1-introduction/>.
5. Highsmith, Jim, et al (2001). *Manifesto for Agile Software Development*. <http://www.agilemanifesto.org/>.
6. Hundermark, Peter (2009). *Measuring for Results*. <http://www.scrumsense.com/coaching/measuring-for-results>.
7. Jeffries, Ron (2001). *Card, Conversation, Confirmation*. <http://www.xprogramming.com/xpmag/expCardConversationConfirmation>.
8. Katzenberg, Jon R. & Smith, Douglas K. (2002). *The Wisdom of Teams*. Collins.
9. Nonaka, Ikujiro & Takeuchi, Hirotaka (1986). *The New, New Product Development Game*. Harvard Business Review, Jan-Feb 1986.
10. Sliger, Michele & Broderick, Stacia (2008). *The Software Project Manager's Bridge to Agility*. Addison Wesley.
11. Schwaber, Ken (2007). *The Enterprise and Scrum*. Microsoft Press.
12. Schwaber, Ken (2009). *Scrum Guide*. <http://www.scrumalliance.com/resources>].
13. Teasley, Covi, Krishnan, & Olson (2000). *How Does Radical Collocation Help a Team Succeed?* Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (pp. 339 - 346). New York: ACM.
14. Wake, William (2003). *INVEST in Good Stories, and SMART Tasks*. <http://xp123.com/xplor/xp0308/index.shtml>
15. Yip, Jason (2006). *It's Not Just Standing Up: Patterns of Daily Stand-up Meetings*. <http://martinfowler.com/articles/itsNotJustStandingUp.html>.