El día a día de un proyecto ágil: Scrum. Desde la conceptualización a la entrega del producto

Marcos Bermejo
Marc Florit
Ramon G. Sedó

PID_00212576

UOC
Universitat Oberta de Catalunya

Índice

1. Introducción al Scrum

Scrum es un marco de trabajo para la definición de procesos que se caracteriza por ser ligero y fácil de entender.

Scrum no define un proceso concreto sino que proporciona las herramientas para que cada equipo adapte el marco de trabajo y encuentre el proceso más adecuado a sus circunstancias.

Esta característica hace que Scrum sea adecuado para entornos complejos en qué los resultados son *a priori* inciertos, en los cuales la tecnología a utilizar no sea trivial e incluso desconocida y donde es difícil predecir qué pasará en el futuro y donde, por lo tanto, sea necesaria una gran capacidad de adaptación, como es el caso del desarrollo de productos multimedia.

1.1. Fundamentos de Scrum

1.1.1. Gestión empírica

Scrum es un **método empírico** y, por lo tanto, se basa en gestionar el proceso de desarrollo a través de la experiencia (observación) y no en función de predicciones. La principal consecuencia que comporta es que las decisiones se toman basándose en hechos conocidos y no en hechos hipotéticos.

Para entender la diferencia entre el enfoque empírico y el enfoque predictivo, supongamos que nos encargan un proyecto que, a priori, estimamos que tendrá un año de duración.

Si gestionamos el proceso de forma predictiva, lo que haremos será una predicción de las tareas necesarias y el esfuerzo que tendremos que dedicar a cada tarea a partir de la cual estableceremos un calendario. Durante la ejecución del proyecto, compararemos la situación real con la predicción e intentaremos ajustar la realidad (por ejemplo, añadiendo más recursos) para que se adecue al calendario establecido.

Con Scrum, en cambio, estableceremos igualmente una predicción inicial pero durante la ejecución del proyecto nos iremos cuestionando, de manera regular, la corrección de la predicción, así la ajustaremos (por ejemplo, ajustando el calendario o modificando el conjunto de funcionalidades) para que se adecue a la realidad.

1.1.2. Ciclo de vida iterativo e incremental

Para poder adaptar el proceso a la realidad, Scrum utiliza el **ciclo de vida iterativo e incremental**. Es decir, organizamos el proyecto en iteraciones cortas (entre una y cuatro semanas preferentemente) para tener un ritmo estable para la revisión del proceso y la identificación de oportunidades de mejora.

Aun así, el producto se construye de manera incremental y así logra un doble objetivo:

1) desde etapas muy tempranas tenemos implementadas las funcionalidades más importantes del proyecto y, al mismo tiempo,

2) las diferentes iteraciones son comparables entre sí (cosa que no pasa, por ejemplo, en el ciclo de vida en cascada, donde cada etapa es totalmente diferente del resto y, por lo tanto, no podemos acumular tanto conocimiento de una etapa a la siguiente).

1.1.3. Transparencia

Todos los aspectos del proceso tienen que ser visibles a los responsables de su resultado. La transparencia requiere de la definición de un criterio común a todos los observadores para que todos interpreten lo que ven del mismo modo.

> Por ejemplo, es muy importante que todas las personas involucradas en el proyecto compartan la misma definición de hecho: que todo el mundo coincida a la hora de valorar si una tarea está acabada. Es habitual que varias personas tengan ideas diferentes sobre esta definición, puesto que, por ejemplo, quizás un desarrollador considera que una funcionalidad está hecha cuando ha acabado su parte, pero el responsable del producto considera que no lo está hasta que no se ha integrado con el resto del sistema.

1.1.4. Inspección y adaptación

Se da mucha importancia a la inspección frecuente del proceso y de sus resultados, así como a la adaptación del proceso cuando los resultados se desvían de lo que cabía esperar de forma que el resultado obtenido no sea aceptable.

En concreto, Scrum nos propone cuatro ocasiones para inspeccionar el proceso y formular propuestas de adaptación:

1) cuando se planifica la iteración (cada dos o cuatro semanas);

2) en la reunión diaria (donde se plantean los problemas que se van encontrando);

3) cuando se revisa y se muestra el trabajo hecho al final de cada iteración, y

4) en la retrospectiva que se hace al final de cada iteración (de hecho, la finalidad de la retrospectiva es, precisamente, reflexionar sobre cómo ha ido la iteración y decidir qué mejoras hay que aplicar).

1.2. Beneficios de Scrum

Uno de los **objetivos de Scrum** es obtener el máximo valor posible del esfuerzo dedicado al desarrollo de productos multimedia.

Mediante el **desarrollo incremental** (donde se desarrolla el producto multimedia basándose en añadir funcionalidades completas al producto existente) y la **priorización** en función del valor que la funcionalidad ofrece al cliente, lo que conseguimos es que el cliente obtenga muy pronto un producto con las principales funcionalidades implementadas de manera completa y, al mismo tiempo, ahorrarnos (si queremos) la implementación de las funcionalidades que aportan menos valor (con el consecuente ahorro de costes).

Aun así, Scrum facilita la **transparencia y la visibilidad de los problemas**. La reflexión constante sobre el proceso y la identificación de impedimentos lo antes posible (hasta el punto de que se habla a diario) nos ayuda a identificar los problemas antes de que se vuelvan demasiado grandes y, por lo tanto, nos ayuda a mantener un nivel más alto de productividad y calidad.

Por ejemplo, si un equipo tiene un problema porque los desarrolladores no pueden avanzar debido a que el representante del cliente no está disponible para solucionar dudas sobre los requisitos, es mucho mejor detectarlo al principio de la iteración que esperar a que se empiecen a incumplir las fechas previstas y se lleve a cabo un análisis del problema.

Otra ventaja de Scrum es que **minimiza la necesidad de gestión** al dar a los equipos de desarrollo la autonomía necesaria para organizarse a través de la comunicación y el consenso. La dirección de la empresa establece las prioridades y el equipo de desarrollo decide cómo satisfacerlas. Esto facilita el trabajo de los gestores (que solo tienen que decidir las prioridades, pero no tienen que asignar las tareas) y aumenta la satisfacción y el compromiso de los desarrolladores puesto que gozan de cierta autonomía.

Cuando empieza el ciclo de desarrollo en Scrum (la iteración), el responsable del producto decide qué funcionalidades se tienen que implementar en ese ciclo. Una vez decidida la lista de funcionalidades, los miembros del equipo de desarrollo pueden escoger, del conjunto de tareas necesarias para implementar todas estas funcionalidades, en cuál quieren trabajar en un momento dado.

Finalmente, como Scrum es un proceso muy orientado a la mejora continua y a la adaptación (tanto del proceso mismo como del producto desarrollado), es más fácil conseguir productos que satisfagan a los clientes y que sean innovadores.

2. Cómo funciona Scrum

1) El *product backlog*

Un proyecto de Scrum está impulsado por una visión de producto elaborada por el propietario del producto y se expresa en la **pila de producto**. El *product backlog* es una lista priorizada de lo que se requiere, por orden de valor para el cliente o negocio, con los elementos de mayor valor en la parte superior de la lista. El *product backlog* evoluciona durante la vida útil del proyecto y los elementos se añaden, se sacan o son repriorizados continuamente.

2) El *sprint*

Scrum estructura el desarrollo de productos en ciclos de trabajo llamado *sprints*, repeticiones de trabajo que tienen típicamente entre una y cuatro semanas de duración. Los *sprints* son de duración determinada y nunca se extienden más allá de la fecha fijada al comienzo, independientemente de si el trabajo planificado por el *sprint* se ha completado o no.

3) Planificación del *sprint*

Al inicio de cada *sprint*, se lleva a cabo la **reunión de planificación del *sprint***. El propietario del producto y el equipo Scrum (con la facilitación del *scrum master*) revisan la **pila de producto**, tratan los objetivos y el contexto de las historias de usuario y el equipo Scrum selecciona los elementos de la pila de producto que se comprometen a realizar para el final del *sprint*, partiendo de la parte superior de la pila de producto, donde se encuentran los elementos de priorización más alta.

