Eusko Jaurlaritzaren Informatika Elkartea
Sociedad Informática del Gobierno Vasco

Diseño Metodología de Creación, coordinación y dinamización de grupos de trabajo

Año 2012

sa/e software askea libre euskadin

Fecha: 30-06-2012

Referencia:

EJIE S.A.

Mediterráneo, 14

Tel. 945 01 73 00*

Fax. 945 01 73 01

01010 Vitoria-Gasteiz

Posta-kutxatila / Apartado: 809

01080 Vitoria-Gasteiz

www.ejie.es

Trabajo realizado para EJIE SA dentro del proyecto: Servicios de apoyo a la Oficina Técnica de Software Libre. Año 2012.

Eusko Jaurlaritzaren Informatika Elkartea
Sociedad Informática del Gobierno Vasco

Control de documentación

Título de documento: Marco de Trabajo

Histórico de versiones

Código:	
Versión:	
Fecha:	
Resumen de cambios:	

Cambios producidos desde la última versión

Primera versión.

Control de difusión

Responsable:	
Aprobado por:	
Firma:	Fecha: 30/06/2012
Distribución:	

Referencias de archivo

Autores: Alfredo Castañeda y Daniel Armendáriz	
Nombre archivo: E3.8.1_guia_colab	
Localización: Forja de Gobierno Vasco y Web SALE (http://sale.euskadi.net)	

Contenido

	Capítulo/sección	Página

Índice

1 Introducción

Se redacta este documento enmarcado dentro del proyecto de Oficina Técnica de Software Libre, SALE, en el que EJIE está desarrollando el tratamiento de todos los aspectos relacionados con el Software Libre dentro del ámbito del Gobierno Vasco. Se establece la necesidad de adoptar una metodología de trabajo para la creación, coordinación y dinamización de grupos de trabajo. Hay que definir un modelo de trabajo colaborativo que pueda ser adoptado dentro de los procesos de desarrollo de proyectos. Se ha tomado como referencia el libro “Scrum y XP from the trenches”, de Henrik Kniberg. Se pretende paliar los posibles problemas que encontramos en el desarrollo de determinados proyectos que se ven necesitados de metodologías ágiles para su implementación.

Esta guía se enfoca desde un punto de vista procedimental. Trata de proporcionar las herramientas que aporten valor para desarrollar proyectos con garantías de éxito.

2 Puesta en marcha de un proyecto

2.1 Configuración del proyecto

Una vez el proyecto es autorizado, éste se activa: se nombra Responsable (líder), Cliente (interno, lo que Scrum llama Dueño del Producto) y se abre un espacio de trabajo colaborativo para la gestión del proyecto(forja). Se tiene una primera reunión con el cliente para establecer de forma inicial el alcance del proyecto que estamos preparando.

Dependiendo del tipo de proyectos, el responsable debe elegir al equipo de trabajo, en el caso de ser una subcontratación del mismo, el proveedor deberá aportar su propio equipo. En función del tipo de proyecto y del mapa de capacidades de la organización, y, teniendo en cuenta su disponibilidad, el responsable del proyecto elegirá:

- Su Core: las personas necesarias para llevar el peso del proyecto en cuanto a tareas.
- Miembros del equipo: personas que pueden echar una mano en aspectos puntuales. Su aportación no es desdeñable, pero no vital.

La mayoría de los libros sobre Scrum afirman que el tamaño óptimo del equipo oscila entre 5-9 personas. Pero vamos, que si el intervalo es 3-8, mejor que mejor. Merece la pena pasar algunos malos ratos con tal de conseguir equipos de ese tamaño. Establecemos 7 como un máximo ideal a asumir por proyectos.

Si el proyecto tiene un alcance superior al que puede abarcar un equipo de este tamaño deberíamos adoptar otra metodología de trabajo para la gestión del mismo.

2.2 La pila del proyecto

La Pila de proyecto es el corazón de la metodología de trabajo Scrum. La Pila de Proyecto es, básicamente, una lista priorizada de requisitos, historias, funcionalidades, o lo que entendemos que puede llegar a ser necesario para la buena marcha del proyecto. Necesidades del cliente que se describen utilizando su terminología. Llamaremos a esto historias.

Nuestras historias incluyen los siguientes campos/elementos:

- ID – un identificador único, simplemente un número auto-incremental. Esto nos permite no perder la pista a las historias cuando cambiamos su nombre.
- Nombre – una descripción corta de la historia. Por ejemplo, “Ver tu historial de transacciones”. Suficientemente claro como para que el Cliente comprenda aproximadamente de qué estamos hablando, y suficientemente clara como para distinguirla de las otras historias. Normalmente, 2 a 10 palabras.
- Importancia – el ratio de importancia que el Cliente da a esta historia. Por ejemplo, 10. O 150. Más alto = más importante.
- Estimación inicial – la valoración inicial del Equipo acerca de cuanto trabajo es necesario para implementar la historia, comparada con otras historias. La unidad son “puntos de historia” y usualmente corresponde a “días-persona ideales”. Pregunta al Equipo: “si tuvierais el número óptimo de personas para esta historia (ni muchos ni pocos, típicamente 2) y os encerraseis en una habitación con cantidad de comida, y trabajaseis sin distracciones, ¿en cuantos días saldríais con una implementación terminada, demostrable, testeada y a ?”. Si la respuesta es “con 3 personas encerrados en una habitación nos llevaría 4 días”, entonces la estimación inicial son 12 puntos. Lo importante no es que las estimaciones absolutas sean correctas (es decir, que una historia de 2 puntos deba durar 2 días), lo importante es que las estimaciones relativas sean correctas (es decir, que una historia de 2 puntos debería durar la mitad que una historia de 4 puntos)
- Cómo probarlo – una descripción a alto nivel de como se demostrará esta historia en la Demo al final del Sprint. Se trata, esencialmente, de una simple especificación de un test: “Haz esto, entonces haz lo otro, y entonces debería ocurrir aquello”.

- Notas – cualquier otra información, clarificación, referencia a otras fuentes de información, etc. Normalmente muy breve.
- Tabla de ejemplo:

Pila de producto (ejemplo)

ID	Nombre	Imp.	Est.	Como probarlo	Notas
1	Depósito	30	5	Entrar, abrir página de depósito, depositar 10€, ir a página de balance y comprobar que se ha incrementado en 10€.	Necesita un diagrama UML. No preocuparse por encriptación aun
2	Ver tu historial de transacciones	10	8	Entrar, ver transacciones. Realizar un depósito de 10€. Ir a transacciones y comprobar que se ha actualizado con el nuevo depósito	Utilizar paginación para no hacer consultas muy grandes a la BB.DD. Diseño similar a la página de usuario.

Mantenemos esta tabla en un documento compartido y accesible a través del entorno de trabajo colaborativo. Oficialmente, el Cliente es el propietario del documento, pero no queremos dejar al resto de usuarios fuera. Muchas veces una miembro del equipo necesita abrir el documento para clarificar algo o cambiar una estimación.

Por la misma razón, no colocamos este documento en el repositorio de control de versiones; en vez de eso, lo almacenamos en la parte “Documentación” de la Forja del proyecto. Esta debe ser la manera más simple de permitir múltiples editores diferentes sin causar problemas de bloqueo o fusión de documentos. Sin embargo, casi todos los demás artefactos se colocan en el repositorio de control de versiones (svn).

2.3 Campos de Historia adicionales

A veces usamos campos adicionales en la Pila de Producto, fundamentalmente como comodidad para el Cliente a la hora de decidir sus prioridades.

- Categoría – una categorización básica de la historia, por ejemplo “backoffice” o “optimización”. Así, el cliente puede filtrar fácilmente “optimización” y cambiar todas las prioridades de este tipo a “baja”, etc.
- Componentes - usualmente implementado en la forma de “checkboxes” en la hoja de cálculo, por ejemplo “base de datos, servidor, cliente”. Aquí, el Cliente puede identificar qué componentes técnicos estarán involucrados en la implementación de la historia. Esto es útil cuando tienes varios equipos Scrum, por ejemplo un equipo de backoffice y otro equipo de cliente, y quieres que sea fácil para cada equipo saber a qué historias deben dedicarse.
- Solicitante – el Dueño de Producto puede querer mantener un historial acerca de qué cliente o persona interesada pidió originalmente la historia, para poder así ofrecerle información actualizada sobre el progreso de la misma.
- Bug tracking ID – si tienes un sistema de seguimiento de errores, es útil mantener un historial de cualquier correspondencia directa entre una historia y uno o más errores reportados.

2.4 Como mantenemos la Pila de Producto a nivel de negocio

Si el Cliente tiene una formación técnica, puede que añada historias del tipo “añadir índices a la tabla de eventos”. ¿Por qué quiere algo así? El auténtico objetivo subyacente probablemente será algo como “aligerar el formulario de búsqueda de eventos en el backoffice”.

Puede ocurrir que los índices no fueran el cuello de botella que hiciera al formulario ir lento. Puede que fuera algo completamente diferente. El equipo está normalmente mejor capacitado para averiguar como resolver algo, así que el Cliente debería concentrarse en los objetivos de negocio.

Cuando veamos historias con orientación técnica como esta, normalmente es conveniente hacer al Cliente una serie de preguntas tipo “pero ¿Por qué?” hasta que encontremos el objetivo subyacente. Entonces, reformulamos la historia en términos del objetivo subyacente (“aligerar el formulario de búsqueda de eventos del backoffice”). La descripción técnica original acaba siendo una nota (“indexar la tabla de eventos podría resolver esto”).

3 Preparación del Primer Sprint

Lección: Como Responsable del Proyecto, asegúrate de que la Pila de Proyecto está perfectamente lista antes de la reunión de planificación de Sprint.

¿Y esto qué significa? ¿Que todas las historias deban estar perfectamente bien definidas? ¿Qué las estimaciones sean correctas? ¿Qué todas las prioridades hayan sido fijadas? ¡No! Lo que significa realmente es:

- ¡La pila de producto debe existir! 8-0
- Debería haber una Pila de Proyecto y un Cliente.
- Todos los elementos importantes deberían tener ratios de importancia asignados, diferentes ratios de importancia. En realidad, da igual si los elementos menos importantes tienen todos el mismo valor, ya que probablemente no se discutirán durante la planificación de Sprint en cualquier caso. Cualquier historia sobre la que el Cliente piense que tiene una remota posibilidad de incluirse en el Sprint debería tener un nivel de importancia único definido. El ratio de importancia se emplea solo para ordenar los elementos por relevancia. Así que si el elemento A tiene una importancia de 20 y el elemento B una importancia de 100, simplemente significa que B es más importante que A. No significa que B sea cinco veces más importante que A. Si B tuviera una importancia de 21, ¡aun significaría lo mismo! Es útil dejar espacio entre la secuencia de números por si aparece un elemento C que es más importante que A pero menos importante que B. Por supuesto, le podríamos dar un ratio de importancia de 20,5 a C, pero queda mal, así que en vez de eso dejamos espacio entre números.
- El Cliente debe comprender cada historia (normalmente él es el autor, pero en algunos casos otras personas añaden solicitudes, que el Cliente puede priorizar). No necesita saber cómo exactamente debe implementarse, pero debería entender por qué la historia está ahí. A veces, el Cliente no tiene la formación técnica necesaria para detectar todas las historias iniciales, por eso es conveniente que se reúna con el Responsable del Proyecto a la hora de definir la Pila de Proyecto, que actuará en este caso como asesor técnico.

Nota: Otras personas aparte del Cliente pueden añadir sus historias a la Pila de Producto. Pero no pueden asignarles niveles de importancia, ese es un cometido exclusivo del Cliente. Tampoco pueden establecer estimaciones, ése es un cometido exclusivo del Responsable(s) del proyecto.

4 La planificación del Sprint

Cuando tenemos el Cliente, el Responsable de Proyecto, la Pila de Proyecto con historias priorizadas por importancia y el equipo de trabajo, es hora de planificar el Sprint.

La planificación de Sprint es una reunión crítica, probablemente la más importante de Scrum. Una planificación de Sprint mal ejecutada puede arruinar por completo todo el Sprint.

El propósito de la planificación de Sprint es proporcionar al equipo suficiente información como para que puedan trabajar en paz y sin interrupciones durante unas pocas semanas, y para ofrecer al Dueño de Producto suficiente confianza como para permitírselo.

Vale, ha quedado un poco difuso. Una planificación de Sprint produce, concretamente:

- Una meta de Sprint.
- Una lista de miembros (y su nivel de dedicación, si no es del 100%).
- Una Pila de Sprint (lista de historias incluidas en el Sprint).
- Una fecha concreta para la Demo del Sprint.
- Una hora definida para el Scrum Diario.

4.1 Reuniones de planificación de Sprint que duran, y duran…

Lo más difícil de las Planificaciones de Sprint es que:

La gente no piensa que vayan a durar tanto…¡Pero lo hacen!

Todo en Scrum tiene una duración determinada (time-boxed). Intentaremos cumplirla al cien por cien. Así que ¿qué hacemos cuando la reunión de planificación de Sprint está llegando al final y no hay señales de una meta de Sprint o Pila de Sprint? ¿Limitamos la reunión al tiempo establecido? ¿O la extendemos una hora? ¿O terminamos por hoy y continuamos al día siguiente?

Esto nos pasará una y otra vez, especialmente con los equipos nuevos. Si no han producido un Plan de Sprint realista en 2 – 8 horas (o lo que establezca como reunión de planificación), probablemente no se logrará en otra hora más.

Así que acabamos con la reunión en el tiempo establecido. El Sprint sufre, pero la ventaja, en cualquier caso, es que el equipo ha aprendido una lección muy valiosa, y en la próxima reunión de planificación de Sprint serán mucho más eficientes. Adicionalmente, ofrecerán menos resistencia cuando se proponga una duración para las reuniones que anteriormente hubieran considerado excesiva.

Aprender a mantener tus duraciones determinadas, aprender a establecer duraciones realistas es fundamental. Esto se aplica tanto a las reuniones como a los Sprints.

4.2 Agenda de la reunión de planificación de Sprint

Tener algún tipo de agenda u orden del día de la reunión de planificación de Sprint reducirá el riesgo de sobrepasar la duración determinada. Este es un ejemplo típico:

Reunión de planificación de Sprint: 10:00 – 14:00 (10 minutos de descanso cada hora)

- 10:00 – 10:30. El Dueño de Producto comenta la meta del Sprint y resume la Pila de Producto. Se establece un lugar, fecha y hora para la Demo.
- 10:30 – 12:00. El equipo da estimaciones de tiempo, y divide los elementos tanto como sea necesario. El Dueño de Producto actualiza los ratios de importancia. Se clarifican los elementos. Para todos los elementos de alta importancia se establece el "Como probarlo".
- 12:00 – 13:00. El equipo selecciona las historias que se incluirán en el Sprint. Se realizan cálculos de velocidad para chequear si es factible.
- 13:00 – 14:00. Se selecciona una hora para el Scrum Diario (si es que cambia respecto al último Sprint). Se continúa dividiendo las historias en tareas.

La agenda no es en absoluto inamovible. El Responsable del Proyecto puede ampliar o acortar los periodos según sea necesario conforme progresa la reunión.

4.3 *Definiendo la duración del Sprint*

Uno de los resultados de la planificación de Sprint es una fecha de Demo de Sprint definida. Eso significa que se debe determinar una duración del Sprint. ¿Y cuanto debería durar un Sprint?

Bueno, los Sprints cortos están bien. Permiten a la compañía ser "ágil", es decir, cambiar de dirección frecuentemente. Sprints cortos = ciclo de feedback corto = más entregas y más frecuentes = más feedback del cliente = menos tiempo desarrollando en dirección incorrecta = aprender y mejorar más rápido, etc.

Pero los Sprints largos tampoco están mal. El equipo tiene más tiempo para conseguir impulso, tienen más espacio para recuperarse de los problemas que surjan y aun así cumplir la meta del Sprint, tiene menos carga de gestión en términos de reuniones de planificación de Sprints, Demos, etc.

Generalmente, los Dueños de Producto prefieren los Sprints cortos y a los desarrolladores les gustan los Sprints largos. Así que la duración del Sprint es unvalor de compromiso. Se puede experimentar con varias duraciones y al final encontrar la más adecuada para ese equipo en concreto.

Se parte de una duración de 3 semanas. Suficientemente corto para proporcionarnos agilidad corporativa, suficientemente largo para lograr flujo y recuperarse de los problemas que aparezcan durante el Sprint.

