v.4.0 (2024)

SCRUM MASTER

Scrum Manager®

Scrum Master

Temario troncal 1

Versión 4 (2024)

Scrum Manager ®

Fecha de la versión: mayo 2024.
Autora de la versión: Marta Palacio.
Ilustraciones: María de la Fuente Soro y Ana Andrés Soria.
Agradecimientos: Alexander Menzinsky, Claudia Marcionni y Rubén Álvarez.
Uncovering Better Ways SLU es la editora y propietaria de los derechos de distribución, que libera en los términos de la licencia Creative Commons-by-nd-nc 4.0.
Derechos registrados en Safe Creative. Nº de registro: 2405168009113

Índice de contenidos

Prefacio **4**
Introducción **6**
Agilidad 6
Procesos y gestión predictiva 7
El Manifiesto Ágil 9
Los 12 principios del Manifiesto Ágil 13
Origen de scrum 14
Desmontando la gestión de proyectos 16
Ingeniería secuencial 18
Gestión evolutiva: ingeniería concurrente y agilidad 18
Scrum 19
Diferenciando prácticas de principios y valores 20
PARTE I: EL CICLO SCRUM **21**
El ciclo scrum 22
Comprometidos e implicados 25
Roles 26
Propietario del producto 26
Desarrollador 27
Scrum master 27
Artefactos 28
Pila del producto: los requisitos del cliente 29
Pila del sprint 31
Incremento 32
Eventos 33
Sprint 34
Reunión de planificación del sprint 34
Scrum diario 37
Revisión del sprint 38
Retrospectiva del sprint 39
Medición y estimación ágil 41
Unidades relativas: puntos de historia 42
Tiempo real y tiempo ideal 43
PARTE II: PRINCIPIOS Y VALORES **44**
Principios y valores scrum **45**
Principios 46
Valores 47
Las personas y sus roles 47

Propietario del producto.... 48
Desarrolladores.... 48
Scrum master.... 48
Artefactos.... 49
Pila del producto.... 49
Pila del sprint.... 49
Incremento.... 49
Eventos.... 49
Sprint.... 49
Reunión de planificación del sprint.... 50
Scrum diario.... 50
Revisión del sprint.... 50
Retrospectiva del sprint.... 50
Prácticas para flexibilizar scrum.... 51
Control de avance.... 51
Estimación.... 53
Métodos de trabajo.... 55
APÉNDICE.... 58
Información, recursos y comunidad profesional.... 59
Redes sociales y profesionales.... 59
Mejora continua y control de calidad.... 59
Referencias bibliográficas.... 59

守破離

Shu Ha Ri

El aprendizaje de cualquier habilidad tiene tres etapas:
Shu: se elige una técnica y, asumiendo que es correcta, se intenta imitar.
Ha: se coleccionan más técnicas.
Ri: se experimenta e inventan nuevas técnicas mezclando, combinando y modificando.

Las técnicas de etapa shu son aplicables en general. Las técnicas de etapa ri sólo funcionan en casos concretos, y requieren de conocimiento experto para saber cuándo y cómo aplicarlas.

«No puedes ganar en una industria competitiva utilizando técnicas shu.»
Alistair Cockburn

Prefacio

Presentamos el temario de la certificación oficial Scrum Master de Scrum Manager®. En él se explica cómo implantar y mejorar *scrum* aplicado a la gestión ágil de proyectos, equipos y organizaciones.

Nos dirigimos a todas las personas interesadas en el conocimiento del modelo de gestión ágil denominado scrum, bien para aplicarlo en su trabajo diario o en el de su equipo, o para aprender a gestionar de forma ágil en diversos proyectos.

Aunque este modelo surgió dentro de empresas del sector tecnológico, hoy en día se encuentra en todo tipo de entornos innovadores; el factor común es siempre la producción de conocimiento, la inestabilidad y el cambio rápido y constante. Muchas empresas han descubierto que en este tipo de industrias la gestión ágil es la que mejor se adapta.

El temario es adecuado si se trabaja en cualquiera de las llamadas «empresas del conocimiento», en un entorno cambiante, y especialmente si se trabaja en la **gestión de culturas y personas**. Sobre todo, es adecuado si **el valor de su producto depende del talento de personas motivadas**, más que de los procesos y herramientas que éstas emplean.

***

El contenido está dividido en tres partes:

La **introducción** contextualiza el nacimiento de scrum y define qué es la agilidad en el contexto empresarial. Se recomienda su lectura, sobre todo si es la primera vez que leemos sobre gestión ágil. De no ser así, se puede omitir, aunque también puede que se aprenda algo nuevo. Además se explica la diferencia que Scrum Manager® marca entre «prácticas» y «principios y valores» ágiles, clave para lo que viene a continuación.

La **primera parte** se centra en las **prácticas más extendidas de scrum**. Desarrolla los «roles», «artefactos» y «eventos» que se han asentado como estándar con el tiempo y que se manejan en entornos con este modelo de gestión.

La **segunda parte** ahonda en los **principios y valores** de los que surgen estos conceptos. Se explica lo que Scrum Manager® denomina «principios» y «valores» scrum; un uso del modelo más libre pero más consciente. Por último, aparecen algunas prácticas más que, aunque no forman parte del marco estándar, suelen emplearse y completan el inventario de herramientas de gestión ágil.

***

Al hablar sobre scrum se manejan muchos anglicismos y términos que adquieren un significado concreto dentro del mundo de la gestión ágil. Cuando se introduzcan

palabras de este tipo se señalarán yendo entre comillas « », para facilitar la búsqueda de información y señalar que se trata de una palabra con connotación especial.

Introducción

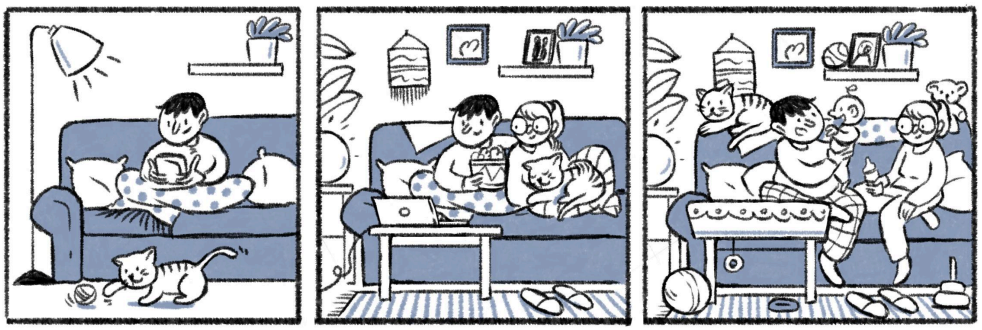
Agilidad

La gestión ágil surgió como antítesis a un cierto modelo de gestión al que haremos referencia con frecuencia: la gestión de proyectos predictiva. Ambas tienen sus virtudes y resultan más útiles en ciertas industrias. La predictiva se centra en planificar, en calcular un presupuesto y marcar plazos de entrega. Si el proyecto final se termina en la fecha acordada, sin exceder el coste y con todas las funcionalidades del plan inicial, se considera un éxito.

Por muy razonable que suene esta estrategia, si trabajamos en industrias que se caractericen por su constante y rápida evolución, encontraremos que tiene muchos inconvenientes. Esa definición de un proyecto exitoso sirve en un entorno estable, con productos que son resultado de una atención escrupulosa a procesos y protocolos.

La **gestión predictiva** es fruto de la Revolución Industrial: viene del mundo de la construcción, de la automovilística, de las fábricas. Por ejemplo: si lo que el cliente busca es una casa, ésta tendrá que construirse de forma que sea sólida, segura, cumpla con las necesidades de sus habitantes; y, en un escenario ideal, dentro del plazo previsto y sin exceder el coste.

Pero hoy en día se fabrican y venden productos que no tienen nada que ver. Primero, porque pueden ser abstractos, como una película o una *app* móvil. Se pueden probar cosas nuevas durante el desarrollo, viendo de forma empírica lo que funciona y lo que no. Es posible realizar ajustes sobre la marcha. Además, se puede partir de un primer esbozo con lo básico que se necesita e ir creciendo. El escenario puede cambiar, y una funcionalidad que parecía esencial al principio puede estar desfasada para la fecha de entrega. O puede ser que un competidor lance una novedad interesante y eso lleve a revisar las prioridades del producto. Para ser competitivo se necesita la capacidad de responder rápido en escenarios de trabajo inciertos; donde no se cuenta con requisitos estables al concebir nuevos productos o servicios; con clientes que necesitan empezar a usar el producto lo antes posible y mejorarlo de forma continua; productos en los que la innovación es un valor clave.



Éstas y más razones que veremos son las que llevaron a cuestionar los modelos de gestión predictiva, que parecían no encajar con la realidad de lo que se necesitaba en las llamadas «empresas del conocimiento». Entendiendo como tales aquellas que desarrollan productos o servicios basándose en el conocimiento más que en las herramientas y los procesos.

El entorno de trabajo de estas empresas se parece muy poco al que originó la gestión de proyectos predictiva.

Ahora existen mercados con una evolución tan rápida que es inútil pretender iniciar proyectos con un plan cerrado. Se necesitan estrategias que entreguen resultados tangibles y pronto; que permitan responder a tiempo a los cambios. Se construye el producto al mismo tiempo que se modifican e introducen nuevos requisitos. El cliente parte de una visión más o menos clara, pero el nivel de innovación que requiere, así como la velocidad a la que se mueve el entorno de su negocio, no le permiten prever con detalle cómo será el resultado final.

Hoy hay directores de producto que no necesitan conocer cuáles van a ser las 200 funcionalidades que tendrá el producto final, ni si estará terminado en 12 o en 16 meses. Hay clientes que necesitan disponer de una primera versión con funcionalidades mínimas en cuestión de semanas, y no un producto completo dentro de uno o dos años. Su interés es poner en el mercado rápidamente un concepto nuevo y desarrollar de forma continua su valor.

De dónde venimos y por qué la agilidad se suele asociar con la informática

El conocimiento evoluciona siguiendo un patrón dialéctico de tesis, antítesis y síntesis. A cada tesis le surge una antítesis, que pone en evidencia sus problemas y contradicciones. La antítesis también resulta inadecuada de alguna manera, y de la confrontación de las dos surge un tercer momento llamado síntesis: una resolución y nueva comprensión del problema.

> «Es en este momento que la tesis y antítesis anteriores se reconcilian y trascienden. No obstante, con el tiempo, incluso la síntesis resultará tendenciosa en algún aspecto. Servirá entonces como tesis para un nuevo movimiento dialéctico, y así el proceso continúa de forma zigzagueante en espiral.» (Nonaka 2004)

Los marcos de prácticas ágiles no surgieron como una tesis de conocimiento, sino como antítesis al que la ingeniería del *software* venía desarrollando.

Procesos y gestión predictiva

En 1968, durante la llamada «crisis del software», la organización OTAN celebró la primera conferencia centrada en analizar los problemas de la programación. Se puso de manifiesto la necesidad de crear una disciplina científica que permitiera aplicar un enfoque sistemático y cuantificable al desarrollo, operación y mantenimiento de sistemas informáticos. Esto se tradujo en el intento de aplicar ingeniería de procesos al software,

surgiendo así la «ingeniería del software» (Bau 1969). Esta primera estrategia (tesis) se basó en dos pilares:

- **Ingeniería de procesos:** en los entornos de producción industrial existía un principio básico de calidad, contrastado con éxito: «la calidad del resultado depende de la calidad de los procesos empleados». Dicho de otra forma: no se necesita a gente brillante o muy cualificada; mientras los procesos empleados sean de calidad, el resultado será de calidad.
- **Gestión predictiva:** un tipo de gestión que se centra en garantizar el cumplimiento de agendas y presupuestos.

Mientras la disciplina evolucionaba y se perfeccionaba a través de diferentes modelos de procesos y cuerpos de conocimiento para gestión de proyectos (MIL-Q9858, ISO9000, ISO9000-3, ISO 12207, SPICE, SW-CMM…) en la industria del software surgían dudas y se cuestionaba esta estrategia.

Desde mediados de los 90 hasta 2005-2010 han sido habituales las posturas radicales entre los defensores de los modelos de procesos (tesis) y de los marcos ágiles (antítesis).

> «La diferencia entre un atracador de bancos y un teórico de CMM es que con el atracador se puede negociar.» (Orr 2002)

> «La evaluación en CMM depende más de una buena presentación en papel que de la calidad real del producto de software. Tiene que ver más con el seguimiento a ciegas de una metodología que con el desarrollo y puesta en producción de un sistema en el panorama tecnológico.» (Orr 2002)

> «Si uno pregunta a un ingeniero de software típico si cree que CMM se puede aplicar a los métodos ágiles, responderá o con una mirada de sorpresa o con una carcajada histérica.» (Turner & Jain 2002)

No estaba claro que la planificación predictiva fuese apropiada para cualquier proyecto. En la práctica puede verse que a veces los criterios del éxito no son siempre el cumplimiento de fechas, costes y funcionalidades preestablecidas. Por otra parte, también se cuestiona si en el desarrollo de software, como en otros trabajos basados en el conocimiento, se puede producir con patrones de procesos industriales. Se empieza a aceptar que el conocimiento tácito de la persona que realiza el trabajo puede aportar más al valor del resultado que la tecnología y los procesos empleados (→ «Desmontando la gestión de proyectos», «Conocimiento»).

El *Manifiesto Ágil*

> «Estamos poniendo al descubierto mejores métodos para desarrollar software, haciéndolo y ayudando a otros a que lo hagan. Con este trabajo hemos llegado a valorar:
>
> - A los individuos y su interacción, por encima de los procesos y las herramientas.
> - El software que funciona, por encima de la documentación exhaustiva.
> - La colaboración con el cliente, por encima de la negociación contractual.
> - La respuesta al cambio, por encima del seguimiento de un plan.
>
> Aunque hay valor en los elementos de la derecha, valoramos más los de la izquierda.»

En febrero de 2001, 17 profesionales del software fueron convocados por Kent Beck, que había publicado un par de años antes el libro en el que explicaba la nueva metodología *Extreme Programming* (Beck 1999). Todos ellos tenían algo en común: eran críticos de los modelos de producción basados en procesos.

Se reunieron en Salt Lake City para discutir sobre los procesos empleados por los equipos de programación.

En la reunión se acuñó el término «métodos ágiles» para definir a aquellos que estaban surgiendo como alternativa a las metodologías formales, tales como CMM-SW (precursor de CMMI), PMI, SPICE (proyecto inicial de ISO 15504, a su vez precursor de ISO 33000)... a los que consideraban excesivamente pesados y rígidos por su carácter normativo y fuerte dependencia de planificaciones detalladas previas al desarrollo.

