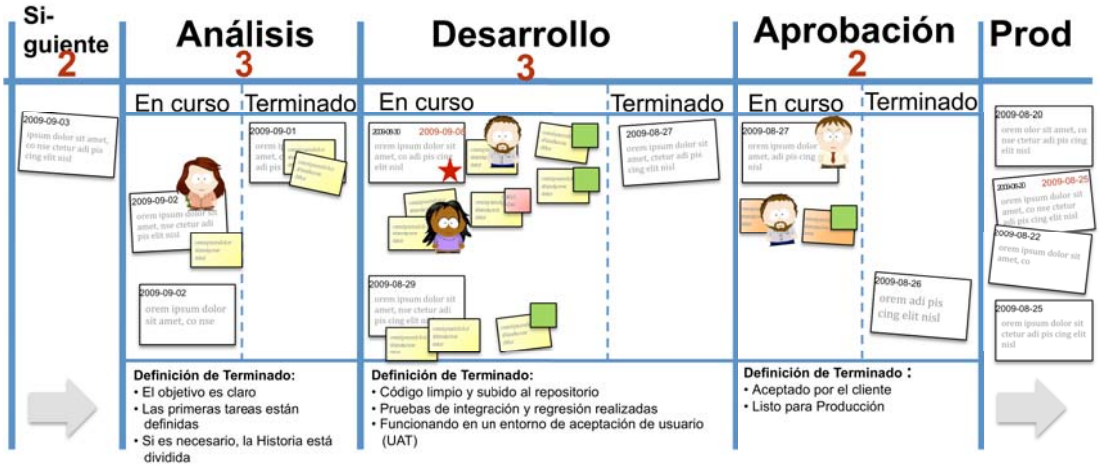
Kanban y Scrum – obteniendo lo mejor de ambos



Henrik Kniberg & Mattias Skarin
Prólogo de Mary Poppendieck & David Anderson

C4Media, editores de InfoQ.com.

Este libro es parte de la serie de libros Desarrollo Empresarial de Software de InfoQ.

Para información sobre cómo solicitar este u otros libros de InfoQ, por favor contacte con books@c4media.com.

Editor Jefe: Diana Plesa
Diseño de Portada: Bistrian Iosip
Composición: Accurance

Datos de catalogación de la publicación por la Librería del Congreso:
ISBN: 978-0-557-13832-6
Impreso en los Estados Unidos de América

Traducción al castellano: Equipo de contenidos de Agile Spain (www.agilespain.com)
Rodrigo Corral – Plain Concepts - http://www.plainconcepts.com/
Xavier Quesada-Allue – Agilar - http://www.agilar.org/
Jorge Uriarte – Gailen Tecnologías - http://www.gailen.es
Agustín Yagüe – UPM - https://syst.eui.upm.es/
Teo Sánchez - http://teosanchez.blogspot.com
Juan Palacio - http://www.navegapolis.net/
Gregorio Mena - http://eclijava.blogspot.com/
Ángel Agueda - http://legnita.wordpress.com
Laura Morillo-Velarde
Jorge Jiménez
Javier Sánchez
Juan Quijano

Coordinación de la traducción: Ángel Medinilla -– http://www.proyectalis.com

Contenido

Prólogo por Mary Poppendieck

Henrik Kniberg es una de esas pocas personas que puede extraer la esencia de una situación complicada, seleccionar las ideas principales de entre las distracciones incidentales, y proporcionar una explicación clara y cristalina que es increíblemente sencilla de entender. En este libro, Henrik realiza un trabajo brillante explicando las diferencias entre Scrum y Kanban. Nos aclara que son simplemente herramientas, y que lo que realmente necesitamos es tener el juego completo de herramientas, comprender sus fortalezas y limitaciones, y como se usa cada una de ellas.

En este libro aprenderás en qué consiste Kanban, sus fortalezas y limitaciones, y cuando usarlo. También obtendrás buenas ideas sobre cómo y cuando mejorar Scrum, o cualquier otra herramienta que puedas estar usando. Henrik demuestra que lo importante no es la herramienta con la que empiezas, sino la forma en la que mejores constantemente el uso de esa herramienta y expandas tu conjunto de herramientas con el tiempo.

La segunda parte de este libro, realizada por Mattias Skarin, hace que el libro sea aun más efectivo mostrando paso a paso la aplicación de Scrum y Kanban en una situación de la vida real. Aquí verás un ejemplo de cómo las herramientas fueron empleadas tanto por separado como de forma combinada para mejorar un proceso de desarrollo de software. Notarás que no hay una sola "mejor forma" de hacer las cosas; debes pensar por ti mismo y averiguar, basándote en tu situación, cuál es tu siguiente paso para conseguir un mejor desarrollo de software.

Mary Poppendieck

Prólogo por David Anderson

Kanban se basa en una idea muy simple: el trabajo en curso (Work In Progress, WIP) debería limitarse, y sólo deberíamos empezar con algo nuevo cuando un bloque de trabajo anterior haya sido entregado o ha pasado a otra función posterior de la cadena. El Kanban (o tarjeta señalizadora) implica que se genera una señal visual para indicar que hay nuevos bloques de trabajo que pueden ser comenzados porque el trabajo en curso actual no alcanza el máximo acordado. Esto no suena muy revolucionario ni parece que vaya a afectar profundamente el rendimiento, cultura, capacidad y madurez del equipo y de la organización que les rodea. ¡Lo increíble es que sí lo hace! Kanban parece un cambio muy pequeño pero aun así cambia todos los aspectos de una empresa.

Hemos llegado a la conclusión de que Kanban es una aproximación a la gestión del cambio. No es un proceso de desarrollo de software o de gestión de proyectos. Kanban es una aproximación a la introducción de cambios en un ciclo de vida de desarrollo de software o metodología de gestión de proyectos. ya existente El principio de Kanban es que comienzas con lo que sea que estés haciendo ahora mismo. Comprendes tu actual proceso mediante la realización de un mapa del flujo de valor y entonces acuerdas los límites de trabajo en curso (WIP) para cada fase del proceso. A continuación comienzas a hacer fluir el trabajo a través del sistema comenzándolo ("pull") cuando se van generando las señales Kanban.

Kanban ha demostrado ser útil en equipos que realizan desarrollo Ágil de software, pero también están ganando fuerza en equipos que utilizan métodos más tradicionales. Kanban se está introduciendo como parte de iniciativas Lean para transformar la cultura de las organizaciones y fomentar la mejora continua.

Dado que el WIP está limitado en un sistema Kanban, cualquier elemento que se bloquea por cualquier razón tiende a atascar el sistema. Si un número suficiente de elementos se bloquean todo el proceso se

para en seco. Esto tiene el efecto de que todo el equipo y la organización se concentran en resolver el problema, desbloqueando estos elementos para permitir la restauración del flujo productivo.

Kanban usa un mecanismo de control visual para hacer seguimiento del trabajo conforme este viaja a través del flujo de valor. Típicamente, se usa un panel o pizarra con notas adhesivas o un panel electrónico de tarjetas. Las mejores prácticas apuntan probablemente al uso de ambos. La transparencia que esto genera contribuye también al cambio cultural. Las metodologías Ágiles han obtenido buenos resultados proporcionando transparencia respecto al trabajo en curso y completado, así como en el reporte de métricas como la velocidad (cantidad de trabajo realizada en una iteración). Kanban sin embargo va un paso más allá y proporciona transparencia al proceso y su flujo. Kanban expone los cuellos de botella, colas, variabilidad y desperdicios. Todas las cosas que impactan al rendimiento de la organización en términos de la cantidad de trabajo entregado y el ciclo de tiempo requerido para entregarlo. Kanban proporciona a los miembros del equipo y a las partes interesadas visibilidad sobre los efectos de sus acciones (o falta de acción). De esta forma, los casos de estudios preliminares están demostrando que Kanban cambia el comportamiento y motiva a una mayor colaboración en el trabajo. La visibilidad de los cuellos de botella, desperdicios y variabilidades y su impacto también promueve la discusión sobre la posibles mejoras, y los equipos comienzan rápidamente a implementar mejoras en su proceso.

Como resultado, Kanban propicia la evolución incremental de los procesos existentes, una evolución que generalmente está alineada con los valores del Agilismo y de Lean. Kanban no pide una revolución radical de la forma en la que las personas trabajan, sino que sugiere un cambio gradual. Es un cambio que surge del entendimiento y el consenso entre todos los trabajadores y sus colaboradores.

Kanban, por su naturaleza de Sistema Pull ("arrastre", "tirar"; la producción se realiza cuando el cliente retira un elemento terminado), también propicia diferir el compromiso tanto en la priorización del trabajo nuevo como en la entrega del trabajo existente. Típicamente, los equipos llegarán a un acuerdo sobre la cadencia de reuniones de priorización con el bloque funcional que les precede en el proceso para decidir en qué deben trabajar a continuación. Estas reuniones se pueden mantener de forma frecuente porque habitualmente son bastante cortas. Se responde a una pregunta simple, algo como "Desde nuestra última reunión hemos liberado dos huecos de trabajo, y nuestro tiempo de ciclo actual es de 6 semanas para entregar. Qué dos cosas son las que más os

gustaría ver entregadas dentro de 6 semanas?". Esto tiene un efecto doble. Formular una pregunta simple generalmente proporciona respuestas rápidas y de buena calidad, y mantiene las reuniones cortas. La naturaleza de la pregunta muestra que el compromiso acerca de en qué trabajar se difiere hasta el último momento responsable. Esto mejora la Agilidad mediante una gestión de las expectativas, un acortamiento de los tiempos de ciclo desde el compromiso hasta la entrega y eliminando los re-trabajos, ya que las probabilidades de un cambio en las prioridades se minimizan.

Un último comentario sobre Kanban es que el efecto de limitar el WIP proporciona predecibilidad sobre el tiempo de ciclo y hace que los entregables sean más fiables. La estrategia de "parar el proceso" ante impedimentos y bugs también parece promover altos niveles de calidad y un rápido descenso del re-trabajo.

Aunque todo esto se volverá evidente con las increíblemente claras explicaciones de este libro, no ocurre lo mismo con la explicación de cómo hemos llegado hasta aquí. Kanban no se concibió en una sola tarde mediante una increíble epifanía. Surgió a lo largo de muchos años. Muchos de sus profundos aspectos psicológicos y sociológicos que cambian la cultura, capacidad y madurez de las organizaciones nunca fueron imaginados en un principio. Fueron más bien descubiertos. Muchos de los resultados de Kanban son contra-intuitivos. Lo que parece ser una aproximación muy mecánica - limitar el WIP y retirar el trabajo - tiene en realidad profundos efectos en las personas y en como interactúan y colaboran unas con otras. Ni yo, ni ninguno de los que estuvieron involucrado en los inicios de Kanban, previmos esto.

Implementé lo que luego se convirtió en Kanban como una estrategia para la gestión del cambio que encontrase una resistencia mínima. Tenía esto claro ya en el 2003. También lo implementé por los beneficios mecánicos. Como ya estaba descubriendo en aquellos días mediante la aplicación de técnicas Lean, si gestionar el WIP tenía sentido, entonces limitarlo tenía más sentido aun, ya que eliminaba el coste de gestión asociado. Así que en 2004 decidí intentar implementar un sistema pull desde sus principios básicos. Tuve la oportunidad cuando un Gerente de Microsoft contactó conmigo y me pidió que le ayudase a gestionar en cambio en su equipo de mantenimiento evolutivo de aplicaciones TI internas. La primera implementación estaba basada en el Sistema Pull de Teoría de las Limitaciones conocido como Drum-Buffer-Rope (tambor-inventario-cuerda). Fue un gran éxito: el tiempo de ciclo bajo un 92%, el caudal de salida (troughput) se incrementó en más del triple y la

predecibilidad (cumplimiento de fechas) fue de un muy aceptable 98%. En 2005, Donald Reinertsen me convenció de implementar un sistema totalmente Kanban. Tuve la oportunidad en 2006, cuando asumí la dirección del Departamento de Ingeniería de Corbis en Seattle. En 2007 comencé a reportar los resultados. La primera presentación fue en el Lean New Product Development Summit de Mayo de 2007 en Chicago. Continué con un Open Space en Agile 2007 en Washington DC en Agosto de ese año. 25 personas asistieron. 3 de ellas eras de Yahoo! : Aaron Sanders, Jarl Scotland y Joe Arnold. Volvieron a sus hogares de California, India y el Reino Unido e implementaron Kanban con sus equipos, que ya estaban luchando con Scrum. También crearon un grupo de discusión de Yahoo! que, en el momento de escribir estas líneas, cuenta con casi 800 miembros. Kanban estaba comenzando a diseminarse y los "early adopters" empezaban a hablar de sus experiencias.

Ahora en 2009, la adopción de Kanban está creciendo realmente y más y más informes de campo están apareciendo. Hemos aprendido mucho de Kanban en los últimos 5 años y seguimos aprendiendo todos los días. He concentrado mi propio trabajo en hacer Kanban, escribir sobre Kanban, hablar sobre Kanban y pensar sobre Kanban para así poder entenderlo mejor y explicárselo a otros. Me he abstenido deliberadamente de compararlo con otros métodos Ágiles existentes, aunque en 2008 invertimos algo de esfuerzo en explicar por qué Kanban debía ser considerada como una aproximación Ágil compatible.

He dejado a otros con mayor experiencia contestar preguntas como "¿En qué se diferencian Kanban y Scrum?". Me alegra que Henrik Kniberg y Mattias Skarin hayan surgido como líderes en este aspecto. Usted, el trabajador del conocimiento en el día a día, necesita información para poder tomar decisiones más informadas y poder progresar con su trabajo. Henrik y Mattias están solucionando sus necesidades de una forma que yo nunca pude. Estoy particularmente impresionado con la sesuda comparación de Henrik y su presentación equilibrada, imparcial y basada en los hechos. Sus dibujos e ilustraciones son particularmente perspicaces y muchas veces te ahorran tener que leer muchas páginas de texto. El informe de campo de Mattias es importante, ya que demuestra que Kanban es mucho más que una teoría y nos muestra mediante el ejemplo cómo podría ser útil para usted en su organización.

Espero que disfruten este texto de comparación entre Kanban y Scrum y que les aporte una mayor visión de Agile en general y de Kanban y Scrum en particular. Si desea aprender más sobre Kanban, por favor,

visite el sitio Web de nuestra comunidad, la Limited WIP Society, http://www.limitedwipsociety.org/

David J. Anderson
Sequim, Washington, USA. 8 de Julio de 2009

Introducción

Normalmente no escribimos libros. Preferimos pasar el tiempo metidos a fondo en las trincheras ayudando a los clientes a optimizar, corregir y refactorizar su proceso de desarrollo y su organización. No obstante, hemos notado una clara tendencia últimamente, y nos gustaría compartir algunas reflexiones sobre ella. Este sería un caso típico:

- **Jim:** "¡Por fin hemos conseguido implantar Scrum del todo!"
- **Fred:** " ¿Y qué tal os va"
- **Jim:** "Bueno, mucho mejor que lo que teníamos antes..."
- **Fred:** "...¿pero?"
- **Jim:** "... pero claro, está el equipo de operación y mantenimiento."
- **Fred:** "sí, ¿Y?"
- **Jim:** "Bueno, nos encanta todo lo de organizar por prioridades en una Pila de Producto, los equipos auto-organizados, el Scrum diario, las retrospectivas, etc...."
- **Fred:** "¿Y cuál es el problema?"
- **Jim:** "Seguimos fracasando en nuestros Sprints"
- **Fred:** "¿Por qué?"
- **Jim:** "Porque nos resulta muy difícil comprometernos a una planificación de 2 semanas. Las iteraciones no tienen mucho sentido para nosotros, simplemente nos ponemos con lo más urgente que tenemos cada día. ¿Quizás deberíamos hacer iteraciones de una semana?

- **Fred:** "¿Os podríais comprometer al trabajo de una semana? ¿Se os permitiría concentraros en eso y trabajar en paz durante una semana?"

- **Jim:** "En realidad no, tenemos asuntos que van surgiendo en el día a día. Quizás si hiciéramos sprints de un día…"

- **Fred:** "¿Vuestras tareas tardan menos de un día en solucionarse?"

- **Jim:** "No, a veces tardan varios días"

- **Fred:** "Así que los sprints de 1 día tampoco funcionarían . ¿Habéis considerado eliminar por completo los sprints?"

- **Jim:** "Bueno, la verdad es que eso nos gustaría. Pero ¿no va eso en contra de Scrum?"

- **Fred:** "Scrum es solo una herramienta. Tú eliges cuándo y cómo usarla. ¡No seas su esclavo!"

- **Jim:** "¿Qué deberíamos hacer entonces?"

- **Fred:** "¿Has oído hablar de Kanban?"

- **Jim:** "¿Qué es eso? ¿En qué se diferencia de Scrum?"

- **Fred:** "¡Toma, lee este libro!"

- **Jim:** "Pero a mi me gusta el resto de Scrum, ¿tengo que cambiar ahora?"

- **Fred:** "¡No! Puedes combinar ambas técnicas"

- **Jim:** "¿Qué? ¿Cómo?"

- **Fred:** "Sólo sigue leyendo.."

Propósito de este libro

Si estás interesado en el desarrollo de software Ágil probablemente habrás oído hablar de Scrum, y puede que también hayas oído hablar de Kanban. Una pregunta que cada vez nos hacen con más frecuencia es "¿Y qué es Kanban, y en qué se diferencia de Scrum? ¿Dónde se complementan? ¿Hay conflictos potenciales?

El propósito de este libro es aclarar la niebla, de forma que puedas entender como Kanban y Scrum pueden ser útiles en tu entorno.

¡Haznos saber si lo hemos conseguido!

Parte I – Comparación

Esta primer parte del libro es un intento de realizar una comparación práctica y objetiva de Scrum y Kanban. Es una versión algo actualizada del artículo "Kanban vs. Scrum de Abril de 2009. Este artículo se hizo muy popular, así que decidí convertirlo en un libro y le pedí a mi colega Mattias que lo aliñase con un caso de estudio "desde las trincheras" de uno de nuestros clientes. ¡Cosa fina! Siéntete libre de saltar directamente a la Parte II si prefieres comenzar con el caso de estudio, no me ofenderé. Bueno, quizás solo un poco.

/Henrik Kniberg

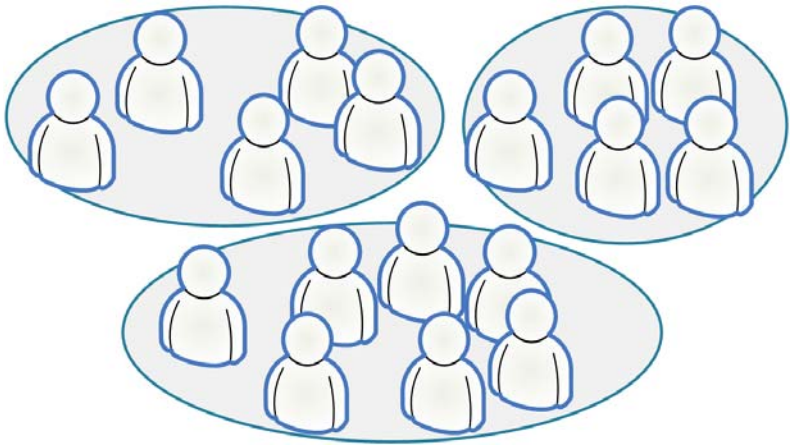
1

Bueno pero, al fin y al cabo, ¿qué son Scrum y Kanban?

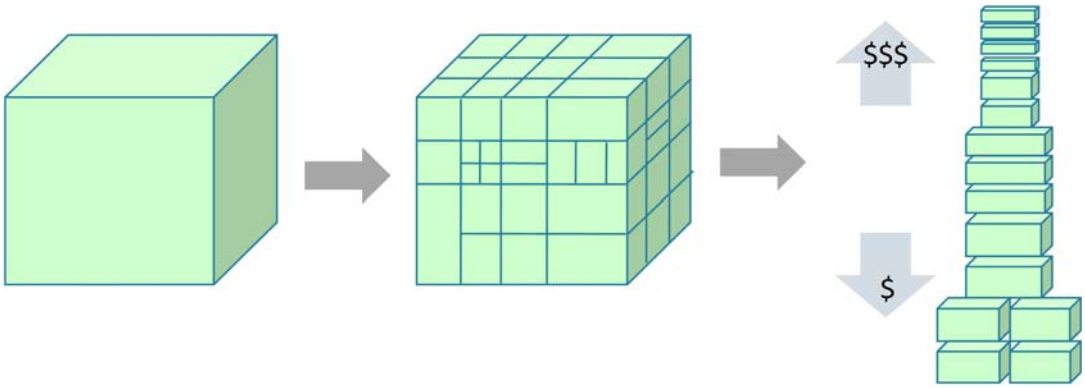
De acuerdo, tratemos de resumir Scrum y Kanban en menos de 100 palabras cada uno.

Scrum en pocas palabras

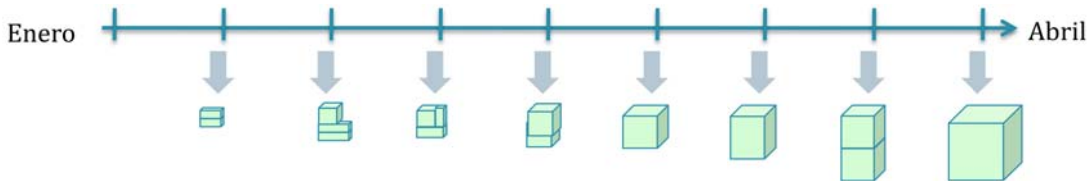
· **Divide tu organización** en equipos pequeños, interdisciplinarios y auto-organizados.



- **Divide el trabajo** en una lista de entregables pequeños y concretos. Ordena la lista por orden de prioridad y estima el esfuerzo relativo de cada elemento.



- **Divide el tiempo** en iteraciones cortas de longitud fija (generalmente de 1 a 4 semanas), con código potencialmente entregable y demostrado después de cada iteración.



- **Optimiza el plan de entregas** y actualiza las prioridades en colaboración con el cliente, basada en los conocimientos adquiridos mediante la inspección del entregable después de cada iteración.

- **Optimiza el proceso** teniendo una retrospectiva después de cada iteración.

Así en lugar de un **grupo numeroso** pasando **mucho tiempo** construyendo algo **grande,** tenemos un **equipo menor** pasando un **tiempo más corto** construyendo algo **menor.** Pero **integrando** con **regularidad** para ver el conjunto.

138 palabras ... bastante cerca.

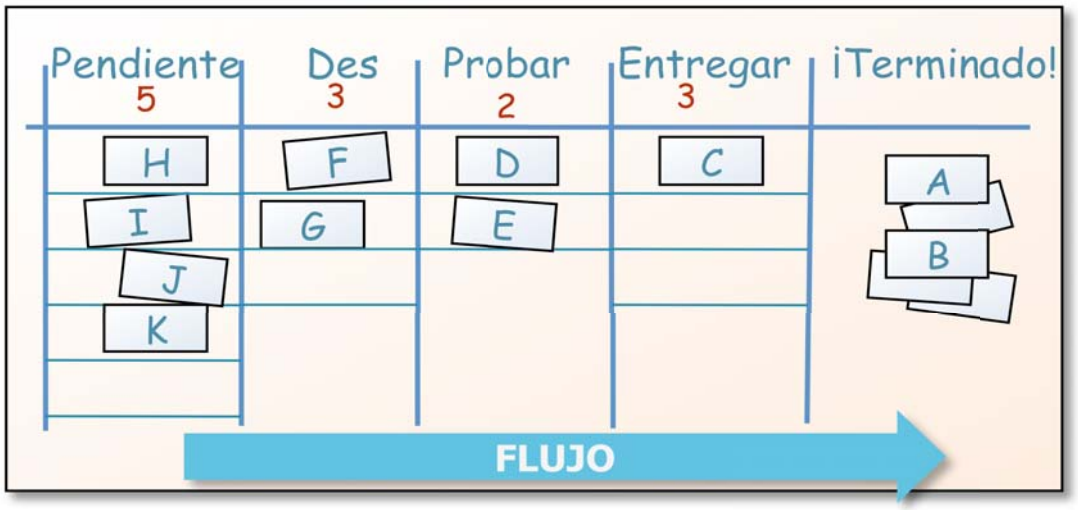
Para más detalles, consulte "Scrum y XP desde las trincheras". El libro es de lectura gratuita online. Conozco al autor, es un buen tipo: o)

http://www.crisp.se/ScrumAndXpFromTheTrenches.html

Para obtener más enlaces sobre Scrum echa un vistazo a http://www.crisp.se/scrum

Kanban en pocas palabras

- **Visualiza el flujo de trabajo**
  - Divide el trabajo en bloques, escribe cada elemento en una tarjeta y ponlo en el muro.
  - Utiliza columnas con nombre para ilustrar dónde está cada elemento en el flujo de trabajo.
- **Limita el WIP** (Work in Progress, trabajo en curso) - asigna límites concretos a cuántos elementos pueden estar en progreso en cada estado del flujo de trabajo.
- **Mide el lead time** (tiempo medio para completar un elemento, a veces llamado "tiempo de ciclo"), optimiza el proceso para que el lead time sea tan pequeño y predecible como sea posible.