Cada elemento seleccionado en la pila de producto se descompone después en una serie de tareas individuales. La lista de tareas se registra en un documento denominado ***sprint backlog***.

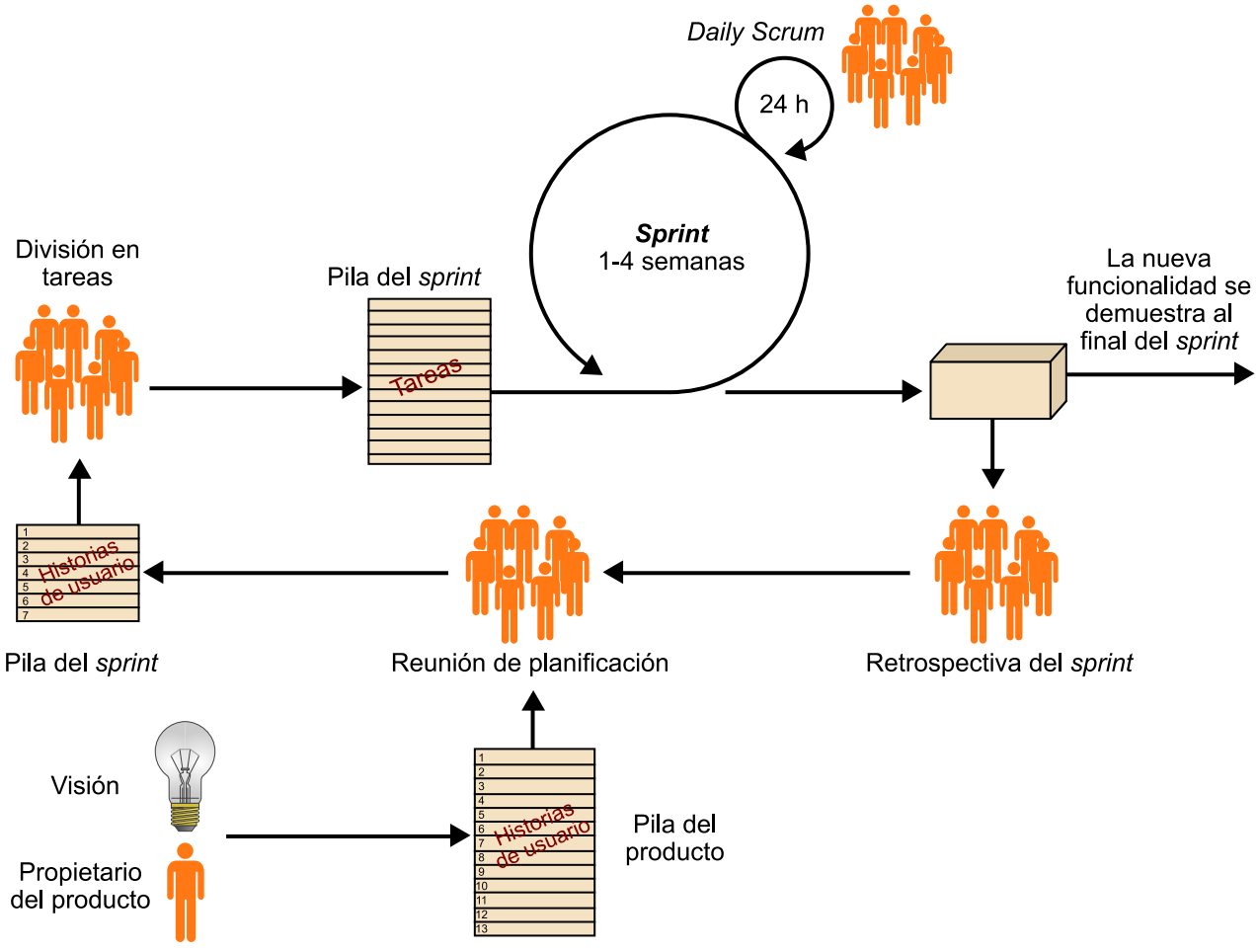
4) Reunión diaria de Scrum

Una vez se ha empezado el *sprint*, el equipo Scrum se dedica a otra de las principales prácticas de Scrum, como es la **reunión diaria de Scrum** o ***dailys-tand-up*** en su nombre original en inglés. Se trata de un breve encuentro (de máximo quince minutos) que tiene lugar cada día de trabajo a una hora determinada. Todos los miembros del equipo asisten a la misma. En esta reunión se presenta la información necesaria para inspeccionar el avance desde el día anterior. Esta información puede resultar en la replanificación y en debates sobre la funcionalidad después del *daily stand-up*.

5) Revisión del *sprint* y retrospectiva

Una vez finalizado el *sprint*, se produce la reunión de revisión del mismo, donde el equipo Scrum y las partes interesadas inspeccionan lo que se hizo durante el proceso y lo tratan. En esta reunión están presentes el propietario del producto, los miembros del equipo y el *scrum master*, además de los clientes, los implicados en el negocio, expertos, ejecutivos y cualquier otra persona interesada.

Tras la revisión del *sprint*, el equipo se reúne para la retrospectiva de este, que es una oportunidad para que el equipo trate lo que ha funcionado bien durante el *sprint* y lo que no funciona, y se ponga de acuerdo en los cambios que pretenden intentar durante el siguiente *sprint* con objeto de conseguir ser más productivos.



3. Las funciones en el Scrum

El equipo Scrum se compone de tres funciones claramente diferenciadas:

1) El propietario del producto. Toma las entradas de lo que el producto tiene que ser y las traduce en una visión de producto.

2) El equipo. Desarrolla el producto previsto por el propietario del producto.

3) El *Scrum master*. Tiene todo lo necesario para que el equipo Scrum alcance el éxito. Esto pasa por eliminar los obstáculos de organización, facilitar las reuniones y actuar como un guardián para que nadie obstaculice el trabajo del equipo.

3.1. El propietario del producto

El **propietario del producto** es el responsable de maximizar el retorno sobre la inversión (ROI) mediante la identificación de las características del producto, la traducción de estas en una lista de características de prioridades, decidir cuál tiene que estar en la parte superior de la lista para el siguiente *sprint* y continuamente repriorizar y refinar la lista.

El propietario del producto tiene la responsabilidad de las pérdidas y ganancias para el producto, suponiendo que sea un producto comercial. En el caso de una aplicación interna, el propietario del producto no es el responsable del retorno de la inversión en el sentido de un producto comercial (que generará ingresos), pero sigue siendo el responsable de maximizar el retorno de la inversión en el sentido de escoger –cada *sprint*– los elementos de mayor valor de negocio y de menor coste.

El propietario del producto no es un gerente de producto.

En algunos casos, el propietario del producto y el cliente son la misma persona, lo que es común para las aplicaciones internas. En otros casos, como cliente podría haber millones de personas con una variedad de necesidades, aquí la función de propietario del producto es similar a la de un responsable de producto o de marketing del producto en muchas organizaciones. Sin embargo, el propietario del producto es algo diferente al tradicional gerente de producto, ya que activa y a menudo interactúa con el equipo, personalmente, ofreciendo las prioridades y revisando los resultados de cada iteración de dos a cuatro semanas, en vez de delegar las decisiones de desarrollo a un gerente de

proyecto. Es importante señalar que en Scrum hay una y solo una persona que ejerce como propietario del producto (y tiene la autoridad final como tal). En los programas de varios equipos, este propietario del producto puede delegar el trabajo en los propietarios de productos que lo representan en los equipos de subordinados, pero todas las decisiones y la dirección provienen del más alto nivel, del propietario único del producto.

3.2. El equipo

El **equipo** construye el producto del cual el cliente va a hacer uso: la aplicación o página web, por ejemplo. El equipo en Scrum es multifuncional e incluye todos los conocimientos necesarios para entregar el producto **potencialmente entregable** en cada *sprint*. También es autoorganizado (el equipo se autogestiona), con un alto grado de autonomía.

Por lo tanto, no hay jefe de equipo o jefe de proyecto en Scrum. En cambio, los miembros del equipo deciden a qué se comprometen y cuál es la mejor manera de cumplir con este compromiso. El equipo es **autoorganizado**.