Los integrantes de la reunión resumieron en cuatro postulados lo que ha quedado denominado como *Manifiesto Ágil*, que son los valores sobre los que se asientan estos métodos. Son los que abren y se desarrollan en este apartado. También establecieron 12 principios, que mencionaremos al final.

«Valoramos más a los individuos y su interacción que a los procesos y las herramientas.»

El postulado más importante. Es indudable que los procesos ayudan: sirven de guía de operación, y disponer de las herramientas adecuadas mejora la eficiencia. Pero hay tareas que requieren talento y necesitan personas motivadas que lo aporten.

En una producción basada en procesos, lo que se persigue es que la calidad del resultado sea consecuencia de éstos, más que del conocimiento aportado por las personas que los ejecutan. En el desarrollo ágil, en cambio, los procesos son sólo una ayuda; un soporte para guiar el trabajo.

La defensa a ultranza de los procesos lleva a afirmar que con ellos se pueden conseguir resultados extraordinarios con personas mediocres, pero lo cierto es que esto no es así cuando se necesita creatividad e innovación.

«Valoramos más el software que funciona, por encima de la documentación exhaustiva.»

El *Manifiesto Ágil* no considera inútil la documentación: sólo la innecesaria. Los documentos son un soporte físico que permite registrar y comunicar información relevante para el proyecto. Además, por cuestiones legales o normativas, pueden ser obligatorios. Pero su relevancia debe ser menor que la del producto.

¿A qué nos referimos? Poder anticipar cómo funcionará el producto final observando prototipos y partes ya terminadas supone un feedback estimulante y enriquecedor, que genera ideas imposibles de concebir en un primer momento. Es por eso que elaborar un documento de requisitos muy detallado antes de empezar supone a menudo una pérdida de tiempo.

Se puede argumentar que la documentación detallada facilita transmitir información entre las personas implicadas en el proyecto, pero rara vez es así. Le falta la riqueza y producción de valor que se logra con la comunicación directa y la interacción con prototipos del producto. De hecho, no sólo carece de esas ventajas, sino que entorpece y crea barreras burocráticas entre departamentos e individuos.

Por eso, siempre que sea posible, se debe reducir al mínimo indispensable el uso de documentación. Lo ideal es eliminar toda aquella que consuma trabajo sin aportar un valor directo al producto.

«Valoramos más la colaboración con el cliente que la negociación contractual.»

El objetivo de un proyecto ágil no es controlar la ejecución para garantizar que los planes iniciales se cumplen, sino proporcionar de forma continua el mayor valor posible al producto.

Como ya hemos comentado, al desarrollar productos en evolución continua (como una aplicación web, por ejemplo) no se puede definir en un documento de requisitos cerrado cómo debería ser el producto final. Es más eficiente tomar *feedback* directo a la vez que se desarrolla el producto, y en consecuencia redefinir y mejorar los requisitos de las partes que quedan.

Para que el cliente sea consciente de los cambios conforme van surgiendo, debe acompañar al equipo durante el proceso. La mejor relación entre cliente y equipo es una de implicación y colaboración directa, no una contractual, que tiende a delimitar responsabilidades al principio del proyecto y ya está.

«Valoramos más la respuesta al cambio que el seguimiento de un plan.»

Los principales valores de la gestión ágil son la anticipación y la adaptación, diferentes a los de la gestión de proyectos ortodoxa: planificación y control para garantizar el cumplimiento del plan.

Para desarrollar productos de requisitos inestables, en los que es inherente el cambio y la evolución rápida y continua, resulta mucho más valiosa la capacidad de respuesta que la de seguimiento y aseguramiento de planes.

Los 12 principios del *Manifiesto Ágil*

Además de los cuatro postulados que acabamos de ver, el *Manifiesto Ágil* establece estos 12 principios:

1. Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.
2. Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
3. Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los períodos breves.
4. Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
5. Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.
6. La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.
7. El software que funciona es la principal medida del progreso.
8. Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica enaltece la agilidad.

10. La simplicidad como arte de maximizar la cantidad de trabajo que no se hace, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos que se autoorganizan.
12. En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

Origen de scrum

Scrum es un modelo de desarrollo ágil caracterizado por:

- Equipos autónomos y autogestionados que comparten su conocimiento de forma abierta y aprenden juntos.
- Una estrategia de desarrollo incremental, en lugar de la planificación completa del producto.
- Basar la calidad del resultado en el conocimiento tácito de las personas y su creatividad; no en la calidad de los procesos empleados.
- Solapar las diferentes fases del desarrollo, en lugar de realizarlas una tras otra en un ciclo secuencial o «de cascada».

El origen de la palabra se encuentra en un ámbito muy alejado del de la gestión de proyectos: en el deporte. En *rugby*, «scrum» es el término que define a la formación en la que ambos equipos, agazapados y atenazados entre sí, empujan para obtener el balón sin tocarlo con la mano.

Ahora bien, para lo que a nosotros nos interesa, tenemos que desplazarnos al Japón de los 80, cuando los investigadores Ikujiro Nonaka y Hirotaka Takeuchi dieron una dimensión polisémica al término.

Identificaron una novedosa forma de desarrollo en las empresas de manufactura industrial que estaban obteniendo los mejores resultados de innovación y tiempo de salida al mercado: Fuji Xerox, Canon, Honda, Nec, Epson, Brother, 3M y Hewlett-Packard (Nonaka 1986). Compararon su forma de trabajo en equipos autogestionados con el avance en formación de los jugadores de rugby, de ahí el término.

Aunque esta forma de trabajo surgió en empresas de productos tecnológicos, en la manufactura industrial, a partir de 1995 se empezaron a aplicar también a la industria del software. En este año, Ken Schwaber presentó en OOPSLA (conferencia anual *Object-Oriented Programming, Systems, Languages & Applications)* una metodología de desarrollo de software basada en un ambiente scrum, usando ese mismo término (Schwaber 1995). Este primer marco presentaba una serie de fases y «artefactos»: *pregame, game, postgame, planning, sprints, wrap...* Algunos se han mantenido y los veremos, pero en general las reglas del juego han cambiado mucho desde entonces.

No existe una autoridad que determine lo que es scrum y lo que no. Ha cambiado y seguirá evolucionando con las aportaciones de la comunidad de profesionales, que

define las prácticas que resultan más útiles. El espíritu original, eso sí, se mantiene: las prácticas deben ayudar a equipos a autogestionarse y mantener un flujo de avance continuo, produciendo resultados de forma iterativa y frecuente. Es lo bonito y lo emocionante de trabajar en este tipo de empresas.

Entre los eventos y prácticas que se han ido incorporando se incluyen las reuniones retrospectivas, las reuniones de refinamiento de la pila de producto, *DoR (Definition of Ready)*, *story maps…*

Scrum Manager® usa el término «scrum» entendido con el significado original, el que le dieron Nonaka y Takeuchi.

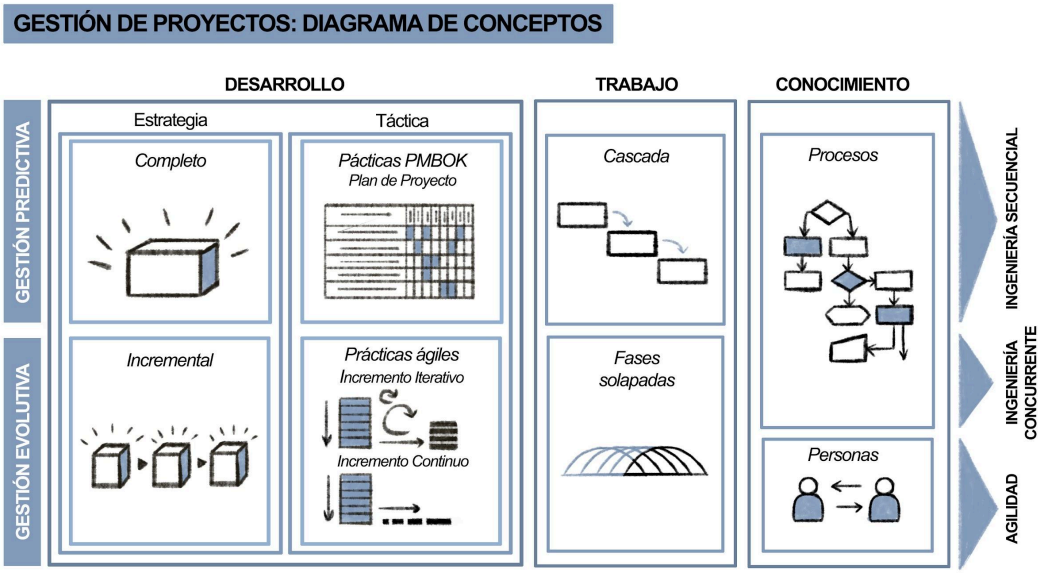
Desmontando la gestión de proyectos

Desde los 80 se han desarrollado tantos modelos y prácticas para mejorar la calidad y eficiencia de los proyectos que puede resultar abrumador. En este apartado vamos a trascender las etiquetas y a resumir la base de estos marcos, los principios que subyacen y las estrategias con las que se desarrollan.

Usaremos como coordenadas tres conceptos y dos modelos de gestión.

- Los tres primeros son: desarrollo, trabajo y conocimiento.
- Los modelos, que ya han sido mencionados, son: gestión predictiva y gestión evolutiva.

Con estas cinco ideas se despeja y simplifica el aparente laberinto de modelos de marcos de trabajo, pues al final todos ellos se pueden entender dentro de este diagrama.



1. Desarrollo

El desarrollo del proyecto puede darse de forma completa o incremental.

- **En el caso de un desarrollo completo**, la descripción de lo que se desea obtener está disponible al inicio del proyecto; es completa y detallada y sirve de base para estimar. Con el plan inicial se planifican la agenda de trabajo y los recursos necesarios. Durante la ejecución se gestiona el cumplimiento de lo que se ha previsto.
- **En los desarrollos incrementales**, la descripción completa de lo que se desea obtener no está disponible al inicio. Se complementa y evoluciona durante el desarrollo, que se puede gestionar con dos tácticas diferentes:

- **Desarrollo incremental continuo:** empleando técnicas para lograr un flujo continuo de desarrollo de las funcionalidades o partes del producto, que se entregan de forma continua al cliente.

- **Desarrollo iterativo:** empleando técnicas de tiempo prefijado o «timeboxing» para mantener la producción de incrementos del producto a un ritmo fijo. Este es el marco de producción empleado al aplicar el marco estándar de scrum, que define como *sprint* (→ «Eventos») a cada iteración de desarrollo, al final de la cual se produce un «incremento» del producto: una parte entregable y lista para usarse.

2. Trabajo

La forma de trabajar puede ser secuencial («en cascada») o concurrente.

- **El trabajo secuencial divide el trabajo en fases.** Una fase nueva comienza cuando se termina la anterior. El ejemplo más habitual es el ciclo de cascada definido en ingeniería del software, cuyas fases son: definición de requisitos, análisis, diseño, codificación, pruebas e implementación.
- **Trabajar de forma concurrente** significa solapar en el tiempo las diferentes fases. Siguiendo con el ejemplo de ingeniería de software, todas las fases del párrafo anterior se revisarían de forma simultánea y continua.

3. Conocimiento

Los diferentes modelos pueden ubicar el conocimiento o bien en los procesos o en las personas.

- **En una producción basada en procesos:** el conocimiento es explícito. La calidad del resultado se encuentra, en mayor medida, en los procesos y la tecnología empleada.
- **En la producción basada en las personas:** el conocimiento es tácito. La calidad del resultado depende de la experiencia de los miembros de la organización. No de seguir un proceso de manera correcta, sino de que quienes trabajen sean personas motivadas y con talento.

Un ejemplo de conocimiento explícito y tácito sería la diferencia entre la cena preparada por un robot de cocina o de forma libre. Cualquiera puede preparar la primera siguiendo las instrucciones.

El resultado siempre será el mismo sin importar la habilidad. En el segundo caso, prima el talento de la persona. Un aficionado no preparará lo mismo que un chef de alta cocina. Ambos se beneficiarán

de tener buenas herramientas, como sartenes que no se peguen o cuchillos bien afilados, pero son sólo una ayuda.

> «El conocimiento tácito es personal, específico del contexto, y por tanto difícil de formalizar y comunicar. El conocimiento explícito o «codificado», por otro lado, es conocimiento que puede transmitirse con lenguaje formal y sistemático.» (Nonaka 1995)

Ingeniería secuencial

La ingeniería secuencial, también llamada gestión predictiva, tiene como objetivo ofrecer resultados predecibles. Un proyecto exitoso según estos modelos desarrollará el producto previsto sin exceder el plazo ni los recursos acordados. El tipo de desarrollo de la gestión predictiva es «completo», y se emplean prácticas de planificación tradicional.

En el mundo del software los principales referentes en el desarrollo de conocimiento para este tipo de gestión son PMI e IPMA y los modelos de procesos CMMI, ISO 33000, SPICE… Todos ellos emplean ingeniería secuencial y producción basada en procesos.

Gestión evolutiva: ingeniería concurrente y agilidad

La gestión evolutiva tiene como objetivo entregar lo antes posible un producto mínimo viable, e incrementar su valor de forma continua. Emplea una estrategia de solapamiento de las fases de trabajo y desarrollo incremental, que se puede obtener manteniendo un ritmo de iteraciones breves y cíclicas o un flujo de desarrollo constante.

Puede llevarse a cabo con producción basada en procesos (ingeniería concurrente) o con producción basada en personas (agilidad). Es importante esta distinción, porque sin ella se generan situaciones confusas que llegan a considerar agilidad a la simple aplicación de las reglas estándar de scrum (ciclo de incremento iterativo con roles y artefactos definidos), o al simple uso de técnicas de gestión visual *kanban* para mantener un flujo continuo de trabajo.

Ingeniería concurrente	Agilidad
Emplea recursos propios de la gestión ágil: solapamiento de fases de desarrollo, equipos multidisciplinares e iteraciones frecuentes de mejora. Se centra en la calidad de los procesos.	Reduce o elimina tareas administrativo-burocráticas que no aportan valor al producto o al sistema de desarrollo. Propia de las empresas del conocimiento. Se centra en el conocimiento tácito de las personas, en la cultura y el talento.

Vídeo explicativo: Gestión predictiva y evolutiva (→ «Referencias bibliográficas»).

Vídeo explicativo: Objetivo de la gestión predictiva y de la gestión ágil (→ «Referencias bibliográficas»).