Seleccionamos una duración X y realizamos un Sprint o dos, luego ya tendremos datos para saber si está funcionando o se debe modificar. En cualquier caso, una vez establecida una duración, se mantiene durante un buen periodo de tiempo. Si tenemos la misma duración conseguimos un latido corporativo al que todo el mundo se acostumbra confortablemente. No hay discusiones sobre las fechas de entrega ya que todo el mundo sabe que cada X semanas hay una entrega.

4.4 Definiendo la meta del Sprint

Pasa todo el tiempo. En algún momento durante la reunión de planificación de Sprint, preguntas "¿Y cual es la meta del Sprint"?, y todo el mundo te mira anonadado mientras el Cliente arruga la frente y se rasca la barbilla.

Por alguna razón, es difícil conseguir una meta de Sprint. Pero aun así, realmente, merece la pena exprimir una. Mejor una medio-meta que no tener ninguna meta. La meta podría ser "impresionar al Cliente" o "hacer el sistema lo suficientemente bueno como para entregarlo a un grupo de usuarios beta reales" o "añadir soporte de backoffice básico". Lo importante es **que esté descrito en términos de negocio**. Eso significa en términos en los que la gente de fuera del equipo lo pueda entender.

La meta de Sprint debería responder a la pregunta fundamental "**¿Por qué hacemos este Sprint en vez de irnos todos de vacaciones?**". De hecho, una forma de obtener la meta del Cliente es precisamente hacerle esa misma pregunta.

La meta **debería ser algo que no se haya logrado aún**. "Impresionar al Cliente" puede ser una meta aceptable, pero sólo si no está impresionado aún por el producto/desarrollo, etc. tal y como se encuentra ahora mismo. Si fuera así, todo el mundo podría irse a casa y aun así lograr la meta del Sprint.

La meta del Sprint puede parece bastante baladí y artificial durante la planificación de Sprint, pero muchas veces resulta útil a mediados de Sprint, cuando la gente comienza a sentirse confusa acerca de lo que deberían estar haciendo. Si hay bastante equipos Scrum trabajando en diferentes proyectos, es muy útil poder listar todas las metas de Sprint y publicitarla en un sitio público donde todo el mundo en la empresa pueda saber qué se está haciendo.

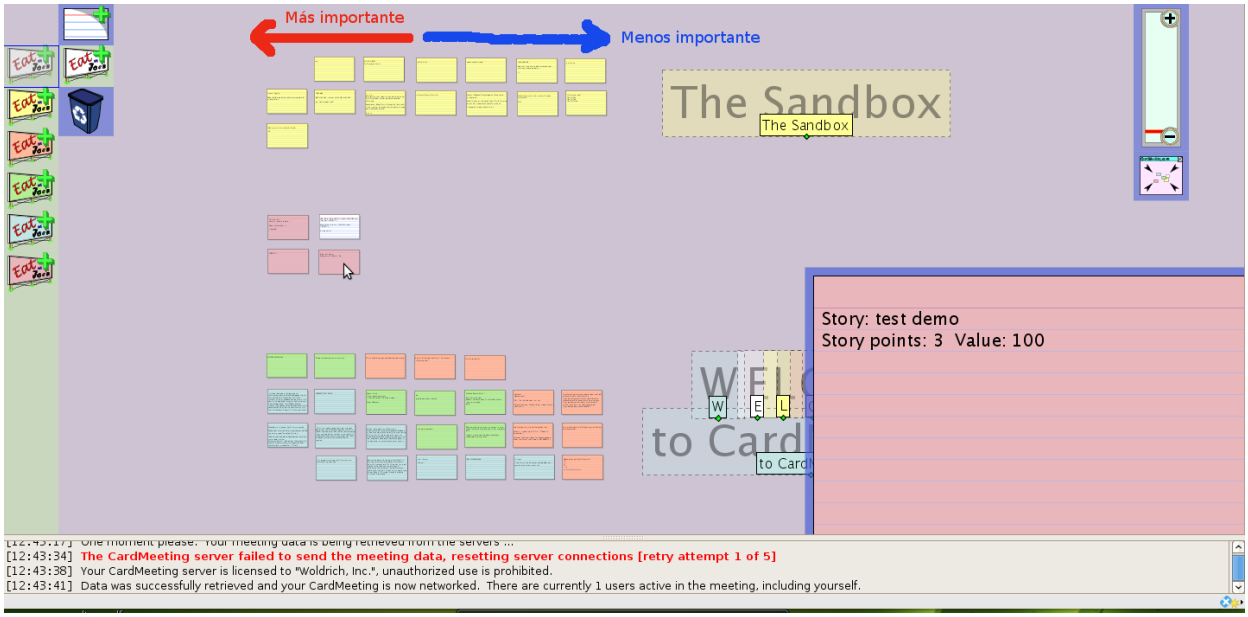
4.5 Clarificando Historias

La mayor parte de la reunión de planificación de Sprint se dedica a las historias de la Pila de Producto. Estimándolas, repriorizándolas, clarificándolas, dividiéndolas, etc. ¿Cómo hacemos esto en la práctica?

Bueno, por defecto, los equipos (distribuidos) tienden a abrir la hoja de cálculo de la Pila y alguien (típicamente el Cliente o el Responsable del Proyecto) toma el teclado, murmura algo sobre cada historia e invita a la discusión. Conforme el equipo y el Cliente discuten las prioridades y los detalles, la persona en el teclado actualiza cada historia directamente en la hoja de cálculo.

¿Suena bien? Bueno, pues no. De hecho es lo peor que se puede hacer. El equipo no se da cuenta de que no funciona hasta que llega el final de la reunión y no han conseguido revisar toda la lista de historias.

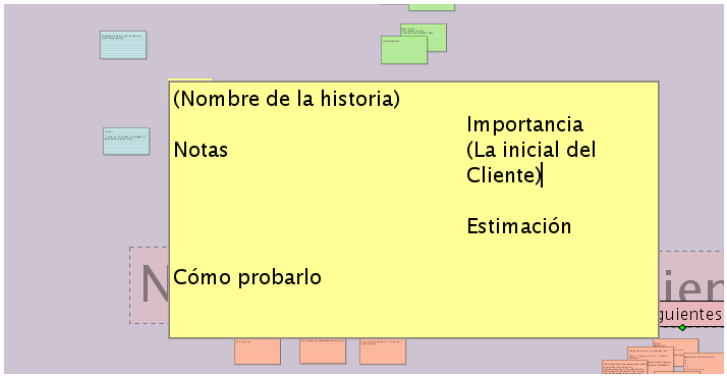
Una solución que funciona mucho mejor es crear tarjetas y ponerlas a disposición de todo el equipo[1].



Este es un interfaz muy superior, comparado con una hoja de cálculo compartida, debido a que:

- Todo el mundo se siente personalmente más involucrado (y no solamente el tipo del teclado).
- Se pueden editar múltiples historias simultáneamente.
- Volver a priorizar es trivial – simplemente se trata de mover las tarjetas.
- Tras la reunión, puede usarse como un tablón de tareas.

1 http://www.cardmeeting.com/



Importante: Tras la reunión de planificación de Sprint, el Responsable de Proyecto actualiza manualmente la Pila de Proyecto en la Hoja de Cálculo respecto a cualquier cambio que se haya realizado sobre las tarjetas de historia físicas. Sí, es una pequeña molestia administrativa, pero lo encontramos perfectamente asumible considerando cuánto más eficiente es la reunión de planificación de Sprint (y los Scrum diarios) utilizando esta herramienta.

Una nota sobre el campo "importancia": se trata de la importancia tal y como figura en la Pila de Proyecto en la hoja de cálculo inicial. Tenerla en las tarjetas hace que sea más sencillo ordenarlas físicamente por importancia (normalmente, ponemos las más importantes a la izquierda y las menos importantes a la derecha). En cualquier caso, una vez que las tarjetas están en el tablón puedes ignorar el ratio de importancia y en lugar de ello usar el orden físico para indicar la importancia relativa. Si el cliente cambia dos historias de sitio no pierdas el tiempo actualizando los ratios de importancia en la tarjeta. Simplemente asegúrate de actualizarlas en la hoja de cálculo después de la reunión.

La descripción de "como probarlo" de cada historia puede (y debe) ser muy breve. De otra forma no terminarás con la planificación de Sprint a tiempo. Es básicamente una descripción básica a alto nivel, en castellano simple, de cómo ejecutar el escenario de pruebas más común manualmente. "Haz esto, entonces haz lo otro y verifica aquello". Estas descripciones simples a menudo descubren malentendidos importantes sobre el alcance de una historia.

4.5.1. Dividir Historias en Historias más pequeñas

Las historias no deberían ser demasiado pequeñas ni demasiado grandes (en términos de estimación). Si hay un montón de historias de 0.5 puntos, probablemente, se centra todo en la microgestión. Por otra parte, una historia de 40 puntos corre un importante riesgo de acabar parcialmente completa, lo que no aporta valor y simplemente incremente la administración. Lo que es más, si tu velocidad estimada es 70 y tus dos historias más prioritarias pesan 40 puntos cada una, la planificación se vuelve difícil.

Hay que elegir entre comprometerse a más de lo que se debe (escoger las dos historias) o a menos (elegir sólo una).

Casi siempre es posible dividir una historia grande en historias más pequeñas. Simplemente hay que asegurarse de que las historias pequeñas siguen representando entregables con valor de negocio.

Normalmente buscaremos historias con estimaciones de 2 a 8 días/persona.

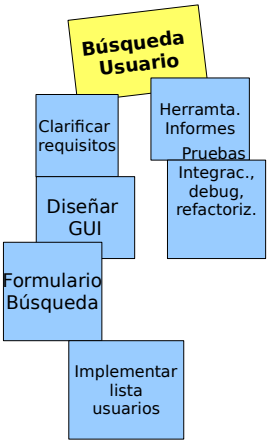
Si la velocidad es usualmente 40-60 para un equipo típico, nos da para unas 10 historias por Sprint. A veces sólo 5, y a veces hasta 15. Es un número de tarjetas gestionable con el que trabajar.

4.5.2. Dividir la Historia en Tareas

¿Cuál es la diferencia entre historias y tareas?

La diferencia es simple: Las historias son entregables de los que el Cliente se preocupa. Las tareas son no entregables, o aspectos de los que el Cliente no se preocupa.

Las estimaciones de tiempo son normalmente más fáciles de hacer (y más exactas) si una historia se subdivide en tareas.



Esto es algo que también se hace de forma sencilla y "agradable" con las tarjetas de historia. Puedes hacer que el equipo se divida en parejas y que cada una subdivida una historia en paralelo. Físicamente, hacemos esto añadiendo pequeñas notas post-it (virtuales) bajo cada historia, de forma que cada post-it representa una tarea dentro de dicha historia.

No actualizamos la Pila de Proyecto en documentos digitales respecto a nuestra división en tareas por dos razones:

- La división en tareas suele ser bastante volátil, es decir, se cambia y se refina con frecuencia durante el Sprint, así que es bastante molesto mantener la Pila de Producto sincronizada.
- De todas formas, el Cliente ("propietario" del documento) no necesita estar involucrado a ese nivel de detalle.

Algunas observaciones interesantes:

- Los equipos que están empezando con Scrum son reticentes a perder tiempo dividiendo un montón de historias en tareas desde el principio.
- Algunos piensan que es una aproximación tipo "cascada".
- Para historias que se entienden bien, es tan fácil hacer esto desde el principio como hacerlo más tarde.
- Este tipo de división frecuentemente revela trabajo adicional que hace que las estimaciones suban, con lo que se consigue un plan de Sprint más realista.

- Este tipo de división desde el principio hace que los Scrum diarios sean notablemente más eficientes.
- Incluso si la división es inexacta y cambia una vez que empezamos, todas las ventajas anteriormente mencionadas siguen siendo válidas.

Tratamos que el tiempo disponible para la planificación de Sprint sea suficientemente largo como para que dé tiempo a hacer esto, pero si el tiempo se acaba lo dejamos como esté, como hemos apuntado anteriormente (y repetiremos posteriormente).

4.5.3. Historias técnicas

He aquí un asunto complejo: historias técnicas. O elementos no-funcionales. Nos referimos a cosas que deben hacerse pero que no son un entregable ni están directamente relacionadas con ninguna historia específica, y no son de valor inmediato para el Cliente. Las llamamos “historias técnicas”.

Por ejemplo:

- Instalar un servidor de compilación continua:
  - Por qué debe hacerse: porque ahorra cantidades inmensas de tiempo a los desarrolladores y reduce el riesgo de problemas explosivos de integración al final de la iteración.
- Escribir una descripción general del diseño:
  - Por qué debe hacerse: porque los desarrolladores olvidan constantemente el diseño general, y entonces escriben código inconsistente. Necesitan una visión global documentada para mantener a todo el mundo en la misma línea de diseño.

¿Son historias en el sentido normal? ¿O son tareas que no están conectadas a ninguna historia específica? ¿Quién las prioriza? ¿Debería involucrarse el Cliente en estos asuntos?

El Cliente no siempre está cualificado para manejar estos compromisos. Así que esto es lo que hacemos:

1. Intentamos evitar las historias técnicas. Hay que buscar formas de transformar las historias técnicas en historias normales con valor de negocio mensurable. Así el Cliente tendrá mejores oportunidades para realizar decisiones correctas entre los pros y los contras.
2. Si no se puede transformar una historia técnica en una historia normal, intentamos hacerla como una tarea dentro de otra historia. Por ejemplo, “refactorizar la capa de acceso a datos” podría ser una tarea dentro de la historia “editar usuario”, ya que esto involucra utilizar la capa de acceso a datos.
3. Si lo anterior falla, definir como historia técnica y mantener una lista separada con dichas historias. Permitimos al Cliente que vea dicha lista, pero no que la modifique. Usamos los factores de dedicación y la velocidad estimada para negociar con el Cliente y sacar algo de tiempo del Sprint para implementar estas historias.

Por supuesto, la otra opción es simplemente mantener al Cliente fuera del ciclo o darle un factor de dedicación no negociable. Pero no hay excusa para no intentar llegar a un consenso primero.

Si el Cliente es un tipo competente y razonable es mejor mantenerle lo más informado posible y permitirle establecer las prioridades generales.

4.5.4. Sistema de seguimiento de errores vs. Pila de Proyecto

La hoja de cálculo es un formato estupendo para la Pila de Proyecto. Aun así tenemos un sistema de seguimiento de errores/gestor de bug. Así que, ¿cómo incluimos los bug en la reunión de Planificación de Sprint? No sirve ignorarlos y concentrarse solo en las historias.

Hay varias estrategias:

1. El cliente, en la reunión de planificación de Sprint, los coloca en el tablón junto al resto de historias (especificando así implícitamente la prioridad de estos elementos comparados con las otras historias).
2. El Cliente crea historias que se refieren a los bug. Por ejemplo, “Arreglar los errores más críticos de informes de backoffice, bug#124, bug#126 y bug#180”.
3. La corrección de bug se considera algo fuera del Sprint. El equipo mantiene un factor de dedicación suficientemente bajo (por ejemplo, el 50%) para asegurar que tienen tiempo suficiente para arreglar errores. Entonces simplemente se asume que el equipo pasará parte de su tiempo arreglando los errores reportados en el Gestor de bug de la Forja.
4. Tratar los errores como cualquier otra historia.

No podemos determinar qué estrategia es mejor. Igual varía de equipo a equipo y de Sprint a Sprint. Tenderemos no obstante a favorecer la primera estrategia. Es sencilla y agradable.

4.5.5. Problemas de "concepto"

¿Cómo te aseguras de que el concepto que tiene el Cliente sobre una historia coincida con el concepto del equipo? ¿O de que cada miembro del equipo tenga el mismo concepto sobre la historia? No se puede. Hay algunas técnicas simples para identificar los malentendidos más flagrantes. La técnica más simple es asegurarse de que todos los campos están rellenos para cada historia (o más específicamente, para cada historia con suficiente importancia como para ser considerada en este Sprint).

4.5.5.1. Ejemplo 1

El equipo y el Cliente están contentos con el plan de Sprint y están listos para terminar con la reunión. El Responsable del Proyecto dice "esperad un segundo, esta historia llamada "añadir usuario", no tiene estimación. Hagamos una estimación." Tras un par de rondas de planificación el equipo acuerda 20 puntos de historia, el Cliente no está de acuerdo. Tras unos minutos de discusión, resulta que el equipo entendió mal el alcance de "añadir usuario" y pensaban que significaba "un bonito interfaz Web para añadir, eliminar, buscar usuarios", cuando el Cliente quería decir "añadir usuarios a mano haciendo sentencias MySQL contra la base de datos". Estiman de nuevo y acaban con 5 puntos de historia.

