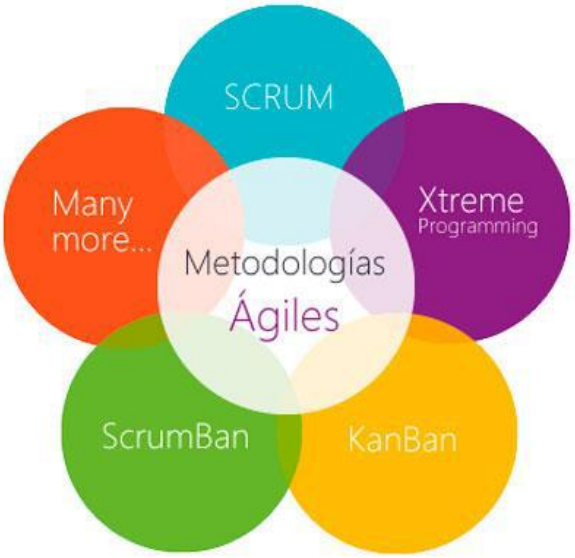
Trabajo práctico.

Metodologías de desarrollo. Investigar y desarrollar este tema en por lo menos en una página completa y menos de 3. tamaño A4

Scrum

Scrum es un enfoque ágil para desarrollar productos y servicios innovadores. Se encuentra dentro del grupo de metodologías ágiles.



Scrum se basa en la teoría de control de procesos empírica o empirismo. El empirismo asegura que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce. Scrum emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo.

Tres pilares soportan toda la implementación del control de procesos empírico: transparencia, inspección y adaptación.

Transparencia Los aspectos significativos del proceso deben ser visibles para aquellos que son responsables del resultado. La transparencia requiere que dichos aspectos sean definidos por un estándar común, de tal modo que los observadores compartan un entendimiento común de lo que se están viendo.

Inspección Los usuarios de Scrum deben inspeccionar frecuentemente los artefactos de Scrum y el progreso hacia un objetivo para detectar variaciones indeseadas. Su inspección no debe ser tan frecuente como para que interfiera en el trabajo. Las inspecciones son más beneficiosas cuando se realizan de forma diligente por inspectores expertos en el mismo lugar de trabajo.

Adaptación Si un inspector determina que uno o más aspectos de un proceso se desvían de límites aceptables y que el producto resultante será inaceptable, el proceso o el material que

está siendo procesado deben ajustarse. Dicho ajuste debe realizarse cuanto antes para minimizar desviaciones mayores.

Scrum prescribe cuatro eventos formales, contenidos dentro del Sprint, para la inspección y adaptación, tal y como se describen en la sección Eventos de Scrum del presente documento.

• Planificación del Sprint (Sprint Planning)

• Scrum Diario (Daily Scrum)

• Revisión del Sprint (Sprint Review)

• Retrospectiva del Sprint (Sprint Retrospective)

El marco propone muchas reuniones, por lo que es recomendable mantenerlas ágiles y apegarse a una agenda para evitar quitar tiempo a equipo de desarrollo y también evitar malestares (es algo probado que a la gente de sistemas le fastidian las frecuentes reuniones).

Visión de Scrum

Scrum (n): Un marco de trabajo por el cual las personas pueden abordar problemas complejos adaptativos, a la vez que entregar productos del máximo valor posible productiva y creativamente.

Scrum es:

• **Liviano**

• **Fácil de entender**

• **Difícil de dominar y aplicar en un ambiente de trabajo “real”**

Scrum es un marco de trabajo de procesos que ha sido usado para gestionar el trabajo en productos complejos desde principios de los años 90. Scrum no es un proceso, una técnica o método definitivo. En lugar de eso, es un marco de trabajo dentro del cual se pueden emplear varios procesos y técnicas.

El marco de trabajo Scrum consiste en los Equipos Scrum y sus roles, eventos, artefactos y reglas asociadas. Cada componente dentro del marco de trabajo sirve a un propósito específico y es esencial para el éxito de Scrum y para su uso.

Valores de Scrum

Compromiso: La definición general de "compromiso" es "el estado o la calidad de estar dedicado a una causa, actividad, etc.". Puede ser ilustrado por el entrenador de un equipo diciendo "No podría culpar a mis jugadores por compromiso " (aunque podrían haber perdido un juego).

Esto describe exactamente cómo se pretende el compromiso en Scrum. El compromiso se trata de dedicación y se aplica a las acciones y la intensidad del esfuerzo. No se trata del resultado final, ya que esto en sí mismo es a menudo incierto e impredecible para desafíos complejos en circunstancias complejas.

Sin embargo, hubo una interpretación errónea ampliamente difundida de la palabra compromiso en un contexto de Scrum. Esto se origina principalmente de la expectativa pasada del marco de Scrum que decía los equipos deben "comprometerse" con un Sprint. A través del lente del paradigma industrial tradicional esto se tradujo erróneamente en una expectativa de que todo el alcance seleccionado en la Planificación de Sprint se completará al final del Sprint, no importa. "Compromiso" fue erróneamente convertido en un contrato codificado.

En el complejo, creativo y altamente impredecible mundo del desarrollo de nuevos productos, una promesa de alcance exacto contra tiempo y presupuesto no es posible. Demasiadas variables que pueden influir en el resultado son desconocidas o pueden comportarse de manera impredecible.

Para reflejar mejor la intención original y conectarse más efectivamente con el empirismo, "compromiso" en el contexto del alcance de un Sprint fue reemplazado por "pronóstico".

Sin embargo, el compromiso sigue siendo y sigue siendo un valor central de Scrum: Los jugadores se comprometen con el equipo. Comprométete con la calidad. Comprométete a colaborar. Comprometerse a aprender. Comprométete a hacer lo mejor que pueda, todos los días nuevamente. Comprométete con el objetivo de Sprint. Comprometer para actuar como profesionales. Comprometerse a autoorganizarse. Comprométete con la excelencia. Comprométete con lo ágil valores y principios. Comprométase a crear versiones de trabajo del producto. Comprometerse a buscar mejoras Comprometerse con la definición de Hecho. Comprometerse con el marco de Scrum. Comprometer para centrarse en el valor. Comprométete a terminar el trabajo. Comprometerse a inspeccionar y adaptarse. Comprometerse a ser transparente. Comprométete a desafiar el statu quo.

Foco: Las responsabilidades equilibradas pero distintas de Scrum permiten a todos los jugadores concentrarse en su tarea.

El cronograma de Scrum alienta a los jugadores a centrarse en lo que es más importante ahora sin ser molestado por consideraciones de lo que podría tener la posibilidad de convertirse en importante en algún momento en el futuro. Se centran en lo que saben ahora.

Los jugadores se centran en lo inminente ya que el futuro es muy incierto y quieren aprender del presente para ganar experiencia para el trabajo futuro. Se centran en el trabajo necesario para obtener “cosas hechas”. Se centran en la cosa más simple que posiblemente podría funcionar.

El objetivo de Sprint se enfoca en un período de 4 semanas o menos. Dentro de ese período, el Scrum Diario ayuda a las personas a enfocarse en colaboración en el trabajo diario inmediato necesario para lograr el mejor progreso posible hacia la meta de Sprint.

Franqueza (Apertura): El empirismo de Scrum requiere transparencia, apertura y honestidad. Los jugadores-El inspector quieren verificar la situación actual para hacer adaptaciones razonables. Los jugadores son abiertos sobre su trabajo, progreso, aprendizajes y problemas. Pero también están abiertos para las personas y trabajar con personas; reconociendo que las personas son personas, y no "recursos", robots, engranajes o piezas de maquinaria reemplazables.

Los jugadores están abiertos a colaborar entre disciplinas, habilidades y descripciones de trabajo. Son abiertos a colaborar con las partes interesadas y el entorno más amplio. Abierto para compartir comentarios y aprendiendo el uno del otro.

Están abiertos al cambio a medida que cambian la organización y el mundo en el que operan; impredecible, inesperada y constantemente.

Respeto: El ecosistema de Scrum más amplio se nutre del respeto por las personas, su experiencia y sus antecedentes personales. Los miembros respetan la diversidad. Respetan las distintas opiniones. Respetan las habilidades, experiencia e ideas de los demás.

Respetan el entorno más amplio al no comportarse como una entidad aislada en el mundo. Respetan el hecho de que los clientes cambien de opinión. Muestran respeto por los patrocinadores al no crear o mantener funciones que nunca se usan y que aumentan el costo del producto. Muestran respeto al no malgastar dinero en cosas que no tienen valor, que no se aprecian o que pueden no ser implementadas o utilizadas de ninguna manera. Muestran respeto por los usuarios solucionando sus problemas.

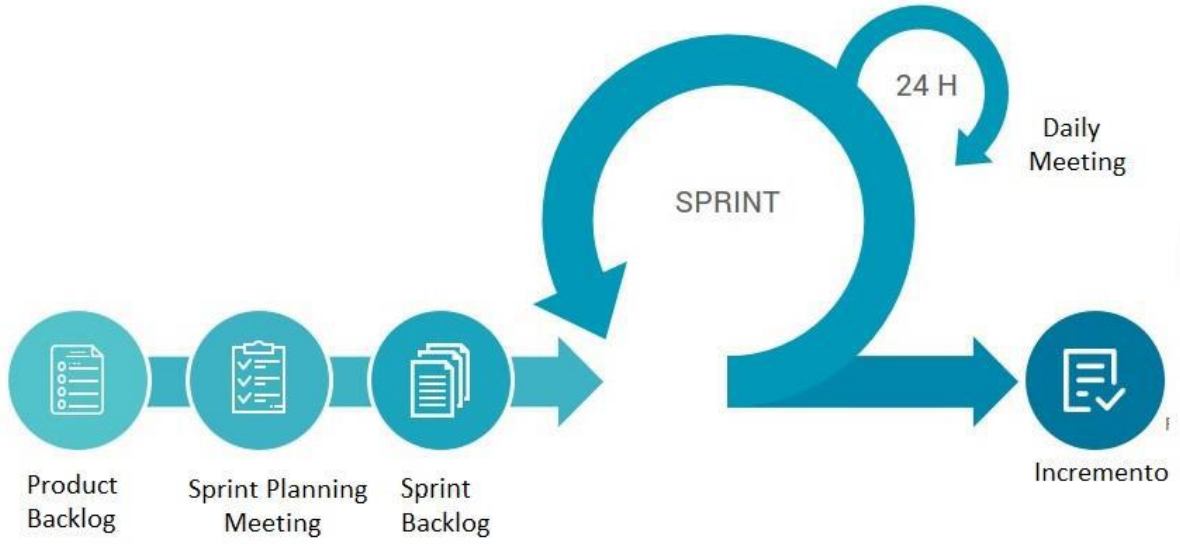
Todos los miembros respetan el framework de Scrum. Respetan las responsabilidades de Scrum.

Coraje: Los miembros muestran coraje al no desarrollar cosas que nadie quiere. Coraje al admitir que los requerimientos nunca serán perfectos y que ningún plan puede capturar toda la realidad y complejidad.

Muestran respeto al considerar el cambio como una fuente de inspiración e innovación. Coraje al no entregar versiones de producto no hechas. Coraje al compartir toda la información posible que pueda ayudar al equipo y a la organización. Coraje en admitir que nadie es perfecto. Coraje al cambiar de rumbo. Coraje al compartir riesgos y beneficios. Valor para dejar atrás las finas certezas del pasado.

Los jugadores demuestran coraje promoviendo Scrum y el empirismo para hacer frente a la complejidad. Muestran coraje apoyando los valores de Scrum. El coraje para tomar una decisión, actuar y progresar, no estancarse. Y aún más coraje para cambiar esa decisión.

Metodología de trabajo



Con un enfoque ágil, se comienza creando un *Product Backlog*: una lista priorizada de las características y otras capacidades necesarias para desarrollar un producto exitoso. Guiado por el backlog del producto, siempre se trabaja primero en los elementos más importantes o de mayor prioridad. Cuando se le acaben los recursos (como el tiempo), cualquier trabajo que no se haya completado deberá ser el de menor prioridad que el trabajo completado.

El trabajo en sí mismo se realiza en iteraciones breves de tiempo (*Sprints*), que generalmente varían de una semana a un mes calendario. Durante cada iteración, un equipo autoorganizado y multifuncional realiza todo el trabajo, como el diseño, la construcción y las pruebas, necesarios para producir características de trabajo completas que se puedan poner en producción. Normalmente, la cantidad de trabajo en la cartera de productos es mucho mayor que la que puede completar un equipo en una iteración de corta duración. Entonces, al comienzo de cada iteración, el equipo planifica qué subconjunto de alta prioridad del backlog del producto para crear en la próxima iteración.

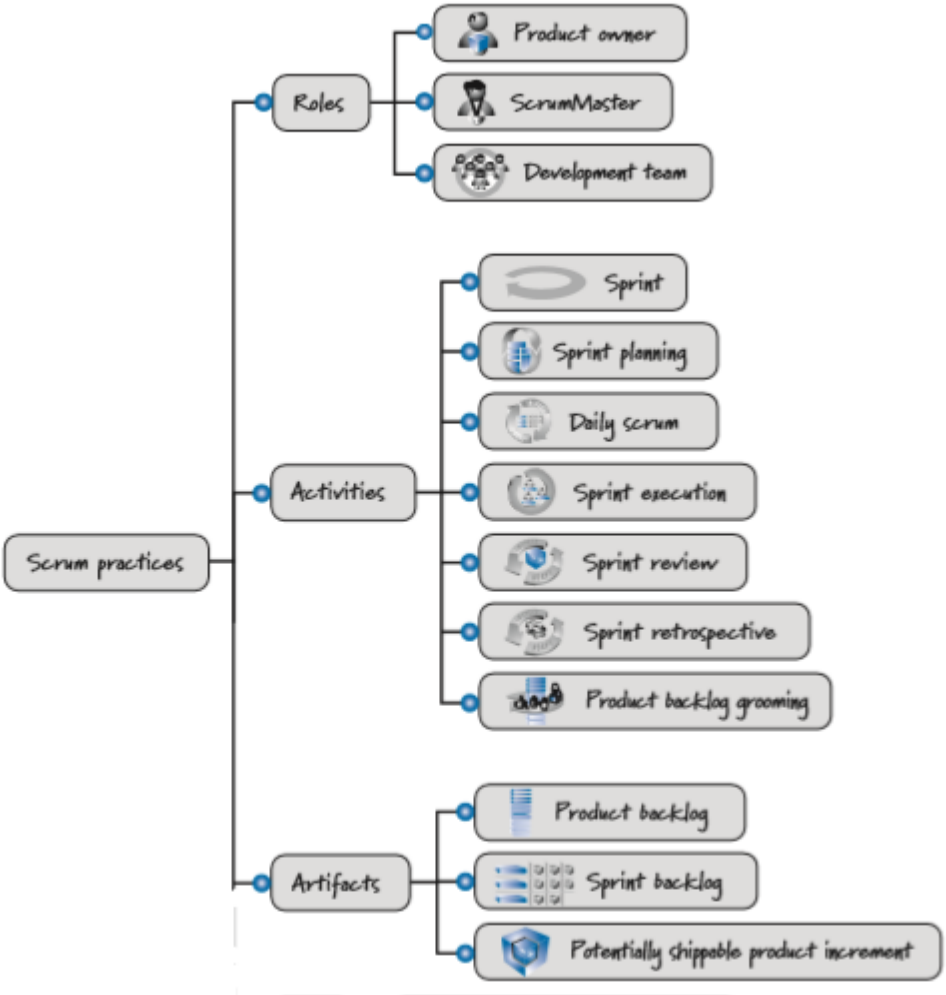
Al final de la iteración (Sprint), el equipo revisa las características completadas con las partes interesadas para obtener feedback. Según los comentarios, el Product Owner y el equipo pueden alterar tanto en lo que planean trabajar a continuación como en cómo el equipo planea hacer el trabajo. Por ejemplo, si las partes interesadas ven una feature (característica) completa y luego se dan cuenta de que otra feature que nunca consideraron también debe incluirse en el producto, el *Product Owner* puede simplemente crear un nuevo elemento que represente esa característica e insertarlo en el backlog, en el correcto orden, para trabajar en una iteración futura. Al final de cada iteración, el equipo debe tener un "producto potencialmente entregable" (o un incremento del producto), uno que se pueda liberar si es

apropiado. Si la publicación después de cada iteración no es apropiada, se puede lanzar un conjunto de características de varias iteraciones juntas.

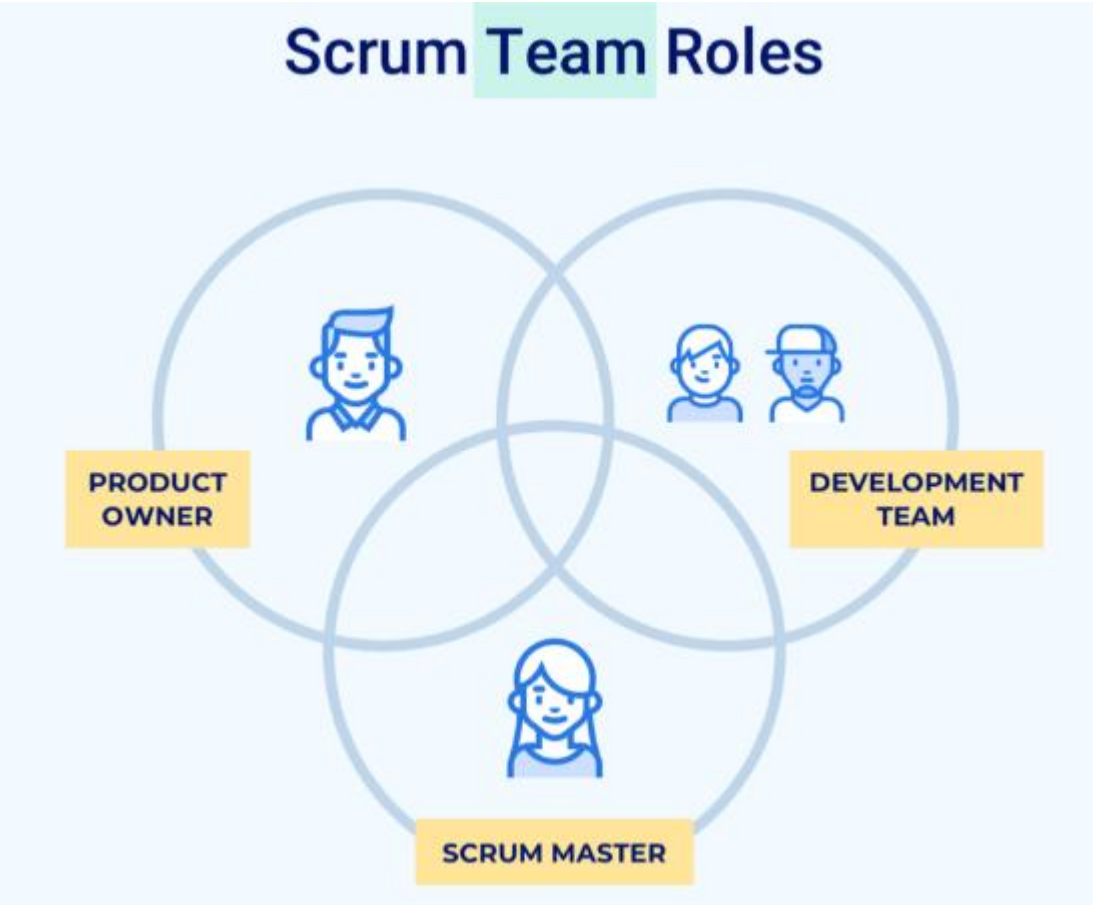
A medida que finaliza cada iteración, todo el proceso comienza de nuevo con la planificación de la próxima iteración.