Scrum

Recapitulando, volvemos a encontrarnos con las características de scrum (→ «Origen de scrum») y vemos cómo encajan dentro de las características de la gestión ágil:

- Utiliza una estrategia de desarrollo incremental (que puede ser iterativo, usando «timeboxing», o continuo).
- Solapa las diferentes fases del desarrollo.
- Basa la calidad del resultado en el conocimiento tácito de las personas y su creatividad.
- Además, scrum se caracteriza también por el trabajo en equipos autónomos y autogestionados, que comparten su conocimiento y aprenden juntos. De ahí el nombre y la metáfora de «avanzar en scrum».

Diferenciando prácticas de principios y valores

Cuando se empieza a trabajar con scrum, como con cualquier otra herramienta, es recomendable leer el manual de Scrum Master y seguir las instrucciones; es decir, adoptar el marco tradicional o estándar. En esta primera parte explicamos en qué consiste, y cuáles son los roles, artefactos y eventos que lo configuran. El marco es, por así decir, una plantilla, pero no un molde. **No son reglas** que deban seguirse al pie de la letra para garantizar velocidad o calidad, sino de un buen **punto de partida** para empezar a trabajar de forma iterativa.

Conviene no engañarse: **si nuestro foco está puesto en el proceso, en seguir las reglas, y no en el conocimiento tácito o talento de las personas, no se está haciendo agilidad, sino ingeniería concurrente.** La agilidad requiere confianza en las capacidades del equipo. Las herramientas y procesos que se usan deben ayudar a gestionar el trabajo; no a las personas. Ya que el objetivo final es que los equipos sean capaces de **autogestionarse.**

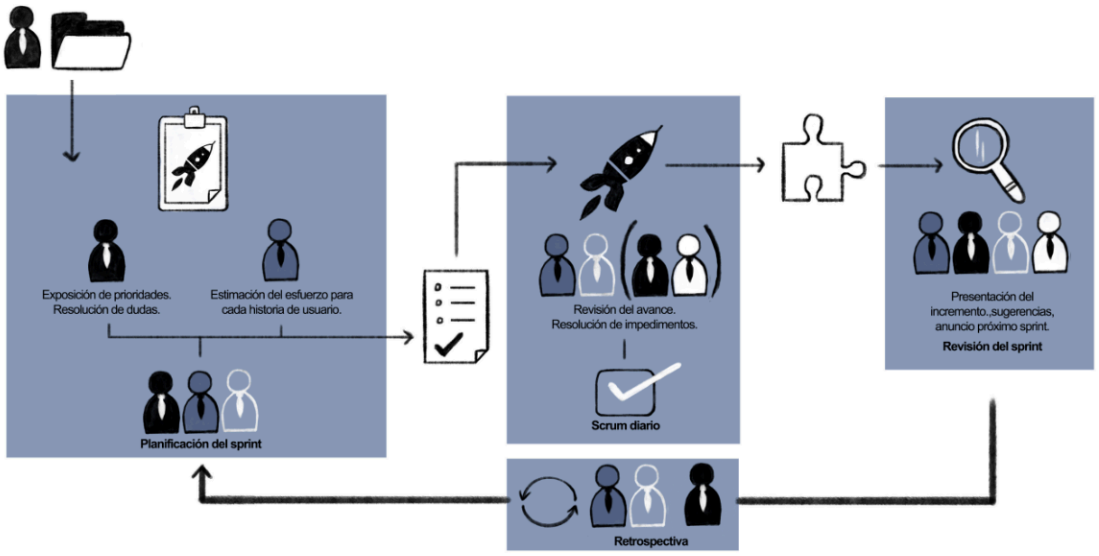
Cuando se alcanza un flujo de avance iterativo, se puede intentar ir más allá de un marco estándar de scrum. Llega el momento de desaprender las prácticas y de apoyarse en los principios y valores de scrum, adaptando éste y otras técnicas y marcos a las características concretas del proyecto o del equipo. En la mayoría de empresas ágiles estas prácticas se pueden adaptar, y de hecho se adaptan.

La primera parte desarrolla las técnicas más extendidas de scrum, el marco estándar que puede encontrarse aquí, en Internet y en otros manuales: las reglas de aplicación, roles, eventos y artefactos. En la segunda parte explicaremos cómo quitar los ruedines de la bici, que vienen muy bien al principio pero pueden entorpecernos a la larga, para seguir avanzando.

PARTE I: EL CICLO SCRUM

Aprendiendo las prácticas estándar

El ciclo scrum



ROLES

PROPIETARIO DEL PRODUCTO
Determina las prioridades.
Una sola persona.

DESARROLLADOR
Construye el producto.

SCRUM MASTER
Gestiona y facilita la ejecución de las reglas del Scrum.

INTERESADOS
Resto de implicados. Asesoran y observan.

ARTEFACTOS

PILA DEL PRODUCTO
Relación de requisitos del producto, no es necesatio excesivo detalle. Priorizados. Lista en evolución y abierta a todos los roles. El propietario del producto es su responsable y quien decide.

PILA DEL SPRINT
Requisitos comprometidos por el equipo para el sprint con nivel de detalle suficiente para su ejecución.

INCREMENTO
Parte del producto desarrollada en un sprint, en condiciones de ser usada (pruebas, codificación limpia y documentada).

EVENTOS

PLANIFICACIÓN DEL SPRINT
1 jornada de trabajo (máx.) El propietario del producto explica las prioridades. El equipo estima el esfuerzo de los requisitos prioritarios y se elabora la pila del sprint. El equipo define en una frase el objetivo del sprint.

SPRINT
Ciclo de desarrollo básico en el marco estándar de scrum, de duración recomendada inferior a un mes y nunca mayor de 6 semanas.

SCRUM DIARIO
15 minutos máximo. Responsabilidad del equipo. cada miembro expone: Lo que hizo ayer, lo que va a hacer hoy y si tiene o prevé problemas. Se actualiza la pila del sprint.

REVISIÓN DEL SPRINT
Informativa, máximo 4 horas. Presentación del incremento, planteamiento de sugerencias y anuncio del prócimo sprint.

RETROSPECTIVA
El equipo autoanaliza la forma de trabajo. Identificación de fortalezas y debilidades. Refuerzo de las primeras, plan de mejora de las segundas.

A día de hoy, los componentes del ciclo estándar de scrum son:

- Equipo scrum, compuesto de los siguientes roles:
  - Desarrollador.
  - Propietario del producto.
  - *Scrum master*.
- Artefactos:
  - Pila del producto.
  - Pila del sprint.
  - Incremento.
- Eventos:
  - Sprint.
  - Reunión de planificación del sprint.
  - Scrum diario.
  - Revisión del sprint.
  - Retrospectiva del sprint.

Se comienza con la visión general del resultado que se desea, y a partir de ella se especifica y da detalle a las funcionalidades que se desean obtener en primer lugar.

Cada ciclo de desarrollo o iteración (sprint) finaliza con la entrega de una parte operativa del producto (incremento). **La duración de cada sprint puede ser de entre 1 y 3 semanas.** Lo más habitual es que tengan siempre la misma medida, marcando una cadencia, pero ésta puede ir **evolucionando o ajustarse.**

Vídeo explicativo: El marco de desarrollo scrum (→ «Referencias bibliográficas»).

Scrum maneja empíricamente la evolución del proyecto con las siguientes tácticas:

Revisión de las iteraciones

Al finalizar cada sprint se revisa funcionalmente el resultado, con todos los implicados en el proyecto. Por tanto, la duración del sprint es el período de tiempo máximo para descubrir planteamientos erróneos, mejorables, o malinterpretaciones en las funcionalidades del producto.

Desarrollo incremental

No se trabaja con diseños o abstracciones. El desarrollo incremental ofrece al final de cada iteración una parte de producto operativa, que se puede usar, inspeccionar y evaluar.

Solapamiento de fases

Durante la construcción se depura el diseño y la arquitectura, y no se cierran en una primera fase del proyecto. Las distintas fases que el desarrollo en cascada realiza de forma secuencial, una tras otra, en scrum se solapan y avanzan de forma simultánea.

Autogestión

La gestión predictiva asigna al rol de gestor del proyecto la responsabilidad de su gestión y resolución. En scrum los equipos son autogestionados, con un ámbito de decisión suficiente para adoptar las resoluciones que consideren oportunas. Esto agiliza la toma de decisiones y permite responder con rapidez ante imprevistos.

Colaboración

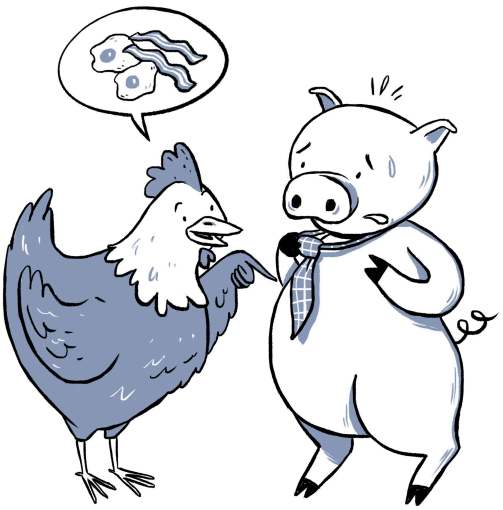
Todos los miembros del equipo colaboran de forma abierta con los demás, según sus capacidades y no según su rol o su puesto.

Mediante la autogestión y la colaboración se puede gestionar con solvencia la labor que de otra forma realizaría un gestor de proyectos.

Comprometidos e implicados

Durante el desarrollo del proyecto son muchas las personas que intervienen y aportan valor, si bien con diferentes niveles de compromiso y responsabilidad, en función de lo cual se suele diferenciar entre «comprometidos» e «implicados».

- **Comprometidos:** intervienen directamente en la construcción del producto o el desarrollo del servicio.
- **Implicados:** otras partes interesadas, tales como dirección, gerencia, comerciales, marketing, operadores del sistema que se desarrolla, soporte a usuarios, etc.



***

Una gallina y un cerdo paseaban por la calle. La gallina preguntó al cerdo:
-¿Quieres abrir un restaurante conmigo?
El cerdo consideró la propuesta y respondió:
-Sí, me gustaría. ¿Cómo lo llamaríamos?
-Huevos con Jamón.
El cerdo se detuvo, hizo una pausa y contestó:
-Pensándolo mejor, creo que no voy a abrir un restaurante contigo. Yo estaría realmente comprometido... mientras que tu estarías sólo implicada.

***

En círculos de scrum es frecuente llamar a los comprometidos (sin ninguna connotación peyorativa) «cerdos» y a los implicados «gallinas». El origen de estos nombres es esta historia, que ilustra la diferencia entre compromiso e implicación en el proyecto.

Roles

Propietario del producto

El «propietario del producto» o «product owner» es quien toma las decisiones del cliente. Su responsabilidad es el valor del producto.

Para simplificar la comunicación y toma de decisiones es necesario que este rol recaiga en una única persona. Si el cliente es una organización grande, o con varios departamentos, puede adoptar la forma de comunicación interna que consideren oportuna, pero en el equipo de trabajo sólo se integra a una persona. Ésta representa al cliente y debe tener el conocimiento y las atribuciones necesarias para tomar las decisiones que le corresponden.

En los desarrollos internos para la propia empresa, suele asumir este rol el product manager o el responsable de *marketing*. En desarrollos para clientes externos, el responsable del proceso de adquisición del cliente. Según las circunstancias del proyecto es posible incluso que el propietario del producto delegue en el equipo, o en alguien de su confianza, pero incluso en esos casos mantiene sus responsabilidades:

- Desarrollo y administración de la pila del producto.
- Exposición de la visión e historias de usuario, y participación en la reunión de planificación de cada sprint (→ «Eventos»).

Es quien está a cargo de la «pila del producto» (→ «Artefactos»). Esto quiere decir que es quien decide en última instancia cómo será el producto final y el orden en el que se van construyendo los incrementos; qué se pone y qué se quita, así como cuál es la prioridad de las «historias de usuario» (→ «Artefactos»). Conoce el plan del producto, sus posibilidades, plan de inversión y el retorno esperado a la inversión realizada. Como representante del cliente, también se responsabiliza de cumplir con los plazos de entrega previstos de las versiones del producto.

Tiene conocimiento experto sobre el entorno de negocio del cliente; sabe cuáles son sus necesidades y el objetivo que se persigue con el proyecto. Esto es fundamental para poder compartir esta visión con el equipo y priorizar requisitos. Debe estar al corriente y realizar análisis constantes del entorno de negocio: evolución del mercado, competencia, alternativas… Y combinar esta información con la que surja del equipo durante el proceso de creación: sugerencias, alternativas técnicas, pruebas y evaluación de cada incremento. Es necesario que conozca scrum, para realizar con solvencia las tareas que le corresponden. Y, en circunstancias ideales, que conozca y haya trabajado previamente con los mismos desarrolladores. La organización debe respetar sus decisiones y no modificar prioridades ni elementos de la pila del producto.

Enlace de interés: Las 5 claves de un buen product owner, Scrum Manager Podcast (→ «Referencias bibliográficas»).

Desarrollador

Cada uno de los profesionales que realizan el «incremento» (→ «Artefactos») de cada sprint. Aunque se les denomine «desarrolladores» en scrum esto no significa que se hable siempre de programadores. Es un término que engloba a las personas cuyo trabajo contribuye directamente a «desarrollar» o construir el «incremento».

Se recomienda que haya entre 3 y 9 desarrolladores. Más allá de 9 resulta difícil mantener la comunicación directa, y se manifiestan con más intensidad los roces habituales de la dinámica de grupos, que comienzan a partir de 6 personas.

Son profesionales multifuncionales, que trabajan de forma solidaria con responsabilidad compartida. Es posible que algún desarrollador esté especializado en áreas concretas, pero la responsabilidad del desarrollo recae sobre todos por igual. Las principales responsabilidades, más allá de la autogestión y uso de tecnologías ágiles, son las que marcan la diferencia entre un «grupo de trabajo» y un «equipo»:

- En **un «grupo de trabajo»** las personas tienen una asignación específica de responsabilidades, y siguen un proceso o pautas de ejecución. Aunque el grupo trabaje en la misma organización, cada quien responde de forma individual.
- **Un «equipo»** tiene espíritu de colaboración y un propósito común: conseguir el mayor valor posible para la visión del cliente. Los desarrolladores se autogestionan para trabajar y responder de forma conjunta. No hay un gestor para delimitar, asignar y coordinar el trabajo. Son los desarrolladores los que se coordinan. Todos aportan y colaboran con el propietario del producto en el desarrollo de la pila del producto, participan en la toma de decisiones, y respetan las opiniones y aportes de los demás. Comparten el objetivo de cada sprint y la responsabilidad del logro. Por último, todos están familiarizados con scrum.

