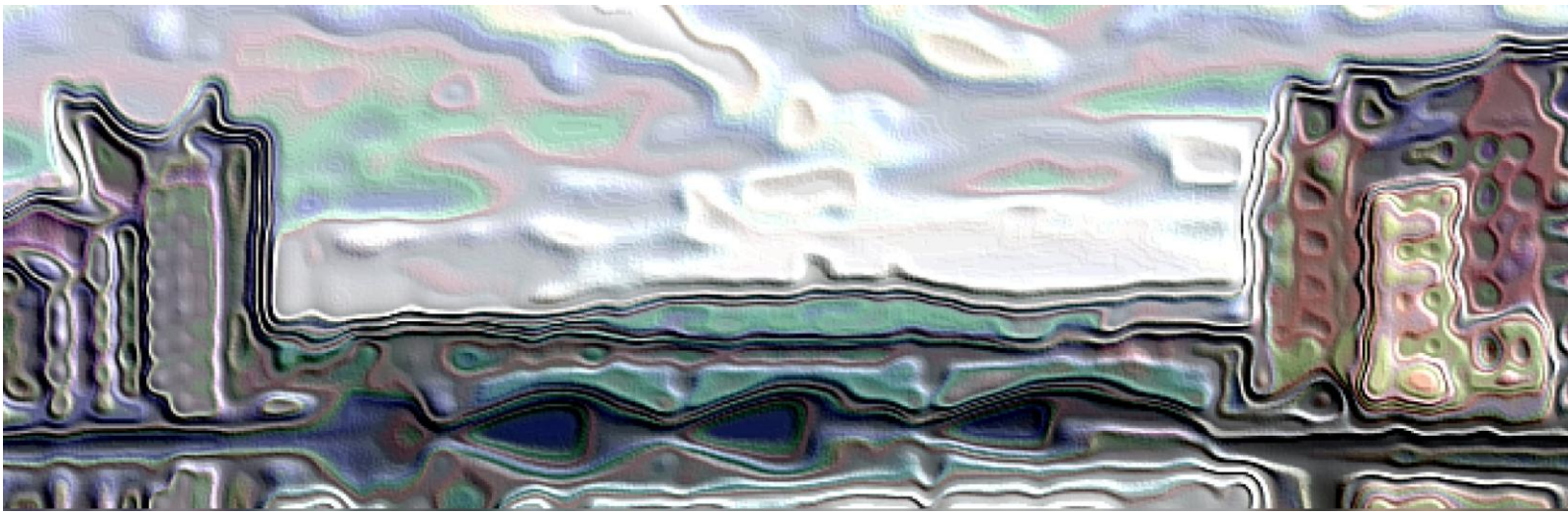
Scrum Manager®



Gestión de proyectos Scrum Manager

v. 2.5.1

Gestión de proyectos Scrum Manager

(Scrum Manager I y II)

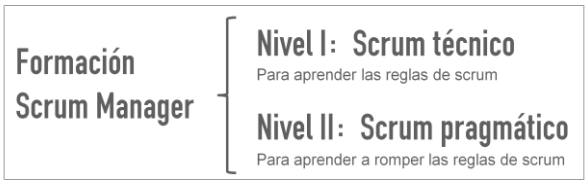
Versión 2.5.1 – Abril 2015

Diseño de cubierta: Scrum Manager.
Imagen derivada de la original: The Albert Bridge 04 – Belfast de William Murphy (http://www.streetsofdublin.com/)

Contenido

Formación Scrum Manager

Este libro cubre los niveles I y II del conocimiento troncal de formación Scrum Manager®



Los contenidos de formación se mantienen regularmente actualizados. Puede descargar la última versión, o consultarla en línea en la dirección: http://www.scrummanager.net/bok

Este documento es un ***recurso educativo abierto (OER)*** y puede emplearse gratuitamente para consulta y autoformación.

No está permitido su uso para actividades comerciales.

Los recursos educativos abiertos de Scrum Manager son posibles gracias al soporte de los:

Servicios de formación y asesoría Scrum Manager

- ***Cursos*** en convocatorias presenciales, o a medida para empresas o grupos.
  - Puede consultar el calendario de convocatorias en diferentes ciudades en la página de cursos de http://www.scrummanager.net:
  - Para información de cursos a medida: contacte con un Centro Oficial de Formación Scrum Manager (http://scrummanager.net/centros) o en la dirección admin@scrummanager.net

- ***Exámenes presenciales de certificación.*** con supervisión de la ejecución de la prueba, e identificación del alumno
  - Incluidos en los cursos presenciales.
  - Examen de certificación sin curso: en centros examinadores.

Más información:

http://www.scrummanager.net – (preguntas frecuentes)
http://www.scrummanager.net/oks
admin@scrummanager.net

Mejora continua y control de calidad Scrum Manager

Gracias por elegir los servicios de formación de Scrum Manager

Su valoración es el criterio del control de calidad de Scrum Manager, y decide la validez o no de los servicios de formación, y en su caso la continuidad de los cursos, centros y profesores.

Si ha participado en una actividad de formación auditada por Scrum Manager, le rogamos y agradecemos que valore la calidad del material, profesor, temario, etc. así como tus comentarios y sugerencias.

Scrum Manager anonimiza la información recibida, de forma que comparte con los profesores, y centros autorizados las valoraciones y aspectos de mejora, pero en ningún caso los nombres de los alumnos que las han realizado.

Puede realizar la valoración en la página accesible a miembros de Scrum Manager: http://scrummanager.net/qa.

PRIMERA PARTE

Las reglas de scrum.

Agilidad

El entorno de trabajo de las empresas del conocimiento se parece muy poco al que originó la gestión de proyectos predictiva. Ahora se necesitan estrategias para el lanzamiento de productos orientadas a la entrega temprana de resultados tangibles, y a la respuesta ágil y flexible, necesaria para trabajar en mercados de evolución rápida.

Ahora se construye el producto mientras se modifican y aparecen nuevos requisitos. El cliente parte de una visión medianamente clara, pero el nivel de innovación que requiere, y la velocidad a la que se mueve su sector de negocio, no le permiten predecir con detalle cómo será el resultado final.

Quizá ya no hay “productos finales”, sino productos en continua evolución y mejora.

> La gestión de proyectos ágil no se formula sobre la necesidad de anticipación, sino sobre la de adaptación continua.

¿La gestión de proyectos predictiva es la única posible? ¿Los criterios para determinar el éxito son siempre el cumplimiento de fechas y costos? ¿Puede haber proyectos cuya gestión no busque realizar un trabajo previamente planificado, con un presupuesto y en un tiempo previamente calculado?

Hoy hay directores de producto que no necesitan conocer cuáles van a ser las 200 funcionalidades que tendrá el producto final, ni si este estará terminado en 12 o en 16 meses.

Hay clientes que necesitan disponer de una primera versión con funcionalidades básicas en cuestión de semanas, y no un producto completo dentro de uno o dos años. Clientes cuyo interés es poner en el mercado rápidamente un concepto nuevo, y desarrollar de forma continua su valor.

Hay proyectos que no necesitan gestionar el seguimiento de un plan, y que fracasan por haber empleado un modelo de gestión inapropiado.

La mayoría de los fiascos se producen por aplicar ingeniería secuencial y gestión predictiva tanto en el proceso de adquisición (requisitos, contratación, seguimiento y entrega) como en la gestión del proyecto, en productos que no necesitan tanto garantías de previsibilidad en la ejecución, como respuesta rápida y flexibilidad para funcionar en entornos de negocio que cambian y evolucionan rápidamente.

El Manifiesto Ágil

En marzo de 2001, 17 críticos de los modelos de producción basados en procesos, convocados por Kent Beck, que había publicado un par de años antes el libro en el que explicaba la nueva metodología Extreme Programming (Beck, 2000) se reunieron en Salt Lake City para discutir sobre el desarrollo de software. En la reunión se acuñó el término “Métodos Ágiles” para definir a aquellos que estaban surgiendo como alternativa a las metodologías formales: CMM-SW, (precursor de CMMI) PMI, SPICE (proyecto inicial de ISO 15504), a las que consideraban excesivamente “pesadas” y rígidas 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.

Hasta 2005, entre los defensores de los modelos de procesos y los de modelos ágiles fueron frecuentes las posturas radicales, más ocupadas en descalificar al otro, que en estudiar sus métodos y conocerlos para mejorar los propios.

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.

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

Este es el valor más importante del manifiesto.

Por supuesto que los procesos ayudan al trabajo. Son una guía de operación. Las herramientas mejoran la eficiencia, pero hay tareas que requieren talento y necesitan personas que lo aporten y trabajen con una actitud adecuada.

La producción basada en procesos persigue que la calidad del resultado sea consecuencia del *know-how* "explicitado" en los procesos, más que en el conocimiento aportado por las personas que los ejecutan.

Sin embargo en desarrollo ágil los procesos son 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, y lo cierto es que este principio no es cierto cuando se necesita creatividad e innovación.

Valoramos más el software que funciona que la documentación exhaustiva.

Poder anticipar cómo será el funcionamiento del producto final, observando prototipos previos, o partes ya elaboradas ofrece un "*feedback*" estimulante y enriquecedor, que genera ideas imposibles de concebir en un primer momento, y difícilmente se podrían incluir al redactar un documento de requisitos detallado en el comienzo del proyecto.

El manifiesto ágil no da por inútil la documentación, sólo la de la documentación innecesaria. Los documentos son soporte de hechos, permiten la transferencia del conocimiento, registran información histórica, y en muchas cuestiones legales o normativas son obligatorios, pero su relevancia debe ser mucho menor que el producto final.

La comunicación a través de documentos no ofrece la riqueza y generación de valor que logra la comunicación directa entre las personas, y a través de la interacción con prototipos del producto.

Por eso, siempre que sea posible debe preferirse reducir al mínimo indispensable el uso de documentación, que requiere trabajo sin aportar un valor directo al producto.

Si la organización y los equipos se comunican a través de documentos, además de ocultar la riqueza de la interacción con el producto, forman barreras de burocracia entre departamentos o entre personas.

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

Las prácticas ágiles están indicadas para productos cuyo detalle resulta difícil prever al principio del proyecto; y si se detallara al comenzar, el resultado final tendría menos valor que si se mejoran y precisan con retroinformación continua durante el.

También son apropiadas cuando se prevén requisitos inestables por la velocidad de cambio en el entorno de negocio del cliente.

El objetivo de un proyecto ágil no es controlar la ejecución conforme a procesos y cumplimiento de planes, sino proporcionar el mayor valor posible al producto.

Resulta por tanto más adecuada una relación de implicación y colaboración continua con el cliente, más que una contractual de delimitación de responsabilidades.

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

Para desarrollar productos de requisitos inestables, que tienen como factor 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 principales valores de la gestión ágil son la anticipación y la adaptación, diferentes a los de la gestión de proyectos ortodoxa: planificación y control que evite desviaciones del plan.

Los 12 principios del manifiesto ágil

El manifiesto ágil, tras los postulados de estos cuatro valores en los que se fundamenta, 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 periodos 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 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:

- ***Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.***
- ***Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos autoorganizados, que en la calidad de los procesos empleados.***
- ***Solapamiento de las diferentes fases del desarrollo, en lugar de realizarlas una tras otra en un ciclo secuencial o de cascada.***

Este modelo fue identificado y definido por Ikujiro Nonaka e Hirotaka Takeuchi a principios de los 80, al analizar cómo desarrollaban los nuevos productos las principales empresas de manufactura tecnológica: Fuji-Xerox, Canon, Honda, Nec, Epson, Brother, 3M y Hewlett-Packard (Nonaka & Takeuchi, The New New Product Development Game, 1986)

En su estudio, Nonaka y Takeuchi compararon la nueva forma de trabajo en equipo, con el avance en formación de scrum de los jugadores de Rugby, a raíz de lo cual quedó acuñado el término “scrum” para referirse a ella.

Aunque esta forma de trabajo surgió en empresas de productos tecnológicos, es apropiada para proyectos con requisitos inestables y para los que requieren rapidez y flexibilidad, situaciones frecuentes en el desarrollo de determinados sistemas de software.

En 1995 Ken Schwaber presentó “Scrum Development Process” en OOPSLA 95 (Object-Oriented Programming Systems & Applications conference) (SCRUM Development Process), un marco de reglas para desarrollo de software, basado en los principios de scrum, y que él había empleado en el desarrollo de Delphi, y Jeff Sutherland en su empresa Easel Corporation (compañía que en los macrojuegos de compras y fusiones, se integraría en VMARK, y luego en Informix y finalmente en Ascential Software Corporation).

Adopciones de scrum: técnica y pragmática

Scrum se puede adoptar de forma técnica, aplicando reglas definidas, o pragmática, adoptando los valores originales scrum con reglas personalizadas. ..

Esta primera parte del temario (Scrum Manager I) enseña scrum técnico, basado en la aplicación de reglas concretas en un marco de roles, eventos y artefactos definidos. El aprendizaje de scrum técnico es el primer paso aconsejable para familiarizarse con la agilidad.

Una vez iniciados en agilidad, y con el conocimiento que el propio equipo acumula a través de las retrospectivas, se pueden ir "rompiendo" las reglas y adoptar scrum pragmático, personalizado y más adecuado a las propias circunstancias del propio equipo y proyecto.

Scrum técnico

Reglas

Marco de reglas para desarrollo de software

Autores: Ken Schwaber y Jeff Sutherland
"Scrum Development Process OOPSLA'95" 1995

Aplicación de reglas definidas

Roles

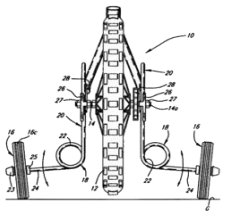
- Dueño de producto
- Equipo de desarrollo
- Scrum Master

Eventos

- El Sprint
- Reunión de planificación
- Scrum diario
- Revisión de sprint
- Retrospectiva de sprint

Artefactos

- Pila de product
- Pila de sprint
- Incremento



Aprender las reglas de Scrum

Scrum pragmático

Valores

Concepto original Scrum

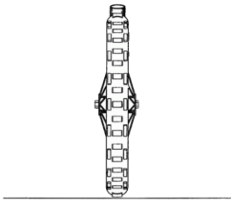
Autores: Hirotaka Takeuchi e Ikujiro Nonaka
"The New New Product Development Game" 1986

Aplicación de valores ágiles

- Personas > procesos
- Resultado > documentación
- Colaboración > negociación
- Cambio > planificación

… "Para avanzar en Scrum"

- Incertidumbre
- Autoorganización
- Fases de desarrollo solapadas
- "Multiaprendizaje"
- Control sutil
- Difusion del conocimiento



Aprender a avanzar en Scrum sin reglas

Introducción al modelo

El marco técnico de scrum, por su sencillez, resulta apropiado para equipos y organizaciones que quieren comenzar a “avanzar en scrum”

Está formado por un conjunto de prácticas y reglas que resultan válidos para dar respuesta a los siguientes principios de desarrollo ágil:

- Gestión evolutiva del avance, en lugar de la tradicional o predictiva.
- Trabajar basando la calidad del resultado en el ***conocimiento tácito de las personas***, más que en el explícito de los procesos y la tecnología empleada.
- Estrategia de ***desarrollo incremental a través de iteraciones*** (sprints) y revisiones.
- Seguir los pasos del desarrollo ágil: desde el concepto o visión general de la necesidad del cliente, construcción del producto de forma incremental a través de iteraciones breves que comprenden fases de especulación – exploración y revisión. Estas iteraciones (en scrum llamadas sprints) se repiten de forma continua hasta que el cliente da por cerrada la evolución del producto.

Se comienza con la visión general de lo que se desea obtener, y a partir de ella se especifica y da detalle a las partes de mayor prioridad, y que se desean tener cuanto antes.

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 desde una, hasta seis semanas, aunque se recomienda que no excedan de un mes.

En scrum, el equipo monitoriza la evolución de cada sprint en reuniones breves diarias donde se revisa en conjunto el trabajo realizado por cada miembro el día anterior, y el previsto para el día en curso. Esta reunión diaria es de tiempo prefijado de 5 a 15 minutos máximo, se realiza de pie junto a un tablero o pizarra con información de las tareas del sprint, y el trabajo pendiente en cada una. Esta reunión se denomina “reunion de pie” o “scrum diario” y si se emplea la terminología inglesa: “stand-up meeting”, también: “daily scrum” o “morning rollcall”.

Gestión de la evolución del proyecto

Scrum maneja de forma empírica 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. Es por tanto la duración del sprint, 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 durante toda la construcción del producto.

El desarrollo incremental ofrece al final de cada iteración una parte de producto operativa, que se puede usar, inspeccionar y evaluar.

Scrum resulta adecuado en proyectos con requisitos inciertos y, o inestables.

¿Por qué predecir la versión definitiva de algo que va a estar evolucionando de forma continua? scrum considera a la inestabilidad como una premisa, y adopta técnicas de trabajo para facilitar la evolución sin degradar la calidad de la arquitectura y permitir que también evolucione durante el desarrollo.

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, en scrum se solapan y realizan de forma continua y simultánea.

Autoorganización

Son muchos los factores impredecibles en un proyecto. La gestión predictiva asigna al rol de gestor del proyecto la responsabilidad de su gestión y resolución.

En scrum los equipos son autoorganizados, con un margen de maniobra suficiente para tomar las decisiones que consideren oportunas.

Colaboración

Es un componente importante y necesario para que a través de la autoorganización se pueda gestionar con solvencia la labor que de otra forma realizaría un gestor de proyectos.

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

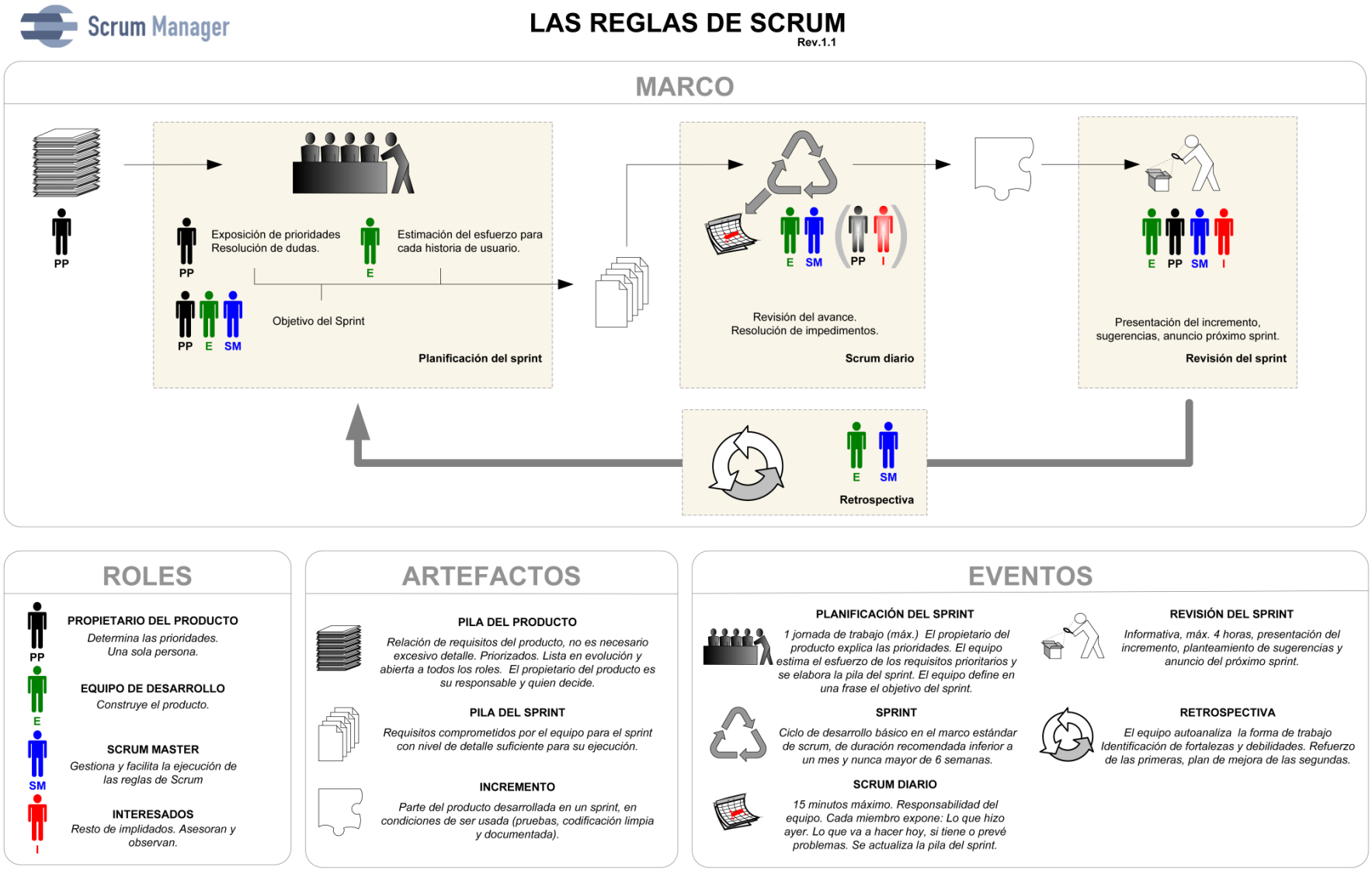


Ilustración 1: Marco scrum técnico

Scrum técnico

Scrum técnico

El marco técnico de scrum está formado por:

- ***Roles***:
  - El equipo scrum.
  - El dueño del producto.
  - El 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.

Y la pieza clave es el ***sprint***.

Se denomina sprint a cada ciclo o iteración de trabajo que produce una parte del producto terminada y funcionalmente operativa (***incremento***)

Como se verá más tarde, al abordar scrum pragmático, las implementaciones más flexibles de scrum pueden adoptar dos tácticas diferentes para mantener un avance continuo en el proyecto:

- ***Incremento iterativo***: basado en pulsos de tiempo prefijado (timeboxing)
- ***Incremento continuo***: basado en el mantenimiento de un flujo continuo, no marcado por pulsos o sprints.

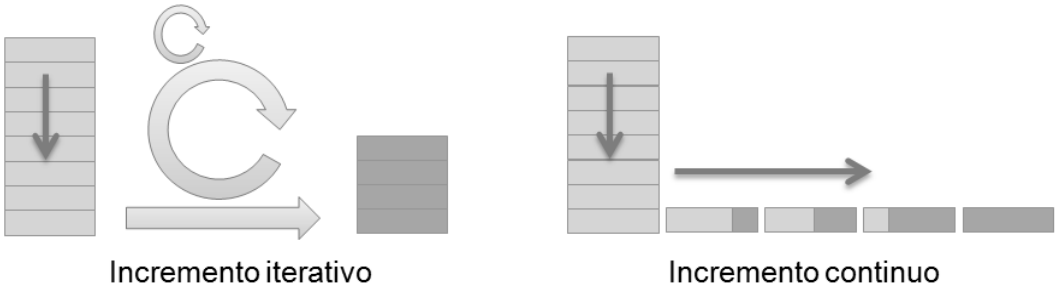


Ilustración 2: Incremento iterativo / continuo

Al usar scrum técnico se trabaja con sprints, y por tanto con incremento iterativo.