Roles y eventos principales

SM; PO; ST , Sprint, Daily meeting, Retrospective meeting, Planning Meeting, Sprint Review meeting,



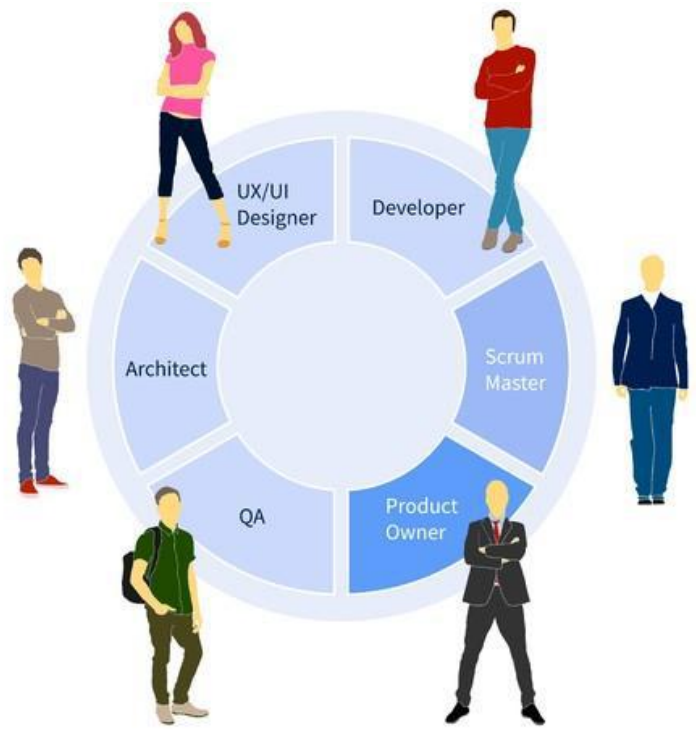
Equipo de Scrum



El Equipo Scrum consiste en un Dueño de Producto (Product Owner), el Equipo de Desarrollo (Development Team) y un Scrum Master. Los Equipos Scrum son autoorganizados y multifuncionales.

Los equipos autoorganizados eligen la mejor forma de llevar a cabo su trabajo y no son dirigidos por personas externas al equipo. Los equipos multifuncionales tienen todas las competencias necesarias para llevar a cabo el trabajo sin depender de otras personas que no son parte del equipo. El modelo de equipo en Scrum está diseñado para optimizar la flexibilidad, la creatividad y la productividad.

Los Equipos Scrum entregan productos de forma iterativa e incremental, maximizando las oportunidades de obtener retroalimentación. Las entregas incrementales de producto "Terminado" aseguran que siempre estará disponible una versión potencialmente útil y funcional del producto.



Product Owner

Es la cara del "negocio" en el equipo de Sistemas. Se recomienda que el PO sea un ex miembro del equipo del "negocio" con inclinación y conocimientos de IT. Pero existen casos en los que las empresas contratan personas externas con formación de PO (las cuales ya traen consigo conocimiento del mismo negocio de otras empresas o bien son formadas en el mismo).

El Dueño de Producto es el responsable de maximizar el valor del producto resultante del trabajo del Equipo de Desarrollo.

El Dueño de Producto es la única persona responsable de gestionar la Lista del Producto (Product Backlog). La gestión de la Lista del Producto incluye:

• Expresar claramente los elementos de la Lista del Producto; (en la práctica la especificación de los elementos –historias- es una tarea compartida por todo el equipo de *Discovery*.)

• Ordenar los elementos en la Lista del Producto para alcanzar los objetivos y misiones de la mejor manera posible; (Priorizarlos)

• Optimizar el valor del trabajo que el Equipo de Desarrollo realiza;

• Asegurar que la Lista del Producto es visible, transparente y clara para todos y que muestra aquello en lo que el equipo trabajará a continuación; y,

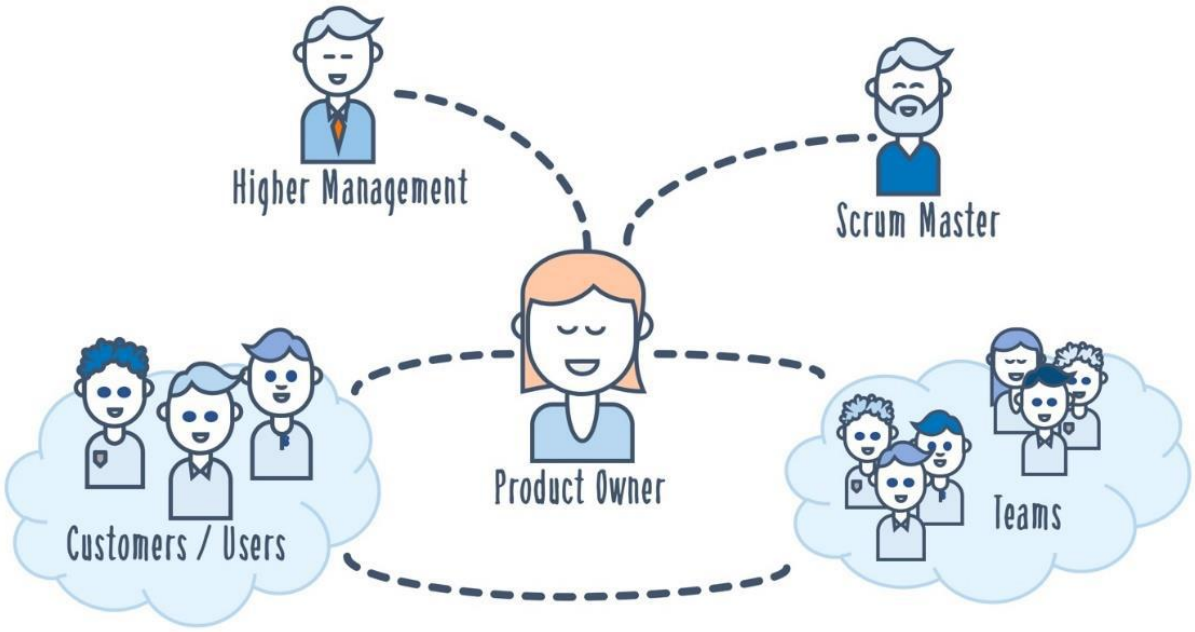
• Asegurar que el Equipo de Desarrollo entiende los elementos de la Lista del Producto al nivel necesario.

El Dueño de Producto podría hacer el trabajo anterior o delegarlo en el Equipo de Desarrollo. Sin embargo, en ambos casos el Dueño de Producto sigue siendo el responsable de dicho trabajo.

El Dueño de Producto es una única persona, no un comité. El Dueño de Producto podría representar los deseos de un comité en la Lista del Producto, pero aquellos que quieran cambiar la prioridad de un elemento de la Lista deben hacerlo a través del Dueño de Producto.

Para que el Dueño de Producto pueda hacer bien su trabajo, toda la organización debe respetar sus decisiones. Las decisiones del Dueño de Producto se reflejan en el contenido y en la priorización de la Lista del Producto. Nadie puede forzar al Equipo de Desarrollo a que trabaje con base en un conjunto diferente de requisitos.

Está disponible la certificación como Product Owner en varias plataformas on-line.



El PO tiene que hacer foco en al menos 2 direcciones: los Stakeholders (partes interesadas) y Equipo Scrum.

Por un lado, el propietario del producto debe comprender las necesidades y prioridades de las partes interesadas de la organización, los clientes y los usuarios lo suficientemente bien como para actuar como su voz. A este respecto, el propietario del producto actúa como gerente de producto, asegurando que se desarrolle la solución correcta. Por otro lado, el propietario del producto debe comunicar al equipo de desarrollo qué construir y el orden en que se debe construir. El propietario del producto también debe asegurarse de que se especifiquen los criterios para aceptar características y que las pruebas que verifican esos criterios se ejecuten posteriormente para determinar si las características están completas. El propietario del producto no escribe pruebas de nivel de detalle, pero se asegura de que las de

alto nivel se escriban para que el equipo pueda determinar cuándo considerará completa la función.

Responsabilidades del PO

Economía de Proyecto y Release

En cada release, el propietario del producto continuamente realiza compensaciones en alcance, fecha, presupuesto y calidad a medida que llega un flujo de información económicamente importante durante el desarrollo del producto. Las compensaciones hechas al comienzo de un lanzamiento pueden no ser apropiadas en presencia de nueva información que llega durante el lanzamiento. Por ejemplo, ¿qué pasa si varias semanas en un esfuerzo de desarrollo de fecha fija de seis meses reconocemos una oportunidad para aumentar los ingresos en un 50% si tomamos una semana adicional (4% de descuento en el cronograma) para agregar una característica recientemente identificada al lanzamiento? ¿Deberíamos cambiar el tiempo de una semana y el costo adicional por los ingresos adicionales? El propietario del producto supervisa esta decisión. En muchos casos, puede tomar la decisión unilateralmente. Otras veces, el propietario del producto puede recomendar una decisión, pero aún así trabajar con otros para asegurar su aporte (y en ocasiones aprobación) para ejecutar la decisión. Además, al final de cada sprint, el propietario del producto supervisa la decisión de financiar o no el próximo sprint. Si se logra un buen progreso hacia la meta de lanzamiento o si el próximo sprint está justificado económicamente, se financiará el próximo sprint. Si se avanza poco o la economía no respalda gastos adicionales, el esfuerzo podría cancelarse. Un propietario de producto satisfecho también puede supervisar la decisión de dejar de financiar el desarrollo adicional al final de un sprint si el producto está listo para enviarse y los gastos adicionales simplemente no están justificados. Por ejemplo, supongamos que planeamos un lanzamiento de diez sprints. Después del sprint 7, el propietario del producto revisa los elementos restantes de la cartera de pedidos de productos y concluye que el costo para crear esos elementos es mayor que el valor que entregan. El propietario del producto podría concluir que enviar el producto antes de tiempo es una decisión viable ya que recorta costos y los elementos no desarrollados del backlog no implican que el producto no sea aceptado por los Stakeholders. Esta flexibilidad para entregar temprano se habilita al garantizar que los elementos de mayor valor “en la parte superior” del backlog se trabajen primero y que el equipo complete el trabajo en cada sprint de acuerdo con una sólida Definición de Completo (DoD - Definition of Done).

Priorizar el Backlog

El propietario del producto es responsable de priorizar el backlog del producto. Cuando las condiciones económicas cambian, las prioridades en la cartera de productos probablemente también cambien. Por ejemplo, supongamos que al comienzo de un lanzamiento, el propietario del producto cree que una característica es valiosa para un gran porcentaje de los usuarios objetivo y el equipo cree que solo se requiere un esfuerzo modesto para crearla. Sin

embargo, después de algunos sprints, el equipo descubre que la función requerirá un gran esfuerzo para completarse y es valiosa solo para una fracción de los usuarios objetivo. Debido a que la relación costo / beneficio de esta función ha cambiado drásticamente, el propietario del producto debe priorizar la acumulación de productos para reflejar este conocimiento, tal vez eliminando los elementos de la cartera de productos asociados con la función.

Participar en la planificación

El propietario del producto es un participante clave en las actividades de planificación de portfolio (cartera), producto, lanzamiento y sprint. Durante la planificación de la cartera, el PO trabaja con las partes interesadas internas (quizás un comité de aprobación o gerencia) para posicionar el producto correctamente y determinar cuándo comenzar y finalizar el desarrollo del producto. Durante la planificación del producto, el propietario del producto trabaja con los Stakeholders para visualizar el producto. Durante la planificación del lanzamiento, el propietario del producto trabaja con los Stakeholders y el equipo para definir el contenido del próximo lanzamiento. Durante la planificación del sprint, el propietario del producto trabaja con el equipo de desarrollo para definir un objetivo de sprint. También proporciona información valiosa que permite al equipo de desarrollo seleccionar un conjunto de elementos de la cartera de productos que el equipo puede entregar de manera realista al final del sprint.

En la realidad el PO junto con un Delivery manager o responsable del equipo de IT le presentan y sugieren a los directivos y gerentes el camino a tomar. La decisión final siempre es de la “mesa chica”. En las planning meetings junto al Scrum Master y demás participantes priorizan los elementos del backlog dependiendo de las estimaciones de los mismos. Por ejemplo, si la capacidad del equipo solo admite 9 puntos de historia por sprint y tenemos 4 historias priorizadas para trabajar en el sprint; 2 de 4 hp y 2 de 1 hp (un total de 10 hp), es el PO quien decide cual dejar afuera del Sprint que se está planificando), todo esto está ajustado a la opinión de todos los miembros de equipo (más técnicos que el PO).

Refinar el backlog (backlog grooming)

El propietario del producto supervisa la preparación del trabajo atrasado del producto, que incluye crear y refinar, estimar y priorizar los elementos del trabajo atrasado del producto. El propietario del producto no realiza personalmente todo el trabajo de refinamiento. Por ejemplo, es posible que no escriba todos los elementos de la cartera de pedidos del producto; otros pueden contribuir con ellos. El propietario del producto tampoco estima los elementos (el equipo de desarrollo lo hace), pero está disponible para preguntas y aclaraciones durante la estimación. Sin embargo, el propietario del producto es en última instancia responsable de asegurarse de que las actividades de refinamiento tengan lugar de una manera que promueva el flujo fluido del valor entregado.

Define los criterios de aceptación y verifica que se cumplan

El propietario del producto es responsable de definir los criterios de aceptación para cada elemento del backlog. Estas son las condiciones bajo las cuales el propietario del producto estaría satisfecho de que se hayan cumplido los requisitos funcionales y no funcionales. El propietario del producto también puede escribir pruebas de aceptación correspondientes a los criterios de aceptación, o puede solicitar la asistencia de expertos en la materia o miembros del equipo de desarrollo. En cualquier caso, el propietario del producto debe asegurarse de que estos criterios de aceptación (y con frecuencia pruebas de aceptación específicas) se creen antes de considerar un artículo en una reunión de planificación de sprint. Sin ellos, el equipo tendría una comprensión incompleta del artículo y no estaría listo para incluirlo en un sprint. El propietario del producto es el responsable final de confirmar que se cumplen los criterios de aceptación. Una vez más, el propietario del producto puede optar por realizar pruebas de aceptación él mismo o puede solicitar la asistencia de usuarios expertos para ayudar a confirmar que el elemento del producto atrasado cumple con las condiciones de satisfacción. El equipo podría ayudar a crear una infraestructura de prueba que permita al propietario del producto o analistas funcionales y testers ejecutar estas pruebas de manera más eficiente, pero el propietario del producto debe ser el juez final para determinar si un artículo cumple con las expectativas.

Colaborar con el equipo de desarrollo El PO debe colaborar estrechamente con el equipo de desarrollo con frecuencia. El propietario del producto es un rol comprometido, comprometido y cotidiano. Muchas organizaciones que recién comienzan a adoptar Scrum no logran fomentar el compromiso adecuado del propietario del producto con el equipo de desarrollo, retrasando la retroalimentación esencial y reduciendo sustancialmente el valor de esa retroalimentación cuando ocurre. Esta falta de participación también puede ocurrir cuando las personas nuevas en el rol de propietario del producto asumen que su nivel de participación al usar Scrum debe parecerse a su participación durante el desarrollo basado en fases. Usando el desarrollo tradicional escalonado, los clientes tienen una participación inicial considerable para ayudar a definir el conjunto completo de requisitos. Una vez que el esfuerzo pasa a fases más técnicas (como diseño, codificación y ciertos tipos de pruebas), los clientes "ya no son necesarios". Como tal, su nivel de compromiso es bastante bajo o inexistente durante la mayoría del esfuerzo. De hecho, durante el desarrollo tradicional, los clientes no vuelven a ingresar al proceso hasta casi el final, cuando se les exige que realicen pruebas de aceptación del usuario sobre lo que se construyó. Lo que los clientes suelen descubrir en este momento es que lo que se ha construido no es exactamente lo que querían. Para empeorar las cosas, generalmente es demasiado tarde o demasiado costoso para que hagan cambios, al menos en esta versión. Los clientes que llegan esperando estar encantados se van sorprendidos, frustrados y decepcionados. Esto es cuando la punta del dedo se mueve a alta velocidad. El cliente afirma: "Si ustedes hubieran leído mi documento de requisitos con más cuidado, habrían construido lo que realmente quería", y el equipo de desarrollo responde: "Bueno, si hubieran escrito su documento más claramente, habríamos creado algo diferente". . ¡Construimos lo que pediste! ”Con Scrum, creamos una función a la vez, no un escalón a la

vez. Esto significa que realizamos todas las actividades para crear una característica particular (diseño, código, integración, prueba) durante un sprint. Por lo tanto, un alto nivel constante de compromiso por parte del propietario del producto es esencial. Con una interacción tan estrecha en iteraciones breves y de tiempo limitado, hay muchas menos posibilidades de que el propietario del producto y el equipo de desarrollo se desconecten.