3.2.1. Equipo dedicado

El equipo en Scrum es de siete más menos dos personas, es decir, un mínimo de cinco personas un máximo de nueve. Para un producto multimedia el equipo puede incluir programadores, diseñadores de interfaz y los probadores. El equipo desarrolla el producto y ofrece ideas para el propietario sobre cómo obtener un gran producto. Es esencial que el equipo esté dedicado al cien por cien a trabajar para un único producto durante el *sprint*, ya que, si no es así, la multitarea provocada por la colaboración en múltiples productos o proyectos limitará agudamente su rendimiento. Equipos estables se asocian con una mayor productividad, de forma que cambiar los miembros del equipo también es algo que se debe evitar.

Los grupos de aplicaciones con muchas personas se organizan en varios equipos Scrum, cada uno enfocado a las diferentes características del producto, con una estrecha coordinación de sus esfuerzos. Desde un equipo se hace todo el trabajo (planificación, análisis, programación y prueba) para una función completa centrada en el cliente, por lo que los equipos de Scrum también se conocen como **equipos de características o funcionalidades**. En los programas técnicamente muy complejos y los productos, puede haber equipos organizados por nivel arquitectónico, como cuando se emplea la arquitectura de una familia de productos formada por más de uno. Sin embargo, la integración antes del final del *sprint* es más difícil cuando los equipos están tan estructurados.

3.3. El *scrum master*

El ***scrum master*** ayuda al grupo del producto a aprender y aplicar Scrum para conseguir valor de negocio. El *scrum master* es quien tiene en su poder ayudar al equipo a tener éxito.

El *scrum master* no es el mánager del equipo o un director de proyecto, sino que sirve al equipo, lo protege de la interferencia externa y educa y orienta al propietario del producto y al equipo en el uso de Scrum. El *scrum master* asegura que todos en el equipo (incluyendo el propietario del producto y los de gestión) entiendan y sigan las prácticas de Scrum. También ayuda a dirigir la organización a través del cambio, una actividad a menudo difícil pero necesaria para conseguir el éxito con el desarrollo ágil.

3.3.1. El compromiso es importante

Como Scrum hace visibles muchos impedimentos y amenazas a la eficacia del equipo y del propietario del producto, es importante tener un *scrum master* comprometido que trabaje con energía para ayudar a resolver estas cuestiones. Si no es así, será más difícil para el equipo o el propietario del producto tener éxito. Los equipos de Scrum deben tener una dedicación a tiempo completo de un *scrum master*, aunque un equipo más pequeño puede tener un miembro del equipo que ejerza este papel (que comporta una carga más ligera de trabajo habitual, cuando lo hacen). Los *scrum masters* pueden proceder de cualquier fondo o disciplina, como ingeniería, diseño, pruebas, gestión de producto, gestión de proyectos o de calidad.

El *scrum master* y el propietario del producto no pueden ser la misma persona, dado que a veces el *scrum master* puede ser llamado para hacer retroceder al propietario del producto (por ejemplo, si este trata de introducir nuevos productos en mitad de un *sprint*). A diferencia de un director de proyecto, el *scrum master* no le dice a la gente qué hacer o asigna tareas. Si el *scrum master* tenía previamente un cargo de gestión del equipo, tendrá que cambiar mucho su forma de pensar y estilo de interacción con el equipo para tener éxito con Scrum. En el supuesto de que un gestor haga la transición a la función de *scrum master*, lo mejor es usar a un equipo diferente del que trabajaba antes como gestor con objeto de evitar conflictos potenciales.

¿Qué pasa con los gerentes?

Tened en cuenta que no hay papel de jefe de proyecto en Scrum. A veces, un (ex) jefe de proyecto puede entrar en el papel de *scrum master*, pero esto tiene un historial de éxitos y fracasos. Existe una diferencia fundamental entre las dos funciones, tanto en las responsabilidades del día a día como en la mentalidad necesaria para alcanzar el éxito.

Además de los tres papeles principales, hay otros colaboradores para el éxito del producto, incluidos los directivos. Mientras que sus papeles cambian, siguen siendo valiosos, por ejemplo:

- para crear un modelo de negocio que funciona y proporcionar los recursos que el equipo necesita,
- para apoyar al equipo, respetando las reglas y el espíritu de Scrum,
- para ayudar a eliminar los obstáculos que el equipo identifica,
- para poner sus conocimientos y experiencia a disposición del equipo, y
- para desafiar al equipo a ir más allá de la mediocridad.

En Scrum, estos individuos reemplazan el tiempo que antes se dedicaba a desempeñar el papel de niñera (asignación de tareas, obtener informes de estado y otras formas de microgestión) con el tiempo como gurú y siervo del equipo (*mentoring*, *coaching*, ayudar a eliminar obstáculos, ayudar a resolver problemas, hacer aportaciones creativas y guiar el desarrollo de las competencias de los miembros del equipo). En este cambio, los gerentes pueden necesitar cambiar su estilo de gestión, por ejemplo, mediante el interrogatorio socrático para ayudar al equipo a descubrir la solución de un problema, en vez de decidir una solución e implantarla en el equipo.

4. Primeros pasos

Iniciar un proyecto Scrum no es difícil, siempre y cuando uno dé un paso a la vez y se asegure de que todo el mundo se sienta incluido.

4.1. El product backlog

Un proyecto de Scrum está impulsado por una visión de producto elaborada por el propietario del producto y se expresa en la **pila de producto**. El *product backlog* es una lista priorizada de lo que se requiere, por orden de valor para el cliente o negocio, con los elementos de mayor valor en la parte superior de la lista. El *product backlog* evoluciona durante la vida útil del proyecto y los elementos se añaden, se sacan o son repriorizados continuamente.

Este registro no solo existe y evoluciona a lo largo de la vida útil del producto, sino que es la hoja de ruta del producto.

En cualquier momento, la **pila de producto** es el punto de vista único y definitivo de todo lo que se puede hacer por el equipo, en orden de prioridad. Solo existe una única pila de producto.

La pila de producto incluye una variedad de elementos, sobre todo las nuevas características del producto (“permitir a los usuarios colocar el libro en la cesta de la compra”), pero también los objetivos de mejora de ingeniería (“revisar el módulo de procesamiento de transacciones para que sea escalable”), de exploración o trabajo de investigación (“investigar soluciones para acelerar la validación de la tarjeta de crédito""), los requisitos de rendimiento y seguridad y, posiblemente, los defectos conocidos (”diagnosticar y corregir los errores de procesamiento de pedidos de secuencia”), si solo hay unos cuantos problemas (un sistema con muchos defectos en general tiene un sistema de seguimiento de defectos por separado).

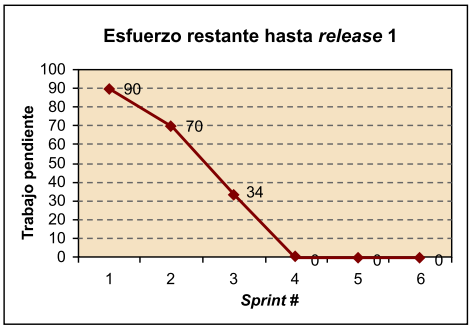
Por lo general, se articulan los requisitos en términos de **historias de usuario**, que son descripciones concisas y claras de la funcionalidad en términos de su valor para el usuario final del producto.

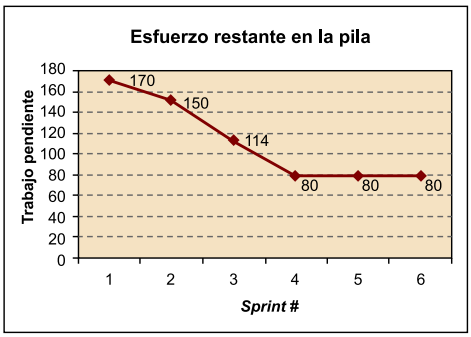
El subconjunto de la pila de producto que se destina para la versión actual se conoce como la **pila de la versión**, o ***release backlog*** en el inglés original, y en general esta parte es el objetivo principal del propietario del producto.