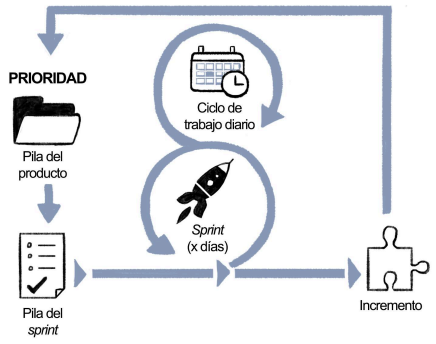
Scrum master

Es el responsable del cumplimiento de las reglas del marco de scrum. Se asegura que éstas son entendidas por la organización y de que se trabaja conforme a ellas. Asesora y da la formación necesaria al propietario del producto y a los desarrolladores, y configura, diseña y mejora de forma continua las prácticas ágiles de la organización. El fin es que el cliente y el equipo de desarrollo sean capaces de organizarse y trabajar con autonomía.

También es responsabilidad suya moderar las reuniones de scrum diarias, gestionar las dificultades de dinámica de grupo que puedan surgir en el equipo, y solucionar los impedimentos detectados durante el scrum diario para que el sprint siga avanzando.

Artefactos

Los artefactos de scrum son sus herramientas, sus bloques de construcción elementales. Ayudan a los «roles» durante los «eventos».



Artefactos más extendidos

Podemos destacar tres artefactos clave para el funcionamiento del marco estándar:

- **Pila del producto / *product backlog:*** lista priorizada de las necesidades del cliente, expresadas como historias de usuario. Al principio del proyecto refleja el MVP (producto mínimo viable) o la visión inicial. Es un documento vivo: durante el desarrollo, crece y evoluciona.
- **Pila del sprint / *sprint backlog:*** lista de trabajo que van a realizar los desarrolladores durante el sprint. (→ «Unidades de trabajo»)
- **Incremento:** resultado de cada sprint. Es una parte del producto que funciona correctamente; debe estar probada y lista para usarse.

Enlace de interés: Cómo crear un backlog, Scrum Manager Podcast (→ «Referencias bibliográficas»).

Otros artefactos

- **Gráfico de avance o *burn down chart*:** indica el trabajo pendiente y la velocidad a la que se están completando las unidades de trabajo, para deducir si se completarán todas en el tiempo estimado. Los desarrolladores lo actualizan a diario.
- **Gráfico de producto o *burn up chart*:** si el gráfico de avance mide lo que falta, el de producto mide cuánto se ha construido o completado.
- **Definition of Ready (DoR):** un acuerdo que define cuándo una historia de la pila del producto está «listo» («ready») para incluirlo en un sprint.
- ***Definition of Done* (DoD):** acuerdo sobre los criterios para considerar que una parte del trabajo (tarea, historia…) está terminada.

Enlace de interés: Definition of Done, Scrum Manager Podcast (→ «Referencias bibliográficas»).

Unidades de trabajo

La «unidad de trabajo» es el elemento que el equipo utiliza para gestionar el trabajo dentro de un sprint. Las unidades más usadas son las historias de usuario, las historias técnicas y las tareas.

- **Historia de usuario:** una descripción corta, simple y específica del valor que un producto o servicio debe proporcionar al usuario final, escrita desde la perspectiva de éste. Se generan durante las conversaciones entre desarrolladores y propietario de producto para lograr una visión alineada, y sirven para recordar lo hablado.
- **Historia técnica:** similar a la historia de usuario, pero en casos de funcionalidades que no son visibles directamente por el usuario final. Por ejemplo, refactorizaciones o implementación de medidas de seguridad.
- **Tarea:** unidades de trabajo más pequeñas que las dos anteriores. Son actividades que deben completarse para cumplir con una historia, como escribir código, diseñar interfaces, realizar pruebas…

Cuando una historia es demasiado grande y está muy poco definida se suele denominar «épica» o «*epic*». Las epics, como las historias y tareas, son requisitos ágiles, pero debido a su tamaño y falta de detalle no sirven como unidades de trabajo. Un ejemplo de epic puede ser, por ejemplo, desarrollar un nuevo sistema de gestión de tickets.

Enlace de interés: Cómo se crea una historia de usuario, Scrum Manager Podcast (→ «Referencias bibliográficas»).

Enlace de interés: 5 características de una buena historia de usuario, Scrum Manager Podcast (→ «Referencias bibliográficas»).

Cuando se empieza a trabajar usando scrum, es habitual descomponer las «historias» en tareas. Esto permite ver el avance día a día, lo cual puede ser especialmente útil en equipos con miembros con menos experiencia, por ejemplo para evitar entrar en puntos muertos.

Sin embargo, conforme se va ganando experiencia, descomponer cada historia en tareas puede resultar tedioso o incluso generar cierta parálisis. Si los desarrolladores son capaces de gestionar su trabajo durante el sprint directamente a nivel de historia, obligar a descomponer en tareas se vuelve una microgestión innecesaria. Usar unas unidades u otras depende del contexto.

Pila del producto: los requisitos del cliente

La «pila del producto» es el inventario de funcionalidades, mejoras, tecnología y corrección de errores que deben incorporarse al producto a través de los sucesivos

sprints. Representa todo aquello que esperan cliente, usuarios y demás partes interesadas. Todo lo que implique trabajo para el equipo debe estar reflejado aquí. Lo más común es referirse a las entradas de esta pila como «historias de usuario». Algunos ejemplos:

- «Ofrecer a los usuarios la consulta de archivos publicados por un determinado miembro de la plataforma».
- «Consultar los pedidos realizados por un vendedor en un rango de fechas.»
- «Ofrecer la consulta de un archivo a través de un API web.»

La característica esencial de este artefacto es que **contiene información viva**, en continua evolución, y que más que un documento de requisitos es una herramienta que facilita la comunicación de información al equipo. Al comenzar el proyecto la lista contiene unos pocos requisitos, aquellos conocidos y mejor entendidos en ese momento, porque se ampliará y modificará conforme avance el desarrollo. Este carácter dinámico permite que el producto se adapte a circunstancias cambiantes.

Se suele elaborar tras una reunión en la que el cliente comparte con el equipo una visión general del objetivo de negocio que persigue. Una vez que la pila del producto tiene historias suficientes para realizar un primer sprint es suficiente para empezar.

A partir de entonces el propietario del producto mantendrá las historias de la pila ordenadas según su prioridad. El nivel de urgencia vendrá marcado por lo necesaria y valiosa que sea cada funcionalidad.

Por otro lado, **el grado de concreción de las historias de usuario** deberá ser proporcional a su prioridad. Las más prioritarias deben estar lo bastante detalladas como para incluirse en el sprint. Acordar una DoR puede ayudar.

A la labor de priorizar, detallar y preestimar las historias, se le suele llamar **«refinado» o «preparación»**. Se trata de una tarea colaborativa; el propietario del producto y los desarrolladores pueden realizar una reunión de refinado en cualquier momento, sin que esto consuma nunca más del 10% de la capacidad de trabajo del equipo. Conviene recordar que las estimaciones, el esfuerzo previsible para cada elemento de la pila del sprint, son a juicio de los desarrolladores, ya que son quienes realizarán el trabajo (→ «Medición y estimación ágil»).

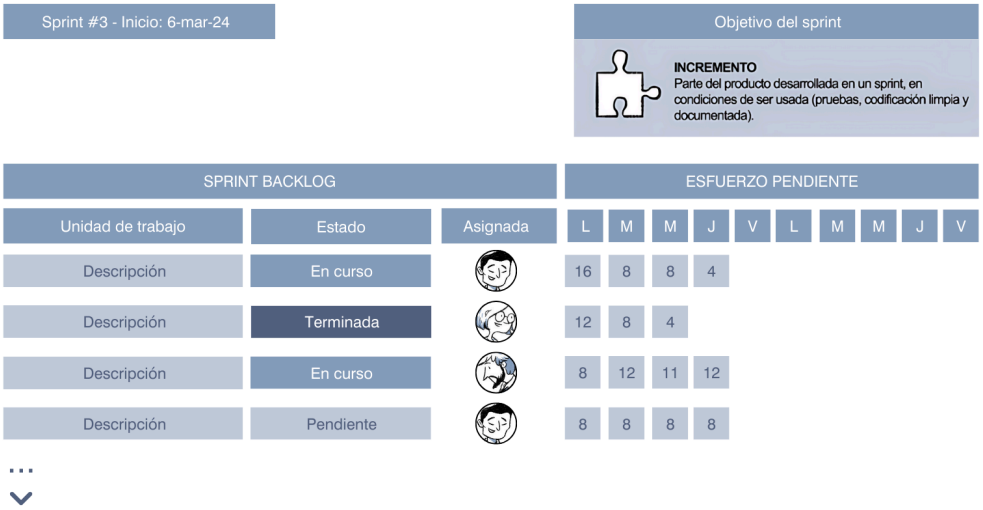
Las historias de usuario de la pila del producto que pueden ser incorporados a un sprint se denominan **«preparadas» o «ready»**. Se aplica este término (o similar) para indicar que el propietario del producto y el equipo de desarrollo están de acuerdo en que la historia está lista para ser seleccionada para el sprint. Es decir: está definida, preestimada, es asumible por sí misma en un único sprint, y se han establecido los criterios para considerarla terminada, así como la persona que los verificará.

Para confeccionar y mantener la pila de producto, lo ideal es emplear medios simples, conocidos y compartidos por todo el equipo. Es un radiador de información útil y

una herramienta que facilita la comunicación directa. Se pueden anotar las historias de usuario, por ejemplo, en etiquetas adhesivas sobre tableros, ordenándolas según su prioridad; o emplear una herramienta de gestión con la que todo el equipo esté familiarizado, tipo Trello.

El objetivo de la pila del producto es describir el estado que tendrá el producto en el futuro y que dibuja la visión compartida por el equipo para planificar.

Pila del sprint



La «pila del sprint» o «sprint backlog» es la lista de todas las unidades de trabajo (historias, tareas...) para completar el incremento. En ella las historias de usuario se descomponen en unidades de tamaño adecuado para monitorizar el avance a diario, así como para identificar riesgos y problemas sin procesos de gestión complejos.

Todo el equipo colabora en la confección de esta pila, durante la «reunión de planificación del sprint» (→ «Eventos»), indicando para cada unidad de trabajo el esfuerzo previsto para realizarla. Es habitual estimar este esfuerzo en «puntos» o «tiempo ideal» (→ «Medición y estimación ágil») empleando técnicas como la estimación de póquer (→ «Prácticas para flexibilizar scrum», «Estimación de póquer»).

Durante la reunión de planificación los desarrolladores pueden, si así lo deciden, dividir las historias en tareas más pequeñas. En tal caso se aconseja que una tarea no dure más de un día de trabajo.

Si la pila del producto es territorio del propietario del producto, **la pila del sprint es territorio de los desarrolladores.** Son los únicos que pueden modificarla durante el sprint.

Proporciona además comunicación visual directa y sobre ella el equipo de desarrollo revisa a diario el avance del sprint. Lo ideal es que se encuentre en un tablero

o pared en el mismo espacio físico donde se trabaja, para que sea visible para todos. Algunos soportes habituales son tableros físicos, hojas de cálculo compartidas, y herramientas colaborativas de gestión de proyectos tales como Todoist, Flow o Trello. Lo apropiado es utilizar el formato más cómodo para todos, teniendo en cuenta los siguientes criterios:

- Debe incluir sólo la información necesaria:
    - Meta del sprint.
    - Lista de historias o tareas.
    - Persona que se ha auto-asignado, en principio, esa unidad de trabajo.
    - Estado en el que se encuentra; cuánto queda para completarla.
- Debe servir de medio para registrar, en cada reunión diaria del sprint, el «esfuerzo» que le queda a cada unidad de trabajo.
- Debe facilitar la consulta y la comunicación diaria y directa del equipo.

La pila del sprint debe definir y estar alineada con el objetivo del sprint, que marca un hito en el avance hacia la visión del producto.

Incremento

El «incremento» es la parte de producto producida en un sprint y que se encuentra en condiciones de ser entregada al cliente; es decir: terminada, probada y operativa. No se deben considerar incrementos a prototipos, módulos o ni a partes pendientes de pruebas.

Idealmente, en scrum:

- Cada elemento de la pila del producto se refiere a funcionalidades entregables, no a trabajos internos del tipo «diseño de la base de datos».
- Se produce un incremento en cada iteración / sprint.

Sin embargo, el primer sprint suele ser una excepción. Se suele denominar «sprint cero» cuando tiene objetivos como «contrastar la plataforma y el diseño», que son necesarios al comenzar algunos proyectos. Implican trabajos de diseño o desarrollo de prototipos, para contrastar las herramientas o métodos de trabajo previstos.

Si la parte desarrollada requiere documentación, o procesos de validación y verificación documentados, éstos también tienen que estar realizados para considerar al incremento «terminado» o «done», es decir: entregable al cliente.

El objetivo del incremento es cumplir con las medidas de calidad que requiere el producto. La "definición de hecho", debe ser conocida y compartida por todo el equipo, y el incremento no se considera terminado hasta que no se alcanza el criterio de “hecho”.

Eventos

En este apartado se detallan las prácticas y actividades que constituyen la rutina de trabajo en scrum.

Sprint: es el núcleo fundamental de scrum, en torno al que se organizan todos los demás. A veces se llama también «iteración». Es el nombre que recibe cada etapa de trabajo con un objetivo concreto dentro del proyecto. La división del trabajo en sprints, que tienen una duración fija y constante (a esto se suele denominar «timeboxing») permite mantener el ritmo de avance.

Reunión de planificación del sprint: marca el inicio de cada sprint. En ella se determina cuál es el objetivo de éste y las tareas necesarias para conseguirlo.

Scrum diario: breve reunión diaria en la que el equipo hace punto de situación para confirmar que se está avanzando al ritmo adecuado, o si hay algún impedimento detectarlo y actuar en consecuencia lo antes posible.

El formato estándar es que cada miembro informe de lo realizado el día anterior, lo que tiene previsto hacer a continuación y si prevé algún impedimento.

Cada persona actualiza en la pila del sprint el tiempo o esfuerzo pendiente del trabajo que tiene asignado, y con esta información se actualiza a su vez el gráfico con el que el equipo monitoriza el avance del sprint: el gráfico de avance o burn down.

Revisión del sprint: análisis e inspección del «incremento» generado, y adaptación de la pila del producto si resulta necesario.