Colaborar con los Stakeholders

El propietario del producto es la voz única de toda la comunidad de partes interesadas, interna y externa. Las partes interesadas internas pueden incluir propietarios de sistemas comerciales, gestión ejecutiva, gestión de programas, marketing y ventas. Las partes interesadas externas pueden incluir clientes, usuarios, socios, organismos reguladores y otros. El product owner debe trabajar estrechamente con toda la comunidad de partes interesadas para recopilar aportes y sintetizar una visión coherente para guiar el desarrollo del producto. Si el propietario del producto se abruma y se extiende demasiado, le será difícil colaborar tanto con el equipo de desarrollo como con las partes interesadas en el nivel requerido. En algunas circunstancias, la carga de trabajo puede ser más de lo que una persona puede realizar razonablemente, en cuyo caso el propietario del producto puede solicitar la asistencia de otros para ayudar a cumplir con las responsabilidades del rol.

Skills del PO

- **People Skills**
- **Responsabilidad**
- **Toma de decisiones**
- **Priorización**
- **Disponibilidad**

Product Owner combinado con otros roles

Si la capacidad lo permite, la misma persona puede actuar como PO para más de un equipo Scrum. Por lo general, es más fácil para esta persona ser el propietario del producto de varios equipos en el mismo esfuerzo de desarrollo, porque el trabajo de estos equipos probablemente estará muy relacionado. Aunque hay momentos en que el mismo individuo puede ser el propietario del producto y un miembro del equipo de desarrollo, se considera una mala idea que la misma persona sea tanto el propietario del producto como el ScrumMaster en el mismo equipo Scrum. Estos dos roles se compensan entre sí; tener a una persona desempeñando ambos roles crea un conflicto de intereses que deberíamos tratar de evitar.

Equipo de Product Owners

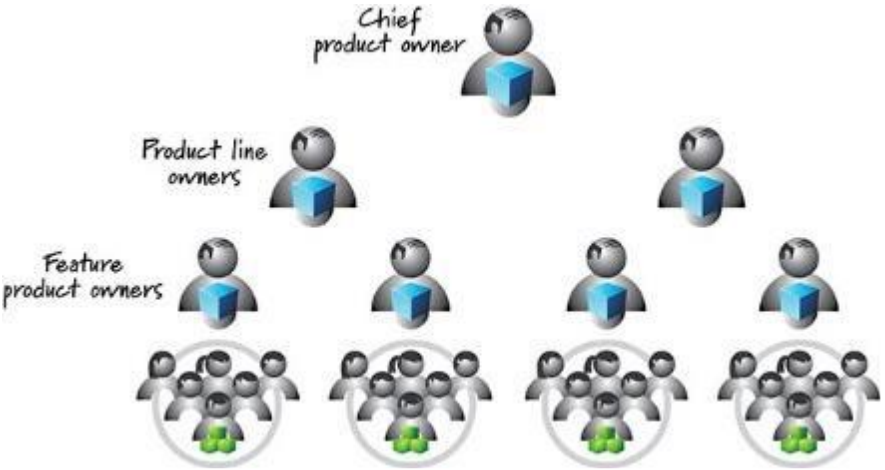
Cada equipo Scrum debe tener una sola persona que se identifique como el propietario del producto y sea la única persona facultada y responsable de cumplir con las responsabilidades del propietario del producto para ese equipo Scrum. ¿Deberíamos permitir que un equipo de personas desempeñe el rol de propietario del producto? Si por equipo nos referimos a un grupo de personas con responsabilidad y toma de decisiones compartidas, definitivamente no. Para aplicar Scrum adecuadamente, necesitamos que un individuo sea el propietario del producto, que tome decisiones y actúe como la voz única de las comunidades de partes interesadas para el equipo Scrum. Dicho esto, algunas organizaciones pueden formar lo que llaman un "equipo de Product Owners" porque reconocen que, en sus circunstancias, el propietario del producto no puede hacer el trabajo sin un grupo selecto de personas para proporcionar información y orientación. En otras compañías, la carga de trabajo de ser propietario de un producto puede ser mayor que la que cualquier persona de tiempo completo pueda realizar razonablemente. En esos casos, el propietario del producto delega algunas responsabilidades del propietario del producto a otras personas. *Formar un equipo propietario del producto en cualquiera de estas circunstancias es aceptable siempre y cuando haya una persona en el equipo que tome la decisión final* y siempre que tener un equipo propietario del producto no se degrade en el diseño por comité, y cada decisión necesita aprobación de otras ocho personas. Tenga cuidado al crear equipos de propietarios de productos. Los propietarios de productos que no están debidamente capacitados para ser el punto central de liderazgo del producto no necesitan un comité, necesitan un rol diferente. Del mismo modo, los propietarios de productos que están demasiado ocupados para cumplir con sus responsabilidades podrían no necesitar un equipo. Quizás el verdadero problema es que la organización ha optado por comenzar demasiados esfuerzos de desarrollo al mismo tiempo, o hay muy pocos propietarios de productos para cubrir los productos necesarios.

Alternativamente, quizás el producto que estamos construyendo es demasiado grande y debería dividirse en una serie de piezas más pequeñas con lanzamientos más frecuentes. Con piezas pequeñas, una sola persona podría cumplir más fácilmente el rol de propietario del producto. Además, si hemos estructurado mal nuestros equipos, o hemos concebido mal las estructuras del backlog, un propietario de un solo producto podría tener dificultades para hacer su trabajo. Asegúrese de que los equipos de propietarios de sus productos sean realmente necesarios y no solo enmascaren un problema subyacente; de lo contrario, la situación se volverá más complicada y pondrá en peligro su resultado general.

CPO (Chief Product Owner) - Produc Owner Jefe

Otra situación en la que se crea con frecuencia un equipo propietario del producto es en productos muy grandes. Anteriormente dijimos que una sola persona podría ser el propietario del producto para un par de equipos Scrum, pero ¿qué pasa con los escenarios que involucran a muchos equipos? Por ejemplo, una organización que tenía un esfuerzo de desarrollo que involucraba a más de 2,500 personas. Con un tamaño de equipo promedio de menos de 10 personas, la organización tenía más de 250 equipos en el esfuerzo. Una persona no puede ser

el propietario del producto para 250 equipos. De hecho, una persona no puede ser el propietario del producto comprometido y cotidiano de más de unos pocos equipos. En casos como estos, el rol del propietario del producto debe escalar jerárquicamente como se muestra en la Figura. En última instancia, la persona etiquetada como propietario principal del producto (CPO) es el propietario del producto para *todo* el producto. Sin embargo, el propietario principal del producto tiene un equipo de propietarios de productos para garantizar que el rol de PO se cumpla correctamente en cada nivel inferior de la jerarquía. Si elige utilizar este enfoque, asegúrese de que los POs del equipo individual sigan facultados para tomar la gran mayoría de las decisiones en sus niveles, en lugar de tener que pasar esas decisiones a la jerarquía que se tomará en los niveles superiores.



El equipo de desarrollo (Development Team)

El Equipo de Desarrollo consiste en los profesionales que realizan el trabajo de entregar un Incremento de producto "Terminado" que potencialmente se pueda poner en producción al final de cada Sprint. Un Incremento "Terminado" es obligatorio en la Revisión del Sprint. Solo los miembros del Equipo de Desarrollo participan en la creación del Incremento.

La organización es la encargada de estructurar y empoderar a los Equipos de Desarrollo para que estos organicen y gestionen su propio trabajo. La sinergia resultante optimiza la eficiencia y efectividad del Equipo de Desarrollo.

Los Equipos de Desarrollo tienen las siguientes características:

• Son auto organizados. Nadie (ni siquiera el Scrum Master) indica al Equipo de Desarrollo cómo convertir elementos de la Lista del Producto en Incrementos de funcionalidad potencialmente desplegables;

• Los Equipos de Desarrollo son multifuncionales, esto es, como equipo cuentan con todas las habilidades necesarias para crear un Incremento de producto;

• Scrum no reconoce títulos para los miembros de un Equipo de Desarrollo independientemente del trabajo que realice cada persona; *Esto en la realidad no siempre se da, ya que por lo general los equipos cuentan con especialistas en cada ámbito (aunque cada*

miembro del equipo tiene algunos conocimientos de las facultades de los demás y puede cumplir con alguna de sus tareas).

• Scrum no reconoce subequipos en los equipos de desarrollo, no importan los dominios que requieran tenerse en cuenta, como pruebas, arquitectura, operaciones o análisis de negocio; y,

• Los Miembros individuales del Equipo de Desarrollo pueden tener habilidades especializadas y áreas en las que estén más enfocados, pero la responsabilidad recae en el Equipo de Desarrollo como un todo.

El tamaño del equipo de desarrollo

El tamaño óptimo del Equipo de Desarrollo es lo suficientemente pequeño como para permanecer ágil y lo suficientemente grande como para completar una cantidad de trabajo significativa. Tener menos de tres miembros en el Equipo de Desarrollo reduce la interacción y resulta en ganancias de productividad más pequeñas. Los Equipos de Desarrollo más pequeños podrían encontrar limitaciones en cuanto a las habilidades necesarias durante un Sprint, haciendo que el Equipo de Desarrollo no pudiese entregar un Incremento que potencialmente se pueda poner en producción. Tener más de nueve miembros en el equipo requiere demasiada coordinación. Los grandes Equipos de Desarrollo generan demasiada complejidad como para que un proceso empírico les sea de utilidad. Los roles de Dueño de Producto y Scrum Master no cuentan en el cálculo del tamaño del equipo a menos que también estén contribuyendo a trabajar en la Lista de Pendientes de Sprint (Sprint Backlog).

Cuando un equipo tiene más de 8 integrantes se recomienda dividirlo en 2 Scrums para facilitar la planificación y organización del mismo. Algunos miembros pueden ser roles cross a los 2 Scrums.

Responsabilidades del equipo de desarrollo

Colaborar con los Stakeholders

Durante la ejecución del sprint, los miembros del equipo de desarrollo realizan el trabajo práctico y creativo de diseñar, construir, integrar y probar los elementos de la cartera de productos en incrementos de la funcionalidad potencialmente transportable. Para hacer esto, se autoorganizan y colectivamente deciden cómo planificar, administrar, llevar a cabo y comunicar el trabajo. El equipo de desarrollo pasa la mayor parte de su tiempo realizando la ejecución del sprint.

Inspeccionar y adaptarse día a día

Se espera que cada miembro del equipo de desarrollo participe en cada scrum diario, durante el cual los miembros del equipo inspeccionan colectivamente el progreso hacia la meta del sprint y adaptan el plan para el trabajo del día actual. Si algunos miembros del equipo no

participan, el equipo puede perder partes del panorama general y puede no lograr su objetivo de sprint.

Refinar el backlog

Parte de cada sprint debe gastarse preparándose para el siguiente. Una gran parte de ese trabajo se centra en la preparación del trabajo atrasado del producto, que incluye crear y refinar, estimar y priorizar los elementos del backlog. El equipo de desarrollo debe asignar hasta el 10% de su capacidad disponible en cada sprint para ayudar al propietario del producto con estas actividades.

Planear el Sprint

Al comienzo de cada sprint, el equipo de desarrollo participa en la planificación del sprint. En colaboración con el propietario del producto y con la facilitación de ScrumMaster, el equipo de desarrollo ayuda a establecer el objetivo para el próximo sprint. Luego, el equipo determina qué subconjunto de alta prioridad de los elementos del backlog se desarrollarán para lograr ese objetivo. Para un sprint de dos semanas, la planificación del sprint generalmente dura aproximadamente medio día. Un sprint de cuatro semanas puede necesitar hasta un día completo para la planificación del sprint. Tenga en cuenta que la planificación se realiza de forma iterativa. En lugar de centrarse en un plan muy grande, incierto y demasiado detallado al comienzo de un esfuerzo de desarrollo, el equipo hace una serie de planes más pequeños, más seguros y más detallados justo a tiempo al comienzo de cada sprint.

Inspeccionar y adaptar el producto y el proceso

Al final de cada sprint, el equipo de desarrollo participa en las dos actividades de inspección y adaptación: revisión de sprint y retrospectiva de sprint. La revisión del sprint es donde el equipo de desarrollo, el propietario del producto, ScrumMaster, las partes interesadas, los patrocinadores, los clientes y los miembros interesados de otros equipos revisan las características recién completadas del sprint actual y discuten cómo avanzar mejor. La retrospectiva del sprint es donde el equipo Scrum inspecciona y adapta su proceso Scrum y sus prácticas técnicas para mejorar la forma en que utiliza Scrum para entregar valor comercial.

Características/Skills

Autoorganizado

Los miembros del equipo se autoorganizan para determinar la mejor manera de lograr el objetivo del sprint. No hay un gerente de proyecto u otro gerente que le diga al equipo cómo hacer su trabajo (y un ScrumMaster nunca debería presumir de esto, solo puede ayudar a priorizar las tareas de un miembro del equipo cuando esta sobrecargado de tareas y bugs). La autoorganización es una propiedad emergente de abajo hacia arriba del sistema: no existe una fuerza dominante externa que aplique la gestión tradicional de arriba hacia abajo, de

comando y control. *En la realidad muchos desarrolladores jr necesitan coaching y guía en términos de organización, hasta que encuentran su ritmo y consiguen manejar prioridades.*

Multifuncionalmente diverso y suficiente

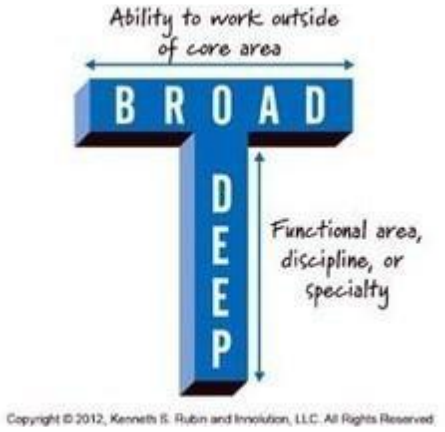
Los miembros del equipo de desarrollo deberían ser diversos y de roles cruzados; colectivamente, deben poseer el conjunto de habilidades necesario y suficiente para hacer el trabajo. Un equipo bien formado puede eliminar un elemento de la cartera de pedidos del producto y producir una función de trabajo de buena calidad que cumpla con la definición de hecho del equipo Scrum. Los equipos compuestos únicamente por personas con las mismas habilidades (equipos de silos tradicionales) pueden, como máximo, hacer parte del trabajo. Como resultado, los equipos de silos terminan entregando productos de trabajo a otros equipos de silos. Por ejemplo, el equipo de desarrollo entrega el código al equipo de prueba, o el equipo de IU entrega los diseños de pantalla al equipo de lógica de negocios. Las transferencias representan una excelente oportunidad para la falta de comunicación y los errores costosos. Tener equipos diversos minimiza el número de traspasos. Y la creación de equipos diversos no nos impide tener varios miembros del equipo que puedan ser altamente calificados en la misma disciplina, como el desarrollo o las pruebas de Java o C ++. Los equipos con diversas funciones cruzadas también aportan múltiples perspectivas, lo que lleva a mejores resultados. Un equipo con diversas funciones cruzadas tiene miembros de diferentes orígenes. Cada miembro del equipo trae un conjunto de herramientas cognitivas para la resolución de problemas; Estas herramientas pueden involucrar diferentes interpretaciones (de los mismos datos), diferentes estrategias (o heurísticas) para resolver problemas, diferentes modelos mentales de cómo funcionan las cosas y diferentes preferencias tanto para los enfoques como para las soluciones. Este tipo de diversidad generalmente conduce a mejores resultados en términos de soluciones más rápidas, entregables de mayor calidad y mayor innovación, todo lo cual se traduce en un mayor valor económico. También debemos luchar por la diversidad del equipo al tener una buena combinación de personal de nivel senior y junior en el mismo equipo. Demasiadas personas de alto nivel pueden causar turbulencias innecesarias, similar a tener demasiados cocineros en la cocina. Sin embargo, hay demasiadas personas jóvenes y el equipo podría no estar lo suficientemente capacitado para hacer el trabajo. Una buena combinación promueve un ambiente de aprendizaje saludable y colaborativo. Un equipo Scrum ideal cuenta con varios full stack developers de diferentes edades y culturas; esto es raramente posible en la realidad por lo tanto el Scrum debe adaptarse a cada situación, como ser la pérdida de un miembro de equipo, la falta de conocimiento de un área, etc.

T-Shapped Skills

Los equipos de desarrollo flexibles están compuestos por miembros con habilidades en forma de T. Las habilidades en forma de T significan que un miembro del equipo (por ejemplo, Sue) tiene habilidades profundas en su área funcional, disciplina o especialidad preferida. Por ejemplo, Sue es una gran diseñadora de experiencia de usuario (UX), esa es su especialidad y

prefiere trabajar en eso. Sin embargo, Sue también puede trabajar fuera de su área de especialidad principal, haciendo algunas pruebas y documentación. No es tan buena como tester o documentalista como aquellos que se especializan en esas áreas, pero puede ayudar con las pruebas o la documentación si es donde el equipo está experimentando un cuello de botella y necesita enjambrar a las personas para hacer el trabajo. A este respecto, Sue tiene amplias habilidades que le permiten trabajar fuera de su área central. No es realista creer que cada persona en un equipo pueda trabajar en cada tarea. Esa es una meta elevada para tener. Por ejemplo, en dominios con especialización intensa, como el desarrollo de videojuegos, donde un equipo podría tener un artista, animador, ingeniero de audio, programador de inteligencia artificial (AI) y probador, no es razonable suponer que todos pueden hacer todo el trabajo. Si estuviera en un equipo desarrollando un videojuego, podría trabajar en la IA y hacer algunas pruebas, pero no podría trabajar en el diseño de arte (¡y no querrías que lo hiciera!). Sin embargo, podría ayudar a los artistas con trabajos de diseño no artístico, como el uso de Photoshop para convertir formatos de archivo o crear guiones para aplicar operaciones en varios archivos. Los gerentes deben enfocarse en formar equipos que tengan el mejor conjunto de habilidades en forma de T que sean posibles con el personal disponible. Sin embargo, puede que no sea posible obtener exactamente el conjunto de habilidades de equipo deseado desde el principio, por lo que el conjunto de habilidades deseado podría evolucionar con el tiempo a medida que evolucionen las necesidades del esfuerzo de desarrollo de productos. Por lo tanto, es fundamental tener un entorno en el que las personas estén constantemente aprendiendo y agregando a sus conjuntos de habilidades, ya sea que incluyan conocimiento de dominio, conocimiento técnico, habilidades de pensamiento u otras capacidades. La gerencia necesita apoyar a los miembros del equipo con tiempo para aprender y experimentar. ¿Está bien tener especialistas puros en el equipo? Tomemos nuestro ejemplo anterior de Sue y supongamos que es una gran diseñadora de experiencia de usuario, pero eso es todo lo que puede hacer. Y porque tenemos capacidad de tan pocos diseñadores de UX, realmente no queremos que haga nada más que un trabajo crítico de diseño de UX. Necesitamos sus habilidades en el equipo, pero solo podremos ocupar aproximadamente el 10% de su tiempo con trabajo relacionado con el equipo. En estos casos, una solución obvia es dividir el tiempo de Sue entre varios equipos. Sin embargo, debemos ser prácticos. Sue estaría demasiado fracturada si dividiera su tiempo en incrementos del 10% para muchos equipos al mismo tiempo. Pronto se convertiría en un cuello de botella. Nuestro objetivo no debería ser mantener a las personas como Sue 100% utilizadas. En cambio, deberíamos estar más preocupados por el trabajo inactivo que ocurre cuando dependemos demasiado de un recurso sobreutilizado. Por lo tanto, podríamos asignar a Sue como especialista a una cantidad razonable de productos, pero no a tantos como para que ella sea la causa del cuello de botella. Alternativamente, debido a que nuestro objetivo es lograr un buen flujo con los miembros del equipo que tienen amplias habilidades en forma de T, debemos alentar a Sue a ayudar a otros miembros del equipo a adquirir conocimientos razonables de diseño UX para que ya no necesitemos depender tanto de especialistas. Para resumir, entonces, nuestro objetivo es formar un equipo con miembros que tengan las habilidades adecuadas para cubrir las áreas

de especialidad básicas y, en conjunto, tengan cierta superposición de habilidades para proporcionar flexibilidad adicional. Para cumplir con este objetivo, muchos miembros del equipo deben tener habilidades en forma de T, pero aún podríamos tener algunos especialistas en la mezcla.