Se han agrupado enlaces útiles sobre Kanban en: http://www.crisp.se/kanban

2

Entonces, ¿Cómo se relacionan Kanban y Scrum entre sí?

Scrum y Kanban son ambos herramientas de proceso

Herramienta = cualquier cosa usada como medio para realizar una tarea o propósito

Proceso = cómo trabajas.

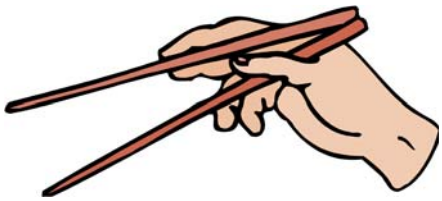
Scrum y Kanban son *herramientas de proceso* que te ayudan a trabajar más eficazmente, en cierta medida, diciéndote qué hacer. Java también es una herramienta, te ofrece una forma más sencilla de programar una computadora. Un cepillo de dientes también es una herramienta, te ayuda a llegar a tus dientes para que puedas limpiarlos.

Compara herramientas para entender, no para juzgar

Cuchillo o tenedor - ¿Qué herramienta es mejor?

¿Es una pregunta bien expresada? Porque la respuesta depende del contexto. Para comer albóndigas el tenedor es probablemente mejor. Para cortar las

setas el cuchillo es probablemente mejor. Para golpear la mesa cualquiera de los dos servirá. Para comer un filete probablemente querrás utilizar los dos de manera conjunta. Para comer arroz... Bueno... algunos prefieren el tenedor mientras que otros prefieren los palillos.



Así, cuando comparamos herramientas debemos ser cuidadosos. Comparar para comprender, no para juzgar..

Ninguna herramienta es completa, ninguna herramienta es perfecta

Como cualquier herramienta, Scrum y Kanban no son ni perfectas ni completas. No te dicen *todo* lo que tienes que hacer, solo proporcionan ciertas restricciones y directrices. Por ejemplo, Scrum te obliga a tener iteraciones de duración fija y equipos interdisciplinarios, y Kanban te obliga a usar tableros visibles y a limitar el tamaño de tus colas.

Curiosamente, el valor de una herramienta es que *limita tus opciones*. Una herramienta de proceso que te permite hacer cualquier cosa no es muy útil. Podríamos llamar a ese proceso "Hacer lo que sea" o qué tal "Hacer lo correcto". El proceso "Hacer lo correcto" garantiza que funciona, ¡es una bala de plata! Porque si no funciona, es obvio que no estabas siguiendo el proceso :o)

Usar las herramientas adecuadas te ayudará a triunfar, pero no garantizará el éxito. Es fácil confundir el éxito/fracaso del proyecto, con el éxito / fracaso de la herramienta.

- Un proyecto puede triunfar debido a una gran herramienta.
- Un proyecto puede triunfar a pesar de una pésima herramienta.
- Un proyecto puede fallar debido a una pésima herramienta.

- Un proyecto puede fallar a pesar de una gran herramienta.

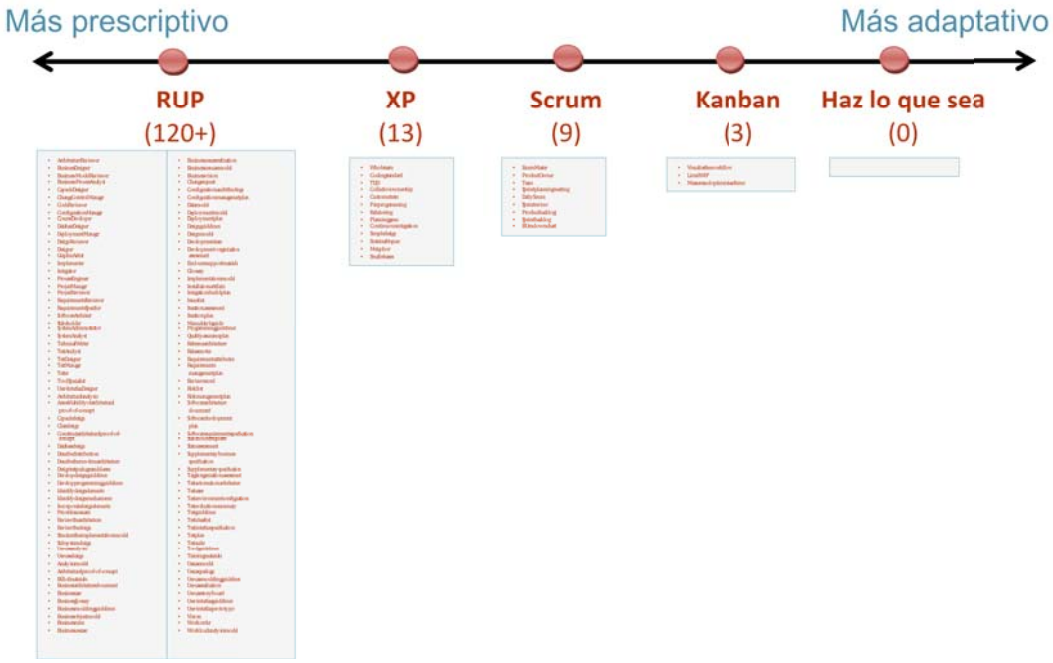
Scrum es más prescriptivo que Kanban

Podemos comparar herramientas viendo cuántas reglas proporcionan. *Prescriptivo* significa "más reglas a seguir" y *adaptativo* significa "menos reglas a seguir". 100% prescriptivo significa que no te deja usar el cerebro, hay una regla para todo. 100% adaptativos significan Haz Lo Que Sea, no hay ninguna regla o restricción. Como puedes ver, los dos extremos de la escala son de alguna manera ridículos.

Los métodos ágiles se denominan a veces los métodos *ligeros,* en concreto, porque son menos restrictivos que los métodos tradicionales. De hecho, el primer principio del Manifiesto Ágil es "Individuos e interacciones sobre procesos y herramientas".

Scrum y Kanban son muy adaptables, pero en términos relativos Scrum es más restrictivo que Kanban. Scrum te da más limitaciones, y así deja menos opciones abiertas. Por ejemplo Scrum prescribe el uso de iteraciones de duración fija, Kanban no.

Vamos a comparar algunas herramientas de proceso más en la escala restrictivo vs adaptativo:



RUP es muy restrictivo- tiene más de 30 perfiles, más de 20 actividades, y más de 70 artefactos, una cantidad enorme de cosas a aprender. Realmente no se espera que uses todos los que hay, si bien, se supone que seleccionarás un subconjunto adecuado para tu proyecto. Lamentablemente, esto parece ser difícil en la práctica. "Hmmmm ... ¿Necesitaremos artefactos *para la Configuración de auditoría de los resultados?*¿Necesitaremos un *perfil* de *gerente de Control de cambios* ?No estoy seguro, así que mejor los mantenemos por si acaso ". Esto puede ser una de las razones por lasque las implementaciones de RUP suelen ser bastante pesadas en comparación con los métodos ágiles como Scrum y XP.

XP (eXtreme Programming) es más restrictivo en comparación con Scrum. Se incluye la mayoría de Scrum + un montón de buenas prácticas específicas de ingeniería, tales como desarrollo dirigido por pruebas y la programación en parejas.

Scrum es menos restrictivo que XP, ya que no establece ninguna práctica específica de ingeniería. Sin embrago, Scrum es más restrictivo que Kanban ya que prescribe cosas como iteraciones y equipos interdisciplinarios.

Una de las principales diferencias entre Scrum y RUP es que en RUP tienes demasiado, y se supone que quitarás aquello que no necesites. En Scrum tienes demasiado poco, y se supone que añadirás el material que falta.

Kanban deja casi todo abierto. Las únicas normas son Visualiza tu Flujo de trabajo y Limita tu WIP (Work In Progress, Trabajo en curso). A pocos centímetros de Haz lo que Sea, pero sigue siendo sorprendentemente poderoso.

¡No te limites a una única herramienta!

¡Mezcla y combina las herramientas que necesites! Me cuesta imaginar un exitoso equipo Scrum que no incluye la mayoría de los elementos de XP, por ejemplo. Muchos equipos Kanban hacen reuniones diarias (una práctica de Scrum). Algunos equipos Scrum escriben algunos de sus elementos de la pila como Casos de Uso (una práctica de RUP) o limitan el tamaño de las colas (una práctica de Kanban). Usa lo que sea que funcione para ti.

Musashi lo dijo muy bien (famoso Samurai del siglo 17º, famoso por su técnica de lucha de la doble espada)

No te ciñas a una única arma o a una única escuela de lucha.

- Miyamoto Musashi

Sin embargo, presta atención a las limitaciones de cada herramienta. Por ejemplo, si utilizas Scrum y decides dejar de utilizar iteraciones de duración fija (o cualquier otro aspecto central de Scrum), luego no digas que estás utilizando Scrum. Scrum es bastante minimalista en sí mismo, si quitas cosas y todavía lo llamas Scrum entonces la palabra carecerá de sentido y confundirá. Llámalo algo así como "inspirado en Scrum" o "un subconjunto de Scrum", o algo así como "Scrumish" :o)

3

Scrum prescribe roles

Scrum prescribe 3 roles: dueño de producto (establece la visión del producto y las prioridades), equipo (implementa el producto) y Scrum Master (elimina los impedimentos y proporciona liderazgo en el proceso).

Kanban no establece ningún rol en absoluto.

¡Eso no significa que no puedas o no debas tener un papel de dueño de producto en Kanban! Sólo significa que no *tienes que*. En ambos, Scrum y Kanban, eres libre de añadir los roles adicionales que necesites.

Ten cuidado sin embargo al añadir roles, asegúrate de que los roles adicionales realmente añaden valor y no entran en conflicto con otros elementos del proceso. ¿Estás seguro de que necesitas el rol de jefe de proyectos? En un proyecto grande puede ser una gran idea, tal vez es el tipo que ayuda a múltiples equipos y dueños de producto a sincronizarse. En un proyecto pequeño puede ser un desperdicio, o peor, puede dar lugar a la sub-optimización y la microgestión.

La mentalidad general, tanto en Scrum como en Kanban es "menos es más". Así que en caso de duda, comienza con menos.

En el resto del artículo voy a utilizar el término "dueño de producto" para representar a quien establece las prioridades de un equipo, independientemente del método utilizado.

4

Scrum prescribe iteraciones de tiempo fijo

Scrum se basa en iteraciones de tiempo fijo. Puedes elegir la duración de la iteración, pero la idea general es mantener la misma longitud de la iteración durante un período de tiempo, determinando así una *cadencia.*

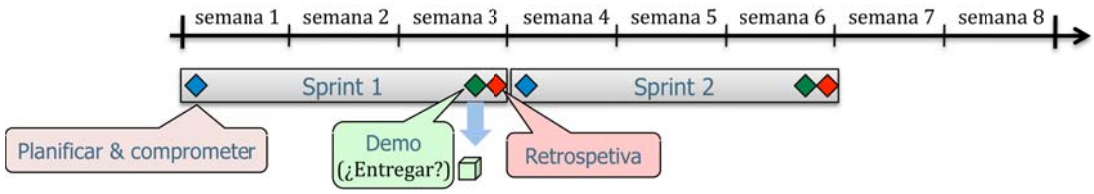
- **Inicio de la iteración:** Se crea un plan de iteración, es decir, el equipo saca un número específico de elementos de la pila de producto, en base a las prioridades del dueño de producto y a cuánto piensa el equipo que puede terminar en una iteración.
- **Durante la iteración:** El equipo se centra en completar los elementos a los que se comprometió. Se fija el alcance de la iteración.
- **Final de la iteración:** El equipo muestra el código funcionando a las partes interesadas, idealmente este código debe ser *potencialmente entregable* (es decir, probado y listo para llevar). Entonces, el equipo hace una retrospectiva para discutir y mejorar su proceso.

Así que una iteración de Scrum es una única cadencia de tiempo fijo que combina tres actividades distintas: la planificación, la mejora de procesos, y (idealmente) la entrega.

En Kanban no se prescriben iteraciones de tiempo fijo. Puedes elegir el momento de hacer la planificación, la mejora de procesos, y la entrega. Puedes elegir hacer estas actividades de manera regular ("la entrega todos los lunes") o bajo demanda ("la entrega cada vez que tengamos algo útil para entregar").

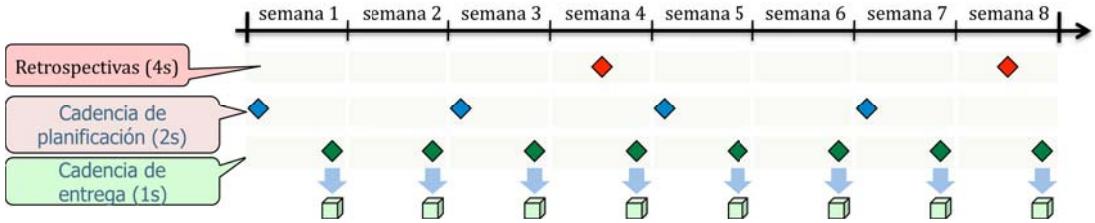
Equipo #1 (cadencia simple)

"Nosotros hacemos iteraciones Scrum"



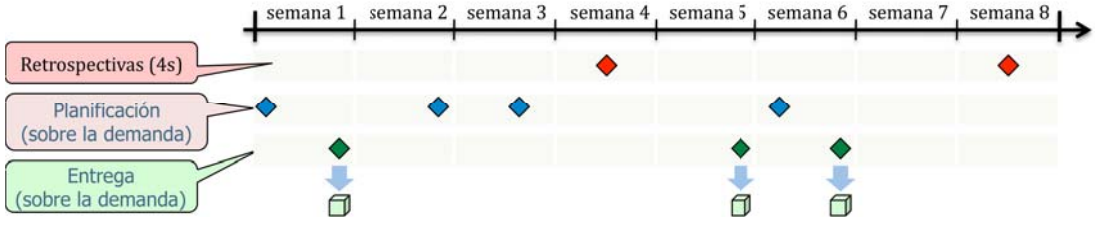
Equipo # 2 (tres cadencias)

"Tenemos tres cadencias diferentes. Cada semana entregamos lo que está listo para su entrega. Cada segunda semana tenemos una reunión de planificación, actualización de nuestras prioridades y los planes de entrega. Cada cuarta semana tenemos una reunión retrospectiva para modificar y mejorar nuestro proceso".



Equipo # 3 (normalmente dirigido por eventos)

"Lanzamos una reunión de planificación cada vez que comenzamos a quedarnos sin cosas que hacer. Lanzamos una entrega siempre que hay un MMF (Minimum Marketable Feature Set, conjunto mínimo de características comercializables) listo para entregar. Lanzamos un círculo de calidad espontáneo siempre que nos topamos con un mismo problema por segunda vez. Además hacemos una retrospectiva más en profundidad cada cuatro semanas. "

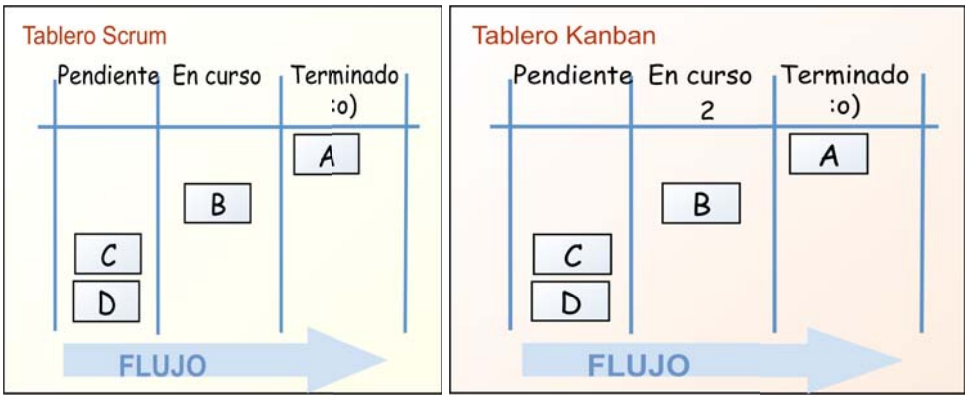


5

Kanban limita el WIP por estado en flujo de trabajo, Scrum limita el WIP por iteración

En Scrum, la pila de sprint muestra qué tareas han de ser ejecutadas durante la iteración actual (= "sprint" en terminología Scrum). Se representa comúnmente usando tarjetas en la pared, llamada una pizarra de Scrum o pizarra de tareas.

Entonces, ¿cuál es la diferencia entre una pizarra de Scrum y una pizarra de Kanban? Vamos a empezar con un proyecto simple y comparar las dos:



En ambos casos estamos siguiendo un grupo de elementos a medida que avanzan a través de un flujo de trabajo. Hemos seleccionado tres estados: pendiente, en curso, y terminado. Puedes elegir los estados que quieras - algunos equipos añaden estados tales como integrar, probar, liberar, etc. No te olvides del principio de *"menos es más*".

Entonces, ¿cuál es la diferencia entre estas dos pizarras de ejemplo? Sí, el pequeño 2 en la columna central en la pizarra Kanban. Eso es todo. Ese 2 significa que "no puede haber más de 2 elementos en esta columna en un momento dado".

¡En Scrum no hay ninguna regla que impida que el equipo ponga todos los elementos en la columna en curso al mismo tiempo! Sin embargo, existe un límite implícito ya que la iteración en sí tiene un alcance fijo. En este caso, el límite implícito por columna es de 4, ya que sólo hay 4 elementos en la pizarra. Así que Scrum limita el WIP indirectamente, mientras que Kanban limita el WIP directamente.

La mayoría de los equipos de Scrum aprenden finalmente que es una mala idea tener demasiados elementos en curso, y desarrollan una cultura de intentar tener los elementos actuales terminados antes de comenzar con nuevos elementos. Algunos incluso deciden limitar explícitamente el número de elementos permitidos en la columna de en curso y, a continuación - ¡TADAAA! - ¡la pizarra de Scrum se ha convertido en una pizarra de Kanban!

Así que tanto Scrum como Kanban limitan el WIP, pero de diferentes maneras. Los equipos de Scrum suelen medir la *velocidad* - cuántos elementos (o unidades correspondientes, tales como "puntos de historia") se hacen por iteración. Una vez que el equipo sabe su velocidad, ésta se convierte en su límite de WIP (o al menos una directriz). Un equipo que tiene una velocidad media de 10 no suele tomar más de 10 elementos (o puntos de historia) para un sprint.

De forma que en Scrum el *WIP se limita por unidad de tiempo.*

En Kanban el *WIP se limita por el estado del flujo de trabajo.*

En el ejemplo anterior de Kanban, como mucho puede haber 2 elementos en el estado de flujo de trabajo "en curso" en un momento dado, independientemente de las longitudes de cadencia. Tienes que elegir qué límite aplicar a qué estados del flujo de trabajo. Pero la idea general es limitar el WIP de *todos los* estados del flujo de trabajo, empezando por "lo más pronto posible" y terminando "lo más tarde posible" a lo largo de la cadena de valor. Así, en el ejemplo anterior deberíamos considerar añadir un límite al WIP también en el estado "pendiente" (o la que quiera que sea tu cola de entrada). Una vez que tenemos los límites de WIP en su lugar, podemos empezar a medir y predecir el tiempo de entrega, es decir, el tiempo medio de un elemento para realizar todo el camino a través de la pizarra. Tener tiempos de entrega predecibles nos permite comprometer los SLA (acuerdos de nivel de servicio, en inglés service-level agreements) y hacer planes de entrega realistas.

Si los tamaños de los elementos varían drásticamente entonces podrías considerar límites de WIP definidos en términos de puntos de historia en su lugar, o cualquiera que sea la unidad de medida que utilices. Algunos equipos invierten sus esfuerzos en la descomposición de elementos de aproximadamente el mismo tamaño para evitar este tipo de consideraciones y reducir el tiempo empleado en la estimación de las cosas (podrías incluso considerar que la estimación es un desperdicio de tiempo). Es más fácil para crear un buen sistema de flujo si los artículos son más o menos de igual tamaño.

6

Ambos son empíricos

Imagina que hubiese mandos en estos contadores y tú pudieses configurar tu proceso simplemente girando los mandos. "Quiero alta capacidad, bajo *lead time,* alta calidad y alta predecibilidad. Entonces giraré los mandos a 10, 1, 10 y 10 respectivamente".

¿No sería maravilloso? Por desgracia no existen estos controles directos. Al menos no que yo sepa. Hacédmelo saber si vosotros los encontráis.
En su lugar contamos con ese puñado de controles *indirectos*.

Scrum y Kanban son ambos empíricos en el sentido de que se espera que experimentes con el proceso y lo adaptes a tu entorno. De hecho, *tienes* que experimentar. Ni Scrum ni Kanban proporcionan todas las respuestas – simplemente nos proporcionan una serie de reglas y limitaciones a la hora de guiar la mejora de nuestros procesos.

- Scrum dice que debemos tener equipos multidisciplinares. Entonces, ¿Quién debe estar en cada equipo? No lo sé, experimenta.
- Scrum dice que el equipo selecciona cuanto trabajo incluir en un sprint. Entonces, ¿Cuánto deben incluir? No lo sé, experimenta.
- Kanban dice que debemos limitar en trabajo en curso. Entonces, ¿Cuál debe ser ese límite? No lo sé, experimenta.

Como he mencionado con anterioridad, Kanban impone menos limitaciones que Scrum. Esto significa menos parámetros sobre los que preocuparse, pero más mandos que girar. Esto puede ser tanto una ventaja como una desventaja dependiendo del contexto. Cuando abres el cuadro de diálogo de configuración de una herramienta de software, ¿prefieres tener 3 o 100 opciones que ajustar? Probablemente una cantidad entre ambas. Depende de cuánto necesites ajustar el software a tus necesidades y cómo de bien conozcas la herramienta.

Digamos que reducimos el límite de trabajo en curso, basándonos en la hipótesis de que esto va a mejorar nuestro proceso. Observamos cómo otros factores como la capacidad, el *lead time*, la calidad, y la predecibilidad evolucionan. Sacamos conclusiones de los resultados y ajustamos algunas cosas más, mejorando continuamente nuestro proceso.
Hay diferentes términos para referirse a este proceso. *Kaizen* (mejora continua en terminología Lean), Inspección y Adaptación (en terminología Scrum), control empírico de procesos o, por qué no, El Método Científico.

El elemento más crítico de este proceso es el ciclo de retroalimentación, el *feedback*. Cambia algo => Observa cómo funciona => Aprende de ello => Cambia algo de nuevo. En general, lo que buscamos es obtener *feedback* en ciclos lo más cortos posibles, de manera que podamos adaptar nuestro proceso rápidamente.