Retrospectiva del sprint: reunión al finalizar el sprint en la que el equipo analiza aspectos operativos de su forma de trabajo y crea un plan de mejoras, para aplicarlo en la siguiente iteración.

Un evento que no forma parte del marco tradicional de scrum pero ha ido ganando popularidad son las **reuniones de refinamiento del backlog**, que ya se ha mencionado al hablar de los artefactos y la estimación de unidades de trabajo. No hay consenso sobre cuándo hacer estas reuniones; se acaba resolviendo de una forma u otra en cada casuística. Parece frecuente que tengan lugar antes de la reunión de planificación del sprint. Realizar estas reuniones, sea cuando sea, permite llegar a las de planificación con historias ya detalladas y estimadas por los desarrolladores.

Enlace de interés: Eventos de scrum, Scrum Manager Podcast (→ «Referencias bibliográficas»).

Sprint

El evento clave de scrum para mantener un ritmo de avance continuo es el sprint: un periodo de tiempo acotado, cuya duración se recomienda que no exceda de 4 semanas, durante el que se construye un incremento del producto.

El incremento, como ya se vio en «Artefactos», debe estar terminado; esto es: operativo y útil para el cliente, en condiciones de ser desplegado o distribuido.

Al comenzar a trabajar con scrum es recomendable considerar el sprint como el evento contenedor de todos los demás:

- Marca el ritmo de avance diario y permite visualizarlo y compartirlo en las «reuniones de pie».
- Marca un ritmo fijo para comprobar el desarrollo del producto en las reuniones de planificación y revisión del sprint.
- A ese mismo ritmo se introducen las reuniones de retrospectiva, para reflexionar y mejorar.

En implementaciones más maduras de scrum, sin embargo, es posible considerar que el ámbito del sprint es sólo la construcción del incremento, dejando a un lado las reuniones.

Esto puede interesar, por ejemplo, para tener mayor flexibilidad al realizar sprints de duraciones diferentes; o para separar la frecuencia de las retrospectivas de la de los sprints.

Reunión de planificación del sprint

Esta reunión marca el inicio de cada sprint. En ella se toman como base las prioridades y necesidades de negocio del cliente y se determinan cuáles y cómo van a ser las funcionalidades que se incorporarán al producto al terminar el sprint.

Se trata de una reunión conducida por el scrum master (o, en su ausencia, un desarrollador) a la que deben asistir el propietario del producto y los desarrolladores, y en la que también pueden estar presentes otros implicados en el proyecto. Puede durar hasta una jornada de trabajo completa, según el volumen o complejidad de los elementos de la pila del producto (historias de usuario) que se desean incluir en el próximo incremento.

La reunión debe dar respuesta a tres cuestiones:

1. ¿Por qué es valioso este sprint?

El propietario del producto expone en qué forma el producto puede incrementar su valor con el resultado del sprint que se va a realizar. Esta cuestión determina cuál es el objetivo del sprint.

2. ¿Qué se puede hacer en el sprint?

Una vez compartido cuál es el incremento de valor que espera el propietario del producto, los desarrolladores determinan los elementos de la pila del producto que van a realizar. En este proceso pueden refinarse los elementos de la pila que puedan necesitar mayor concreción o explicación.

3. ¿Cómo se va a realizar el trabajo seleccionado?

Este punto puede ser opcional; es recomendable en equipos con menos experiencia. Puede ser beneficioso que los desarrolladores descompongan cada historia del sprint en tareas, de un día de trabajo o menos. Ayuda a ir generando una mejor intuición de lo que entrañan diferentes tipos de historia, y a visualizar el avance del sprint día a día.

Se recomienda articular la reunión en **dos partes de duración similar**, separadas por una pausa:

1. Qué se entregará al terminar el sprint.
2. Cómo se conseguirá el incremento, estimando con puntos de historia o como el equipo considere los requisitos necesarios.

Precondiciones	
La organización tiene determinados y asignados los recursos disponibles para llevar a cabo el sprint. Ya están «preparadas» las historias de usuario de mayor prioridad de la pila del producto, de forma que ya tienen un nivel de concreción suficiente y una estimación previa del trabajo que requieren. Los desarrolladores tienen conocimiento de las tecnologías empleadas y del negocio de productos suficientes para realizar estimaciones, y para comprender los conceptos del negocio que expone el propietario del producto.	
Entradas	**Resultados**
Pila del producto. Producto desarrollado en los incrementos anteriores (excepto en el sprint 0). Velocidad o rendimiento en el último sprint, como criterio para estimar la cantidad de trabajo. Circunstancias de negocio del cliente y del escenario tecnológico empleado y valor que espera obtener el propietario del producto.	Pila del sprint.

Primera mitad: ¿por qué es valioso este sprint y qué se puede hacer en él?

En esta primera mitad, el propietario del producto expone las historias de usuario de mayor prioridad, explicando qué se necesita y qué prevé que se podrá desarrollar en el siguiente sprint. Si la pila ha tenido cambios significativos desde la anterior reunión, explica las causas que los han ocasionado.

El objetivo es que todos comprendan, con un nivel de detalle suficiente, el incremento que se desea obtener con el sprint. La exposición debe estar abierta a preguntas y se pueden solicitar aclaraciones. Cualquier desarrollador puede proponer sugerencias, modificaciones y soluciones alternativas, y modificar la pila en consecuencia.

Esta reunión es un punto caliente de scrum para favorecer la fertilización cruzada de ideas y añadir valor a la visión del producto.

Tras reordenar y replantear las historias de la pila, el equipo define el «objetivo del sprint»: una frase que sintetiza cuál es el valor que se va a entregar al cliente. Exceptuando sprints dedicados a colecciones de historias desordenadas, la elaboración de este lema de forma conjunta en la reunión es una garantía de que todo el equipo comprende y comparte la finalidad del trabajo, y durante el sprint sirve de criterio de referencia en la toma de decisiones.

Segunda mitad: ¿cómo se conseguirá el incremento?

Esta segunda parte debe considerarse como una «reunión del equipo», en la que deben estar todos los desarrolladores y ser ellos quienes descompongan, estimen y asignen el trabajo. El papel del propietario del producto es atender a dudas y comprobar que se comprende y comparte el objetivo.

El equipo compone la pila del sprint y desglosa las historias que necesiten mayor nivel de detalle en unidades más pequeñas. Se establecen cuáles serán las historias o tareas prioritarias para los primeros días. La priorización sigue el orden marcado por el product owner; una vez establecido, los desarrolladores se asignan las unidades de trabajo que mejor se ajusten a su perfil respetando el orden de prioridad, procurando distribuir el trabajo de forma homogénea.

Funciones del scrum master durante la reunión de planificación del sprint

En caso de haber un scrum master asignado, éste será el moderador de la reunión para garantizar que:

1. Que con esta reunión arranque el sprint.
2. Asegurar que se cuenta con una pila del producto preparada por el propietario del producto.
3. Ayudar a mantener el diálogo entre el propietario del producto y los desarrolladores.

4. Asegurar que se llegue a un acuerdo entre el propietario del producto y los desarrolladores respecto a lo que incluirá el incremento.
5. Ayudar a comprender la visión y necesidades de negocio del cliente.
6. Asegurar que se ha realizado una descomposición y estimación del trabajo realistas, y ha considerado las posibles tareas necesarias de análisis, investigación o apoyo.
7. Asegurar que al final de la reunión están objetivamente determinados:
    - Los elementos de la pila del producto que se van a ejecutar.
    - El objetivo del sprint.
    - La pila del sprint con todas las unidades de trabajo estimadas, detalladas y asignadas.

Scrum diario

En scrum, el equipo monitoriza la evolución de cada sprint en **reuniones diarias muy breves**, coloquialmente llamadas ***dailies***. Reciben otros nombres como «reunión de pie» (*stand-up meeting*), «scrum diario» (*daily scrum*) o *morning rollcall*. **El tiempo máximo recomendado es de 15 minutos,** y aunque ahora es frecuente que también se hagan en remoto, en persona se aconseja hacerlas de pie. En cualquier caso, con acceso a un tablero o pizarra con información sobre el avance del sprint.

Durante estos minutos, los desarrolladores sincronizan el trabajo y establecen el plan para las 24 horas siguientes.

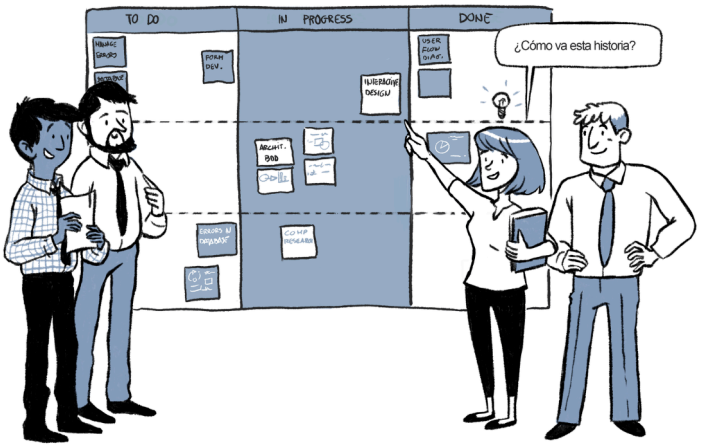
Entradas	Resultados
Pila del sprint y gráfico de avance con la información de la reunión anterior. Información del avance de cada desarrollador.	Pila del sprint y gráfico de avance actualizados.

Formato de la reunión

Se recomienda realizarla de pie junto a un tablero con la pila del sprint y el gráfico de avance o tablero kanban, para que todos puedan compartir la información y anotar.

En la reunión están presentes todos los desarrolladores, y pueden asistir también otras personas relacionadas con el proyecto o la organización, aunque éstas no intervienen.

Los desarrolladores conducen esta reunión, no el scrum master. No se trata de una reunión de control: es para que el equipo colabore, comparta el estado del trabajo y posibles dificultades. El papel del scrum master es facilitar, y encargarse de realizar las gestiones adecuadas para resolver estas últimas tras la reunión.



Es frecuente que cada desarrollador explique lo que ha logrado desde el anterior scrum diario, lo que va a hacer hasta el siguiente, y si está teniendo algún problema o si prevé que puede encontrar algún impedimento. Al final de la reunión el equipo de desarrollo refresca el gráfico de avance del sprint (→ «Prácticas para flexibilizar scrum», «Gráfico burn down») con las estimaciones actualizadas.

Sin embargo, se ha observado que este formato de participación individual puede tener efectos negativos en la motivación y las dinámicas de equipo. Aunque puede ser un punto de partida, se recomienda **evolucionar hacia un formato de reunión que se enfoque en el trabajo, no en las personas.** Este formato alternativo de la reunión sería así: los desarrolladores, que son quienes llevan la reunión, observan el estado de avance. Viendo qué hay en curso, se auto-asignan el trabajo aún pendiente siguiendo el orden de prioridad y asegurándose de que no se formen cuellos de botella. Este formato, que se centra en las historias o tareas en lugar de ir persona a persona, reduce la duración de las reuniones, aumenta la autonomía de los equipos, y mejora la colaboración. Por ejemplo, se ve con más frecuencia que varios desarrolladores decidan trabajar juntos en la misma historia prioritaria para garantizar que salga adelante si puede generar retrasos.

Revisión del sprint

Reunión de una o dos horas que se realiza al final del sprint para enseñar el incremento y recoger sugerencias. En caso de incrementos más relevantes o complejos puede durar hasta cuatro horas. Asiste todo el equipo scrum y todas las personas interesadas en el proyecto que lo deseen.

Esta reunión marca a intervalos regulares el ritmo de construcción y la trayectoria que va tomando la visión del producto, permitiendo que los interesados aporten sugerencias para seguir evolucionando. Los desarrolladores presentan las historias de usuario que planificaron, las que han desarrollado, las que no y los impedimentos

encontrados. Es importante para que las expectativas de los interesados sean realistas desde el principio de la reunión.

Entradas	Resultados
Incremento terminado.	Feedback para el propietario del producto: hito de seguimiento del avance del proyecto e información para mejorar su valor. Convocatoria de la reunión del siguiente sprint.

Formato de la reunión

Se trata de una reunión informal, en la que se muestra el resultado de la iteración, el incremento, en funcionamiento. Según las características del proyecto puede incluirse también documentación de usuario o técnica.

Más adelante, con la información obtenida, el propietario del producto tratará las posibles modificaciones que surjan.

Protocolo recomendado:

1. Los desarrolladores exponen el objetivo del sprint, la lista de funcionalidades que se incluían y las que se han desarrollado.
2. Hacen una introducción general del sprint y demuestran el funcionamiento de las partes construidas. Se abre un turno de preguntas y sugerencias. Esta parte genera información valiosa para que el propietario del producto y el equipo en general puedan mejorar la visión del producto.

Retrospectiva del sprint

Reunión que se realiza tras la revisión de cada sprint, antes de la reunión de planificación del siguiente. La duración recomendada es de una a tres horas. Es un ejemplo de práctica que no se encontraba en el marco original de scrum, pero que ha ido consolidándose con el tiempo.

En ella el equipo scrum reflexiona sobre su forma de trabajar. Se identifican fortalezas y puntos débiles, para afianzar las primeras y planificar acciones de mejora sobre los segundos.

El hecho de que se realice normalmente al final de cada sprint lleva a veces a considerarla erróneamente como una reunión de «revisión de sprint», cuando es aconsejable tratarlas por separado, porque sus objetivos son diferentes. El objetivo de la revisión del sprint es analizar «qué» se está construyendo, el producto, mientras que una reunión retrospectiva se centra en el marco de trabajo, el «cómo».

En la retrospectiva del sprint participa todo el equipo scrum, incluyendo al propietario de producto. Es importante que éste se considere «equipo» más que «cliente». Que la persona que desempeña el rol sea participativa y conocedora de los principios y valores de scrum. Si esto no fuera así, el scrum master debe actuar como facilitador para lograr su compromiso y participación. Aunque lo que se analiza en la retrospectiva es la manera de trabajar, la perspectiva del propietario de producto puede aportar mucho valor, ya que es responsable de mantener el foco en construir lo que el cliente necesita.

Medición y estimación ágil

En este apartado vamos a explicar los conceptos básicos sobre medición y estimación ágil que conviene conocer.

Dicho esto: cuantas menos métricas, mejor. El objetivo de scrum es producir el mayor valor posible de forma continua, así que cabe preguntarse siempre cómo contribuye el uso del indicador en el valor que se proporciona al cliente. Medir es costoso y debe servir a un propósito mayor, no convertirse en un fin en sí mismo.