Actitud de mosquetero

Los miembros del equipo de desarrollo (¡y el equipo de Scrum en su conjunto!) Deben tener la misma actitud que los Tres Mosqueteros: "Todos para uno y uno para todos". Esta actitud de Mosqueteros refuerza el punto de que los miembros del equipo poseen la responsabilidad colectiva de hacer el trabajo. Ganan como equipo o fracasan como equipo. En un equipo Scrum que funciona bien, nunca esperaría que alguien dijera: “Hice mi parte. No cumpliste tu parte. Por lo tanto, fallamos”. Los miembros del equipo deben apreciar que deben trabajar juntos para cumplir con sus compromisos, porque si fallan, al final será el problema de todos. Tener miembros del equipo con una actitud de mosquetero es fundamental para lograr el éxito compartido. Tener miembros del equipo con habilidades en forma de T fomenta esta actitud y la hace práctica porque las personas son capaces de trabajar en más de un tipo de tarea. En estos equipos no espero escuchar a una persona que sea capaz de hacer el trabajo decir: "Ese no es mi trabajo". Sin embargo, debido a que no siempre es posible que una persona haga todos los trabajos, es posible que escuche a alguien decir: "No soy capaz de hacer ese trabajo". En este caso, el equipo podría elegir tener a la persona sin las habilidades de aprendiz con una persona que tenga las habilidades para que en el futuro el equipo tenga mayores capacidades agregadas. Incluso si las limitaciones de habilidades impiden que las personas trabajen de manera multifuncional, los miembros del equipo aún pueden organizar su trabajo para garantizar un buen flujo a través del sprint para que ningún miembro del equipo esté sobrecargado. Por ejemplo, mantener todo el trabajo de prueba hasta el final del sprint para que el "probador" pueda hacer el trabajo es sin duda una receta para el fracaso. Por lo tanto, con una actitud de mosquetero, nadie está simplemente "a lo largo del viaje". Cada miembro del equipo es responsable de asegurarse de que esté completamente comprometida en todo momento. Con frecuencia, esto significará hablar y participar en actividades fuera de la especialidad para agregar a la diversidad de la discusión. Por ejemplo, aunque la especialidad de un miembro del equipo podría estar probando, si cree que hay un

problema en el diseño que el equipo está elaborando para una característica dada, es su deber hablar, en lugar de encogerse de hombros y decir: "No es mi trabajo; ellos saben mejor que yo de todos modos ".

Comunicación de alto ancho de banda

Los miembros del equipo de desarrollo deben comunicarse entre sí, así como con el propietario del producto y el Scrum Master, de manera de gran ancho de banda, donde se intercambia información valiosa de manera rápida y eficiente con una sobrecarga mínima. Las comunicaciones de gran ancho de banda aumentan tanto la frecuencia como la calidad del intercambio de información. Como resultado, el equipo Scrum tiene oportunidades más frecuentes para inspeccionar y adaptarse, lo que lleva a una toma de decisiones mejor y más rápida. Debido a que el valor económico de la información es sensible al tiempo, acelerar la tasa de intercambio de información permite al equipo maximizar su valor. Al explotar rápidamente las oportunidades emergentes y reconocer las situaciones de derroche, el equipo puede evitar gastar más recursos yendo por el camino equivocado. Hay varias formas en que un equipo puede lograr comunicaciones de gran ancho de banda. El Manifiesto Ágil (Beck et al. 2001) establece que la comunicación cara a cara es un enfoque preferido. Ciertamente, los miembros del equipo que están físicamente separados o utilizan principalmente la comunicación no interactiva (como documentos) están en desventaja con respecto a los miembros del equipo ubicados en una colaboración cara a cara en tiempo real. Siempre que sea posible, prefiero que los miembros de mi equipo sean ubicados. Sin embargo, muchas organizaciones, por diversas razones comerciales, han creado equipos distribuidos, por lo que la colocación puede no ser siempre posible o práctica. He trabajado con muchos equipos distribuidos que han logrado los beneficios de las comunicaciones de gran ancho de banda, por lo que estar cara a cara no es la única forma de lograr el objetivo, pero es un excelente lugar para comenzar si las condiciones comerciales lo permiten. Para los equipos distribuidos, un cierto nivel de soporte tecnológico puede ayudar a mejorar el ancho de banda de la comunicación. He trabajado con organizaciones donde los miembros del equipo se distribuyeron ampliamente. Mediante el uso de algunos equipos de teleconferencia bastante impresionantes, participé en discusiones que sentían que todos estaban colocados. ¿Fue tan bueno como ser colocado? No. Pero la tecnología contribuyó en gran medida a mejorar el ancho de banda de comunicación entre los miembros del equipo. Tener equipos compuestos por miembros de equipos multifuncionales es un paso crítico para lograr comunicaciones de gran ancho de banda. Dichos equipos tienen canales de comunicación más simplificados simplemente porque tienen fácil acceso a las personas necesarias para hacer el trabajo. Además, estos equipos de diversas funciones cruzadas tienen muchas menos probabilidades de tener transferencias formales (que generalmente toman la forma de documentos escritos) de un equipo a otro. Con todos en el mismo equipo, la frecuencia y la formalidad de las transferencias se reducen, lo que mejora la velocidad de comunicación. También debemos reducir el tiempo dedicado a las ceremonias en las que los miembros del equipo realizan un proceso que agrega poco o ningún valor. Por ejemplo, si los miembros del equipo tienen que

pasar por tres niveles antes de poder hablar con un cliente o usuario real, la ceremonia de "hablar con un cliente" es probablemente un impedimento serio para las comunicaciones de gran ancho de banda. Tener que crear documentos de poco o ningún valor o requerir procedimientos de aprobación y aprobación largos y potencialmente innecesarios reduce el ancho de banda. Necesitamos identificar y eliminar estos impedimentos para mejorar el rendimiento general de la comunicación del equipo. Finalmente, tener equipos pequeños también mejora el ancho de banda. Los canales de comunicación dentro de un equipo no se escalan linealmente con el número de miembros del equipo, sino que aumentan al cuadrado del número de personas en el equipo de acuerdo con la fórmula N (N - 1) / 2. Entonces, si hay 5 personas en el equipo, hay 10 canales de comunicación. Si hay 10 personas en el equipo, hay 45 canales de comunicación. Más personas significa más sobrecarga de comunicación y, por lo tanto, menor ancho de banda.

Comunicación transparente

Además de ser de gran ancho de banda (rápido y eficiente con una sobrecarga mínima), la comunicación dentro del equipo debe ser transparente. La comunicación transparente proporciona una comprensión clara de lo que realmente está sucediendo para evitar sorpresas y ayudar a generar confianza entre los miembros del equipo. Siempre he sentido que los equipos deben comunicarse de una manera que se alinee con el espíritu del principio de menor asombro. En pocas palabras, las personas deben comunicarse de una manera que sea menos probable que se sorprendan mutuamente. Por ejemplo, recuerdo que en un equipo Scrum que entrenaba, un individuo en particular elegía constantemente sus palabras durante los scrums diarios para ocluir lo que había logrado y lo que planeaba hacer. La gente frecuentemente se sorprendió ("sorprendió") al enterarse luego de que sus comunicaciones eran intencionalmente opacas y diseñadas para engañar. Esto dio como resultado que otros miembros del equipo no confiaran en este individuo, lo que a su vez impidió la capacidad del equipo de autoorganizarse y cumplir sus objetivos de velocidad.

Scrum Master

El Scrum Master se enfoca en ayudar a todos a comprender y adoptar los valores, principios y prácticas de Scrum. El Scrum Master actúa como entrenador tanto para el equipo de desarrollo como para el propietario del producto. Un Scrum Master también proporciona liderazgo en el proceso, ayudando al equipo de Scrum y al resto de la organización a desarrollar su propio enfoque Scrum de alto rendimiento y específico de la organización.

Responsabilidades del Scrum Master

SM como Coach

El Scrum Master es el entrenador ágil para el equipo Scrum, tanto el equipo de desarrollo como el propietario del producto. Al entrenar ambos roles, el Scrum Master puede eliminar

las barreras entre los roles y permitir al propietario del producto impulsar directamente el desarrollo. De manera análoga al entrenador de un equipo deportivo, el Scrum Master observa cómo el equipo está usando Scrum y hace todo lo posible para ayudarlo a alcanzar el siguiente nivel de rendimiento. Cuando surgen problemas que el equipo puede y debe poder resolver, la actitud del Scrum Master, como la de cualquier buen entrenador, es "No estoy aquí para resolver sus problemas por usted; en cambio, estoy aquí para ayudarlo a resolver sus propios problemas". Si el problema es un impedimento que el equipo no puede resolver, el Scrum Master se hace cargo de resolverlo.

Líder de Servicio

El Scrum Master a menudo se describe como un líder de servicio del equipo Scrum. Incluso cuando actúa como coach del equipo, el Scrum Master es, ante todo, un servidor del equipo Scrum, lo que garantiza que se satisfagan sus necesidades de mayor prioridad. Un líder de servicio nunca preguntaría: "Entonces, ¿qué vas a hacer por mí hoy?" En cambio, un líder de servicio pregunta: "Entonces, ¿qué puedo hacer hoy para ayudarte a ti y al equipo a ser más efectivos?"

Autoridad de Proceso

El Scrum Master es la autoridad de proceso del equipo Scrum. En esta capacidad, el Scrum Master está facultado para garantizar que el equipo Scrum promulgue y se adhiera a los valores, principios y prácticas de Scrum junto con los enfoques específicos del equipo Scrum. El Scrum Master ayuda continuamente al equipo de Scrum a mejorar el proceso, siempre que sea posible, para maximizar el valor comercial entregado. La autoridad en este contexto no es el mismo tipo de autoridad que tendría un gerente funcional o gerente de proyecto. Por ejemplo, el Scrum Master no contrata ni dispara y no puede dictar al equipo qué tareas debe hacer o cómo realizarlas. El SM tampoco es responsable de asegurarse de que el trabajo se realice. En cambio, el SM ayuda al equipo a definir y adherirse a su propio proceso para asegurarse de que el trabajo se realice.

Escudo

El Scrum Master protege al equipo de desarrollo de la interferencia externa para que pueda concentrarse en brindar valor comercial en cada sprint. La interferencia puede provenir de cualquier cantidad de fuentes, desde gerentes que desean redirigir a los miembros del equipo en medio de un sprint (como así también otros miembros del equipo), hasta problemas que se originan en otros equipos. No importa cuál sea la fuente de la interferencia, el Scrum Master actúa como un interceptor (respondiendo consultas, abordando la gestión y arbitrando disputas) para que el equipo pueda concentrarse en brindar valor.

Removedor de impedimentos

El SM también se responsabiliza de eliminar los impedimentos que inhiben la productividad del equipo (cuando los miembros del equipo no pueden eliminarlos razonablemente). Por

ejemplo, un equipo de Scrum que constantemente no puede cumplir sus objetivos de sprint. El impedimento eran los servidores de producción inestables que el equipo usó durante las pruebas. El equipo en sí no tenía control sobre estos servidores, era responsabilidad del vicepresidente de operaciones. Debido a que el equipo en sí no pudo eliminar el impedimento, El Scrum Master se hizo cargo de mejorar la estabilidad del servidor al trabajar con el vicepresidente de operaciones y otros que realmente podrían hacer algo sobre el problema de la estabilidad. *En el día a día aparecen mil impedimentos de infraestructura, permisos, etc los cuales deben ser atendidos por el SM con el responsable de cada área para minimizar o eliminar dicho impedimento y poder continuar con el desarrollo del producto.*

Agente de cambio

El Scrum Master debe ayudar a cambiar más que servidores defectuosos e impedimentos similares. Un buen ScrumMaster también debe ayudar a cambiar de opinión. Scrum puede ser muy perjudicial para el status quo; El cambio que se requiere para tener éxito con Scrum puede ser difícil. El SM ayuda a otros a comprender la necesidad de cambio, los impactos de Scrum fuera del equipo de Scrum y los amplios beneficios que Scrum puede ayudar a lograr. El Scrum Master también garantiza que se produzca un cambio efectivo en todos los niveles de la organización, lo que permite no solo el éxito a corto plazo sino, lo que es más importante, los beneficios a largo plazo del uso de Scrum. En organizaciones grandes, los Scrum Masters pueden unirse para convertirse en una fuerza más efectiva para el cambio.

Características / Skills de un Scrum Master

Erudito del Scrum y Conocedor de tecnologías y procesos

Para ser un entrenador de procesos eficaz, el Scrum Master debe estar muy bien informado sobre Scrum. El Scrum Master también debe comprender los problemas técnicos que el equipo debe abordar y las tecnologías que el equipo utilizará para crear soluciones. Un Scrum Master no necesita tener conocimientos de nivel de liderazgo técnico o de desarrollo, pero el conocimiento técnico razonable es una ventaja. El Scrum Master tampoco necesita ser un experto en el dominio comercial (el dueño del producto sí), pero nuevamente, el conocimiento práctico del dominio comercial es muy útil.

Cuestionador

Los SMs utilizan sus habilidades de coaching junto con sus conocimientos de procesos, técnicos y comerciales para hacer grandes preguntas. Se involucran en una investigación intencional, haciendo el tipo de preguntas que hacen que las personas se detengan y digan: “Hmm. Nunca pensé en eso. Ahora que haces esa pregunta, me hace pensar que podría haber otro camino a seguir ”. Los mejores Scrum Masters casi nunca responden directamente una pregunta, sino que responden reflexivamente con su propia pregunta, no una pregunta molesta o una pregunta por el simple hecho de preguntar. Una pregunta, pero más bien una

pregunta reflexiva, profunda, inquisitiva, ayudando así a las personas a darse cuenta de que tienen la idea de encontrar sus propias respuestas (una forma de cuestionamiento socrático).

Paciente

Debido a que los SM prefieren no dar respuestas, deben ser pacientes, dando tiempo a los equipos para llegar a las respuestas apropiadas por su cuenta. A veces es difícil para mí ser un Scrum Master porque veo el problema con el que está lidiando el equipo y "sé" la respuesta. Bueno, al menos creo que sé la respuesta! Es arrogante para mí (o cualquier Scrum Master) creer que soy más inteligente que la inteligencia colectiva del equipo. Entonces, a veces solo tengo que morderme la lengua y ser paciente, dejando que el equipo encuentre la solución, periódicamente haciendo preguntas de sondeo para ayudar a guiar las cosas.

Colaborativo

El Scrum Master debe tener excelentes habilidades de colaboración para trabajar con el propietario del producto, el equipo de desarrollo y todas las demás partes, incluso aquellos que podrían no estar directamente involucrados con Scrum. Además, como entrenador de procesos, el Scrum Master siempre está buscando oportunidades para ayudar a los miembros del equipo Scrum a lograr un nivel envidiable de colaboración dentro del equipo. Un Scrum Master puede ayudar en este esfuerzo al exhibir personalmente habilidades efectivas de colaboración.

Protector

El Scrum Master debe ser muy protector con el equipo. La analogía común es que el Scrum Master actúa como un perro pastor, protegiendo a la manada de lobos que podrían intentar atacar. En nuestro contexto, los lobos podrían ser impedimentos organizacionales o personas con diferentes agendas. El SM es experto en garantizar la protección del equipo dentro del contexto más amplio de tomar decisiones comerciales económicamente sólidas. Con una aguda sensibilidad hacia la protección del equipo y las necesidades comerciales, el Scrum Master ayuda al equipo Scrum a lograr un equilibrio saludable. El Scrum Master también ayuda a los miembros del equipo que comienzan a alejarse del rebaño. Cuando las cosas se ponen difíciles, es fácil para las personas recurrir a enfoques familiares y no ágiles. En este caso, es el trabajo de Scrum Master ayudar a pastorear a los miembros del equipo extraviados, ayudándoles a superar sus dificultades al reforzar cómo usar Scrum de manera más efectiva.