Artefactos

- ***Pila del producto***: (product backlog) lista de requisitos de usuario, que a partir de la visión inicial del producto crece y evoluciona durante el desarrollo.
- ***Pila del sprint***: (sprint backlog) lista de los trabajos que debe realizar el equipo durante el sprint para generar el incremento previsto.
- ***Incremento***: resultado de cada sprint.

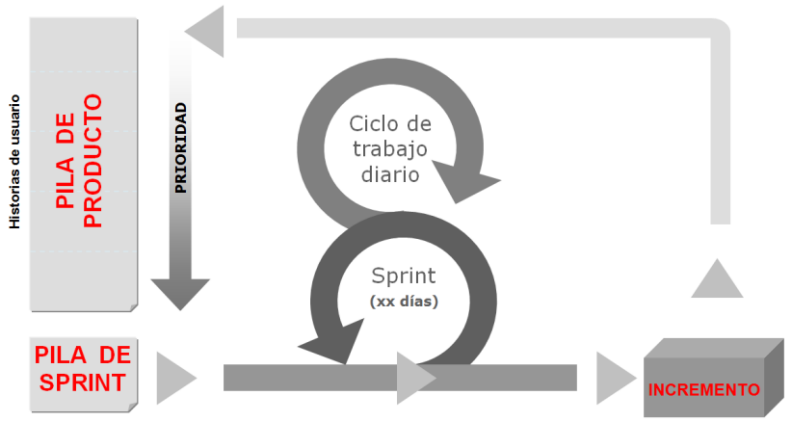


Ilustración 3: Diagrama del ciclo iterativo scrum

Otro artefacto propio del modelo estándar de scrum es el gráfico de avance o gráfico *burn down* que el equipo actualiza a diario para comprobar el avance. Este elemento, junto con la práctica de estimación de póquer y el gráfico de producto o *burn up* se encuentra incluido en el capítulo de Métricas Ágiles.

Pila del producto y pila del sprint: los requisitos en desarrollo ágil.

La ingeniería del software clásica diferencia dos ámbitos de requisitos:

- Requisitos del sistema
- Requisitos del software

Los requisitos del sistema forman parte del proceso de adquisición, y por tanto es responsabilidad del cliente la definición del problema y de las funcionalidades que debe aportar la solución.

No importa si se trata de gestión tradicional o ágil. La pila del producto es responsabilidad del cliente, aunque se aborda de forma diferente en cada caso.

Ilustración 4: Requisitos completos / evolutivos

- En los proyectos predictivos, los requisitos del sistema suelen especificarse en documentos formales; mientras que en los proyectos ágiles toman la forma de pila del producto o lista de historias de usuario.

- Los requisitos del sistema formales se especifican de forma completa y cerrada al inicio del proyecto; sin embargo una pila del producto es un documento vivo, que evoluciona durante el desarrollo.
- Los requisitos del sistema los desarrolla una persona o equipo especializado en ingeniería de requisitos a través del proceso de obtención (elicitación) con el cliente. En scrum la visión del cliente es conocida por todo el equipo (el cliente colabora con el equipo de desarrollo) y la pila del producto se realiza y evoluciona de forma continua con los aportes de todos.

Scrum, aplicado al software, emplea dos formatos para registrar los requisitos:

- Pila del producto (Product Backlog)
- Pila del sprint (Sprint Backlog)

La pila del producto registra los requisitos vistos desde el punto de vista del cliente. Un enfoque similar al de los requisitos del sistema o “ConOps” de la ingeniería tradicional. Está formada por la lista de funcionalidades o "historias de usuario" que desea obtener el cliente, ordenadas por la prioridad que el mismo le otorga a cada una.

La pila del sprint refleja los requisitos vistos desde el punto de vista del equipo de desarrollo. Está formada por la lista de tareas en las que se descomponen las historias de usuario que se van a llevar a cabo en el sprint.

En el desarrollo y mantenimiento de la pila del producto lo relevante no es tanto el formato, sino que:

- Las funcionalidades que incluye den forma a una visión del producto definida y conocida por todo el equipo.
- Las funcionalidades estén individualmente definidas, priorizadas y pre-estimadas.
- Esté realizada y gestionada por el cliente (propietario del producto).

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 el cliente, los usuarios, y en general los interesados. Todo lo que suponga un trabajo que debe realizar el equipo debe estar reflejado en esta pila.

Estos son algunos ejemplos de posibles entradas a una pila de producto:

- Permitir a los usuarios la consulta de las obras publicadas por un determinado autor.
- Reducir el tiempo de instalación del programa.
- Mejorar la escalabilidad del sistema.
- Permitir la consulta de una obra a través de un API web.

La pila de requisitos del producto nunca se da por completada; está en continuo crecimiento y evolución. Al comenzar el proyecto incluye los requisitos inicialmente conocidos y mejor entendidos, y conforme avanza el desarrollo, y evoluciona el entorno en el que será usado, se va desarrollando.

En definitiva su continuo dinamismo refleja aquello que el producto necesita incorporar para ser el más adecuado a las circunstancias, en todo momento.

Para comenzar el desarrollo se necesita la visión del objetivo de negocio que se quiere conseguir con el proyecto, comprendida y conocida por todo el equipo, y elementos suficientes en la pila para llevar a cabo el primer sprint.

Habitualmente se comienza a elaborar la pila con el resultado de una reunión de “tormenta de ideas”, o "fertilización cruzada", o un proceso de “Exploración” (eXtreme Programming) donde colabora todo el equipo partiendo de la visión del propietario del producto.

El formato de la visión no es relevante. Según los casos, puede ser una presentación informal del responsable del producto, un informe de requisitos del departamento de marketing, u otros.

Sin embargo, sí es importante disponer de una visión real, comprendida y compartida por todo el equipo.

El propietario del producto mantiene la pila ordenada por la prioridad de los elementos, siendo los más prioritarios los que confieren mayor valor al producto, o por alguna razón resultan más necesarios, y determinan las actividades de desarrollo inmediatas.

El detalle de los requisitos en la pila del producto debe ser proporcional a la prioridad: Los elementos de mayor prioridad deben tener mayor nivel de comprensión y detalle que los del resto. De esta forma el equipo de desarrollo puede descomponer un elemento de prioridad alta en tareas con la precisión suficiente para ser hecho en un sprint.

Los elementos de la pila del producto que pueden ser incorporados a un sprint se denominan “preparados” o “accionables” y son los que pueden seleccionarse en la reunión de planificación del sprint.

Preparación de la pila del producto

Se denomina “preparación” (grooming) de la pila del producto a las actividades de priorización, detalle y estimación de los elementos que la componen. Es un proceso que realizan de forma puntual, en cualquier momento, continua y colaborativa el propietario del producto y el equipo de desarrollo. No debe consumir más del 10% de la capacidad de trabajo del equipo.

La responsabilidad de estimar el esfuerzo previsible para cada elemento, es de las personas del equipo que previsiblemente harán el trabajo.

Formato de la pila del producto

Scrum prefiere la comunicación verbal o de visualización directa, a la escrita.

La pila del producto no es un documento de requisitos, sino una herramienta de referencia para el equipo.

Si se emplea formato de lista, es recomendable que al menos incluya la siguiente información para cada elemento:

- Identificador único de la funcionalidad o trabajo.
- Descripción de la funcionalidad/requisito, denominado “historia de usuario”.
- Campo o sistema de priorización.
- Estimación del esfuerzo necesario.

Dependiendo del tipo de proyecto, funcionamiento del equipo y la organización, pueden ser aconsejables otros campos:

- Observaciones.
- Criterio de validación.
- Persona asignada.
- Nº de Sprint en el que se realiza.
- Módulo del sistema al que pertenece.
- Entre otros.

Es preferible no adoptar formatos rígidos. Los resultados de scrum no dependen de las formas, sino de la institucionalización de sus principios y la implementación adecuada a las características de la empresa y del proyecto. He aquí un sencillo ejemplo de pila de producto:

Id	Prioridad	Descripción	Est.	Por
1	Muy alta	Plataforma tecnológica	30	AR
2	Muy Alta	Interfaz de usuario	40	LM
3	Muy Alta	Un usuario se registra en el sistema	40	LM
4	Alta	El operador define el flujo y textos de un expediente	60	AR
5	Alta	xxx	999	CC

Ilustración 5: Ejemplo de pila de producto

Pila del Sprint

La pila del sprint (*sprint Backlog*) es la lista que descompone las funcionalidades de la pila del producto (historias de usuario) en las tareas necesarias para construir un incremento: una parte completa y operativa del producto.

La realiza el equipo durante la reunión de planificación del sprint, autoasignando cada tarea a un miembro del equipo, e indicando en la misma lista cuánto tiempo o esfuerzo se prevé que falta para terminarla.

La pila del sprint descompone el trabajo en unidades de tamaño adecuado para monitorizar el avance a diario, e identificar riesgos y problemas sin necesidad de procesos de gestión complejos.

Es también una herramienta para la comunicación visual directa del equipo.

Condiciones

- Realizada de forma conjunta por todos los miembros del equipo.
- Cubre todas las tareas identificadas por el equipo para conseguir el objetivo del sprint.
- Sólo el equipo la puede modificar durante el sprint.
- Las tareas demasiado grandes deben descomponerse en otras más péquelas. Se deben considerar “grandes” .las tareas que necesitan más de un día para realizarse.
- Es visible para todo el equipo. Idealmente en un tablero o pared en el mismo espacio físico donde trabaja el equipo.

Formato y soporte

Son soportes habituales:

- Tablero físico o pared.
- Hoja de cálculo.
- Herramienta colaborativa o de gestión de proyectos.

Y sobre el más adecuado a las características del proyecto, oficina y equipo, lo apropiado es diseñar el formato más cómodo para todos, teniendo en cuenta los siguientes criterios:

- Incluir la siguiente información: Pila del sprint, persona responsable de cada tarea, estado en el que se encuentra y tiempo de trabajo que queda para completarla.
- Incluir sólo la información estrictamente necesaria.
- Debe servir de medio para registrar en cada reunión diaria del sprint, el tiempo que le queda a cada tarea.
- Facilitar la consulta y la comunicación diaria y directa del equipo.

Ejemplo:

PROYECTO			Sáb 07 Ene	Dom 08 Ene	Lun 09 Ene	Mar 10 Ene	Mié 11 Ene
Inicio	Fin	Jornada					
7-ene-12	1-abr-12	40 hs					
		Tareas pendientes	15	15	14	14	11
		Horas pendientes	172	162	148	142	124
		Fecha de Cierre	12-ene	12-ene	12-ene	13-ene	16-ene

PILA DEL PRODUCTO					OBJETIVO DEL SPRINT				
Categoria	Tarea	Responsable	Estimado en horas	Estado	*Crear y publicar versión básica del sitio web público*				
Diseño	Crear diseño de base de datos	Juan	24	Completo	24	16	8	4	
Diseño	Validar diseño de base de datos	Pedro	4	Completo	4	4	4	4	
Desarrollo	Contratar servicio de hosting	Pedro	4	Completo	4	2			
Desarrollo	Crear layout y estilos de sitio web	Maria	16	Activo	8	8	4	2	
Desarrollo	Crear página principal	Maria	24	Pendiente	24	24	24	24	24
Desarrollo	Mostrar resumenes de noticias por sección	Juan	16	Pendiente	16	16	16	16	8
Desarrollo	Crear banners de publicidad	Luis	24	Pendiente	24	24	24	24	24
Desarrollo	Visualizar un Artículo	Luis	8	Pendiente	8	8	8	8	8
Desarrollo	Imprimir un Artículo	Luis	4	Pendiente	4	4	4	4	4

Ilustración 6: Ejemplo de pila de sprint con hoja de cálculo

Durante el sprint, el equipo actualiza a diario en ella los tiempos pendientes de cada tarea. Al mismo tiempo, con estos datos traza el gráfico de avance o trabajo consumido (burn-down), que se describe más adelante, en el capítulo de métricas ágiles.

El Incremento

El incremento es la parte de producto producida en un sprint, y tiene como característica el estar completamente terminada y operativa, en condiciones de ser entregada al cliente.

No se deben considerar como Incremento a prototipos, módulos o sub-módulos, ni partes pendientes de pruebas o integración.

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.

Sin embargo es una excepción frecuente el primer sprint. En el que objetivos del tipo “contrastar la plataforma y el diseño” pueden resultar necesarios, e implican trabajos de diseño o desarrollo de prototipos para contrastar las expectativas de la plataforma o tecnología que se va a emplear. Teniendo en cuenta esta excepción habitual:

> Incremento es la parte de producto realizada en un sprint potencialmente entregable: **terminada** y **probada**.

Si el proyecto o el sistema requiere documentación, o procesos de validación y verificación documentados, o con niveles de independencia que implican procesos con terceros, éstos también tienen que estar realizados para considerar que el incremento está “hecho” .

Eventos

- ***Sprint***: nombre que recibe cada iteración de desarrollo. Es el núcleo central que genera el pulso de avance por tiempos prefijados (time boxing).
- ***Reunión de Planificación del sprint***: reunión de trabajo previa al inicio de cada sprint en la que se determina cuál va a ser el objetivo del sprint y las tareas necesarias para conseguirlo.
- ***Scrum diario:*** breve reunión diaria del equipo, en la que cada miembro responde a tres cuestiones:

  1.- El trabajo realizado el día anterior.

  2.- El que tiene previsto realizar.

  3.- Cosas que puede necesitar o impedimentos que deben eliminarse para poder realizar el trabajo.

  Cada persona actualiza en la pila del sprint el tiempo o esfuerzo pendiente de sus tareas, y con esta información se actualiza a su vez el gráfico con el que el equipo monitoriza el avance del sprint (burn-down)
- ***Revisión del sprint:*** análisis e inspección del incremento generado, y adaptación de la pila del producto si resulta necesario.

Una cuarta reunión se incorporó al marco estándar de scrum en la primera década de 2.000:

- ***Retrospectiva del sprint:*** revisión de lo sucedido durante el Sprint. Reunión en la que el equipo analiza aspectos operativos de la forma de trabajo y crea un plan de mejoras para aplicar en el próximo sprint.

Planificación del sprint

Descripción

En esta reunión 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 en el siguiente sprint.

Se trata de una reunión conducida por el responsable del funcionamiento del marco scrum (Scrum Master en scrum técnico, o un miembro del equipo, en scrum pragmático) a la que deben asistir el propietario del producto y el equipo completo, y a la que también pueden asistir otros implicados en el proyecto.

La reunión puede durar una jornada de trabajo completa, cuando se trata de planificar un sprint largo (de un mes de duración) o un tiempo proporcional para planificar un sprint más breve.

Esta reunión debe dar respuesta a dos cuestiones:

- Qué se entregará al terminar el sprint.
- Cuál es el trabajo necesario para realizar el incremento previsto, y cómo lo llevará a cabo el equipo.

La reunión se articula en dos partes de igual duración, para dar respuesta a una de estas cuestiones, en cada una.

Precondiciones

- La organización tiene determinados los recursos disponibles para llevar a cabo el sprint.
- Ya están “preparados” los elementos prioritarios de la pila del producto, de forma que ya tienen un nivel de detalle suficiente y una estimación previa del trabajo que requieren.
- El equipo tiene un conocimiento de las tecnologías empleadas, y del negocio del producto suficiente para realizar estimaciones basadas en juicio de expertos, y para comprender los conceptos del negocio que expone el propietario del producto.

Entradas

- La pila del producto.
- El producto desarrollado hasta la fecha en los incrementos anteriores (excepto si se trata del primer sprint).
- Dato de la velocidad o rendimiento del equipo en el último sprint, que se emplea como criterio para estimar la cantidad de trabajo que es razonable suponer para el próximo sprint.
- Circunstancias de las condiciones de negocio del cliente y del escenario tecnológico empleado.

Resultados

- Pila del sprint.
- Duración del sprint y fecha de la reunión de revisión.
- Objetivo del sprint.

Formato de la reunión

Esta reunión marca el inicio de cada sprint.

Duración máxima: un día.

Asistentes: Propietario del producto, equipo de desarrollo y Scrum Master.

Pueden asistir: todos aquellos que aporten información útil, ya que es una reunión abierta.

Consta de dos partes separadas por una pausa de café o comida, según la duración.

Primera parte: Qué se entregará al terminar el sprint.

El propietario del producto presenta la pila de producto, exponiendo los requisitos de mayor prioridad que necesita y que prevé que se podrán desarrollar en el siguiente sprint. Si la pila del producto ha tenido cambios significativos desde la anterior reunión, explica las causas que los han ocasionado.

El objetivo es que todo el equipo conozca las razones y los detalles con el nivel suficiente para comprender el trabajo del sprint.

Propietario del producto:

- Presenta las funcionalidades de la pila del producto que tienen mayor prioridad y que estima se pueden realizar en el sprint.
- La presentación se hace con un nivel de detalle suficiente para transmitir al equipo toda la información necesaria para construir el incremento.

El equipo

- Realiza las preguntas y solicita las aclaraciones necesarias.
- Propone sugerencias, modificaciones y soluciones alternativas.

Los aportes del equipo pueden suponer modificaciones en la pila.

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

Tras reordenar y replantear las funcionalidades de la pila del producto, el equipo define el "objetivo del sprint" o frase que sintetiza cuál es el valor que se le va a entregar al cliente.

Exceptuando sprints dedicados exclusivamente a refactorización o a colecciones de tareas desordenadas (que deberían ser los menos), 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 las decisiones que autogestiona el equipo.

Segunda parte: Cómo se conseguirá hacer el incremento.

El equipo desglosa cada funcionalidad en tareas, y estima el tiempo para cada una de ellas, componiendo así las tareas que forman la pila del sprint. En este desglose, el equipo tiene en cuenta los elementos de diseño y arquitectura que deberá incorporar el sistema.

Los miembros del equipo establecen cuáles van a ser las tareas para los primeros días del sprint, y se las autoasignan tomando como criterios sus conocimientos, intereses y una distribución homogénea del trabajo.

Esta segunda parte debe considerarse como una "reunión del equipo", en la que deben estar todos sus miembros, y ser ellos quienes descompongan estimen y asignen el trabajo.

El papel del propietario del producto es atender a dudas y comprobar que el equipo comprende y comparte su objetivo.

El Scrum Master actúa de moderador de la reunión.

Funciones del Scrum Master

El Scrum Master, o el moderador de la reunión es responsable y garante de:

1.- Realizar esta reunión antes de cada sprint.

2.- Asegurar que se cuenta con una pila de producto adecuadamente preparada por el propietario del producto.

3.- Ayudar a mantener el diálogo entre el propietario del producto y el equipo.

4.- Asegurar que se llegue a un acuerdo entre el propietario del producto y el equipo respecto de lo que incluirá el incremento.

5.- Ayudar al equipo a comprender la visión y necesidades de negocio del cliente.

6.- Asegurar que el equipo 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 tareas estimadas.
- La duración del sprint y la fecha de la reunión de revisión.

El Scrum Master modera la reunión para que no dure más de un día. Debe evitar que el equipo comience a profundizar en trabajos de análisis o arquitectura que son propios del trabajo del sprint.

Ejemplo de tablero operativo para la reunión

Es recomendable, que el propietario del producto emplee una hoja de cálculo o alguna herramienta similar para guardar en formato digital la pila del producto. Aunque no es aconsejable usarla como base para trabajar sobre ella en la reunión.

En la reunión es preferible usar una pizarra o un tablero y fichas o etiquetas removibles.

El tablero facilita la comunicación y el trabajo de la reunión.

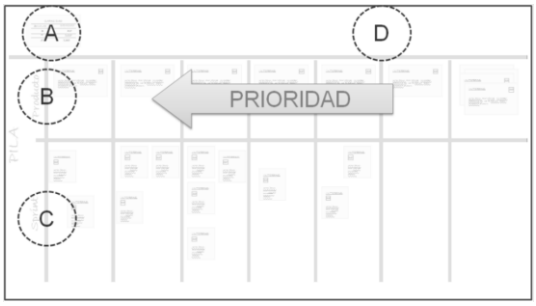


Ilustración 7: Ejemplo de pizarra de trabajo

Algunos soportes habituales:

- Pizarra blanca y notas adhesivas tipo Post-it®.
- Tablero de corcho laminado y chinchetas para sujetar las notas.
- Tablero de acero vitrificado y soportes magnéticos para sujetar notas.

Con cinta adhesiva removible se marcan líneas para delimitar:

- Un área superior donde el equipo anota:
  - (A) las unidades de trabajo que según la velocidad media del equipo se podrían realizar en sprints de 2, 3, 4 y 5 semanas.
  - (D) Duración que finalmente tendrá el sprint, así como el objetivo establecido, duración, hora fijada para las reuniones diarias y fecha prevista para la reunión de revisión del sprint.
- B.- Una franja para ordenar los elementos de la pila del producto de mayor a menor prioridad.
- C.-Una franja paralela para descomponer cada elemento de la pila del producto en las correspondientes tareas de la pila del sprint.

Scrum diario

Descripción

Reunión diaria breve, de no más de 15 minutos, en la que el equipo sincroniza el trabajo y establece el plan para las 24 horas siguientes.

Entradas

- Pila del sprint y gráfico de avance (burn-down) actualizados con la información de la reunión anterior.
- Información del avance de cada miembro del equipo.

Resultados

- Pila del sprint y gráfico de avance (burn-down) actualizados.
- Identificación de posibles necesidades e impedimentos.

Formato de la reunión

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

En la reunión está presente todo el equipo, y pueden asistir también otras personas relacionadas con el proyecto o la organización, aunque éstas no pueden intervenir.

En esta reunión cada miembro del equipo de desarrollo explica:

- Lo que ha logrado desde el anterior scrum diario.
- Lo que va ha hacer hasta el próximo scrum diario.
- Si están teniendo algún problema, o si prevé que puede encontrar algún impedimento.

Y actualiza sobre la pila del sprint el esfurezo que estima pendiente en las tareas que tiene asignadas, o marca como finalizadas las ya completadas.

Al final de la reunión:

- El equipo refresca el gráfico de avance del sprint, con las estimaciones actualizadas,
- El Scrum Master realiza las gestiones adecuadas para resolver las necesidades o impedimentos identificados.

El equipo es el responsable de esta reunión, no el Scrum Master; y no se trata de una reunión de “inspección” o “control” sino de comunicación entre el equipo para compartir el estado del trabajo, chequear el ritmo de avance y colaborar en posibles dificultades o impedimentos.

Revisión del sprint

Descripción

Reunión realizada al final del sprint para comprobar el incremento. .

No debe durar más de 4 horas, en el caso de revisar sprints largos. Para sprints de una o dos semanas, con una o dos horas de duración debería ser suficiente.