En Scrum el ciclo básico de obtención de *feedback* es el sprint. Existen otros, sin embargo. Especialmente si combinas Scrum con XP (eXtreme programming):

Cuando se realizan correctamente, Scrum + XP nos proporcionan un buen puñado de ciclos de *feedback* extremadamente valiosos.

El ciclo de *feedback* más interno, la programación en parejas, proporciona retroalimentación en unos pocos segundos. Los errores son detectados y corregidos en segundos desde que ocurren (¡He!, ¿no se supone que esa variable debe valer 3?). Este es el ciclo de *feedback* de "¿Estamos construyendo bien las cosas?".

El ciclo de *feedback* más externo, el sprint, nos proporciona un ciclo de retroalimentación de unas cuantas semanas. Este el ciclo de *feedback* de "¿Estamos construyendo las cosas adecuadas?

¿Y en Kanban, que? Bueno, en primer lugar, puedes (y seguramente debes) poner todos los anteriores ciclos de *feedback* en tus proyectos uses o no Kanban. Lo que Kanban te proporciona es una serie de métricas en tiempo real muy útiles.

- *Lead time* medio: Actualizado cada vez que un elemento alcanza el nivel de hecho (o como quiera que llames a tu columna más a la derecha).
- Cuellos de Botella: Un síntoma típico es que la columna X está abarrotada de elementos mientras que la columna X+1 está vacía. Busca "burbujas de aire" en tu panel.

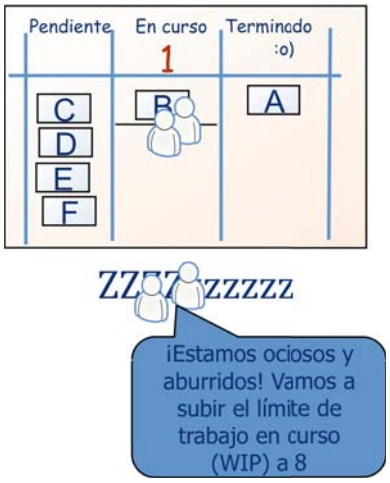
Lo mejor de las métricas en tiempo real es que tú puedes elegir la longitud de tu ciclo de *feedback*, basándote en con qué frecuencia quieres analizar estas métricas e introducir cambios. Un ciclo demasiado largo de retroalimentación significa que la mejora de tu proceso será lenta. Uno demasiado corto significa que tu proceso no podría no tener tiempo para estabilizarse entre cambios, lo que puede producir desperdicio de esfuerzos.

De hecho, la longitud del ciclo de retroalimentación en sí misma es uno de los aspectos con los que puedes experimentar… en una especie de meta-ciclo del *feedback*. OK, es suficiente por ahora.

Ejemplo: Experimentando con los límites para el trabajo en curso en Kanban

Uno de los típicos "puntos de ajuste" de Kanban es el límite de trabajo en curso. ¿Cómo sabemos si lo estamos estableciendo correctamente?

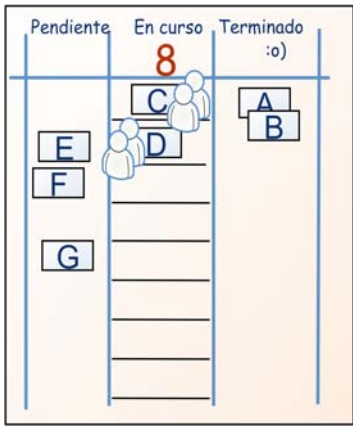
Digamos que tenemos un equipo de 4 personas y que decidimos comenzar con un límite para el trabajo en curso de 1.



En cuanto comencemos a trabajar en un elemento, no podremos comenzar ninguno nuevo hasta que el primer elemento esté Hecho. En consecuencia concluiremos el primer elemento rápidamente.

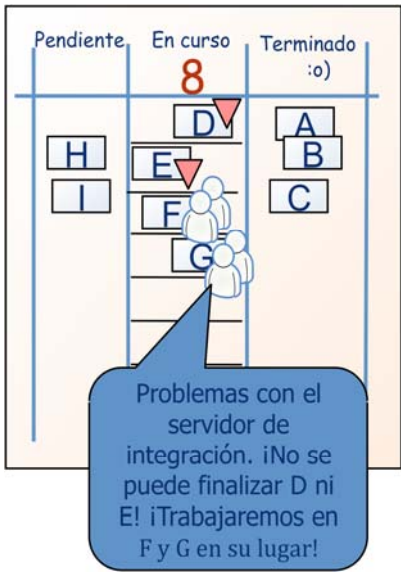
¡Excelente!, pero resulta que habitualmente no es factible que 4 personas trabajen en el mismo elemento (al menos en el contexto de este ejemplo), entonces tenemos gente mirando. Si esto solo ocurre de tanto en cuanto no será un problema, pero si esta situación se da con regularidad, la consecuencia es que el *lead time* medio se incrementará. Básicamente, un trabajo en curso de 1 significa que los elementos pasaran por el estado "En curso" realmente rápido una vez llegan a esta situación, pero permanecerán en estado "Pendiente" más tiempo del necesario, de manera que el *lead time* a lo largo del ciclo de trabajo completo será innecesariamente alto.

Entonces si un trabajo en curso de 1 es demasiado bajo, ¿por qué no incrementarlo a 8?

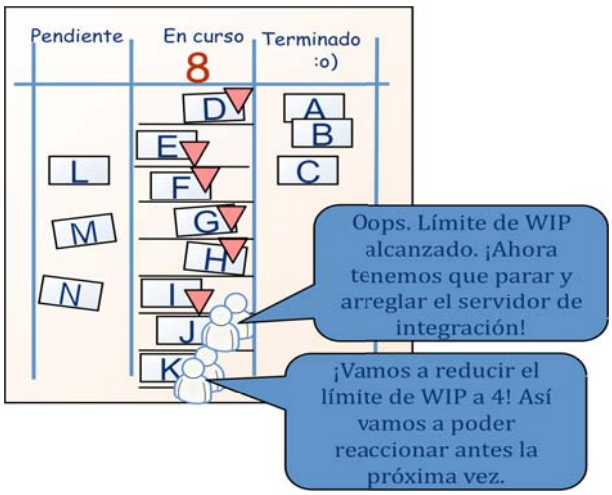


Esto funciona mejor durante algún tiempo. Hemos descubierto que, de media, trabajando en parejas realizamos el trabajo más rápido. De esta manera, en un equipo de cuatro personas, habitualmente, tendremos 2 elementos en curso en un instante dado. ¡El trabajo en curso de 8 es solo un límite superior, luego tener menos elementos en progreso es correcto!

Imaginemos ahora, sin embargo que encontramos un problema con el servidor de integración, de tal manera que no podemos completar totalmente ningún elemento (nuestra definición de Hecho incluye integración). ¿Estas cosas pasan, no?

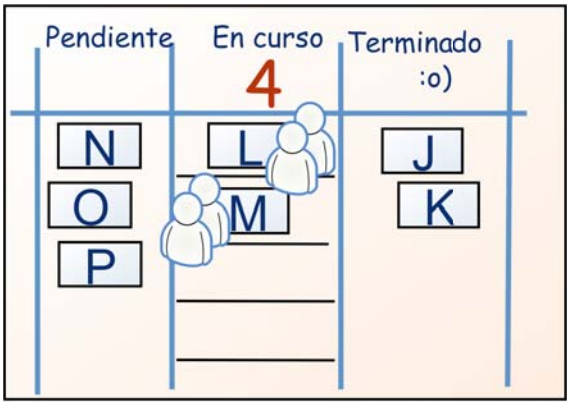


Como no podemos completar los elementos D o E, comenzamos a trabajar en el elemento F. No podemos integrar ninguno de estos tampoco, luego cogeremos un nuevo elemento G. Tras un tiempo alcanzaremos nuestro límite Kanban – 8 elementos “En curso”.



En este punto, ya no podemos coger ningún elemento más. ¡He, mejor arreglemos ese maldito servidor de integración! El límite para el trabajo en curso nos impulsa a reaccionar y corregir el cuello de botella en lugar de seguir acumulando un montón de trabajo sin terminar.

Esto está bien. Pero si el límite de trabajo en curso hubiese sido 4, habríamos reaccionado mucho antes, de este modo tendríamos un mejor *lead time* medio. Medimos nuestro *lead time* medio y continuamente optimizamos nuestro límite de trabajo en curso para optimizar el *lead time*.



Tras un periodo de tiempo podríamos encontrar que los elementos se acumulan en la columna “Pendiente”. Quizás es el momento de añadir un límite de trabajo en curso aquí también.

En cualquier caso, ¿por qué necesitamos una columna “Pendiente”?. Bueno, si el cliente estuviese siempre disponible para decir al equipo que hacer cada vez que preguntase, la columna “Pendiente”, no sería necesaria. Pero en el caso de que el cliente esté a veces no disponible, la

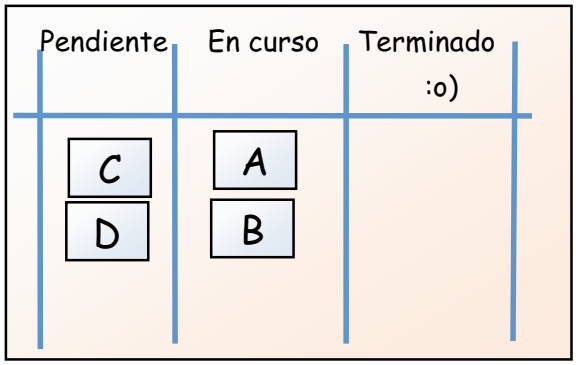
columna “Pendiente” da al equipo un pequeño *buffer* del que coger trabajo a medio plazo.

Experimenta o, como los “*Scrumistas*” dicen, Inspección y Adaptación.

7

Scrum se resiste a los cambios durante la iteración

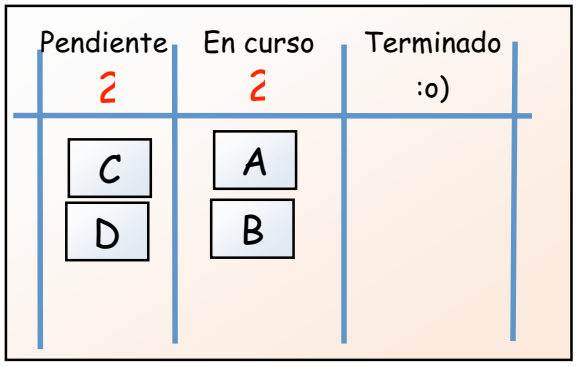
Digamos que nuestro tablero Scrum tiene este aspecto:



¿Qué ocurre si alguien cambia de idea y quiere añadir E al tablero?

Típicamente un equipo Scrum diría algo como “No, lo siento, nos hemos comprometido a A+B+C+D en este sprint. Pero tomate la liberad de añadir E a la pila de producto. Si el dueño del producto considera que es de alta prioridad será añadido al siguiente sprint”. Sprints de longitud adecuada permiten al equipo mantenerse enfocados suficiente tiempo como para lograr hacer algo, permitiendo además que el dueño del producto gestione y actualice prioridades de manera regular.

¿Pero entonces que es lo que dice el equipo Kanban?



La respuesta de Kanban podría decir "Añade con libertad E a la columna Pendiente pero hay un límite de 2 para esta columna, en consecuencia tendrás que quitar C o D. Ahora estamos trabajando en A y B, pero tan pronto como tengamos capacidad disponible cogeremos el primer elemento de la columna Pendiente".

El tiempo de respuesta (cuánto tiempo pasa hasta que respondemos a cambios de prioridades) de un equipo Kanban es tan largo como el tiempo que trascurre hasta que tenemos capacidad disponible, siguiendo el principio general de "un elemento sale = un elemento entra" (según los límites para el trabajo en curso).

En Scrum, el tiempo de respuesta medio es la mitad de la duración del Sprint.

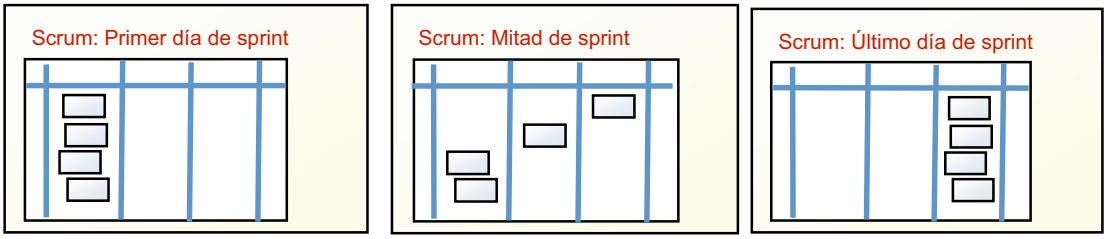
En Scrum, el dueño del producto no puede tocar el tablero Scrum una vez el equipo se ha comprometido a completar un conjunto de elementos en la iteración. En Kanban tú tienes que establecer tu propias reglas sobre quién puede cambiar qué en el tablero. Típicamente el dueño del producto controla una columna del tipo "Pendiente", "Listo", "Propuesto", "Pila de Producto" en la parte más a la izquierda en el panel, en la que él puede hacer todos los cambios que desee.

Sin embargo, estas dos aproximaciones no son excluyentes entre sí. Un equipo Scrum *podría* decidir permitir al dueño del producto cambiar prioridades a mitad de sprint. Y un equipo Kanban *podría* decidir añadir restricciones sobre cuando las prioridades pueden ser cambiadas. Un equipo Kanban puede incluso decidir utilizar iteraciones limitadas en el tiempo y con compromisos prefijados, exactamente igual que Scrum.

8

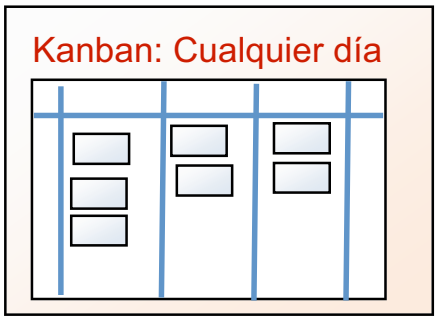
El tablero sprint se limpia entre iteraciones

Un tablero Scrum suele tener un aspecto similar a este durante las diferentes etapas de un sprint.



Cuando se finaliza un sprint, se limpia el tablero - todos los elementos son eliminados. Se inicia un nuevo sprint y tras la reunión de planificación del sprint tendremos un nuevo tablero Scrum, con nuevos elementos en la primera columna. Técnicamente esto es una pérdida de tiempo, pero para equipos Scrum experimentados no suele llevar mucho, y el proceso de limpiar el tablero puede dar una grata sensación de logro y cierre. Es algo así como lavar los platos tras la cena - hacerlo resulta pesado pero sienta bien al finalizar.

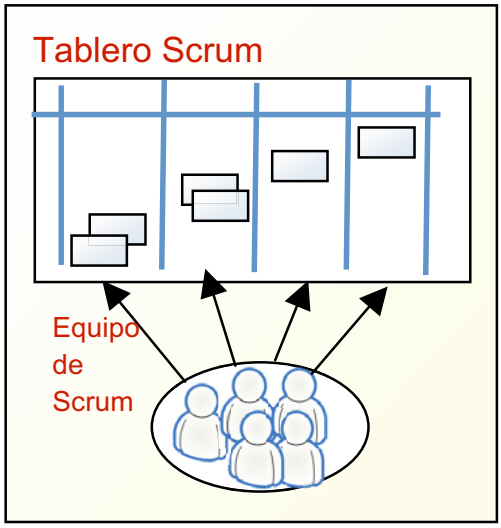
En Kanban, el tablero normalmente es algo persistente - no se necesita limpiarlo y volver a empezar.



9

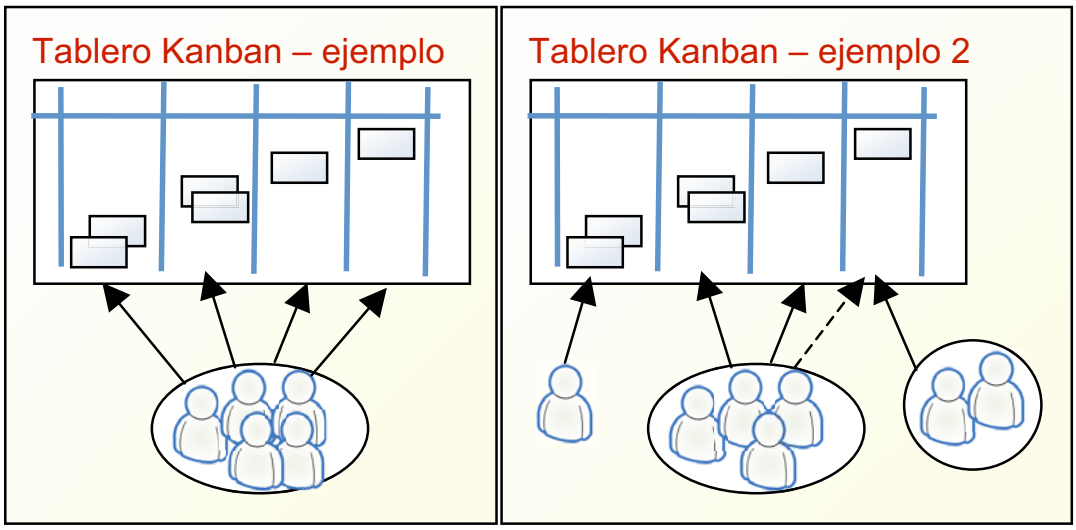
Scrum prescribe equipos multi-funcionales

Un tablero Scrum es propiedad de solamente un equipo Scrum. Un equipo Scrum es multi-funcional, contiene todos los conocimientos necesarios para completar todos los elementos de la iteración. Un tablero Scrum suele ser visible para cualquiera que esté interesado, pero sólo el equipo Scrum al que pertenece puede editarlo - es su herramienta para administrar su compromiso para esta iteración.



En Kanban, los equipos multi-funcionales son opcionales, y un tablero no necesita estar en manos de un equipo específico. Un tablero está relacionado con un flujo de trabajo, no necesariamente con un equipo.

He aquí dos ejemplos:



Ejemplo 1: Todo el tablero está al servicio de un equipo multi-funcional. Justo como en Scrum.

Ejemplo 2: El dueño de producto establece las prioridades en la columna 1. Un equipo de desarrollo multi-funcional realiza el desarrollo (columna 2) y prueba (columna 3). La puesta en producción (columna 4) es realizada por un equipo de especialistas. Existe un ligero solapamiento de competencias, de modo que si el equipo de puesta en producción se convierte en un cuello de botella uno de los desarrolladores les ayudará.

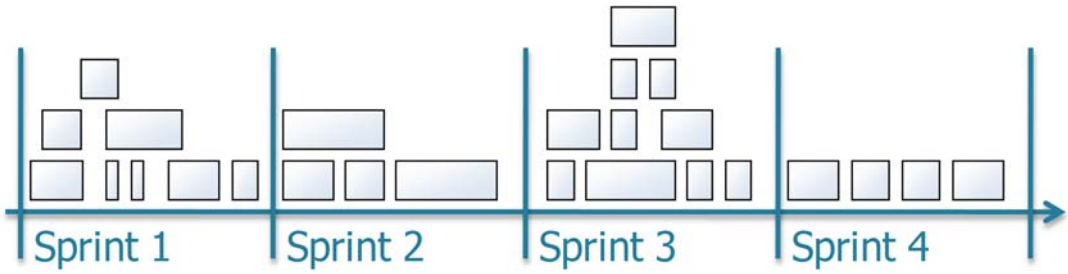
Así, en Kanban es necesario establecer algunas reglas básicas para quién usa el tablero y cómo, luego se experimenta con las normas para optimizar el flujo.

10

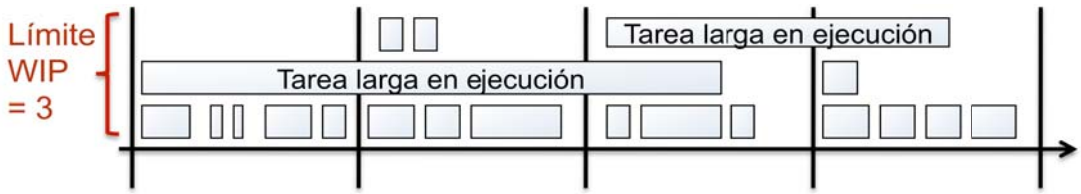
Los elementos de la pila de producto deben caber en un sprint

Ambos Scrum y Kanban se basan en el desarrollo incremental, por ejemplo, descomponer el trabajo en trozos más pequeños.

Un equipo Scrum sólo se comprometerá con los elementos que creen que pueden terminar en una iteración (basado en la definición de "Hecho"). Si un elemento es demasiado grande para caber en un sprint, el equipo y el propietario del producto intentarán encontrar la manera de partirlo en pedazos más pequeños hasta que se pueda abordar en un sprint. Si los elementos tienden a ser grandes, las iteraciones deberán ser más largas (aunque generalmente no más de 4 semanas).



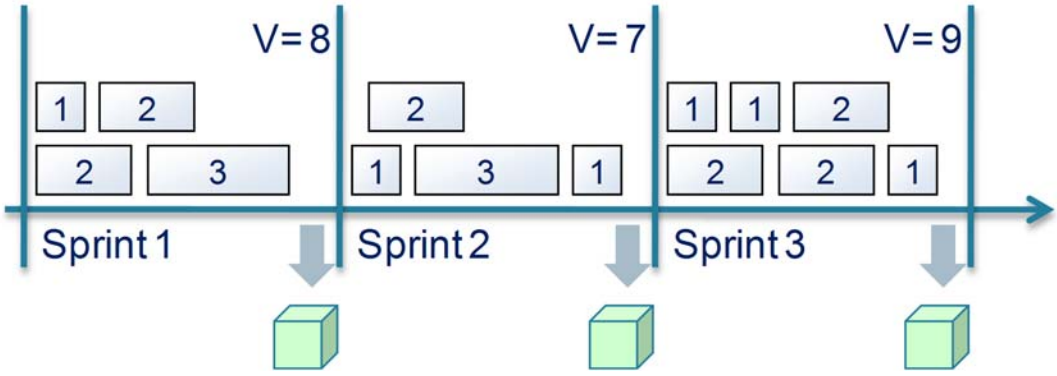
Los equipos de Kanban tratan de minimizar el tiempo de entrega y el nivel de flujo, por lo que, indirectamente, se crea un incentivo para descomponer los elementos en pedazos relativamente pequeños. Pero no hay ninguna norma explícita que indique que los elementos deben ser lo suficientemente pequeños como para caber en un intervalo de tiempo específico. En el mismo tablero podría haber un elemento que necesita un mes para terminarse y otro elemento que necesita un día.


Límite
WIP
= 3
Tarea larga en ejecución
Tarea larga en ejecución

11

Scrum prescribe la estimación y la velocidad

En Scrum, los equipos tienen que estimar el tamaño relativo (= cantidad de trabajo) de cada elemento al que se comprometen. Sumando el tamaño de cada elemento completado al final de cada sprint, obtenemos la velocidad. La velocidad es una medida de la capacidad - la cantidad de cosas que podemos ofrecer por Sprint. He aquí un ejemplo de un equipo con una velocidad media de 8.



Saber que la velocidad media es de 8 es bueno, porque entonces podemos hacer predicciones realistas sobre los elementos que se pueden completar en los próximos sprints, y por lo tanto hacer planes de entrega realistas.

En Kanban, no se prescribe la estimación. Así que si necesitas comprometerte necesitas decidir la forma de garantizar la previsibilidad.

Algunos equipos optan por hacer estimaciones y medir la velocidad como en Scrum. Otros equipos eligen omitir la estimación, pero tratan de descomponer cada elemento en partes de aproximadamente el mismo tamaño -entonces pueden medir la velocidad simplemente en función de cuántos elementos se completan por unidad de tiempo (por ejemplo, por semana).