Aplicación de climatología para dispositivos móviles

ID	Descripción	Sprint #	1	2	3	4	5	6
	Esfuerzo requerido para la primera versión al principio del *sprint*		90	70	34	0	0	0
1	Montar el sistema de integración continua		5	0	0	0	0	0
2	Crear el esqueleto de la aplicación		5	0	0	0	0	0
3	Mostrar la temperatura actual de la forma más simple		13	0	0	0	0	0
4	Montar el servidor para obtener datos de temperatura		3	0	0	0	0	0
5	Implementar soporte del protocolo WeatherML en el servidor		13	0	0	0	0	0
Sprint 1	*Obtener datos de prueba desde el servidor al cliente*							
6	Soporte de gráficos al lado cliente		20	0	0	0	0	0
16	Dibujar con la librería de gráficos un icono y un texto de temperatura al cliente		-	13	0	0	0	0
17	Dibujar la pantalla de temperatura real		-	8	0	0	0	0
7	Implementar el soporte para múltiples días		8	8	0	0	0	0
8	Implementar el soporte para lluvia/nieve		2	2	0	0	0	0
9	Soportar el cambio de ciudad		-	5	0	0	0	0
Sprint 2	*Versión mínima funcionando*							
10	Capturar la temperatura de un día del proveedor de climatología		?	13	13	0	0	0
11	Obtener datos de lluvia, nieve, etc. del proveedor		8	8	8	0	0	0
12	Obtener datos por mes de un día		5	5	5	0	0	0
13	Refresco automático al cliente		8	8	8	0	0	0
Sprint 3	*Versión con datos de temperaturas reales*							
Release 1	**Versión para vender en el *marketplace***							
14	Inyectar *ads* simulados desde el servidor		20	20	20	20	20	20
15	Obtener y mostrar *ads* reales		20	20	20	20	20	20
16	Cambiar de ciudad automáticamente a partir de los datos del móvil		40	40	40	40	40	40
Sprint 4	*Soporte para ads publicitarios (monetización)*							
Release 2	**Versión con publicidad**							
	Esfuerzo total en la pila del producto		170	150	114	80	80	80

Estado del *backlog* después del *sprint* 3 **Versión** 1





Ejemplo de una pila de producto

La pila de producto la actualiza constantemente el propietario del mismo para reflejar los cambios en las necesidades del cliente, las nuevas ideas o puntos de vista, los movimientos de la competencia, obstáculos técnicos que aparecen y así sucesivamente. El equipo ofrece al propietario del producto las estimaciones del esfuerzo requerido para cada elemento de la pila de producto. Además, el propietario del producto es responsable de asignar una estimación del valor de negocio a cada elemento individual. Esto suele ser una práctica desconocida para un propietario del producto novel.

Como tal, es algo que un *scrum master* puede ayudar al propietario del mismo a aprender a hacer. Con estas dos estimaciones (**esfuerzo** y **valor**) y quizás con las estimaciones de riesgo adicionales, el propietario del producto da prioridad a la cartera de pedidos para maximizar el retorno de la inversión (eligiendo elementos de alto valor con poco esfuerzo) o, en segundo lugar, para reducir algún riesgo importante. Como veremos, estas estimaciones de esfuerzo y valor se pueden actualizar en cada *sprint* dado que la gente aprende y, en consecuencia, se produce una repriorización continua de la actividad y la pila de producto está en evolución constante.

Scrum no obliga a ejecutar las estimaciones de la pila de producto de una forma determinada, pero es común el uso de estimaciones relativas, calificadas como **puntos de historia** en lugar de unidades absolutas de esfuerzo, como horas-persona. Con el tiempo, y un buen seguimiento de los resultados del equipo, se pueden extraer conclusiones en función de la cantidad de puntos de historia relativos que un equipo de producción multimedia es capaz de implementar para cada *sprint* (por ejemplo, una media de 26 puntos de historia por *sprint*). Con esta información se puede proyectar una fecha de lanzamiento para completar todas las características o cuántas características se comple-

tarán con probabilidad antes de una fecha. Los puntos completados por *sprint* reciben el nombre de **velocidad del equipo**. Un plan de lanzamiento realista siempre se basa en la velocidad del equipo.

Los elementos de la pila de producto pueden variar significativamente en tamaño y esfuerzo. Los más grandes se rompen en elementos más pequeños, normalmente durante reuniones de refinamiento de la pila del producto o bien en la reunión de planificación del *sprint*.

¿La cantidad de detalles?

Uno de los mitos sobre Scrum es que impide escribir las especificaciones detalladas. En realidad, son el propietario del producto y el equipo quienes deben decidir cuánto detalle se requiere y esto puede variar de un elemento a otro, dependiendo de la visión del equipo y de otros factores. Se tiene que procurar reflejar lo que es importante en la menor cantidad de espacio necesario, en otras palabras, no describir todos los detalles posibles de un elemento, simplemente dejar claro lo que hace falta para que se entienda. Los elementos de baja prioridad, que son propensos a ser implementados en una etapa posterior, tendrán en general menos detalles respecto a los requisitos de alta prioridad, que pronto se llevarán a cabo y por lo tanto tienden a tener más detalles.

4.1.1. La visión del producto: pensando en el usuario. Las historias de usuario

Una **historia de usuario** es una redacción breve de una funcionalidad que por sí misma da valor al usuario.

Se compone de un título, una descripción breve sobre qué tiene que hacer la funcionalidad y unas condiciones de aceptación. Opcionalmente, se pueden añadir anotaciones y acotaciones a la historia.

Es muy aconsejable usar la siguiente estructura:

Como [papel de usuario] quiero [funcionalidad] para [objetivo]

Cómo tiene que ser una historia de usuario

Una buena historia de usuario sigue las siglas **INVEST**, es decir, es:

- **I**ndependiente,
- **N**egociable,
- **V**aliosa para usuarios y clientes,
- **E**stimable (se puede medir),
- ***S****mall* (suficientemente pequeña para poder trabajar con ella), y
- **T**esteable.

1) Independiente

Las dependencias entre historias a veces derivan en problemas en la planificación y priorización. Si una historia muy poco prioritaria está vinculada con una que lo es mucho, la priorización será complicada.

Cuando dos historias están vinculadas, hacen muy difícil la estimación, puesto que surgen dudas tales como "Si desarrolláramos primero la historia A, la historia B sería más fácil de hacer y supondría menos esfuerzo, pero la historia A no es prioritaria". En la gestión ágil de proyectos siempre se hace antes lo más prioritario, así que entraríamos en un conflicto.

Cuando se presentan estos tipos de dependencias entre historias hay dos caminos para solucionarlo:

1) Combinar las historias dependientes en una más grande, pero independiente del resto.

2) Encontrar un camino diferente en la definición de estas historias.

2) Negociable

Las historias de usuario son descripciones cortas de funcionalidades, cuyos detalles deben ser negociados en conversaciones posteriores entre el cliente y el equipo de producción. Tal como hemos dicho al comienzo, una historia se compone de una breve descripción acompañada opcionalmente de anotaciones o acotaciones de la historia. Estas acotaciones se tendrán que ir negociando a lo largo del proyecto.

Acotar la funcionalidad

Las anotaciones de una historia deben acotar su funcionalidad. Es decir, si tenemos una funcionalidad de "Como usuario administrador quiero poder generar un documento Word con el listado de nombres de los productos del sistema", una anotación como "la funcionalidad también debe sacar el listado en PDF" no acota ni detalla la funcionalidad, la aumenta. Una anotación posible sería "el documento Word tiene que estar en el formato oficial de la empresa".

En muchos proyectos, se confunde dar más detalles con dar más precisión. No hay que entrar en la precisión excesiva, si la historia se puede estimar y priorizar se pueden dejar los detalles precisos para conversaciones posteriores.

Podemos ver una historia como un recordatorio de tener una conversación entre el equipo de producción y el cliente. Donde la descripción indica sobre qué funcionalidad tenemos que hablar y las anotaciones serían los detalles a los que se han llegado en estas conversaciones.