Objetivos:

- El propietario del producto comprueba el progreso del sistema. Esta reunión marca, a intervalos regulares, el ritmo de construcción, y la trayectoria que va tomando la visión del producto.
- El propietario del producto identifica las funcionalidades que se pueden considerar “hechas” y las que no.
- Al ver y probar el incremento, el propietario del producto, y el equipo en general obtienen feedback relevante para revisar la pila del producto.
- Otros ingenieros y programadores de la empresa también pueden asistir para conocer cómo trabaja la tecnología empleada.

Precondiciones

- Se ha concluido el sprint.
- Asiste todo el equipo de desarrollo, el propietario del producto, el Scrum Master y todas las personas implicadas en el proyecto que lo deseen.

Entradas

- Incremento terminado.

Resultados

- Feedback para el propietario del producto: hito de seguimiento de la construcción del sistema, e información para mejorar el valor de la visión del producto.
- Convocatoria de la reunión del siguiente sprint.

Formato de la reunión

Es una reunión informal. El objetivo es ver el incremento realizado. Están prohibidas las presentaciones gráficas y “powerpoints”.

El equipo no debe invertir más de una hora en desarrollar la reunión, y lo que se muestra es el resultado final: terminado, probado y operando en el entorno del cliente (incremento).

Según las características del proyecto puede incluir también documentación de usuario, o técnica.

Es una reunión informativa. **Su misión no es la toma de decisiones ni la crítica del incremento**. Con la información obtenida, posteriormente el propietario del producto tratarán las posibles modificaciones sobre la visión del producto.

Protocolo recomendado:

1.- El equipo expone el objetivo del sprint, la lista de funcionalidades que se incluían y las que se han desarrollado.

2.- El equipo hace una introducción general del sprint y demuestra el funcionamiento de las partes construidas.

3.- 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.

4.- El Scrum Master, de acuerdo con las agendas del propietario del producto y el equipo, cierra la fecha para la reunión de preparación del siguiente sprint.

Retrospectiva

Reunión que se realiza tras la revisión de cada sprint, y antes de la reunión de planificación del siguiente, con una duración recomendada de una a tres horas, según la duración del sprint terminado.

En ella el equipo realiza autoanálisis de su sobre su forma de trabajar, e identifica fortalezas y puntos débiles. El objetivo es consolidar y 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 considerarlas erróneamente como reuniones 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, mientras que una reunión retrospectiva se centra en “CÓMO” lo estamos construyendo: “CÓMO” estamos trabajando, con el objetivo de analizar problemas y aspectos mejorables.

Las reuniones "retrospectivas" realizadas de forma periódica por el equipo para mejorar la forma de trabajo, se consideran cada vez más un componente del marco técnico de scrum, si bien no es una reunión para seguimiento de la evolución del producto, sino para mejora del marco de trabajo.

Roles

Todas las personas que intervienen, o tienen relación directa o indirecta con el proyecto, se clasifican en dos grupos: comprometidos e implicados. En círculos de scrum es frecuente llamar a los primeros (sin ninguna connotación peyorativa) “cerdos” y a los segundos “gallinas”.

El origen de estos nombres está en la siguiente metáfora que ilustra de forma gráfica la diferencia entre “compromiso” e “implicación” en el proyecto:

Una gallina y un cerdo paseaban por la carretera. La gallina preguntó al cerdo: “¿Quieres abrir un restaurante conmigo?”.

El cerdo consideró la propuesta y respondió: “Sí, me gustaría. ¿Y cómo lo llamaríamos?”.

La gallina respondió: “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”.

COMPROMETIDOS (CERDOS)	IMPLICADOS (GALLINAS)
Propietario del producto	Otros interesados (dirección, gerencias, comerciales, marketing, etc.)
Miembros del equipo	

Ilustración 8: Roles estándar de scrum

- Propietario del producto: es la persona responsable de lograr el mayor valor de producto para los clientes, usuarios y resto de implicados.
- ***Equipo de desarrollo***: grupo o grupos de trabajo que desarrollan el producto.

Una observación en este punto, sobre el rol de Scrum Master, por ser en ocasiones frecuente la duda de considerar si es un rol “comprometido” o “implicado”. Partiendo de que la división entre personas comprometidas y personas implicadas es más “conceptual” que “relevante”, pero cuando se trabaja con este rol presente, su responsabilidad es el funcionamiento de un scrum técnico en la organización.

Su responsabilidad directa, su misión, es por tanto la forma de trabajo, siendo por tanto el producto elaborado en los proyectos un objetivo de segundo nivel, o indirecto.

Por esta razón en el cuadro anterior no se considera el rol de Scrum Master, aunque que en cualquier caso no es una cuestión especialmente relevante. Si hubiera que forzar una respuesta, desde el criterio de que no está comprometido en el proyecto (sino en la mejora de la forma de trabajo) se debería considerar como un rol "implicado"

Propietario del producto

El propietario del producto (*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 desarrollo sólo se integra una persona en representación del cliente, y ésta debe tener el conocimiento suficiente del producto y las atribuciones necesarias para tomar las decisiones que le corresponden.

En resumen, el propietario de producto es quien:

- Decide en última instancia cómo será el resultado final, y el orden en el que se van construyendo los sucesivos incrementos: qué se pone y qué se quita de la pila del producto, y cuál es la prioridad de las funcionalidades.
- Conoce el plan del producto, sus posibilidades y plan de inversión, así como del retorno esperado a la inversión realizada, y se responsabiliza sobre fechas y funcionalidades de las diferentes versiones del mismo.

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 delegue en el equipo de desarrollo, o en alguien de su confianza, pero la responsabilidad siempre es suya.

Para ejercer este rol es necesario:

- ***Conocer perfectamente el entorno de negocio del cliente***, las necesidades y el objetivo que se persigue con el sistema que se está construyendo.
- Tener la ***visión del producto***, así como las necesidades concretas del proyecto, para poder priorizar eficientemente el trabajo.

- Disponer de ***atribuciones y conocimiento del plan del producto suficiente*** para tomar las decisiones necesarias durante el proyecto, incluidas para cubrir las expectativas previstas de retorno de la Inversión del proyecto.
- Recibir y analizar de forma continua ***retroinformación del entorno de negocio*** (evolución del mercado, competencia, alternativas) y del proyecto (sugerencias del equipo, alternativas técnicas, pruebas y evaluación de cada incremento).

Es además recomendable que el propietario de producto:

- Conozca scrum para realizar con solvencia las tareas que le corresponden:
  - 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.
- Conozca y haya trabajado previamente con el mismo equipo.

La organización debe respetar sus decisiones y no modificar prioridades ni elementos de la pila del producto.

Equipo de desarrollo

Lo forman el grupo de profesionales que realizan el incremento de cada sprint.

Se recomienda que un equipo scrum tenga entre 3 y 8 personas. Más allá de 8 resulta más difícil mantener la comunicación directa, y se manifiestan con más intensidad los roces habituales de la dinámica de grupos (que comienzan a aparecer a partir de 6 personas). En el cómputo del número de miembros del equipo *de desarrollo* no se consideran ni el Scrum Master ni el propietario del producto.

No se trata de un grupo de trabajo formado por un arquitecto, diseñador o analista, programadores y testers. Es un equipo ***multifuncional***, en el que todos los miembros trabajan de forma solidaria con responsabilidad compartida. Es posible que algunos miembros sean especialistas en áreas concretas, pero la responsabilidad es el incremento de cada sprint y recae sobre el equipo de desarrollo en conjunto.

Las principales responsabilidades, más allá de la autoorganización y uso de tecnologías ágiles, son las que se marcan la diferencia entre “grupo de trabajo” y “equipo”.

Un grupo de trabajo es un conjunto de personas que realizan un trabajo, con una asignación específica de tareas, responsabilidades y siguiendo un proceso o pautas de ejecución. Los operarios de una cadena, forman un grupo de trabajo: aunque tienen un jefe común, y trabajan en la misma organización, cada uno responde por su trabajo.

El equipo tiene espíritu de colaboración, y un propósito común: conseguir el mayor valor posible para la visión del cliente.

Un equipo scrum responde en su conjunto. Trabaja de forma cohesionada y autoorganizada. No hay un gestor para delimitar, asignar y coordinar las tareas. Son los propios miembros los que lo realizan.

En el equipo:

- Todos conocen y comprenden la visión del propietario del producto.
- Aportan y colaboran con el propietario del producto en el desarrollo de la pila del producto.
- Comparten de forma conjunta el objetivo de cada sprint y la responsabilidad del logro.
- Todos los miembros participan en las decisiones.
- Se respetan las opiniones y aportes de todos.
- Todos conocen el modelo de trabajo con scrum.

Scrum Master

Es el responsable del cumplimiento de las reglas de un marco de scrum técnico, asegurando que se entienden en la organización, y se trabaja conforme a ellas.

Propociona la asesoría y formación necesaria al propietario del producto y al equipo.

Realiza su trabajo con un modelo de liderazgo servil: al servicio y en ayuda del equipo y del propietario del producto.

Proporciona:

- Asesoría y formación al equipo para trabajar de forma autoorganizada y con responsabilidad de equipo.
- Revisión y validación de la pila del producto.
- Moderación de las reuniones.
- Resolución de impedimentos que en el sprint pueden entorpecer la ejecución de las tareas.
- Gestión de las “dinámicas de grupo” en el equipo.
- Configuración, diseño y mejora continua de las prácticas de scrum en la organización. Respeto de la organización y los implicados, con las pautas de tiempos y formas de scrum.

Al crecer la fluidez de la organización y evolucionar hacia un marco de scrum más pragmático, puede eliminarse el rol de Scrum Master, cuando estas responsabilidades ya estén institucionalizadas en la organización.

Cultura y Valores

Scrum técnico define un marco que ayuda a organizar a las personas y el flujo de trabajo. Es la “carrocería” o el interfaz visible, pero el motor de la agilidad son los valores ágiles.

Las reglas de un equipo scrum pueden ser las de este marco técnico u otras. La agilidad no la proporciona el cumplimiento de prácticas, sino de valores.

- ***Delegación de atribuciones*** (empowerment) al equipo para que pueda autoorganizarse y tomar las decisiones sobre el desarrollo.
- ***Respeto entre las personas***. Los miembros del equipo deben confiar entre ellos y respetar sus conocimientos y capacidades.
- ***Responsabilidad y autodisciplina*** (no disciplina impuesta).
- Trabajo centrado en el ***valor para el cliente*** y el desarrollo de lo comprometido.
- Información, ***transparencia*** y visibilidad del desarrollo del proyecto.

Medición y estimación ágil

¿Por qué medir?

La información es la materia prima para la toma de decisiones, y la que puede ser cuantificada proporciona criterios objetivos de gestión y seguimiento.

Desde el nivel concreto de la programación, hasta los más generales de la gestión global de la organización, tres son los fondos de escala o niveles de zoom con los que se puede medir el *trabajo:*

- Desarrollo y gestión de la solución técnica.
- Gestión de proyecto.
- Gestión de la organización.

En el primero se puede medir, por ejemplo, la proporción de polimorfismo del código de un programa, en el segundo, el porcentaje del plan del proyecto realizado, y en el tercero, también por ejemplo, el nivel de satisfacción laboral.

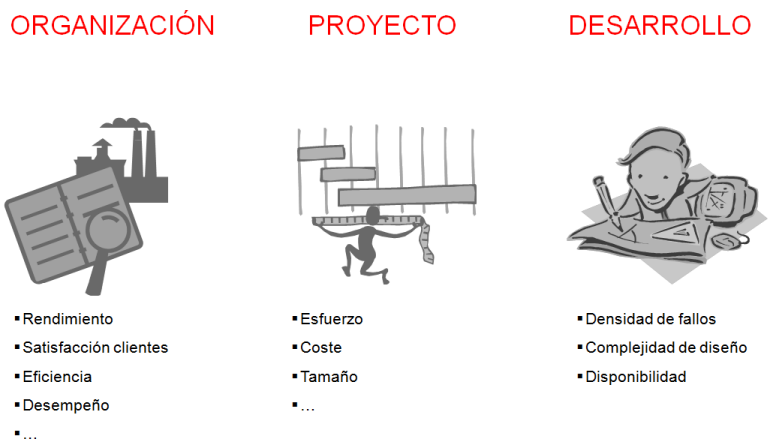


Ilustración 9: Ámbitos de medición

Este texto cubre la medición ágil en el ámbito proyecto, aunque las consideraciones generales de esta introducción son comunes a los tres.

Flexibilidad y sentido común

Medir es costoso

Antes de incorporar un procedimiento de medición, se debe cuestionar su conveniencia, y la forma en la que se aplicará.

Medir no es un fin en sí mismo

No se deben implantar procesos de medición simplemente por que sí.

Tomar una lista más o menos prestigiosa de métricas, e incorporarla a los procedimientos de la empresa puede seducir, al ofrecer un marco de trabajo monitorizado con indicadores y mediciones, pero no dice mucho a favor de las razones por las que se han adoptado.

Criterios para el diseño y aplicación de métricas

Cuantas menos, mejor

- Medir es costoso.
- Medir añade burocracia.
- El objetivo de un equipo scrum es ofrecer la mejor relación valor / simplicidad.

Aspectos que deben cuestionarse antes de monitorizar y medir un indicador:

- **¿Por qué?**
- **¿Qué valor proporciona esta medición?**
- **¿Qué valor se pierde si se omite?**

El objetivo de scrum es producir el mayor valor posible de forma continua, y la cuestión clave para la incorporación de indicadores en la gestión de proyectos es:

¿Cómo contribuye el uso del indicador en el valor que el proyecto proporciona al cliente?

¿El indicador es apropiado para el fin que se debe conseguir?

Medir no es, o no debería ser, un fin en sí mismo.

¿Cuál es el fin?

¿Cumplir agendas, mejorar la eficiencia del equipo, las previsiones…?

Sea crítico. Si después de analizarlo no le convence, si prefiere no incorporar un indicador, o cambiarlo: usted es el gestor.

Determinar qué medir es la parte más difícil. En el mejor de los casos, las decisiones erróneas sólo supondrán un coste de gestión evitable; pero muchas veces empeorarán lo que se intentaba mejorar.

Ejemplo: Medición de la eficiencia de los trabajos de programación

La organización XYZ ha adoptado métricas estándar de eficiencia de Ingeniería del Software:

LOC/Hour: Número total de líneas de código nuevas o modificadas en cada hora.

Además para aumentar la productividad, ha vinculado los resultados de esta métrica a la retribución por desempeño de los programadores, de forma que ha logrado producir más líneas de código sin incrementar la plantilla.

Para evitar que se trate de un incremento “hueco” de líneas de código, o aumente el número de errores por programar más deprisa, se ha dotado de mayor “rigor” al sistema de métrica, incorporando al poco tiempo otras métricas para complementar y mejorar el sistema de calidad:

Test Defects/KLOC, Compile Defects/KLOC y Total Defects/KLOC, para controlar que no crezca el número de errores deslizados en el código.

También se han incorporado indicadores “appraisal time” para medir tiempo y costes del diseño y la ejecución de las revisiones de código.

Y por temor a que el sistema de medición pueda resultar excesivamente costoso se incluyen indicadores de coste de calidad (COQ) que miden los tiempos de revisión y los contrastan con las mejoras en los tiempos eliminados por reducción de fallos.

¿Lo que vamos a medir es un indicador válido de lo que queremos conocer?

Hay tareas de programación relativamente mecánicas, orientadas más a la integración y configuración que en al desarrollo de nuevos sistemas.

Para aquellas puede resultar medianamente acertado considerar la eficiencia como volumen de trabajo realizado por unidad de tiempo.

Para las segundas sin embargo, es más apropiado pensar en la cantidad de valor integrado por unidad de desarrollo; expresadas éstas en horas, iteraciones o puntos de función.

¿Qué queremos conocer: la cantidad de líneas, o el valor entregado al cliente?

¿Está relacionado lo uno con lo otro?

¿Se puede medir objetivamente el valor entregado al cliente?

En nuestro trabajo son muchos los parámetros que se pueden medir con criterios objetivos y cuantificables: el tiempo de tarea, los tiempos delta, y los de las interrupciones, el nº de puntos de función, la inestabilidad de los requisitos, la proporción de acoplamiento, el nº de errores por línea de código…

¿No estaremos muchas veces midiendo esto, simplemente porque es cuantificable?

¿No estaremos midiendo el nº de líneas que desarrollan las personas cuando en realidad queremos saber el valor de su trabajo?

¿No nos estará pasando lo mismo cuando pretendemos medir: la facilidad de uso, la facilidad de mantenimiento, la flexibilidad, la transportabillidad, la complejidad, etc.?

Velocidad, trabajo y tiempo

Velocidad, trabajo y tiempo son las tres magnitudes que componen la fórmula de la velocidad, en gestión de proyectos ágil, definéndola como la cantidad de trabajo realizada por unidad de tiempo.

Velocidad = Trabajo / Tiempo

Así por ejemplo, se puede decir que la velocidad de un equipo de 4 miembros es de 20 puntos por semana o de 80 puntos por sprint.

Tiempo

Para mantener un ritmo de avance continuo, el desarrollo ágil emplea dos tácticas posibles: incremento iterativo, o incremento continuo.

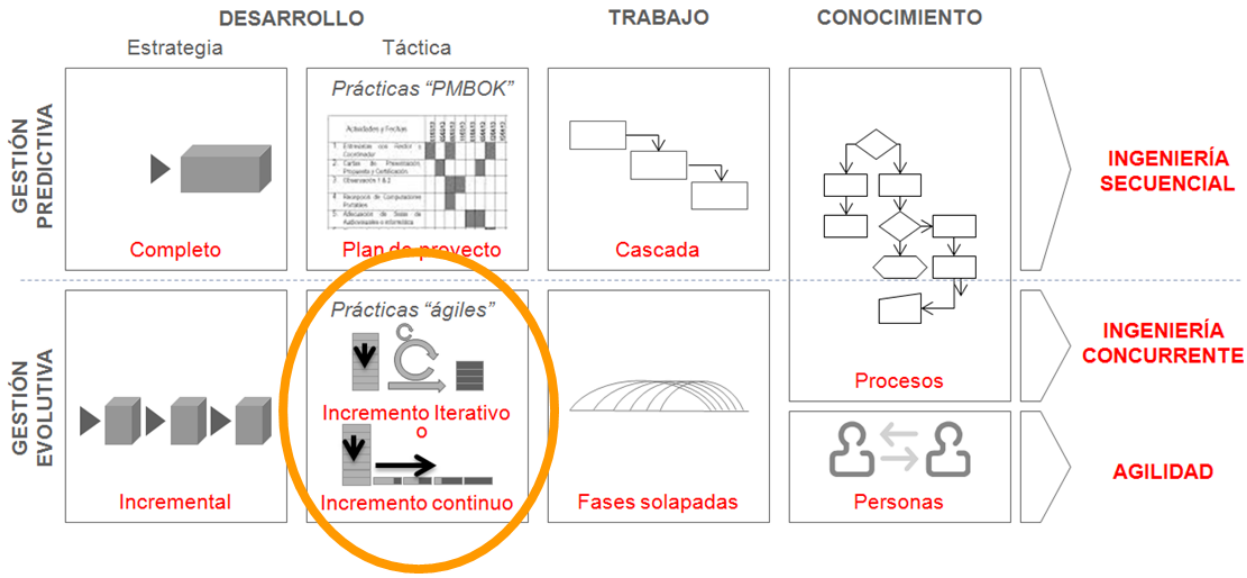


Ilustración 10: Agilidad con incremento iterativo o continuo

El avance a través de incrementos iterativos mantiene el ritmo apoyándose en pulsos de sprints. Por esta razón emplea normalmente el sprint como unidad de tiempo, y expresa la velocidad como trabajo o tareas realizadas en un sprint.

Nota: scrum técnico usa incremento iterativo, y por tanto define la velocidad como la cantidad de trabajo realizado en un sprint.

El avance a través de un incremento continuo mantiene un flujo de avance constante sin puntos muertos ni cuellos de botella. No hay sprints, y por tanto las unidades de tiempo son días, semanas o meses, de forma que la la velocidad se expresa en "puntos" (cantidad de trabajo) por semana, día, o mes

Tiempo real y tiempo ideal

Una observación importante: la diferencia entre tiempo "real" y tiempo "ideal".

Tiempo real, es el tiempo de trabajo. Equivale a la jornada laboral.

Para un equipo de cuatro personas con jornada laboral de ocho horas el tiempo real en una semana (cinco días laborables) es:

4 * 8 * 5 = 160 horas

Tiempo ideal se refiere sin embargo al tiempo de trabajo en condiciones ideales, esto es, eliminando todo lo que no es estrictamente "trabajo", suponiendo que no hay ninguna pausa por interrupción o atención de cuestiones ajenas a la tarea y que la persona se encuentra en buenas condiciones de concentración y disponibilidad.

El tiempo ideal se emplea normalmente en estimaciones, como unidad de trabajo o esfuerzo necesario. Ej: "Esa tarea tiene un tamaño de 3 horas ideales".

Es un concepto similar al que PSP[1] denomina "Delta Time" como la parte del tiempo laboral que es realmente tiempo efectivo de trabajo.

> Tiempo ideal no es una unidad de tiempo, sino de trabajo o esfuerzo necesario.

Trabajo

Medir el trabajo puede ser necesario por dos razones: para registrar el ya hecho, o para estimar anticipadamente, el que se debe realizar.

En ambos casos se necesita una unidad, y un criterio objetivo de cuantificación.

Trabajo ya realizado

Medir el trabajo ya realizado no entraña especial dificultad.

Se puede hacer con unidades relativas al producto (p. ej. líneas de código) o a los recursos empleados (coste, tiempo de trabajo…)

Para medirlo, basta contabilizar lo ya realizado con la unidad empleada: líneas de código, puntos de función, horas trabajadas, etc.

La gestión de proyectos ágil no mide el esfuerzo realizado para calcular el avance del trabajo.

> La gestión ágil no determina el grado de avance del proyecto por el trabajo realizado, sino por el pendiente de realizar.

Es posible que otros procesos de la organización necesiten registrar el esfuerzo invertido, y por lo tanto sea necesario su registro, pero no debe emplearse para calcular el avance del proyecto.

Trabajo pendiente de realizar

Scrum mide el trabajo pendiente para:

[1] Personal Software Process

- Estimar esfuerzo y tiempo previsto para realizar un trabajo (tareas, historias de usuario o epics).
- Determinar el grado de avance del proyecto, y en especial en cada sprint.

Determinar con precisión, de forma cuantitativa y objetiva el trabajo que necesitará la construcción de un requisito, es un empeño cuestionable.

Ilustración 12: Medición del trabajo pendiente

El trabajo necesario para realizar un requisito o una historia de usuario no se puede prever de forma absoluta, porque las funcionalidades no son realidades de solución única, y en el caso de que se pudiera, la complejidad de la medición haría una métrica demasiado pesada para la gestión ágil.