Algunos equipos agrupan los elementos en MMF's (características mínimas de comercialización) y miden el tiempo de entrega promedio por MMF, y lo usan para establecer SLA (acuerdos de nivel de servicio,

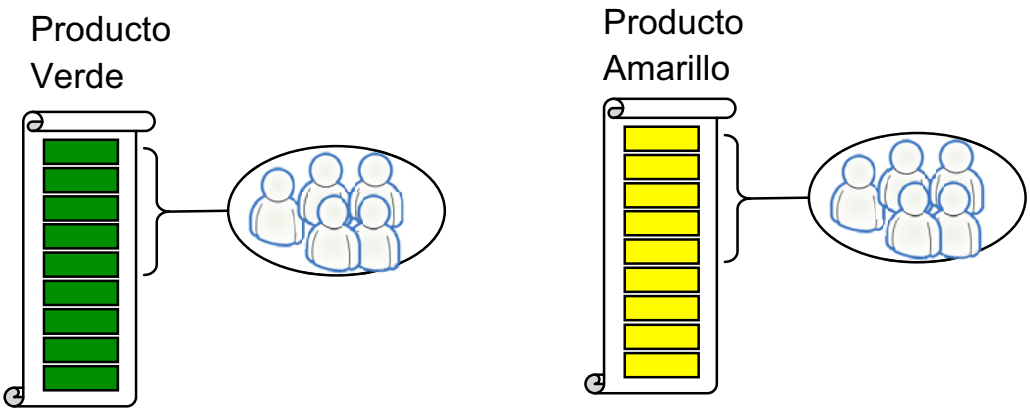
en inglés service-level agreements) - por ejemplo, "cuando nos comprometemos con un MMF siempre será entregado en 15 días ".

Hay todo tipo de técnicas interesantes para el estilo de planificación de entregas y gestión de compromisos de Kanban- pero no se prescribe ninguna técnica específica, así que adelante, googlea y prueba diferentes técnicas hasta que encuentres una que se adapte a su contexto. Probablemente veremos algunas "mejores prácticas" emerger con el tiempo.

12

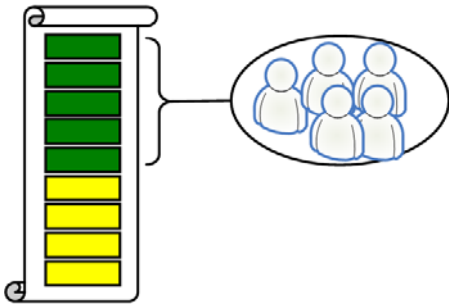
Ambos permiten trabajar en múltiples productos simultáneamente

En Scrum, la Pila de Producto es un nombre un tanto desacertado pues implica que todos los ítems deben pertenecer a un mismo producto. Aquí vemos dos productos, uno verde y el otro amarillo, cada uno con su respectiva pila y su respectivo equipo:

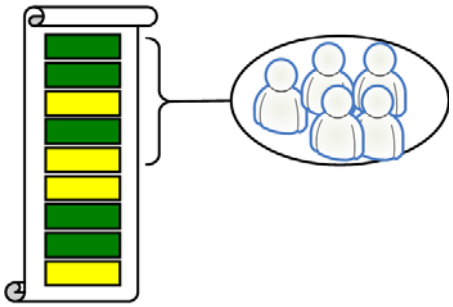


¿Que pasa si tenemos un sólo equipo para varios productos? Bien, pensemos en la Pila de Producto más bien como una Pila de Equipo. Lista las prioridades para las siguientes iteraciones del equipo (o conjunto de equipos) en particular. De esta forma, si un equipo mantiene más de un producto, debe fusionar ambos productos en una sola lista. Esto nos obliga a priorizar entre productos, lo cual puede resultar útil en algunos casos. Hay varias formas de hacer esto en la práctica:

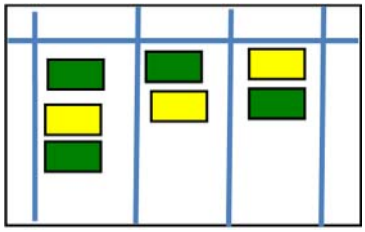
Una estrategia sería hacer que el equipo se enfoque en un sólo producto durante el sprint:



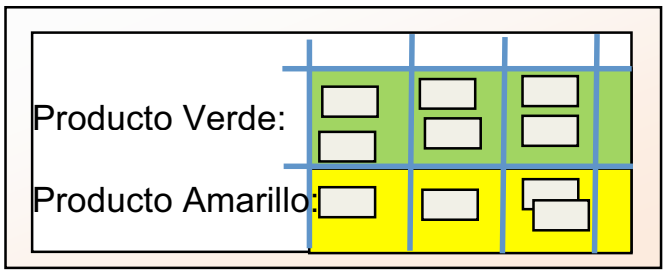
Otra estrategia sería hacer que el equipo trabaje en la funcionalidad de ambos productos cada sprint:



Es lo mismo en el caso de Kanban. Podemos tener varios productos fluyendo por el mismo tablero. Tal vez podemos distinguirlos utilizando tarjetas de diferentes colores:



... o mediante distintas “pistas” o “calles” (“*swinlanes*”) :



13

Ambos son Lean y Ágiles

No voy a revisar aquí el Pensamiento 'Lean', ni el Manifiesto Ágil. Pero se puede generalizar que tanto Scrum como Kanban están bien alineados con los valores y principios de éstos.

Por ejemplo:

- Tanto Scrum como Kanban son sistemas de planificación tipo "Pull", principio de gestión de inventario 'Just In Time' (JIT) propio de Lean. Esto significa que el equipo elige cuándo y cuánto trabajo acometer. Ellos (los componentes del equipo) "tiran" del trabajo cuando están listos, en contraposición a que desde el exterior se "empuje" al equipo a hacerlo. Al igual que una impresora tira de la siguiente página solo cuando esta lista para imprimir en ella (aunque tenga hojas de papel en la bandeja).

- Scrum y Kanban se basan en procesos de optimización continuos y empíricos, que se corresponden con el principio Kaizen de Lean.

- Scrum y Kanban dan más importancia a la respuesta al cambio que al seguimiento de un plan (aunque Kanban permite, típicamente, una respuesta más rápida que Scrum). Uno de los cuatro principios del manifiesto ágil.

...y mas.

Desde un determinado punto de vista, Scrum se puede ver como no-tan-lean, ya que contempla agrupar tareas por lotes dentro de iteraciones de tiempo prefijado. Pero eso depende de la longitud de tu iteración, y también de con qué se compare. En un proceso más tradicional, en el que quizás se integren versiones unas 2-4 veces al año, un equipo Scrum produciendo código entregable cada 2 semanas se puede considerar muy Lean.

Por consiguiente, si haces iteraciones cada vez más cortas, esencialmente te estás aproximando a Kanban. Una vez que se empieza

a hablar de hacer durar la iteración menos de una semana, se debería considerar abandonar definitivamente las iteraciones a tiempo cerrado.

Ya lo he dicho antes y lo seguiré diciendo: Experimenta hasta que encuentres algo que te funcione! y entonces continúa experimentando.

14

Diferencias menores

He aquí algunas diferencias que parecen ser menos relevantes en comparación con las mencionadas arriba. De todas formas, es bueno estar al tanto de ellas.

Scrum prescribe una Pila de Producto priorizada

En Scrum, la priorización se hace siempre ordenando la pila de producto y los cambios en las prioridades tienen efecto en el siguiente sprint (no en el actual). En Kanban, puedes elegir cualquier esquema de priorización (o incluso ninguno) y los cambios tienen efecto tan pronto como la capacidad está disponible (y no en períodos de tiempo preestablecidos). Puede que exista o no un pila de producto, y puede estar priorizada o no.

En la práctica, la diferencia es muy pequeña. En un tablero Kanban la columna de más a la izquierda típicamente satisface el mismo propósito que la pila de producto en Scrum. Tanto en el caso de que la lista esté ordenada por prioridad, como si no, el equipo necesita una regla de decisión que permita saber cuál es el siguiente elemento del que tirar. Ejemplos de reglas de decisión:

- Siempre toma la primera tarea.
- Siempre toma la tarea más antigua (cada tarea tiene que tener fecha, claro).
- Toma cualquier tarea.
- Emplea aproximadamente el 20% en tareas de mantenimiento y el 80% en nuevos desarrollos.
- Divide la capacidad del equipo, de forma aproximada, entre el producto A y el producto B.

- Siempre toma las tareas en rojo, si las hay.

En Scrum, la pila de producto también se puede usar al estilo Kanban. Podemos limitar su tamaño y crear las reglas de decisión sobre cómo se debería priorizar.

En Scrum se establecen reuniones diarias

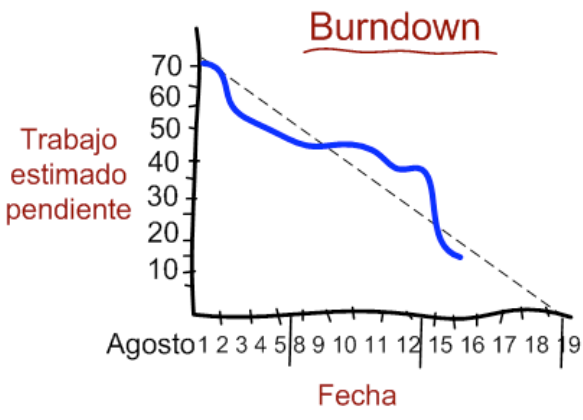
Un equipo Scrum tiene una reunión corta (de aproximadamente 15 minutos) cada día, a la misma hora y en el mismo lugar. El objeto de estas reuniones es compartir información sobre lo que está pasando, planificar el día de trabajo actual e identificar cualquier problema significativo. El término que se emplea a veces en inglés, 'daily standup', refleja el hecho de que normalmente se celebra de pie (para que sea breve y mantener un nivel alto de energía).

Las reuniones diarias no se utilizan en Kanban, pero la mayoría de los equipos Kanban parece que lo hacen de todos modos. Es una gran técnica, con independencia del proceso que utilices.

En Scrum, este formato de reunión está muy orientado a la persona. Cada miembro del equipo va haciendo su reporte uno a uno. Muchos equipos Kanban utilizan un formato más orientado al panel, poniendo el foco en los cuellos de botella y otros problemas visibles. Esta última aproximación es más escalable. Si tienes 4 equipos compartiendo el mismo panel y haciendo sus reuniones diarias juntas, puede que no sea necesario tener que escuchar hablar a cada uno en tanto y en cuanto nos enfoquemos en las partes que son cuellos de botella del panel.

En Scrum se usan diagramas de Burndown

Un gráfico *burndown* representa, diariamente, la cantidad de trabajo restante en la iteración actual.



La unidad del eje-Y es la misma que la utilizada en las tareas del sprint. Típicamente, horas o días (si el equipo convierte la pila en tareas) o puntos de historia (si el

equipo no lo hace). Existen muchas variaciones de esto.

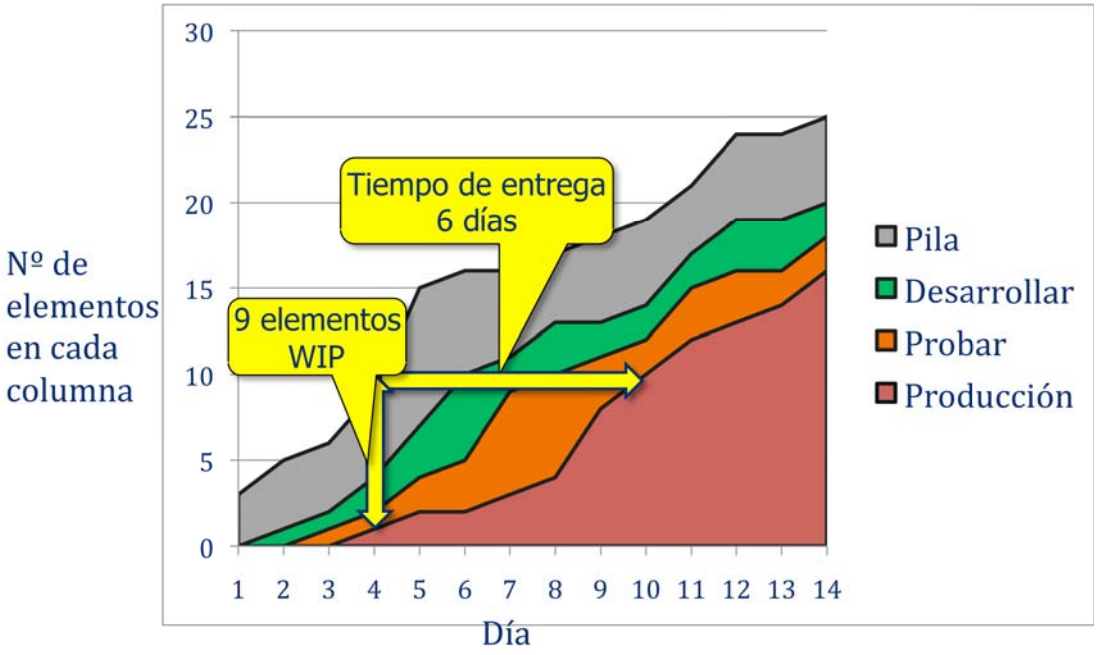
En Scrum, los gráficos *burndown* se usan como herramienta primordial para el seguimiento del progreso de una iteración.

Algunos equipos también utilizan gráficos *burndown* de versión que sigue el mismo formato pero a nivel de versión - típicamente muestran cuantos puntos de historia quedan pendientes en la pila después de cada sprint.

El principal propósito de un gráfico Burndown es encontrar, fácilmente y tan pronto como sea posible, si nos encontramos avanzados o retrasados respecto a planificación para poder adaptarnos.

En Kanban, no se prescriben gráficos burndown. De hecho, no existe ningún tipo particular de gráfico. Pero, desde luego que se permite utilizar cualquier tipo de gráfico que se quiera (incluyendo los *burndowns*).

A continuación se muestra un ejemplo de Diagrama de Flujo Acumulativo. Este tipo de gráficos representan finamente la suavidad del flujo y cómo el WIP afecta a tu plazo de entrega.



La flecha horizontal nos muestra que las tareas añadidas a la pila el día 4 costaron una media de 6 días en llegar a producción. Aproximadamente la mitad de ese tiempo fueron Test. Podemos ver que si se limitáramos

el WIP en Test y Pila, reduciríamos significativamente el plazo de entrega total.

La pendiente del área azul-oscura nos muestra la velocidad (por ejemplo, número de tareas desarrolladas al día). Con el tiempo podemos ver como velocidades mayores reducen el plazo de entrega, mientras que un mayor WIP lo incrementa.

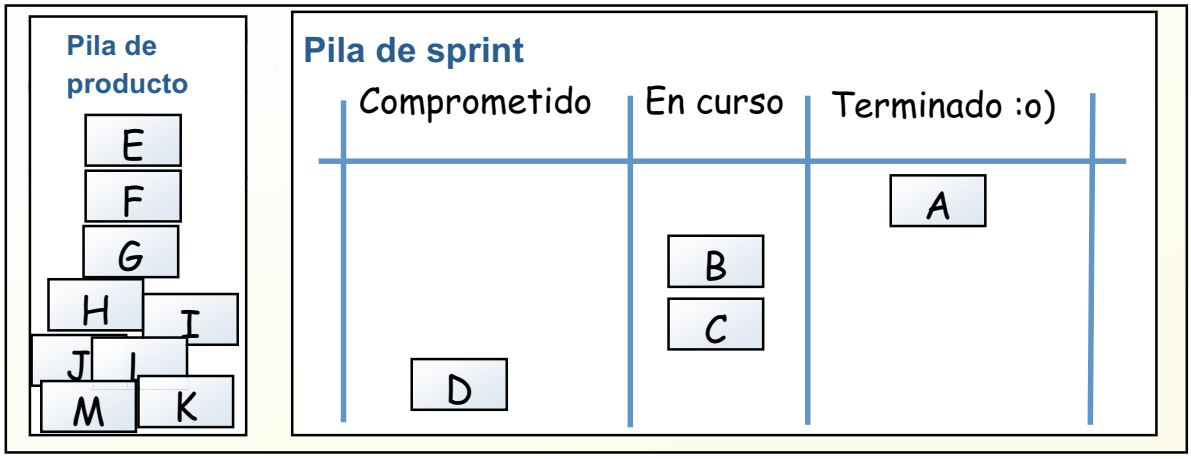
La mayoría de las organizaciones quieren tener realizado el trabajo más rápido (= reducir el plazo de entrega). Desafortunadamente muchas caen en la trampa de asumir que esto significa contratar más personas o trabajar horas extras. Normalmente la forma más efectiva de hacer el trabajo más rápidamente es suavizar el flujo y limitar el trabajo a la capacidad. No añadir más gente o trabajar más duro. Este tipo de diagrama muestra por qué. Y de este modo se incrementa la probabilidad de que el equipo y la gerencia colaboren de forma efectiva.

Esto se ve de forma aún más clara si distinguimos entre estados 'a cola' (como por ejemplo "esperando para test") y estados de 'trabajo' (como por ejemplo "Testeando"). Queremos, minimizar de forma absoluta el número de actividades esperando 'a cola'. Y el Diagrama de Flujo Acumulativo ayuda a proporcionar los incentivos adecuados para esto.

15

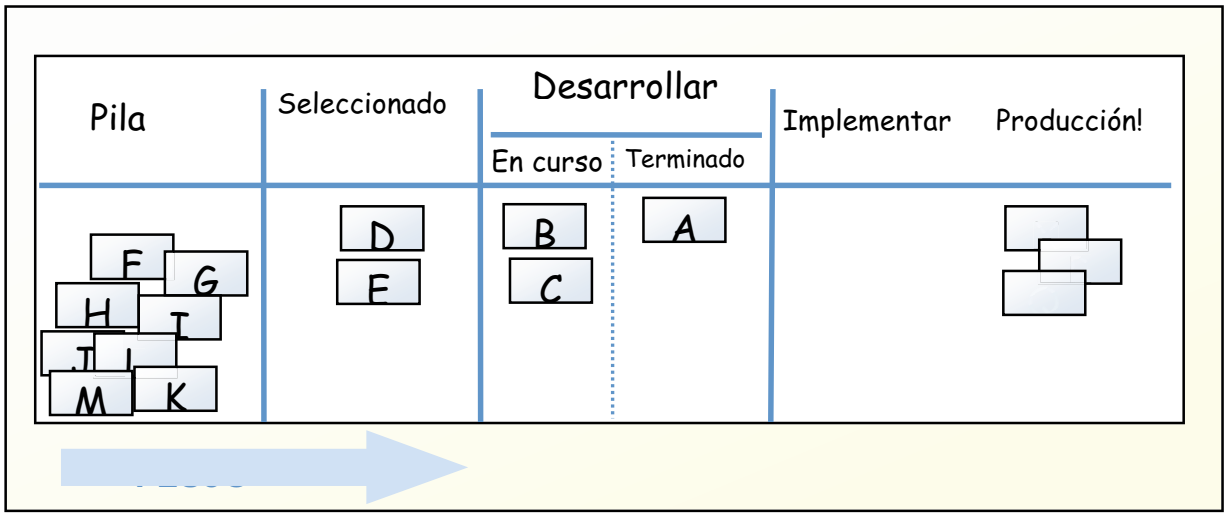
El tablero Scrum vs. el tablero Kanban – un ejemplo menos trivial

En Scrum, la pila de sprint es sólo una parte de la foto - la parte que muestra lo que está haciendo el equipo durante el sprint actual. La otra parte es la pila de producto - la lista de cosas que el dueño del producto quiere tener hecho en futuros sprints.



Cuando se hace el sprint, el equipo "facilita el código potencialmente entregable" al dueño del producto. De este modo, cuándo el equipo finaliza el sprint lo revisa y, con orgullo, muestra las características A, B, C y D al dueño del producto. El dueño del producto puede decidir ahora si quiere o no entregar el sprint. Esta última parte - el hecho de entregar el producto - no suele estar incluido en el sprint, y por lo tanto no es visible en la pila de sprint.

En este escenario, un tablero Kanban podría ser algo como esto:



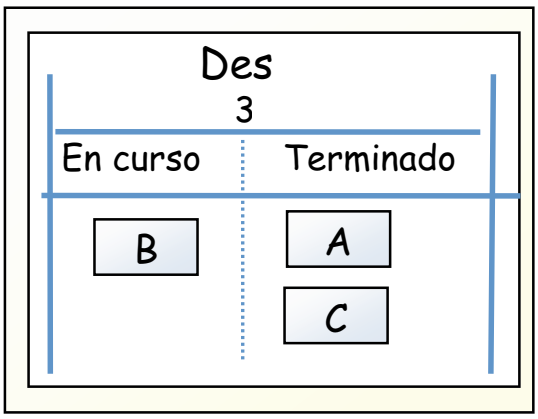
Ahora, el flujo de trabajo completo se encuentra en el mismo tablero - no sólo estamos mirando lo que un equipo Scrum está haciendo en una iteración.

En el ejemplo anterior la columna “Pila” es sólo una lista general de deseos sin ningún orden en particular. La columna "Seleccionado" contiene los elementos de prioridad alta, con un límite Kanban de 2. Por lo tanto, sólo puede haber 2 elementos de prioridad alta en un momento dado. Cada vez que el equipo esté listo para comenzar a trabajar en un nuevo elemento tendrá que tomar el primer elemento de la columna "Seleccionado". El dueño del producto puede hacer cambios en las columnas “Pila” y “Seleccionado” en cualquier momento, pero no en las otras columnas.

La columna "Des" (dividida en dos subcolumnas) muestra lo que se está desarrollando, con un límite Kanban de 3. En términos de red, el límite Kanban corresponde al "ancho de banda” y el tiempo de entrega corresponde a "ping" o tiempo de respuesta.

¿Por qué hemos dividido la columna "Des" en dos subcolumnas "En curso" y "Terminado"? Es para dar al equipo de producción la oportunidad de conocer los elementos que pueden arrastrar – pull – a producción.

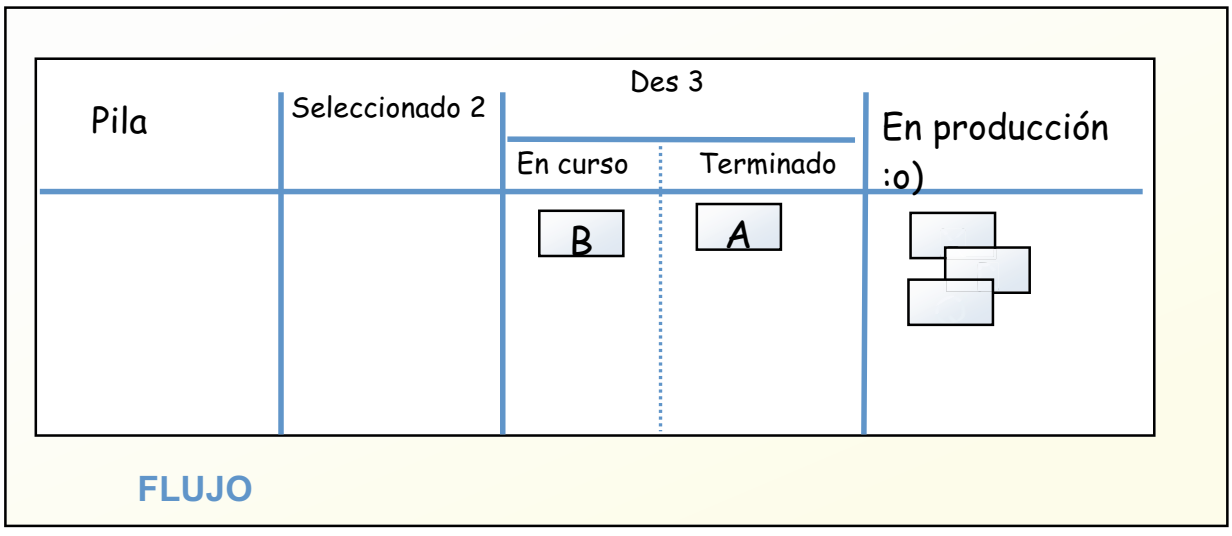
El límite "Des" de 3 es compartido entre las dos subcolumnas. ¿Por qué? Digamos que hay 2 artículos en "Terminado":



Esto significa que sólo puede estar 1 elemento “En curso". Por lo tanto, significa que habrá exceso de capacidad, los desarrolladores podrían comenzar un nuevo elemento, pero no están autorizados a causa del límite Kanban. Esto les da un fuerte incentivo para concentrar sus esfuerzos y ayudar a poner cosas en producción, para borrar de la columna "Terminado" y maximizar el flujo. Este efecto es agradable y progresivo - a más cosas en "Terminado", menos cosas se permiten “En curso" – lo que ayuda a centrar la atención del equipo en las cosas adecuadas.