Transparente

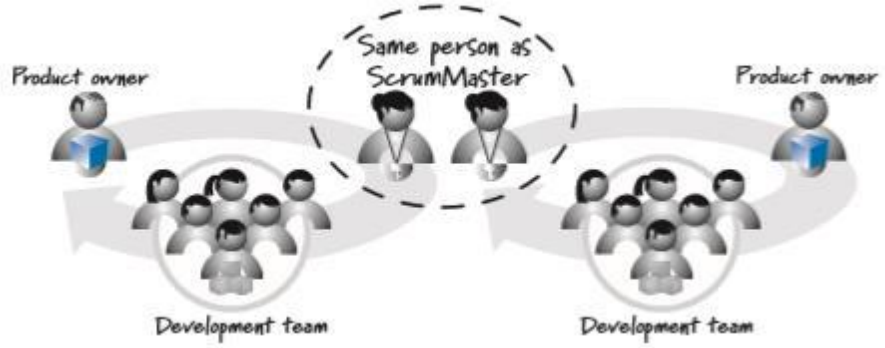
Finalmente, el Scrum Master es transparente en todas las formas de comunicación. Cuando se trabaja con miembros del equipo, no hay espacio para agendas ocultas; lo que ves y escuchas del Scrum Master debe ser lo que obtienes. La gente no espera menos de un líder de servicio. El SM también promueve la comunicación transparente fuera del equipo Scrum. Sin acceso transparente a la información, es difícil para la organización inspeccionar y adaptarse para lograr los resultados comerciales deseados mediante el uso de Scrum.

El Rol del Scrum Master a medida que pasa el tiempo

Un equipo Scrum que ha estado trabajando juntos durante un período prolongado de tiempo y se ha vuelto altamente competente con Scrum podría requerir menos entrenamiento que un nuevo equipo compuesto por personas que nunca han trabajado juntas y que son nuevas en Scrum. Aunque es posible que el Scrum Master necesite pasar menos tiempo con el equipo día a día a medida que el equipo madura, el rol de Scrum Master sigue siendo crítico para el éxito de Scrum dentro de la organización. Por lo general, a medida que disminuye la necesidad del equipo Scrum de un Scrum Master, aumenta la necesidad de que el SM se concentre en impedimentos organizacionales más amplios y sea un agente de cambio en toda la cadena de valor de la organización. En la mayoría de los casos, el rol Scrum Master sigue siendo un compromiso significativo de tiempo. En aquellos casos en los que no es un compromiso a tiempo completo, puede tener lugar una combinación de roles.

Scrum Master combinado con otros Roles

Si la capacidad lo permite y una sola persona es tanto un Scrum Master talentoso como un miembro del equipo de desarrollo, esa persona puede actuar en ambos roles. Sin embargo, esta combinación podría sufrir un conflicto de intereses cuando la persona trata de usar ambos sombreros. Por ejemplo, ¿qué pasa si la persona en los roles combinados tiene actividades importantes de Scrum Master (como eliminar impedimentos) para realizar y también tiene un trabajo crítico a nivel de tarea que hacer? Debido a que ambos son importantes, comprometerlos reducirá la efectividad del equipo Scrum. Para complicar la compensación es el hecho de que los impedimentos pueden ocurrir de manera impredecible y llevar mucho tiempo abordarlos. Esto hace que sea aún más difícil predecir cuánto tiempo un Scrum Master como miembro del equipo tendrá realmente disponible para realizar un trabajo a nivel de tarea. Sin embargo, hay un enfoque diferente que a menudo es mejor. *Si un ScrumMaster realmente tiene capacidad disponible, en muchos casos mi preferencia es que esa persona sea el ScrumMaster para más de un equipo Scrum*. Convertirse en un buen Scrum Master requiere adquirir un conjunto de habilidades valiosas y no tan comunes. Prefiero que una persona que tenga esas habilidades las comparta con varios equipos en lugar de pasar tiempo realizando actividades que no sean las de Scrum Master. No existe un enfoque correcto o incorrecto, aunque puede haber uno en un contexto organizacional específico.



El servicio del Scrum Master al Product Owner

El Scrum Master da servicio al Dueño de Producto de varias formas, incluyendo:

• Asegurar que los objetivos, el alcance y el dominio del producto sean entendidos por todos en el equipo Scrum de la mejor manera posible;

• Encontrar técnicas para gestionar la Lista de Producto de manera efectiva;

• Ayudar al Equipo Scrum a entender la necesidad de contar con elementos de Lista de Producto claros y concisos;

• Entender la planificación del producto en un entorno empírico;

• Asegurar que el Dueño de Producto conozca cómo ordenar la Lista de Producto para maximizar el valor;

• Entender y practicar la agilidad; y,

• Facilitar los eventos de Scrum según se requiera o necesite.

El servicio del Scrum Master al Development Team

El Scrum Master da servicio al Equipo de Desarrollo de varias formas, incluyendo:

• Guiar al Equipo de Desarrollo en ser autoorganizado y multifuncional;

• Ayudar al Equipo de Desarrollo a crear productos de alto valor;

• Eliminar impedimentos para el progreso del Equipo de Desarrollo;

• Facilitar los eventos de Scrum según se requiera o necesite; y,

• Guiar al Equipo de Desarrollo en entornos organizacionales en los que Scrum aún no haya sido adoptado y entendido por completo.

El servicio del Scrum Master a la organización

El Scrum Master da servicio a la organización de varias formas, incluyendo:

• Liderar y guiar a la organización en la adopción de Scrum;

• Planificar las implementaciones de Scrum en la organización;

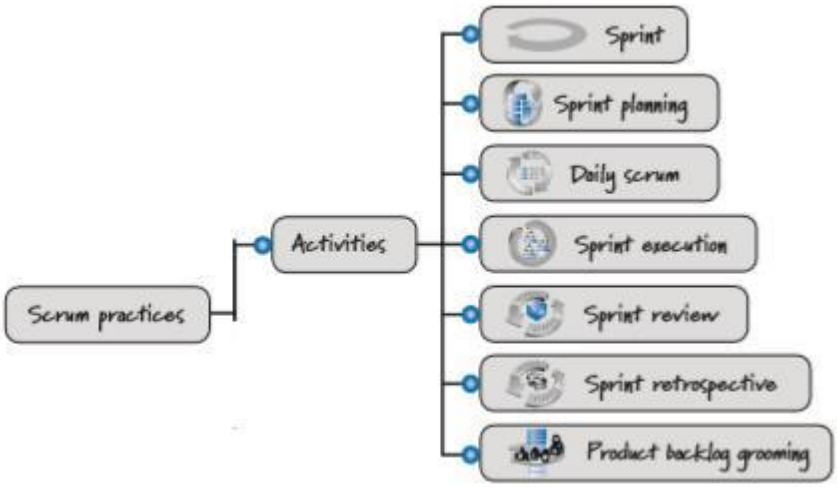
• Ayudar a los empleados e interesados a entender y llevar a cabo Scrum y el desarrollo empírico de producto;

• Motivar cambios que incrementen la productividad del Equipo Scrum; y,

• Trabajar con otros Scrum Masters para incrementar la efectividad de la aplicación de Scrum en la organización.

Eventos, profundización y desarrollo

En Scrum existen eventos predefinidos con el fin de crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum. Todos los eventos son bloques de tiempo (time-boxes), de tal modo que todos tienen una duración máxima. Una vez que comienza un Sprint, su duración es fija y no puede acortarse o alargarse. Los demás eventos pueden terminar siempre que se alcance el objetivo del evento, asegurando que se emplee una cantidad apropiada de tiempo sin permitir desperdicio en el proceso.



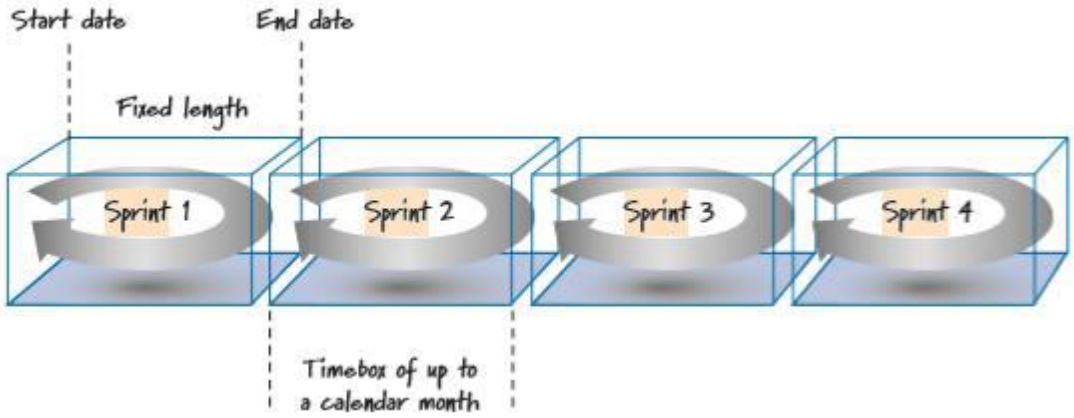
Sprint

El corazón de Scrum es el Sprint, es un bloque de tiempo (time-box) de un mes o menos durante el cual se crea un incremento de producto "Terminado" utilizable y potencialmente desplegable. Es más conveniente si la duración de los Sprints es consistente a lo largo del esfuerzo de desarrollo. Cada nuevo Sprint comienza inmediatamente después de la finalización del Sprint anterior.

Los Sprints contienen los siguientes eventos y prácticas la Planificación del Sprint (Sprint Planning), los Scrums Diarios (Daily Scrums), el trabajo de desarrollo, la Revisión del Sprint (Sprint Review), y la Retrospectiva del Sprint (Sprint Retrospective).

Cada Sprint puede considerarse un proyecto con un horizonte no mayor de un mes. Al igual que los proyectos, los Sprints se usan para lograr algo. Cada Sprint tiene una meta de lo que se construirá, un diseño y un plan flexible que guiará su construcción, el trabajo del equipo y el incremento de producto resultante.

Los Sprints están limitados a un mes calendario. Cuando el horizonte de un Sprint es demasiado grande la definición de lo que se está construyendo podría cambiar, la complejidad podría incrementarse y el riesgo podría aumentar. Los Sprints habilitan la predictibilidad al asegurar la inspección y adaptación del progreso al menos en cada mes calendario. Los Sprints también limitan el riesgo al costo de un mes calendario.



El trabajo completado en cada sprint debería crear algo de valor tangible para el cliente o usuario.

Timeboxing es una técnica para limitar la cantidad de WIP (trabajo en proceso). WIP representa un inventario de trabajo que se inició pero aún no se terminó. No gestionarlo adecuadamente puede tener graves consecuencias económicas. Debido a que el equipo planeará trabajar solo en aquellos elementos que cree que pueden comenzar y terminar dentro del sprint, timeboxing establece un límite de WIP en cada sprint.

Timeboxing nos obliga a *priorizar* y realizar la pequeña cantidad de trabajo que más importa. Esto agudiza nuestro enfoque en hacer que algo valioso se haga rápidamente.

Timeboxing mejora la previsibilidad. Aunque no podemos predecir con gran certeza exactamente el trabajo que completaremos dentro de un año, es completamente razonable esperar que podamos predecir el trabajo que podemos completar en el próximo sprint corto. *La estimación es una de las partes más difíciles de todo proceso de desarrollo de sistemas. Esta mejora con el tiempo y a medida que los desarrolladores tengan más experiencia e información acerca de la feature a estimar.*

La duración de cada Sprint es determinada por cada equipo y puede ser alterada dependiendo de las diferentes experiencias, lo ideal es encontrar la duración que mejor se acomode al equipo y mantenerla en el tiempo restante al proyecto (tampoco se aconsejan demasiados cambios, sino más bien dar tiempo al equipo a que se "acomode" a la duración del Sprint).

Cancelación de un Sprint

Un Sprint puede cancelarse antes de que el bloque de tiempo llegue a su fin. Solo el Dueño de Producto tiene la autoridad para cancelar el Sprint, aunque puede hacerlo bajo la influencia de los interesados, del Equipo de Desarrollo o del Scrum Master.

Un Sprint se cancelaría si el Objetivo del Sprint llega a quedar obsoleto. Esto podría ocurrir si la compañía cambia la dirección o si las condiciones del mercado o de la tecnología cambian. En general, un Sprint debería cancelarse si no tuviese sentido seguir con él dadas las

circunstancias. Sin embargo, debido a la corta duración de los Sprints, su cancelación rara vez tiene sentido.

Cuando se cancela un Sprint se revisan todos los Elementos de la Lista de Producto que se hayan completado y "Terminado". Si una parte del trabajo es potencialmente entregable, el Dueño de Producto normalmente la acepta. Todos los Elementos de la Lista de Producto no completados se vuelven a estimar y se vuelven a introducir en la Lista de Producto. El trabajo finalizado en ellos pierde valor con rapidez y por lo general debe volverse a estimar. *En la realidad, generalmente el trabajo realizado se reutiliza y rara vez se reestima, solo si hay algún cambio en la historia de usuario o se considera que fue mal estimada en un primer lugar. También puede ser reestimada si un nuevo miembro del equipo de desarrollo va a trabajar en la historia de usuario.*

Las cancelaciones de Sprint consumen recursos ya que todos se reagrupan en otra Planificación de Sprint para empezar otro Sprint. Las cancelaciones de Sprint son a menudo traumáticas para el Equipo Scrum y son muy poco comunes.

Planificación de Sprint (Sprint Planning)

El trabajo a realizar durante el Sprint se planifica en la Planificación de Sprint. Este plan se crea mediante el trabajo colaborativo del Equipo Scrum completo.

La Planificación de Sprint tiene un máximo de duración de ocho horas para un Sprint de un mes *(en ambientes de trabajo "reales" los miembros del equipo no "toleran" reuniones largas, por lo tanto duran menos y se recomienda dividirlas en 2 días si toman mucho tiempo)*. Para Sprints más cortos el evento es usualmente más corto. El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito. El Scrum Master enseña al Equipo Scrum a mantenerse dentro del bloque de tiempo.

La Planificación de Sprint responde a las siguientes preguntas:

• ¿Qué puede entregarse en el Incremento resultante del Sprint que comienza?

• ¿Cómo se conseguirá hacer el trabajo necesario para entregar el Incremento?

En colaboración con el propietario del producto y con la facilitación de ScrumMaster, el equipo de desarrollo ayuda a establecer el objetivo para el próximo sprint. Luego, el equipo determina qué subconjunto de alta prioridad de los elementos de la cartera de productos que se creará para lograr ese objetivo. Tenga en cuenta que la planificación se realiza de forma iterativa. En lugar de centrarse en un plan muy grande, incierto y demasiado detallado al comienzo de un esfuerzo de desarrollo, el equipo hace una serie de planes más pequeños, más seguros y más detallados justo a tiempo al comienzo de cada sprint.

El equipo completo de Scrum colabora durante la planificación del sprint. El propietario del producto comparte el objetivo de sprint inicial, presenta la cartera de pedidos priorizada del producto y responde cualquier pregunta que el equipo pueda tener con respecto a los

elementos de la cartera de productos. El equipo de desarrollo trabaja diligentemente para determinar lo que puede ofrecer y luego se compromete de manera **realista** al final de la planificación del sprint. El ScrumMaster, actuando como coach del equipo Scrum, observa la actividad de planificación, hace preguntas de sondeo y facilita ayudar a garantizar un resultado exitoso. Debido a que Scrum Master no está a cargo del equipo de desarrollo, no puede decidir en nombre del equipo de desarrollo qué compromiso tomar. Sin embargo, el Scrum Master puede y debe desafiar el compromiso del equipo de garantizar que sea realista y apropiado.

Imputs del Sprint Planning

Backlog del producto: Antes de la planificación del sprint, los elementos principales de la cartera de productos deben estar preparados para que estén listos.

Velocity del equipo: La velocidad histórica del equipo es un indicador de cuánto trabajo es práctico para que el equipo complete en un sprint.

Restricciones: Se identifican restricciones comerciales o técnicas que podrían afectar materialmente lo que el equipo puede ofrecer.

Capacidades del equipo: Las capacidades tienen en cuenta qué personas están en el equipo, qué habilidades tiene cada miembro del equipo y qué tan disponible estará cada persona en el próximo sprint

Sprint Goal Inicial: Este es el objetivo comercial que al propietario del producto le gustaría ver cumplido durante el sprint.

Outputs del Sprint Plannings:

Sprint Goal refinado y los elementos del Product Backlog que el equipo de desarrollo de "*compromete*" a completar para alcanzar este objetivo.

Scrum Diario (Daily Scrum)

El scrum diario es una actividad crítica de inspección y adaptación diaria para ayudar al equipo a lograr un flujo más rápido y flexible hacia la solución. El scrum diario es una actividad de 15 minutos, que se realiza una vez cada 24 horas. El scrum diario sirve como una actividad de inspección, sincronización y planificación adaptativa diaria que ayuda a un equipo autoorganizado a hacer mejor su trabajo. El objetivo del scrum diario es que las personas que se centran en alcanzar el objetivo del sprint se reúnan y compartan el panorama general de lo que está sucediendo para que puedan comprender colectivamente cuánto trabajar, en qué elementos comenzar a trabajar y cómo para organizar mejor el trabajo entre los miembros del equipo. El scrum diario también ayuda a evitar la espera. Si hay un problema que bloquea el flujo, el equipo nunca tendría que esperar más de un día para discutirlo. Imagínese si los miembros del equipo se juntaran solo una vez a la semana: se negarían a sí mismos los

beneficios de la retroalimentación rápida. En general, el scrum diario es esencial para la gestión del flujo.

"La daily" se debe realizar todos los días al comienzo del mismo y no debe llevar más de 15 minutos.

Un enfoque común para realizar el scrum diario hace que Scrum Master facilite y cada miembro del equipo se turne para responder tres preguntas en beneficio de los otros miembros del equipo:

- ¿Qué logré desde el último scrum diario?
- ¿En qué planeo trabajar para el próximo scrum diario?
- ¿Cuáles son los obstáculos o impedimentos que me impiden avanzar?

Al responder estas preguntas, todos entienden el panorama general de lo que está ocurriendo, cómo están progresando hacia la meta del sprint, cualquier modificación que quieran hacer en sus planes para el trabajo del día siguiente y qué problemas deben abordarse. El scrum diario es esencial para ayudar al equipo de desarrollo a administrar el flujo de trabajo rápido y flexible dentro de un sprint. El scrum diario no es una actividad de resolución de problemas. Más bien, muchos equipos deciden hablar sobre problemas después del scrum diario y lo hacen con un pequeño grupo de personas interesadas. *El scrum diario tampoco es una reunión de estado tradicional, especialmente el tipo convocado históricamente por los gerentes de proyecto para que puedan obtener una actualización sobre el estado del proyecto. Sin embargo, un scrum diario puede ser útil para comunicar el estado de los elementos del backlog de Sprint entre los miembros del equipo de desarrollo.* Principalmente, el scrum diario es una actividad de inspección, sincronización y planificación diaria adaptativa que ayuda a un equipo de autoorganización a hacer mejor su trabajo.