Y si no resulta posible estimar con precisión la cantidad de trabajo que hay en un requisito, tampoco se puede saber cuánto tiempo necesitará, porque además de la incertidumbre del trabajo, se suman las inherentes al "tiempo":

- No es realista hablar de la cantidad o de la calidad del trabajo que realiza una persona por unidad de tiempo, porque son muy grandes las diferencias de unas personas a otras.
- Una misma tarea, realizada por una misma personar requerirá diferentes tiempos en o situaciones distintas.

Sobre estas premisas:

- No es posible estimar con precisión, ni el trabajo de un requisito, ni el tiempo necesario para desarrollarlo.
- La complejidad de las técnicas de estimación crece exponencialmente en la medida que:
  - Intentan incrementar la fiabilidad y precisión de los resultados.
  - Aumenta el tamaño del trabajo estimado.

La estrategia empleada por la gestión ágil es:

- Trabajar con estimaciones aproximadas.
- Estimar con la técnica "juicio de expertos.
- Descomponer las tareas en subtareas más pequeñas, si las estimaciones superan rangos de medio, o un día de tiempo real.

Unidades de trabajo

Un trabajo puede dimensionarse midiendo el producto que se construye, como los tradicionales puntos de función de COCOMO; o el tiempo que cuesta realizarlo.

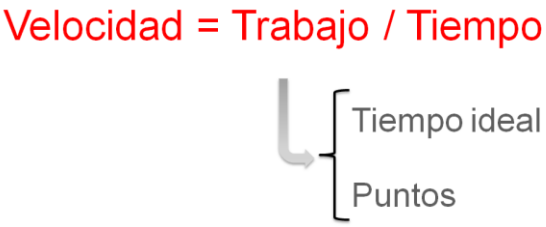


Ilustración 13: Velocidad

En gestión ágil se suelen emplear “puntos” como unidad de trabajo, empleando denominaciones como “puntos de historia” o simplemente “puntos” “puntos.

La unidad “Story Point” de eXtreme Programming se define como la cantidad de trabajo que se realiza en un “día ideal”.

Cada organización, según sus circunstancias y su criterio institucionaliza su métrica de trabajo definiendo el nombre y las unidades.

Puede definir su “punto”

- Como tamaño relativo de tareas conocidas que normalmente emplea.

  Ej: El equipo de un sistema de venta por internet, podría determinar que un “punto” representara el tamaño que tiene un “listado de las facturas de un usuario”.

- En base al tiempo ideal necesario para realizar el trabajo.

  Ej: Un equipo puede determinar que un “punto” es el trabajo realizado en 4 horas ideales.

Es importante que la métrica empleada, su significado y la forma de aplicación sea consistente en todas las mediciones de la organización, y conocida por todas las personas:

Que se trate de un procedimiento de trabajo institucionalizado.

Velocidad

Velocidad es la magnitud determinada por la cantidad de trabajo realizada en un periodo de tiempo.

Velocidad en scrum técnico es la cantidad de trabajo realizada 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.

Al trabajar en implantaciones de scrum pragmático, que pueden realizar sprints de diferentes duraciones, o no siempre con el mismo número de miembros en el equipo, la velocidad se expresa indicando la unidad de tiempo y en su caso también si se refiere a la total del equipo, o a la media por persona. Así por ejemplo: “La velocidad media del equipo es de 100 puntos por semana.” “La velocidad media de una persona del equipo es de 5 puntos por día.”

Medición: usos y herramientas

Gráfico de producto.

El gráfico de producto o gráfico "*burn up*" es una herramienta de planificación del propietario del producto, que muestra visualmente la evolución previsible del producto.

Proyecta en el tiempo su construcción, en base a la velocidad del equipo.

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 o en tiempo real.

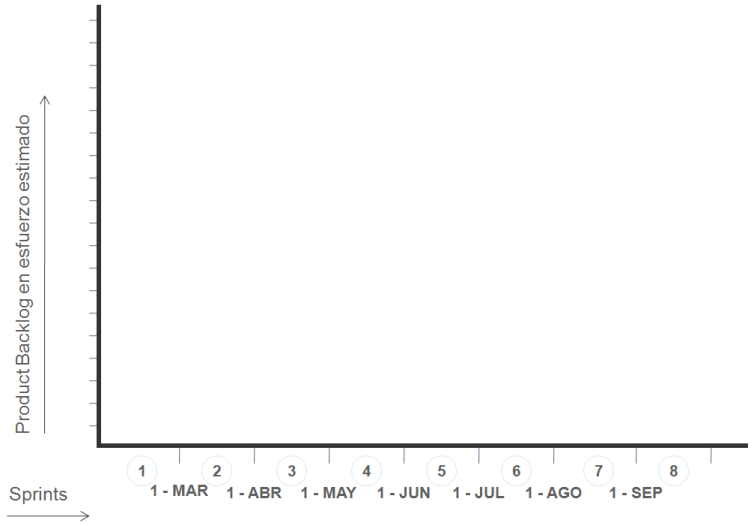


Ilustración 14: Gráfico de producto

Ejemplo

Convenciones empleadas por el equipo:

- Unidad para estimar el trabajo: puntos de scrum.
- Está previsto trabajar con sprints de duración fija: mensual (20 días laborables)
- El equipo está formado por 4 personas, y desarrolla una velocidad media de 400 puntos por sprint.

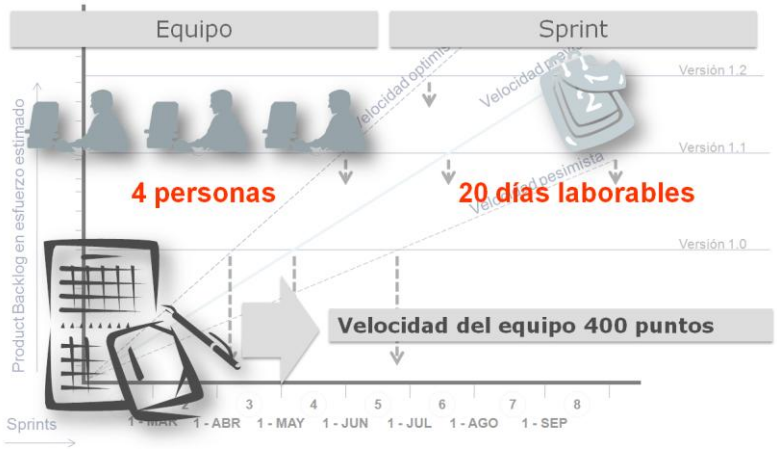


Ilustración 15: Gráfico de producto como plan de producto

Se traza en el gráfico la línea que representa el ritmo de avance previsto, según la velocidad media del equipo (en este ejemplo 400 puntos por sprint).

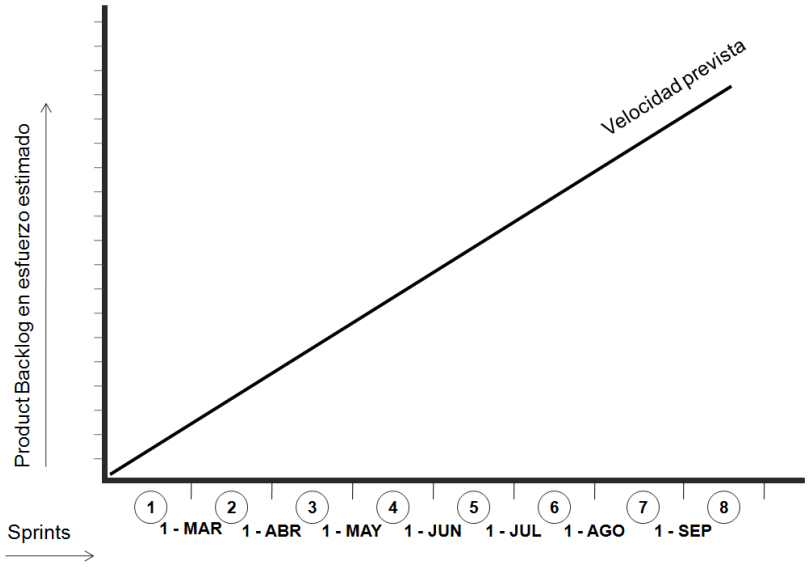


Ilustración 16: Gráfico de producto: velocidad prevista

Es recomendable trazar también los ritmos de avance con una previsión pesimista y otra optimista. Se dibujan basándose en la velocidad obtenida en los sprints anteriores que han ido peor y mejor de lo previsto, o en su defecto estableciendo un margen según el criterio del equipo (ej. +- 20%).

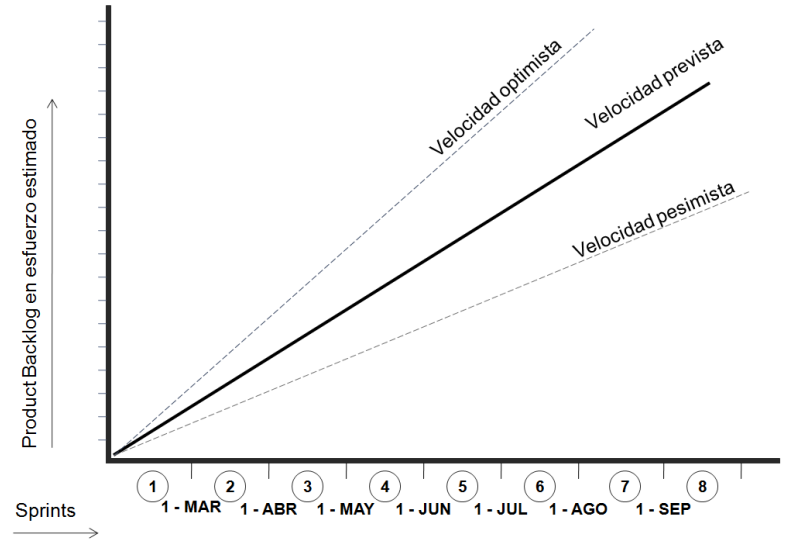


Ilustración 17: Gráfico de producto: velocidad optimista y pesimista

A continuación se toma la pila del producto. La figura siguiente representa la empleada en este ejemplo:

Id	Historias	Trabajo	Criterio de validación
1	Historia A 1.0	150	Lorem ipsum dolor sit amet
2	Historia B 1.0	250	consectetuer adipiscing elit
3	Historia C 1.0	250	Aliquam vehicula accumsan tortor
4	Historia D 1.0	300	Pellentesque turpis
5	Historia A 1.1	250	Phasellus purus orci
6	Historia D 1.1	350	penatibus et magnis dis parturient
7	Historia E 1.0	150	Quisque volutpat ante sit amet velit
8	Historia B 1.1	500	Cras iaculis pede eu tellus
9	Historia C 1.1	150	Vestibulum vel diam sed pede blandit
10	Historia E 1.1	200	Suspendisse aliquam felis et turpis
11	Historia F 1.0	TBD	Nullam imperdiet lorem vitae justo
12	Historia A.1.2	TBD	Suspendisse potenti. In nec nunc
13	Historia B 1.2	TBD	Nam eros tellus, facilisis sed, pretium
14	Historia F 1.1	TBD	Morbi arcu tellus, condimentum

Ilustración 18: Ejemplo de pila del producto

En este caso, el propietario del producto tiene previsto lanzar la versión 1.0 cuando disponga de las cuatro primeras historias, que tienen un esfuerzo estimado en 950 puntos (150+250+250+300).

Además tiene también esbozadas las previsiones para versiones posteriores: 1.1 y 1.2 tal y como muestra la figura siguiente:

Id	Historias	Trabajo	Criterio de validación
1	Historia A 1.0	150	Lorem ipsum dolor sit amet
2	Historia B 1.0	250	consectetuer adipiscing elit
3	Historia C 1.0	250	Aliquam vehicula accumsan tortor
4	Historia D 1.0	300	Pellentesque turpis
5	Historia A 1.1	250	Phasellus purus orci
6	Historia D 1.1	350	penatibus et magnis dis parturi
7	Historia E 1.0	150	Quisque volutpat ante sit amet velit
8	Historia B 1.1	500	Cras iaculis pede eu tellus
9	Historia C 1.1	150	Vestibulum vel diam sed pede
10	Historia E 1.1	200	Suspendisse aliquam felis et turpis
11	Historia F 1.0	TBD	Nullam imperdiet lorem vitae justo
12	Historia A.1.2	TBD	Suspendisse potenti. In nec nunc
13	Historia B 1.2	TBD	Nam eros tellus, facilisis sed, pretium
14	Historia F 1.1	TBD	Morbi arcu tellus, condimentum

Estimación: 950 PUNTOS
Versión 1.0
1.700 PUNTOS
Versión 1.1
2.550 PUNTOS
Versión 1.2

Ilustración 19: Versiones del producto previstas

Para trazar la previsión, se sitúa cada versión en el eje vertical en la posición correspondiente al esfuerzo calculado para construir todas las historias que incluye.

Siguiendo con el ejemplo, la posición de la versión 1.0 se situaría sobre el valor 950 del eje de ordenadas:

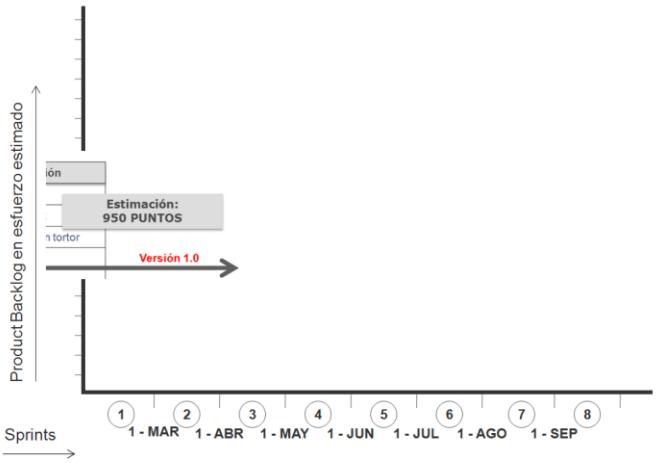


Ilustración 20: Representación de la versión 1 sobre el gráfico de producto

Los puntos de corte que marca esta posición con las líneas de velocidad del equipo (pesimista, realista y optimista) proyectan en el eje horizontal la fecha o sprint en el que se espera completar la versión.

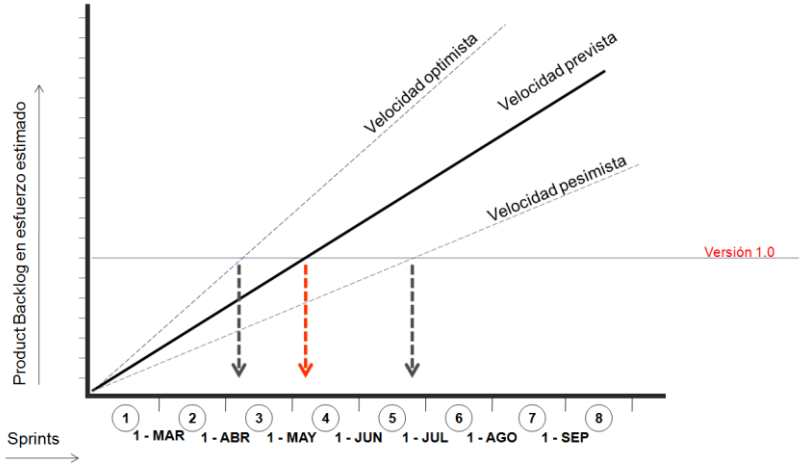


Ilustración 21: Previsión de fechas sobre el gráfico de producto

De igual forma se pueden proyectar las estimaciones tempranas de las futuras versiones previstas.

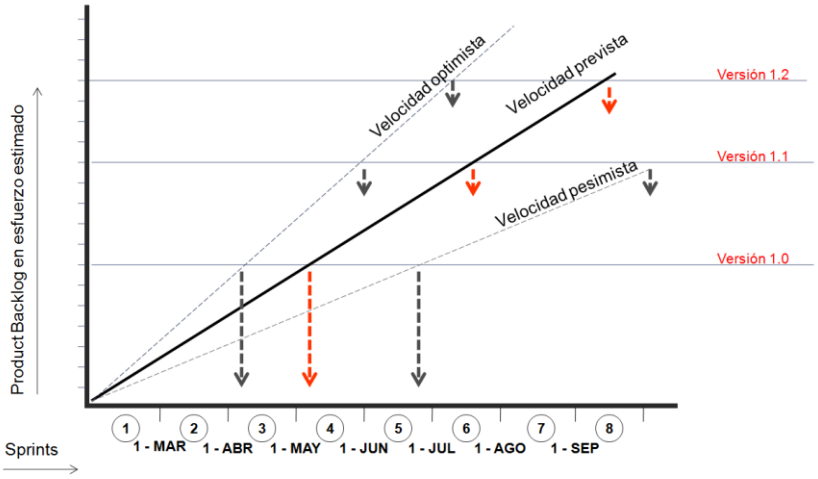


Ilustración 22: previsión de lanzamiento de versiones sobre gráfico de producto

Esta herramienta proyecta la previsión de la pila del producto, que es un documento vivo cuya evolución prevé la del producto.

Como herramienta ágil no debe considerarse como la representación de un plan estable, sino como la previsión de la pila del producto.

Gráfico de avance: monitorización del sprint

También se suele llamar a este gráfico con su nombre inglés: burn-down".

Lo actualiza el equipo en el scrum diario, para comprobar el ritmo de avance, y detectar desde el primer momento si es el previsto, o por el contrario se puede ver comprometida o adelantada la entrega prevista al final de sprint.

La estrategia ágil para el seguimiento del proyecto se basa en:

- Medir el trabajo que falta, no el realizado.
- Seguimiento cercano del avance (diario de ser posible).

Y este gráfico trabaja con ambos principios:

- Registra en el eje Y el trabajo pendiente.
- Se actualiza a diario.

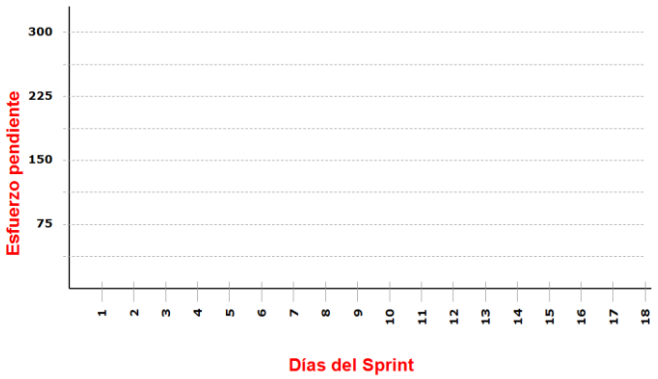


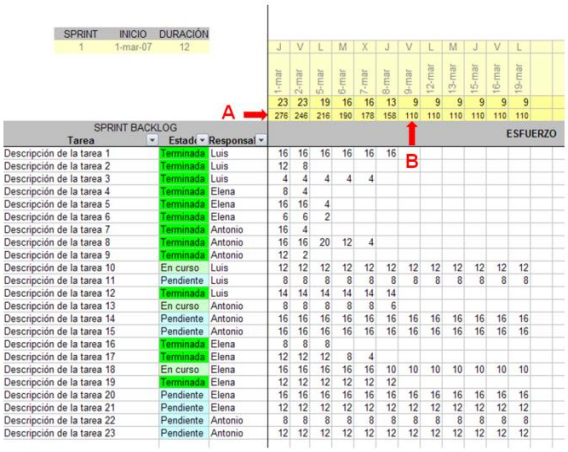
Ilustración 23: Gráfico de avance

El equipo dispone en la pila del sprint, de la lista de tareas que va a realizar, y en cada una figura el esfuerzo pendiente.

Esto es: el primer día, en la pila de tareas figura para cada tarea el esfuerzo que se ha estimado, puesto que aún no se ha trabajado en ninguna de ellas.

Día a día, cada miembro del equipo actualiza en la pila del sprint el tiempo que le queda a las tareas que va desarrollando, hasta que se terminan y van queda 0 como tiempo pendiente.

La figura siguiente muestra un ejemplo de pila en el sexto día del sprint: las tareas terminadas ya no tienen esfuerzo pendiente, y del esfuerzo total previsto para el sprint: 276 puntos (A), en el momento actual quedan 110 (B).



SPRINT	INICIO	DURACIÓN
1	1-mar-07	12

	J	V	L	M	X	J	V	L	M	J	V	L
	1-mar	2-mar	5-mar	6-mar	7-mar	8-mar	9-mar	12-mar	13-mar	15-mar	16-mar	19-mar
	23	23	19	16	16	13	9	9	9	9	9	9
A →	276	246	216	190	178	158	110	110	110	110	110	110

SPRINT BACKLOG

Tarea	Estado	Responsable												ESFUERZO
Descripción de la tarea 1	Terminada	Luis	16	16	16	16	16	16						
Descripción de la tarea 2	Terminada	Luis	12	8										
Descripción de la tarea 3	Terminada	Luis	4	4	4	4	4							
Descripción de la tarea 4	Terminada	Elena	8	4										
Descripción de la tarea 5	Terminada	Elena	16	16	4									
Descripción de la tarea 6	Terminada	Elena	6	6	2									
Descripción de la tarea 7	Terminada	Antonio	16	4										
Descripción de la tarea 8	Terminada	Antonio	16	16	20	12	4							
Descripción de la tarea 9	Terminada	Antonio	12	2										
Descripción de la tarea 10	En curso	Luis	12	12	12	12	12	12	12	12	12	12	12	
Descripción de la tarea 11	Pendiente	Luis	8	8	8	8	8	8	8	8	8	8	8	
Descripción de la tarea 12	Terminada	Luis	14	14	14	14	14	14						
Descripción de la tarea 13	En curso	Antonio	8	8	8	8	8	6						
Descripción de la tarea 14	Pendiente	Antonio	16	16	16	16	16	16	16	16	16	16	16	
Descripción de la tarea 15	Pendiente	Antonio	16	16	16	16	16	16	16	16	16	16	16	
Descripción de la tarea 16	Terminada	Elena	8	8	8									
Descripción de la tarea 17	Terminada	Elena	12	12	12	8	4							
Descripción de la tarea 18	En curso	Elena	16	16	16	16	16	10	10	10	10	10	10	
Descripción de la tarea 19	Terminada	Elena	12	12	12	12	12	12						
Descripción de la tarea 20	Pendiente	Elena	16	16	16	16	16	16	16	16	16	16	16	
Descripción de la tarea 21	Pendiente	Elena	12	12	12	12	12	12	12	12	12	12	12	
Descripción de la tarea 22	Pendiente	Antonio	8	8	8	8	8	8	8	8	8	8	8	
Descripción de la tarea 23	Pendiente	Antonio	12	12	12	12	12	12	12	12	12	12	12	

B

Ilustración 24: Pila del sprint

Con esta información de la pila del sprint se actualiza el gráfico poniendo cada día el esfuerzo pendiente total de todas las tareas que aún no se han terminado.