4.5.5.2. Ejemplo 2

El equipo y el Cliente está contentos con el plan de Sprint y están listos para acabar la reunión. El Responsable del Proyecto dice "esperad un segundo, esta historia que se llama "añadir usuario", ¿cómo debería demostrarse?". Alguien dice "bueno, primero nos autenticamos en la Web y luego…" y el Cliente interrumpe "¿autenticarse en la Web? No, esta funcionalidad no debería de ninguna forma ser parte del sitio Web, debería ser sólo un script muy sencillo para los administradores técnicos".

4.6 Estimación de Tiempos de Historia

La estimación es una labor de equipo: todos los miembros del equipo deben involucrarse en estimar cada historia. ¿Por qué?

- A la hora de planificar, normalmente no sabemos exactamente quién implementará qué partes de cada historia.
- Las historias normalmente involucran a bastantes personas y de diferentes áreas de experiencia (diseño de interfaz de usuario, programación, pruebas, etc.).
- Para poder proporcionar una estimación, un miembro del equipo necesita comprender de alguna forma de qué trata la historia. Pidiendo a todo el mundo que estime la historia nos aseguramos de que cada miembro del equipo comprende de qué trata cada elemento. Esto incrementa las posibilidades de que unos miembros del equipo ayuden a otros durante el Sprint. También mejora las posibilidades de que aparezcan pronto las preguntas importantes sobre cada historia.
- Cuando pedimos a todo el mundo que de estimaciones muchas veces encontramos discrepancias en las que dos miembros del equipo tienen estimaciones tremendamente distintas sobre la misma historia. Es mejor descubrir esas cosas al principio.

En Scrum, normalmente, las tareas se estiman en horas, no en días. Nuestra fórmula general podría ser:

1 día-persona real = 8 horas-persona reales

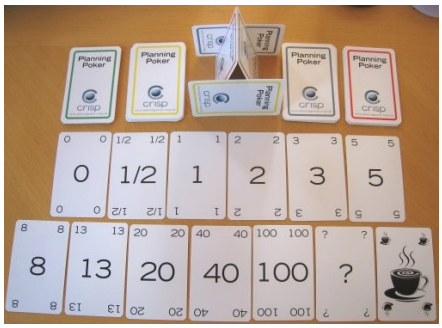
Ahora bien, no recomendamos hacer eso, por las siguientes razones:

- Las estimaciones en horas-persona son demasiado granulares. Esto provoca una tendencia a estimar muchas tareas en 1-2 horas, y con ello a la microgestión.

- De todas formas resultará que todo el mundo estaba pensando en términos de días-persona, y simplemente multiplican por 8 para escribir las horas-persona. “Esta tarea debería llevar más o menos un día. Oh, tengo que escribirlo en horas, pues entonces escribo ocho horas”.
- Dos unidades diferentes causan confusión. “¿Esa es la estimación en días-persona o en horas-persona?”.

Así que ahora usaremos días-persona como base para todas nuestras estimaciones (aunque lo seguiremos llamando puntos de historia). Nuestro valor más bajo es 0.5, es decir, cualquier tarea que es menor de 0.5 se elimina o se combina con otras tareas, o se le deja una estimación de 0.5.

Si el equipo proporciona una estimación, normalmente la persona que entiende mejor la historia será el primero en soltar una. Desafortunadamente, esto afectará severamente a las estimaciones de los demás. Hay una técnica excelente para evitar esto: se llama **planning poker**[2]



Cada miembro del equipo cuenta con una baraja de 13 cartas, como las que se muestran en la imagen. Cada vez que hay que estimar una historia, cada miembro del equipo selecciona una carta que representa su estimación de tiempo (en puntos de historia) y la coloca bocabajo en la mesa.

Cuando todos los miembros del equipo han preparado sus cartas, se les da la vuelta al mismo tiempo. Así obligamos a cada miembro del equipo a pensar por si mismo en lugar de seguir la estimación de otro.

Si hay mucha discrepancia entre dos estimaciones, el equipo discute las diferencias y trata de construir una imagen común del trabajo necesario para la historia. Pueden hacer algún tipo de división en tareas. Después, el equipo estima de nuevo. Este bucle se repite hasta que la estimación de tiempo converge, es decir, que todas las estimaciones sean aproximadamente las mismas para esa historia.

Es importante que los miembros del equipo recuerden que deben estimar el total de tiempo necesario para la historia. No solamente “su” parte del trabajo. El encargado de pruebas no debería estimar sólo la cantidad de trabajo de pruebas.

La secuencia de números no es lineal. Por ejemplo, no hay nada entre 40 y 100. ¿Por qué? Esto es para evitar una falsa sensación de exactitud para las estimaciones de tiempo más grandes. Si una historia se estima aproximadamente en 20 puntos, no es relevante discutir si deberían ser 20, 18 o 21. Todo lo que sabemos es que es una historia grande y es difícil de estimar. Así que 20 es nuestra idea aproximada.

¿Estimaciones más detalladas? Hay que dividir la historia en historias más pequeñas y tratar de estimar las historias pequeñas. No se puede falsear el resultado combinando un 5 y un 2 para hacer un 7. Hay que escoger entre 5 y 8, no hay 7.

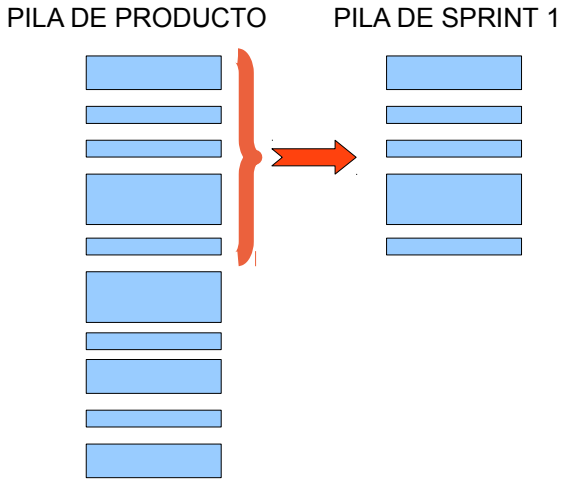
Algunas cartas especiales sobre las que hablar:

- 0 = “esta historia ya está hecha” o “esta historia es prácticamente nada, apenas unos minutos de trabajo”.
- ¿ = “no tengo ni la más remota de las ideas. Nada”.
- Taza de café = “estoy demasiado cansado para pensar. Tomemos un descanso”

4.7 Decidiendo qué historias incluir en el Sprint

Una de las principales actividades durante la planificación de Sprint es decidir qué historias se incluyen en el Sprint. Más específicamente, **qué historias de la Pila de Producto copiar en la Pila de Sprint**.

2 Herramienta online: http://www.planningpoker.com/



Cada rectángulo representa una historia, ordenadas por importancia. La historia más importante está al principio de la lista. El tamaño de cada rectángulo representa el tamaño de la historia (es decir, el tiempo estimado en puntos de historia). La altura del corchete naranja representa la velocidad estimada del equipo, es decir, cuántos puntos de historia cree el equipo que puede completar durante el próximo Sprint.

La Pila de Sprint de la derecha es una instantánea de las historias la Pila de Proyecto. **Representa las historias a las que el equipo se compromete durante este Sprint. El equipo decide cuántas historias incluirá en el Sprint. No el Dueño de Producto ni nadie más.**

Esto plantea dos cuestiones:

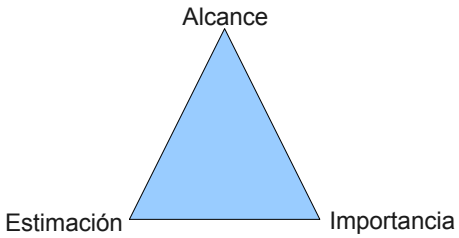
1. ¿Cómo decide el equipo qué historias incluir en el Sprint?
2. ¿Cómo puede el Dueño de Producto alterar la decisión del equipo?

Empezaremos con la segunda cuestión.

4.7.1. ¿Cómo puede el Cliente alterar las historias que se incluyen en el Sprint?

4.7.1.1. Opción 1: Comprometiendo la calidad

Cada historia contiene tres variables que son muy dependientes unas de otras.



El alcance y la importancia los fija el Cliente. La estimación la proporciona el equipo. Durante una planificación de Sprint, estas variables sufren un ajuste fino y continuo a través del diálogo cara a cara entre el equipo y el Cliente.

Normalmente, el Cliente comienza la reunión resumiendo cuál es su meta para el Sprint y las historias más importantes. A continuación, el equipo las repasa y les asigna una estimación (puntos de historia), comenzando con la más importante.

Conforme van haciéndolo, aparecerán dudas importantes respecto al alcance: “esta historia sobre ‘borrar usuario’, ¿incluye repasar todas las transacciones pendientes del usuario y cancelarlas?”. En algunos casos, las respuestas sorprenderán al equipo y les obligarán a cambiar sus estimaciones. En otros casos, la estimación para una historia no será la que el Cliente esperaba. Esto puede forzarle a cambiar la importancia de la historia o su alcance, lo que obligará al equipo a volver a estimarla, etc.

En el triángulo mencionado anteriormente hemos dejado intencionadamente fuera una cuarta variable: la calidad. Tiendo a distinguir entre calidad interna y calidad externa.

- Calidad externa: es lo que perciben los usuarios del sistema. Un interfaz de usuario lento y poco intuitivo es un ejemplo de baja calidad externa.
- Calidad interna: se refiere a aquellos aspectos que normalmente no son visibles al usuario, pero que tienen un profundo efecto en la mantenibilidad del sistema. Cosas como consistencia del diseño del sistema, cobertura de pruebas, legibilidad del código, refactorización, etc.

Generalizando, un sistema con alta calidad interna puede, aun así, tener una baja calidad externa. Pero un sistema con baja calidad interna rara vez tendrá buena calidad externa. Es difícil construir algo sobre unos cimientos pobres.

Trataremos la calidad externa como parte del alcance. En algunos casos puede tener sentido, desde el punto de vista de negocio, liberar una versión del producto que tenga un interfaz de usuario torpe y lento, y más tarde liberar una versión mejorada. Dejaremos esa decisión al Cliente, ya que él es el responsable de definir el alcance.

Sin embargo, **la calidad interna es algo que no puede ser discutido**. Es responsabilidad del equipo mantener la calidad del sistema bajo toda circunstancia y simplemente no es negociable.

¿Y como distinguimos la diferencia entre aspectos de calidad externa y aspectos de calidad interna? Supongamos que el Cliente dice “Respeto vuestras estimaciones de esfuerzo de 6 puntos de historia, pero estoy seguro de que podríamos hacerlo en la mitad de tiempo si lo pensamos aunque no sea lo más adecuado”.

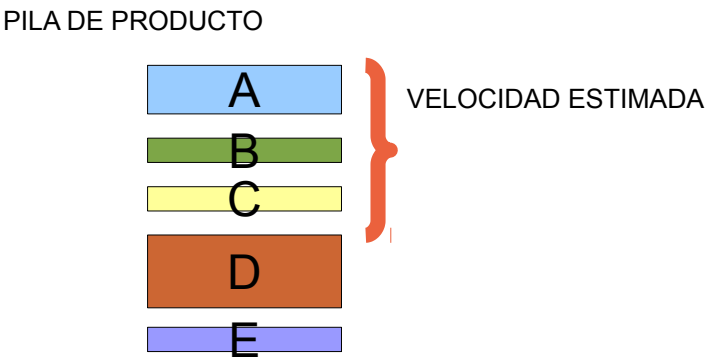
Está intentando usar la calidad interna como una variable. ¿Cómo lo sabemos? Porque quiere que reduzcamos la estimación de la historia sin “pagar el precio” de reducir el alcance. La frase “no más adecuado” debería disparar una alarma…

¿Y por qué no permitimos esto? La experiencia es que sacrificar la calidad interna es, prácticamente siempre, una idea terrible. El tiempo que se ahorra es mucho menor que el coste, tanto a corto como a largo plazo. Una vez que permites que una base de código comience a deteriorarse es muy duro volver a conseguir su calidad original más adelante.

En lugar de eso intentaremos reconducir la discusión hacia el alcance. “Ya que es importante para ti que entreguemos esta historia pronto, ¿podemos reducir su alcance de forma que sea más rápida de implementar? Quizás podríamos simplificar el manejo de errores y convertir “Manejo Avanzado de Errores” en una historia separada que reservemos para el futuro. O podríamos reducir la prioridad de otras historias de forma que podamos centrarnos en esta”. Una vez que el Cliente aprende que la calidad interna no es negociable, normalmente se hace muy bueno en manipular las otras variables, como veremos a continuación.

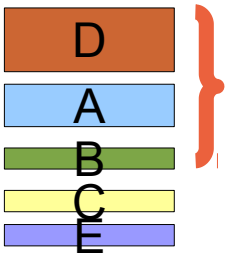
4.7.1.2. Opción 2: Repriorizando, modificando el alcance, dividiendo historias.

Supongamos que tenemos la siguiente situación durante una reunión de planificación de Sprint.

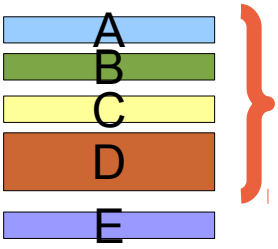


Al Dueño de Producto no le gusta que la historia D no se vaya a incluir en el Sprint. ¿Cuáles son sus opciones durante la reunión de planificación de Sprint?

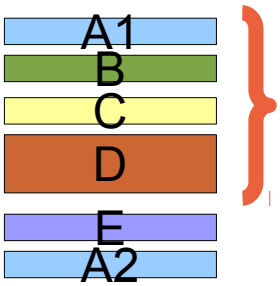
Volviendo a priorizar: Si le da al elemento D la mayor importancia, el equipo se verá obligado a añadirlo al Sprint en primer lugar (descartando en ese caso la historia C).



Cambiando el alcance: La segunda opción es cambiar el alcance – reducir el alcance de la historia A hasta que el equipo crea que la historia D podría caber en el Sprint.



Dividiendo historias: La tercera sería dividir una historia. El Dueño de Producto podría decidir que hay algunos aspectos de la historia A que no son tan importantes, así que dividiría la historia A en dos historia A1 y A2 con diferentes niveles de importancia.



Como veis, aunque el Dueño de Producto no puede controlar normalmente la velocidad estimada, hay varias formas en las que puede influenciar qué historias entran en cada Sprint.

4.7.2. Cómo decide el equipo qué historias incluir en el Sprint?

Hay 3 técnicas para esto:

a) La mala: que pasen todas y a ver hasta dónde llegamos en 3 semanas. Lo más probable es que a ningún sitio. Eso sí, estresados y frustrados. No se contempla.

b) La regular: **Por aproximación basándonos en la experiencia del equipo**.

-

Esto funciona bastante bien para equipos pequeños y Sprints cortos.

c) La mejor: **usando Cálculos de Velocidad**

Esta técnica consta de dos pasos

1. Decidir la velocidad estimada
2. Calcular cuántas historias se pueden añadir sin sobrepasar la velocidad estimada.

Es bastante simple: hasta ahora tenemos una duración del sprint y una serie de historias ordenadas por importancia y estimadas en "puntos de historia". Calcularemos nuestra velocidad estimada para ver a cuántas historias nos comprometemos en este sprint. Para ello, podemos utilizar dos (o tres, como veremos más adelante) técnicas:

c.1) El tiempo que hizo ayer. Una manera muy fácil de estimar la velocidad es revisar la historia del equipo.¿Cuál fue su velocidad durante los últimos Sprints? Y entonces asumir que la velocidad será más o menos la misma en el próximo Sprint.

Sólo es factible para equipos que ya han hecho algunos Sprints (de forma que haya estadísticas disponibles) y que harán el próximo Sprint más o menos de la misma manera, con el mismo tamaño de equipo, las mismas condiciones de trabajo, etc.

c.2) Cálculo de recursos. Digamos que estamos planificando un Sprint de 3 semanas (15 días laborables) con un equipo de 4 personas. Laura estará de vacaciones 2 días. David sólo estará disponible al 50% y estará un día de vacaciones. Poniéndolo todo junto…

	Días Disponibles
Laura	**15**
Angela	**13**
Cesar	**15**
David	**7**
	50 DIAS-HOMBRE DISPONIBLES

…Tenemos 50 días-hombre disponibles en este Sprint.