3) Valiosa

La mejor manera de asegurarse de que una historia es valiosa para el cliente es dejarlos escribir a ellos las historias. Es habitual que los clientes se muestren incómodos con esta propuesta, ya que están acostumbrados a que todo lo que

escriben es un contrato y se puede usar en su contra ("Como no escribiste que querías la validación del DNI..."). En la gestión ágil de proyectos no se busca esta manera de "pasar la pelota". Es muy extraño que detalles importantes no se comenten en las conversaciones entre cliente y equipo de producción.

Si logramos tener a un cliente que confía en nosotros y se involucra, tenemos mucho ganado en la definición de historias.

4) Estimable

Hay tres razones comunes por las que una historia de usuario no se puede estimar:

1) poco dominio por parte del equipo de producción del lenguaje del cliente;

2) poco dominio técnico por parte del equipo de producción sobre la tecnología que se va a usar; y

3) la historia es demasiado grande.

Si el equipo de producción no entiende la historia tal cual se ha escrito, la solución es celebrar reuniones entre el equipo de producción y el cliente. En la gestión ágil de proyectos no está permitido ahorrarnos conversaciones ni suponer o interpretar conceptos. Si el equipo de producción no está seguro de lo que tiene que hacer, se tiene que hablar para entenderlo.

Si el problema es tecnológico, la solución es reservar uno o dos programadores para un *spike*.

Un ***spike*** es un breve experimento para aprender cómo funciona una tecnología y qué implicaría en nuestro proyecto usarla.

Durante el *spike*, se pide a los programadores que experimenten y aprendan todo aquello necesario solo para estimar la funcionalidad, no que la desarrollen.

Nota

Es recomendable que un *spike* sea limitado por un *time-box*, o porción de tiempo, donde, si se sobrepasa y no se ha conseguido nada, podemos pensar en descartar la tecnología seleccionada.

Si el problema es que la historia de usuario es demasiado grande para estimarla con cierta precisión, la única salida es dividirla, más adelante aprenderemos cómo dividir las historias de usuario.

Historia de usuario pequeña

Puede ser que una historia de usuario como la siguiente:

"Como usuario visitante de la web, tengo que poder registrarme para entrar en el sistema".

sea suficientemente pequeña, ya que la tecnología empleada da herramientas prehechas para afrontar esta tarea.

Aunque una historia sea demasiado grande, a veces son útiles como épicas. Las **historias épicas** son recordatorios de las grandes partes del sistema que tienen que ser desarrolladas.

5) *Small*

El tamaño ideal de las historias varía en función del equipo, las capacidades y la tecnología empleada.

Si la historia se tiene que dividir, se deben tener en cuenta las consignas siguientes:

a) Dividir historias por su composición

Tenemos que pensar en el flujo de acciones que el usuario va a ejecutar en toda esta funcionalidad, por ejemplo "Como usuario administrador quiero poder administrar los productos del sistema" se podría dividir en "ver una lista de productos", "crear un producto", "eliminar un producto", "editar un producto". Tener en mente las siglas **CRUD** (***create***, ***read***, ***update***, ***delete***) nos puede ayudar en esta tarea.

b) Dividir historias complejas

Esto es menos obvio. Son aquellas historias que son muy complejas técnicamente o que no se han hecho nunca y no sabemos cuánto nos pueden costar, la división recomendada en este caso es **investigar-realizar-refactorizar**. Podemos crear una historia de usuario que sea "Investigar cómo hacer un registro de usuario en la nueva tecnología" y se asignaría un *time-box* limitado, el otro "realizar la funcionalidad" y otra de "refactorizar[1] la funcionalidad".

(1)Refactorizar quiere decir revisar el código programado para poderlo hacer más rápido, mantenible a largo plazo y comprensible para otros programadores, entre otros.

Historia de usuario de investigar

Es importante que si se decide definir una historia de usuario de investigar tenga un fin. Puede ser un documento con conclusiones, una versión tosca de la funcionalidad pero con los detalles primordiales que se han investigado. Si no indicamos una finalidad esta historia se puede convertir en una pérdida de tiempo.

Es posible que nos encontremos con historias demasiado pequeñas, como arreglar algún *bug* o corregir literales. Detectaremos una historia demasiado pequeña enseguida, ya que el equipo de producción dirá que es más rápido hacerla que escribirla.

Hay algunas historias que hacen referencia a todo el proyecto, por ejemplo "La web tiene que ser rápida". Esto no es una historia, es una *constraint* y las ***constraints*** son condiciones que deben cumplir todas las historias que se desarrollen y en general todo el proyecto. "Ninguna página de la web tiene que tardar más de tres segundos en cargarse" sería un buen ejemplo de *constraint*.

Las *constraints* están presentes en todo el proyecto y las deben tener siempre en cuenta el equipo de producción antes de finalizar una historia de usuario.

6) Testeable

¿Cómo sabemos si una historia está acabada si no se puede probar? Una historia debe tener ciertas condiciones de validación que justifiquen que la historia está acabada. Una historia se tiene que poder probar que funciona.

Los beneficios de las historias de usuario

Las historias de usuario comportan, entre otros, estos beneficios principales:

1) Promueven la comunicación verbal

Los seres humanos tienen una gran tradición oral. Cuentos y leyendas han sido transmitidos oralmente entre generaciones, pero en algún momento, en torno al año 1970, se empezó a sentir la necesidad de registrarlo todo por escrito. Hemos cambiado el compartir conocimiento por compartir documentos. Parece que dejar algo por escrito evita desacuerdos y malentendidos, pero no es así.

Comunicación verbal

Si en un restaurante vemos que en el primer plato sirven bacalao o lomo con guarnición, ¿significa que tanto la carne como el pescado vienen acompañados de guarnición? Si ya nos cuesta leer el menú en el restaurante, imaginad la definición de una funcionalidad de un proyecto de producción multimedia.

La palabra *debería* no se tiene que usar nunca. ¿Qué quiere decir "el sistema debería avisar si el usuario ha introducido un DNI incorrecto"? ¿Significa que el sistema puede no avisar?

Cuando cliente y proveedor usan la comunicación oral, se produce un *feedback* constante, del que se extrae mucha más información que por escrito. Promueve el aprendizaje y el entendimiento mutuo.

2) Son comprensibles por todo el mundo

Como una historia de usuario está escrita en lenguaje natural, es comprensible tanto por usuarios como por el equipo de producción y por todo el que participe en el proyecto. Promueve la participación y la opinión de todo tipo de perfiles, tanto diseñadores como programadores o comerciales, entre otros. No se restringe la opinión a ninguna persona que lea la funcionalidad.

3) Tienen el tamaño justo para planificar

Ni demasiado grandes ni demasiado pequeñas, una historia de usuario tiene el tamaño justo para que tanto cliente como desarrolladores se sientan cómodos con ella, tanto para definirla como para desarrollarla, probarla o demostrarla al cliente en una demo.

4) Funcionan muy bien en el proceso de producción iterativo

En la gestión ágil de proyectos, donde se trabaja de manera empírica, iterativa e incremental, las historias de usuario son la mejor unidad mínima de funcionalidad para desarrollar. Iteración tras iteración se pueden usar para ir construyendo el proyecto multimedia final.

4.2. Planificación del sprint

Al principio de cada *sprint* se lleva a cabo la **reunión de planificación del *sprint***. En la primera parte de la planificación, el propietario del producto y el equipo (con la facilitación del *scrum master*) revisan los elementos de alta prioridad en la pila del producto que el propietario del producto está interesado en que se lleven a cabo en ese *sprint*. Se tratan los objetivos y el contexto de estos elementos de alta prioridad de la pila de producto y se da al equipo información sobre cuál es la idea del propietario del producto sobre cada elemento.

El propietario del producto y el equipo también fijan la definición de hecho (*definition of done* o DoD según el término original en inglés).

Esta **definición de hecho** es un breve documento donde se indican todas las condiciones generales que las funcionalidades tienen que cumplir para ser dadas como hechas.

Esta definición de hecho garantiza la transparencia y la calidad apta para la finalidad del producto y la organización.

Definición de hecho

Por ejemplo, "la funcionalidad tiene que estar traducida a tres idiomas" o bien "la funcionalidad cumple los mínimos de seguridad especificados por el Departamento de Sistemas" podrían ser ejemplos de condiciones del documento de definición de hecho.