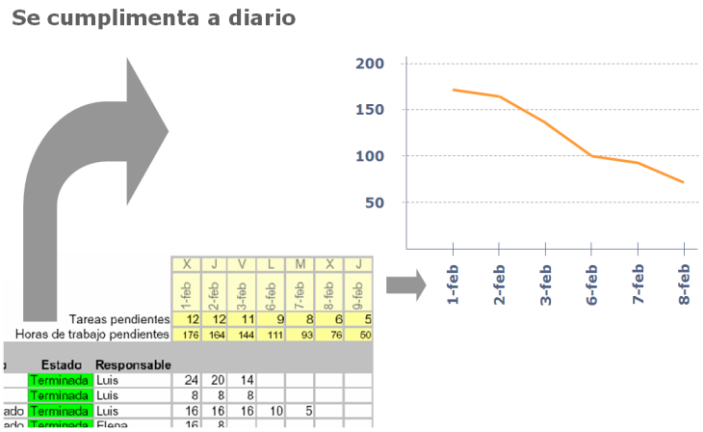


Ilustración 25: De la pila del sprint al gráfico de avance

El avance ideal de un sprint estaría representado por la diagonal que reduce el esfuerzo pendiente de forma continua y gradual hasta completarlo el día que termina el sprint.

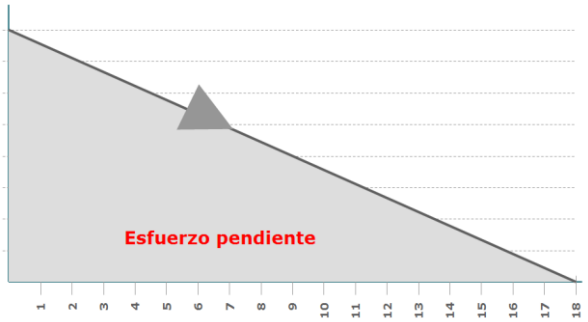


Ilustración 26: Gráfica de avance previsto

Las gráficas de diagonal perfecta no son lo habitual, y la siguiente imagen es un ejemplo de un patrón de avance más normal.

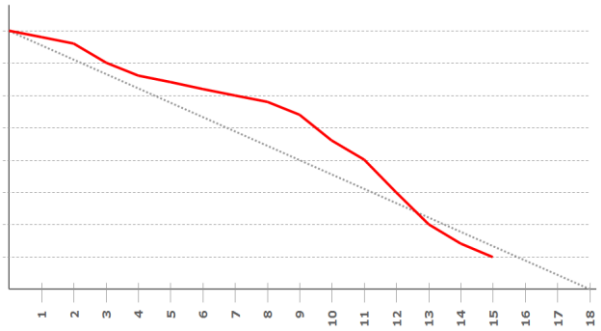


Ilustración 27: Gráfica de avance real

El siguiente sería el aspecto de la gráfica en un “sprint subestimado”

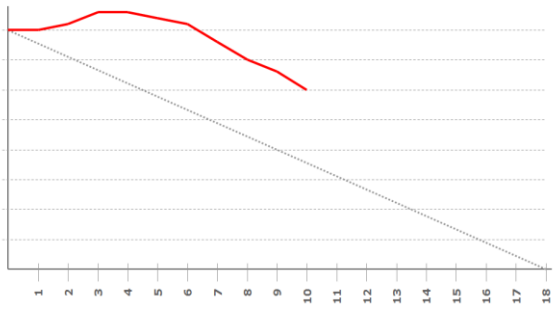


Ilustración 28: Gráfica de avance de un sprint subestimado

La estimación que realizó el equipo en la reunión de inicio del sprint es inferior al esfuerzo real que están requiriendo las tareas.

Y el siguiente sería el patrón de gráfica de un “sprint sobreestimado” .

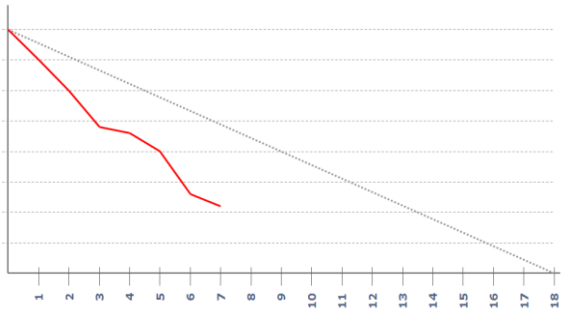


Ilustración 29: Gráfica de avance de un sprint sobreestimado

Estimación de póquer

Es una práctica ágil, para conducir las reuniones en las que se estima el esfuerzo y la duración de tareas.

James Grenning ideó este juego de planificación para evitar discusiones dilatadas que no terminan de dar conclusiones concretas.

El modelo inicial de Grenning consta de 8 cartas, con los números representados en siguiente figura,

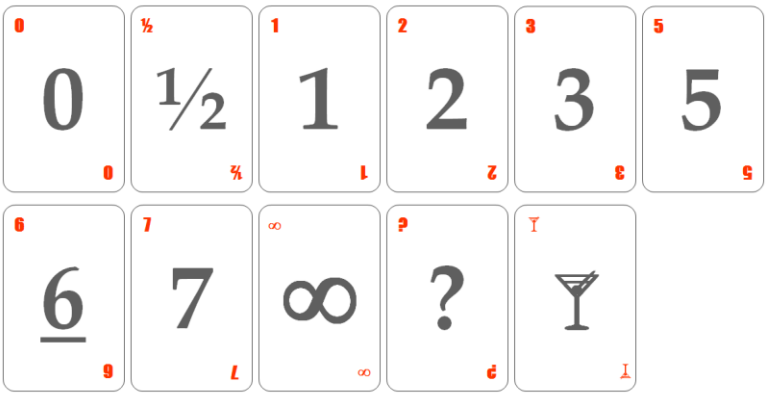


Ilustración 30: Cartas para planificación de póquer

El funcionamiento es muy simple: cada participante dispone de un juego de cartas, y en la estimación de cada tarea, todos vuelven boca arriba la combinación que suma el esfuerzo estimado.

Cuando se considera que éste es mayor de x horas ideales (el tamaño máximo considerado por el equipo para una historia), se levanta la carta "∞".

Las tareas que exceden el tamaño máximo deben descomponerse en subtareas de menor tamaño.

Cada equipo u organización puede utilizar un juego de cartas con las numeraciones adecuadas a la unidad de esfuerzo con la que trabajan, y el tamaño máximo de tarea o historia que se va a estimar.

Variante: sucesión de Fibonacci

Basado en el hecho de que al aumentar el tamaño de las tareas, aumenta también la incertidumbre y el margen de error, se ha desarrollado esta variante que consiste en emplear sólo números de la sucesión de Fibonacci, de forma que:

- El juego de cartas está compuesto por números en sucesión de Fibonacci.
- La estimación no se realiza levantando varias cartas para componer la cifra exacta, sino poniendo boca arriba la carta con la cifra más aproximada a la estimación.

Así, si por ejemplo una persona cree que el tamaño adecuado de una tarea es 6, se ve obligado a reconsiderar y, o bien aceptar que el tamaño puede ser 5, o bien aceptar una estimación más conservadora y levantar el 8.

Para estimar tareas puede ser válido un juego de cartas como éste:

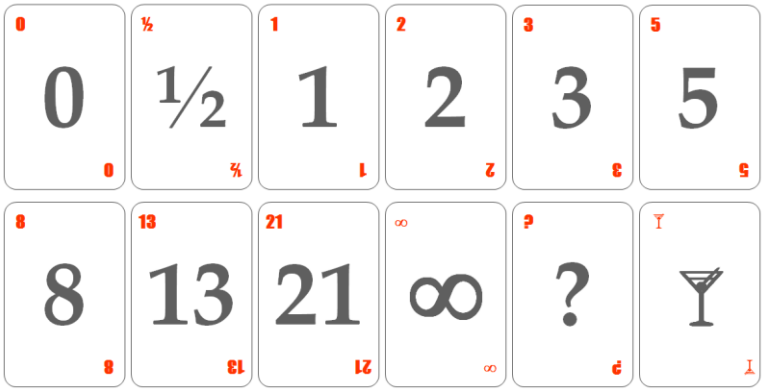


Ilustración 31: Cartas para estimación de póquer (Fibonacci)

Es frecuente emplear 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.

Operativa

- Cada participante de la reunión tiene un juego de cartas.
- Para cada tarea (historia de usuario o funcionalidad, según sea el nivel de requisitos que se va a estimar) el cliente, moderador o propietario del producto expone la descripción empleando un tiempo máximo.
- Hay establecido otro tiempo para que el cliente o propietario del producto atienda a las posibles preguntas del equipo.
- Cada participarte selecciona la carta, o cartas que representan su estimación, y las separa del resto, boca abajo.
- Cuando todos han hecho su selección, se muestran boca arriba.
- Si la estimación resulta "infinito", por sobrepasar el límite máximo establecido, la tarea debe dividirse en sub-tareas de menor tamaño.
- Si las estimaciones resultan muy dispares, quien asume la responsabilidad de gestionar la reunión, con su criterio de gestión, y basándose en las características del proyecto, equipo, reunión, nº de elementos pendientes de evaluar, puede optar por:
    - Preguntar a las personas de las estimaciones extremas: ¿Por qué crees que es necesario tanto tiempo?, y ¿por qué crees que es necesario tan poco tiempo? Tras escuchar las razones, repetir la estimación.
    - Dejar a un lado la estimación de esa tarea y retomar al final o en otro momento aquellas que hayan quedado pendientes.

- Pedir al cliente o propietario del producto que descomponga la funcionalidad y valorar cada una de las funcionalidades resultantes.
- Tomar la estimación menor, mayor, o la media.

Este protocolo de moderación, evita en la reunión los atascos de análisis circulares en ping-pong entre diversas opciones de implementación, hace participar a todos los asistentes, reduce el cuarto de hora o la media hora de tiempo de estimación de una funcionalidad, a escasos minutos, consigue alcanzar consensos sin discusiones, y además resulta divertido y dinamiza la reunión.

SEGUNDA PARTE

Scrum en su concepción original.

1.- Conocimiento en continua evolución

Los marcos de prácticas ágiles no llegan a los proyectos TIC como “tesis” de conocimiento, sino como “antítesis” al que la Ingeniería del Software venía desarrollando.

Comenzaremos viendo qué significa esto, y así tomar la distancia necesaria para ver con perspectiva las razones por las que los proyectos TIC abrazaron la agilidad a finales del siglo pasado, y sus diferencias con la ingeniería de procesos; no desde las prácticas concretas, sino desde los principios en los que se basan, y con ello comprender las fortalezas y debilidades de la agilidad.

Alcanzar una visión de las razones y los principios de cada metodología, más allá de la concreción de un modelo es clave para dar el salto de gestión técnica a gestión experta. Esto es, de gestión basada en la aplicación de prácticas a gestión basada en la aplicación del propio conocimiento.

Gestión técnica: Gestión basada en la aplicación de modelos de prácticas y procesos.

Gestión experta: Gestión basada en el conocimiento tácito del gestor

El patrón dialéctico

Al cuestionar el conocimiento, se inicia su evolución que sigue un patrón dialéctico de: tesis, antítesis y síntesis.

De manera esquemática el patrón dialéctico puede definirse como el ritmo de avance que contrapone una ***antítesis*** a una concepción previa, entendida como ***tesis***. La antítesis muestra los problemas y contradicciones de la tesis, y de la confrontación surge un tercer momento llamado ***síntesis***, una resolución o una nueva comprensión del problema.

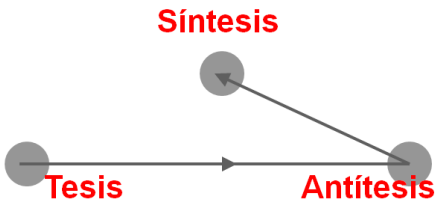


Ilustración 32: Patrón dialéctico

De esta forma la estrategia de abordar con ingeniería de procesos los retos de los proyectos de software, supuso la primera tesis para dar respuesta a la “crisis del software”, y sus problemas y contradicciones han sido puestos de manifiesto por su antítesis: la agilidad.

En 1968, en la primera conferencia sobre desarrollo de software celebrada por la organización OTAN, se analizaron los problemas de la programación del software, y en ella se acuñó el término “crisis del software” para referirse a ellos.

La conclusión de la conferencia (Bauer, Bolliet, & Helms, 1969) fue la necesidad de crear una disciplina científica que, como ocurría en otras áreas, permitiera aplicar un enfoque sistemático disciplinado y cuantificable al desarrollo, operación y mantenimiento de los sistemas del software, es decir, la aplicación de la ingeniería de procesos al software. Fue el nacimiento de la Ingeniería del Software.

La primera estrategia de la Ingeniería del software (tesis) se ha basado en dos pilares:

- Ingeniería de procesos:
- Gestión predictiva:

El primero para aplicar el principio básico de calidad contrastado con éxito en los entornos de producción industrial: “la calidad del resultado depende de la calidad de los procesos empleados”.

El segundo para garantizar el cumplimiento de agendas y presupuestos.

Mientras esta 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.

¿La planificación predictiva es apropiada para cualquier proyecto? ¿Los criterios de éxito son siempre el cumplimiento de fechas, costes y funcionalidades preestablecidas?

Empiezan a surgir proyectos cuya finalidad no es construir un sistema previamente definido y planificado en su totalidad, y para los que no es realista trazar un plan cerrado desde el inicio. Proyectos en los que no interesa saber si el sistema final tendrá 20 o 200 funcionalidades, ni conocer cómo serán éstas en detalle: Su interés es poner una novedad en el mercado lo antes posible, y desde ese momento evolucionar la visión y valor de forma continua.

Por otra parte también se cuestiona si el software se puede producir con patrones de procesos industriales, y se empieza a aceptar que en la calidad del resultado puede ser más importante el conocimiento tácito de la persona que lo realiza que el *know-how* aportado a través del proceso y la tecnología empleada.

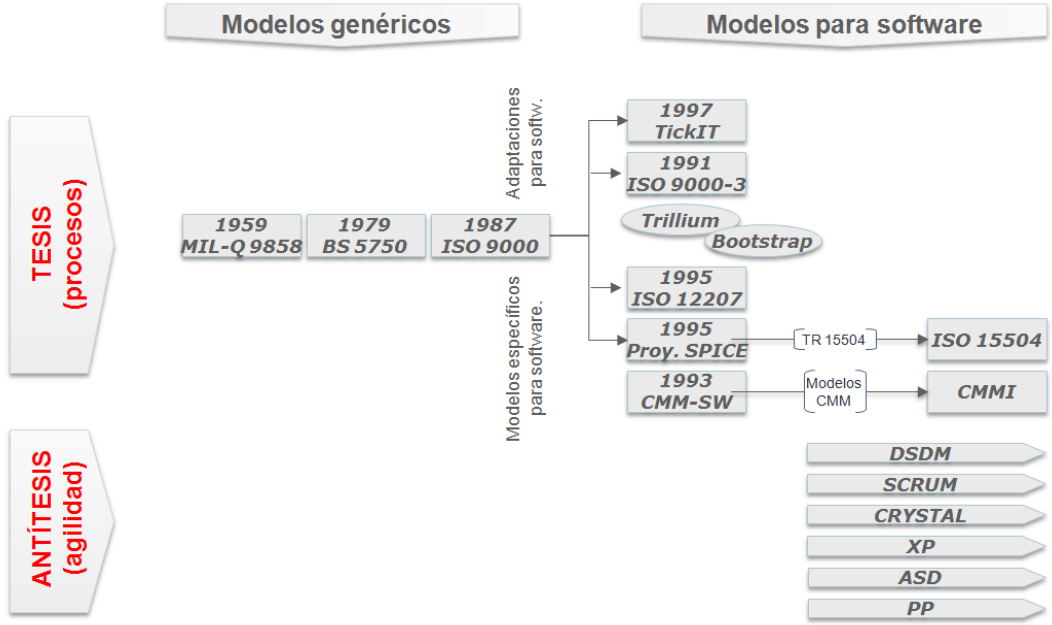


Ilustración 33: Evolución de los primeros modelos de mejora

Desde los orígenes de la agilidad, a mediados de los 90, hasta 2005-2010 han sido habituales las posturas radicales entre los defensores de los modelos de procesos y de los marcos ágiles, posiblemente más enfocados en descalificar al otro que en revisar y depurar los propios métodos.

Algunos ejemplos de esta tensión:

"La diferencia entre un atracador de bancos y un teórico de CMM es que con el atracador se puede negociar"…

"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., 2003)

"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)

Espiral dialéctica del conocimiento.

El conocimiento profesional evoluciona de forma continua porque la realidad en la que se aplica está en permanente movimiento, y también porque la mejora siempre es posible.

La puesta en funcionamiento de nuevas técnicas, procesos o modelos genera sus propias antítesis al revelarse las debilidades, contradicciones y puntos de mejora, y el enfrentamiento de ambos conduce hacia una síntesis, que pasará a ser una nueva tesis, cuyo posterior uso producirá su antítesis, desarrollando a través de este patrón dialéctico una espiral de evolución continua del conocimiento (Nonaka & Takeuchi, 2004)

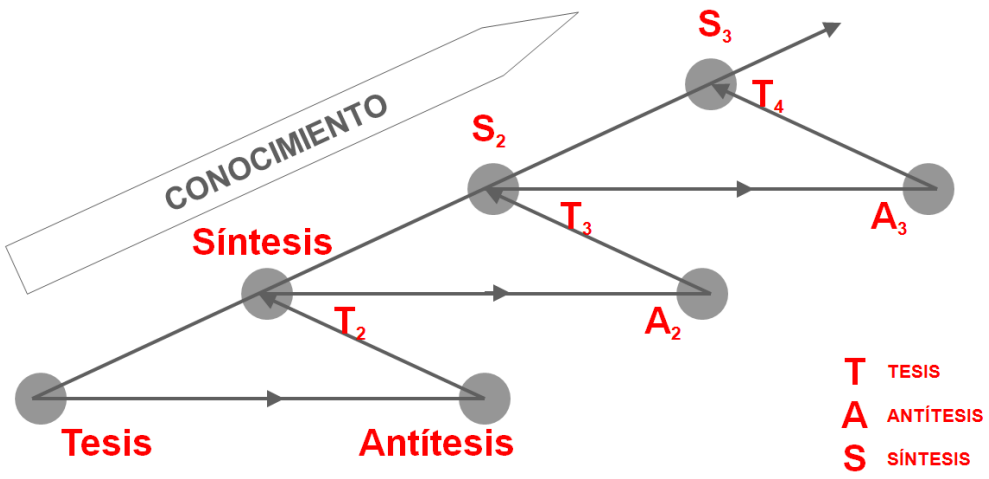


Ilustración 34: Espiral dialéctica del conocimiento

En disciplinas no técnicas y en generaciones anteriores el ritmo de avance sobre esta espiral dialéctica permitía a los profesionales desempeñarse con los conocimientos adquiridos en su licenciatura durante toda su carrera profesional. Sin embargo hoy esto no es posible, en especial, en el sector TIC

No hay métodos, prácticas o modelos de trabajo que nos ayuden con solvencia durante mucho tiempo, sino conocimiento en evolución. Esta es una consideración clave en el marco de Scrum Manager y la razón por la que no define un modelo fijo, sino un conocimiento actualizado como base para una gestión más experta que técnica. Más basada en el criterio documentado y experto del gestor que en la aplicación de prácticas o procesos.

2.- Empresa como sistema

Las empresas no están formadas por departamentos o áreas más o menos independientes. Son realidades sistémicas, y su eficiencia es proporcional a la armonía de los modos de trabajo de los diferentes departamentos. La consideración de scrum como el marco de reglas de trabajo propias del ámbito de gestión de proyectos, cuyas prácticas se pueden adoptar sin implicaciones en el resto de la organización produce resultados limitados e incluso contraproducentes.

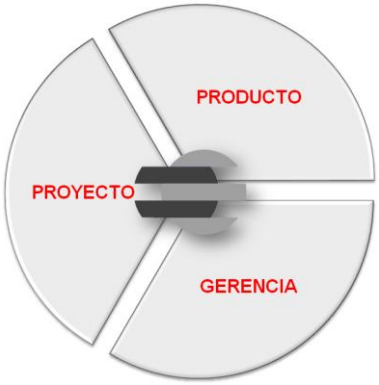


Ilustración 35: La empresa como sistema

Por ejemplo, en una organización cuya gerencia dirige con orientación a modelos de producción industrial, y el área de ingeniería en consecuencia trabaja con modelos basados en procesos con ciclos de vida secuenciales o de cascada, la adopción de prácticas ágiles en el área de gestión de proyectos generará problemas de funcionamiento.

3.- Flexibilidad

El objetivo no es implantar un marco de scrum basado en reglas. El objetivo es alcanzar una organización ágil en su conjunto, capaz de “avanzar en scrum” en su concepción original. Capaz de responder en escenarios de trabajo que evolucionan rápidamente, o tienen dosis altas de incertidumbre por las que no cuentan con requisitos estables al concebir nuevos productos o servicios. Se trata de clientes que necesitan empezar a usar un producto lo antes posible y mejorarlo de forma continua. De productos en los que la innovación es un valor clave.

Un principio básico de la implantación pragmática de scrum es la flexibilidad, que consiste en que las prácticas de scrum se adapten a la organización y no al revés. Se trata en definitiva de realizar una gestión experta más que una gestión técnica. Una gestión dirigida desde el conocimiento, experiencia y criterio del gestor y no tanto una gestión orientada a la búsqueda e implantación del mejor modelo. Una gestión basada en la persona antes que en el modelo.

El conocimiento de las distintas técnicas y metodologías amplía el criterio y el fondo de recursos del gestor.

Para seguir la evolución del conocimiento profesional y para ampliar y mejorar de forma continua el criterio e inventario de recursos profesionales propios es aconsejable:

- Vencer la resistencia al cambio y evitar actitudes de adopción o defensa dogmática de un modelo.
- Espíritu crítico-constructivo: Cuestionar continuamente de forma “antitética” los modos actuales, con el conocimiento y criterio profesional adecuar el sistema de trabajo propio a las características del proyecto, equipo y organización.

Scrum pragmático

Scrum Pragmático

Adaptar las prácticas scrum a las circunstancias de la propia organización, permite emplear técnicas de incremento continuo o iterativo; tableros kanban con el formato más adecuado a cada proyecto, y en general las prácticas y reglas que mejor encajan en las circunstancias de cada caso.

De esta forma se van abandonando los renglones de guía de las reglas definidas, y aplicando directamente los valores de scrum.