¿Es esta nuestra velocidad estimada? ¡No! Porque nuestra unidad de estimación son puntos de historia lo que, en nuestro caso, corresponde más o menos a "días-hombre ideales". Un día-hombre ideal es un día perfectamente efectivo, sin distracciones, lo cuál es raro. Lo que es más, debemos tener en cuenta cosas como trabajo no planificado que se añade al Sprint, gente que se pone enferma, etc.

Así que nuestra velocidad estimada será sin duda menor de 50. ¿Pero cuanto menor? Para esto usamos el "**factor de dedicación**". El factor de dedicación es una estimación de cómo de centrado va a estar el equipo. Un factor de dedicación bajo puede significar que el equipo espera encontrar muchas distracciones e impedimentos o que considera que sus propias estimaciones son optimistas.

La mejor manera de determinar un factor de dedicación razonable es **estudiar el último Sprint (o incluso mejor, la media de los últimos Sprints)**. Así,

FACTOR DE DEDICACIÓN DEL ÚLTIMO SPRINT

(VELOCIDAD REAL)

(FACTOR DE DEDICACIÓN) = ----------

(DIAS-HOMBRE DISPONIBLES)

La velocidad real es la suma de las estimaciones iniciales que se completaron en el último Sprint. Digamos que en el último Sprint se completaron 18 puntos de historia utilizando un equipo de 3 personas formado por Toni, Laura y César trabajando las tres semanas hasta un total de 45 días-hombre. Y ahora estamos intentando calcular la velocidad del próximo Sprint. Para complicar las cosas, un nuevo tipo, David, se une al equipo para este Sprint. Teniendo en cuenta las vacaciones y demás asuntos tenemos 50 días-hombre ideales este Sprint.

FACTOR DE DEDICACIÓN DEL ÚLTIMO SPRINT

18 PUNTOS HISTORIA

---------- = 40%

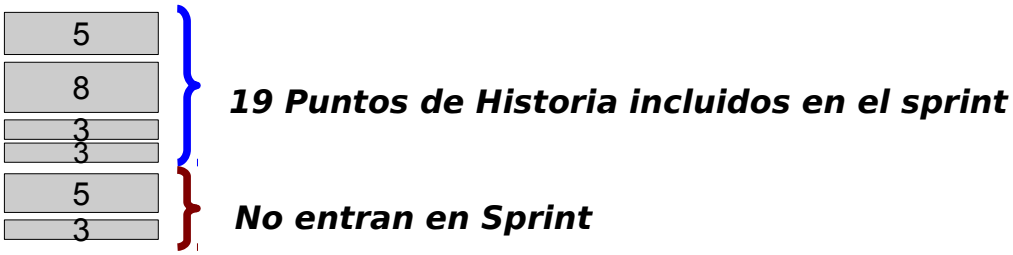
45 DÍAS-HOMBRE

VELOCIDAD ESTIMADA DE ESTE SPRINT

DÍAS-HOMBRE DISPONIBLES X FACTOR DE DEDICACIÓN = VELOCIDAD ESTIMADA

50 DÍAS-HOMBRE X 40% = 20 PUNTOS HISTORIA

Así que nuestra velocidad estimada para el próximo Sprint es de 20 puntos de historia. Eso significa que el equipo debe añadir historias al Sprint hasta que sume aproximadamente 20.



En este caso, el equipo puede escoger las 4 historias más importantes hasta un total de 19 puntos de historia, o las 5 historias más importantes hasta 24 puntos de historia. Digamos que escogen 4 historias, ya que se aproximan más a los 20 puntos de historia. **Siempre que haya dudas, escoge añadir menos historias.**

Dado que las 4 historias suman 19 puntos, la velocidad estimada para este Sprint es de 19.

"El tiempo que hizo ayer" es una técnica sencilla, pero deberemos usarla con cierta dosis de sentido común. Si el último Sprint fue especialmente malo porque la mayoría del equipo estuvo enfermo una semana, entonces podría ser adecuado asumir que no volverás a tener tan mala suerte y podrías estimar un factor de dedicación mayor el próximo Sprint. Si el equipo de sistemas ha instalado recientemente un sistema super-rápido de compilación continua probablemente también podrás incrementar el factor de dedicación gracias a ello. Si una nueva persona se une a este Sprint deberías reducir su factor de dedicación para tener en cuenta su formación, etc.

Siempre que sea posible, ten en cuenta varios Sprints y saca medias para conseguir estimaciones más acertadas.

¿Qué ocurre si el equipo es completamente nuevo y no tienes ninguna estadística? Mira al factor de dedicación de otros equipos en circunstancias similares. ¿Qué pasa si no tienes otros equipos en los que fijarte? Adivina un factor de dedicación. Las buenas noticias son que sólo deberás adivinar para el primer Sprint. Después, tendrás estadísticas que podrás ir midiendo continuamente para aproximar mejor el factor de dedicación y la velocidad estimada.

Antes hemos mencionado la posibilidad de una tercera técnica, que es la que frecuentemente usaremos: **combinar (ojo, hasta cierto punto) por aproximación, cálculo de velocidad basado en el tiempo que hizo ayer y cálculo de velocidad basado en días-hombre disponibles y factor de dedicación**. Realmente no conlleva mucho esfuerzo.

Miramos el factor de dedicación y la velocidad real del último Sprint. Miramos al total de recursos disponibles para este Sprint y estimamos el factor de dedicación. Discutimos cualquier diferencia entre estos dos factores de dedicación y hacemos los ajustes que sean necesarios.

Una vez que tenemos la lista preliminar de historias a incluir en el Sprint hacemos un chequeo a "ojo de buen cubero". Le pedimos al equipo que ignore los números por un momento y que piensen sobre si les parece realista como para hacerlo en un Sprint.

Si parece demasiado, quitamos una historia o dos. Y viceversa. Al final del día, el objetivo es simplemente decidir qué historias se incluyen en el Sprint. Factores de dedicación, disponibilidad de recursos y velocidad estimada son sólo medios para conseguir dicho objetivo.

4.8 Definir sitio y hora para el Scrum periódico

Una de las cuestiones frecuentemente olvidadas de la planificación de Sprint es fijar "un sitio y una hora determinados para el Scrum diario o semanal". Sin ello, el Sprint está condenado a un mal comienzo. El primer Scrum diario o semanal es esencialmente el lanzamiento, donde todo el mundo decide por dónde va a empezar a trabajar.

Normalmente fijaremos las reuniones a primera hora de la mañana.

- Desventaja de Scrums por las tardes: cuando llegas al trabajo por la mañana, tienes que acordarte de qué le dijiste a la gente sobre lo que deberían hacer hoy.
- Desventaja de los Scrums por las mañanas: cuando llegas al trabajo por la mañana, debes acordarte de qué hiciste ayer para informar sobre ello hoy.

Lo más importante es que sea a una hora a la que todo el equipo esté disponible y acepte con total convencimiento.

4.9 Reuniones de Planificación que duran y duran...(II): Dónde trazar la línea

De todos los asuntos que queremos resolver durante la planificación de Sprint, ¿qué abandonamos si nos quedamos sin tiempo?

Lista de prioridades:

- **Prioridad 1:** Una meta de Sprint y una fecha para la demo. Esto es lo mínimo que necesitas para comenzar un Sprint. El equipo tiene una meta y una fecha de finalización, y pueden trabajar directamente con la Pila de Proyecto. Apesta, sí, y deberías considerar seriamente organizar una nueva reunión de planificación de Sprint mañana mismo, pero si realmente necesitas que el Sprint comience entonces probablemente puedas hacerlo con esto. Para ser honestos, no te lo recomendamos.
- **Prioridad 2:** Lista de qué historias ha aceptado terminar el equipo en este Sprint.
- **Prioridad 3:** Una estimación para cada historia del Sprint.
- **Prioridad 4:** "Como probarlo", relleno para cada historia del Sprint.
- **Prioridad 5:** Cálculos de velocidad y recursos, como chequeo de la planificación del Sprint. Incluyendo una lista de los miembros del equipo y sus compromisos (de otra forma, no podrías calcular la velocidad).
- **Prioridad 6:** Un sitio y hora específicos para la realización del Scrum diario. Sólo necesitas un momento para decidirlo, pero si te quedas sin tiempo el Responsable de Proyecto puede simplemente decidir esto después de la reunión y mandar un correo a todo el mundo.
- **Prioridad 7:** Historias divididas en tareas. Esta división puede hacerse diariamente durante los Scrum diarios, pero interferirá levemente el flujo del Sprint.

4.10 ¡Por fín acabó la reunión de Planificación de Sprint!

La reunión de planificación de Sprint es probablemente lo más importante que se hace en Scrum. Emplea una buena cantidad de esfuerzo en hacerla bien, y el resto será mucho más fácil.

5 Comunicando los Sprints

Es importante mantener a toda la compañía informada sobre lo que está ocurriendo.

Para ello, introduciremos **en el apartado "Noticias"** de la forja del proyecto una entrada que contenga la siguiente información:

- Nombre del equipo
- Número de Sprint para ese proyecto: Sprint#1
- Pila de Sprint, con las historias y sus tiempos.
- Velocidad estimada.
- Calendario: inicio y fecha de finalización, hora y lugar scrum diario, hora y lugar demo final.
- Miembros del equipo, con su dedicación y (RP) identificando al responsable del proyecto.

Tan pronto como sea posible tras la reunión de planificación de Sprint, el Responsable del Proyecto crea la noticia con los datos.

6Cómo hacemos Pilas de Sprint

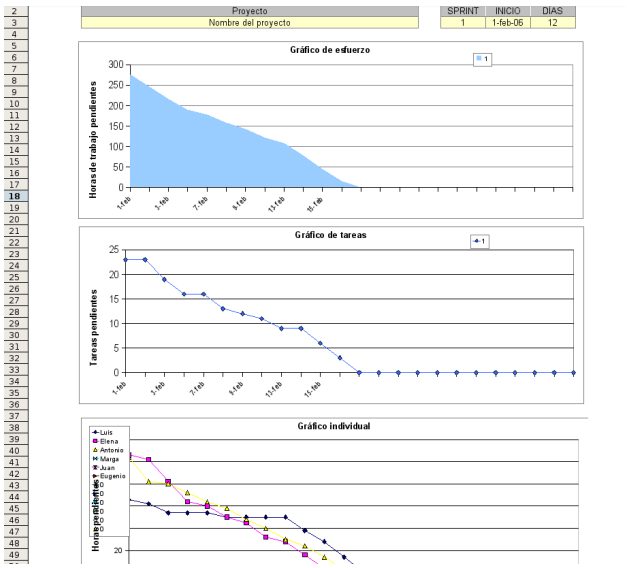
6.1 Formato de la Pila

Partimos de la Pila de Proyecto actualizada tras la Planificación y nos quedamos con las historias que han entrado en el Sprint, desglosadas por tareas hasta cierto punto. La avanzada herramienta que utilizaremos en un principio se trata de una simple hoja de cálculo (en "documentos" en la forja del proyecto). Podemos partir de una plantilla al uso[3] e ir adaptándola a nuestras necesidades.

SPRINT	INICIO	DURACIÓN
1	1-feb-06	12

							X	J	V	L	M	X	J	V	L	M	X	J														
							1-feb	2-feb	3-feb	6-feb	7-feb	8-feb	9-feb	10-feb	13-feb	14-feb	15-feb	16-feb														
					Tareas pendientes		23	23	19	16	16	13	12	11	9	9	6	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
					Horas de trabajo pendientes		276	246	216	190	178	158	143	122	108	78	46	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PILA DEL SPRINT							ESFUERZO																								
Backlog ID	Tarea	Tipo	Estado	Responsable																											
	Descripción de la tarea 1	Análisis	Terminada	Luis			16	16	16	16	16	16	16	16	16	4															
	Descripción de la tarea 2	Prototipado	Terminada	Luis			12	8																							
	Descripción de la tarea 3	Pruebas	Terminada	Luis			4	4	4	4	4																				
	Descripción de la tarea 4	Codificación	Terminada	Elena			8	4																							
	Descripción de la tarea 5	Codificación	Terminada	Elena			16	16	4																						
	Descripción de la tarea 6	Pruebas	Terminada	Elena			6	6	2																						
	Descripción de la tarea 7	Codificación	Terminada	Antonio			16	4			0	0	0	0	0	0	0	0													
	Descripción de la tarea 8	Codificación	Pendiente	Antonio			16	16	20	12	4																				
	Descripción de la tarea 9	Pruebas	Pendiente	Antonio			12	2																							
	Descripción de la tarea 10	Pruebas	Pendiente	Luis			12	12	12	12	12	12	12	12	12	12	6														
	Descripción de la tarea 11	Codificación	Pendiente	Luis			8	8	8	8	8	8	8	8	8	8	8	4													
	Descripción de la tarea 12	Codificación	En curso	Luis			14	14	14	14	14	14	14	14	14	14	14	10													
	Descripción de la tarea 13	Codificación	Pendiente	Antonio			8	8	8	8	8	6																			
	Descripción de la tarea 14	Codificación	Pendiente	Antonio			16	16	16	16	16	16	12	4																	
	Descripción de la tarea 15	Codificación	Pendiente	Antonio			16	16	16	16	16	16	16	16	10	4															
	Descripción de la tarea 16	Codificación	Pendiente	Elena			8	8	8																						
	Descripción de la tarea 17	Pruebas	Pendiente	Elena			12	12	12	8	4																				
	Descripción de la tarea 18	Codificación	Pendiente	Elena			16	16	16	16	16	10	5																		
	Descripción de la tarea 19	Pruebas	Pendiente	Elena			12	12	12	12	12	12	12	4																	
	Descripción de la tarea 20	Pruebas	Pendiente	Elena			16	16	16	16	16	16	16	16	16	4															
	Descripción de la tarea 21	Pruebas	Pendiente	Elena			12	12	12	12	12	12	12	12	12	12	4														
	Descripción de la tarea 22	Pruebas	Pendiente	Antonio			8	8	8	8	8	8	8	8	8	8	2														
	Descripción de la tarea 23	Pruebas	Pendiente	Antonio			12	12	12	12	12	12	12	12	12	12	2														

La ventaja: es sencilla y te genera automáticamente el gráfico Burn-Down, incluso el gráfico por integrante del equipo 8-0

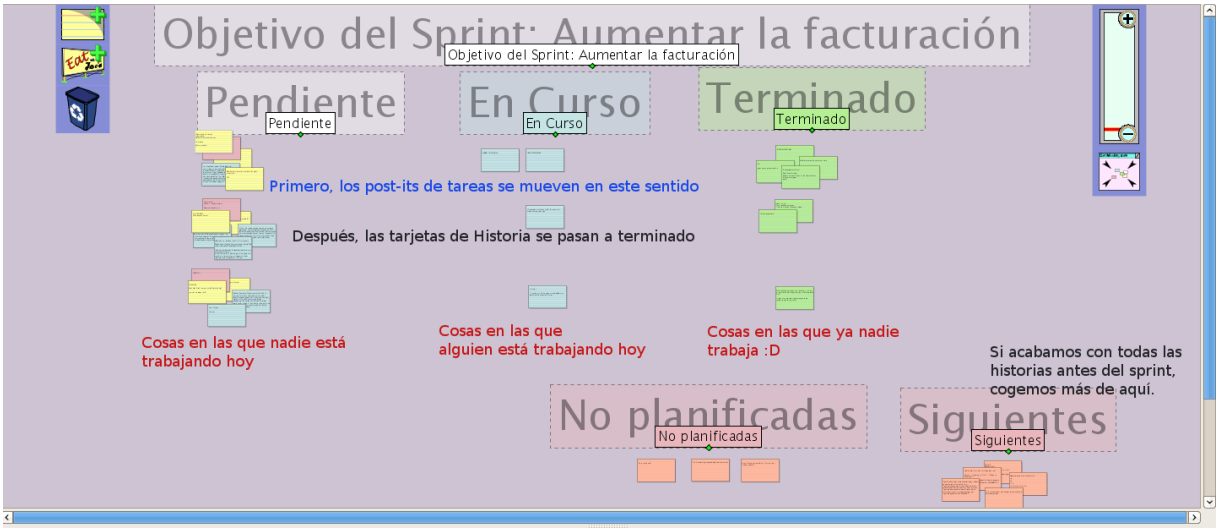


3 http://www.navegapolis.net/files/cis/plantilla_sprint.xls

Si los Scrums diarios o semanales se retrasan por la incomodidad de editar simultáneamente la hoja de cálculo, podemos utilizar de nuevo nuestro Tablón[4] y que después el Responsable del Proyecto actualice la Hoja:

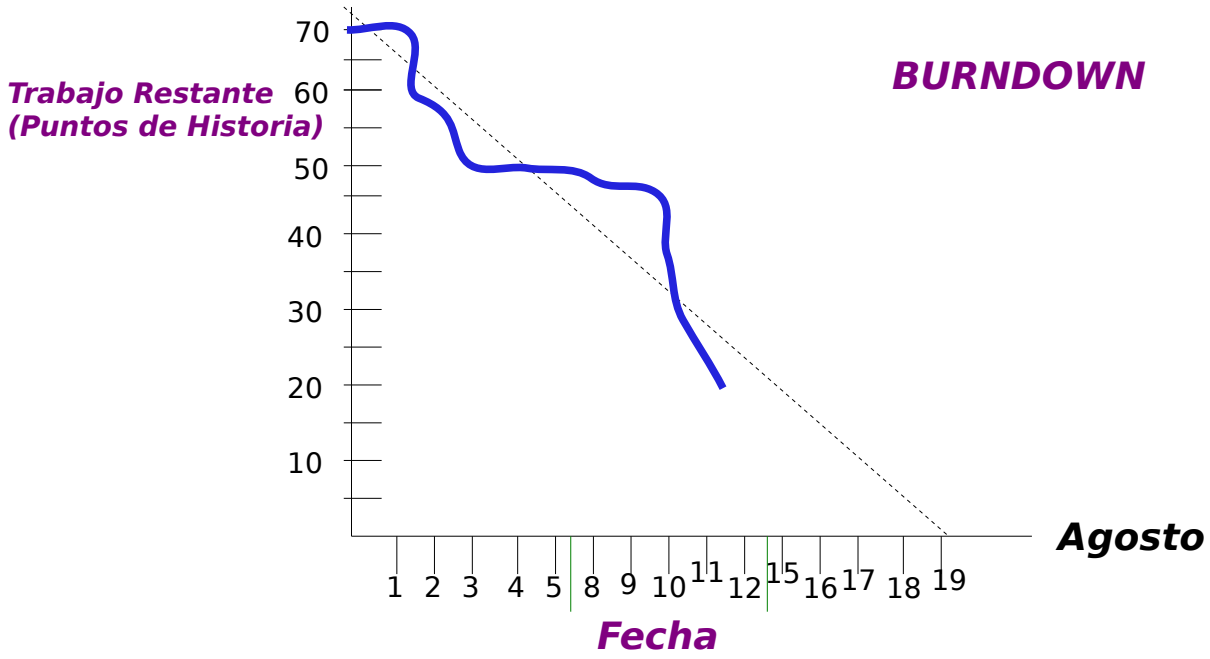
Las tarjetas de historia pueden ir en blanco y cada miembro del Core un color de post-it, por ejemplo.

Tenemos tres elementos no planificados, como puede verse abajo a la derecha. Esto es útil para recordar cuando hagamos retrospectiva del Sprint.



4 http://www.infoq.com/articles/agile-kanban-boards

6.2 Cómo funciona el diagrama Burn-Down

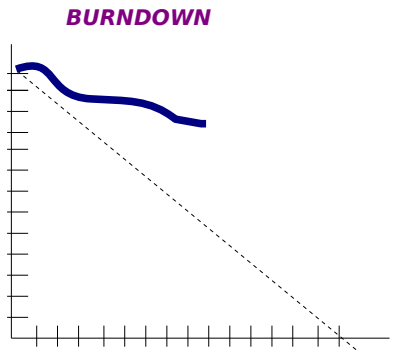


Este diagrama muestra que:

- En el primer día del Sprint, 1 de Agosto, el equipo estimó que habían aproximadamente 70 puntos de historia en los que trabajar. Esta era, consecuentemente, la velocidad estimada para todo el Sprint.
- El 12 de Agosto el equipo estima que quedan aproximadamente 20 puntos de historia por hacer. La línea de puntos nos muestra que estamos incluso algo avanzados respecto a la planificación, es decir, que a este paso completaríamos todo al final del Sprint.

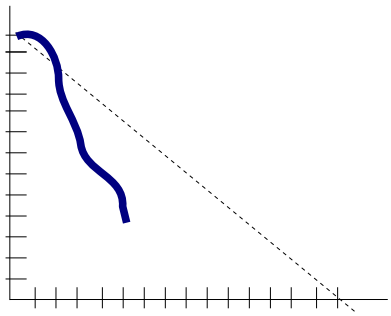
Nos saltamos los fines de semana en el eje X ya que rara vez se trabaja los fines de semana. Si incluimos los fines de semana, esto hará los burn-down algo confusos ya que se “aplanan” durante los fines de semana, lo que puede parecer una señal de peligro.

6.2.1. Cuidado con estas situaciones



No llegamos! Una de dos: o conseguimos más gente para el core o, si eso no sirve de mucho, hay que quitar algunos elementos de la Pila de Sprint. Puede que elementos no planificados estén arruinando el sprint.

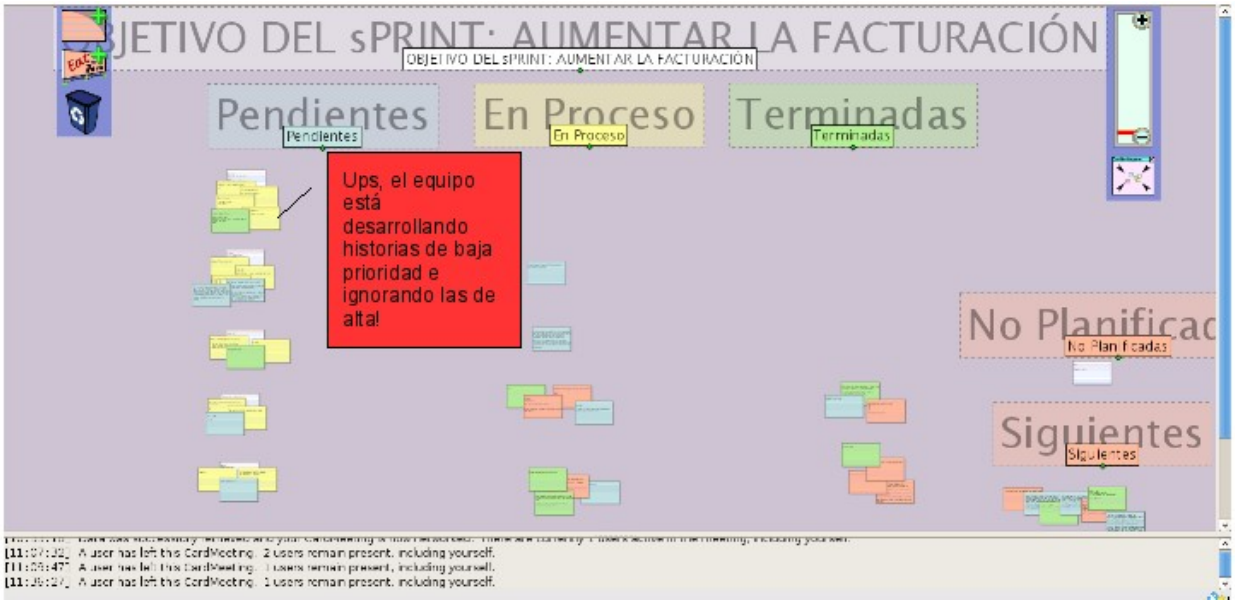
BURNDOWN



Hay que añadir algunas historias al Sprint. Malas estimaciones.

6.3 Trazabilidad

La hoja de Sprint nos proporciona la información detallada y actualizada día a día, y la evolución gráfica. Adicionalmente, podemos tomar un pantallazo del tablón después de cada Scrum diario. Al finalizar el sprint, tendremos una película del mismo.

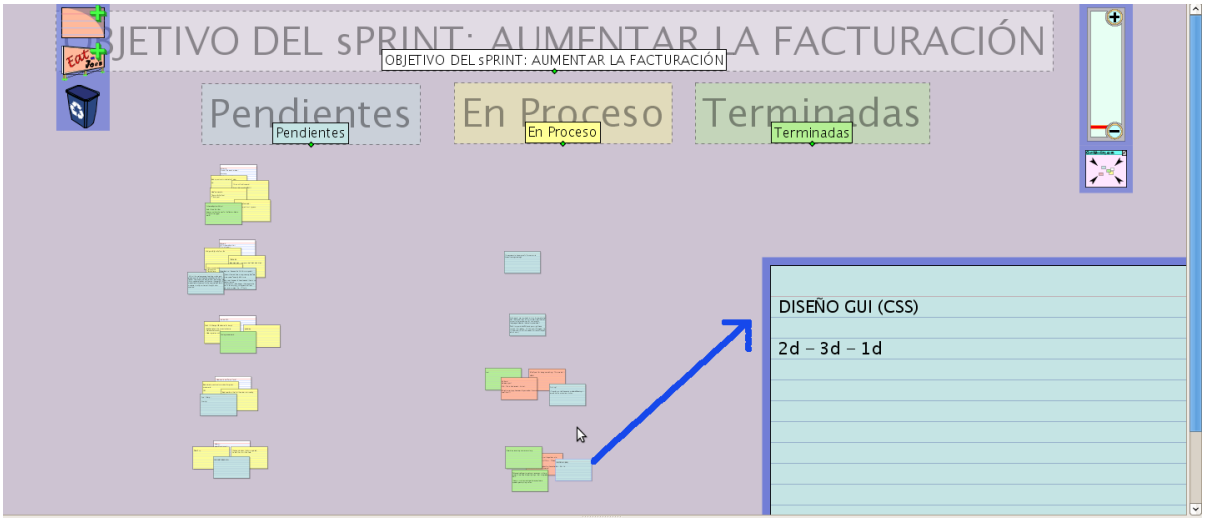


7Cómo hacemos Scrums diarios o semanales

Los scrums diarios o semanales se deben fijar de antemano al inicio del proyecto en función de las necesidades del mismo. Deben empezar exactamente a su hora, cada día o semana en el mismo sitio. Salvo casos de extraordinaria y urgente necesidad.

7.1 Cómo actualizamos la Pila de Sprint

Normalmente actualizaremos el tablón de tareas durante los Scrum diarios o semanales. Conforme cada persona describe lo que hizo el día anterior y lo que hará hoy, mueve los post-it en el tablón. Conforme describe elementos no planificados, pone un pos-it nuevo para cada uno de ellos. Conforme actualiza sus estimaciones, escribe una nueva estimación en el post-it correspondiente y tacha la anterior estimación. A veces el Responsable del Proyecto hace todo esto mientras los demás hablan.



Algunos equipos tienen la política de que cada persona debe hacer la actualización del tablón que le corresponda antes del cada reunión. Eso también funciona bien. Simplemente hay que decidir qué política utilizar y ceñirse a ella.

Sea cual sea el formato de la Pila de Sprint, hay que involucrar a todo el equipo en la labor de mantener la Pila de Sprint actualizada.

Sprints en los que el Responsable de Proyecto es el único que mantiene la Pila de Sprint y debe hablar con todo el mundo todos los días y preguntarles por sus estimaciones de trabajo pendiente tienen las siguientes desventajas:

- El Responsable del Proyecto pasa demasiado tiempo administrando asuntos, en vez de dar soporte al equipo y eliminando impedimentos.
- Los miembros del equipo no están al corriente del estado del Sprint, ya que la Pila del Sprint es algo de lo que no necesitan preocuparse. Esta falta de feedback reduce la agilidad y el enfoque generales del equipo. Si la Pila de Sprint está bien diseñada debería ser igual de fácil para cada miembro del equipo actualizarla el mismo.

Inmediatamente tras el Scrum diario, alguien actualiza las estimaciones de tiempo en la Pila de Sprint y ya tenemos un nuevo punto en el burn-down de Sprint.

7.2 Las personas impuntuales

Algunos equipos tienen una lata de monedas y billetes. Cuando llegas tarde, incluso aunque sea sólo por un minuto, añades una cantidad prefijada en la lata. Sin preguntas.

Si llamas antes de la reunión y avisas de que vas a llegar tarde, aun así tienes que pagar. Sólo te salvas de la multa si tienes una buena excusa como una cita con el médico, tu propia boda o algo similar.

El dinero de la lata puede usarse para destinar a proyectos sociales.

Esto funciona bien. Pero sólo es necesario en equipos donde es frecuente que la gente llegue tarde. Algunos equipos ni siquiera necesitan algo similar.

7.3 "No sé qué hacer hoy"

Ocurre a veces que alguien dice "Ayer hice bla bla bla, pero hoy no tengo la más remota idea de lo que hacer ". ¿Y ahora qué?

Digamos que Toni y Laura son los que no saben qué hacer hoy. Si soy el Responsable del Proyecto simplemente sigo con la reunión y dejo que el siguiente miembro del equipo hable, pero anoto quiénes son las personas que no saben qué hacer.

Después de que todo el mundo haya hablado, repaso el tablón de tareas con todo el equipo, de arriba a abajo, y me aseguro de que todo esté sincronizado, que todo el mundo sabe lo que significa cada elemento, etc.

Invito a todo el mundo a añadir más post-its. Entonces vuelvo a las personas que no sabían qué hacer y les digo "ahora que hemos repasado el tablón de tareas, ¿tenéis idea de lo que podéis hacer hoy?".

Si no, considera si no tienes aquí una buena oportunidad para hacer programación (o gestión, o lo que sea) por parejas. Digamos que Mikel va a implementar la interfaz de usuario de administración de usuarios del back-office hoy. En ese caso, sugiero educadamente si quizás Toni o Laura podrían emparejarse con Mikel para ello. Eso suele funcionar.

Si eso tampoco funciona y:

- Estáis a final del sprint, trata de que Toni y Laura preparen la Demo de Sprint
- Si estáis a mitad de sprint, da la enhorabuena al equipo por el buen trabajo que han realizado, agarra una o dos historias de la sección "siguientes" del tablón y colócalas en la columna de "pendiente" a la izquierda. Entonces haz de nuevo el Scrum diario. Notifica al Dueño de Producto que habéis añadido algunos elementos más al Sprint.

Puede ocurrir que el equipo no haya alcanzado aun el objetivo del Sprint y a pesar de ello Toni y Laura sigan negándose a especificar algo útil en lo que vayan a trabajar. Normalmente consideraremos alguna de las siguientes estrategias. Ninguna de ellas es especialmente agradable, pero ten en cuenta que se trata de un último recurso:

- Vergüenza: "Bueno, si no tenéis idea de cómo podéis ayudar al equipo os sugiero que os vayáis a casa, o leáis un libro o algo. O simplemente andar por aquí hasta que alguien del equipo os pida ayuda"
- Vieja escuela: Simplemente asígnales una tarea.
- Presión de los compañeros: Di "sentiros libres de tomaros el tiempo que haga falta, Toni y Laura, todos los demás simplemente nos quedaremos aquí en silencio hasta que se os ocurra algo que pueda ayudarnos a conseguir el objetivo del Sprint".

Si una persona te fuerza frecuentemente a ir tan lejos, probablemente deberías coger a esa persona en un aparte y hacer algo de coaching. Si el problema continúa, necesitas evaluar si esa persona es importante para el equipo o no.

Si no es demasiado importante, repórtalo al órgano competente e intenta que lo aparten de tu equipo. Si es demasiado importante, entonces intenta emparejarlo con alguien que pueda actuar como su "guía". Toni puede ser un desarrollador estupendo, pero simplemente puede preferir que sean otros los que le digan lo que tiene que hacer. Dale a Mikel la tarea de ser su guía permanente. O hazte cargo tú mismo de esa labor. Si Toni es suficientemente importante para el equipo, merecerá la pena el esfuerzo. Para todo lo demás, está la Dirección de la empresa.

8Cómo hacemos la demo del Sprint

La demo de Sprint (o revisión de Sprint, o entregable, o como la queráis llamar) es una parte importante de Scrum que se tiende a subestimar:

“Oh, ¿realmente tenemos que hacer una demo? ¡En realidad no hay mucho que podamos enseñar!” “¡No tenemos tiempo para montar una &%$# demo!” “¡No tengo tiempo para asistir a las demo de los demás equipos!”