Se intenta que en la daily participen tanto los miembros de Dev Team como así también los miembros del equipo de Discovery ya que está comprobado que cuando se logra la atención de todos y se involucran unos con el trabajo de otro se llegan a mejores resultados.

Sprint Review (DEMO)

Al final del sprint hay dos actividades adicionales de inspección y adaptación. Uno se llama la revisión de sprint. El objetivo de esta actividad es inspeccionar y adaptar el producto que se está construyendo. Es fundamental para esta actividad la conversación que se lleva a cabo entre sus participantes, que incluye el equipo Scrum, las partes interesadas, los patrocinadores, los clientes y los miembros interesados de otros equipos. La conversación se centra en revisar las características recién completadas en el contexto del esfuerzo de desarrollo general. Todos los asistentes obtienen una visibilidad clara de lo que está ocurriendo y tienen la oportunidad de ayudar a guiar el desarrollo futuro para garantizar que se cree la solución más adecuada para el negocio. Una revisión exitosa da como resultado un flujo de información bidireccional. Las personas que no están en el equipo Scrum pueden

sincronizarse con el esfuerzo de desarrollo y ayudar a guiar su dirección. Al mismo tiempo, los miembros del equipo Scrum obtienen una apreciación más profunda del lado comercial y de marketing de su producto al recibir comentarios frecuentes sobre la convergencia del producto hacia clientes o usuarios encantados. Por lo tanto, la revisión de sprint representa una oportunidad programada para inspeccionar y adaptar el producto. Como práctica, las personas ajenas al equipo de Scrum pueden realizar revisiones de características dentro del sprint y proporcionar comentarios para ayudar al equipo de Scrum a lograr mejor su objetivo de sprint.

Dependiendo de la organización y el proyecto en que trabajemos, a esta reunión solo asisten ciertas partes interesadas como el PO y el delivery manager; y miembros de la Gerencia de la organización (de ser un proyecto interno), de ser un proyecto para una compañía externa, debe ser realizada como indica la teoría.

Debe realizarse cierto trabajo previo al evento (por más que el mismo sea de aspecto informal) ya que se realizará una DEMO de lo completado en el Sprint a los Stakeholders. Por ende debemos asegurarnos que todo el trabajo está completo y puede presentarse de forma tangible. Otro punto a considerar es decidir a quién invitar (Stakeholders internos y externos) a la Sprint Review y quién será el encargado de llevar a cabo la presentación y DEMO del producto y features completadas.

Es el único evento del Sprint el cual puede contar con invitados externos al Scrum Team.

Si existiese un mayor esfuerzo de desarrollo con múltiples equipos Scrum, podría tener sentido considerar hacer una revisión conjunta de sprint. Esta es simplemente una revisión que incluye el trabajo realizado por múltiples equipos altamente interrelacionados. Hay varios beneficios para este enfoque. Primero, las partes interesadas tienen que asistir solo a una revisión de sprint en lugar de varias. En segundo lugar, si se suponía que el trabajo debía integrarse, tendría sentido que la revisión se centrara en el trabajo integrado, no en una colección de incrementos independientes. Para lograr este objetivo, todos los equipos deben asegurarse de que sus definiciones de hecho incluyan pruebas de integración, como probablemente deberían hacer de todos modos. La desventaja de celebrar una reunión conjunta de revisión de sprint con más de un equipo es que probablemente tomará un poco más de tiempo y podría requerir una sala más grande de la que cualquier equipo hubiera necesitado.

Sprint Retrospective

La segunda actividad de inspección y adaptación al final del sprint es la retrospectiva del sprint. Esta actividad ocurre frecuentemente después de la revisión del sprint y antes de la próxima planificación del sprint. Mientras que la revisión del sprint es un momento para inspeccionar y adaptar el producto, la retrospectiva del sprint es una oportunidad para inspeccionar y adaptar el proceso. Durante la retrospectiva del sprint, el equipo de desarrollo, ScrumMaster y el propietario del producto se unen para discutir qué funciona y qué no

funciona con Scrum y las prácticas técnicas asociadas. La atención se centra en la mejora continua del proceso necesaria para ayudar a que un buen equipo Scrum sea excelente. Al final de una retrospectiva del sprint, el equipo Scrum debería haber identificado y comprometido con un número práctico de acciones de mejora de procesos que el equipo Scrum llevará a cabo en el próximo sprint. Una vez completada la retrospectiva del sprint, se repite todo el ciclo nuevamente, comenzando con la siguiente sesión de planificación del sprint, que se lleva a cabo para determinar el conjunto de trabajo de mayor valor actual para que el equipo se concentre.

La retrospectiva del sprint es una de las prácticas más importantes y menos apreciadas en el marco de Scrum. Es importante porque brinda a los equipos la oportunidad de personalizar Scrum a sus circunstancias únicas. Se subestima porque algunas personas tienen una visión equivocada de que les lleva tiempo hacer trabajos de diseño, construcción y prueba "reales". La retrospectiva del sprint es un contribuyente crucial para la mejora continua que ofrece Scrum. Y aunque algunas organizaciones pueden esperar para hacer una retrospectiva hasta el final de un gran esfuerzo de desarrollo, los equipos de Scrum tienen retrospectivas cada sprint, lo que permite a los equipos aprovechar las ideas y los datos antes de que se pierdan. Debido a que un equipo Scrum se reúne al final de cada sprint para inspeccionar y adaptar su proceso Scrum, puede aplicar el aprendizaje temprano e incremental a lo largo del proceso de desarrollo y, por lo tanto, afectar significativamente el resultado del proyecto.

Una retrospectiva de sprint puede ser tan simple como que los miembros del equipo Scrum se unan para discutir preguntas como

- ¿Qué funcionó bien este sprint que queremos seguir haciendo?
- ¿Qué no funcionó bien este sprint que deberíamos dejar de hacer?
- ¿Qué debemos comenzar a hacer o mejorar?

Con base en sus discusiones, los miembros del equipo determinan algunos cambios accionables para realizar y luego continúan con el próximo sprint con un proceso mejorado gradualmente.

Deben participar todos los integrantes del equipo: Scrum Master, Product Owner, equipo Discovery y equipo de desarrollo.

Las “retros” pueden ponerse tensas, por los diferentes comentarios, quejas y reclamos de los diferentes miembros del equipo, por lo tanto se pueden llevar a cabo de manera muy informal, con música de fondo, y de hecho hay un gran acabado de recursos con opciones para hacer retrospectivas de diferentes formas y enfoques, pero cuyo resultado sea siempre el mismo: enfocar al equipo en la mejora contínua.

Ejemplo:

Con frecuencia se les pide a los participantes que hagan una lluvia de ideas y luego las capturen en tarjetas y las coloquen en una pared compartida u otra superficie para que todos puedan verlas. Otra fuente de información podría ser la acumulación de información del equipo, una lista priorizada de información generada previamente que aún no se ha implementado. Si existe tal backlog, explíquelo para ver qué ideas les gustaría incluir y considerar a los participantes durante la retrospectiva actual. Cualquier información existente debe representarse con tarjetas y colocarse en la pared junto con las nuevas ideas. Después de crear un contexto compartido y extraer los datos para obtener información, los participantes deberían haber identificado muchas áreas para mejorar su uso de Scrum y, por extensión, la forma en que trabajan juntos para ofrecer valor. Algunas de estas ideas pueden conducir a discusiones más profundas entre los participantes para comprender mejor las causas subyacentes, o patrones o relaciones importantes. Cuando todas las ideas se han discutido y organizado en el muro, es hora de determinar qué hacer con toda la información.

Determinar acciones en la retrospectiva

Las ideas son nuestras ideas o percepciones de cosas que pueden mejorarse. Para extraer valor a largo plazo de estas ideas, necesitamos pasar de discutirlas a tomar acciones demostrables para aprovecharlas. Por ejemplo, si la idea es "Estamos perdiendo demasiado tiempo porque el sistema de administración de código sigue fallando", la acción de mejora en el futuro podría ser "Hacer que Tanya aplique los parches del proveedor al sistema de administración de código para hacerlo más estable". Tanya, miembro del equipo de desarrollo, puede tomar esta acción en el próximo sprint. Los participantes también deben tomarse el tiempo para revisar lo que sucedió con las acciones de mejora desde la última retrospectiva. Si esas acciones no se han completado (o incluso comenzado), los participantes deben saber por qué antes de comenzar a abordar nuevas ideas. Pueden optar por llevar a cabo acciones previas o priorizarlas frente a las nuevas ideas que acaban de identificar.

Dar seguimiento

Para asegurarse de que lo que sucede en la retrospectiva del sprint no solo se quede en la retrospectiva del sprint, los participantes deben hacer un seguimiento de las acciones que decidieron completar. Algunas acciones (como que todos lleguen a tiempo a las reuniones diarias) solo necesitan ser reiteradas y reforzadas por los miembros del equipo y el ScrumMaster. Otros deberán ser abordados durante la próxima actividad de planificación de sprint. Las acciones que no requieren tiempo del miembro del equipo probablemente encontrarán un hogar en la lista de impedimentos del Scrum Master. Y las acciones que están destinadas a otros equipos o la organización en su conjunto se pueden ubicar en la cartera de pedidos adecuada para las personas que se espera que hagan el trabajo; *el Scrum Master normalmente realiza un seguimiento con las partes externas para ayudar a garantizar que estas acciones realmente se lleven a cabo*.

Sprint 0

Desde un punto de vista técnico, el Sprint 0 es un **paso previo** al desarrollo de un proyecto Scrum. Su importancia está en que nos permite establecer al menos 3 líneas maestras, con las que trabajaremos durante todo el tiempo que dure el proyecto.

Cualquier idea que quieras aplicar en tu proyecto se puede empezar a desarrollar en este **sprint previo**. En cualquier caso, hay al menos **3 aspectos** que suelen quedar ya muy definidos. Son los siguientes:

 Características del producto: En este Sprint 0 tenemos que recabar todo tipo de información acerca del producto y las características que debe tener para su lanzamiento. ¿Cómo? Trabajando activamente con el Product Owner.

 Estructura del Backlog: Fundamental durante esta fase. Definimos cómo serán las historias de usuario y las dejamos preparadas para fases posteriores.

 Equipo y estructura: En una reunión con el equipo se definen funciones y se perfila el método de trabajo que vamos a seguir.

Ventajas del Sprint 0

 Obtenemos una definición contrastada con el cliente en forma de historias de usuarios

 El equipo participa en la preparación del desarrollo identificando necesidades, dificultades, ventajas...

 La preparación de cada sprint será más fácil de realizar porque el equipo conoce con detalle el proyecto

Actividades que podemos realizar en el Sprint 0

 Definir los releases y plan del producto (se puede emplear una técnica como el user story mapping)

 Definir la arquitectura base del producto sobre la cual se va a evolucionar.

 Definir las pruebas a realizar

 Definir los lineamientos gráficos del sistema (si aplica)

 Realizar todas las instalaciones en los diferentes ambientes (desarrollo, pruebas, pre-producción y producción)

o IDEs

o Servidores de aplicaciones

o motores de bases de datos

o Nota: Que todos los ambientes sean lo más similares entre sí posibles (sistemas operativos, versiones de productos, conectividad, etc)

 Preparar y configurar los repositorios de control de versiones

 Configurar los scripts para el servidor de integración continua de forma que logremos que una linea de codigo quede probado y desplegado en los respectivos ambientes

 Escribir el 'hola mundo" que logre pasar por:

o el repositorio del control de versiones

o probado por las diferentes pruebas definidas

o que en los diferentes servidores automáticamente quede:

 código validado con reglas automáticas

 compilado

 probado

 y desplegado en los diferentes ambientes

o Nota: Es como tender la carretera pavimentada para que las líneas de código pasen sin inconvenientes de inicio a fin.

 Escribir las historias de usuario para el primer planning. Posiblemente mostrarlas al equipo para que realice un primer acercamiento.

 Definir herramientas de gestión que utilizará el equipo

o Kanban

o Burndowns

o hojas de cálculo

 Definir la DoD (Defnition of Done((Definición de Hecho/Realizado/Completado) para el proyecto

 Definir los criterios de liberación del producto / liberación de releases

o Aprobaciones

o historias completadas

o valor de negocio entregado

 Definir como se calculará el ROI del proyecto y/o el éxito del mismo

 Definir forma de puntuar las historias

 Definir estándares de codificación y documentación del código

 Definir documentación a emplear y como se va a actualizar en el proyecto

 Definir tamaño de los sprints

No todas estas tareas son obligatorias, son opcionales y probablemente un equipo que venga trabajando hace tiempo en diferentes proyectos, solo realice una pequeña parte de las mismas (las relacionadas con story mapping y Discovery, como las de preparación de desarrollo y modelos de datos).

Herramientas y artefactos de Scrum:

Product Backlog; Definition of Done, Capacity y Estimaciones (incluye herramientas de estimación como Poker planning), Jira / Azure Dev/Ops demostración ciclo de vida de una historia

Technical Debt; Product Backlog grooming/Refinamiento, Scrum of Scrums, Equipo dev y equipo Discovery

Product Backlog

La Lista de Producto es una lista ordenada de todo lo que se conoce que es necesario en el producto. Es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El Dueño de Producto (Product Owner) es el responsable de la Lista de Producto, incluyendo su contenido, disponibilidad y ordenación.

Una Lista de Producto nunca está completa. El desarrollo más temprano de la misma solo refleja los requisitos conocidos y mejor entendidos al principio. La Lista de Producto evoluciona a medida de que el producto y el entorno en el que se usará también lo hacen. La Lista de Producto es dinámica; cambia constantemente para identificar lo que el producto necesita para ser adecuado, competitivo y útil. Si un producto existe, su Lista de Producto también existe.

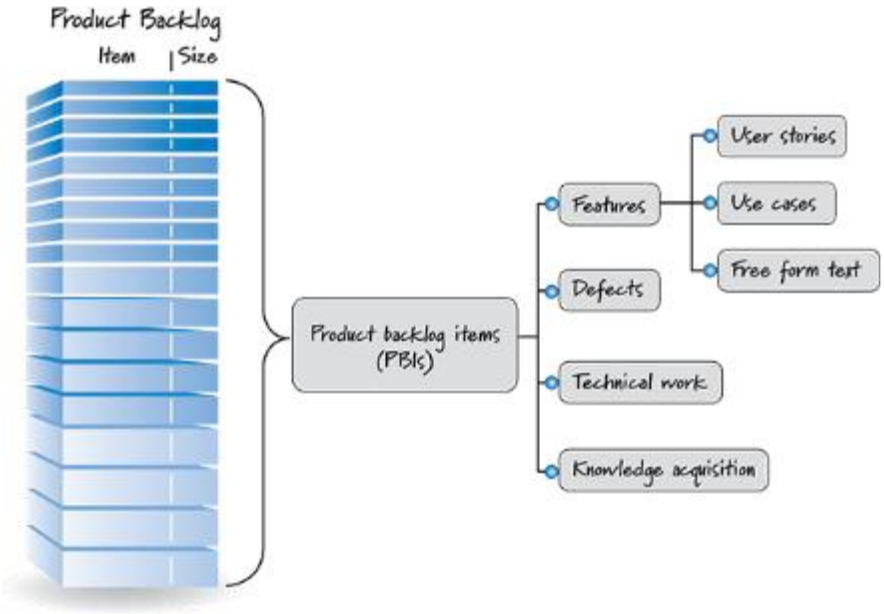
La Lista de Producto enumera todas las características, funcionalidades, requisitos, mejoras y correcciones que constituyen cambios a realizarse sobre el producto para entregas futuras. Los elementos de la Lista de Producto tienen como atributos la descripción, el orden, la estimación y el valor. Los elementos de La Lista de Producto muchas veces incluyen descripciones de las pruebas que demostrarán la completitud de tales elementos cuando estén “Terminados”.

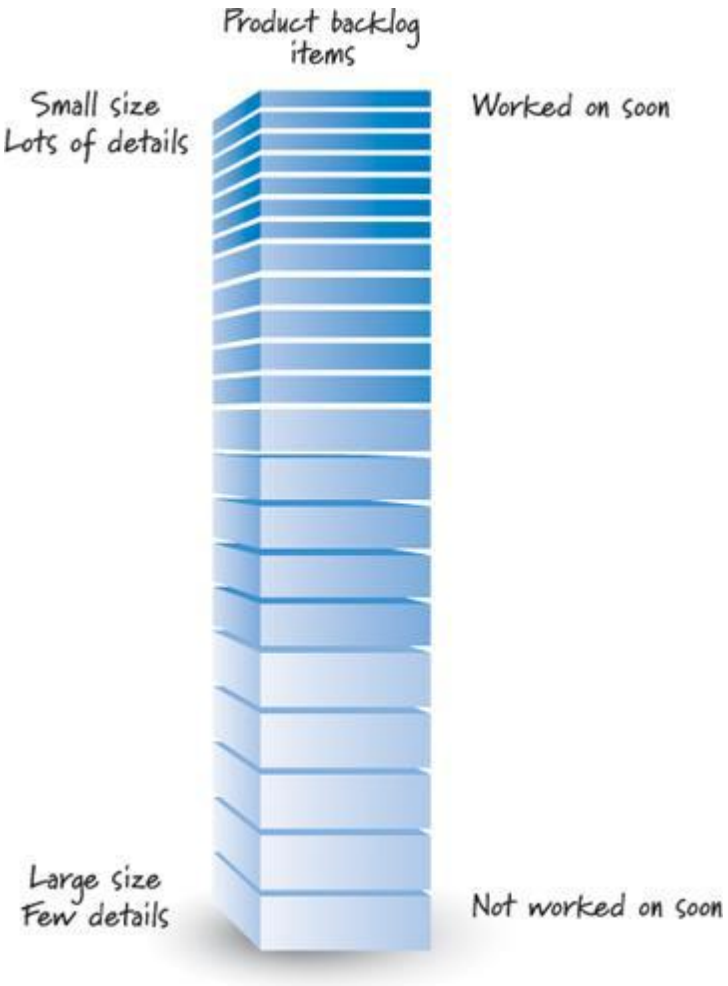
A medida que un producto es utilizado y se incrementa su valor y el mercado proporciona retroalimentación, la Lista de Producto se convierte en una lista más larga y exhaustiva. Los requisitos nunca dejan de cambiar así que la Lista de Producto es un artefacto vivo. Los

cambios en los requisitos de negocio, las condiciones del mercado o la tecnología podrían causar cambios en la Lista de Producto.