Scrum técnico

Reglas

Marco de reglas para desarrollo de software

Autores: Ken Schwaber y Jeff Sutherland
"Scrum Development Process OOPSLA'95" 1995

Aplicación de reglas definidas

Roles

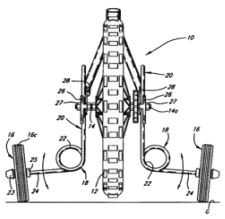
- Dueño de producto
- Equipo de desarrollo
- Scrum Master

Eventos

- El Sprint
- Reunión de planificación
- Scrum diario
- Revisión de sprint
- Retrospectiva de sprint

Artefactos

- Pila de product
- Pila de sprint
- Incremento



Aprender las reglas de Scrum

Scrum pragmático

Valores

Concepto original Scrum

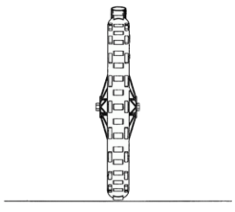
Autores: Hirotaka Takeuchi e Ikujiro Nonaka
"The New New Product Development Game" 1986

Aplicación de valores ágiles

- Personas > procesos
- Resultado > documentación
- Colaboración > negociación
- Cambio > planificación

… "Para avanzar en Scrum"

- Incertidumbre
- Autoorganización
- Fases de desarrollo solapadas
- "Multiaprendizaje"
- Control sutil
- Difusion del conocimiento



Aprender a avanzar en Scrum sin reglas

Responsabilidades

Al pasar del scrum técnico, basado en reglas, al scrum pragmático, para aplicar directamente principios de gestión ágil con en el conocimiento y experiencia de los equipos, y con una cultura, ya ágil en la organización, el ámbito de responsabilidades que se deben cubrir va más allá de los roles de proyecto:

La organización, como realidad sistémica debe dar respuesta de forma coordinada y alineada con su visión, a responsabilidades en tres áreas: Gerencia, procesos y producción.

Ilustración 36: Áreas de responsabilidad Scrum Manager

De gerencia

- Equilibrio sistémico de la organización
- Coherencia del modelo
- Medios y formación

De procesos

- Configuración flexible de scrum
- Mejora continua
- Garantía de funcionamiento de scrum en cada proyecto ***(en scrum técnico asignada al rol de Scrum Master)***

De producción

- Producto***(en scrum técnico asignada al rol de Propietario del producto)***
- Auto-organización ***(en scrum técnico asignada al equipo)***
- Tecnología ágil ***(en scrum técnico asignada al equipo)***

El uso de prácticas y tecnologías ágiles, el trabajo en equipos autoorganizados, disponer de una visión de producto definida y gestionada durante todo el proyecto, y garantizar el funcionamiento de scrum durante la ejecución, son responsabilidades que pertenecen al ámbito del proyecto.

Que las diferentes áreas de la empresa se encuentren comunicadas y alineadas con una visión común, coherente con un modelo de trabajo ágil, dispongan de medios para el diseño e implantación de una implantación ágil adecuada a la empresa, mejora continua del modelo y formación para las personas, son responsabilidades en el ámbito de la organización.

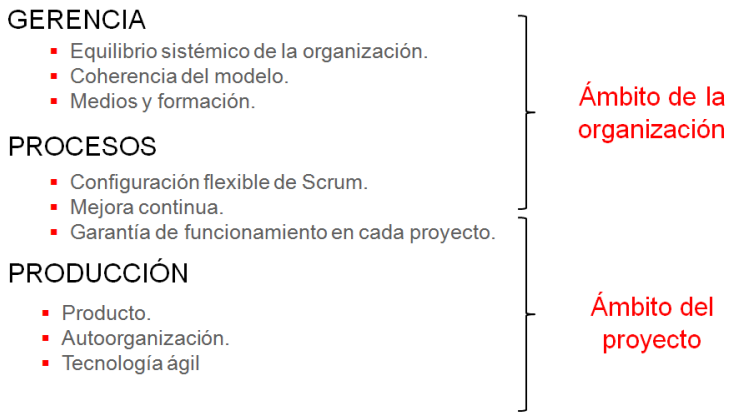


Ilustración 37: Ámbitos de responsabilidad Scrum Manager

En scrum técnico, las responsabilidades del ámbito del proyecto las asumen roles definidos:

- La responsabilidad de funcionamiento de scrum se asigna a un rol de gestor específico para el funcionamiento de scrum: Scrum Master.
- La responsabilidad de visión y gestión del producto al rol específico de propietario del producto, o product owner.
- La responsabilidad de autoorganización y uso de prácticas y tecnologías ágiles es propia del equipo.

Lo más aconsejable en fases de implantación, en equipos no familiarizados con desarrollo ágil es la adopción del modelo de roles de scrum técnico.

En la evolución hacia un nivel más maduro y global de agilidad en la organización es aconsejable adaptar el marco de scrum a la realidad de la organización, de forma que lo relevante no sea la presencia de determinados roles y reglas, sino cubrir adecuadamente todas las responsabilidades necesarias a nivel de organización.

Un ejemplo de asignación flexible de las responsabilidades del ámbito de proyecto sobre el esquema de puestos ya existente en la organización podría ser:

- Garantía de funcionamiento de scrum => Calidad o procesos
- Garantía de gestión de producto => Product manager
- Autoorganización y tecnología ágil = Equipo

Tanto si en la implantación de agilidad, las responsabilidades necesarias se asignan a roles de la estructura de la empresa, o se crean nuevos puestos (Propietario de producto o Scrum Master), lo importante es que las personas que los desempeñan tengan la experiencia y conocimiento profesional necesario.

Metodologías

Mapa de metodologías.

Desde los 80 se han desarrollado tantos modelos de procesos, marcos y prácticas de trabajo para mejorar la calidad y eficiencia en los proyectos de software, que resulta útil trascender las etiquetas y llegar a la base de los principios que subyacen, y las estrategias con las que los desarrollan; de forma que usando como coordenadas tres conceptos: “desarrollo, trabajo y conocimiento”, y dos modelos de gestión: “predictiva y evolutiva” se despeja y simplifica el aparente laberinto de modelos de procesos, marcos o prácticas de trabajo a los que nos referimos: CMM-SW, CMMI, PMBOK, DSDM, Crystal, ISO 15504, RUP, XP, scrum, ITIL, ASD, PRINCE 2, LEAN, KANBAN, TDD, etc..

Las diferentes prácticas y metodologías responden a combinaciones de tres conceptos y dos patrones de gestión de proyectos.

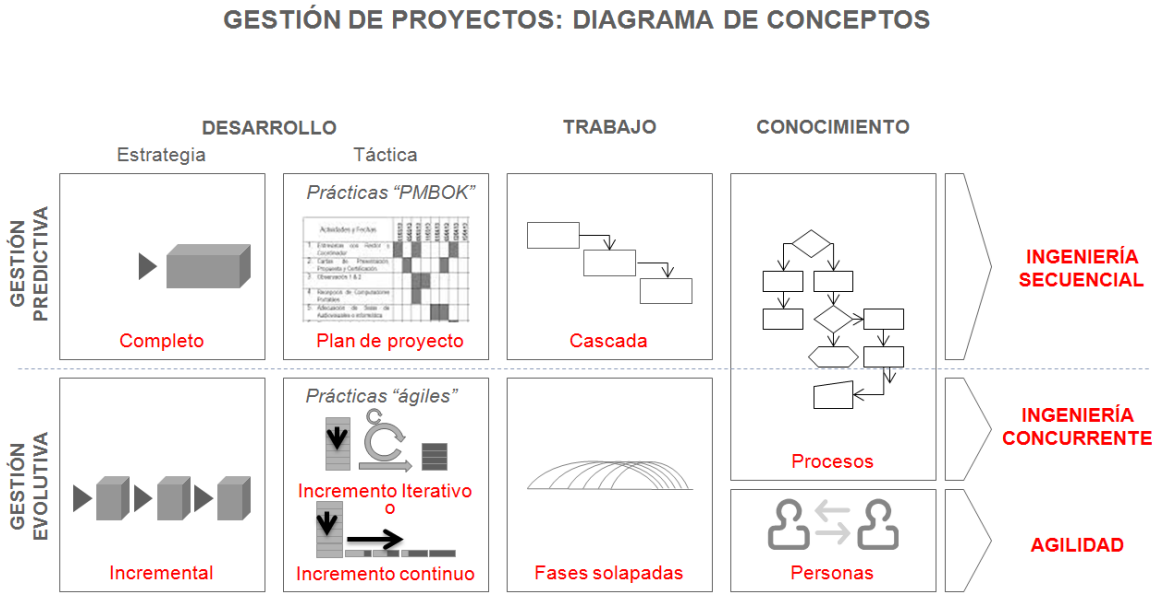


Ilustración 38: Diagrama de conceptos de la gestión de proyectos

Conceptos

1.- Desarrollo

Completo: La descripción de lo que se desea obtener está disponible al inicio del proyecto, es completa y detallada, sirve de base para estimar el plan del proyecto: tareas, recursos y agenda de trabajo. Durante la ejecución se gestiona su cumplimiento.

Incremental: La descripción de lo que se desea obtener no está disponible de forma completa y detallada al inicio: se complementa y evoluciona en paralelo al desarrollo, que genera el resultado de forma incremental y 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 de forma cíclica y continua. Este es el marco de producción empleado al aplicar el marco estándar de scrum, que define como sprint a cada iteración de desarrollo, al final de la cual se produce un incremento del producto.

2.- Trabajo

Secuencial (cascada): Divide el trabajo en fases, y cada fase comienza al terminar la anterior. El ejemplo más habitual es el ciclo de cascada definido en Ingeniería del software con las fases de requisitos, análisis, diseño, codificación, pruebas e implementación.

Concurrente: Solapa en el tiempo las diferentes fases. Siguiendo con el ejemplo de ingeniería de software, la definición de requisitos, el análisis, la codificación y el despliegue del resultado se realiza y revisa de forma simultánea y continua.

3.- Conocimiento

¿Dónde se encuentra el principal conocimiento empleado, en la correcta ejecución del proceso o en el saber hacer de la persona?

Producción basada en procesos: El conocimiento o know-how, responsable de la calidad del resultado se encuentra en mayor medida en los procesos y la tecnología empleada: “La calidad del resultado depende de la calidad de los procesos empleados“.

Producción basada en personas: El conocimiento o know-how responsable de la calidad del resultado se encuentra en mayor medida en el “saber hacer” tácito de las personas que lo construyen.

Patrones de gestión del proyecto

Gestión predictiva

Modelo de gestión cuyo objetivo es ofrecer resultados predecibles: desarrollo del producto previsto, en el tiempo previsto, e invirtiendo los recursos previstos. Emplea una estrategia de desarrollo completo con prácticas de planificación tradicional. Los principales referentes en el desarrollo de conocimiento para este tipo de gestión son PMI e IPMA y los modelos de procesos CMMI, ISO 15504, SPICE, entre otros, que emplean ingeniería secuencial y producción basada en procesos.

Gestión evolutiva

Modelo de gestión cuyo objetivo es entregar lo antes posible un producto mínimo viable, e incrementar su valor de forma continua. Emplea una estrategia de fases de trabajo solapadas, y desarrollo incremental, que se puede obtener con tácticas iterativas o de mantenimiento de flujo continuo. Puede emplearse 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 tareas.

Agilidad y gestión evolutiva no son lo mismo. Se puede hacer gestión evolutiva empleando agilidad o empleando ingeniería concurrente

Personas, Procesos y Tecnología

Scrum Manager reconsidera dos vértices del triángulo clásico de los factores de producción: Personas - Procesos y Tecnología. El de procesos y el de personas.

Procesos

Para diferenciar los ~~procesos~~ [2] procedimientos en sus dos tipos posibles, podemos decir que en uno, las personas ayudan al proceso, y en el otro son ~~los procesos~~ las prácticas las que ayudan a las personas.

En el primer caso el proceso es el protagonista, el que sabe cómo hacer el trabajo, y la persona se integra en el sistema como instrumento, como operario de apoyo.

En el segundo, el artífice es la persona y ~~el proceso~~ la práctica una ayuda, una herramienta que simplifica aspectos rutinarios para que pueda lograr más eficiencia y no diluir el esfuerzo en rutinas mecánicas.

Por eso a los primeros los llamamos procesos y a los segundos prácticas.

La principal diferencia entre unos y otros es el tipo de conocimiento con el que trabajan.

El conocimiento pueden ser:

- ***Explícito:*** contenido en los procesos y la tecnología
- ***Tácito:*** que es contenido por la persona

Scrum Manager aporta una consideración sobre el triángulo tradicional personas-procesos-tecnología, considerando que los procedimientos de trabajo pueden ser:

- ***Procesos:*** Si su ejecución aporta conocimiento clave para lograr el resultado. Son por tanto contenedores de conocimiento "explicitado" en el proceso y la tecnología que emplea.
- ***Prácticas:*** Si el procedimiento ayuda a las persona, que es quien aporta con su conocimiento tácito, el "saber hacer" clave para lograr el resultado.

Se puede decir que en los primeros la persona ayuda al procedimiento, y en los segundos es el procedimiento el que ayuda a la persona.

Ilustración 39: Personas, procedimientos y tecnología

Los modelos de ingeniería de procesos, consideran al binomio "proceso-tecnología" como principal responsable de la calidad del resultado. Su antítesis, la agilidad, da el protagonismo del resultado a las personas.

[2] No los llamaremos procesos sino "procedimientos" dejando así el nombre "proceso" para el procedimiento que tiene explicitado en él el principal conocimiento para la obtención del resultado

Desde el punto de vista de Scrum Manager, ambas opciones son válidas, pero para tipos de trabajos distintos. En entornos de producción industrial, las personas aportan trabajo para ejecutar y supervisar los procesos. Sin embargo para las empresas del conocimiento que trabajan en escenarios rápidos e innovadores, las personas aportan con su talento el *know-how* que da valor al resultado.

Personas

Las organizaciones que necesitan imprimir un componente innovador importante y frecuente, o que se mueven en sectores de innovación muy rápido, obtienen mejores resultados si hacen responsables de esa innovación al talento de las personas más que a la ejecución de procesos.

En este tipo de organizaciones es importante asegurar, además del nivel de creatividad del equipo, su capacidad para aprehender. El modelo de conversión del conocimiento definido por Nonaka y Takeuchi (Nonaka & Takeuchi, The Knowledge-Creating Company, 1995) define con sus 4 fases el proceso para la adquisición de las personas del conocimiento tácito a través de compartir experiencias, comunicación directa, documentos, manuales y tradiciones, que añade conocimiento novedoso a la base colectiva de la organización.

Gestión visual kanban para scrum

Adaptación de las prácticas a la organización.

Gestión visual kanban para obtener incremento continuo.

La imagen siguiente muestra dónde se sitúa scrum técnico, según lo descrito en el capítulo anterior:

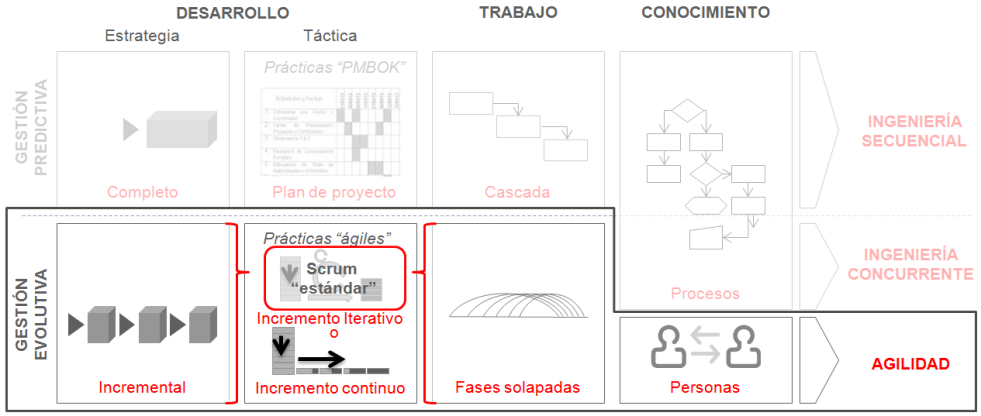


Ilustración 40: Agilidad con desarrollo incremental iterativo

Su característica principal es el uso de ***pulsos de sprint***, para emplear tiempo prefijado (*timeboxing*) como motor de avance al ritmo marcado por la secuencia de sprints.

La táctica de *timeboxing* ayuda al equipo a avanzar, al tiempo que mitiga la tendencia habitual a dilatar los tiempos de entrega previstos.

Los equipos originales de scrum observados y descritos por Nonaka y Takeuchi (Nonaka & Takeuchi, The New New Product Development Game, 1986) y los principios de la agilidad no prescriben el uso de una determinada táctica para lograr un desarrollo incremental. De hecho también es posible trabajar con un avance constante de las tareas una tras otra, sin empaquetar en incrementos.

Lograr un flujo continuo de tareas sin usar sprints no es fácil porque suelen formarse cuellos de botella que bloquean el avance, mientras en otras áreas del equipo se producen tiempos muertos sin tareas que realizar.

La gestión visual kanban es la técnica más empleada actualmente para regular un flujo de avance continuo en proyectos TIC y de servicios del conocimiento gestionados evolutivamente sin sprints.

Introducción: De la artesanía a la producción lean

Las técnicas de gestión visual que vamos a ver se inspiran en la producción lean. Esta introducción muestra de forma breve cuáles han sido los principales hitos en la producción industrial hasta llegar a lean.

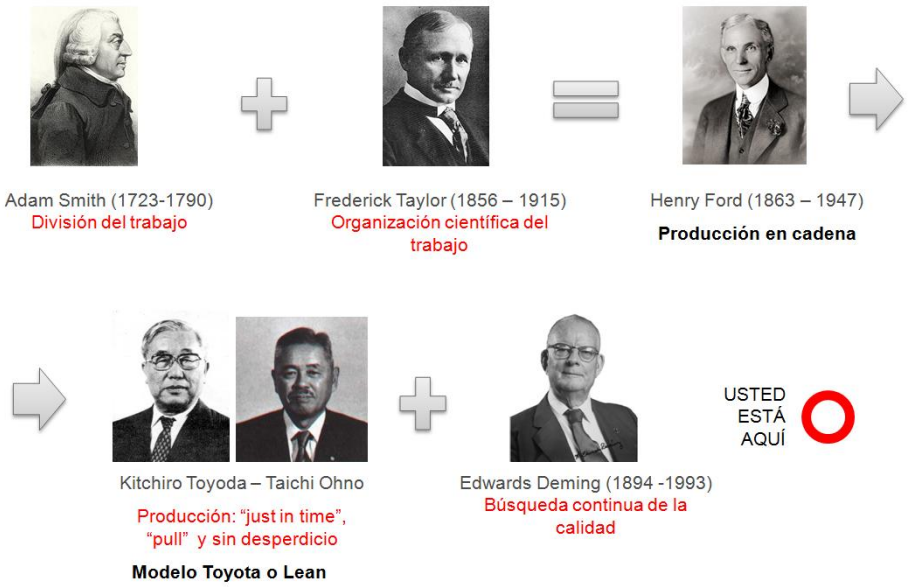


Ilustración 41: De la artesanía a la producción lean

División del trabajo

"La riqueza de las naciones" (Smith, 1776) es la obra más célebre de Adam Smith. Fue publicada en 1776 y se considera el primer libro moderno de economía. En él desarrolla la teoría económica sobre la división del trabajo.

Para Smith el principal factor para mejorar la productividad del trabajo es su división, que ilustró con su célebre ejemplo de la manufactura de alfileres, en el que comparaba la producción que puede alcanzar un herrero realizando todas las tareas necesarias, con la obtenida en un sistema con división del trabajo entre obreros especializados en cada tipo de tarea (estirado del alambre, cortado, afilado, etc.).

Como demostró, la división del trabajo hace posible producir 5.000 alfileres diarios por obrero, frente a los 50 que produciría un artesano.

Organización científica del trabajo

Frederick W. Taylor fue el primero en analizar y estudiar el trabajo de forma sistemática.

Taylorismo es el nombre dado al método de producción cuyo principal objetivo fue el aumento de la productividad, basado en la división y especialización del trabajo. Al taylorismo se le denomina también "organización científica del trabajo" o "gestión científica del trabajo" por aplicar principios básicos del método científico en el diseño de los procesos de trabajo.

Taylor expuso su sistema de organización racional del trabajo en su obra "Principles of Scientific Management" (Taylor, 1911) que de forma somera se puede condensar en cuatro principios:

- Reemplazar los métodos artesanales por métodos basados en el análisis científico del trabajo.
- Seleccionar y formar a los empleados con criterios científicos, en lugar de dejar que aprendan de forma autónoma.
- Dividir las tareas en gestión y trabajo, de forma que los gerentes puedan gestionar los principios de planificación y ejecución del trabajo.
- Proporcionar instrucciones y supervisión detallada a cada trabajador.

Producción en cadena

La producción en cadena, también denominada *fordismo*, es el sistema de producción desarrollado por el fabricante de automóviles Henry Ford para la fabricación de los primeros automóviles de su factoría en la primera década del siglo XX.

Pone en práctica los principios de la división y la organización científica del trabajo, y ha sido ampliamente utilizado para la producción industrial hasta que en la década de los 70 comenzó a ser reemplazada por el *toyotismo*.

El *fordismo* hizo posible:

- Reducción significativa del costo de producción. (El precio del "Ford T" pasó de $850 en 1908 a 250$ al producirse en cadena en 1927).
- Flujo constante de la producción.
- Ingeniería de procesos: la calidad del resultado depende de la calidad del proceso empleado en su fabricación.

Sistema de producción Toyota

El fundador de Toyota, Sakichi Toyoda, su hijo Kiichiro Toyoda y el ingeniero Taiichi Ohno, evolucionaron los sistemas de producción de sus factorías transformando el fordismo hasta llegar a principios de los 70 en lo que acabaría siendo y denominándose "Sistema de Producción Toyota" o toyotismo.

Cuando intentaron importar los sistemas de producción masiva norteamericanos para rehacer su industria tras la segunda guerra mundial, se dieron cuenta de que el fordismo no les valía. Creaba empresas para producir grandes cantidades del mismo producto, pero no servía para producir pequeñas cantidades de productos diferentes.

El sistema fordista, al producir muchas existencias del producto generaba sobreproducción y almacenes llenos ante la falta de clientes. Necesitaban un sistema que fabricara con costos reducidos, pero no a costa de incrementar la producción, puesto que no tenían gran demanda.

Estas circunstancias evolucionaron la producción hacia un sistema que elimina el número de existencias en el proceso de producción y el "sobreefectivo" tanto de personal como de equipo para lograr la fábrica mínima, capaz de producir lo necesario: lo que un cliente está esperando.

El sistema fordista es un sistema de empuje ("push") mientras que el toyotismo es un sistema de arrastre ("pull"). En el fordismo cada puesto empuja continuamente al siguiente, enviándole trabajo de forma sistemática. Sobre la estimación de consumo, se pone en marcha la cadena y se produce al ritmo previsto.

El toyotismo es un sistema de arrastre: un puesto no empuja al siguiente sino que tira del anterior, pidiéndole trabajo por la demanda de un cliente.

Para lograr la simplicidad organizativa y flexibilidad que permitan al sistema adaptarse a variaciones del flujo de demanda y diferencias en el producto, el toyotismo emplea diversas técnicas de apoyo:

- Información y control visual kanban.
- Información y control visual andon.
- Sistemas de prevención de errores Poka-yoke.
- Herramientas polivalentes y de cambio rápido.