La **primera parte** de la reunión de planificación del *sprint* se centra en la comprensión de lo que quiere el propietario del producto. De acuerdo con las reglas de Scrum, al final de la primera parte, el propietario del producto (siempre ocupado) puede dejar la reunión a pesar de que tiene que estar disponible (por ejemplo, por teléfono) durante la segunda parte de la reunión. Sin embargo, es bienvenido y se recomienda que también asista a la segunda parte.

La **segunda parte** de la reunión de planificación se centra en la planificación de tareas detalladas para la forma de aplicar los elementos que el equipo ha decidido asumir. El equipo selecciona los elementos de la pila de producto que se comprometen a ejecutar para el final del *sprint*, empezando por la parte

superior de la pila de producto (en otras palabras, a partir de los elementos que son la máxima prioridad para el propietario del producto) y bajando la lista en orden.

Mientras que el propietario del producto no tiene control sobre la cantidad que el equipo se compromete a poder hacer, sabe que los elementos a los que el equipo se ha comprometido se han extraído de la parte superior de la pila de producto,o sea, los elementos que ha calificado como los más importantes.

Planificación hecha por el equipo

Una práctica clave en Scrum es que el equipo decide la cantidad de trabajo que se compromete a completar, en lugar de que el propietario del producto se la haya asignado. Esto se hace para conseguir un compromiso más fiable, ya que el equipo está haciendo lo que cree que puede entregar en función de su propio análisis y planificación, en lugar de tener que hacer lo que ha decidido otro.

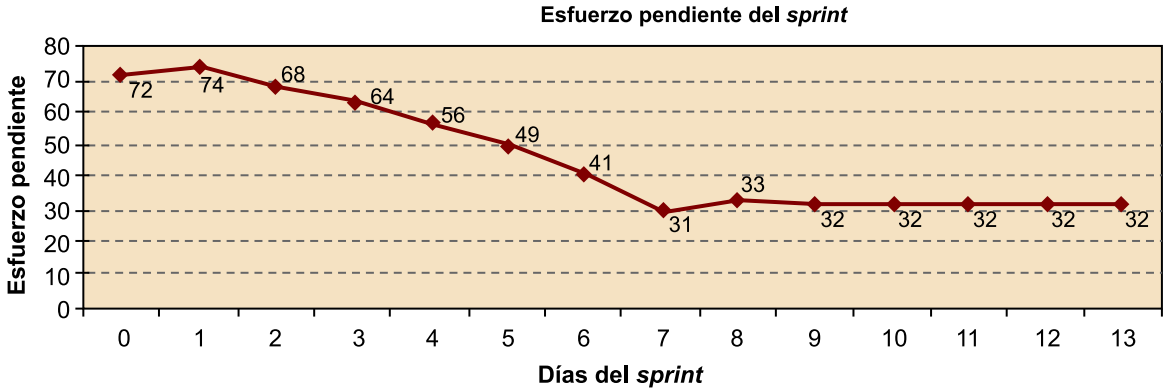
El equipo tiene la autoridad para seleccionar también los temas de más abajo de la lista en consulta con el propietario del producto. Esto sucede cuando el equipo y el propietario del producto se dan cuenta de que un producto de prioridad más baja se adapta fácilmente y de forma más apropiada a las cuestiones de alta prioridad que se harán en este *sprint*.

La reunión de planificación del *sprint* se limita a cuatro horas para un *sprint* de cuatro semanas y a dos horas para un *sprint* de dos semanas. Por eso, el equipo tiene que ayudar al propietario del producto mediante la estimación del tamaño de las historias antes de la reunión de planificación de *sprint*[2]. Un equipo basa sus compromisos en sus velocidades en el pasado. Si un equipo es nuevo, en la tecnología o el campo, no se pueden tener velocidades fiables hasta que no hayan trabajado juntos durante tres o cuatro *sprints*.

[(2)]El equipo está haciendo un serio compromiso para completar el trabajo y este compromiso requiere un análisis cuidadoso para tener éxito.

Una vez determinada la capacidad de los equipos disponibles, el equipo empieza con el primer elemento de la pila de producto –en otras palabras, lo que el propietario del producto ha fijado con más alta prioridad– y se descompone en tareas individuales, que se registran en la pila del *sprint* o *sprintbacklog*. Tal como se ha mencionado, el propietario del producto tiene que estar disponible durante la segunda parte (por ejemplo, a través del teléfono) de forma que las aclaraciones y decisiones en cuanto a los enfoques alternativos sean posibles. Al final de la reunión, el equipo ha elaborado una lista de tareas con las estimaciones (en general en horas o fracciones de un día). La lista es un punto de partida, pero pueden salir más tareas que se desconocían a medida que el equipo avanza durante el *sprint*.

ID	Historia/tarea	días del sprint/esfuerzo restante												
		0	1	2	3	4	5	6	7	8	9	10	11	12	13
		72	74	68	64	56	49	41	31	33	32	32	32	32	32
10	**Capturar la temperatura de un día del proveedor de climatología**														
	Abrir la conexión desde nuestro servidor y autenticamos en el sistema proveedor	4	16	12	8	3	3	3	3	3	3	3	3	3	3
	Leer los datos de temperatura del proveedor	8	7	7	7	4									
	Parchear la temperatura tomada por el proveedor en nuestro formato	6	6	4	4	4	1	1	1	1	1	1	1	1	1
	Enviar los datos de temperatura al cliente	16	16	16	16	16	16	8	2						
11	**Obtener datos de lluvia, nieve, etc. del proveedor**														
	Parchear datos sobre nieve/lluvia del proveedor	4	4	4	4	4	4	4							
	Enviar los datos de nieve/lluvia al cliente	4	4	4	4	4	4	4	4	4					
	Rediseñar la pantalla cliente para mostrar los nuevos datos										3	3	3	3	3
	Refactorizar el código en el servidor									4	4	4	4	4	4
12	**Obtener datos por mes de un día**														
	Parchear los datos de climatología en paquetes de más de un día	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	Enviar los datos de más de un día al cliente	3	3	3	3	3	3	3	3	3	3	3	3	3	3
13	**Refresco automático al cliente**														
	Hacer que el cliente llame automáticamente al servidor cada dos horas	6	6	6	6	6	6	6	6	6	6	6	6	6	6
	Hacer que el servidor actualice el cliente tras la petición	2	2	2	2	2	2	2	2	2	2	2	2	2	2



Actualizado a día 9

Ejemplo de una pila del *sprint*

Scrum anima a que los equipos de trabajo sean **polivalentes**, en lugar de solo trabajar en lo que dice su etiqueta, es decir, un programador solo programa o un probador solo hace pruebas. En otras palabras, los miembros del equipo siguen el principio de ir donde haya trabajo y ayudar como sea posible. Si hay muchas tareas de prueba, todos los miembros del equipo pueden ayudar a probar. Esto no implica que todo el mundo sea generalista, sin duda, algunas personas son muy habilidosas en la prueba (y así sucesivamente), sino que el equipo ha demostrado ser un enfoque valioso para el **intercambio de conocimientos**.

Dicho todo esto, hay momentos en los que solo un miembro del equipo puede desempeñar una determinada tarea, puesto que perdería demasiado tiempo que lo hiciera otro miembro o no es posible que otros aprendan a hacerlo en un tiempo razonable (quizás es la única persona con alguna habilidad artística para dibujar). En este caso, es posible pero hay que tratar de evitarlo al máximo dado que dependemos de un único miembro del equipo para desempeñar un determinado trabajo (puede ser necesario preguntarse si las tareas planificadas totales de dibujo que este miembro del equipo debe desempeñar son factibles dentro del corto *sprint*).

Uno de los pilares de Scrum es que, una vez el equipo acepta su compromiso, las adiciones o cambios tienen que ser aplazados hasta el próximo *sprint*. Eso significa que, si en mitad del *sprint* el propietario del producto decide que hay un elemento nuevo que le gustaría que el equipo trabajara, no puede introducir el cambio hasta el inicio del siguiente *sprint*. Si una circunstancia externa parece que cambia de forma significativa las prioridades, y significa que el equipo estaría perdiendo el tiempo si continuara trabajando, el propietario del producto o el equipo pueden acabar el *sprint*. El equipo se detiene y se inicia una nueva reunión de planificación del *sprint*. El coste de hacerlo (lo que se llama **romper un *sprint***) es en general grande y sirve como un desincentivo para que el propietario del producto o el equipo recurran a esta decisión drástica.