8.1 Por qué insistimos en que todos los Sprints acaben con una demo

Una demo de Sprint bien ejecutada, aunque parezca poco espectacular, tiene un efecto muy profundo:

- El equipo obtiene reconocimiento por sus logros. Se sienten bien.
- La demo consigue feedback vital de las personas interesadas.
- Hacer una demo fuerza al equipo a acabar realmente las cosas y entregarlas (incluso aunque sea sólo en entorno de pruebas). Sin las demos, seguimos consiguiendo enormes montones de cosas terminadas al 99%. Con las demos puede que consigamos menos cosas terminadas, pero estas están realmente terminadas, lo que (en nuestro caso) es mucho mejor que tener una enorme pila de cosas que están más o menos listas y que se pulirán en el próximo Sprint.

8.2 Lista de comprobación para demos de Sprint

- Presentar claramente el objetivo del Sprint.
- No perder mucho tiempo preparando la demo, especialmente en llamativas presentaciones. Concéntrarse en mostrar código (o lo que tengamos) funcionando.
- Concentrar la preparación en hacer que la demo sea rápida en lugar de bonita.
- Mantener la demo a nivel de negocio, aparta los detalles técnicos.
- Concentrarse en “qué hemos hecho” en lugar de “cómo lo hemos hecho”.
- En la medida de lo posible, deja que la audiencia pruebe el producto por sí misma.
- No mostrar pequeños errores solucionados y funcionalidades triviales. No se muestran, sólo se mencionan. Normalmente se tarda mucho y desvía la atención de las historias más importantes.

8.3 Tratando con historias “indemostrables”

Ejemplo práctico

- Moe: “No voy a demostrar esta historia porque no puede demostrarse. La historia es ‘mejorar la escalabilidad de forma que el sistema pueda aguantar 10.000 usuarios simultáneos’. A ver como leches invito a 10.000 usuarios simultáneos a la demo.”
- R.P.: “¿Has terminado la historia?”
- Moe: “Sí, por supuesto”
- R.P.: “¿Cómo lo sabes?”
- Moe: “Monté el sistema en un entorno de pruebas de rendimiento, arranqué ocho servidores de carga y le di caña al sistema con solicitudes simultáneas”.
- R.P.: “Pero no tienes ninguna indicación de que pueda aguantar 10.000 usuarios simultáneos”

- Moe: “Sí. Las máquinas de pruebas están echas polvo, pero aun así pudieron aguantar 50.000 solicitudes simultáneas durante el test.
- R.P.: “¿Cómo lo sabes?”
- Moe: (frustrado) “¡Bueno, tengo este informe! ¡Muestra cómo se montó la prueba y cuántas solicitudes se enviaron!”
- R.P.: “¡Oh, excelente! Entonces ahí tienes tu ‘demo’. Simplemente muestra el informe y coméntaselo a la audiencia. Mejor que nada, ¿no?”
- Moe: “Ah, ¿basta con eso? Pero está muy feo, necesito ponerlo en bonito y…”
- R.P.: “Vale, pero no pierdas mucho tiempo. No se trata de que sea bonito, simplemente informativo”.

9Cómo hacemos retrospectivas de Sprint

9.1 Justificación

Lo más importante de las retrospectivas es **asegurarse de que tienen lugar.**

Por alguna razón, los equipos no siempre parecen inclinados a hacer retrospectivas.

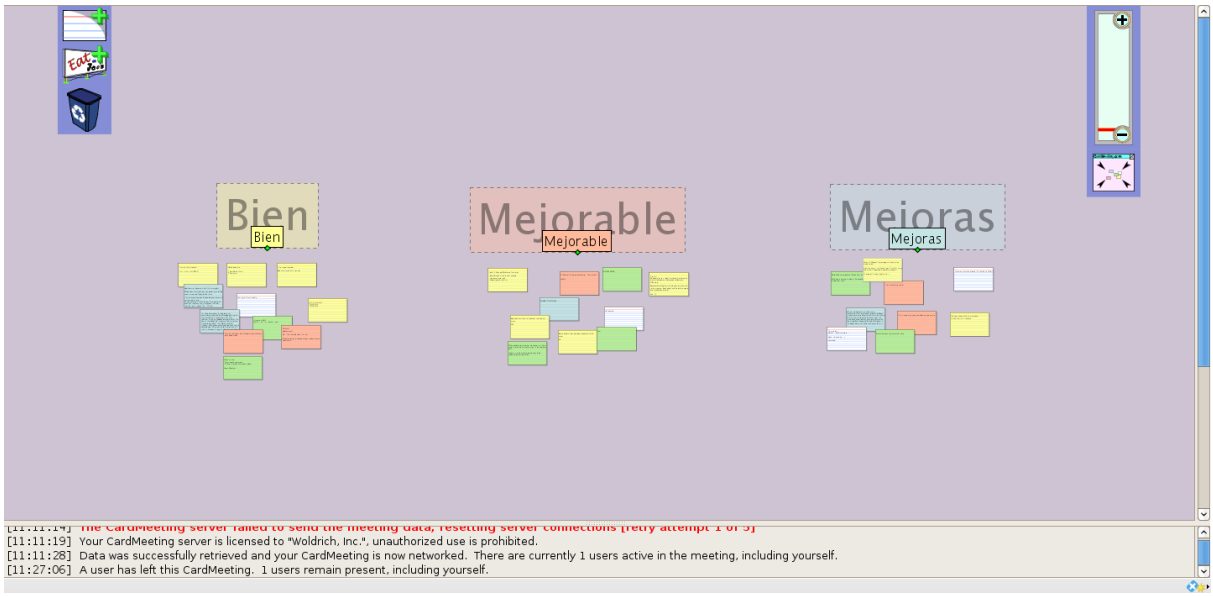
Todo el mundo coincide en que las retrospectivas son extremadamente útiles. **Es el segundo evento más importante de Scrum** (siendo el primero la reunión de planificación de Sprint) ya que **¡es la mejor oportunidad para mejorar!**

Sin las retrospectivas encontrarás que el equipo sigue cometiendo los mismos errores una y otra vez.

9.2 Cómo organizamos las retrospectivas

- Reservamos 1-3 horas, dependiendo de cuánta discusión esperemos.
- Participantes: el Cliente, el Equipo y el R.P.
- Alguien es designado secretario.
- El R.P. muestra la Pila de Sprint y, con ayuda del equipo, resume el Sprint. Eventos importantes, decisiones, etc.
- Hacemos “la ronda”. Cada persona tiene una oportunidad de decir, sin ser interrumpida, qué piensan que ha ido bien, que podría haber ido mejor y que piensan que debería hacerse de forma diferente en el próximo Sprint.
- Observamos la velocidad estimada frente a la real. Si hay una gran diferencia, intentamos analizar por qué.
- Cuando el tiempo casi se ha acabado, el R.P. trata de resumir las sugerencias concretas cobre qué puede hacerse mejor el próximo Sprint.

El tema subyacente es siempre el mismo: **“qué podemos hacer mejor el próximo Sprint”**.



Tres columnas:

- Bien: si hiciéramos el Sprint otra vez, volveríamos a hacer estas cosas igual.
- Mejorable: si hiciéramos otra vez el Sprint, haríamos estas cosas de forma diferente.

- Mejoras: ideas concretas sobre cómo podemos mejorar en el futuro.

Así que las columnas 1 y 2 son una mirada al pasado, mientras que la columna 3 mira al futuro.

Después de que el equipo genere todas estas ideas en post-its, utilizan "votación por puntos" para determinar en qué mejoras centrarse el próximo Sprint.

Basándonos en esto, seleccionamos 5 mejoras de procesos en los que concentrarse, y los evaluamos en la siguiente retrospectiva.

9.3 Trasmitiendo el conocimiento

La información que surge durante una retrospectiva de Sprint es casi siempre tremendamente valiosa. ¿Tiene este equipo problemas de concentración porque los gerentes de ventas insisten en secuestrar programadores para participar como "expertos técnicos" en reuniones de ventas? Esta es una información importante. Quizás otros equipos tengan los mismos problemas. ¿Deberíamos estar formando más a los responsables de producto acerca de nuestros productos, de forma que puedan hacer el soporte a ventas ellos mismos?

Una retrospectiva de Sprint no trata sólo de cómo este equipo puede hacerlo mejor el próximo Sprint, tiene implicaciones más amplias que esa.

Una opción sencilla es designar una persona que atienda a todas las reuniones de retrospectiva y actúa como puente de conocimiento.

Otra alternativa sería que cada equipo Scrum publique un informe de la retrospectiva de Sprint. El peligro de esto es que no mucha gente lee esos informes, y muchos menos actúan basándose en ellos. Enlazando esta metodología con ITIL/ISO 20000, más que mediante informes (que también) se trata de generar en base a ellos una **"Base de Conocimiento"** que recoja cuestiones recurrentes y mejores prácticas. Así, ante una cuestión, el responsable de proyecto puede recurrir a ella para ver si otro equipo se ha encontrado en la misma y cómo lo ha solucionado.

A continuación mostramos algunas de esas cuestiones, susceptibles de engrosar en la Base de Conocimiento:

9.3.1. Ejemplos de cosas que suelen surgir en las retrospectivas

- "Deberíamos haber pasado más tiempo dividiendo historias en subhistorias y tareas"

  Esta es muy habitual. Cada día en el Scrum diario, los miembros del equipo se encuentran a si mismos diciendo "Realmente no se qué hacer hoy". Así que después de cada Scrum diario tienes que pasar un tiempo encontrando tareas concretas. Es generalmente más efectivo hacer este trabajo desde el principio.

  Acciones típicas: ninguna. El equipo probablemente corregirá esto por si mismo en el próximo Sprint. Si esto ocurre repetidas veces, incrementa el tiempo para la planificación de Sprint.

- "Demasiadas distracciones"

  Acciones típicas:

  - Pide al equipo que reduzca su factor de dedicación el próximo Sprint de forma que tengan una planificación más realista.
  - Pide al equipo que lleven un registro de las interrupciones el próximo Sprint. Quién interrumpe, cuánto tiempo. Así será más fácil resolver el problema más adelante.
  - Pide al equipo que canalicen todas las distracciones a través del R.P. o el Cliente.
  - Pide al equipo que designe a una persona como "portero", y que todas las interrupciones se le envíen a él, de forma que el resto del equipo pueda concentrarse. Podría ser el R.P. o una posición rotatoria.

- "Nos sobre-comprometimos y sólo hicimos la mitad"

  Acciones típicas: ninguna. El equipo probablemente no se sobre-comprometerá en el próximo Sprint. O al menos no se sobre-comprometerá tanto.

- ¨Problemas de comunicación¨

Digamos que el equipo concluye que “nos hemos comunicado muy poco entre los miembros del equipo, así que hemos estado pisándonos unos a otros estropeando los diseños de los demás”.

En muchos casos, simplemente identificar el problema es suficiente para que se resuelva por si mismo automáticamente en el próximo Sprint. Si no, prueba a pasarles esta “Herramienta para medir la calidad de la interacción y comunicación del equipo[5]” para detectar el problema y actuar en consecuencia.

9.4 Descansos entre sprints

En la vida real no puedes estar siempre esprintando.

Lo mismo ocurre en Scrum y en el desarrollo de software en general. Los Sprints son muy intensos. Como desarrollador nunca puedes relajarte, todos los días o semanalmente tienes que estar de pie en la reunión y decirle a todo el mundo lo que conseguiste ayer.

Además del propio descanso, hay otra buena razón para dejar algo de espacio entre Sprints. Después de la demo y la retrospectiva tanto el Cliente como el Equipo estarán llenos de información y de ideas por digerir. Si se ponen de inmediato a planificar el próximo Sprint, es probable que nadie tenga la oportunidad de digerir ninguna información o lección aprendida, el Cliente no tendrá tiempo de ajustar sus prioridades después de la demo, etc.

Una forma de hacer esto son los **“días de laboratorio”**

Días en los que a los miembros del equipo se les permite hacer esencialmente cualquier cosa que deseen y que esté relacionada con sus conocimientos técnicos. Por ejemplo, leer sobre las últimas herramientas y API’s, estudiar para una certificación, discutir asuntos técnicos con colegas, programar un proyecto personal como hobby, etc.

Nuestro objetivo es tener **un día de laboratorio entre cada Sprint**. De esta forma obtenemos un descanso natural entre Sprints, y tendrás un equipo que tendrá una oportunidad realista de mantener su conocimiento al día.

5 http://www.navegapolis.net/files/blog/evaluacion_interaccion_equipo.xls

10 Planificación de entregas y contratos a precio cerrado

A veces necesitamos planificar por adelantado más de un Sprint a la vez. Cuando tenemos un contrato a precio cerrado en el que tenemos que planificar por adelantado, o correr el riesgo de firmar algo que no podemos entregar a tiempo.

La planificación de entregas es para nosotros un intento de resolver la pregunta “cuándo, como muy tarde, seremos capaces de entregar la versión 1.0 de este nuevo sistema”.

Esta versión de planificación de entregas es simple pero nos puede servir. Si necesitamos algo más elaborado tendríamos que adoptar otra.

10.1 Define tus umbrales de aceptación

Además de la Pila de Producto habitual, el Cliente define una lista de umbrales de aceptación, que son una simple clasificación de qué significan los niveles de importancia de la Pila de Producto en términos del contrato.

He aquí un ejemplo de umbrales de aceptación:

- Todos los elementos con importancia >=100 deben estar incluidos en la versión 1.0, o seremos penalizados hasta la muerte.
- Todos los elementos de importancia 50-99 deberían estar incluidos en la versión 1.0, pero podríamos pasar sin ellos si los incluyésemos en otra entrega poco después.
- Los elementos con importancias 25-49 son requisitos, pero podemos incluirlos en una versión 1.1.
- Los elementos con importancia <25 son puramente especulativos y puede que ni siquiera hagan falta.

He aquí un ejemplo de Pila de Producto con un código de colores basado en las reglas anteriores:

Importancia	Historia
130	Gasteiz
120	Bilbo
115	Santiago
110	Jaén
100	Madrid
95	Donosti
80	Iruña
70	Pisuerga
60	Salamanca
40	Leon
35	Toledo
10	Barna
10	Amunt

Rojo = debe incluirse en la versión 1.0 (Gasteiz – Madrid)

Amarillo = debería incluirse en la versión 1.0 (Donosti – Salamanca)

Verde = puede hacerse más tarde (León – Amunt)

Así que si lo entregamos todo desde Gasteiz a Salamanca en la fecha límite, estamos a salvo. Si el tiempo se nos acaba, podríamos salir adelante abandonando Donosti, Iruña, Pisuerga o Salamanca.

Cualquier cosa por debajo de ésta es un plus.

10.2 Estimación de los elementos más importantes

Para poder hacer la planificación de entregas el Cliente necesita estimaciones, al menos para todas las historias incluidas en el contrato. Al igual que en la planificación de Sprint, se trata de un esfuerzo cooperativo entre en Cliente y el equipo – el equipo estima, el Dueño de Producto describe los elementos y responde a las preguntas.

- Deja que el equipo haga las estimaciones.
- No dejes que le dediquen demasiado tiempo.
- Asegúrate de que entiendan que se trata de estimaciones, no compromisos.

Una estimación de tiempos es valiosa si resulta ser casi correcta, menos valiosa si resulta que falla por, digamos, un 30% y completamente inútil si no tiene ninguna conexión con la realidad.

Usualmente el Cliente reúne a todo el equipo y les dice que el objetivo de la reunión es estimar las principales 20 (o las que sean) historias de la Pila de Proyecto. Enuncia cada historia una vez y deja que el equipo se ponga manos a la obra. El Cliente se queda en la sala para responder preguntas y aclarar el alcance de cada historia si es necesario. Igual que en la planificación de Sprint, el campo “como probarlo” es muy útil para reducir el riesgo de malentendidos.

Esta reunión debe de ser limitada en tiempo, o si no el equipo tiende a perder demasiado tiempo estimando muy pocas historias.

Si el Cliente quiere que se dedique más tiempo simplemente organiza otra reunión para más adelante. El equipo debe asegurarse de que el impacto de estas reuniones en sus actuales Sprints sea claramente visible para el Cliente, de forma que entienda que su trabajo de estimación no es gratuito.

He aquí un ejemplo de cómo podrían acabar las estimaciones (en puntos de historia):

Importancia	Historia	Estimación
130	Gasteiz	12
120	Bilbo	9
115	Santiago	20
110	Jaén	8
100	Madrid	20
95	Donosti	12
80	Iruña	10
70	Pisuerga	8
60	Salamanca	10
40	Leon	14
35	Toledo	4
10	Barna	
10	Amunt	

10.3 Estimando la velocidad

Ahora tenemos algunas estimaciones rudimentarias para las historias más importantes.