Los objetivos que perseguimos con las herramientas de estimación que vamos a ver ahora son: poder planificar la duración de cada sprint de forma realista, marcar el ritmo de avance (sobre todo en equipos que están empezado a trabajar de forma ágil), sincronizar equipos y cerrar fechas de entrega.

Dos conceptos clave:

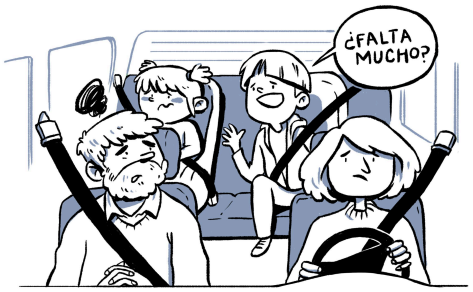
- No se mide el trabajo realizado, sino el que queda.
- Se mide empleando unidades relativas.

¿Cuánto queda?

Medir el trabajo puede ser necesario por dos razones: para registrar el que ya se ha hecho, o para estimar por adelantado el que se debe realizar. En ambos casos se necesita una unidad y un criterio objetivos de cuantificación.

Medir el trabajo ya realizado no entraña dificultad. Se puede hacer con unidades relativas al producto, como historias completadas; o a los recursos, como coste o tiempo de trabajo.

Pero la gestión de proyectos ágil no mide el trabajo ya hecho para calcular el avance del trabajo; es decir, restándolo del tiempo previsto. Por ejemplo: «Esto debía costar una semana. Como han pasado tres días, quedan cuatro para que esté terminado.» Esto, que suena lógico, en la realidad no se suele cumplir. Surgen imprevistos o se encuentran atajos que hacen que el tiempo estimado al principio no sea exacto. Teniendo esto en cuenta, no se determina el grado de avance por el trabajo realizado, sino por el que queda pendiente.



Esto lo entienden muy bien los niños. Cuando se sale de viaje, ¿qué es lo que preguntan una y otra vez? ¿Cuánto tiempo llevan en el coche? No, no, no... ¡Eso no les importa! La pregunta es: ¿cuánto queda?

Es posible que otros procesos de la organización necesiten registrar el esfuerzo invertido, pero calcular el avance del proyecto es diferente.

Scrum mide el trabajo pendiente, primero: para estimar el esfuerzo y tiempo previstos para realizar determinadas tareas, historias de usuario y «epics» (historias de gran tamaño). Y segundo: para determinar el grado de avance del proyecto, y en especial de cada sprint.

Enlace de interés: *Estimaciones ágiles, métricas de productividad, #Noestimates y otros animales* (→ «Referencias bibliográficas»).

Unidades relativas: puntos de historia

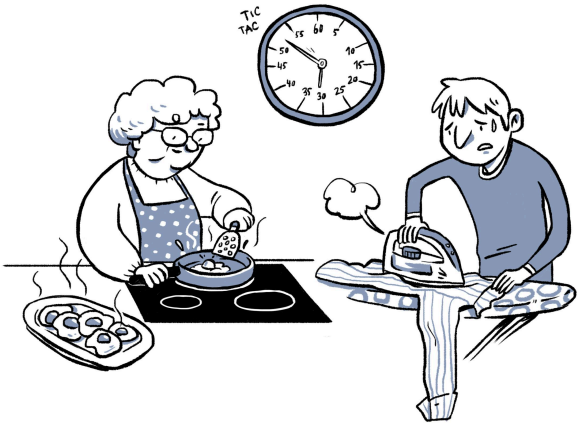
El trabajo necesario para completar un requisito o una historia de usuario no se puede prever de forma absoluta, porque rara vez tienen una única solución. En el caso de que se pudiera, por otra parte, la métrica sería demasiado pesada y compleja para la gestión ágil.

A la incertidumbre del trabajo se suman las inherentes al tiempo: no tiene sentido estimar la cantidad o calidad del trabajo de la «persona media» por unidad de tiempo, porque las diferencias de unas personas a otras son demasiado grandes. Es más: la misma tarea realizada por la misma persona requerirá diferentes tiempos según sus circunstancias.

Por todas estas razones, al estimar de forma ágil, se prefiere emplear unidades relativas. La unidad de trabajo más habitual es el «punto» o «punto de historia».

Cada organización, según sus circunstancias y criterio, institucionaliza su métrica de trabajo, su «punto». Es el tamaño relativo de tareas que se suele emplear. Es importante que el significado y la forma de aplicar la métrica sea siempre la misma en las mediciones de la organización, y que sea conocida por todos.

El tipo de «punto» dependerá de la organización. En un equipo de programación el punto puede ser equivalente a preparar una pantalla de login; para un equipo de diseño gráfico, la maquetación de un tríptico.



El «punto» ayuda, por un lado, a dimensionar una historia o tarea comparándola con una ya conocida, y por otro lado, a contrastar la dificultad que el trabajo concreto presenta para cada miembro del equipo según sus especialidades. Un ejemplo para ilustrar esto último podría ser el esfuerzo que cuesta freír un huevo. Si se estima cuántos «huevos fritos» costaría planchar una camisa, la respuesta dependerá de la persona. Alguien puede ser muy habilidoso friendo huevos, pero muy torpe para

planchar camisas, y estimará que eso le costaría «8 huevos fritos»; es decir, «8 puntos». Alguien muy acostumbrado a las tareas domésticas, en cambio, podría estimar la tarea en «un punto» o «un huevo frito».

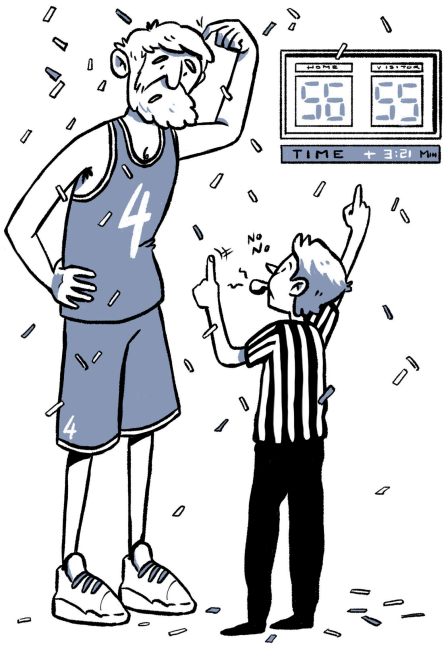
Ambos tienen razón: la cuestión es que la persona que estime sea la que va a realizar la tarea. Los «puntos de historia» suelen ser una unidad relativa o abstracta basada en algo con lo que el equipo esté muy familiarizado.

Por último, con esto se puede estimar la velocidad: en scrum, ésta es igual a la cantidad de trabajo realizado por el equipo en un sprint. Así, por ejemplo, una velocidad de 150 puntos indica que el equipo realiza 150 puntos de trabajo en cada sprint. No obstante, al salir del marco estándar de scrum podemos encontrar sprints de diferentes duraciones. Cuando esto sucede, se puede expresar la velocidad por unidad de tiempo en lugar de por sprint. Es decir: «la velocidad media del equipo es de x puntos por semana».

La práctica de medir velocidad con puntos de historia ha recibido críticas y merece consideración, pues un mal uso puede generar dinámicas no deseadas (→ *«Prácticas para flexibilizar scrum», «Estimación sin puntos de historia», «#NoEstimates»*).

Tiempo real y tiempo ideal

Cuando se calcula el calendario de un sprint tendemos a estimar el esfuerzo en «tiempo ideal»: tiempo de trabajo en condiciones ideales. Es lo que nos costaría realizar una tarea en un estado de flujo, concentrados y sin ninguna distracción o impedimento. Es importante ser conscientes de la diferencia por tanto entre el «tiempo ideal» y el «tiempo real» a la hora de estimar.

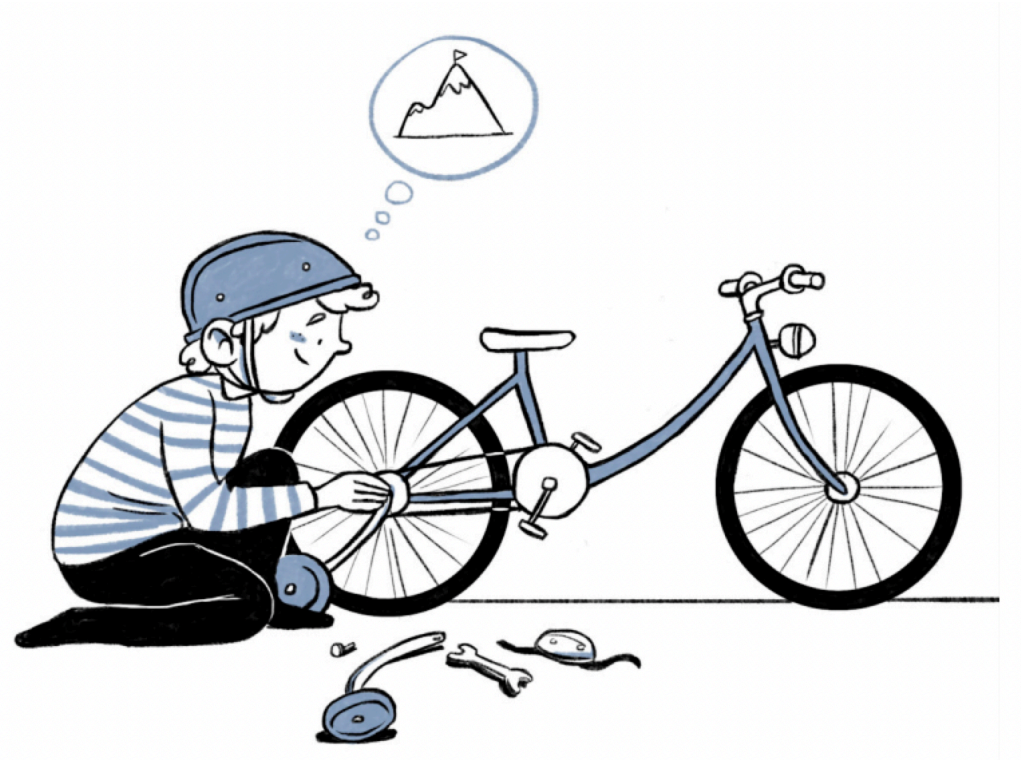


Si nos preguntan cuánto dura un partido de baloncesto, la respuesta en tiempo exacto de juego es 40 minutos, de igual forma que podemos decir que redactar un informe nos cuesta una hora. Pero el tiempo real de un partido de baloncesto suele ser de más de una hora, pues no puede terminar en empate. Se alarga por los tiempos muertos, las faltas, el entretiempo del partido y las prórrogas.

Todos sabemos que es normal que a veces un informe, algo que podría hacerse en una hora de «tiempo ideal», acabe ocupando media jornada de trabajo. Manejar estos dos conceptos de tiempo puede ayudarnos a organizar el trabajo de manera más objetiva y evitar el estrés de metas inalcanzables.

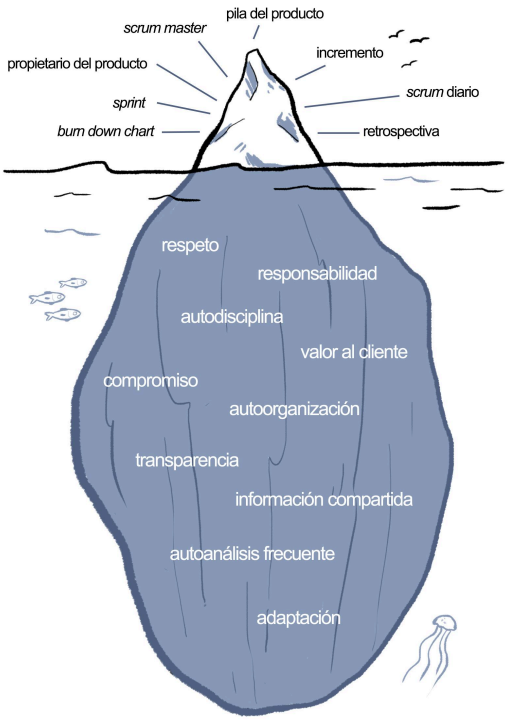
PARTE II: PRINCIPIOS Y VALORES

Interiorizando y adaptando



Principios y valores scrum

Hay algo que no se suele decir en los cursos de scrum: el marco que acabamos de estudiar, las prácticas y herramientas vistas en la primera parte, no funcionan sin unos principios y valores alineados con ellas.



¿Por qué? Porque las «prácticas» de scrum no son «procesos». No son pautas que garanticen resultados las ejecute quien las ejecute.

Scrum se asienta sobre el conocimiento tácito de las personas, así como sobre unos valores organizativos. Las prácticas son sólo las ramas del árbol y, sin unas buenas raíces, no darán frutos. Algo que conviene recordar de vez en cuando para no caer en el error de centrarnos sólo en las herramientas.

Los modelos de gestión facilitan el desarrollo de ciertos principios de trabajo y valores de la organización, sin los cuales trabajaremos en vacío.

Podemos encontrar empresas donde los principios están tan interiorizados que desarrollan su propio modelo de agilidad, con prácticas y técnicas diferentes a las que hemos visto, de otros modelos de agilidad o completamente originales, que dan como resultado productos innovadores y de calidad. Y empresas que, aplicando todas las prácticas del scrum estándar, sólo consiguen empleados alienados, desmotivados, estresados, y resultados mediocres.

Se necesita «flexibilidad» para adaptar lo aprendido a la realidad de cada empresa y proyecto. El objetivo es que la organización sea ágil en su conjunto, capaz de «avanzar en scrum», en escenarios de trabajo innovadores e inestables.

Para hacer una buena gestión, hay que estar al día de las últimas herramientas y después buscar la manera de ajustarlas al equipo, aprovechando lo que funcione y modificando o desechando lo que no; en lugar de pretender que las personas se adapten a métodos que no les convencen.

***

Los valores y principios pueden dividirse, aunque sea por pura semántica, en función de si están detrás de las prácticas ágiles o de la cultura de la organización. Los «principios» son el soporte de las prácticas; los «valores», de la cultura.

Principios

La finalidad de artefactos, eventos y otras técnicas ágiles es lograr que el trabajo se base en estos principios.

Entrega de valor

Entendiendo como tal a la entrega temprana y continua de valor al cliente, para lo que es necesario que éste colabore con el equipo y se comparta y comprenda su visión.

Mejora continua

En agilidad se reflexiona con frecuencia sobre los métodos de trabajo, cuestionando su efectividad y adaptándolos. El mismo esfuerzo autocrítico se aplica también a la mejora de los productos y servicios que se ofrecen.