Al seguir estas reglas de Scrum, el propietario del producto gana dos cosas:

1) En primer lugar, tiene la confianza de saber que el equipo ha alcanzado un compromiso para completar un conjunto realista y claro de las tareas que ha elegido. Con el tiempo, un equipo puede llegar a ser muy habilidoso en la elección y la entrega de un compromiso realista.

2) En segundo lugar, el propietario del producto llega a introducir los cambios que le gustan en la pila de producto antes del comienzo del siguiente *sprint*. En ese momento, las adiciones, supresiones, modificaciones y reordenaciones de la preferencia de todo esto son posibles y aceptables. Aunque el propietario del producto no tiene privilegios para modificar los elementos que han sido seleccionados en el *sprint* actual, sí puede introducir cambios en aquellas funcionalidades que aún no han entrado en el *sprint*. Es decir, el propietario del producto tiene que estar siempre a un *sprint* de distancia por delante del equipo de desarrollo para introducir cualquier cambio. De este modo, se acaba con el estigma en torno al cambio, cambio de dirección, cambio de requisitos o simplemente cambiar de opinión.

4.3. El *daily stand-up*

Una vez el *sprint* ha empezado, el equipo se involucra en otra de las claves de las prácticas de Scrum, el ***daily stand-up***. Se trata de una breve reunión (de quince minutos más o menos) que tiene lugar cada día de trabajo en el mismo sitio y a la misma hora. Todos los formantes del equipo asisten a ella. Recibe su nombre por el hecho de que, por ser breve, se recomienda que todo el mundo permanezca de pie. Es la oportunidad del equipo para informar a los demás de su avance y obstáculos. En el ***daily scrum***, uno por uno, cada miembro del equipo informa de tres (y solo tres) cosas a los demás miembros del equipo:

1) qué han sido capaces de lograr tener hecho desde la última reunión;

2) lo que están planificando tener para acabar antes de la próxima reunión e informar de si se produce un cambio en la estimación inicial para completar la tarea; y

3) todos los obstáculos que se encuentran en el camino.

Tened en cuenta que el ***daily scrum***, no es una reunión para informar a un gestor del avance, es un tiempo en el que un equipo autoorganizado comparte con los demás lo que está pasando para ayudar a coordinar el trabajo entre todos para optimizar la probabilidad de cumplir los compromisos.

Alguien toma nota de los impedimentos y el *scrum master* es responsable de ayudar a los miembros del equipo a resolverlos. No hay debate en el ***daily scrum***, solo la presentación de informes en forma de respuestas a las tres preguntas. Si se requiere tratar un punto, el debate se lleva a cabo inmediatamente después del ***daily scrum***, en una reunión de seguimiento, habitualmente solo con las personas a las que les afecta de forma directa en su trabajo. Esta reunión de seguimiento es un acontecimiento común donde el equipo se adapta a la información que escuchó en el ***daily scrum*** , en otras palabras, otra reunión de inspección y adaptación del ciclo.

En general, se recomienda que los gerentes u otras personas que ocupen puestos de autoridad no asistan al ***daily scrum***, o, si lo hacen, no permitirles intervenir. Se corre el riesgo de que el equipo se sienta controlado –con la presión de reportar avances importantes todos los días (una expectativa poco realista) y se inhibe sobre el informe de problemas– y tiende a socavar la autogestión del equipo e invitar a la microgestión.

4.4. Actualización de la pila del *sprint* y gráfico de *sprint* burndown

Tal como ya hemos comentado, todos los días los miembros del equipo se reúnen y, entre otros, actualizan su estimación de la cantidad de tiempo que queda para completar su tarea actual en la **pila del *sprint***. Después de esta actualización, alguien suma las horas que quedan para el equipo en conjunto y se representa en el **gráfico de sprint *burndown***.

Este gráfico muestra todos los días una nueva estimación de la cantidad de trabajo (medida en horas de la persona o los puntos relativos) y se mantiene hasta que las tareas del equipo se hayan acabado. Idealmente, es un gráfico de una curva que desciende y que se encuentra en una trayectoria para llegar a cero esfuerzo pendiente en el último día del *sprint*. Por eso, se llama un **diagrama de crema** (de tareas). Y aunque a veces se ve bien, a menudo no es así, esta es la realidad del desarrollo del producto. Lo importante es que muestra al equipo su progreso hacia la meta, no en términos de cuánto tiempo se empleó en el pasado (un hecho irrelevante en términos de progreso), sino en términos de la cantidad de trabajo que sigue habiendo en el futuro, lo que separa al

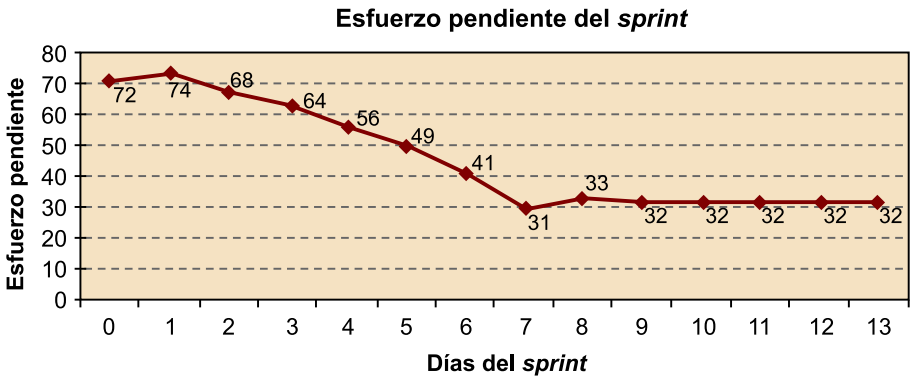
equipo de su objetivo. Si la línea no se quema, es decir, la gráfica de trabajo por completar frente al tiempo que falta no tiene una pendiente negativa, sino que decrece hacia la finalización a mitad del *sprint*, el equipo puede necesitar ejecutar algún procedimiento para corregir la situación:

1) cambiar el enfoque del trabajo o eliminar los obstáculos para aumentar la velocidad;

2) buscar ayuda de alguien de fuera del equipo;

3) reducir el alcance del trabajo;

4) abortar el *sprint* si se considera que no es posible seguir avanzando.



Gráfica de *sprint burndown* correspondiente al *sprint* 3 que se ha explicado antes a título de ejemplo (día 9 del *sprint*)

***Sprint* 3**

El equipo puede ver gráficamente cuál es la evolución del trabajo pendiente y extraer conclusiones con rapidez. A pesar de un buen inicio de *sprint*, el equipo parece haberse encallado durante los últimos tres días en los que el trabajo pendiente no ha disminuido.

4.5. El refinamiento del producto Cartera

Una de las actividades menos conocidas, pero más valiosas en Scrum, es que en cada *sprint* una pequeña parte del esfuerzo la tendría que dedicar el propietario del producto y el equipo al refinamiento (o *grooming*) de la pila de producto. Esto incluye:

1) el análisis detallado de los requisitos;

2) la división de elementos de grandes dimensiones en otros más pequeños;

3) la estimación de los elementos nuevos;

4) la reestimación de los elementos existentes.

Basta una reunión regular semanal con el propietario del producto. Esta actividad no es el refinamiento de los elementos seleccionados por el *sprint* actual, sino que es por los elementos por el futuro, muy probablemente en los próximos uno o dos *sprints*. Con esta práctica, la planificación de *sprint* llega a ser

relativamente simple, ya que el propietario del producto y el equipo Scrum inician la planificación con un claro, muy analizado y calculado conjunto de elementos.

Una señal de que este proceso de refinamiento no se está llevando a cabo (o no se está haciendo bien) es que la planificación de *sprint* incluya preguntas importantes, el descubrimiento de nuevas historias o confusión sobre las existentes.