El toyotismo o "Sistema de Producción Toyota" es conocido actualmente como Lean.

Lean

La palabra “lean” en inglés significa “magra”, es decir, sin grasa.

> Lean es un sistema de mejoramiento de procesos de manufactura basado en la eliminación de desperdicios y actividades que no agregan valor al proceso.

14 principios Lean

1. Las decisiones del negocio están basadas en una **visión a largo plazo**, aún a expensas de las pérdidas financieras a corto.
2. Los ciclos son cortos y rápidos.
3. Se prefieren los sistemas ***“pull”***, que evitan la sobreproducción.
4. La carga de trabajo debe ser balanceada (Heijunka).
5. La cultura lean comprende **detener la producción para arreglar problemas**, así como en enseñar el estudio metódico de los problemas (Jidoka).
6. Las tareas se estandarizan para lograr la **mejora continua** (Kaizen).
7. La **gestión visual simple** revela problemas y permite la coordinación.
8. Se utiliza solamente **tecnología probada** que pueda ser provechosa para la gente y su proceso.
9. Se forman ***líderes*** que comprendan el trabajo, vivan la filosofía de la empresa y la enseñen a otros.
10. Se desarrollan ***equipos y personas excepcionales que siguen la filosofía de la compañía.***
11. Se respeta la red de ***colaboradores y proveedores (Keiretsu) ,*** desafiándolos a crecer y ***ayudándolos a la mejora.***
12. *Se valora que los responsables* ***vayan y miren las situaciones en el lugar de trabajo, para entenderlas y poder ayudar.***
13. Decisiones basadas en el ***consenso y la consideración minuciosa de todas las opciones, y su posterior implementación*** rápida.
14. Empresa como organización que aprende a través de la ***reflexión constante (Hansei) y de la mejora continua (Kaizen)***

Algunas prácticas comunes en la producción lean

He aquí algunas de las prácticas que se utilizan en entornos lean:

- **Kanban**: o presentación de información visual relativa a la producción (identificación de componentes, estado del proceso, etc) presentada físicamente en el lugar de trabajo implicado y permanentemente actualizada.
- **5S**: Metodología de trabajo cuyo nombre procede de las 5 palabras japonesas que la configuran: seiri, sieton, seiso, siketsu and shitsuke (clasificar, ordenar, limpiar, estandarizar y sostener).
- **Poka-yoke**: o sistema a prueba de error, que busca crear mecanismos que sólo permiten hacer el trabajo de la forma adecuada.
- **JIT** (Just in time): producir en el momento que es requerido.
- **Andon**: Sistemas visuales de alertas sobre anomalías, fallos o problemas en el momento y lugar de la producción.
- **Jidoka**. Instalación en el proceso de sistemas que verifican su calidad.
- **Heijunka**. Técnicas para adaptar la producción a una demanda fluctuante del cliente.

Lean Software Development

Este término se refiere a la aplicación de los principios lean en el desarrollo del software. Mary y Tom Poppendieck (Poppendieck & Poppendieck, 2003) lo acuñaron. Gracias a sus aportes y los de la comunidad ágil, Lean Software Development está desarrollando un inventario de prácticas útiles para el desarrollo ágil de software. .

Se basa en 7 principios:

1. ***Eliminar el desperdicio***
   - Las actividades que no crean valor no sirven y deben ser eliminadas. Algunos ejemplos:
     - Tareas que no fueron solicitadas por el cliente.
     - Sobre-documentación del proyecto.
     - Procesos de desarrollo que se ejecutan sin analizar su nivel de eficiencia o vigencia.
     - Un mayor número de líneas de código no siempre es mejor, y además requiere mayor esfuerzo de testeo y de mantenimiento.
     - Los errores, bugs y fallos del software son verdadero desperdicio que se debe reducir.

2. ***Construir con calidad***

   Incluir en el procedimiento prácticas para mejora de la calidad en el producto (traslación de prácticas “poka-yoke” y “andon” al desarrollo de software).
   Un procedimiento que respeta la calidad es aquel que es conocido, entendido y mejorado por los propios participantes. Para lograrlo es necesario compromiso y respeto.

   Algunos ejemplos de prácticas que se deben contemplar al hacer software.
   - Técnicas como TDD (Test Driven Development) permiten que usuarios (clientes), programadores y *tester* definan claramente los requerimientos y confeccionen pruebas de aceptación antes de escribir el código. Ayuda a la comprensión de los programadores y mejora el entendimiento de los requerimientos.
   - El programador es responsable de su propio desarrollo. No debe esperar a que las pruebas o los procedimientos de aseguramiento de calidad descubran los errores.
   - Fomentar el desarrollo de pruebas automatizadas.
   - Refactorización del código, para lograr simplicidad y eliminar duplicidades.

3. ***Compartir conocimiento***

   Conocer lo que necesita el cliente requiere dedicación y esfuerzo, y debe convertirse en el aspecto principal, porque desarrollar un producto que no es útil, es el mayor desperdicio.
   Hacer software implica un proceso de aprendizaje: entender qué es lo que el cliente quiere y cómo entregar la mejor solución posible. El desarrollo incremental proporciona cuantiosa y frecuente retroinformación.

4. ***Diferir el compromiso***

   En los proyectos ágiles que parten con una visión que evoluciona con el desarrollo, el compromiso con el cliente se asienta y evoluciona en la misma medida que se van concretando y comprometiendo los incrementos del producto.

5. ***Entregar rápido***

   La gestión evolutiva realiza entregas rápidas a los clientes, que se encuentran con código operativo desde etapas tempranas. Dicho código debe ser desarrollado con calidad ya que no se puede mantener una velocidad importante de entrega si no se cuenta con calidad y un equipo disciplinado, comprometido y confiable.

6. ***Respetar a las personas***

   Lean se basa en el respecto por las personas que son el elemento único y diferenciador de cada organización.
   Deben estar suficientemente capacitadas y ser responsables de los procesos en los que intervienen, de modo que cuando resultan necesarios cambios y mejoras, cada persona colabora en su desarrollo.

7. ***Optimizar el todo***

   Lean invita a contemplar el proceso completo, es decir todo el flujo de valor, en lugar de hacerlo en cada etapa. El problema de optimizar cada fase por separado es que genera inventarios grandes en los puntos de transición. En el mundo del software, estos "inventarios" representan al trabajo parcialmente terminado (por ejemplo, requisitos completos, pero sin diseñar, codificar o probar). Lean demostró que un flujo de "una pieza" (por ejemplo, enfocarse en construir un ítem de manera completa) es un proceso más eficiente que concentrarse en construir las partes separadas de forma rápida.

Kanban: Origen y definición

El término japonés Kanban, fue el empleado por Taiichi Onho (Toyota), para referirse al sistema de visualización empleado en los procesos de producción que coordinan en una cadena de montaje la entrega a tiempo de cada parte en el momento que se necesita, evitando sobreproducción y almacenamiento innecesario de producto.

Se puede traducir como tablero o tarjeta de señalización, y su origen se remonta finales de los cuarenta o principio de los cincuenta.

El término kanban aplicado a la gestión ágil de proyectos se refiere a técnicas de representación visual de información para mejorar la eficiencia en la ejecución de las tareas de un proyecto

Kanban en el sector TIC

El uso de tableros kanban muestra y gestiona el flujo de avance y entrega, y ayuda a evitar los dos problemas más importantes: cuellos de botella y tiempos muertos.

El desarrollo ágil de software emplea prácticas de gestión visual por ser las que mejor sirven a los principios de comunicación directa y simplicidad en la documentación y gestión.

Desde 2005 es cada vez más frecuente reemplazar los formatos de lista para las pilas de producto y de sprint por notas adhesivas en tableros, que resultan más versátiles al poder cambiar su posición, bien para reordenar las prioridades de las historias de una pila de producto, o para reflejar a través de su posición en, cuáles se están programando, probando, o se encuentran terminadas.

Las prácticas kanban son válidas para gestión evolutiva con entrega continua. Deben emplearse con criterios de flexibilidad, sin considerar prescripciones ni excepciones en el método de trabajo, para lograr la implementación personalizada, que dé la mejor respuesta a los principios ágiles, de ingeniería concurrente, o de síntesis de ambos, con los que trabaje la organización.

Gestión técnica vs. gestión experta

Algunos autores consideran a scrum y kanban como marcos de reglas y prácticas diferentes.

Según Kniberg & Skarin al considerarlos así, se dibujarían las siguientes diferencias entre ellos(Kniberg & Skarin, 2009):

- Scrum prescribe roles, kanban no.
- Scrum trabaja con iteraciones de tiempo fijo, kanban con cadencias (simples, múltiples o dirigidas por eventos).
- Scrum limita el wip por iteración, kanban limita el wip por estado del flujo de trabajo.
- Los equipos de scrum son multidisciplinares, en kanban pueden ser de especialistas.
- Scrum no permite cambiar tareas del sprint, en kanban la tarea puede alterarse hasta entrar en el flujo.
- En scrum la pila de producto debe tener la longitud de al menos un sprint. En kanban se debe atender al ritmo de arrastre de tareas.
- En scrum se deben estimar las historias y las tareas y calcular la velocidad, kanban no mide tareas ni velocidad.
- Scrum necesita una pila de producto priorizada, en kanban es la siguiente historia o tarea arrastrada desde el cliente.
- Scrum prescribe reuniones diarias, kanban no.
- Scrum emplea diagramas burndown, kanban no.
- Los tableros scrum se *resetean* al final de cada sprint.

Al evolucionar hacia un modelo de scrum pragmático, basado en la aplicación de los valores de la agilidad con la experiencia y conocimientos propios, y abandonar los modelos basados en reglas, se aprende a romper éstas y flexibilizar las prácticas, quedando como triviales cuestiones “técnico-metodológicas” que de otra forma distorsionan la realidad y el foco de la gestión:

Situación A: “Por un lado se desea usar kanban, pero por otro se quiere estimar las tareas (por ejemplo para registrar la velocidad por razones organizativas de mi empresa) …”

Situación B: "La empresa gestiona proyectos simultáneos con una organización de oficina de proyectos y por eso encaja mejor el establecimiento de roles; pero sin embargo se quiere trabajar con kanban en lugar de con scrum… ¿Debería hacer scrumban? ¿qué es eso? ¿o lo que se va a hacer es Scrumbut? ¿es la solución de un mal gestor?."

Tableros kanban: conceptos

Las prácticas de gestión visual kanban son útiles para:

1. Gestionar el flujo de trabajo.
2. “Radiar” información.

Un tablero kanban puede emplearse con una de estas finalidades, o con ambas a la vez; aunque en realidad si se usa para gestionar el flujo de trabajo, siempre acompaña el hecho de mostrar la información del trabajo que se está controlando.

1.- Características de kanban para gestionar el flujo de trabajo.

Sobre un tablero kanban se pueden establecer pautas para regular el flujo de avance de las tareas.

La posición de cada tarjeta sobre el tablero refleja el estado en el que se encuentra el trabajo que representa.

Los estados mínimos habituales en un tablero kanban son “pendiente”, “en curso” y “terminado”.

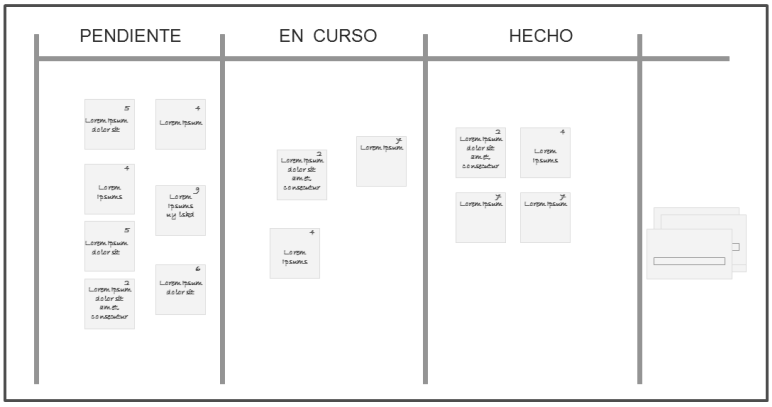


Ilustración 42 – Estructura básica de un tablero kanban

En algunos casos es conveniente incluir estados adicionales (por ejemplo: testeado, validado).

El orden de los trabajos desde el área “pendiente”, refleja la secuencia de tareas prevista, según sus prioridades.

Los trabajos monitorizados pueden ser tareas, historias de usuario o epics, según el uso al que se dedique el tablero

Kanban saca a la superficie la información de los problemas.

Los conflictos en la priorización de los trabajos, los problemas en el flujo por impedimentos o cargas de trabajo, las incidencias en el desarrollo, etc. se ponen de manifiesto de forma inmediata al actualizar sobre el tablero el estado de los trabajos.

Kanban Facilita un ritmo sostenido y evita la ley de Parkinson

Genera un avance continuo de trabajo cuyo ritmo no está “predestinado” por una planificación temporal: Gantt o Sprint (incremento iterativo).

La ausencia de hitos temporales evita la tendencia habitual de alargar el tiempo de trabajo hasta completar el tiempo estimado (ley de Parkinson).

> El trabajo se expande hasta llenar el tiempo que se había previsto
> Ley de Parkinson.

Por otra parte, la ausencia de hitos temporales, sin técnicas de monitorización y gestión del avance generaría alargamiento de tiempos y retrasos por aplazamiento y perfeccionismo.

> Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
>
> Principio del Manifiesto Ágil

2.- Radiador de información.

Kanban como “radiador de información:

Favorece la comunicación directa

- Facilita la comunicación directa del equipo al actualizar la información en reuniones enfrente de un tablero kanban.
- Comparte la visibilidad de la evolución del proyecto con todos los implicados.

Facilita la detección temprana de problemas

- Kanban monitoriza continuamente la evolución del proyecto. La actualización de la información *just-in-time*, ayuda a identificar en un primer momento los posibles impedimentos, problemas y riesgos, que de otra forma pasan desapercibidos hasta que empiezan a producir retrasos o repercusiones ya inevitables.

Favorece una cultura de colaboración y resolución.

- Es un medio de comunicación abierto y transparente para el equipo y todos los participantes.

Kanban: Operativa

Secuencia y polivalencia

Dos son los factores que delimitan cuatro escenarios de trabajo diferentes

- Secuencia (del trabajo).
- Polivalencia (de las personas).

Secuencia.

¿Los trabajos reflejados en las notas adhesivas del tablero deben ejecutarse en un orden determinado o pueden realizarse en cualquier orden?

No es lo mismo diseñar un tablero para el equipo de programadores de un sistema, que para el de mantenimiento de los sistemas informáticos de una empresa. Los primeros deben realizar las tareas en un orden determinado. Así por ejemplo, no es posible realizar la tarea de pruebas si antes no se ha hecho la de programación. Sin embargo las siguientes podrían ser las tareas de un equipo de mantenimiento: “instalación de nueva impresora en el equipo de dirección” “actualización del sistema operativo en el servidor web”, etc. Este tipo de tareas se pueden realizar en cualquier orden. No hay una relación de dependencia entre ellas de forma que no se pueda realizar una si la otra no se ha completado.

Polivalencia

¿Es un equipo polivalente o de especialistas? ¿Por el tipo de trabajo y el perfil de los integrantes del equipo, cualquier miembro puede realizar cualquier tarea?

Siguiendo con los ejemplos anteriores, es posible que en el equipo de mantenimiento una persona pueda indistintamente instalar una impresora, o un sistema operativo; o es posible que no, que haya técnicos de hardware y técnicos de software. De forma similar un proyecto de programación puede incluir tareas específicas de diseño gráfico, programación, integración, testing, etc. que pueden realizar sólo determinados miembros del equipo.

La imagen siguiente refleja el valor clave de kanban: la gestión de un flujo continuo de avance, y los factores que se deben considerar para configurar el tablero con la estructura más adecuada a nuestro trabajo y equipo.

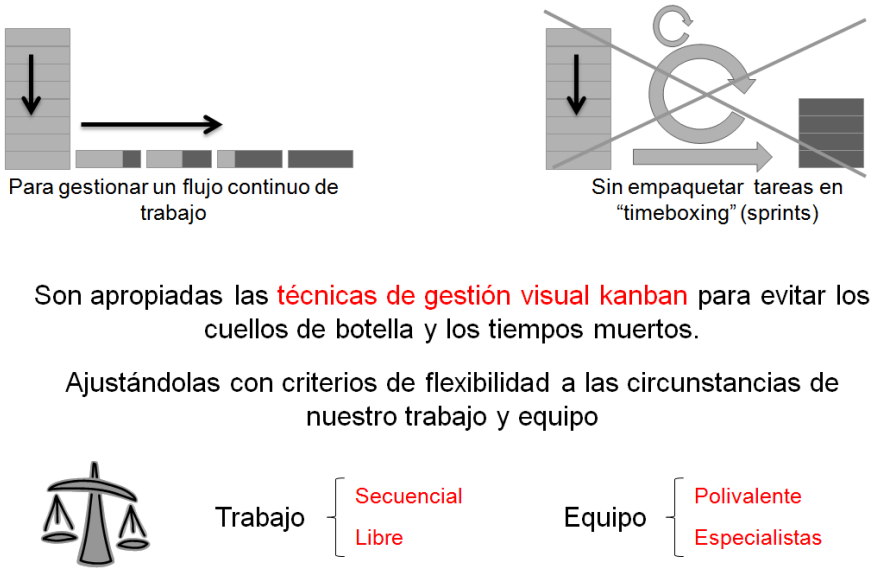


Ilustración 43: Fortaleza y variables clave de los tableros kanban

Cuatro son los patrones posibles según se combinen un tipo de trabajo secuencial o libre, con un equipo polivalente o de especialistas:

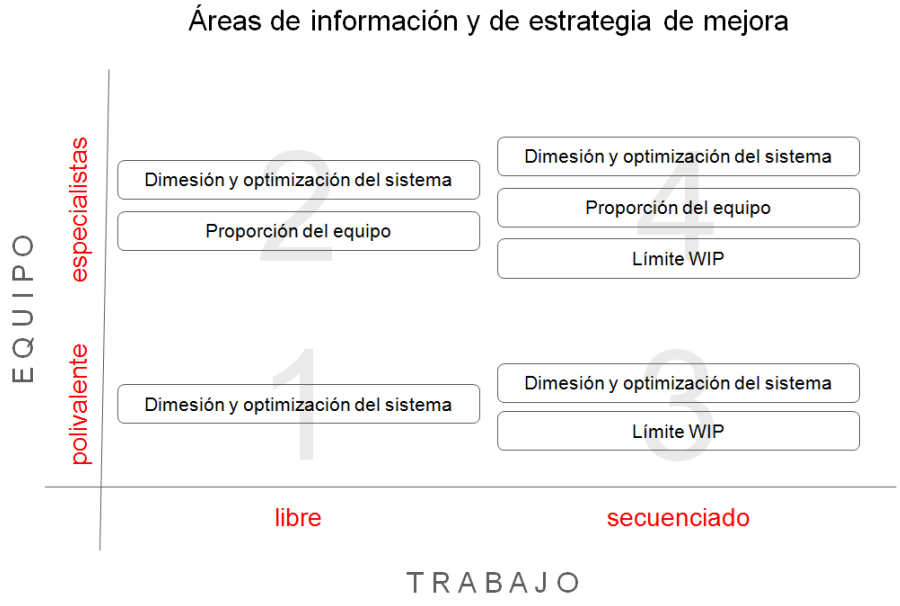


Ilustración 44: Áreas de información y mejora reveladas por los tableros kanban

1.- Equipo polivalente que realiza tareas no secuenciales.

Este es el entorno más fácil de gestionar: cualquier persona del equipo puede hacer cualquier tarea, y las tareas se pueden tomar en cualquier orden.

Para solucionar problemas de saturación de trabajo, o tiempos muertos se debe ajustar la dimensión del equipo y / o la optimización del sistema.

2.- Equipo de especialistas que realiza tareas no secuenciales.

La especialización del equipo aporta un factor de complejidad para solucionar cuellos de botella o los tiempos muertos.

Si se producen cuellos de botella, la estrategia con el tipo de tareas que los provocan debe ir en una o en ambas de las siguientes líneas:

- Dimensionamiento: bien del número de personas capacitadas para realizar ese tipo de tareas, bien en el número de tareas de ese tipo que se pueden comprometer, o en el tiempo de respuesta al cliente.
- Optimización del proceso de ejecución de ese tipo de tareas.

La presencia de tiempos muertos debe cuestionar el dimensionamiento de la demanda, o su distribución no homogénea.

3.- Equipo polivalente que realiza tareas secuenciales

En este caso es la dependencia entre tareas la principal causa de tensiones en el flujo.

La primera línea de mejora en estos casos es el ajuste del WIP de cada fase, antes de considerar modificaciones en el dimensionamiento del equipo y el compromiso.

WIP es un término inglés que en el campo de la manufactura lean, de donde proviene, se emplea para indicar la cantidad de productos en proceso de fabricación, que aún no están terminados.

En los tableros kanban, por analogía se emplea este término para indicar las tareas que se encuentran en una fase del proceso, pendientes de pasar a la siguiente o de completarse; y en este entorno el término WIP indica límite o número máximo de tareas que se pueden acumular en un área determinada. Así por ejemplo, decir que en un tablero kanban para programación de software el área de testing o pruebas “tiene un WIP de 3” quiere decir que no puede haber más de tres tareas simultáneamente en esa fase.

4.- Equipo de especialistas y trabajo que requiere un orden secuenciado.

Este es el entorno más complejo porque requiere el ajuste y revisión en todas las líneas de mejora posible: dimensión y equilibrio de especialistas en el equipo, dimensión o equilibrio de tiempos de respuesta en el compromiso, y ajuste de límites WIP en cada fase.

En cada una de estas cuatro situaciones posibles el tablero saca a la superficie los problemas, y el equipo o gestor puede realizar los ajustes en las líneas de trabajo posibles según cada caso y en función de su criterio y las circunstancias de su organización.

Casos prácticos de tableros kanban

A continuación se muestran ejemplos de posibles tableros para dar soporte a las diferentes posibilidades vistas.

Ejemplo de tablero para ofrecer información del desarrollo del producto.

El ejemplo muestra un tablero que podría encontrarse en la oficina del responsable de producto mostrando el estado en el que se encuentra la construcción del producto.

En este caso no se emplea como herramienta de gestión visual, sino solamente como “radiador” de información.

Tablero para mostrar la siguiente información relativa al estado de desarrollo del producto:

- Historias de usuario sugeridas, que se encuentran en evaluación sin determinar aún si se incorporarán al producto.
- Historias de usuario aprobadas: se incorporarán al producto.
- Historias de usuario “preparadas” (ya valoradas y priorizadas) previstas para ser programadas.
- Historias de usuario que se están programando actualmente.
- Historias de usuario ya programadas que se pueden evaluar en el servidor de pruebas.
- Historias de usuario ya evaluadas, pendientes de desplegar.
- Historias de usuario desplegadas en las dos últimas versiones.