Flujo de una sola pieza

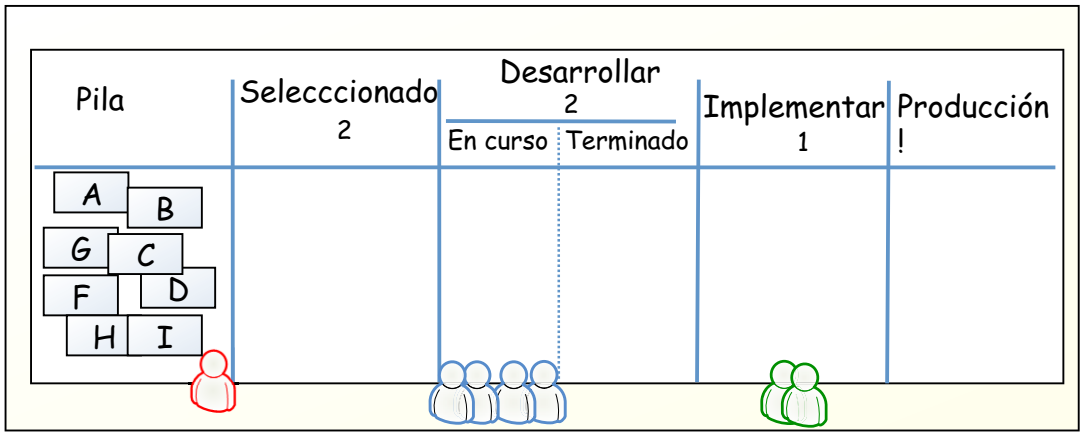
El flujo de una sola pieza es una especie de escenario de "flujo perfecto", donde un elemento fluye a través del tablero sin quedar atrapado en una cola. Esto significa que en cada momento hay alguien trabajando en ese elemento. Aquí puedes ver cómo el tablero podría representar este caso:

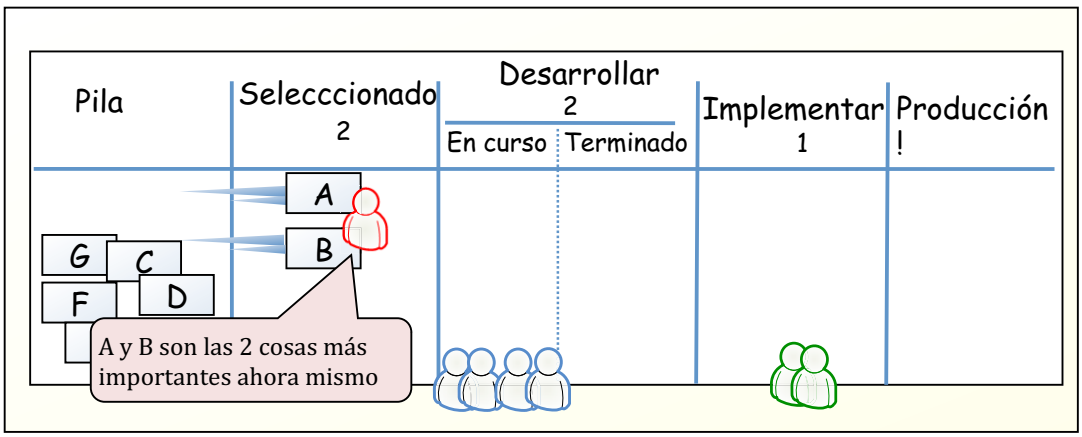


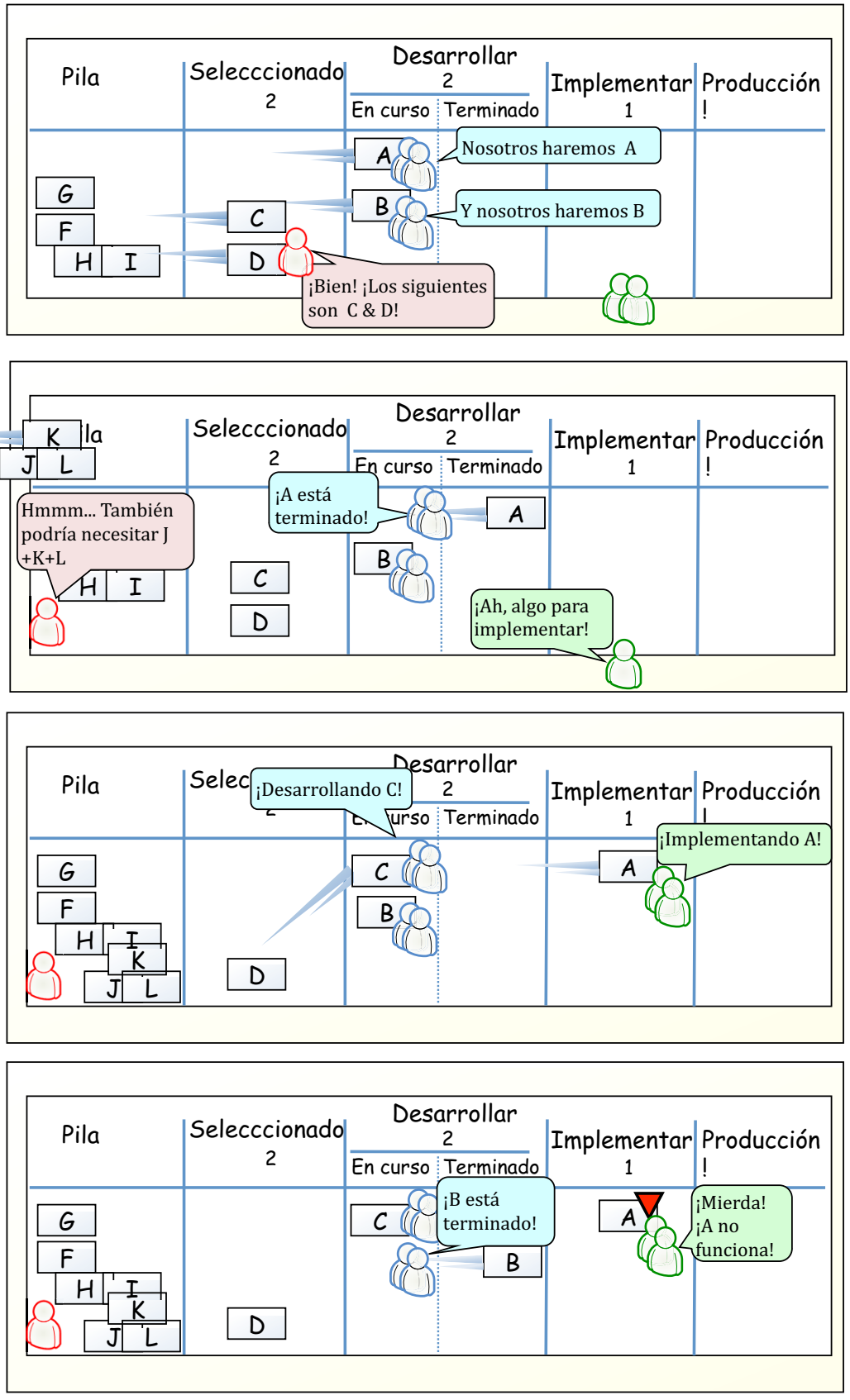
B se está desarrollando en este momento, A se ha puesto en producción en este momento. Cada vez que el equipo está listo para el siguiente elemento pregunta al dueño del producto qué es lo más importante y obtiene una respuesta. ¡Si este escenario ideal persiste podemos deshacernos de las dos colas "Pila" y "Seleccionado" y tener un tiempo de entrega muy corto!

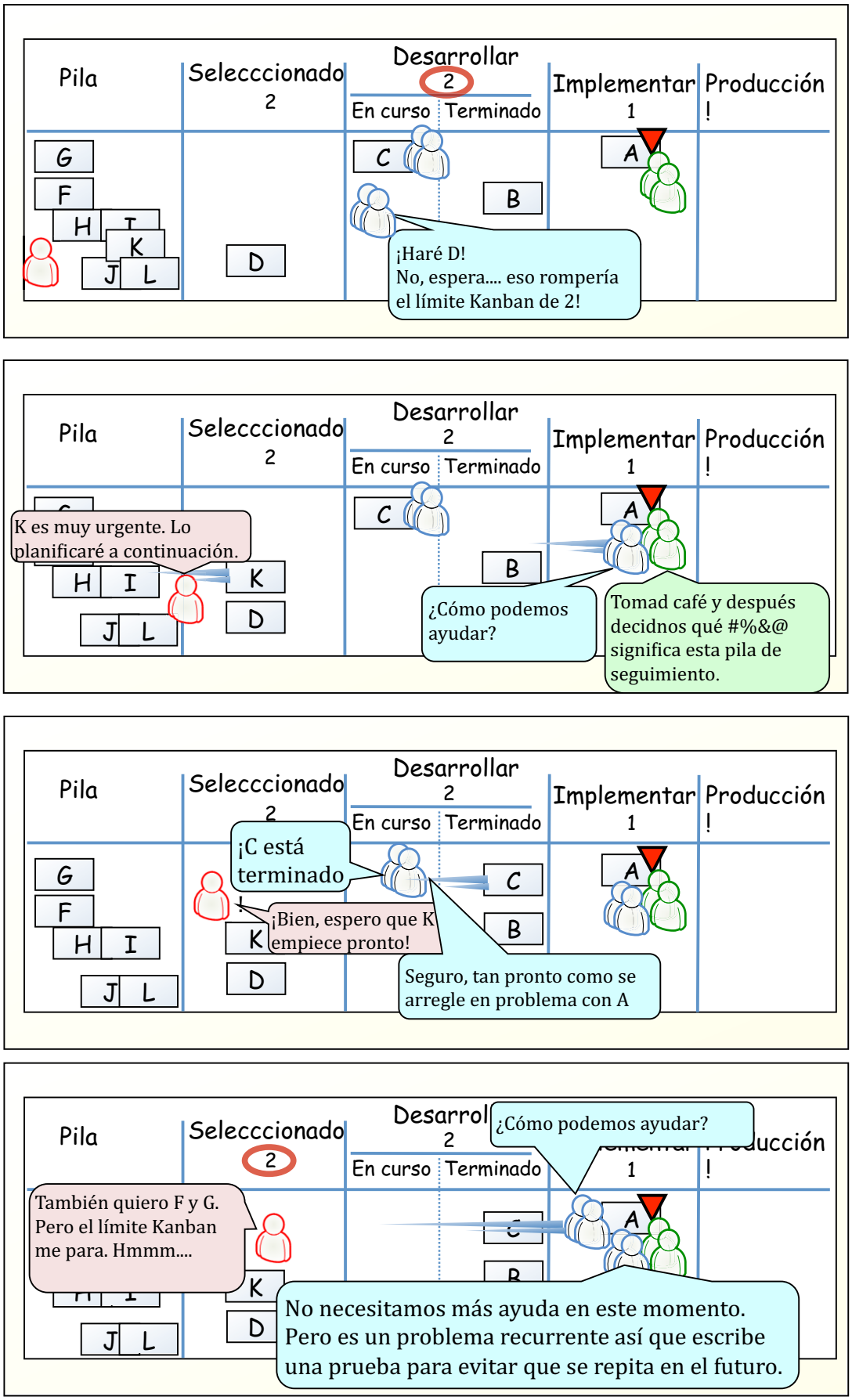
Cory Ladas lo cuenta muy bien: "El proceso ideal de planificación del trabajo siempre debe proporcionar al equipo de desarrollo lo más adecuado para continuar, ni más ni menos".
Los límites del trabajo en curso (WIP) están ahí para evitar problemas antes de que aparezcan. Así, si las cosas están fluyendo sin problemas, los límites del trabajo en curso (WIP) no son realmente utilizados.

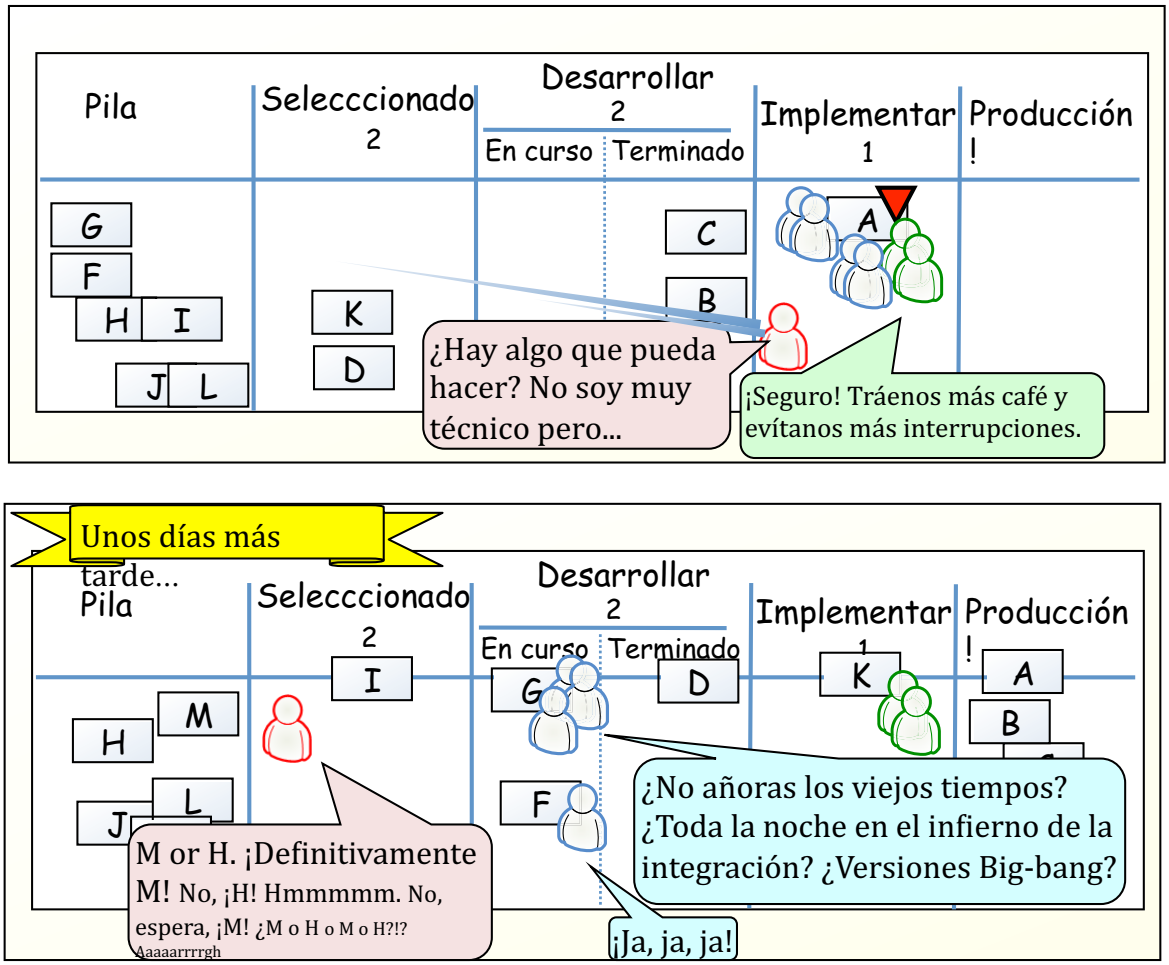
Un día en el país de Kanban






Pila
Seleccionado 2
Desarrollar 2
En curso
Terminado
Implementar 1
Producción !
Nosotros haremos A
Y nosotros haremos B
¡Bien! ¡Los siguientes son C & D!
Hmmm... También podría necesitar J +K+L
¡A está terminado!
¡Ah, algo para implementar!
¡Desarrollando C!
¡Implementando A!
¡B está terminado!
¡Mierda! ¡A no funciona!


Pila
Seleccionado
2
Desarrollar
2
En curso
Terminado
Implementar
1
Producción
!
G
F
H
I
K
J
L
D
C
B
A
¡Haré D!
No, espera.... eso rompería el límite Kanban de 2!
K es muy urgente. Lo planificaré a continuación.
¿Cómo podemos ayudar?
Tomad café y después decidnos qué #%&@ significa esta pila de seguimiento.
¡C está terminado
¡Bien, espero que K empiece pronto!
Seguro, tan pronto como se arregle en problema con A
¿Cómo podemos ayudar?
También quiero F y G. Pero el límite Kanban me para. Hmmm....
No necesitamos más ayuda en este momento.
Pero es un problema recurrente así que escribe una prueba para evitar que se repita en el futuro.



¿El tablero Kanban tiene que tener este aspecto?

No, el tablero anterior fue solo un ejemplo.

Lo único que prescribe Kanban es que el flujo de trabajo debe ser visual, y que el trabajo en curso debe estar limitado. El objetivo es crear un flujo suave a través del sistema y minimizar el tiempo de entrega. Por lo tanto, necesitas plantearte regularmente cuestiones tales como:

¿Qué columnas deberíamos tener?
Cada columna representa un estado del flujo de trabajo, o una cola (buffer) entre dos estados del flujo de trabajo. Empieza de forma sencilla y añade nuevas columnas cuando lo necesites.

¿Cuáles deberían ser los límites Kanban?

Cuando el límite Kanban para "tu" columna se ha alcanzado y no tienes nada que hacer, empieza a buscar un cuello de botella aguas abajo (es decir, acumulando puntos a la derecha de la pizarra) y ayuda a resolver el cuello de botella. Si no hay un cuello de botella puede que sea una indicación de que el límite Kanban pueda ser demasiado bajo, ya que la razón de tener el límite es reducir el riesgo de alimentación de los cuellos de botella derivados.

Si notas que muchos elementos no son atendidos durante mucho tiempo sin que se esté trabajando, puede ser una señal de que el límite de Kanban puede ser demasiado alto.

- Límite Kanban demasiado bajo => gente ociosa => mala productividad
- Límite Kanban demasiado alto => tareas ociosas => mal tiempo de respuesta

¿Cómo de estrictos son los límites Kanban?

Algunos equipos los toman como normas estrictas (es decir, el equipo no podrá superar el límite), otros equipos los toman como directrices o desencadenantes de una discusión (es decir, romper un límite Kanban está permitido, pero debe ser una decisión intencional, con un motivo concreto). Así que una vez más, te toca a ti. ¿Ya te dije que Kanban no era muy prescriptivo?

16

Resumen de Scrum vs Kanban

Parecidos

- Ambos son Lean y Ágiles.
- Ambos emplean sistemas de planificación "pull".
- Ambos establecen límites WIP.
- En ambos la visibilidad del proceso es la base de su mejora.
- Ambos tienen como objetivo la entrega temprana y frecuente de software.
- Ambos trabajan con equipos auto-organizados.
- Ambos necesitan la división del trabajo en partes.
- Ambos revisan y mejoran de forma continua el plan del producto en base a datos empíricos (velocidad / tiempo de entrega)

Diferencias

Scrum	Kanban
Las iteraciones deben ser de tiempo fijo.	**El tiempo fijo en las iteraciones es opcional.** La cadencia puede variar en función del plan del producto y la mejora del proceso. Pueden estar marcadas por la previsión de los eventos en lugar de tener un tiempo pre-fijado.

El equipo asume un **compromiso** de trabajo por iteración.	**El compromiso es opcional**.
La métrica por defecto para la planificación y la mejora del proceso es la **Velocidad**.	La métrica por defecto para la planificación y la mejora del proceso es el **Lead Time** (**tiempo de entrega** o tiempo medio que tarda una petición en salir del ciclo)
Los **equipos deben ser multi-funcionales**.	Los equipos **pueden ser multi-funcionales o especializados**.
Las funcionalidades deben dividirse en **partes que puedan completarse en un sprint**.	No hay ninguna prescripción en cuanto al tamaño de las divisiones.
Deben emplearse gráficos Burndown.	No se prescriben diagramas de seguimiento concretos.
Se emplea una **limitación WIP indirecta** (por sprint).	Se emplea una **limitación WIP directa** (marcada por el estado del trabajo).
Se deben realizar estimaciones.	**Las estimaciones son opcionales**.
No se pueden añadir tareas en medio de una iteración.	Siempre que haya capacidad disponible, **se pueden añadir tareas**.
La pila del sprint pertenece a un equipo determinado.	Varios equipos o personas pueden **compartir la misma pizarra Kanban**.
Se prescriben 3 roles (PP/SM/Equipo).	**No hay roles prescritos**.
En cada sprint se limpia el tablero de seguimiento.	**El tablero Kanban es persistente**.
La pila del producto debe estar priorizada.	**La priorización es opcional.**

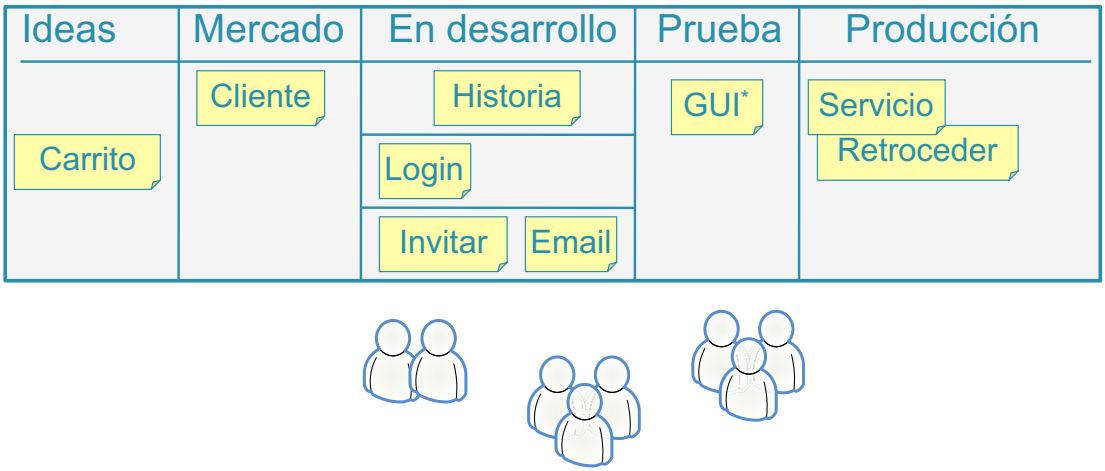
Ala. Esto es todo. Ahora ya conoces las diferencias.

Sin embargo, aún no ha terminado, ¡ahora es el momento de la mejor

parte!. ¡Cálzate las botas, porque es hora de ir a las trincheras con Mattias para ver cómo se pone esto en práctica!

Parte II – Caso de estudio

Kanban en la vida real



* Graphical User Interface: Interfaz Gráfica de Usuario

Esta es la historia de cómo hemos aprendido a mejorar usando Kanban. Cuando empezamos no había mucha información, y el Dr. Google nos dejó con las manos vacías. Hoy en día Kanban está evolucionando satisfactoriamente y hay un cuerpo de conocimiento incipiente. Recomiendo echar un vistazo al trabajo de David Anderson, por ejemplo a "classes of service". Así que aquí viene la primera (y última) advertencia (¡lo prometo!). Independientemente de la solución que vayas a implementar, asegúrate de tratar sus problemas específicos. Una vez dicho, vamos con nuestra historia.