A menudo, varios Equipos Scrum trabajan juntos en el mismo producto. Para describir el trabajo a realizar sobre el producto se utiliza una única Lista de Producto. En ese caso podría emplearse un atributo de la Lista de Producto para agrupar los elementos.

La cartera de productos está compuesta por elementos de la cartera de pedidos, a los que me refiero como PBI, product backlog item. La mayoría de los PBI son características, elementos de funcionalidad que tendrán un valor tangible para el usuario o el cliente. A menudo se escriben como historias de usuarios (aunque Scrum no especifica el formato de los PBI). Los ejemplos de funciones incluyen algo nuevo (una pantalla de inicio de sesión para un nuevo sitio web) o un cambio a una función existente (una pantalla de inicio de sesión más fácil de usar para un sitio web existente). Otros PBI incluyen defectos que necesitan reparación, mejoras técnicas, trabajo de adquisición de conocimiento y cualquier otro trabajo que el propietario del producto considere valioso.





Cada elemento de la cartera de productos tiene una estimación de tamaño correspondiente al esfuerzo requerido para desarrollar el elemento. El propietario del producto utiliza estas estimaciones como una de varias entradas para ayudar a determinar la prioridad de un PBI (y, por lo tanto, su posición) en la cartera de pedidos del producto. Además, un PBI grande de alta prioridad (cerca de la parte superior de la cartera de pedidos) le indica al propietario del producto que es necesario un refinamiento adicional de ese artículo antes de que pueda pasar a un sprint a corto plazo. La mayoría de los PBI se estiman en puntos de la historia o en días ideales. Estas estimaciones de tamaño deben ser razonablemente precisas sin ser demasiado precisas. Debido a que los elementos cerca de la parte superior de la cartera de pedidos son más pequeños y detallados, tendrán estimaciones de tamaño más pequeñas y precisas. Es posible que no sea posible proporcionar estimaciones numéricamente precisas para elementos más grandes (como las Épicas) ubicadas cerca de la parte inferior de la cartera de pedidos, por lo que algunos equipos pueden optar por no estimarlos en absoluto o utilizar estimaciones del tamaño de la ropa (L, XL , XXL, etc.). Como estos elementos más grandes se refinan en un conjunto de elementos más pequeños, cada uno de los elementos más pequeños se estimaría con números.

Aunque el backlog del producto es una lista priorizada de PBI, es poco probable que se prioricen todos los elementos del trabajo atrasado. Es útil priorizar los elementos a corto plazo

destinados a los próximos sprints. Quizás sea valioso priorizar lo más abajo posible en la cartera de pedidos que creemos que podemos obtener en la Versión más próxima (Ej. Versión 1). Ir más allá de ese punto en cualquier otra cosa que no sea un nivel bruto de priorización probablemente no valga la pena. Por ejemplo, podríamos declarar que un artículo está destinado a la Versión 2 o la Versión 3 de acuerdo con nuestra hoja de ruta del producto (Product Roadmap). Sin embargo, si estamos en las primeras etapas del desarrollo de las características de la Versión 1, pasar un tiempo valioso preocupándose por cómo priorizar las características en las que podríamos trabajar algún día en la Versión 2 o la Versión 3 probablemente no sea una buena inversión. Es posible que nunca terminemos haciendo una versión 2 o una versión 3, o nuestras ideas sobre esas versiones pueden cambiar significativamente durante el desarrollo de la versión 1. Por lo tanto, el tiempo dedicado a priorizar tan lejos tiene una alta probabilidad de desperdiciarse. Por supuesto, a medida que surgen nuevos elementos durante el curso del desarrollo, el propietario del producto es responsable de insertarlos en el orden correcto en función de los elementos que existen actualmente en la cartera de pedidos.

Sprint Backlog

La Lista de Pendientes del Sprint es el conjunto de elementos de la Lista de Producto seleccionados para el Sprint, más un plan para entregar el Incremento de producto y conseguir el Objetivo del Sprint. La Lista de Pendientes del Sprint es una predicción hecha por el Equipo de Desarrollo acerca de qué funcionalidad formará parte del próximo Incremento y del trabajo necesario para entregar esa funcionalidad en un Incremento “Terminado”.

La Lista de Pendientes del Sprint hace visible todo el trabajo que el Equipo de Desarrollo identifica como necesario para alcanzar el Objetivo del Sprint. Para asegurar el mejoramiento continuo, la Lista de Pendientes del Sprint incluye al menos una mejora de procesos de alta prioridad identificada en la Retrospectiva inmediatamente anterior.

La Lista de Pendientes del Sprint es un plan con un nivel de detalle suficiente como para que los cambios en el progreso se puedan entender en el Scrum Diario. El Equipo de Desarrollo modifica la Lista de Pendientes del Sprint durante el Sprint y esta Lista de Pendientes del Sprint emerge a lo largo del Sprint. Esto ocurre a medida que el Equipo de Desarrollo trabaja en lo planeado y aprende más acerca del trabajo necesario para conseguir el Objetivo del Sprint.

Cuando se requiere nuevo trabajo, el Equipo de Desarrollo lo adiciona a la Lista de Pendientes del Sprint. A medida que el trabajo se ejecuta o se completa se va actualizando la estimación de trabajo restante. Cuando algún elemento del plan se considera innecesario, es eliminado. Solo el Equipo de Desarrollo puede cambiar su Lista de Pendientes del Sprint durante un Sprint. La Lista de Pendientes del Sprint es una imagen visible en tiempo real del trabajo que el Equipo de Desarrollo planea llevar a cabo durante el Sprint y pertenece únicamente al Equipo de Desarrollo.

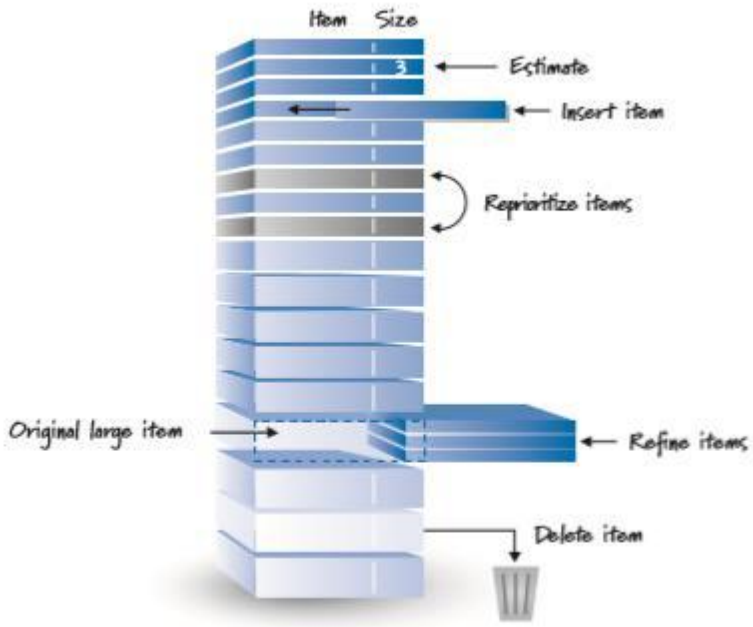
Team Capacity y Velocity

La velocity de un equipo nos permite medir cantidad de trabajo que un equipo es capaz de completar en un Sprint, este es un número que tiende a estabilizarse a medida que pasan los Sprints. Puede cambiar en cada iteración dependiendo el tiempo disponible de cada integrante (Capacity) como así también aumentará a medida que el equipo gane experiencia y aumente en seniority. De no haber modificaciones en los equipos y/o ausencias él número no debería cambiar demasiado en cortos períodos de tiempo.

La velocity se mide en "*tamaño*".

Product Backlog Grooming

Para obtener una buena cartera de pedidos de productos PROFUNDOS, debemos gestionar, organizar, administrar o, como se le conoce habitualmente, proactivamente la cartera de productos.



El grooming se refiere a un conjunto de tres actividades principales: crear y refinar (agregar detalles a) PBI, estimar PBI y priorizar PBI. La Figura ilustra algunas tareas de aseo específicas y cómo afectan la estructura de la cartera de pedidos del producto. En el momento apropiado, se deben estimar todos los PBI para ayudar a determinar su orden en el trabajo atrasado y para ayudar a decidir si se justifica el trabajo de refinamiento adicional. Además, a medida que se dispone de información importante, se crean nuevos elementos y se insertan en la cartera de pedidos en el orden correcto. Por supuesto, si las prioridades cambian, querremos reordenar los elementos en la cartera de pedidos. Y a medida que nos acerquemos a trabajar en un artículo más grande, querremos refinarlo en una colección de artículos más pequeños. También podríamos decidir que un elemento de la cartera de pedidos en particular no es necesario, en cuyo caso lo eliminaremos.

El grooming del Product Backlog es un esfuerzo colaborativo continuo liderado por el Product Owner e incluye una participación significativa de partes interesadas internas y externas, así como el Scrum Master y el equipo de desarrollo. El marco de Scrum solo indica que la preparación debe ocurrir; no especifica cuándo debería suceder. Entonces, ¿cuándo se realiza el aseo? Con Scrum, asumimos un entorno incierto y, por lo tanto, debemos estar preparados para inspeccionar y adaptarnos constantemente. Esperamos que el trabajo atrasado del producto evolucione constantemente en lugar de bloquearse temprano y cambiar solo a través de un proceso secundario para manejar eventos excepcionales e indeseables. Como resultado, debemos asegurarnos de que nuestras actividades de aseo sean una parte esencial e intrínseca de cómo gestionamos nuestro trabajo. La preparación inicial se produce como parte de la actividad de planificación de la liberación. Durante el desarrollo del producto, el propietario del producto se reúne con las partes interesadas a cualquier frecuencia que tenga sentido para realizar una preparación continua. Al trabajar con el equipo de desarrollo, el propietario del producto puede programar un taller de preparación semanal o una vez al sprint durante la ejecución del sprint. Al hacerlo, se garantiza que la preparación se realice en un horario regular y permite al equipo dar cuenta de ese tiempo durante la planificación del sprint. También reduce el desperdicio de tratar de programar reuniones ad hoc (por ejemplo, determinar cuándo hay personas disponibles, encontrar espacio disponible, etc.). A veces, los equipos prefieren extender la preparación a través del sprint, en lugar de bloquear un período predeterminado de tiempo. Toman un poco de tiempo después de sus scrums diarios para realizar una preparación incremental. Esta preparación no tiene que incluir a todos los miembros del equipo.

Product Backlog Items (PBI): Epics, Stories, Themes & Features

User Story

Las Historias de usuarios son la forma más eficiente, centrada en el usuario y el valor, de expresar expectativas hacia las capacidades funcionales de un sistema. Veo una historia de usuario como un requisito ligero, o más ortodoxo "ágil" como "un recordatorio para tener una conversación sobre tal expectativa". Por lo general, los expresamos en forma de "Como <tipo de usuario> Quiero <acción> para que <objetivo>", por ejemplo, "Como cliente, quiero poder seleccionar el método de pago, para poder elegir el más conveniente". Por supuesto, podemos agregar detalles adicionales, como los criterios de aceptación, pero eso no es relevante para esta discusión.

Dos cosas son importantes:

 Una historia debe ser narrativa completa, es decir, proporcionar un beneficio tangible para el usuario de principio a fin, pero al mismo tiempo debe ser lo suficientemente pequeño como para que podamos manejarlo durante el diseño y la entrega de la solución. En la práctica, esto significa que podemos entregarlo dentro de un Sprint. Esto es lo que la distingue de una épica.

 Como requisito comercial, una historia de usuario debe, en términos generales, ser neutral para la solución, es decir, no debe expresar la solución que satisfaga el requisito. Esto es lo que lo distingue de una Característica.

Epic

Una épica es muy parecida a una historia de usuario. También es una expresión de un requisito comercial, también es neutral para la solución, pero es grande y generalmente tendrá que desglosarse para poder abordarla, ya sea por razones de tiempo o complejidad. Un ejemplo relacionado con la historia de usuario anterior sería la siguiente épica: "Como cliente, quiero poder comprar productos en línea". Seguramente esto tendrá que desglosarse en una gran cantidad de Historias de usuarios.

Las épicas tienen dos propósitos:

 Durante el proceso de obtención de requisitos, especialmente cuando se trabaja en Agile, comenzamos con requisitos de alto nivel (Epics) que luego refinamos dividiéndolos en requisitos más detallados (Historias de usuarios).

 Nos ayudan a estructurar y agrupar nuestros requisitos y, por lo tanto, nos ayudan a gestionar nuestra cartera de producto.

Features

Una Feature describe una solución específica a un requisito y ofrece una o varias historias. Con respecto a la historia de usuario anterior, una feature podría ser un selector desplegable que ofrece diferentes opciones de pago.

Themes

Los temas abarcan Epics y / o Historias de usuarios que comparten características o atributos comunes. Con respecto al ejemplo anterior, los temas pueden ser comercio electrónico y personalización. Podemos utilizar temas para vincular objetivos y requisitos (historias épicas / historias de usuarios), así como agruparlos / clasificarlos por varias dimensiones. Los temas pueden ser proyectos específicos o proyectos cruzados.

Como escribir historias de usuario:

La base de una historia de usuario debe inciar de la siguiente manera:

Como [Rol], quiero/necesito [Descripción de la funcionalidad], con la finalidad de [Descripción de la consecuencia].

Ejemplos:

Como Vendedor, quiero registrar los productos y cantidades que me solicita un cliente para crear un pedido de venta.

Como Cliente, quiero suscribirme a un nuevo plan de T.V. por cable por medio del sitio web.

Como Representante de proveedor, quiero poder consultar los procesos de licitación que están en curso.

Como Analista de compras, quiero crear una nueva solicitud de cotización.

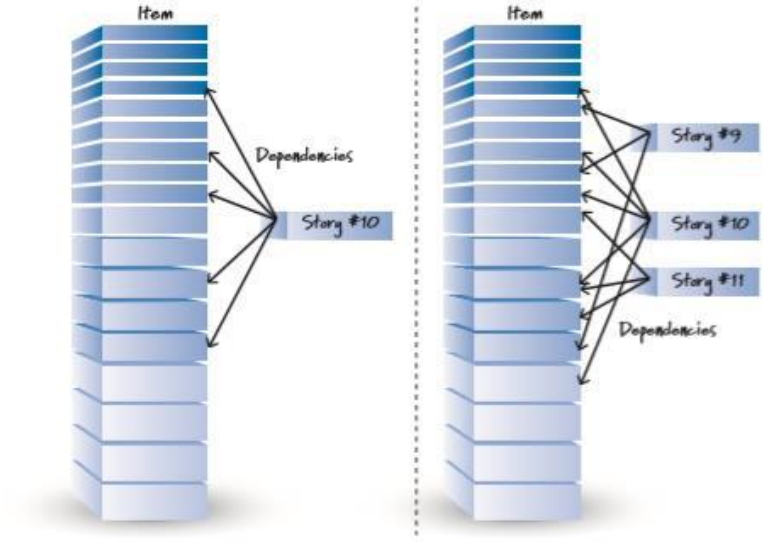
Luego la historia debe contar con todo el detalle y especificación necesaria para la construcción y desarrollo de la misma (Definición de Ready).

¿Cómo se si estoy escribiendo una buena historia de usuario?

Algunos autores definen 6 criterios para saber si estoy escribiendo una buena historia de usuario:

La historia debe ser*: Independiente, negociable, valiosa, estimable, pequeño (tamaño apropiado) y testeable*.

Independiente: Tanto como sea práctico, las historias de usuario deben ser independientes o al menos estar solo vagamente unidas entre sí. Las historias que exhiben un alto grado de interdependencia complican la estimación, la priorización y la planificación. Por ejemplo, si la historia de usuario # 10 depende de muchas otras historias. Antes de que podamos trabajar en la historia # 10, primero debemos desarrollar todas las historias dependientes. En este caso, eso podría no ser tan malo. Sin embargo, imagine que tiene muchas historias diferentes con un alto grado de interdependencia. Intentar determinar cómo priorizar todas estas historias y decidir en qué historias trabajar en un sprint sería difícil de decir lo menos. Al aplicar los criterios independientes, el objetivo no es eliminar todas las dependencias, sino escribir historias de una manera que minimice las dependencias.



Negociable: Los detalles de las historias también deben ser negociables. Las historias no son un contrato escrito en forma de un documento de requisitos por adelantado. En cambio, las historias son marcadores de posición para las conversaciones donde se negociarán los detalles. Las buenas historias capturan claramente la esencia de qué funcionalidad empresarial se desea y por qué se desea. Sin embargo, dejan espacio para que el propietario del producto, los stakeholders y el equipo negocien los detalles. Esta negociabilidad ayuda a todos los involucrados a evitar la mentalidad de nosotros contra ellos, señalando con los dedos que es común con documentos detallados de requisitos por adelantado. Cuando las historias son negociables, los desarrolladores realmente no pueden decir: "Oye, si lo hubieras querido, deberías haberlo incluido en el documento", porque los detalles se negociarán con los desarrolladores. Y la gente de negocios realmente no puede decir: "Oye, obviamente no entendiste el documento de requisitos porque construiste algo incorrecto", porque la gente de negocios mantendrá un diálogo frecuente con los desarrolladores para asegurarse de que haya claridad compartida. Escribir historias negociables evita los problemas asociados con los requisitos detallados por adelantado al dejar en claro que es necesario un diálogo. Un ejemplo común de dónde se viola la negociabilidad es cuando el propietario del producto le dice al equipo cómo implementar una historia. Las historias deben ser sobre qué y por qué, no cómo. Cuando el cómo se vuelve no negociable, las oportunidades para que el equipo sea innovador disminuyen. El desperdicio de innovación resultante podría tener consecuencias económicas devastadoras. Sin embargo, hay momentos en que la forma en que se construye algo es realmente importante para el propietario del producto. Por ejemplo, puede haber una obligación regulatoria de desarrollar una característica de una manera particular, o puede haber una restricción comercial que dirija el uso de una tecnología específica. En tales casos, las historias serán un poco menos negociables porque se requiere algún aspecto del "cómo". Está bien; No todas las historias son totalmente negociables, pero la mayoría de las historias deberían serlo.