Desarrollo iterativo e incremental

El producto final no se construye conforme a un plan inicial detallado y completo, sino que se arranca desde un «mínimo viable» sobre el que se van añadiendo incrementos.

Ritmo de trabajo sostenible

Alcanzar un ritmo de trabajo que evite la Ley de Parkinson (el trabajo se expande hasta llenar el tiempo disponible para que se termine) y la presión al descubrir los retrasos demasiado tarde.

Atención continua a la excelencia

Empleo de técnicas que garanticen la calidad de los productos y servicios y permitan detectar errores con antelación o en el momento de producirse.

Operativa visible

La información se comparte con claridad para facilitar la colaboración, identificar impedimentos de forma temprana y permitir que todo el equipo conozca el estado del producto y aporte ideas.

Cadencia y sincronización global

Este principio es más relevante cuando se intenta sincronizar a varios equipos que trabajan en productos o servicios relacionados. Se busca predecir la frecuencia de reuniones y fechas de entrega, por ejemplo sincronizando los sprints de los diferentes equipos.

Personas sobre procesos

La inteligencia colectiva del equipo, su conocimiento tácito, es responsable de la calidad del producto.

Valores

La cultura de la empresa es la suma de sus características organizacionales y de gobernanza, que pueden acelerar o frenar el desarrollo de la agilidad. En organizaciones cuya cultura se asienta en valores propios de trabajos industriales, basados en procesos, los principios ágiles despliegan resultados de inteligencia colectiva y valor innovador mucho más modestos que en organizaciones con valores que potencian los resultados de trabajar con principios ágiles:

- Asertividad.
- Valoración del talento.
- Claridad.
- Confianza.
- Estructura desjerarquizada.
- Propósito común.

Por último, es necesario el soporte directivo: que la gerencia de la empresa tenga una cultura afín, esté implicada y apoye a las personas con formación y recursos suficientes.

***

Nuestro objetivo es disponer de todas las herramientas para realizar las funciones de scrum master. Para ello vamos a conocer los «principios» asociados a los roles, artefactos y eventos de scrum. Por último, añadiremos algunas prácticas más que, aunque no forman parte del modelo de scrum estándar, se suelen utilizar.

Si se tiene interés en estudiar estos principios y valores ágiles en mayor profundidad, Scrum Manager® ha desarrollado un segundo manual que se encuentra disponible en *AgiLevel*: https://agilevel.com.

Las personas y sus roles

Aunque artefactos y eventos ayudan a desarrollar los principios ágiles, éstos tienen que ser asumidos primero por las personas. Sin personas con talento, comprometidas y conscientes de sus responsabilidades, las prácticas no servirán de nada. Y sin la influencia de ciertos valores culturales (→ AgiLevel), no se puede desarrollar el talento ni el compromiso.

> «Si tienes un equipo de ingenieros brillantes, que usan excelentes herramientas y prácticas de ingeniería, que comprenden de arriba abajo el ámbito tecnológico y del negocio, a los que no se les interrumpe y tienen los recursos tecnológicos que necesitan, ¡entonces puedes usar scrum! Es cierto que las personas así pueden construir un incremento de software en cada iteración. ¡Eso está bien!.
>
> Pero scrum también funciona con idiotas.

> Puedes tener un grupo de idiotas que no hayan pisado una escuela, que no sepan ni de informática, ni técnicas de ingeniería de software, que se odien entre ellos, que no entiendan ni del negocio ni de las herramientas de ingeniería, y regularmente producirán mierda en cada incremento. ¡Eso está bien! Quieres saber al final de cada iteración dónde estás.» (Schwaber 2006)

Propietario del producto

El propietario del producto o *product owner* es el responsable de la pila del producto; quien la gestiona y prioriza los requisitos, con actitud proactiva y realizando cambios cuando lo estime necesario.

El propietario del producto es quien hace posible la entrega temprana y continua de valor. Representa al cliente y es responsable de comunicar su visión de forma clara al equipo, para que todos se alineen hacia el mismo objetivo.

Su actitud es clave para facilitar una comunicación honesta y fluida.

Desarrolladores

El equipo de desarrolladores es responsable de producir un incremento con cada iteración, así como de mantener un ritmo de trabajo sostenible. Comprometidos con su trabajo, deben aplicar una mirada crítica para mejorar sus resultados y sus métodos. Por último, deben ser conscientes de la importancia que tienen su talento e inteligencia colectiva, no sólo a nivel individual.

Scrum master

Es quien garantiza que el marco scrum funcione, moderando las reuniones de scrum diarias y gestionando la resolución de impedimentos identificados en éstas, para mantener el avance del equipo.

Asignar la responsabilidad del funcionamiento de scrum a un scrum master es más aconsejable para equipos con poca experiencia o en organizaciones grandes, con un flujo continuo de rotación o formación de personal. En equipos pequeños, estables y con niveles de agilidad consolidados, las responsabilidades de funcionamiento y mejora del marco de scrum pueden estar ya interiorizadas y asumidas por el equipo en conjunto.

Artefactos

Los artefactos facilitan el desarrollo de los principios ágiles.

Pila del producto

Entrega de valor: permite asumir la variabilidad del entorno de negocio del cliente y centrar los esfuerzos en aquellas historias que aportan mayor valor según las circunstancias.

Desarrollo iterativo e incremental: esta pila, a diferencia de un documento de requisitos cerrado, hace posible el principio de desarrollo iterativo ya que es un documento vivo, que permite que haya cambios en las historias que contiene y en su prioridad.

Operativa visible: es un radiador de información mediante el que el propietario del producto y el equipo comparten la visión del producto en todo momento.

Pila del sprint

Desarrollo iterativo e incremental: es el artefacto que delimita el trabajo de un incremento (sprint) y sirve para marcar el pulso de avance.

Operativa visible: es una herramienta de comunicación interna para el equipo, a la que todos tienen acceso y que permite conocer el estado del sprint con un golpe de vista.

Incremento

Entrega de valor: la presentación de una parte del producto terminada y lista para usarse al final de cada sprint permite comprobar si se está generando valor.

Desarrollo iterativo e incremental: la entrega de los incrementos ayuda a marcar el ritmo de avance.

Eventos

Sprint

Mejora continua: el ritmo de avance en iteraciones breves facilita identificar hitos en los que pararse a reflexionar sobre cómo mejorar la calidad del producto y de los sistemas de trabajo.

Desarrollo iterativo e incremental: es la unidad básica de tiempo durante la que se construye cada incremento, por tanto el engranaje en torno al que gira todo el desarrollo.

Ritmo de trabajo sostenible: marca el pulso de avance.

Operativa visible: permite la identificación temprana de impedimentos.

Cadencia y sincronización global: marca la cadencia de las entregas mediante timeboxing. Permite predecir la frecuencia de reuniones y de fechas de entrega, así como sincronizar el trabajo de diferentes equipos.

Reunión de planificación del sprint

Entrega de valor: durante esta reunión el equipo y el product owner colaboran de forma directa, profundizando en el conocimiento compartido de la visión.

Scrum diario

Operativa visible: en esta reunión el equipo se ubica, comparte el estado en el que se encuentra su trabajo y colabora, aportando ideas y resolviendo los impedimentos que haya.

Revisión del sprint

Entrega de valor: en este evento el equipo vuelve a colaborar directamente con el propietario del producto.

Mejora continua: el propósito de la reunión es analizar el incremento generado, para sacar conclusiones que ayuden a perfilar el siguiente sprint.

Retrospectiva del sprint

Mejora continua: se analiza no el incremento, sino los métodos de trabajo empleados por el equipo, para decidir qué mantener y qué modificar o eliminar.

Prácticas para flexibilizar scrum

La comunidad profesional aporta «prácticas» para dar forma a pilas de producto, historias de usuario, representar con imágenes la visión del producto, conducir eventos y reuniones, y estimar unidades de trabajo. Las dos más empleadas en la formación inicial del marco scrum son el gráfico burn down y la estimación de póquer. Vamos a verlas junto con algunas otras:

- **Control de avance:**
  - Gráfico de avance o burn down chart.
  - Gráfico de producto o burn up chart.
  - Kanban.
- **Estimación:**
  - Estimación de póquer.
  - Estimación en la pared.
  - Estimación sin puntos de historia.
  - #Noestimates.
- **Métodos de trabajo:**
  - Diagramas para retrospectivas: espina de pez y árbol.
  - Técnicas a prueba de errores.
  - Trabajo en pareja.

Para nuestro propósito no nos interesa entrar a explicar la operativa de estas prácticas en profundidad, sino presentarlas como ejemplos y animar al lector a investigar y experimentar. Son sólo algunas de las técnicas de gestión ágil que pueden ayudar a personalizar el modelo de gestión de un proyecto o equipo.

Control de avance

Gráfico de avance o burn down chart

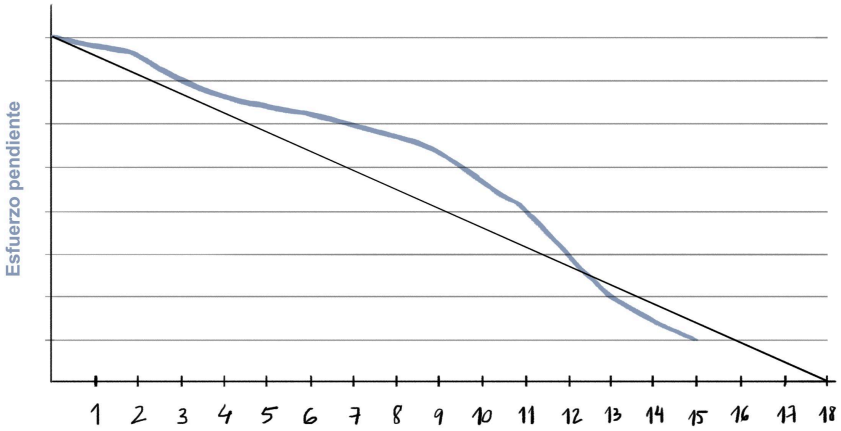
También llamado «gráfico de avance». Los desarrolladores lo actualizan, durante el sprint, a ser posible a diario, para monitorizar el ritmo de avance. Es útil para la detección temprana de desviaciones que puedan comprometer la entrega al final del sprint.

- En el eje Y representa los puntos de trabajo que aún faltan por realizar.
- En el eje X representa los días del sprint.

Día a día, cada desarrollador estima en la pila del sprint el esfuerzo pendiente para cada unidad de trabajo, hasta que se termina. Con esa información se actualiza el gráfico, reflejando cada día el esfuerzo total pendiente.

Cuando se trabaja con historias de usuario en lugar de tareas (que, recordemos, deberían ser de un día de duración o menos) el gráfico de avance se puede actualizar por historia acabada, en lugar de por esfuerzo estimado pendiente en cada historia día a día.

La historia se convierte en la medida de avance principal, y se restan todos los puntos estimados para cada historia una sola vez, cuando ésta se completa.



El avance ideal de un sprint estaría representado por la diagonal que reduce el esfuerzo pendiente de forma continua y gradual hasta el último día previsto. Como es de suponer, esto no es lo habitual. Se puede llevar un patrón de avance adecuado sin que la diagonal del gráfico sea perfecta.

Si la línea de avance se mantiene durante varios días muy por encima de la diagonal, es señal de que se ha subestimado el sprint y de que éste requerirá más tiempo. Cuando ocurre lo contrario y la línea desciende más deprisa que la diagonal, se terminará antes de lo previsto.

Vídeo explicativo: Gráfico de avance "Burn down chart" (→ «Referencias bibliográficas»).

Gráfico de producto o burn up chart

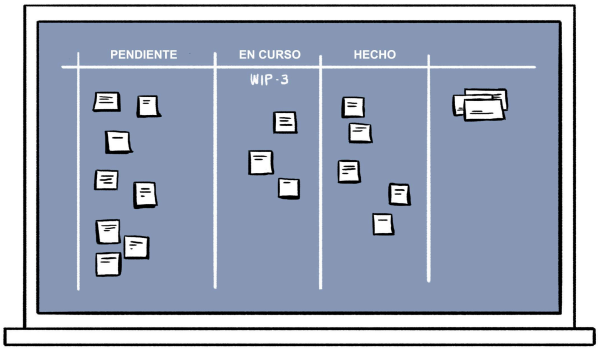
El gráfico de producto o burn up es una herramienta de planificación propia del propietario del producto. Representa la evolución previsible en función de la velocidad del equipo. Se suelen hacer tres estimaciones: pesimista, realista y optimista.

La proyección se realiza sobre un diagrama cartesiano que representa, en el eje de ordenadas, el esfuerzo estimado para construir las diferentes historias de la pila del producto, y en el de las abscisas, el tiempo medido en sprints. Para trazar la previsión se sitúa cada versión en el eje vertical en la posición correspondiente al esfuerzo estimado para construir todas las historias que incluye.

Vídeo explicativo: Gráfico de producto "Burn up chart" (→ «Referencias bibliográficas»).

Se trata de una herramienta para desarrollo ágil, un documento vivo que no debe utilizarse para representar planes estables, sino las previsiones tras cada evolución de la pila del producto.

Kanban



Kanban es una popular técnica para gestionar, de forma visual, un flujo continuo de avance; es decir, un flujo sin timeboxing, sin dividir el trabajo en sprints de duración predeterminada.

El equipo anota las historias o tareas y las posiciona sobre un tablero. Su ubicación determina su estado. Los más habituales en tableros kanban son los que dictan el orden de avance: «pendiente», «en curso» y «terminado». Se ordenan progresivamente de izquierda a derecha. El formato de cada tablero responde a las circunstancias del producto y del equipo. Puede incluir estados adicionales como «testeado» o «validado», por ejemplo.

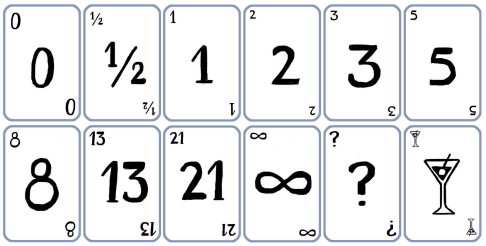
El tablero no sólo ayuda a gestionar de manera clara y visual y a mantener el ritmo, sino que también es una herramienta útil para compartir información entre el equipo. Muestra de inmediato, con cada actualización, el estado de cada unidad de trabajo y si se están formando cuellos de botella.