El siguiente paso es estimar nuestra velocidad media por Sprint:

Esto significa que debemos decidir nuestro factor de dedicación. Lee el punto “Decidiendo qué historias incluir en el Sprint”.

Digamos que determinamos que el factor de dedicación del equipo es del 50% (bastante bajo). Y digamos que la duración del Sprint es de 3 semanas (15 días) y el tamaño del equipo es 6 personas.

Así que cada Sprint tendría 90 días-hombre ideales, pero solo podemos pretender producir el equivalente a 45 días-hombre ideales (debido al factor del 50%).

Así que nuestra velocidad estimada es 45 puntos de historia.

10.4 Uniéndolo todo en un plan de entregas (release plan)

Ahora que tenemos estimaciones de tiempo y una velocidad (45) podemos dividir fácilmente la Pila de Proyecto en Sprints:

Importancia	Historia	Estimación
Sprint1		
130	Gasteiz	12
120	Bilbo	9
115	Santiago	20
Sprint2		
110	Jaén	8
100	Madrid	20
95	Donosti	12
Sprint3		
80	Iruña	10
70	Pisuerga	8
60	Salamanca	10
40	Leon	14
Sprint4		
35	Toledo	4
10	Barna	
10	Amunt	

Cada Sprint incluye tantas historias como sea posible sin exceder la velocidad estimada de 45.

Así, podemos ver que probablemente necesitemos 3 Sprints para finalizar todos los “debe” y los “debería”.

3 Sprints = 9 semanas de calendario = 2 meses. Así que esa es nuestra **fecha de entrega**. Ahora bien, ¿es la fecha que se prometió al cliente? Depende enteramente de la naturaleza del contrato: **como de fijo es el alcance**, etc.

Usualmente añadiremos un **colchón** significativo para protegernos contra las malas estimaciones, problemas inesperados, etc. Así que en este caso podríamos acordar fijar la fecha de entrega dentro de 3 meses, dándonos así un mes de “reserva”.

Lo bonito es que podemos mostrar algo usable al cliente cada tres semanas e invitarle a introducir cambios en los requisitos conforme avanzamos (dependiendo por supuesto de lo que permita el contrato).

10.5 Adaptando el plan de entregas

La realidad no se adaptará ella sola al plan, así que tendremos que hacerlo al revés.

Después de cada Sprint comprobamos la velocidad real de dicho Sprint. Si la velocidad real ha sido muy diferente de la estimada, **revisamos la velocidad estimada para próximos Sprints y actualizamos el plan de entregas**. Si esto nos coloca en una situación problemática, puede que el Cliente empiece a **negociar con el cliente** real **o** se dedique a averiguar cómo podemos **reducir el alcance sin romper el contrato**. O quizás él y el equipo encuentren una forma de aumentar la velocidad eliminando algún impedimento severo que se haya identificado durante el Sprint.

El Cliente (interno) podría llamar al cliente y decirle “hola, vamos un poco retrasados respecto a la planificación, pero creo que podríamos cumplir con la fecha de entrega si eliminásemos la funcionalidad “comecocos embebido” que nos llevaría un montón de tiempo construir. Podríamos añadirla en la siguiente entrega, 3 semanas después de la primera versión, si así lo quieres”.

Quizás no sean buenas noticias para el cliente, pero al menos estamos siendo honestos y le estamos dando al cliente opciones muy pronto: **podemos entregar las funcionalidades más importantes en fecha o entregarlo todo pero tarde**.

11 Cómo haremos pruebas

Es necesario algún tipo de fase de pruebas de aceptación manuales. Se trata de que encargados de pruebas dedicados que no/sí son parte del equipo prueben el sistema con ese tipo de pruebas que el equipo de Scrum no pudo imaginar, o no tuvo tiempo de hacer o no contaban con el hardware necesario para implementar. Los encargados de pruebas acceden al sistema en la forma exacta en la que los usuarios finales lo harán, lo que significa que debe hacerse manualmente (asumiendo que nuestro sistema sea para usuarios humanos).

El equipo de pruebas encontrará errores, el equipo Scrum tendrá que hacer las correcciones necesarias, y tarde o temprano (esperemos que temprano) será posible lanzar una versión 1.0.1 a los usuarios finales en lugar de la inestable 1.0.0.

La fase de pruebas es dura. La sensación es distintivamente no-Ágil. Aunque no podemos librarnos de ella, sí que podemos minimizarla (y lo haremos). Más específicamente, minimizamos la cantidad de tiempo necesario para la fase de pruebas. Esto lo conseguimos:

- Maximizando la calidad del código desarrollado por el equipo Scrum.
- Maximizando la eficiencia del trabajo manual de pruebas (es decir, encontrar los beta-testers, darles la mejores herramientas y asegurarnos de que informan de las tareas que pueden automatizarse).

Así que ¿cómo maximizamos la calidad del código desarrollado por el equipo Scrum? Bueno, hay muchas maneras. He aquí dos que funcionan muy bien:

- Incluir encargados de pruebas en el equipo Scrum
- Hacer menos cosas en cada Sprint

11.1 Incrementar la calidad incluyendo encargados de pruebas en el equipo

Lo que queremos decir con “encargado de pruebas” en este caso es “una persona cuya principal habilidad es hacer pruebas” y no “un tipo cuya única responsabilidad es hacer pruebas”.

Los desarrolladores son frecuentemente muy malos encargados de pruebas. Especialmente los desarrolladores que deben probar su propio código.

Definición de Terminado

Es importante que el Cliente y el equipo estén de acuerdo en una definición clara de “terminado”. ¿Está terminada una historia cuando todo el código ha sido chequeado? ¿O está terminada cuando se ha instalado en un entorno de pre-producción y ha sido verificada por un equipo de pruebas de integración?

Siempre que es posible, utilizaremos la definición **“lista para pasar a producción”**, aunque a veces debemos conformarnos con “instalado en pre-producción y lista para las pruebas”.

Usualmente decimos **“una historia está terminada cuando el encargado de pruebas del equipo Scrum lo dice”**. Así que es labor del encargado de pruebas asegurarse de que la intención del Cliente es correctamente entendida por el equipo, y que el elemento esté lo suficientemente “terminado” como para cumplir con la definición de terminado.

Todas las historias no se pueden tratar igual. Una historia llamada “formulario de consulta de usuarios” se tratará de una forma muy diferente a otra llamada “manual de operaciones”. En el último caso, la definición de terminado puede significar simplemente “aceptada por el equipo de operaciones”. Es por eso que el sentido común es, a menudo, mejor que las listas de comprobación formales.

Si nos sentimos confusos con frecuencia acerca de la definición de terminado probablemente deberíamos tener un campo “definición de terminado” en cada historia individual.

11.1.1. El encargado de pruebas es quien da el visto bueno

Además de ser "sólo" un miembro del equipo, el encargado de pruebas tiene una labor importante. Es el que da el visto bueno. Nada se considera terminado hasta que él dice que está terminado.

¿Cómo sabe el Señor P. (nuestro encargado de pruebas) que algo está terminado? Bueno, antes que nada debería (sorpresa) probarlo. En muchas ocasiones ocurre que algo que el programador consideraba terminado ni siquiera era posible de probar. Porque no se había registrado, o no se había instalado en el servidor de pruebas, o no podía arrancarse o lo que sea. Una vez que el Señor P. ha probado la funcionalidad, debería revisar la lista de comprobación de "terminado" (si tenéis una) con el desarrollador. Por ejemplo, si la definición de "terminado" establece que debería haber una nota de versión, entonces el Señor P. comprueba que haya una nota de versión. Si hay algún tipo de especificación más formal para esta funcionalidad (algo raro en nuestro caso) entonces el Señor P. la comprueba también. Etcétera.

Un bonito efecto secundario de esta práctica es que el equipo tiene ahora una persona que está perfectamente preparada para organizar la Demo del Sprint

11.2 Ciclos de Sprint vs. ciclos de pruebas

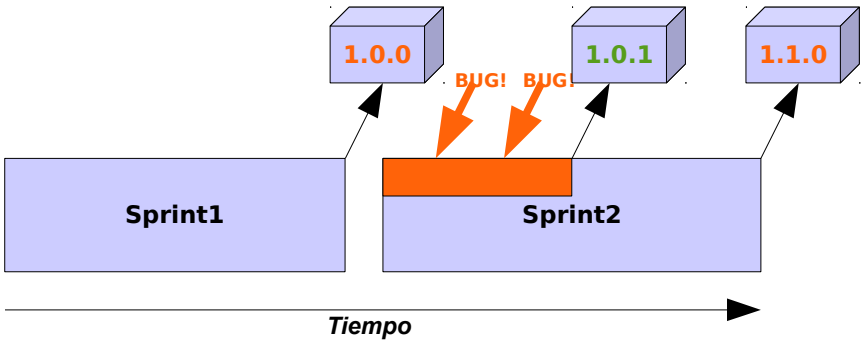
Lo primero de todo, maximiza la calidad del código que el equipo Scrum libera. El coste de encontrar y corregir errores pronto, durante el Sprint, es extremadamente bajo comparado con encontrarlos y corregirlos más tarde.

Pero el problema permanece, incluso aunque minimicemos el número de errores, aun seguirán apareciendo errores después de que se complete el Sprint. ¿Cómo nos enfrentamos a ello?

Básicamente, cuando acabamos un Sprint comenzaremos con el siguiente. Pero esperamos pasar parte del tiempo del próximo Sprint encontrando errores del último Sprint. Si el próximo Sprint queda seriamente dañado porque hemos tenido que pasar demasiado tiempo arreglando errores del Sprint anterior, evaluamos por qué ha ocurrido esto y cómo podemos mejorar la calidad. Nos aseguramos de que los Sprints son suficientemente largos como para soportar una cantidad aceptable de correcciones del último Sprint.

Gradualmente, durante un periodo de muchos meses, la cantidad de tiempo que pasaremos solucionando errores de Sprints anteriores disminuirá.

Adicionalmente, deberemos ser capaces de dedicar menos personas involucradas cuando se producían errores, así que todo el equipo no tendrá que ser molestado en cada ocasión.



Durante las reuniones de planificación de Sprint **establecemos el factor de dedicación suficientemente bajo como para tener en cuenta el tiempo que esperamos dedicar a encontrar y corregir errores del último Sprint**. Con el tiempo, los equipos se deben volver buenos estimando este tiempo. La métrica de velocidad ayuda mucho.

11.3 Sobrecarga del Señor P.

Digamos que las pruebas de aceptación son tu eslabón más lento. Tienes muy pocos encargados de pruebas, o las pruebas de aceptación tardan demasiado por la baja calidad del código. Digamos que tu equipo de pruebas puede probar como mucho 3 funcionalidades a la semana (por ejemplo). Y digamos que tus desarrolladores pueden desarrollar 6 nuevas funcionalidades por semana.

Sería tentador para los gerentes o Clientes (o quizás incluso para el equipo) planificar el desarrollo de 6 nuevas funcionalidades a la semana. No hacerlo. En lugar de eso, planificad 3 nuevas funcionalidades a la semana y pasad el resto del tiempo aligerando el cuello de botella de las pruebas. Por ejemplo:

- Haz que unos cuantos desarrolladores trabajen como encargados de pruebas (oh, te adorarán por ello… ;P
- Implementa herramientas y scripts que hagan las pruebas más fáciles.
- Añade más código de pruebas automatizadas
- Incrementa la longitud de los Sprints y haz que se incluyan pruebas de aceptación en los Sprints.
- Define algunos Sprints como “Sprint de pruebas” en los que todo el equipo trabaje como un equipo de pruebas.

Las retrospectivas son un buen foro en el que identificar el eslabón más débil de la cadena.

12 Manejando múltiples equipos Scrum

Un montón de cosas son más complicadas cuando tienes varios equipos trabajando en el mismo producto. Este problema es universal y realmente no tiene mucho que ver con Scrum. Más miembros del equipo = más complicaciones.

Las preguntas clave son:

- Cuántos equipos crear
- Cómo distribuir a la gente en equipos

12.1 Cuántos equipos crear

Cuando tienes un equipo >8-9 personas, los Scrum diarios o semanales tienden a alargarse más de los 15 minutos. Los miembros del equipo no saben lo que otros miembros están haciendo, así que hay bastante confusión.

La alternativa es dividir en dos equipos. Pero, ¿es esto mejor? No necesariamente.

Si el equipo tiene experiencia y se siente a gusto con Scrum, y hay una manera lógica de dividir el plan de producto en dos caminos distintos, y esos dos caminos no involucran el mismo código, entonces diría que es una buena idea dividir el equipo. En otro caso, consideraríamos mantener un solo equipo a pesar de las desventajas de un equipo grande.

¡Haz equipos pequeños sólo cuando no necesiten interactuar unos con otros!

12.1.1. Equipos virtuales

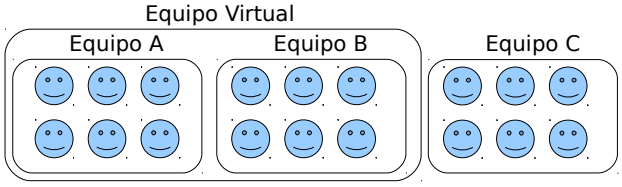
¿Como sabes si has tomado una Buena decisión respecto a los pros y los contras de “equipo grande” frente a “equipo pequeño”? Si mantienes tus ojos y tus oídos abiertos puede que notes como se forman “equipos virtuales”.

12.1.1.1. Ejemplo 1

Escoges tener un equipo grande. Pero cuando observas quién habla con quién durante el Sprint notas que el equipo se ha separado a todos los efectos en dos sub-equipos.

12.1.1.2. Ejemplo 2

Escoges tener tres equipos más pequeños. Pero cuando empiezas a ver quién habla con quién durante el Sprint notas que el equipo 1 y el equipo 2 hablan todo el rato, mientras que el equipo 3 trabaja por su cuenta.



Así que ¿qué significa todo esto? ¿Que tu estrategia de división era errónea?

Sí, si el equipo virtual parece permanente. No, si el equipo virtual parece algo eventual.

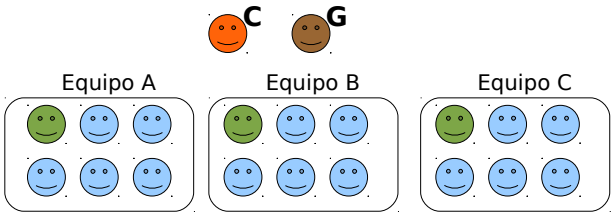
Observa otra vez el ejemplo 1. Si los dos sub-equipos virtuales tienden a cambiar cada cierto tiempo (por ejemplo, la gente cambia de sitio entre los diferentes sub-equipos virtuales) entonces probablemente habrás hecho la decisión correcta al mantenerlos como un solo equipo Scrum. Si los dos sub-equipos se mantienen igual durante todo el Sprint probablemente prefieras dividirlos en dos equipos Scrum para el próximo Sprint.

Ahora observa otra vez el ejemplo 2. Si el equipo 1 y el equipo 2 hablan entre ellos (y no con el equipo 3) durante todo el Sprint, probablemente quieras combinar los equipos 1 y 2 en un solo equipo Scrum el próximo Sprint. Si el equipo 1 y el equipo 2 hablan entre ellos durante la primera mitad del Sprint, y entonces el equipo 1 y el equipo 3 hablan entre ellos durante la segunda mitad del Sprint, entonces quizás deberías considerar combinar los tres equipos en uno, o simplemente dejarlos como tres equipos Scrum. Comenta el tema durante la retrospectiva de Sprint y deja que los equipos decidan por si mismos.

La división en equipos es una de las partes realmente duras de Scrum. No pienses demasiado u optimices en exceso. Experimenta, mantente alerta ante la formación de equipos virtuales y asegúrate de que tienes tiempo de sobra para discutir estos aspectos durante las retrospectivas. Tarde o temprano encontrarás la solución correcta para tu situación concreta. Lo importante es que los equipos se sientan a gusto y no tropiecen unos con otros demasiado a menudo.

12.2 El "Guía de Equipo"

Pongamos que tenemos un solo proyecto con tres equipos:



El tipo de rojo con la C es el Cliente. Los tipos con lupus (de verde) son los responsables de equipo. El resto son esbirros…eh…respetables miembros del equipo.