/Mattias Skarin

17

La naturaleza de las operaciones técnicas

Si alguna vez has estado en un servicio de soporte 24/7, tendrás una idea clara de la responsabilidad que se siente al gestionar un entorno de producción. Se espera que resuelvas la situación en medio de la noche, sin importar si el problema era por tu culpa o no. Nadie lo sabe, por eso te llaman. Es un reto complicado porque tú no eres el fabricante del hardware, los drivers, el sistema operativo ni el software del usuario. A menudo tus opciones se limitan a acotar el problema o su impacto, y a registrar la información necesaria para poder repetir el error, de manera que la persona responsable pueda solucionar el problema del que tu has sido testigo.

La respuesta y la capacidad para resolver problemas son claves, siempre con velocidad y precisión.

18

¿Por qué diablos cambiar?

En 2008 uno de nuestros clientes, una empresa escandinava de desarrollo de juegos, pasó por un por una serie de mejora de procesos. Uno de ellos incluía escalar Scrum a la empresa de desarrollo y uno por uno eliminar los obstáculos que impedían al equipo de desarrollo la entrega de software. Cuando el software empezó a fluir y el rendimiento mejoró, se desplazó la presión hacia el grupo de Operaciones. Anteriormente, los equipos de Operación habían observado el proceso desde fuera, pero ahora estaban cada vez más involucrados como parte activa del proceso de desarrollo.

Figura 1. La organización del grupo de Operación incluía tres equipos; los administradores de bases de datos (DBAs) los administradores de sistemas y la segunda línea de soporte.

Así que ayudar a los equipos de desarrollo no era suficiente. Si manteníamos el foco sólo en los equipos de desarrollo causaríamos retrasos en las mejoras de infraestructura críticas realizadas por los equipos de Operación. Por lo tanto se necesitaban mejoras en ambas áreas.

Además, el progreso en los equipos de desarrollo significaba un aumento de las peticiones a los gerentes para echar una mano en el análisis y la revisión de ideas. Esto significaba que tenían menos tiempo para la priorización de tareas y resolución de problemas en el día a día. El equipo de gerentes se dio cuenta de que necesitaban actuar antes de que la situación se volviera inmanejable.

19

¿Por dónde empezamos?

Una buena forma de empezar era preguntar a los equipos de desarrollo, porque eran los clientes del grupo de Operaciones.

El equipo de operaciones, visto por los desarrolladores

Les pregunté; "¿qué tres cosas os vienen a la cabeza cuando pensáis en 'Operaciones'?". Las respuestas más comunes fueron:

"Conocimiento variable"

"Muy competentes cuando se trata de infraestructura"

"Ellos quieren ayudar, pero en realidad es difícil obtener ayuda"

"Los proyectos duran demasiado"

"Su sistema de workflow apesta"

"¿Qué están haciendo los chicos?"

"Necesito muchos correos electrónicos para hacer cosas simples"

"Difíciles de contactar"

En resumen, así era cómo los desarrolladores veían al equipo de Operación. Comparémoslo con la visión de Desarrollo que tenía el equipo de Operación:

Desarrollo, visto desde el equipo de operaciones

"Nosotros"
(Operaciones)

"Ellos"
(desarrollo)

“¿Por qué no utilizáis las ventajas que os ofrecen las plataformas?”

“¡Hagamos de la distribución algo menos pesado!”

“¡Vuestra baja calidad nos afecta!”

"Tienen que cambiar" - era el tema común en las quejas de ambos equipos. Obviamente, debíamos cambiar ese esquema mental si queríamos avanzar en la resolución de los problemas comunes. Por lo menos, "*muy competentes cuando se trata de infraestructura*" indicaba confianza en las capacidades y me hacía pensar que esa mentalidad de "*nosotros contra ellos"* podía arreglarse si se creaban las condiciones de trabajo adecuadas. Una opción viable podía ser eliminar el sobre-esfuerzo y concentrarnos en la calidad.

20

Iniciando la marcha

Así que necesitábamos iniciar la marcha, pero ¿por dónde empezamos? La único cierto que sabíamos es: el sitio donde empecemos no será donde terminemos.

Mi experiencia es la de un desarrollador, así que seguramente sabía muy poco sobre la naturaleza del trabajo de Operaciones. Y no era cuestión de entrar a lo loco y comenzar a cambiar cosas. Necesitaba un enfoque menos frontal que aún así nos enseñara las cosas relevantes, descartara las no relevantes y fuera fácil de aprender.

Los candidatos eran:

1. Scrum - estaba funcionando bien en equipos de desarrollo.
2. Kanban - era nuevo y no estaba probado, aunque encaja bien con los principios Lean que se echaban en falta.

En discusiones con los gerentes los principios de Kanban y Lean parecían encajar con los problemas que estábamos intentando resolver. Desde su punto de vista los sprints no se adaptarían muy bien ya que hacían repriorización de forma diaria.

Por lo tanto Kanban parecía el punto de partida más lógico, incluso aunque fuera algo nuevo para todos nosotros.

21

Puesta en marcha de los equipos

¿Cómo poner en marcha los equipos? No había ningún manual que dijera cómo hacerlo. Y hacerlo mal podía ser muy arriesgado. Aparte de perder las mejoras, teníamos que tratar con una plataforma de producción con personal altamente especializado y cualificado; difíciles de reemplazar. Dejarlos de lado era una idea muuuy mala.

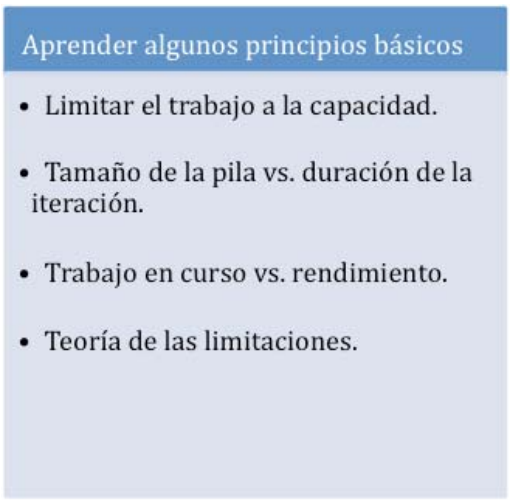
- ¿Deberíamos simplemente ponernos en marcha? ¿Asumir las consecuencias según vayan surgiendo?
- O bien, ¿hacer primero un taller?

Era obvio para nosotros - Hacer el taller, ¿verdad? Pero ¿cómo?

Era un reto conseguir que todo el equipo de soporte participara en un taller (¿Quién contestará al teléfono si alguien llama?). Al final decidimos hacer un taller de medio día, y mantenerlo sencillo y basado en ejercicios.

El taller

Uno de los beneficios del taller era que ayudaría a poner de manifiesto cuanto antes nuestros problemas. Pero además proporcionó un ambiente de alta confianza donde las implicaciones podían ser discutidas directamente con los miembros del equipo. Porque, seamos sinceros, no todo el mundo era demasiado entusiasta acerca de cambiar la actual forma de trabajo. Pero la mayoría de los miembros del equipo estaban abiertos a intentarlo, así que se llevó a cabo un taller de demostración de los principios más importantes y se hizo una simulación a escala de Kanban.



Al final del taller hicimos una votación a mano alzada para comprobar si los equipos estaban dispuestos a ponerlo en práctica. No se plantearon objeciones a este punto, así que teníamos el visto bueno para seguir adelante.

22

Dirigiéndonos a los involucrados

Era muy probable que otras partes interesadas (soporte, clientes, otros equipos, ...) se vieran afectadas por la aplicación de Kanban. Aunque los cambios serían para mejor, significaría que el equipo comenzaría a decir "no" al trabajo que no podían completar, a defender la calidad, y a eliminar tareas de baja prioridad de la pila del equipo.
A pesar de ello, tener una discusión previa siempre es una buena idea.

Los interesados más cercanos eran la primera línea de soporte y los gerentes de departamento. Como habían participado en el taller, ya eran partidarios de seguir adelante. Lo mismo para los equipos de desarrollo (que esperaban en mayor o menor medida las mejoras). Pero, para un equipo, el equipo de soporte, las cosas eran diferentes. Su problema más importante era que estaban sobrecargados de trabajo. Además, ellos gestionaban las solicitudes de los clientes y la empresa se había comprometido a responder a todas las solicitudes.

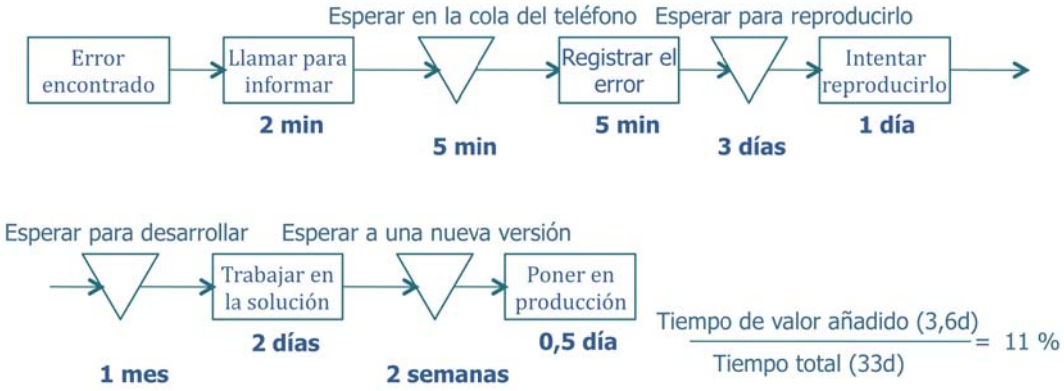
Ahora esto era muy probable que cambiara si aplicábamos Kanban y se comenzábamos a cumplir los límites de WIP (trabajo en curso). Así que hicimos una visita a los principales interesados para presentarles nuestras intenciones, beneficios esperados, y posibles consecuencias.

Para mi alivio, nuestras ideas fueron muy bien acogidas, a veces con una observación: "*estupendo si finalmente podemos dejar estos asuntos en paz*".

23

Construyendo el primer tablero

Una buena manera de comenzar la construcción de un tablero de Kanban es haciendo un mapa de la cadena de valor (value stream map). Se trata básicamente de una visualización de la cadena de valor y proporciona la comprensión de los estados de los trabajos, el flujo y el tiempo a en el sistema (tiempo del ciclo).



Pero empezamos por algo mucho más simple; un tablero de ejemplo Kanban dibujado en papel junto con el gerente. Revisado un par de veces y luego nos pusimos en marcha. Las cuestiones que surgieron en esta etapa incluyen:

- ¿Qué tipos de trabajo tenemos?
- ¿Quién los maneja?
- ¿Debemos compartir la responsabilidad entre los diferentes tipos de trabajo?
- ¿Cómo podemos hacer frente a la responsabilidad compartida teniendo en cuenta que tenemos conocimientos especializados?

Dado que para los diferentes tipos de trabajo había acuerdos de nivel de servicios diferentes, parecía natural permitir que cada equipo llevara el control de su propio tablero. Ellos mismos hicieron las filas y columnas.

La siguiente gran decisión era si utilizar o no una responsabilidad compartida entre los diferentes tipos de trabajo.

"¿Debemos dejar que una parte fija del equipo trate con las preguntas directas (trabajo reactivo) y dejar que el resto del equipo se centre en los proyectos (trabajo proactivo)?"

Decidimos en un primer momento probar la responsabilidad compartida. Una razón fundamental era que habíamos identificado que la auto-organización y el crecimiento de los miembros del equipo eran esenciales para el crecimiento sostenido. El inconveniente de esta decisión eran las potenciales interrupciones para todo el mundo, pero esta era la mejor solución que pensamos para comenzar.

Una pequeña nota al margen: cuando realizamos el taller los equipos se organizaron solos para solucionar este problema. Dejaron que una persona a manejara las solicitudes inmediatas y el resto las cuestiones más extensas.

El primer modelo Kanban

A continuación se muestra el modelo básico que utilizamos para Kanban. Tenga en cuenta que el equipo decidió poner el flujo del producto hacia arriba (como burbujas en el agua) en lugar de usar el modelo de flujo más típico de izquierda a derecha.

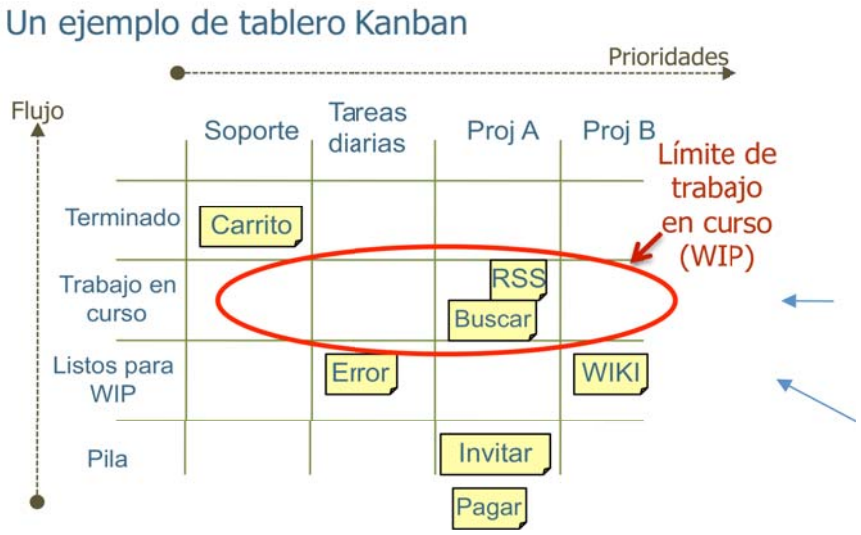


Figura 2. Este es el primer modelo de Kanban. Prioridades de ejecución de izquierda a derecha, flujo de ejecución hacia arriba. Trabajos en curso contados como el número total de tareas en la

línea de trabajo en curso (con círculo rojo). El modelo está influenciado por las experiencias transmitidas por Linda Cook.

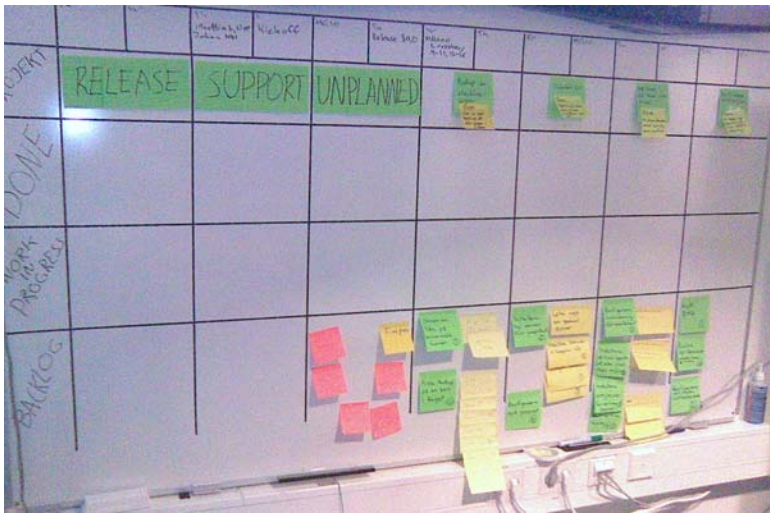


Figura 3. Primer tablero Kanban para el equipo de administración de sistemas.

Filas utilizadas:

Estado del flujo de trabajo (fila)	Como lo definimos
Backlog (Pila de producto)	Historias que el gerente decide que son necesarias.
Listo para WIP	Historias estimadas y divididas en tareas con una duración máxima de 8 horas.
Trabajo en curso	Fila que contenía un límite para el trabajo en curso. Empezamos con un límite de 2 X tamaño del equipo -1 (-1 para colaboración). Así, un equipo de 4 personas tendría un límite de 7.
Hecho	Ejecutable por el usuario.

Columnas utilizadas:

Tipo de trabajo	Como lo definimos
Release (liberación)	Ayudar a los equipos de desarrollo a liberar software.
Soporte	Pequeñas peticiones de otros equipos.
No planificado	Trabajo imprevisto que es necesario realizar pero que no tiene un propietario definido. Por ejemplo, pequeñas mejoras en la infraestructura.
Proyecto A	Proyectos grandes de soporte técnico, por ejemplo cambiar el hardware del entorno de pruebas.
Proyecto B	Otro proyecto largo

No todos los tableros Kanban tenían la misma apariencia. Todos comenzaron con un boceto simple y evolucionaron por el camino.

24

Estableciendo el primer límite de WIP

Nuestro primer límite de WIP (trabajo en curso) era bastante generoso. Supusimos que mediante la visualización del flujo veríamos y experimentaríamos lo que iba sucediendo y que era poco probable que fuéramos capaces de adivinar el límite óptimo desde el principio. Según pasara el tiempo, ajustaríamos los límites de WIP cada vez que encontráramos una buena razón para ello (sólo hacía falta apuntarlo en el tablero).

El primer límite WIP que usamos fue 2n-1 donde “n” era el número de miembros del equipo y “-1” un modo de potenciar la cooperación. ¿Por qué decidimos este límite? Simplemente, porque no teníamos una idea mejor J. Además, parecía algo poco controvertido para empezar. La fórmula proporcionaba una explicación simple y lógica para cualquiera que intentara cargar más trabajo al equipo: “...si cada miembro del equipo sólo puede trabajar en un máximo de dos cosas a la vez, una activa y otra en espera... ¿cómo crees que pueden aceptar más?". Mirando ahora hacia atrás, cualquier límite generoso habría funcionado para un equipo novato. Monitorizando el tablero Kanban es sencillo descubrir los límites correctos sobre la marcha.

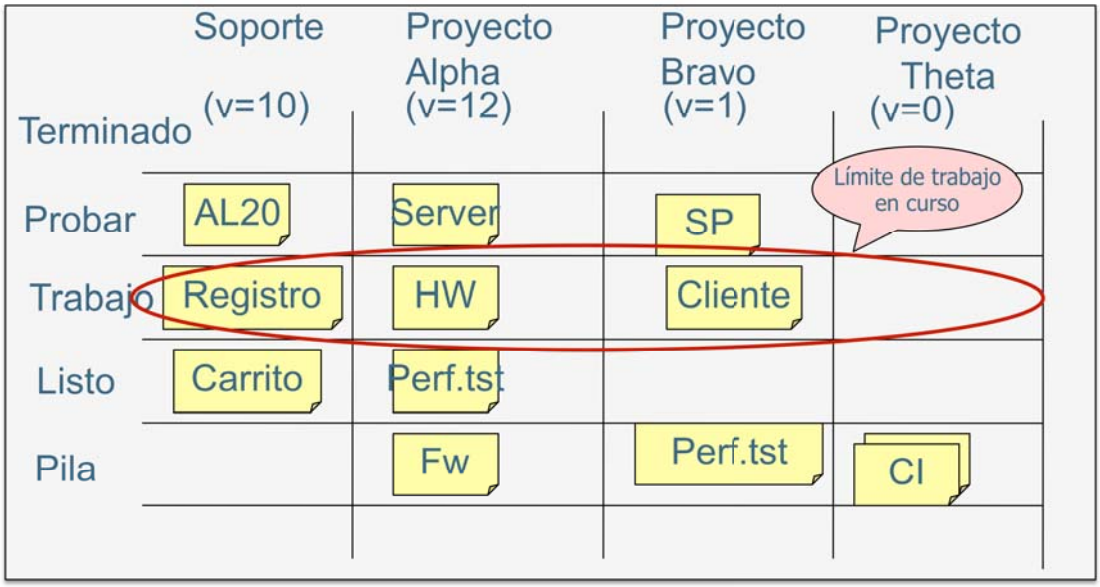


Figura 4. Cómo aplicábamos el límite de WIP para el equipo de DBA y administración de sistemas, con un límite por cada tipo de trabajo

Pronto descubrimos que no era útil definir el límite WIP en puntos de historia porque era difícil de controlar. El único límite suficientemente sencillo de controlar era simplemente la cuenta del número de elementos del panel, es decir, el número de tareas en paralelo.

Para el equipo de suporte establecimos un límite WIP por columna, porque necesitábamos capacidad de reacción rápida si el límite se sobrepasaba.

25

Respetando el límite WIP

Aunque respetar el límite WIP establecido suena fácil en teoría, es una tarea difícil de conseguir en la práctica, porque siempre significa decir "no" en algún momento. Pusimos en práctica diferentes aproximaciones para conseguir esto.

Discutir frente al tablero

Cada vez que se descubría una violación de los límites WIP llevábamos a las partes interesadas ante el tablero Kanban y les preguntábamos qué pretendían conseguir. Al principio, la razón más habitual para esas violaciones era simple inexperiencia. En algunos casos nos encontramos con perspectivas diferentes sobre la priorización, siendo típicos los casos de especialistas trabajando sobre su área específica. Estas fueron las únicas ocasiones en que experimentamos alguna fricción, porque la mayoría de las veces estas violaciones se solucionaban allí mismo discutiéndolo a la vista del tablero.

Creando una sección de sobrecarga

Cuando decir "no" suponía demasiada confrontación, y eliminar elementos del tablero era complicado, movíamos los elementos de menor prioridad a una sección de "sobrecarga" en el tablero, allí donde los límites WIP se habían sobrepasado. Dos reglas aplicaban a las tareas en sobrecarga:

1. No han sido olvidadas. En cuanto tengamos tiempo las trataremos.

2. Si las abandonamos, serás informado.

Justo pasadas dos semanas se hacía obvio que los elementos de sobrecarga incluso ni se tratarían nunca, de forma que con el apoyo del jefe de equipo finalmente se podrían quitar.

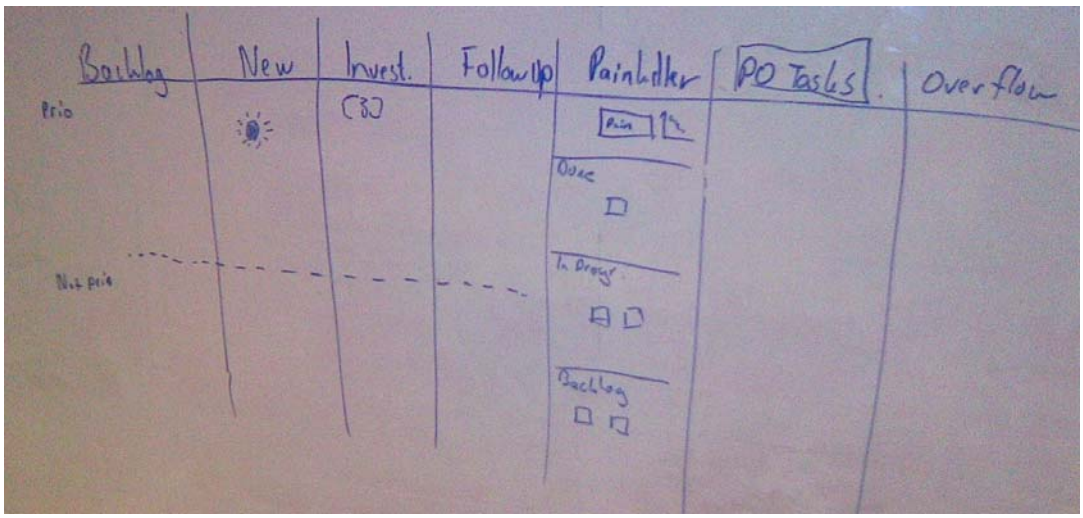


Figura 5. Boceto del tablero Kanban para el equipo de soporte, con la sección de sobrecarga en la última columna.

26

¿Qué tareas llevar al tablero?

Pronto tomamos la decisión de no incluir todo el trabajo hecho por el equipo en el tablero. Monitorizar cosas como una llamada telefónica o tomar un café convertiría el Kanban en un monstruo administrativo. Estábamos aquí para resolver problemas, no para crearlos J. Así que decidimos sólo poner en el tablero las tareas con tamaño mayor de una hora. Todo lo menor de una hora se consideraba "ruido blanco".