La ausencia de hitos temporales como los sprints evita la ley de Parkinson. Por el contrario, esta ausencia podría provocar retrasos, por perfeccionismo o por procrastinación. Pero gracias al WIP *(work in process)*, la estructura y la visibilidad que ofrece el tablero kanban, se consigue paliar este efecto negativo también. El WIP es el número máximo de unidades de trabajo que pueden estar a la vez en la misma fase. Un WIP=3 de la fase «en curso» indica que el equipo no puede trabajar en más de 3 historias o tareas de forma simultánea.

La visibilidad que ofrece el tablero permite identificar a todo el equipo de forma temprana cuellos de botella y tiempos muertos, para ajustar o replantear prioridades. Con la información que se obtiene se pueden sugerir mejoras de flujo y de distribución de los miembros de los equipos.

Estimación

Estimación de póquer



Jane Grenning ideó este juego de planificación que se utiliza para conducir las reuniones en las que se estima el esfuerzo y la duración de las unidades de trabajo (planificación del sprint). El modelo inicial de Grenning consta de 8 cartas, con los valores ½, 1, 2, 3, 5, 6, 7 e infinito.

Cada participante dispone de un juego de cartas y, en la estimación de cada unidad, muestra la combinación que suma el esfuerzo estimado.

El uso más extendido, sin embargo, es empleando una baraja con números en sucesión de Fibonacci, como en la ilustración. En este caso no se combinan números, sólo se muestra una carta para cada estimación: aquella con la cifra más aproximada. Esta variante se basa en que, al aumentar el tamaño de la tarea o historia, aumenta también el margen de error.

Así, por ejemplo, si una persona cree que el tamaño adecuado de una historia es 6, se ve obligado a reconsiderar. Puede aceptar que parte de la incertidumbre apreciada no es tal y levantar la carta de 5, o aceptar una estimación más conservadora y levantar el 8.

Es frecuente añadir una carta con un símbolo de duda o interrogación para indicar que, por las razones que sean, no se puede precisar una estimación. También es posible incluir otra carta con alguna imagen alusiva para indicar que se necesita un descanso. El símbolo de infinito se saca cuando la historia excede el máximo esfuerzo estimable, para indicar que debería dividirse en unidades más pequeñas.

Cuando las estimaciones son muy dispares, el moderador de la reunión puede optar por varias alternativas. Se puede preguntar a las personas con las estimaciones más extremas el por qué de su estimación y repetir el proceso. Otra opción es dejar de lado la unidad a estimar por el momento, y estimarla de nuevo más tarde. O pedir al propietario del producto que la descomponga para valorar cada sub-unidad resultante. También puede tomar la decisión de optar por la estimación más optimista, más pesimista, o sacar la media. Dependerá del trabajo a estimar y del estilo de gestión de cada equipo.

El uso de la estimación de póquer, además de ser divertido y dinamizar las reuniones, evita las habituales discusiones circulares entre diversas opciones de implementación. Permite que todos los asistentes participen, ayuda a alcanzar consensos sin discusiones y a reducir el tiempo de estimación de cada funcionalidad.

Enlace de interés: Serie Fibonacci, Scrum Manager Podcast (→ «Referencias bibliográficas»).

Estimación en la pared

Técnica empleada para estimar y priorizar una lista de historias de usuario, normalmente la pila del producto. Se basa en gestión visual: se sitúan sobre una pared notas adhesivas con las diferentes historias. Los desarrolladores determinan la posición horizontal de cada nota según el tamaño de la historia: a la izquierda las menores, a la derecha las mayores. El propietario del producto determina la posición vertical, en función de la prioridad: a más alta, mayor prioridad.

Vídeo explicativo: Estimación en la pared (→ «Referencias bibliográficas»).

Estimación sin puntos de historia

La estimación con puntos de historia ha recibido críticas por la tendencia a malinterpretar su función. Cuando se usan para medir la velocidad de varios equipos en una empresa, por ejemplo, pueden causar tensiones. Los puntos de historia no son una buena medida de productividad: son relativos y dependen del contexto de cada equipo. Un número mayor de puntos no implica mayor esfuerzo ni calidad, pero esto no es necesariamente intuitivo. Los equipos pueden empezar a inflar estimaciones y a mover historias a la pila del sprint no por prioridad sino para acumular puntos. Bien para mantener la que consideran su velocidad media o para competir con otros. Todo esto funciona en detrimento del proyecto e inhibe la colaboración.

Hay alternativas para poder estimar evitando estos problemas:

- **Tallas de camiseta:** un sistema de estimación intuitivo y que mantiene cierto nivel de detalle, en el que se indica en la historia si su tamaño es XS, S, M, L, o XL.
- **Ricitos de Oro:** usar sólo tres medidas para las historias: demasiado grande, demasiado pequeño, o tamaño adecuado.

Ambos requieren tomar medidas alternativas para sincronizar el ritmo de trabajo entre equipos e identificar cuellos de botella y otros impedimentos con antelación. Pero permiten priorizar de manera intuitiva y detectar historias que requieren más análisis antes de pasarlas a producción.

#NoEstimates

Ésta es otra de las reacciones al mal uso de las estimaciones en agilidad: renegar por completo de ellas. En algunas ocasiones, en equipos de alto rendimiento acostumbrados a trabajar de forma ágil y que mantienen un flujo de trabajo continuo, puede tener sentido no estimar. Ayuda por ejemplo a prevenir la Ley de Parkinson, que dice que «el trabajo se expande hasta llenar el tiempo disponible para que se termine».

Al igual que ocurría con los sistemas de estimación alternativos a los puntos de historia, esta manera de trabajo supone encontrar otras maneras de sincronizar entregas.

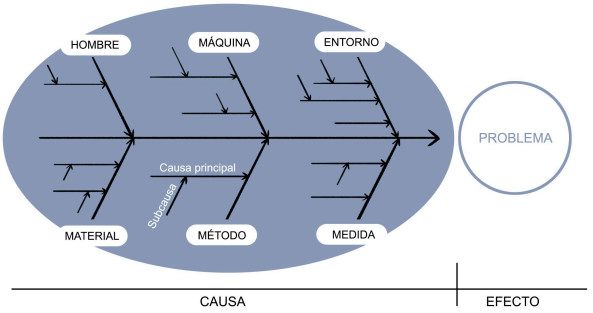
Enlace de interés: Estimar o no estimar, Scrum Manager Podcast (→ «Referencias bibliográficas»).

Métodos de trabajo

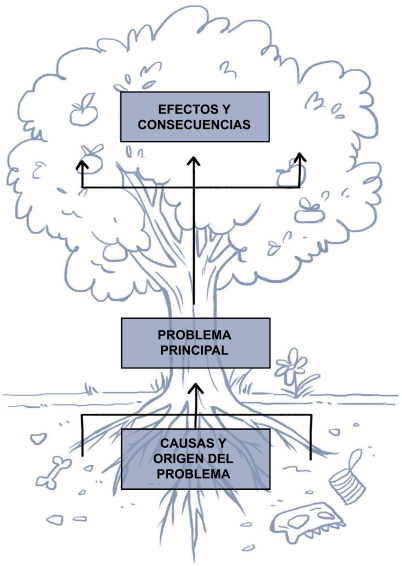
Diagramas para retrospectivas: espina de pez y árbol

El **diagrama de espina** también es llamado a veces «diagrama de Ishikawa», «de cola de pescado», «de causa-efecto», o «de Grandal». Representa las relaciones causa-efecto que actúan sobre un problema a analizar.

Tras identificar el problema, que es el eje central del diagrama (una línea horizontal central), se enumeran en torno a él posibles causas que lo expliquen, que pueden tener «sub-causas». Suele emplearse en gestión de calidad.



***



Los **diagramas de árbol** se suelen utilizar para alcanzar conclusiones positivas desde el análisis de una situación problemática. Se emplean los elementos del árbol (raíces, tronco, ramas, frutos...) para representar los medios o recursos mediante los que solucionar un problema y los posibles resultados. No existe un único tipo de diagrama de árbol ni un sistema estándar, pero como ejemplo sugerimos consultar cómo lo desarrolla Khurram Bhatti.

Técnicas a prueba de errores

Consiste en aplicar modos de trabajo que impiden la producción de errores. En el mundo de la programación ágil es habitual el desarrollo guiado por pruebas o TDD (*test driven development*), que consiste en desarrollar primero las pruebas (*tests*) que el código debe pasar y después el código.

En términos más generales, la agilidad emplea técnicas *poka-yoke* y dispositivos de control *andon.* Ambos conceptos provienen de los marcos de producción de manufactura *lean*.

Las técnicas a prueba de fallos poka-yoke que pueden ser para:

- **Hacer imposible el error humano:** un ejemplo son los enchufes diseñados para que no puedan ser acoplados de forma errónea.
- **Resaltar el error de forma evidente cuando se produce:** como hacen los correctores ortográficos de los editores de texto, o los correctores sintácticos de programación.

Los sistemas de control andon son propios del proceso de producción. Suelen consistir en indicadores con luces de diferente color o representaciones gráficas que reflejan el funcionamiento normal del sistema o posibles fallos. Están situados en el propio entorno de trabajo para alertar de forma inmediata y visual al equipo.

Trabajo en pareja

Este concepto está más identificado en equipos de programación, donde se conoce como *pair programming*.

Consiste en asignar a dos personas la ejecución de una misma tarea, de forma simultánea y conjunta, normalmente alternando entre ellos la ejecución y supervisión. Mientras uno la lleva a cabo, el otro está atento a lo que hace, realizando las observaciones que puedan ser procedentes. Esta práctica puede ser adecuada cuando la calidad del resultado depende, sobre todo, del conocimiento de la persona que lo realiza.

Por esta razón, por ejemplo, los trenes circulan con ayudante y maquinista cuando la tecnología empleada no puede evitar al 100% el fallo humano, o nos sentimos más seguros sabiendo que en la cabina del avión hay dos pilotos trabajando en pareja.

APÉNDICE

Información, recursos y comunidad profesional

- Última versión:
  - https://scrummanager.com/website/c/info/docs-media.php
- Plataforma eLearning, Open Knowledge:
  - https://scrummanager.com/open-knowledge/
- Blog:
  - https://www.scrummanager.com/blog/
- Acerca de la certificación Scrum Manager®:
  - https://scrummanager.com/website/c/info/academic-framework.php
- Preguntas frecuentes sobre cursos y exámenes Scrum Manager®:
  - https://scrummanager.com/website/c/info/faqs.php

Redes sociales y profesionales

Twitter	https://twitter.com/scrummanager
LinkedIn	https://www.linkedin.com/company/scrum-manager
Spotify	https://acortar.link/c8jESa
Ivoox	https://acortar.link/qRRe4L
Youtube	https://www.youtube.com/@scrummanager/featured
Agilidad on the go	https://scrummanager.com/open-knowledge/

Mejora continua y control de calidad

El control de calidad de Scrum Manager se basa en las valoraciones de sus estudiantes. Con tu opinión nos ayudas a mantener el nivel de nuestros materiales, cursos, centros y profesores.

Si has participado en una actividad de formación auditada por Scrum Manager®, te rogamos y agradecemos que valores la calidad de la formación recibida. La información que se recoge es anónima. Puedes enviarnos tus comentarios desde el «Área de miembros»: https://scrummanager.com

Referencias bibliográficas

Bauer, F., Bolliet, L., & Helms, H. (1969). *Software Engineering. Report on a conference sponsored by the NATO*.

Beck (1999). *Extreme Programming Explained*.

Bhatti, Khurram (2019). "The tree retrospective model", *Khurrambhatti.com*: https://khurrambhatti.com/2019/03/08/tree-retrospective-model/

Nonaka,I., & Takeuchi, I.

- (1986). *The New New Product Development Game.*
- (1995). *The Knowledge-Creating Company.*
- (2004). *Hitotsubashi on Knowledge Management.*

Orr, Ken (2002). *CMM versus Agile Development: Religious wars and software development.*

Palacio, Juan (2019). "Estimaciones ágiles, métricas de productividad, #NoEstimates y otros animales", *Scrum Manager Blog*: https://www.scrummanager.com/blog/2019/12/estimaciones-agiles-metricas-de-productividad-noestimates-y-otros-animales/

Schwaber, Ken,

- (1995). *SCRUM Development Process- OOPSLA 95.*
- (2006). *Scrum Et Al.*

Scrum Manager Podcast,

- (2022). *Serie Fibonacci y estimación ágil | Scrum Manager Podcast 1x01:* https://open.spotify.com/episode/5cSiawMjvd1TFUpt0TPsEi?si=D7Qqcq4CSeW60pw04yXP5A
- (2023). *Cómo crear un backlog | Scrum Manager Podcast 1x02:* https://open.spotify.com/episode/7hGoEqV2uZL47uZZN6GXfk?si=2eb790bebe644aa8
- (2023). *Cómo se crea una historia de usuario | Scrum Manager Podcast 1x03:* https://open.spotify.com/episode/07cSgNf9znGHU93kAm2qaU?si=9813747ed602458b
- (2023). *Eventos de scrum | Scrum Manager Podcast 1x04:* https://open.spotify.com/episode/7basfleLN9L5A6FMOGHAf3?si=7d5e6974f32f49af
- (2023). *Definition of Done | Scrum Manager Podcast 1x06:* https://open.spotify.com/episode/0GjNQprSs7wbQ8s6G0lGXu?si=2d94f9fd7d1f4770
- (2023). *Estimar o no estimar | Scrum Manager Podcast 1x10:* https://open.spotify.com/episode/5qa41nYkK9JSmKVg0E5qt0?si=14aafb0778a44d09
- (2024). *Las 5 claves de un buen product owner | Scrum Manager Podcast 2x02:* https://open.spotify.com/episode/5OwC380a6pWE1iV59Mchqo?si=80e27682603a42b4
- (2024). *5 características de una buena historia de usuario | Scrum Manager Podcast 2x07:*

https://open.spotify.com/episode/5PMBIIhu0cuW9GkA2xNGV5?si=6e579095f4e542d5

Scrum Manager Youtube,

- (2017). *Gráfico de producto burn up*: https://www.youtube.com/watch?v=jdyodO-uFBk
- (2017). *Objetivo de la gestión predictiva y de la gestión ágil:* https://www.youtube.com/watch?v=57pcynW3pQo
- (2018). *Gestión predictiva y evolutiva*: https://www.youtube.com/watch?v=vQrFunXuURc
- (2021). *Gráfico de avance "Burn down chart"*: https://www.youtube.com/watch?v=hveuhx7rZqw
- (2021). Estimación en la pared: https://www.youtube.com/watch?v=4O2BRg9kDU4
- (2021). *Scrum (es):* https://www.youtube.com/watch?v=SlZfzLRGYBA

Turner & Jain (2002). *Agile Meets CMMI: Culture Clash or Common Cause?*