En esta constelación, ¿Quién decide qué personas deberían estar en qué equipos? ¿Consejo? ¿El Cliente? ¿Los tres responsables juntos? ¿O debería cada persona seleccionar su equipo? Pero ¿qué pasa si todo el mundo quiere estar en el equipo 1 (porque el Responsable es tan guapo)?

¿Y si luego resulta que realmente no es posible tener más de dos equipos trabajando en paralelo en esta base de código, así que necesitamos transformar esta situación en 2 equipos de 9 personas en lugar de 3 equipos de 6? Eso significa 2 responsables. Así que ¿Cuál de los 3 actuales será despojado de su título?

En algunas empresas éstas pueden ser cuestiones delicadas. Y no queremos problemas, así que introducimos el tipo de marrón con la "G", con el rol de "guía de equipo". Esto correspondería a lo que podríamos llamar el "Scrum de Scrum Masters" o "el Jefe" o "Scrum Master Jefe", etc. **No tiene que liderar ningún equipo, pero es responsable de los asuntos entre equipos como quién debería ser Responsable en cada equipo, cómo debería dividirse la gente entre equipos, etc.**

12.3 Cómo asignamos personas a los equipos

Hay dos estrategias generales para asignar personas a los equipos cuando tienes múltiples equipos para el mismo producto.

- Dejar que una persona designada haga la distribución, por ejemplo el "guía de equipo" de antes, el Cliente o el Consejo (si es que está suficientemente involucrado como para hacer buenas decisiones al respecto).
- Dejar que sean las propias personas las que decidan de alguna forma.

Normalmente la combinación de ambas es lo que mejor funciona:

Antes de la reunión de planificación de Sprint, el guía de equipo convoca una reunión de distribución de personas con el Cliente y todos los Responsables de equipo. Hablamos sobre el último Sprint y decidimos si es necesaria una redistribución del personal. Quizás queramos combinar dos equipos, o cambiar algunas

personas de un equipo a otro. Decidimos algo y lo ponemos por escrito como propuesta de distribución de equipos, y lo llevamos a la reunión de planificación de Sprint. Lo primero que hacemos en la reunión de planificación de Sprint es repasar los elementos más prioritarios de la Pila de Proyecto. El guía de equipos dice algo así como:

"Hola a todos. Sugerimos la siguiente distribución de personas para el próximo Sprint."

Distibución *preliminar* de equipos

Equipo 1	Equipo 2	Equipo 3
Aragorn	Frodo	Gandalf
Legolas	Arwen	Sam
Bilbo	Theoden	Eowyn
Isildur	Boromir	Gollum

"Bueno, esta es una distribución preliminar. Es sólo un punto de partida, para ahorrar tiempo. Conforme la reunión de planificación de Sprint prospere, sentíos libres de cambiad de equipo, dividir vuestro equipo en dos, combinaros con otro equipo o lo que sea. Usad el sentido común en función de las prioridades marcadas por el Cliente."

Vamos, un cierto nivel de control centralizado al principio, seguido de un cierto nivel de optimización descentralizada después.

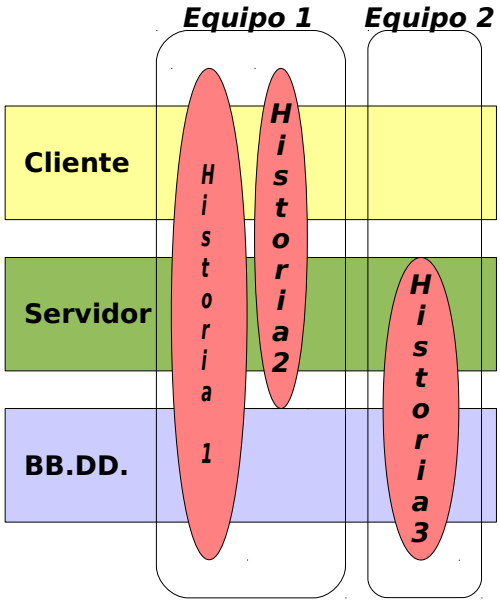
12.4 Equipos especializados Vs multidisciplinares

Pongamos que tu tecnología consta de tres componentes principales: Cliente – Servidor – Base de Datos. Y digamos que tienes 15 personas trabajando en este producto, así que no quieres manejarlos como un solo equipo Scrum. ¿Qué equipos crearías?

Normalmente las personas con conocimientos específicos tienden a agruparse, porque se sienten más cómodos, así que lo típico es que tengas: nosotras (personas con amplios conocimientos de sistemas) nos ocupamos de la parte servidor, nosotras (personas a las que les gusta el diseño web y tal) nos ocupamos de la interfaz usuario y nosotras (desarrolladoras) nos metemos con las tablas en la BB.DD.

Mala idea, porque en el 99% de los casos los 3 equipos tendrán que colaborar para conseguir terminar la historia.

Lo ideal es formar equipos multidisciplinares, de forma que cada equipo puede implementar una historia completa incluyendo las partes de cliente, servidor y base de datos. Los equipos pueden así trabajar de forma más independiente, lo cuál es bueno.



Esto además disminuye el número de casos de “no podemos completar este elemento porque estamos esperando a que los tíos del equipo servidor hagan su parte”.

Sin embargo, de vez en cuando tendremos que crear equipos coyunturales especializados en algún componente cuando hay una fuerte necesidad.

12.5 Miembros a tiempo parcial

¿Tener miembros a tiempo parcial es buena idea?

Digamos que estás a punto de incorporar a José como miembro a tiempo parcial de tu equipo Scrum. Piénsatelo bien primero. ¿Realmente necesitas a José en el equipo? ¿Estás seguro de que no puedes incorporar a José a tiempo completo? ¿Cuáles son sus otros compromisos? ¿Puede otra persona encargarse de los compromisos de José y permitir que José tome un papel menos activo, de soporte, en dichos compromisos? ¿Puede José unirse a tiempo completo el próximo Sprint y mientras tanto ir transfiriendo sus otras responsabilidades a otra persona?

A veces simplemente no hay otra solución. Necesitas a José desesperadamente porque es el único administrador de base de datos en la empresa, pero el resto de equipos también le necesitan así que no puede comprometerse a tiempo completo, y no podemos contratar a más administradores de base de datos. Estupendo. Ese es un caso válido para incorporarle a tiempo parcial. Pero asegúrate de realizar esta evaluación cada vez.

En general todo Responsable de proyecto prefiere tener un equipo de 3 a tiempo completo que de 8 a tiempo parcial.

Si tienes a una persona que divide su tiempo entre diferentes equipos, como el administrador de bases de datos de antes, es buena idea que de todas formas esté asignado principalmente a un equipo. Estima cuál es el equipo que le necesitará más y conviértelo en su “equipo matriz”. Cuando nadie esté tirando de él, atenderá a los Scrum diarios, planificaciones de Sprint, retrospectivas, etc., de ese equipo.

12.6 Scrum de Scrums

Scrum de Scrums es básicamente una reunión periódica a la que los Responsables de equipo y de proyectos acuden para hablar.

Esta reunión es muy importante. La haremos una vez al mes, a veces incluso más a menudo. Discutiremos aspectos de integración, equilibrio entre equipos, preparación de la próxima reunión de planificación de Sprint, etc.

Disponemos de 30 minutos, aunque frecuentemente nos pasaremos. Una alternativa es tener Scrum de Scrums todos los días, pero generalmente es inviable.

La agenda típica Scrum de Scrums:

1. Todo el mundo describe qué ha conseguido su equipo la última semana, qué planean hacer esta semana y qué impedimentos tienen.
2. Cualquier otro asunto inter-equipos que necesite ser discutido, por ejemplo aspectos relativos a la integración.

La agenda del Scrum de Scrums no es realmente importante, lo importante es que se celebren reuniones Scrum de Scrums regularmente.

12.7 Intercalando los Scrums diarios

Si tienes varios equipos Scrum en un solo proyecto y todos ellos hacen el Scrum diario o semanal al mismo tiempo, tienes un problema. El Cliente sólo pueden atender a una de las reuniones de Scrum.

Así que le pediremos a los equipos que intenten no tener las reuniones al mismo tiempo.

Esto es extremadamente útil por dos razones:

1. Personas como el Cliente o alguien de Consejo pueden visitar todos los Scrum diarios en una sola mañana. No hay mejor manera de obtener una visión exacta de cómo esta funcionando el Sprint y cuales son las amenazas clave.
2. Los equipos pueden visitar los Scrum diarios de otros equipos. No ocurre muy a menudo, pero cada cierto tiempo dos equipos pueden acabar trabajando en áreas similares, así que algunos miembros de un equipo se dejan caer por el Scrum diario del otro equipo para mantenerse al tanto.

La desventaja es una menor libertad para el equipo – no pueden escoger la hora que quieran para el Scrum.

12.8 Equipos apagafuegos

Pongamos una situación en la que un proyecto muy grande no cumple porque pasan demasiado tiempo apagando fuegos, es decir, liberando parches de emergencia para su sistema, que ha sido entregado en un estado muy prematuro de desarrollo. Este es un círculo vicioso, ya que están tan ocupados apagando fuegos que no tienen tiempo para trabajar proactivamente en prevenir fuegos (es decir, mejorar el diseño, automatizar las pruebas, crear herramientas de monitorización, herramientas de alarmas, etc.).

Solucionaremos este problema (y otros similares) creando un equipo apagafuegos dedicado y un equipo Scrum.

La labor del equipo Scrum será (con las bendiciones del Cliente) intentar estabilizar el sistema y, efectivamente, prevenir fuegos.

El equipo apagafuegos (al que realmente llamaremos “soporte”) tendrá dos labores:

1. Apagar fuegos
2. Proteger al equipo Scrum de todo tipo de distracciones, incluyendo cosas como defenderse de peticiones de funcionalidades ad-hoc que aparecen de la nada.

En cualquier caso, conviene que esta situación se prolongue lo menos posible, que el proyecto sea suficientemente estable como para eliminar el equipo apagafuegos y crear nuevos equipos Scrum en su lugar. Los apagafuegos se sentirán felices de poder colgar sus aporreados cascos y unirse a equipos Scrum en su lugar.

Si tenéis que montar equipos apagafuegos frecuentemente, no hace falta decir que algo va mal :/

12.9 La planificación de Sprint multiequipo

Digamos que tenemos un producto y dos equipos Scrum. ¿Cuántas Pilas de Proyecto deberías tener? ¿Cuántos Clientes? Hay tres posibilidades:

1. Un Cliente, Una Pila de Proyecto.
2. Un Cliente, múltiples Pilas de Proyecto
3. Múltiples Clientes y Pilas de Proyecto (una por cada)

Nos quedaremos con la primera para simplificar.

Lo bueno de este modelo es que puedes permitir a los equipos formarse ellos mismos en función de las prioridades actuales del Cliente. **El Cliente puede centrarse en lo que él necesita, y dejar a los equipos que decidan como dividir el trabajo.**

He aquí como funcionaría la **reunión de planificación de Sprint** para este equipo:

1. Justo antes de la reunión, el Cliente pone las historias en el Tablón (tarjetas), ordenadas por prioridad relativa. Continúa poniéndolas hasta que el Tablón está lleno, lo cual normalmente es más que suficiente para un Sprint.
2. Cada equipo selecciona una sección de Tablón vacía para sí y colocan su nombre en ella. Cada equipo, entonces, va cogiendo historias de la Pila de Producto comenzando por las de máxima prioridad y las coloca en su propia sección.
3. Conforme la reunión progresa, el Cliente y los equipos negocian con las tarjetas, rotándolas de equipo en equipo, moviéndolas arriba y abajo para cambiar su prioridad, dividiéndolas en historias más pequeñas, etc.
4. Después de una hora más o menos, cada equipo tiene una primera versión de lo que podría ser su Pila de Sprint en su sección. A partir de ahí los equipos trabajan de forma independiente, estimando tiempos y dividiendo en tareas.

Es un trabajoso, caótico y agotador, pero también es muy efectivo. Cuando se acaba el tiempo, todos los equipos tienen usualmente suficiente información como para comenzar con su Sprint.

12.10 Retrospectiva Multiequipo

¿Cómo hacemos las retrospectivas cuando tenemos múltiples equipos trabajando en el mismo proyecto?

Inmediatamente después de la demo de Sprint cada equipo hace su retrospectiva.

Durante la reunión de planificación de Sprint (a la que todos los equipos asisten), la primera cosa que hacemos es dejar que un portavoz de cada equipo se levante y resuma los puntos clave de su retrospectiva. Lleva unos cinco minutos por equipo. Entonces tenemos una discusión abierta durante unos 10 – 20 minutos. A continuación hacemos un descanso. Y después comenzamos con la planificación del Sprint propiamente dicha.

Para productos de un solo equipo, no hacemos resumen de retrospectiva durante la planificación de Sprint. No hace falta, ya que todo el mundo estuvo en la reunión de retrospectiva.

13 Lista de Comprobación (Checklist) del responsable de Proyecto

En este capítulo final mostraremos la “lista de comprobación”, que enumera la mayoría de las rutinas administrativas de los Responsables de Proyecto (R.P.'s). Cosas que es fácil olvidar. Nos saltaremos las evidentes como “eliminar impedimentos del equipos”.

Comienzo del Sprint

- Tras la reunión de planificación de Sprint, crea la hoja de sprint y abre un post en la Forja con la info del Sprint.
- Manda un correo a todo el mundo anunciando que ha comenzado un nuevo Sprint. Incluye el objetivo del Sprint y un enlace a la página de noticias de la Forja.
- Actualiza el documento de estadísticas de Sprint. Añade tu velocidad estimada, tamaño del equipo, longitud del Sprint, etc.

Todos los días

- Asegúrate de que el Scrum diario comienza y termina a su hora.
- Asegúrate de que todas las historias son añadidas o eliminadas de la Pila de Sprint cuando sea necesario para mantener el Sprint en fecha. Asegúrate de que se notifica al Dueño de Producto sobre estos cambios.
- Asegúrate de que el equipo mantiene actualizados la Pila de Producto y el burn-down.
- Asegúrate de que los problemas o impedimentos son solucionados o reportados al Cliente y/o el Resp. De Calidad de Software, Consejo... (a quien corresponda, vaya).

Final de Sprint

- Haz una demo de Sprint abierta.
- Todo el mundo debería ser notificado acerca de la demo uno o dos días antes.
- Haz una retrospectiva de Sprint con todo el equipo y el Cliente. Extrae el conocimiento de la retrospectiva y compártelo.
- Actualiza el documento de estadísticas de Sprint. Añade la velocidad real y los puntos clave de la retrospectiva.

14 Otros recursos

Herramientas Gestion Proyectos Scrum

- http://www.icescrum.org/ Probablemente la mejor herramienta para Scrum open source para tener en cuenta.
- http://www.danube.com/scrumworks/basic
- http://www.scrumdesk.com/ Gran herramienta para Scrum. Privativa.
- http://www.pivotaltracker.com/Herramienta gratuita online para gestión ágil de proyectos
- http://prjplanner.sourceforge.net/Herramienta para desarrollo ágil de software orientada para la auto-gestión de pequeños equipos de desarrollo.
- http://www.offshoreagile.com/resources/toolbox/herramienta fácil y gratuita para gestión y seguimiento de proyectos ágiles de StarSoft.
- http://trac.edgewall.org/ Plug-in's de Scrum para Trac.
- http://agiletrack.org/index.html Herramienta para planificación y seguimiento de proyectos, de interfaz sencillo. Para desarrollo de software en equipos reducidos con metodologías ágiles, especialmente eXtreme Programming. Open Source, licencia AFL.
- http://code.google.com/p/storyverse/
- http://studios.thoughtworks.com/mingle-project-intelligence
- http://www.agile42.com/cms/pages/download/
- http://www.visionproject.se/home.vm
- http://www.versionone.com/products_V1Team_Overview.asp
- http://www.codeplex.com/Tackle

Otras utilidades

- http://www.navegapolis.net/files/blog/evaluacion_riesgos_agilidad.xls
- Herramienta para evaluar el nivel de riesgo antes de empezar.
- Pinta con tus amiguitos http://www.imaginationcubed.com/
- http://www.navegapolis.net/files/cis/ficha_scrum.jpg Scrum: ficha sinóptica
- Nokia test: ¿estamos haciendo scrum? http://www.agilecollab.com/the-nokia-test

Linkografía

- http://agilemanifesto.org/
- http://www.mountaingoatsoftware.com/scrum
- http://www.navegapolis.net
- http://en.wikipedia.org/wiki/Scrum_(development)