El límite de 1h realmente funcionó bastante bien, y fue de las pocas cosas que permanecieron sin cambios a lo largo del tiempo (tuvimos que revisar nuestras previsiones acerca del impacto que el ruido de fondo tenía, pero hablaremos de eso más tarde).

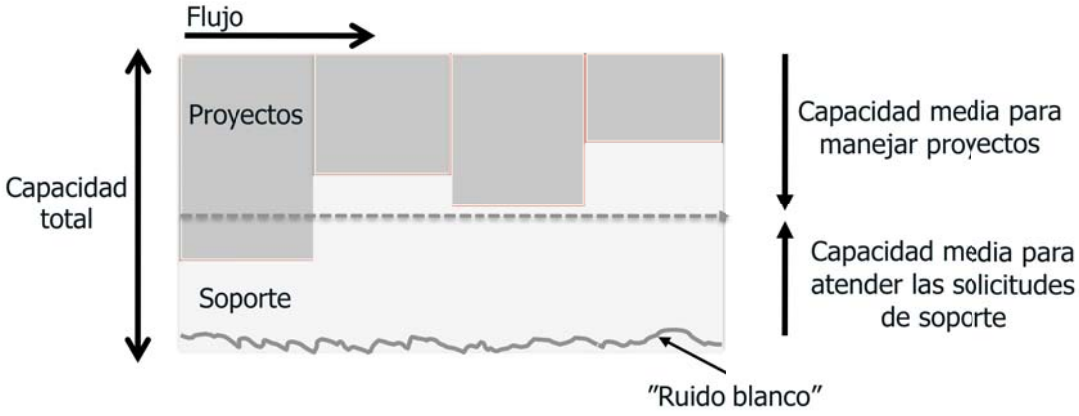


Figura 6. Comenzamos por sumir que la capacidad total era principalmente ocupada en dos tipos de trabajo; "grande" (proyectos) y "pequeños" (soporte). El seguimiento de la velocidad en proyectos nos podía dar pistas acerca de la fecha de entrega si era necesario. Siempre contábamos con que el "ruido blanco" (pequeño soporte menor de 1h, reuniones, café, ayudar a los colegas) estaría por ahí de todos modos.

27

¿Cómo estimar?

Este es un tema siempre recurrente y realmente hay más de una respuesta correcta:

- Estimar regularmente.
- Estimar cuando se necesita.
- Usar estimaciones en “días ideales” o en “puntos de historia”.
- Las estimaciones son inciertas, usa tallas de camiseta (S-pequeña, M-media, L-grande)
- No estimes, o estima sólo cuando exista un coste asociado al retraso que lo justifique.

Ligeramente influidos por Scrum (dado que de ahí es de donde veníamos, de todos modos) decidimos comenzar con puntos de historia. Pero en la práctica, los equipos trataban los puntos de historia como equivalentes a horas-hombre, que les resultaba más natural. Inicialmente, estimábamos todas las historias.

Con el tiempo, los gerentes descubrieron que si mantenían bajo el número de proyectos concurrentes, no tenían a las partes interesadas esperando. También descubrieron que así, en caso de un cambio imprevisto, podían repriorizar las tareas para solucionar el problema.

La necesidad de una fecha de entrega del proyecto había dejado de ser un gran problema, y esto llevó a los gerentes a dejar de solicitar estimaciones a priori. Sólo lo hacían si se temía que tendrían gente esperando.

"Al poco tiempo hubo un gerente que, presionado por una llamada telefónica, prometió la entrega de un proyecto "al final de esta semana". Dado que el proyecto estaba en el tablero Kanban, era fácil estimar el avance (contar historias según se completaban) y concluir que, más o menos el 25% ,se finalizaba cada semana. Así, eran necesarias otras tres semanas adicionales. Enfrentado a este hecho, el gerente cambió la prioridad del proyecto, detuvo el trabajo concurrente, e hizo la entrega posible. Siempre chequea contra el tablero ☺"

¿Qué significa "tamaño estimado"? ¿Plazo o esfuerzo?

Nuestras estimaciones en puntos de historia reflejaban el tiempo de trabajo, es decir, cuántas horas de esfuerzo ininterrumpido esperábamos que una historia necesitara, y no "plazo de entrega" (o días de calendario, o cuántas horas esperar a que el trabajo esté terminado). Midiendo el número de puntos de historia que alcanzan el estado "hecho" cada semana (velocidad), podíamos deducir nuestros tiempos de respuesta medio (el "lead time" medio).

Estimábamos cada nueva historia sólo una vez, y no nos parábamos a revisar las estimaciones de la historia durante su ejecución. Esto nos permitía minimizar el tiempo del equipo que se dedicaba a tareas de estimación.

28

Entonces ¿Cómo trabajábamos realmente?

Kanban impone muy pocas restricciones y te permite trabajar en toda clase de formas. Puedes hacer que el equipo trabaje en actividades planificadas en el tiempo, o puedes elegir realizar las actividades cuando se ha generado la suficiente inercia como para justificarlo.

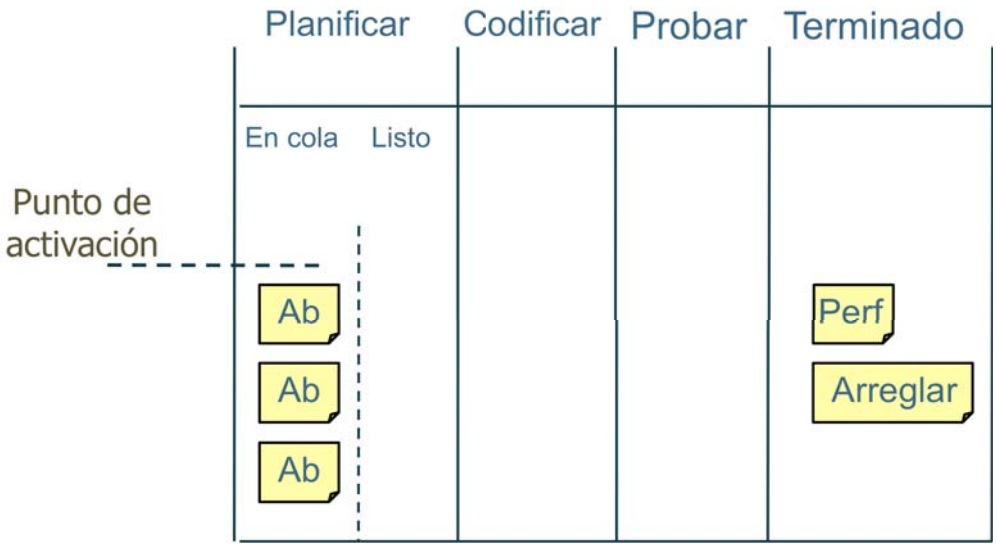
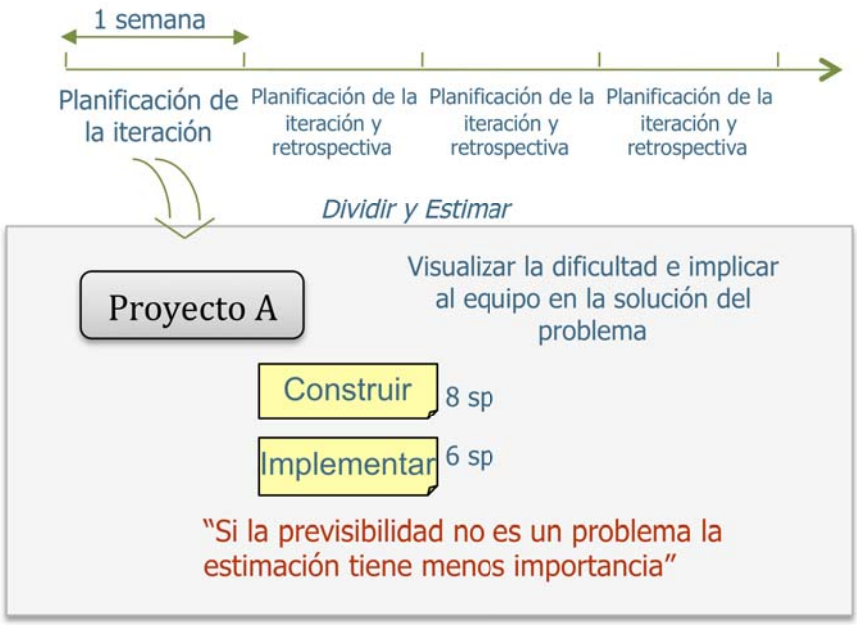


Figura 7 Cuando tres tareas llegaban a la pila de producto se lanzaba el evento de planificación/estimación.

Nosotros decidimos planificar dos eventos recurrentes en el tiempo:

- Reunión diaria – con el equipo en frente del tablero, para sacar a la luz problemas y ayudar a crear una visión compartida de las tareas de todo el equipo.

- Planificación semanal de la iteración – con propósitos de planificación y mejora continua



Este enfoque funcionaba bien en nuestro caso.

Reunión diaria

La reunión diaria era similar a un Scrum diario. Se celebraba cada día, después de la reunión del "Scrum de Scrums" con participación de todos los equipos (desarrollo, pruebas, operación). El Scrum de Scrums generaba información importante para los equipos Kanban, como qué problemas debían tratarse primero, o qué equipo de desarrollo estaba sufriendo más en cada momento. Inicialmente, los gerentes asistían a estas reuniones diarias con frecuencia, proponiendo soluciones y priorizando las decisiones. Pero con el tiempo, según los equipos se hacían mejor auto-organizados, dejaron de asistir tan a menudo (aunque nunca estaban lejos si se les necesitaba).

Planificación de iteración

Una vez a la semana, teníamos una reunión de planificación de la iteración. La hacíamos semanalmente, en un momento planificado, porque descubrimos que si no la planificábamos, ese tiempo se consumía en otras prioridades :), y necesitábamos más charla de equipo.

Un orden del día típico era:

- Actualizar gráficos y tablero. Los proyectos terminados se movían al "Muro de lo hecho".
- Revisar la semana pasada. ¿Qué pasó? ¿Por qué? ¿Qué se podría hacer para mejorarlo?
- Reajuste de los límites de WIP si era necesario.
- División de tareas y estimación de nuevos proyectos (si se veía necesidad).

Básicamente, la reunión de planificación es una combinación entre reunión de estimación y de mejora continua. Los problemas pequeños o medios se resolvían sobre la marcha con el apoyo de una primera línea de gerentes. Pero se hacía complicado mantener la tracción sobre los problemas complejos o de infraestructura. Así que para mejorar esto, dotamos a los equipos de la habilidad de asignar hasta dos "impedimentos de equipo" a sus gerentes.

Las reglas eran:

1. Un gerente puede trabajar en dos huecos en un momento dado del tiempo.
2. Si ambos huecos están ocupados, podéis añadir uno siempre que eliminéis el menos importante de los actuales.
3. El equipo decide cuando se ha resuelto un impedimento.

Esto produjo un cambio muy positivo. De repente, los equipos podían ver cómo los gerentes estaban trabajando para ayudarles incluso en

temas complicados. Podían apuntar a los impedimentos y preguntar "¿cómo va esto?" .Los impedimentos no se podían olvidar, ni ser redefinidos debido a nuevas prioridades estratégicas.

Un ejemplo de impedimento serio fue que el Departamento de Operaciones no estaba obteniendo la ayuda que necesitaba de los desarrolladores cuando sospechaban que había un bug. Necesitaban ayuda para descubrir qué parte del sistema estaba causando el problema, pero como los desarrolladores estaban ocupados en sus sprints desarrollando nuevo material, los problemas se iban acumulando. No era sorprendente que Operaciones sintiera que los desarrolladores no se preocupaban lo suficiente por la calidad.

Cuando este impedimento afloró, se remitió primero al gerente de la línea, y después al gerente del departamento, y este convocó una reunión con el director de desarrollo.

En las conversaciones que siguieron se acordó poner la calidad primero y se estructuró una solución de soporte "round-robin" - cada sprint, un equipo de desarrollo estaría "en línea" para atender instantáneamente las peticiones de ayuda. Tras asegurarse el soporte de sus gerentes, el director de desarrollo pasó una lista de contactos a los equipos de soporte.

La solución se puso a prueba inmediatamente, aunque sospechábamos que el plan no funcionaría a la primera. Pero esta vez los deberes estaban bien hechos; el equipo dio el impedimento por resuelto, y supuso un gran alivio para los equipos de Operaciones.

29

Encontrando una forma de planificar que funcione

Una historia

Recuerdo el momento del cambio para uno de los equipos. Estaba sentado con ellos en su segunda sesión de estimación. El equipo estaba atascado con un proyecto que no sabían cómo estimar. Había demasiadas incógnitas y la sesión de estimación se bloqueó. En lugar de dar un paso al frente y hacerme cargo, les pregunté cómo refinar el proceso para encontrar una solución mejor. Guiados por su gerente, recogieron el guante y se pusieron a diseñar su propia solución. Ese momento fue un punto de inflexión, una victoria importante a partir de la cual se convirtieron en un equipo con confianza. Después de eso comenzaron a evolucionar tan rápidamente que teníamos que apartarnos de su lado.

Dos meses más tarde, su gerente me abordó después de una retrospectiva: "Tengo un problema" dijo señalando el tablero Kanban de su equipo. "No tenemos problemas reales, ¿qué debemos hacer?"

Reinventando la planificación

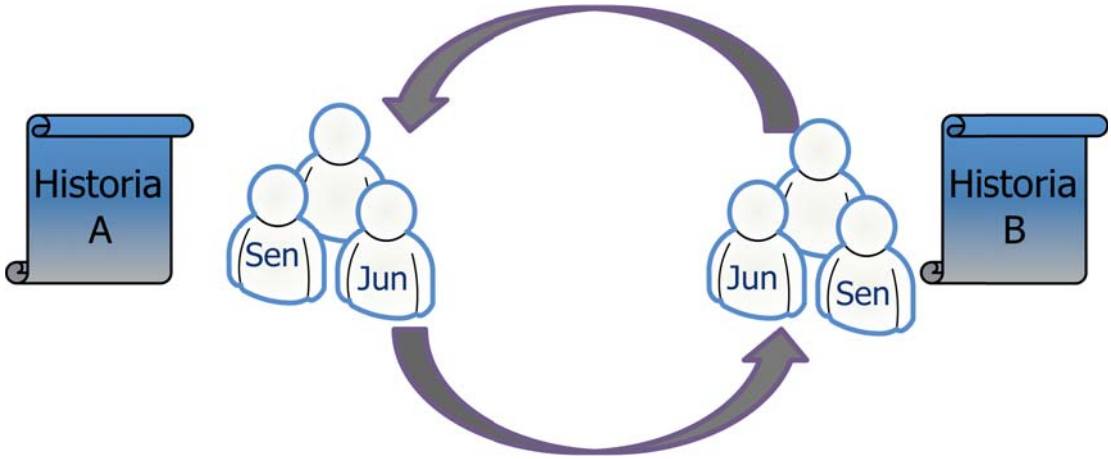
Las sesiones de estimación mediante "planning poker" incluyendo a todos los miembros del equipo no estaban funcionando bien en ninguno de nuestros equipos de operaciones. Algunas razones:

1. El conocimiento estaba desperdigado de manera muy desigual en el equipo.

2. A menudo sólo hablaba una persona.
3. Los miembros del equipo querían resolver los temas urgentes que les quemaban en las manos.

Pero mediante la experimentación, los equipos llegaron de forma independiente a dos procesos de estimación diferente. Cada uno funcionaba bien para su equipo.

Enfoque 1 – intercambio y revisión



- Por cada proyecto/historia, se asigna una pareja de senior + junior para estimarlo (esto es, una persona que conoce esa historia bien, y otra que no la conoce). Esto ayuda a propagar el conocimiento.

- Los restantes miembros del equipo eligen qué historias quieren ayudar a estimar, con el límite de 4 personas por historia para mantener la efectividad de los debates.

- Cada equipo de estimación hace una descomposición de tareas de su historia y, sólo si se requiere, la estima.

- Los equipos intercambian historias y revisan los trabajos de los demás (una persona del equipo original permanece para explicar el trabajo de su equipo a los revisores).

- Hecho!

La sesión de estimación de iteración típica duraba alrededor de 45 minutos, de forma que el nivel de energía se mantenía alto durante la

reunión. Típicamente se hacían un par de ajustes cuando las historias rotaban y eran revisadas por nuevos ojos.

Enfoque 2 – revisión previa por un senior y luego estimar

Dos miembros del equipo con experiencia realizaban una revisión a alto nivel de la historia/proyecto antes de la planificación. Analizaban las posibles soluciones de arquitectura y decidían una para el problema. Una vez establecido, el equipo entra en juego y descompone la historia en tareas usando la solución propuesta como punto de partida.

Figura 8. Descomposición en tareas y revisión por el otro equipo durante la reunión de planificación de la iteración.

30

¿Qué medir?

Hay muchas cosas que pueden medirse – tiempo de ciclo (desde que se descubre una necesidad hasta que se cubre), velocidad, colas, burndowns...

La pregunta importante es ¿qué métricas pueden *usarse* para mejorar el proceso?. Mi consejo es experimentar y ver qué funciona para vuestro caso concreto. Nosotros aprendimos que los diagramas de burndown eran excesivos para cualquier proyecto menor de 4 semanas. El progreso base todavía podía medirse mirando el tablero Kanban (cuántas historias en la pila de producto, cuántas hechas).

Métrica candidata	**Pros**	**Contras**
Tiempo de ciclo	Fácil de medir. No necesita estimar. Comienza y termina en el cliente.	No tiene en cuenta el tamaño.
Velocidad total (sobre todos los tipos de trabajo)	Indicador aproximado pero simple de la dirección de las mejoras y de la variación.	No ayuda a predecir fechas de entrega para tipos de trabajo concretos.
Velocidad por tipo de trabajo.	Más preciso que la Velocidad total.	Para ser útil necesita iniciarse en el cliente y seguirse hasta la entrega final. Es más complicada de medir que la velocidad total.
Longitud de colas	Indicador rápido de la variación de la demanda. Fácil de visualizar.	No distingue si la causa es una demanda inusual o una capacidad inadecuada. Una cola “cero” puede reflejar sobrecapacidad.

Comenzamos por medir “Velocidad por tipo de trabajo” y “longitud de colas”. La velocidad por tipo de trabajo es simple de medir y hace su trabajo. Por su parte las longitudes de cola son buenos indicadores primarios porque pueden vislumbrarse instantáneamente (una vez que sabes donde mirar).

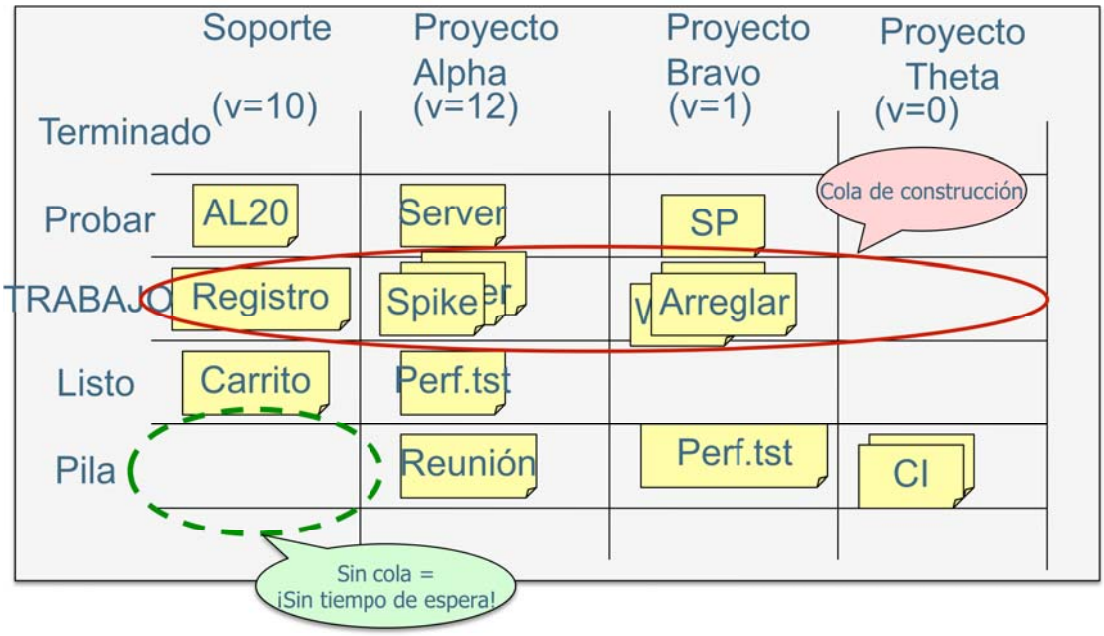
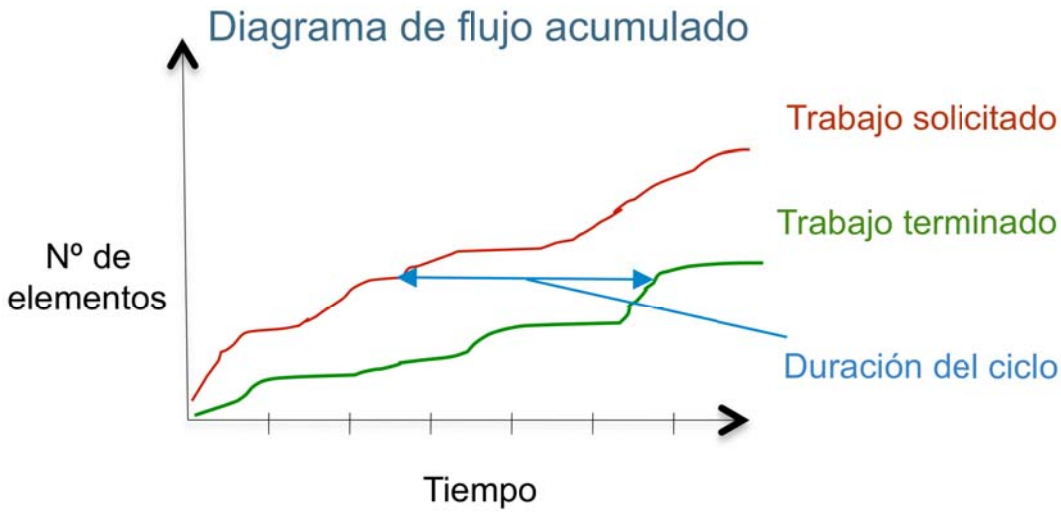


Figura 9. Cuellos de botella y oportunidades. El área roja muestra como las colas se han organizado para provocar un cuello de botella en Pruebas. La ausencia de colas en la columna de soporte indican que no hay tiempo de espera para nuevo trabajo de soporte. Esto es un buena señal para un gran nivel de servicio.

No usábamos diagramas de flujo acumulativos, pero habría sido interesante.



No usábamos diagramas de flujo acumulativos porque el tablero Kanban y el diagrama de velocidad nos daban suficiente información, al menos en nuestro temprano estado de madurez. Los cuellos de botella, las disparidades y el sobretrabajo podían todavía ser fácilmente identificadas, y resolver esos problemas nos mantuvo ocupados durante los primeros seis meses.

31

Cómo empezaron a cambiar las cosas

Tres meses luego de haber introducido Kanban, el equipo de administración de sistemas fue premiado como el "equipo de mejor performance" del departamento de sistemas por la gerencia. Al mismo tiempo, el equipo fue votado como una de las tres "experiencias positivas" en la retrospectiva de la compañía. La retrospectiva de la compañía es un evento de toda la empresa que se realiza cada 6 semanas, y esta era la primera vez que un equipo terminaba en uno de los primeros 3 lugares. Y hace solamente 3 meses, este equipo era un cuello de botella del que casi todos se quejaban.