4.6. Finalizando el *sprint*

Uno de los principios básicos de Scrum es que la duración del *sprint* no se extiende más allá de la fecha asignada, independientemente de si el equipo ha completado el trabajo al que se ha comprometido. Los equipos normalmente acostumbran a comprometerse a más de lo que pueden realizar en los primeros *sprints* y no cumplen con los objetivos. No obstante, en el tercero o cuarto *sprint*, un equipo normalmente se ha dado cuenta de lo que es capaz de entregar y cumple los objetivos.

Se anima a los equipos a tomarse un periodo de tiempo para los *sprints* (por ejemplo, dos semanas) y que no lo cambien. Una duración coherente ayuda al equipo a aprender lo mucho que puede conseguir, lo que ayuda tanto a la estimación como a la planificación a largo plazo. También ayuda al equipo a conseguir un ritmo de su trabajo, por lo que a menudo se califica como el latido del equipo en Scrum.

4.7. Revisión del *sprint*

Una vez finalizado el *sprint*, se lleva a cabo la **revisión de este**, donde el equipo revisa el *sprint* con el propietario del producto. Esto es a menudo mal etiquetado como la **demo**, pero este término no refleja la verdadera intención de esta reunión.

> Una idea clave en Scrum es inspeccionar y adaptar. Se hace para ver y saber lo que está pasando y después evolucionar en función de la información, en la repetición de ciclos.

La revisión del *sprint* es una actividad de inspección y adaptación para el producto. Es un tiempo para que el propietario del producto y las principales partes interesadas aprendan lo que está pasando con el producto y con el equipo (es decir, una revisión del *sprint*) y para que el equipo sepa qué está pasando con el producto y el mercado. En consecuencia, el elemento más importante de la revisión es una conversación en profundidad y la colaboración entre el equipo y el propietario del producto para conocer la situación, obtener con-

sejos y así sucesivamente. La revisión incluye una demostración de lo que el equipo ha construido durante el *sprint*, pero si el enfoque de la revisión es una demostración más que conversación, hay un desequilibrio.

Están presentes en esta reunión el propietario del producto, los miembros del equipo y el *scrum master*, además de los clientes, las partes interesadas, expertos, ejecutivos y cualquier otra persona interesada.

Una guía de Scrum es que se tiene que gastar el menor tiempo posible en la preparación para la revisión del *sprint*; Scrum sugiere no más de dos horas. Es simplemente una demostración de lo que se ha construido. Durante la revisión, cualquiera es libre de plantear preguntas y dar su opinión.

4.8. Retrospectiva del *sprint*

La revisión del *sprint* implica inspeccionar y adaptar en relación con el producto. La **retrospectiva del *sprint***, que sigue a la revisión, implica revisar y adaptar en relación con el proceso.

Esta es una práctica que algunos equipos se saltan, lo que es inaceptable, ya que la autoorganización requiere de la reflexión frecuente ordinaria prevista por la retrospectiva. Es el principal mecanismo para la visibilidad que Scrum proporciona en las áreas de posible mejora con objeto de convertirlo en resultados. Es una oportunidad para que el equipo Scrum trate lo que funciona y lo que no funciona y ponerse de acuerdo en los cambios que desean intentar.

El *scrum master* puede actuar como un facilitador eficaz para la retrospectiva, pero puede ser mejor encontrar un ajeno neutral para facilitar la reunión.

Un buen enfoque es que los *scrum masters* faciliten retrospectivas de los demás, lo que permite la polinización cruzada entre los equipos.

La retrospectiva del *sprint*

Una forma sencilla de estructurar la retrospectiva del *sprint* es poner cuatro columnas en una pizarra con las etiquetas siguientes:

1) ¿qué ha salido bien?

2) ¿qué podría haber sido mejor?

3) ¿qué cosas hay que probar?

4) ¿qué problemas hay para escalar?

Y después ir dando vueltas por la sala y que cada persona añada uno o más elementos a las listas: ideas y experiencias, positivas y negativas, correspondientes a cada una de las columnas. Dado que muchos conceptos se repetirán, se pueden añadir marcas a las repetidas, de forma que los elementos comunes queden claros.

Después, el equipo busca las causas subyacentes y se llega a unas conclusiones y a unas acciones de mejora, de acuerdo con un pequeño número de cambios para tratar en el

próximo *sprint*, junto con el compromiso de revisar los resultados en la cercana retrospectiva de *sprint*.

Una práctica útil al final de la retrospectiva es que el equipo etiquete cada uno de los elementos de cada columna con las siguientes etiquetas:

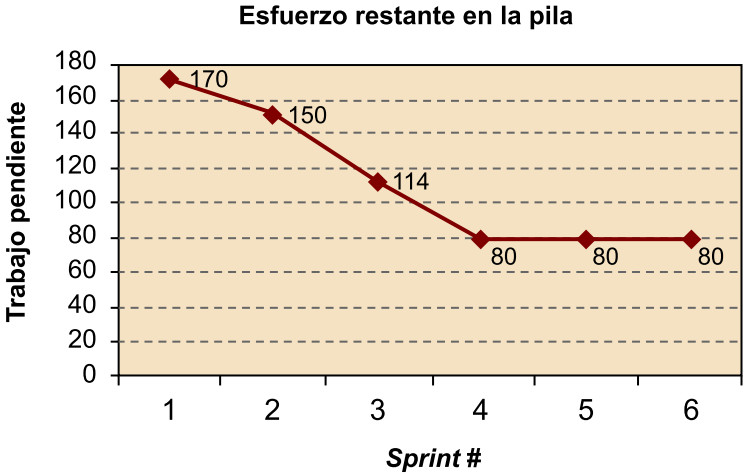
- **S** si es causada por Scrum (en otras palabras, sin Scrum no estaría pasando);
- **E** si está expuesto por Scrum (en otras palabras, estaría sucediendo con o sin pila de producto, pero Scrum lo da a conocer al equipo); y
- **N** si no está relacionado con Scrum (como el clima).

Habitualmente, el equipo puede anotar una gran cantidad de **S** en lo que funciona bien junto al tablero y una gran cantidad de **E** en lo que podría funcionar mejor, lo que es una buena noticia. Si lo que podría funcionar mejor está en la lista más larga, es un buen resultado, puesto que el primer paso para resolver los problemas subyacentes es hacerlos visibles y Scrum logra sacar a la luz los problemas ocultos de la organización.

4.9. Actualización de la pila de producto y *product burndown*

En este punto, algunos elementos se han acabado, algunos se han añadido, algunos tienen nuevas estimaciones y algunos han caído de la meta fijada. El propietario del producto es responsable de asegurar que estos cambios se reflejen en la **pila de producto**.

Además, Scrum incluye un gráfico llamado *product burndown* que muestra el progreso hacia la fecha de lanzamiento. Es análogo al gráfico de *sprint burndown*, pero está en el nivel más alto de los elementos (historias) en lugar de tareas del *sprint*. Como propietario de producto novel, es poco probable que sepa por qué o cómo crear esta tabla, así se trata de otra oportunidad para que un *Scrum master* ayude al propietario del producto.



Ejemplo de un *product burndown* que muestra la cantidad de esfuerzo pendiente para completar el producto después del *sprint* 3.

4.10. Siguiente *sprint*

Después de la revisión del *sprint*, el propietario del producto puede actualizar la pila de producto con cualquier nueva idea. En este punto, el propietario del producto y el equipo están listos para empezar otro **ciclo de *sprint***. No hay tiempo de inactividad entre *sprints*.

Los equipos van normalmente a partir de una retrospectiva de *sprint*, una tarde, en la próxima reunión de planificación de *sprint* a la mañana siguiente o después del fin de semana, por ejemplo.

Uno de los principios del desarrollo ágil es el **ritmo sostenible**, y solo por las horas de trabajo regulares en un nivel razonable los equipos pueden seguir este ciclo indefinidamente.

Bibliografía

Cohn, Make (2009). *Succeeding with Agile: Software Development Using Scrum*. Disponible en: http://www.amazon.com/succeeding-agile-software-development-using/dp/0321579364.

The Scrum Guide. The official rulebook. Disponible en: http://www.scrum.org/scrumguides/