Ilustración 45: Ejemplo de tablero kanban para monitorizar el estado del producto

Ejemplos de tableros para desarrollo evolutivo, con incremento continuo, y con incremento iterativo.

Ilustración 46 – Tableros: incremento continuo – incremento iterativo

Este es un ejemplo de flexibilidad, o adecuación de las prácticas a las características de la organización. Kanban, además de ofrecer información visual, permite aplicar técnicas como limitación del wip, y muestra las áreas susceptibles de mejora, para ajustar la cadencia del flujo. Resulta útil, como ya hemos visto, para trabajar con incremento continuo.

¿Pero puede emplear kanban un equipo que haga scrum con incremento iterativo para emplear la representación visual del avance de las tareas, en lugar de usar un gráfico de avance?

Este apartado muestra ejemplos de tableros para ambos casos.

Caso 1: Incremento continuo.

Las siguientes imágenes representan posibles tableros para guiar la gestión del trabajo de un equipo que está desarrollando un producto con incremento continuo, y en el que se muestra la siguiente información:

- Pila de tareas.
- Tareas “preparadas”.
- Tareas en análisis.
- Tareas en codificación.
- Tareas terminadas.
- Tareas integradas en el servidor de desarrollo (labs).
- Tareas integradas en producción.

Ilustración 47: Ejemplo de tablero kanban para monitorizar y gestionar incremento continuo.

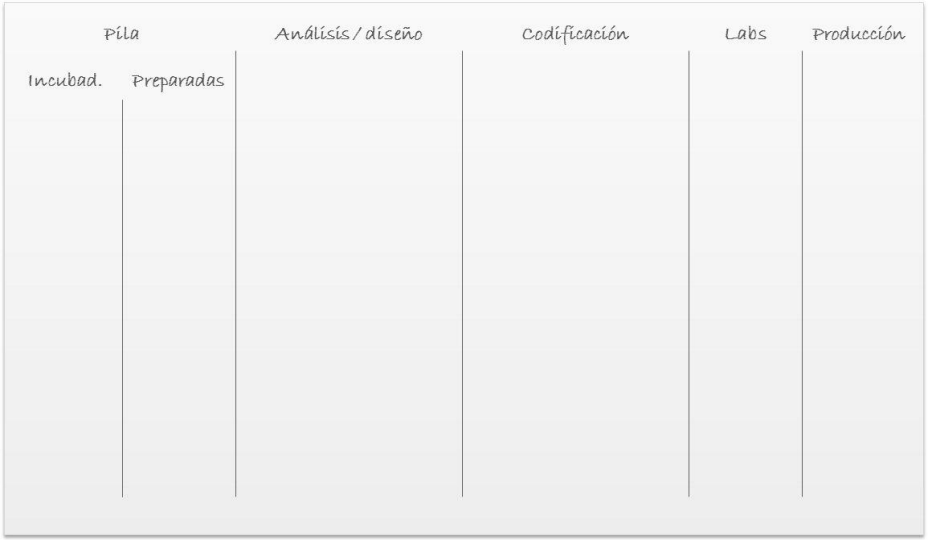


Ilustración 48 – Ejemplo de tablero kanban para monitorizar y gestionar incremento continuo.

Ilustración 49 – Ejemplo de tablero kanban para monitorizar y gestionar incremento continuo

Caso 2: Incremento iterativo.

Tablero para guiar la gestión de trabajo de un equipo que trabaja en incrementos iterativo (sprints) y que muestra:

- Pila de tareas.
- Tareas preparadas.
- Tareas en análisis.
- Tareas en codificación.
- Tareas terminadas.
- Tareas integradas en el servidor de desarrollo (labs)
- Tareas integradas en producción.

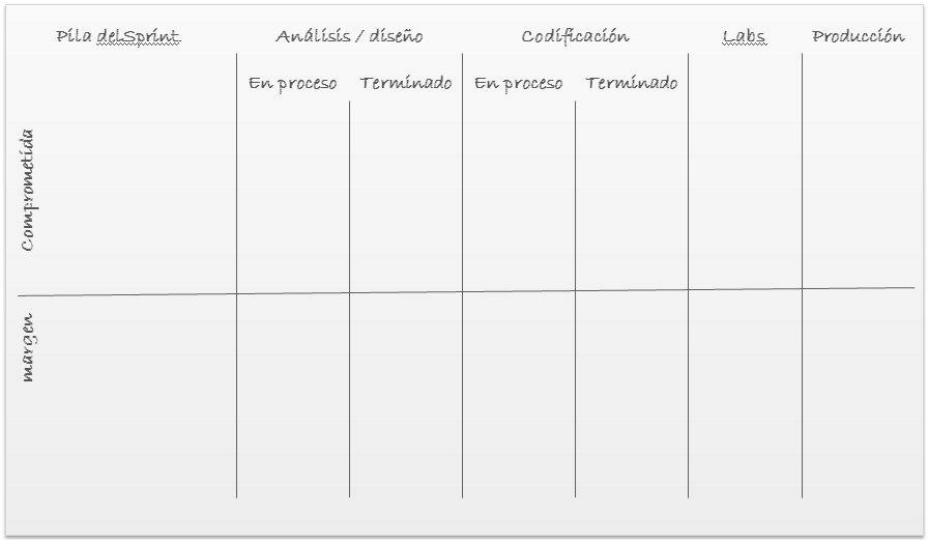


Ilustración 50: Ejemplo de tablero kanban para monitorizar y gestionar incremento iterativo

Ejemplo de tableros para un equipo de operación y mantenimiento.

Tablero guía para la gestión de un equipo de operación y mantenimiento que refleja:

- Estado de las tareas programadas para la semana y persona que está trabajando con cada una.
- Estado de incidencias no previstas y urgentes, y personas que están trabajando en cada una.

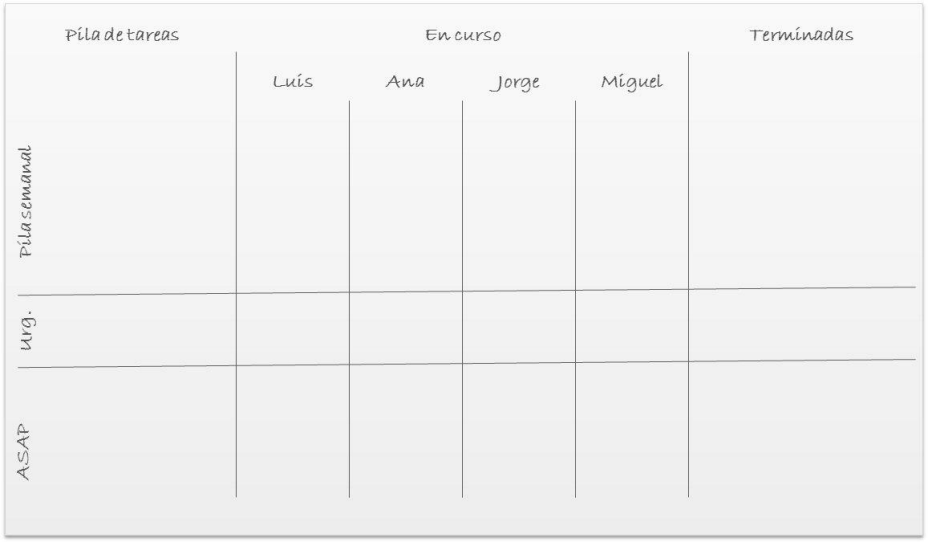


Ilustración 51: : Ejemplo de tablero kanban para monitorizar y gestionar tareas de mantenimiento

Kanban Box

Una práctica diseñada para gestionar tareas de varios proyectos en un mismo departamento de producción de software es una implementación Kanban, denominada Kanban Box.

La configuración es la siguiente:

La organización mantiene una “pila de producción” o lista de historias de usuario preparadas: pendientes, estimadas y priorizadas.

Si la organización trabaja en un único producto, la “pila de producción” es en definitiva la pila del producto.

Si lleva a cabo el desarrollo o mantenimiento simultáneo de varios sistemas, la pila de producción es gestionada por los propietarios de producto, o la oficina de proyectos; en definitiva quienes sean responsables según la estructura de la organización.

Ilustración 52: Ejemplo de pila de producto con tarjetas kanban

En la pila de producción las tareas están preparadas, y ordenadas según los criterios de prioridad compartidos entre los intereses de los diferentes proyectos y de la organización en conjunto.

El equipo que va a hacerse cargo de una historia, la descompone en tareas que se representan en una “caja kanban”:

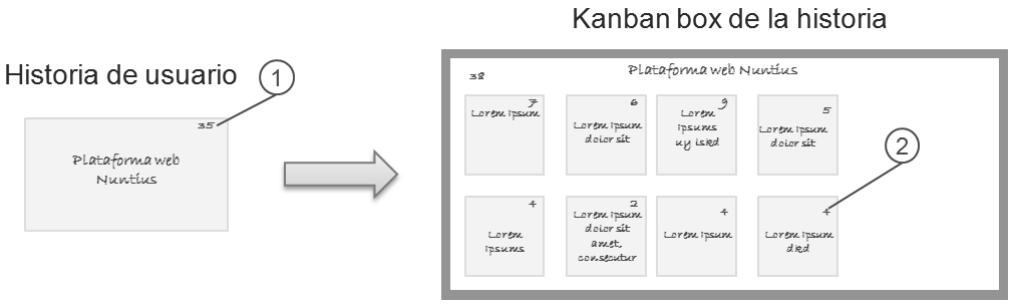


Ilustración 53: Ejemplo de implementación: kanban box

(1) Estimación de la historia.
(2) Estimación de la tarea.

En cada caja se representa:

(1) Puntos totales de esfuerzo estimados.

(2) Tarjetas con las tareas.
(3) Un fondo de escala graduado con los puntos totales de esfuerzo estimados.
(4) Barra dibujada con rotulador “borrable” que representa la velocidad prevista.
(5) Barra de velocidad real.
(6) Una descripción de la historia de usuario.

Ilustración 54: Ejemplo de implementación: kanban box

De esta forma se van “encajando” las historias de usuario, o preparando para pasar a producción.

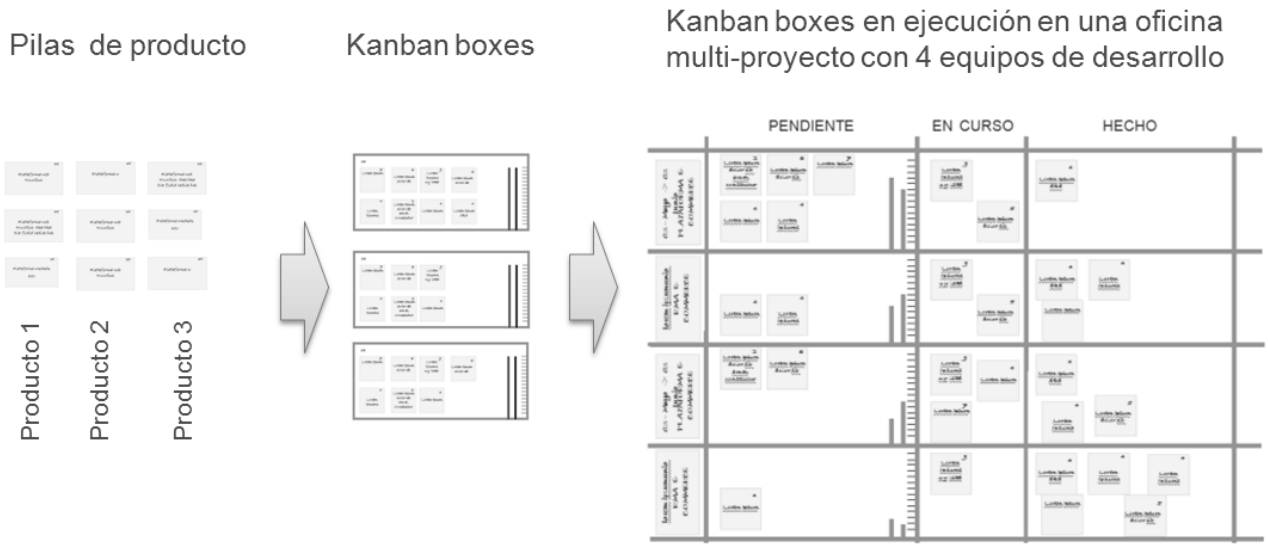


Ilustración 55 : Ejemplo de implementación: kanban box

Las cajas preparadas van entrando en los “slots” disponibles en la columna “pendiente” del tablero general de la organización.

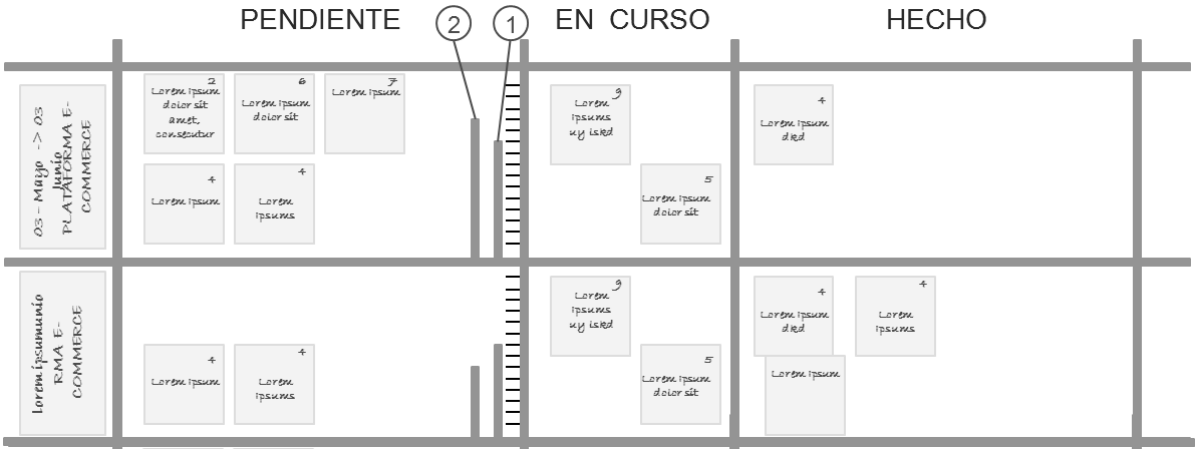


Ilustración 56 : Ejemplo de implementación: kanban box

A diario, cada equipo realiza el scrum diario, actualizando el estado de cada tarea (pendiente -> en curso -> hecho), y las barras de velocidad:

La barra de velocidad prevista (1) se actualiza todos los días considerando la velocidad media de la organización y el nº de miembros del equipo. Si por ejemplo se trata de un equipo de 3 personas y la velocidad media es de 3 puntos por persona/día, cada día la barra de velocidad prevista disminuye de 9 puntos.

La barra de velocidad real (2) representa la suma del esfuerzo de las tareas que aún se encuentran en estado “pendiente” y “en curso”.

La diferencia de altura entre las barras de velocidad muestra desviaciones del esfuerzo previsto, en uno u otro sentido.

Muda, Mura y Muri y consejos para ajustar el flujo.

Muda, Mura y Muri son tres conceptos clave de la mejora continua Kaizen, que la producción Lean incorpora como variables protagonistas para incrementar la eficiencia.

- Muda: Desperdicio
- Mura: Discrepancia. Variabilidad del flujo de trabajo. Interrupciones en el flujo de trabajo. Tiempo muerto.
- Muri: Tensión. Sobrecarga de trabajo que produce cuellos de botella.

Mudas

Las mudas o desperdicios más habituales en los proyectos TIC son:

- Burocracia: Procedimientos, documentación y papeleo innecesario que no aportan valor al resultado.
- Sobreproducción: Desarrollar más características de las necesarias.
- Multiproyecto: Alternar el tiempo de trabajo entre varios proyectos e interrupciones del flujo de trabajo.
- Esperas: Tiempos de espera por falta de cadencia en el flujo de trabajo.
- “Ir haciendo”: Encargar trabajo para ir avanzando algo no definido y así no tener paradas a las personas.
- Desajustes de capacidad: Personas de gran talento asignadas a tareas rutinarias, y viceversa.
- Errores: Retrabajo por bugs.

Los tableros kanban detectan y ayudan a gestionar las otras dos variables kaizen: Mura y Muri.

Factores determinantes de Mura y Muri

Hay que tener en cuenta que cada organización o tipo de proyecto tiene sus características propias de:

- Secuencia: las tareas deben realizarse secuencialmente o no.
- Polivalencia o especialización de los miembros del equipo.

Y estos son los factores que en cada caso determinan la mayor o menor dificultad para mantener un flujo continuo de desarrollo.

Como ya se ha visto, los equipos de miembros polivalentes que trabajan con tareas no secuenciales son los que más fácilmente pueden conseguir un flujo de avance constante.

Cuando surgen dificultades, las variables que se deben combinar, según las posibilidades en cada caso, son:

- Volumen de la demanda.
- Orden del backlog o pila de historias de usuario: si se va a producir un cuello de botella en una fase, se debe procurar que la próxima historia que vaya a entrar al tablero requiera poco esfuerzo de esa fase.
- WIP o límite de tareas en una determinada fase.
- *Staffing*: Tamaño del equipo y especialización o polivalencia.

Muri

El WIP es una variable importante para ajustar los cuellos de botella (Muri):

Al emplear kanban como técnica con la que regular un incremento continuo, desaparece el concepto de sprint. El **incremento** no es el resultado de un sprint, sino cada historia de usuario que se termina y entrega. Para mantener un flujo continuo de funcionalidades que, una a una van incrementando el producto de forma sostenida, es necesario evitar la aparición de cuellos de botella (Muri): la acumulación de tareas en una determinada fase del proceso. Una técnica útil es limitar la cantidad de trabajo que puede acumularse en cada fase y generar cuellos de botella.

Al parámetro que indica el número máximo de tareas en un área del tablero kanban se le denomina WIP: Work In Process, o bien “in-process inventory” (inventario en el proceso). No se debe confundir con Work in progress (trabajo en progreso) término que designa un trabajo que ha comenzado pero aún no está terminado.

Un valor “WIP” demasiado bajo puede producir cuellos de botella en otras fases, en especial si el sistema es demasiado rígido (tareas secuenciales y equipo de especialistas).

La experiencia ayuda al equipo a ir mejorando el ajuste para mantener el flujo lo más continuo posible.

Si no se cuenta con experiencia previa, y considerando que las tareas no deberían tener tamaños mayores de 4 horas ideales, el equipo debe establecer un criterio de inicio, y a partir de él ir ajustando.

En este sentido una recomendación generalmente útil (en equipos de miembros polivalentes) es empezar con un WIP igual al nº de miembros del equipo x 1.5, redondeando el resultado por exceso.

No es aconsejable trabajar con tareas de tamaño que se prevea superior a un día de trabajo, y si esto ocurre lo aconsejable es dividirlas en otras de menor tamaño.

Ejemplo:

La figura siguiente presenta un tablero kanban con límite de trabajo en los estados “Producto analizado” y “En curso”.

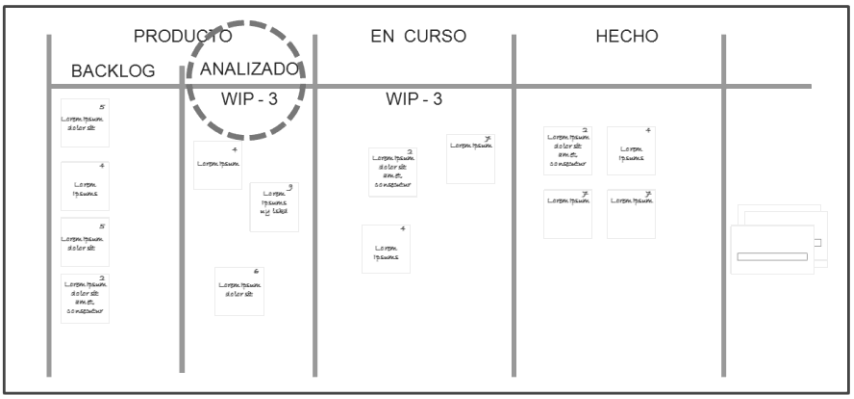


Ilustración 57 – WIP

En este ejemplo, el propietario de producto tiene una zona para ordenar el backlog (A). Es el área en la que el responsable de producto añade, modifica, y reordena la prioridad de cada historia de forma continua.

Pero sólo son tres las historias que pueden estar en estado “analizado” para pasar a producción. Tres con la que ya está previsto analizar y revisar la estimación con el equipo.

De igual forma, el área “en curso” tiene un límite de tres historias. Hasta que una no pasa a “HECHO”, no puede entrar ninguna a producción, y de igual forma mientras haya tres en la zona “ANALIZADO” no se decide cuál será la próxima historia del backlog.

Así se fuerza un flujo de trabajo sin cuellos de botella, continuo y enfocado.

Mura

Los principales factores responsables de la variabilidad del flujo y la aparición de Mura o tiempos muertos son:

- Grado de multifuncionalidad de los miembros del equipo.
- Flexibilidad en el orden en el que se deben hacer las diferentes fases de cada tarea.
- Flexibilidad para alterar el orden de entrada de las historias de usuario desde la pila de producto.

Cuanto mayores sean estos aspectos, más fácil resulta evitar la aparición de tiempos muertos.

Bibliografía

Bauer, F., Bolliet, L., & Helms, H. (1969). Software Engineering. Report on a conference sponsored by the NATO SCIENCE COMITEE. *Software Engineering* (pág. 136). Garmisch: Peter Naur & Brian Randell.

Beck, K. (2000). *Extreme Programming Explained.* Addison-Wesley.

Hino, S. (2006). *Inside the Mind of Toyota: Management Principles for Enduring Growth.* Productivity Press.

Kniberg, H., & Skarin, M. (2009). *Kanban and Scrum, making the most of both.* crisp.

Nonaka, I., & Takeuchi, H. (1995). *The Knowledge-Creating Company.* University Press.

Nonaka, I., & Takeuchi, H. (1986). The New New Product Development Game. *Harvard Business Review* .

Nonaka, I., & Takeuchi, I. (2004). *Hitotsubashi on Knowledge Management.* Singapore: John Wiley & Sons.

Ohno, T. (1988). *The Toyota Production System: Beyond Large-scale Production.* Productivity Press.

Orr., K. (2003). CMM versus Agile Development: Religious wars and software development. *Cutter Consortium, Executive Reports 3(7)* .

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit for Software Development Managers.* Addison Wesley.

Schwaber, K. (1995). *SCRUM Development Process.* Burlington: OOPSLA 95.

Smith, A. (1776). *An Inquiry into the Nature and Causes of the Wealth of Nations.* Londres: W. Strahan & T. Cadell.

Taylor, F. W. (1911). *The Principles of Scientific Management.* New York: Harper & Brothers.

Turner, R., & Jain, A. (2002). Agile Meets CMMI: Culture Clash or Common Cause? *XP/Agile Universe 2002* , 153-165.

Tabla de ilustraciones

Índice