La calidad del servicio claramente había mejorado. ¿Como sucedió esto? El momento esencial fue cuando todos comenzaron a tirar en la misma dirección. Los gerentes proveyeron un foco claro y protegieron al equipo de trabajo del que no perteneciera allí, y los equipos tomaron la responsabilidad de las fechas de entrega y la calidad. Llevó aproximadamente tres a cuatro meses hasta que esto emergió, pero luego fluyó con facilidad. No es que todos los problemas del mundo hayan desaparecido (eso nos dejaría a todos sin trabajo, ¿no? ☺) – pero nos enfrentamos a nuevos desafíos, como por ejemplo "¿Cómo mantenemos a un equipo motivado por mejorar (cuando ya no son un cuello de botella)?"

Una pieza importante del rompecabezas de la autogestión fue la introducción del concepto de "un contacto de operaciones por equipo". Esto significó darle a cada equipo de desarrollo un contacto de soporte personal dentro del equipo de operaciones. Kanban hizo esto posible al permitir al equipo de operaciones autogestionarse alrededor del trabajo, previniendo sobrecarga y permitiendo mejora continua. Antes, una persona al azar sacaría una tarea de la cola, la solucionaría lo mejor posible según sus habilidades, y luego comenzarían con la siguiente.

Cualquier problema de comunicación significaba empezar todo de nuevo con un pedido de soporte nuevo. Cuando el concepto de uno-a-uno fue implementado, el equipo de soporte adquirió la capacidad de

responder rápidamente cuando información incorrecta o problemas de calidad pusieran en riesgo al sistema.

Anécdota curiosa: con el tiempo, los protocolos de comunicación evolucionaron. Personal de operaciones comenzó a utilizar la mensajería instantánea con desarrolladores que conocían bien, correo electrónico para los que escribían mejor de lo que hablaban, y teléfono si esa era la forma más rápida de solucionar el problema. ☺

Antes

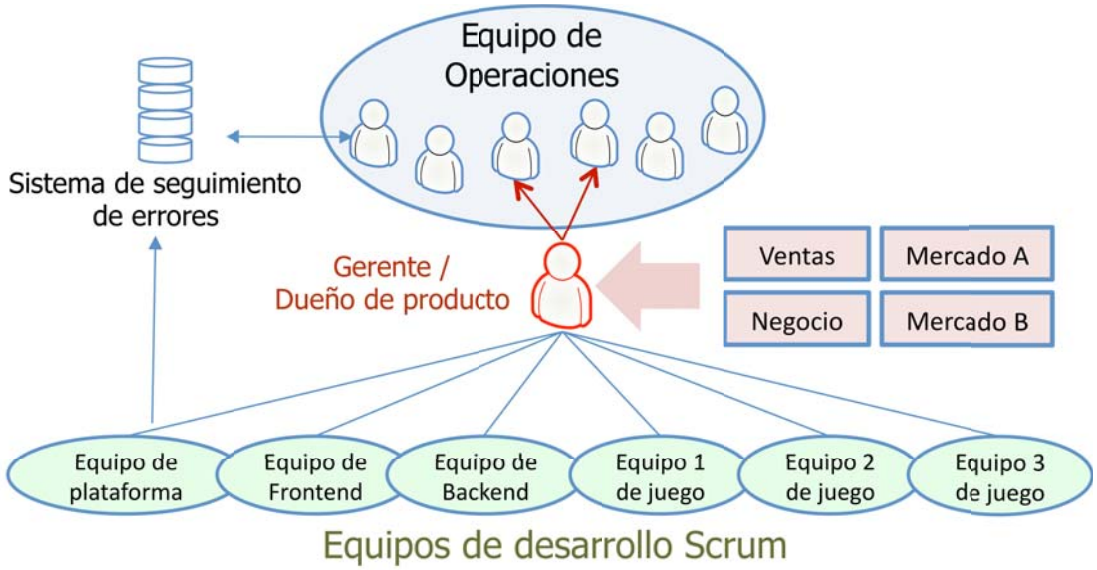


Figura 10. Antes: El gerente de línea es el punto de contacto principal con el equipo. Cualquier cosa importante que haya que hacer debe pasar primero por él. Problemas más pequeños, típicamente los problemas de los desarrolladores, son recibidos a través de un sistema de seguimiento de incidentes. Hay poca interacción directa entre personas.

Después

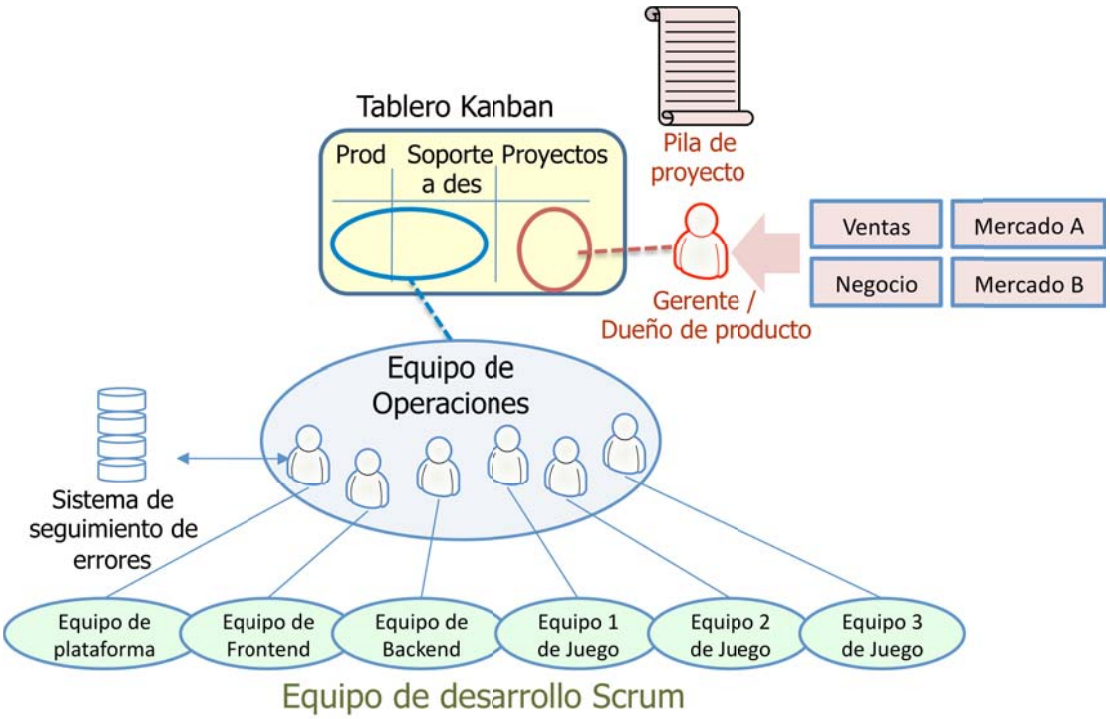


Figura 11. Después: "un contacto de operaciones por equipo" implementado. Los equipos de desarrollo hablan directamente con su contacto en operaciones. Hay muchas interacciones directas entre personas. Los miembros del equipo de operaciones autogestionan su trabajo utilizando el tablero Kanban. El gerente mueve su foco a priorizar proyectos de mayor tamaño y a ayudar cuando surgen problemas complejos.

¿Y que hay del impacto sobre las capacidades del equipo?

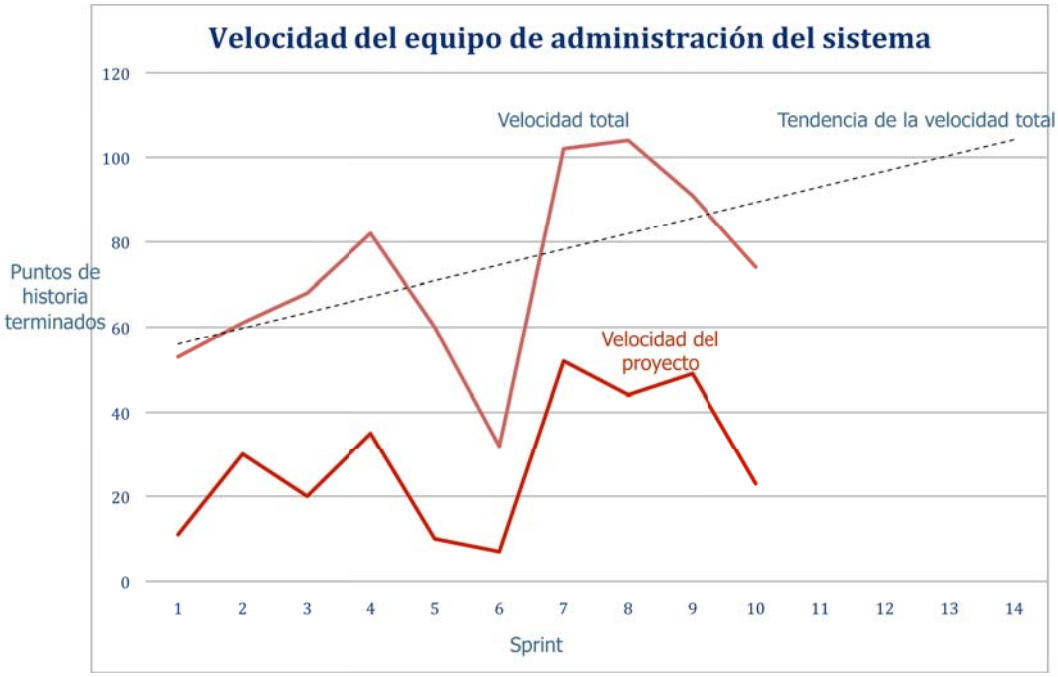


Figura 12. Velocidad total y velocidad de proyecto, medidas en puntos de historia "terminados" por semana. Total es la suma sobre todas las columnas, velocidad de Proyecto representa la parte dedicada a "proyectos" (trabajos de mayor tamaño, por ejemplo una mejora en la plataforma de hardware). Las dos caídas corresponden a 1) una semana en la que casi todos los miembros del equipo estaban de viaje y 2) una entrega grande del equipo de desarrollo.

De esta forma, la velocidad del equipo muestra una tendencia positiva. Al mismo tiempo, el equipo invirtió substancialmente en compartir conocimiento utilizando programación por parejas.

Ya que estamos, echémosle una mirada al rendimiento del equipo de DBAs (administradores de bases de datos):

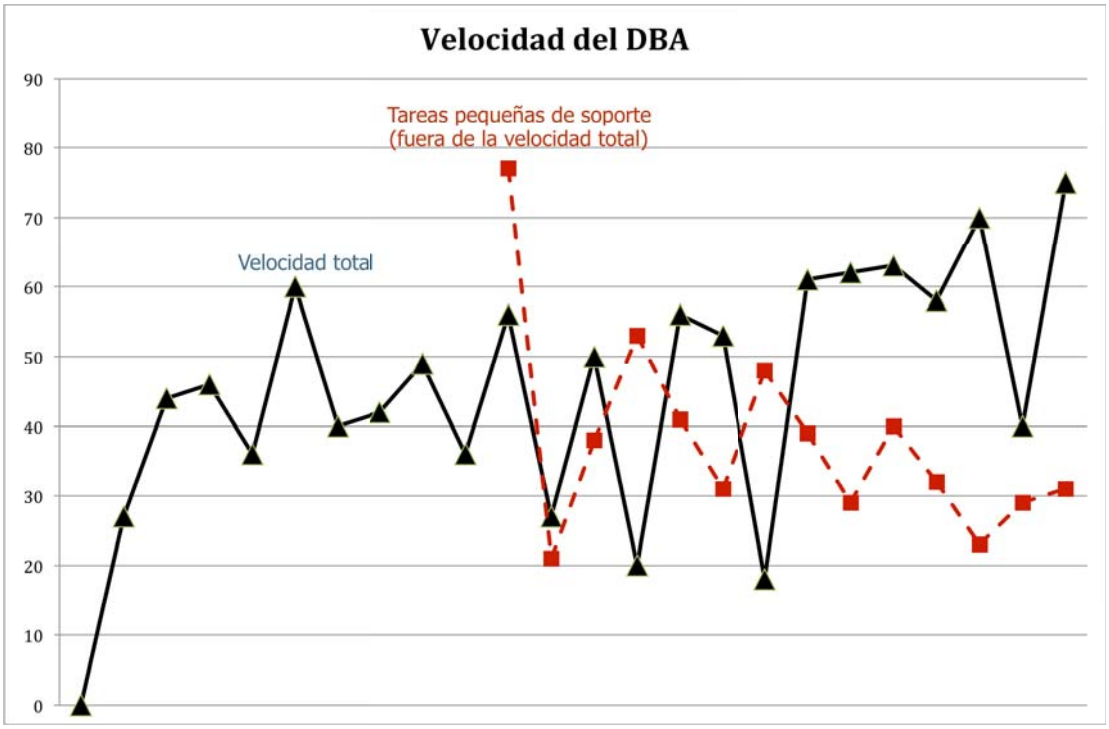


Figura 13: Velocidad total y tareas de soporte pequeñas. La caída en el medio corresponde a la Navidad.

La tendencia de la velocidad total es ascendente, aunque la variación es significativa. El tamaño de la variación inspiró al equipo a monitorizar el numero de pequeñas tareas de soporte (tareas generalmente demasiado pequeñas como para llegar al tablero Kanban). Como pueden ver, el gráfico indica una clara correlación inversa entre la cantidad de tareas pequeñas y la velocidad total.

El equipo de soporte comenzó a hacer Kanban más tarde que los otros dos equipos, así que todavía no tenemos suficiente información confiable.

Maduración

Cuando comenzamos, encontrar áreas problemáticas era tarea sencilla. Pero localizar las oportunidades de mejora más grandes era difícil. El tablero Kanban nos brindó un nuevo nivel de transparencia. No solo era más fácil detectar problemas, sino también surgieron preguntas importantes sobre flujo, varianza y colas. Comenzamos a utilizar colas como una herramienta para detectar problemas. A cuatro meses de comenzar con Kanban, los gerentes ya estaban a la caza de las fuentes de variación que tenían un impacto negativo en sus equipos.

A medida que los equipos evolucionaron de individuos a unidades autogestionadas, los gerentes se encontraron con que se enfrentaban una nueva serie de desafíos respecto a cómo liderar. Necesitaban tratar más con asuntos humanos – ocuparse de quejas, definir objetivos comunes, resolver conflictos, negociar acuerdos. No fue una transición sin dolor – comentaron abiertamente que aprender esto requirió habilidad y energía. Pero aceptaron el desafío y terminaron convirtiéndose en mejores líderes.

32

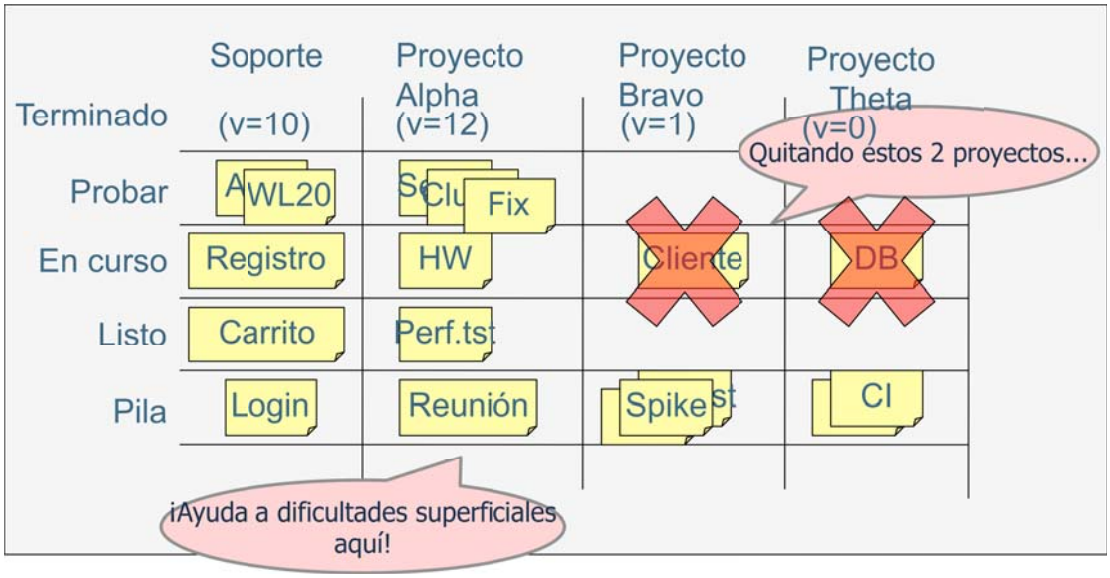
Lecciones aprendidas generales

Conforme diminuye el trabajo en curso, aparecen restricciones

Todos los equipos empezaban con unas limitaciones del Trabajo en Curso (WIP) bastante holgadas. En ese momento la mayoría de la energía se consumía intentando crear el flujo y asegurando que la organización tenía todo el soporte requerido.

Al principio los gerentes querían tener múltiples proyectos ejecutándose simultáneamente, pero después de pocas semanas se evidenciaba que no había capacidad suficiente para tratar con los proyectos de menor prioridad. De un vistazo al panel se veía que nunca se empezaba ningún trabajo que estuviera entre los de baja prioridad. Esto provocó en los gerentes reducir el número de proyectos por equipo.

Con el tiempo, a medida que se estabilizaba el flujo para el trabajo de alta prioridad, empezamos a ajustar los límites del WIP. Esto se hizo reduciendo el número de proyectos (columnas) de tres, a dos, y luego a uno. Mientras ocurría esto, afloraron a la superficie las restricciones del grupo. Los miembros del equipo empezaron a informar de que no estaban recibiendo ayuda a tiempo del resto, así que los gerentes prestaron atención a la gestión de esto.



Otras cuestiones que afloraron fueron como perjudicaban las solicitudes mal realizadas de otros equipos en el rendimiento. Resultaba difícil mantener un flujo suave y rápido cuando las tareas entrantes requerían corrección constante.

Estos problemas no daban la cara hasta que se empezaba. La cuestión era "¿Que problema deberíamos tratar primero?" - y alcanzar un acuerdo común alrededor de él. Con el panel de Kanban todos podían ver cómo un problema específico impactaba en el *flujo*, lo que facilitaba adquirir velocidad para tratar el tema a través de las fronteras organizacionales.

El tablón cambiará con el tiempo, no diseñarlo con indelebles

Todos los paneles Kanban cambiaban a lo largo del camino. Normalmente se pasaba por 2 o tres rediseños hasta que un equipo llegaba a uno con el que trabajaba bien. Así que invertir mucho tiempo en la primera forma probablemente sea una pérdida de tiempo. Asegúrate de que el panel se puede cambiar fácilmente. Nosotros usábamos tiras de cinta negra para la forma. Estas se podían reordenar fácilmente y se podían utilizar tanto en paredes como en pizarras. Otra forma que he visto es dibujar las líneas de rejilla del panel utilizando gruesos marcadores (asegurándonos de que se pueden borrar!).

Abajo se puede ver un ejemplo de optimización. Las prioridades cambiaban frecuentemente al principio, así que hay que evitar tener que mover la columna entera de post-it de notas hacia delante y hacia atrás.

En vez de esto, el equipo pone el número de prioridad encima de cada columna.

Figura 14. Panel Kanban inicial con post-its para las prioridades actuales

No tengas miedo de experimentar y fallar

La lección que saqué de esta aventura es que en realidad no hay punto final. Fallamos en el momento que percibimos que lo hay. Solo existe la experimentación sin final y el aprendizaje. No equivocarse nunca significa no aprender. Nosotros fallábamos muchas veces a lo largo del camino (malos diseños de paneles, estimaciones, gráficos burndown redundantes..) - pero cada vez aprendíamos algo nuevo e importante. Si parábamos de hacer intentos, ¿Cómo podríamos estar aprendiendo entonces?

El éxito de Kanban ha inspirado a equipos de gestión y de desarrollo Scrum a empezar a experimentar con los paneles Kanban también. Puede que este libro ayude!

Puntos finales para el camino

¡Comienza con retrospectivas!

Montones de opciones y cosas sobre las que pensar, ¿verdad?.

Espero que este libro te ayude a aclarar un poco la niebla. Al menos funcionó para nosotros :o)

Si estás interesado en cambiar y mejorar vuestro proceso, déjanos tomar esta decisión por ti ahora. Si no estás haciendo retrospectivas a intervalos regulares, ¡comienza por ahí!

Y asegúrate de que lleven a cambios reales. Consigue un facilitador externo si es necesario.

Una vez que tengas retrospectivas efectivas funcionando, habéis comenzado vuestro viaje hacia evolucionar el proceso adecuado para vuestro contexto, esté este basado en Scrum, XP, Kanban, una combinación de estos, o cualquier otra cosa.

¡Nunca dejes de experimentar!

Kanban o Scrum no son el objetivo. El aprendizaje continuo es el objetivo. Una de las grandes cosas del software es que el ciclo de respuesta es rápido, lo que es clave para el aprendizaje. ¡Así que usa esa respuesta! Cuestiónalo todo, experimenta, falla, y experimenta otra vez.

No te preocupes por hacerlo bien a la primera, porque no lo harás.

Simplemente empieza por algo, y evoluciona desde ahí.

El único fracaso real es no aprender de los fracasos.

Pero, ¡oye!, puedes aprender de esto también.

Buena suerte, y ¡disfruta del viaje!

/Henrik & Mattias, Stockholm 2009-06-24

H: ¿Y esto es todo lo que tenemos?

M: Creo que sí. Dejémoslo aquí.

H: ¿Quizás deberíamos decirles quienes somos?

M: Buen punto. Si hacemos que parezca que somos tíos majos podríamos sacar algo de pasta como consultores.

H: ¡Hagámoslo entonces! Y luego lo damos por cerrado.

M: Sí, tenemos otras cosas que hacer, y también los lectores.

H: Bueno, en realidad ahora empiezan mis vacaciones :o)

M: ¡Eh! ¡No me lo restriegues!

Sobre los autores

Henrik Kniberg y Mattias Skarin son consultores en Crisp en Estocolmo. Disfrutan ayudando a las compañías a triunfar en los lados técnico y humano del desarrollo de software, y han ayudado a docenas de empresas a poner los principios del Lean y Ágiles en práctica.

Henrik Kniberg

Durante la pasada década Henrik ha sido CTO de tres compañías de IT Suecas y ha ayudado a muchas más a mejorar sus procesos. Es "Certified Scrum Trainer" y trabaja regularmente con pioneros de Lean y del Ágilismo como Jeff Sutherland, Mary Poppendieck y David Anderson.

El anterior libro de Henrik, "Scrum y XP desde las Trincheras", tiene más de 150.000 lectores y es uno de los libros más populares sobre el tema. Ha ganado el premio al mejor orador múltiples veces por sus charlas en conferencias internacionales.

Henrik creció en Tokio y ahora vive en Estocolmo con su mujer Sofía y tres niños. Es un músico activo en su tiempo libre. Compone y toca el bajo y los teclados en bandas locales.

henrik.kniberg<at>crisp.se

http://blog.crisp.se/henrikkniberg

http://www.crisp.se/henrik.kniberg

Mattias Skarin

Mattias trabaja como coach de Lean, ayudando a las empresas de software a disfrutar los beneficios de Lean y Agile. Tutela a todas las capas, desde los desarrolladores a la gerencia. Ha ayudado a una compañía de desarrollo de juegos a recortar los tiempos de desarrollo de 24 meses a 4, devuelto la confianza a un departamento de desarrollo completo, y fue uno de los pioneros en el uso de Kanban. Como emprendedor, ha cofundado y gestionado dos empresas.

Mattias tiene un grado en Master de Ciencias y gestión de la Calidad, y ha trabajado como desarrollador durante 10 años en sistemas de misión crítica.

Vive en Estocolmo y disfruta bailando rock & roll, corriendo y esquiando.

mattias.skarin<at>crisp.se

http://blog.crisp.se/mattiasskarin

http://www.crisp.se/mattias.skarin