Valiosa: Las historias deben ser valiosas para un cliente, usuario o ambos. Los clientes (o selectores) seleccionan y pagan el producto. Los usuarios realmente usan el producto. Si una historia no es valiosa para ninguno de los dos, no pertenece a la cartera de pedidos del producto. No puedo imaginar decir: "La historia # 10 no es valiosa para nadie, pero vamos a construirla de todos modos". No haríamos eso. Reescribiríamos la historia para hacerla valiosa para un cliente o usuario, o simplemente la descartaríamos. ¿Qué tal historias que son valiosas para los desarrolladores pero que no tienen un valor obvio para los clientes o usuarios? ¿Está bien tener historias técnicas como la que se muestra en la Figura? *El problema fundamental con las historias técnicas es que el propietario del producto podría no percibir ningún valor en ellas, lo que hace difícil, si no imposible, priorizarlas frente a historias valiosas para el negocio.*

Migrar a nueva versión de Oracle.

Como desarrollador, quiero migrar el sistema para que funcione con la última versión del DBMS de Oracle para que no estemos operando en una versión que Oracle pronto retirará.

Para que exista una historia técnica, el propietario del producto debe comprender por qué está pagando y, por lo tanto, qué valor entregará en última instancia. En el caso de la historia "Migrar a una nueva versión de Oracle", el propietario del producto podría no entender inicialmente por qué es valioso cambiar las bases de datos. Sin embargo, una vez que el equipo explica los riesgos de continuar desarrollándose en una versión no compatible de una base de datos, el propietario del producto puede decidir que la migración de las bases de datos es lo suficientemente valiosa como para diferir la creación de algunas características nuevas hasta que se complete la migración. Al comprender el valor, el propietario del producto puede tratar la historia técnica como cualquier otra historia valiosa para el negocio y hacer compensaciones informadas. Como resultado, esta historia técnica podría incluirse en la cartera de productos. Sin embargo, en la práctica, la mayoría de las historias técnicas como la de la Figura no deberían incluirse en la cartera de productos. En cambio, este tipo de historias deberían ser tareas asociadas con la realización de historias valiosas para el negocio. Si el equipo de desarrollo tiene una definición sólida de done, no debería haber necesidad de escribir historias como estas, porque el trabajo está implícito en la definición de hecho. El quid de los criterios valiosos es que todas las historias en la cartera de pedidos deben ser valiosas (vale la pena invertir) desde la perspectiva del propietario del producto, que representa las perspectivas del cliente y del usuario. No todas las historias son independientes, y no todas las historias son totalmente negociables, pero todas deben ser valiosas.

Estimable: Las historias deben ser estimables por el equipo que las diseñará, construirá y probará. Las estimaciones proporcionan una indicación del tamaño y, por lo tanto, del esfuerzo y el costo de las historias (las historias más grandes requieren más esfuerzo y, por lo tanto, cuestan más dinero desarrollarlas que las historias más pequeñas). Conocer el tamaño de una historia proporciona información procesable para el equipo Scrum. El propietario del producto, por ejemplo, necesita saber el costo de una historia para determinar su prioridad final en la cartera de productos. El equipo Scrum, por otro lado, puede determinar a partir del tamaño de la historia si se requiere un refinamiento o desagregación adicional. Una gran historia en la que planeamos trabajar pronto tendrá que dividirse en un conjunto de historias más pequeñas. Si el equipo no puede dimensionar una historia, la historia es demasiado grande o ambigua para ser dimensionada, o el equipo no tiene el conocimiento suficiente para estimar un tamaño. Si es demasiado grande, el equipo deberá trabajar con el propietario del producto para dividirlo en historias más manejables. Si el equipo carece de conocimiento, se necesitará alguna forma de actividad exploratoria para adquirir la información.

Tamaño apropiado (pequeño): Las historias deben tener el tamaño adecuado para cuando planeamos trabajar en ellas. Las historias trabajadas en sprints deben ser pequeñas. Si estamos haciendo un sprint de varias semanas, queremos trabajar en varias historias que tengan un tamaño de unos pocos días. Si tenemos un sprint de dos semanas, no queremos una historia de dos semanas, porque el riesgo de no terminar la historia es demasiado grande. Así que, en última instancia, necesitamos historias pequeñas, pero solo porque una historia sea grande, eso no significa que sea mala. Digamos que tenemos una historia de tamaño épico en

la que no estamos planeando trabajar para otro año. Podría decirse que esa historia tiene el tamaño adecuado para cuando planeamos trabajar en ella. De hecho, si pasamos tiempo hoy dividiendo esa épica en una colección de historias más pequeñas, fácilmente podría ser una completa pérdida de nuestro tiempo. Por supuesto, si tenemos una épica en la que queremos trabajar en el próximo sprint, no tiene el tamaño adecuado y tenemos más trabajo por hacer para reducirla. Debe considerar cuándo se trabajará la historia al aplicar este criterio.

Testeable: Las historias deben ser comprobables de forma binaria, pueden aprobar o no pasar sus pruebas asociadas. Ser testeable significa tener buenos criterios de aceptación (relacionados con las condiciones de satisfacción) asociados con la historia, que es el aspecto de "confirmación" de una historia de usuario. Sin criterios comprobables, ¿cómo sabríamos si la historia se realiza al final del sprint? Además, debido a que estas pruebas con frecuencia proporcionan detalles importantes de la historia, pueden ser necesarias antes de que el equipo pueda estimar la historia. Puede que no siempre sea necesario o posible probar una historia. Por ejemplo, las historias de tamaño épico probablemente no tienen pruebas asociadas con ellas, ni las necesitan (no construimos directamente las épicas). Además, en ocasiones puede haber una historia que el propietario del producto considere valiosa, pero puede que no haya una forma práctica de probarla. Es más probable que estos sean requisitos no funcionales, como "Como usuario, quiero que el sistema tenga un tiempo de actividad del 99.999%". Aunque los criterios de aceptación pueden ser claros, puede que no haya un conjunto de pruebas que se puedan ejecutar cuando se pone el sistema en la producción que puede demostrar que se ha cumplido este nivel de tiempo de actividad, pero el requisito sigue siendo valioso, ya que impulsará el diseño.

Requerimientos no funcionales

Los requisitos no funcionales representan restricciones a nivel de sistema. Con frecuencia escribo requisitos no funcionales como historias de usuario, pero no siento obligación de hacerlo, especialmente si parece incómodo o más conveniente escribirlos en un formato diferente. Como restricciones a nivel de sistema, los requisitos no funcionales son importantes porque afectan el diseño y las pruebas de la mayoría o todas las historias en la cartera de productos. Por ejemplo, tener un requisito no funcional de "Soporte de navegador web" sería común en cualquier proyecto de sitio web. Cuando el equipo desarrolla las funciones del sitio web, debe asegurarse de que las funciones del sitio funcionen con todos los navegadores especificados. El equipo también debe decidir cuándo probar todos los navegadores. Cada requisito no funcional es un objetivo principal para su inclusión en la definición de hecho del equipo. Si el equipo incluye el requisito no funcional de "Soporte de navegador web" en la definición de hecho, el equipo tendrá que probar cualquier característica nueva agregada en el sprint con todos los navegadores enumerados. Si no funciona con todos ellos, la historia no está terminada. Recomiendo que los equipos intenten incluir la mayor cantidad posible de requisitos no funcionales en sus definiciones de hecho o en los criterios de aceptación de una historia. Esperar para probar los requisitos no funcionales hasta el final del esfuerzo de

desarrollo difiere obtener comentarios rápidos sobre las características críticas del rendimiento del sistema.

Por lo general tener anotado el requerimiento no funcional da pie a la construcción de una historia o una característica del producto

Definition of Ready

La preparación del backlog del producto debe garantizar que los elementos en la parte superior del backlog estén listos para pasar a un sprint para que el equipo de desarrollo pueda comprometerse y completarlos con confianza al final de un sprint. Algunos equipos de Scrum formalizan esta idea estableciendo una definición de listo. Puede pensar en la definición de listo y la definición de hecho como dos estados de elementos de la cartera de productos durante un ciclo de sprint. Tanto la definición de hecho como la definición de listo son listas de verificación del trabajo que debe completarse antes de que un elemento de la cartera de pedidos del producto pueda considerarse en el estado respectivo. Una definición sólida de listo mejorará sustancialmente las posibilidades del equipo Scrum de cumplir con éxito su objetivo de sprint.

Definition of Done

El resultado de cada sprint debe ser un incremento de producto potencialmente deployable o implementable. Dijimos que "potencialmente implementable" no significa que lo que se construyó debe deployarse realmente. El deploy es una decisión comercial que a menudo ocurre con una cadencia diferente; en algunas organizaciones puede no tener sentido deployar al final de cada sprint. El deploy potencial se considera mejor como un estado de confianza de que lo que se construyó en el sprint se hace realmente, lo que significa que no hay un trabajo importante sin hacer (como pruebas o integración importantes, etc.) que deba completarse antes de que podamos implementar los resultados del sprint, si el deploy fuera nuestro deseo comercial. Para determinar si lo que se produjo es potencialmente implementable, el equipo Scrum debe tener una definición bien definida y acordada de hecho (DoD). Conceptualmente, la definición de hecho es una lista de verificación de los tipos de trabajo que se espera que el equipo complete con éxito antes de que pueda declarar que su trabajo es potencialmente implementable. Obviamente, los elementos específicos en la lista de verificación dependerán de una serie de variables:

- La naturaleza del producto que se está construyendo
- Las tecnologías que se utilizan para construirlo
- La organización que lo está construyendo
- Los impedimentos actuales que afectan lo que es posible

La mayoría de las veces, una definición mínima de hecho debería producir una porción completa de la funcionalidad del producto, una que ha sido diseñada, construida, integrada, probada y documentada y que ofrecería un valor validado para el cliente. Sin embargo, para tener una lista de verificación útil, estos elementos de trabajo de mayor nivel deben ser refinados aún más.

La DoD puede variar con el tiempo, la DoD es algo consensuado por todo el equipo.

Una DoD de un equipo Scrum dice que además de x cosas y las pruebas de siempre la nueva funcionalidad debe ser probada por 6 beta users, al tiempo el negocio no quiere dar el tiempo de los beta users o bien se decide que sus pruebas son innecesarias por x motivo por lo tanto se elimina este paso de la definición de hecho.

Cuando un elemento de la Lista de Producto o un Incremento se describe como “Terminado”, todo el mundo debe entender lo que significa “Terminado”. Aunque esto puede variar significativamente para cada Equipo Scrum, los miembros del Equipo deben tener un entendimiento compartido de lo que significa que el trabajo esté completado para asegurar la transparencia. Esta es la definición de “Terminado” para el Equipo Scrum y se utiliza para evaluar cuándo se ha completado el trabajo sobre el Incremento de producto.

A medida que los Equipos Scrum maduran, se espera que su definición de “Terminado” se amplíe para incluir criterios más rigurosos para una mayor calidad. El uso de las nuevas definiciones puede descubrir trabajo por hacer en los incrementos previamente “Terminados”. Cualquier producto o sistema debería tener una definición de “Terminado” que es un estándar para cualquier trabajo realizado sobre él.

Definición del Product Roadmap (Hoja de ruta del producto)

Una vez que tenemos la visión inicial y una cartera de productos de alto nivel, podemos definir nuestra hoja de ruta inicial del producto, una serie de lanzamientos para lograr parte o la totalidad de nuestra visión del producto. Cuando usamos Scrum, siempre desarrollamos de forma incremental. También tratamos de implementar de manera incremental cuando ese enfoque es razonable, lo que significa que nos enfocamos en lanzamientos más pequeños y más frecuentes donde entregamos parte de la solución antes de entregar toda la solución. *Una hoja de ruta del producto es una descripción general inicial de estas implementaciones incrementales*. Por supuesto, si estamos planeando solo un lanzamiento pequeño, no necesitamos una hoja de ruta del producto. La publicación frecuente no significa que establezcamos plazos demasiado agresivos; dichos plazos suelen dar lugar a fechas perdidas. En cambio, enfocamos cada lanzamiento en un pequeño conjunto de características mínimas liberables (MVP – Minimun Viable Product) alrededor del cual la comunidad de partes interesadas comparte un fuerte consenso grupal. Los MVP representan el conjunto más pequeño de características "imprescindibles", las características que simplemente tienen que estar en la versión si queremos cumplir con las expectativas de valor y calidad del cliente. Si bien podríamos elegir entregar más que los MVP en un lanzamiento dado, los clientes no

percibirían suficiente valor si entregáramos menos. Por lo tanto, siempre es importante definir el conjunto mínimo. Para complementar los MVP, algunas organizaciones usan la estrategia de lanzamientos fijos y periódicos, por ejemplo, un lanzamiento cada trimestre, para simplificar la hoja de ruta del producto. Este enfoque tiene varias ventajas. Primero, es fácil de entender y proporciona a todos los involucrados (interna y externamente) versiones predecibles. También establece un ritmo, o cadencia, para liberar que ayuda a reunir recursos de una manera predecible y permite que grupos dispares sincronicen sus planes. Si usamos esta estrategia, aún determinamos los MVP para cada versión. Si los MVP requieren menos tiempo para desarrollarse que el tiempo fijo para el lanzamiento, se crearán algunas características adicionales de alto valor. Los lanzamientos periódicos fijos pueden no ser siempre aplicables si eventos externos (como una conferencia o una fecha de lanzamiento fija de un producto de marca compartida) están impulsando los lanzamientos, pero sus beneficios hacen que valga la pena considerarlo. Cada lanzamiento en la hoja de ruta debe tener un objetivo de lanzamiento claramente definido que comunique el propósito y el resultado deseado del lanzamiento. Se crea un objetivo de lanzamiento al considerar muchos factores, incluidos los clientes objetivo, problemas arquitectónicos de alto nivel, eventos importantes en el mercado, etc. Al crear una hoja de ruta de productos, debemos considerar a los clientes y cómo pueden segmentarse en diferentes mercados. La hoja de ruta debe expresar cómo y cuándo abordar estos diferentes segmentos de clientes.

Al hacer una hoja de ruta del producto, también debemos considerar cuestiones arquitectónicas o tecnológicas de alto nivel. Al definir una hoja de ruta del producto, también podríamos tener que permitir cualquier evento de mercado significativo que pueda influir en el momento de nuestras entregas de funciones. Review Everything, por ejemplo, siempre asiste a la conferencia anual de Social Media Expo. Roger y las partes interesadas están de acuerdo en que tener un lanzamiento disponible para la conferencia de este año (a unos tres meses de distancia) sería un gran lugar para recibir comentarios sobre el servicio. Nuestro objetivo al crear la hoja de ruta del producto es considerar cualquier factor que consideremos relevante para ayudarnos a definir un conjunto objetivo de lanzamientos para nuestra solución. Sin embargo, recuerde que esta hoja de ruta es simplemente una primera aproximación de uno o algunos lanzamientos cercanos. Debemos tener derecho a actualizar la hoja de ruta a medida que haya mejor información disponible. También debemos considerar qué tan lejos en el futuro se extenderá nuestra hoja de ruta del producto. Aunque nuestra visión puede ser lo suficientemente grande y audaz como para requerir muchos años para realizarla por completo, es poco probable que intentemos producir una hoja de ruta detallada que se extienda por completo a través de dicha visión. Cuando usamos Scrum, producimos la hoja de ruta del producto tan lejos en el futuro como sea razonable y deseable. Hasta qué punto debe extenderse su hoja de ruta dependerá de sus circunstancias particulares. Como mínimo, su hoja de ruta probablemente deba cubrir al menos el período de tiempo que le está pidiendo a la gente que financie.

En un ambiente real de trabajo los releases probablemente no seán de MVPs completos sino partes de dichos MVPs. Cada MVP será un checkpoint luego de varios releases en nuestro Roadmap.

User Story Mapping

User Story Mapping es una técnica que consiste en organizar el *Product Backlog* en dos dimensiones con el objetivo de construir un Roadmap. Este mapa se compone de dos ejes, en uno de ellos las releases (vertical) y en el otro las funcionalidades (horizontal).

Es por ello que tras la fase de Agile Inception o Sprint 0, vamos a construir el mapa utilizando un tablero de post-its, en el que identificamos la prioridad de las historias de usuario, y el orden de implementación agrupado en releases. Esto permite mejorar la visión de producto y construir una hoja de ruta clara y compartida por el equipo.

Los pasos necesarios para construir nuestro **User Story Mapping** son:

1. Identificar objetivos: Lo primero que debemos de realizar es **definir nuestros objetivos**, para ello escribimos en tarjetas las ideas a muy alto nivel, de lo que representa el producto para negocio.

2. Identificar Actividades: A continuación debemos de identificar las actividades de alto nivel a partir de los objetivos definidos en el punto anterior. Estas actividades denominadas **"épicas"**, deben de posicionarse horizontalmente de tal manera que sigan un orden literario, como si de un storytelling se tratara. Esto nos proporciona una primera foto del producto.

3. Describir el timeline: A continuación, seguimos **descomponiendo las épicas en historias de usuario** más pequeñas, que sean más concretas y definan funcionalidades más específicas. Estas historias compondrán nuestras releases, las que iremos seleccionando en cada Sprint.

Para ello se colocan las historias debajo de su épica y las ordenamos en un orden lógico desde la perspectiva del usuario.

4. Ordenar las historias: Una vez descompuestas y ordenadas debemos desplazarlas a lo largo del eje vertical, quedando arriba las más importantes y abajo las menos importantes.

5. Identificar Releases: Una vez descompuestas y ordenadas, debemos de focalizarnos en las historias superiores que definirán el conjunto de funcionalidades necesarias para **definir el mínimo producto viable** (MVP). De esta manera trazamos líneas horizontales, agrupando las historias de usuario en "releases" que marcarán nuestra hoja de ruta y aquellas entregas de funcionalidad que debemos de ir desarrollando. Se pueden mover historias para que el "release" tenga sentido y sea consistente.

La técnica de **User Story Mapping**, permite de una manera visual y colaborativa definir el mapa de funcionalidades de un producto. De un solo vistazo, tenemos el '*big picture*' de nuestro sistema en una sola instantánea, descompuesto de las ideas más generales a las más

específicas. Además, la dinámica del ordenar por prioridad, nos permite establecer las necesidades del negocio.

Una vez que tenemos plasmado nuestro mapa es muy sencillo agrupar las funcionalidades en releases para tener una hoja de ruta que seguir, aunque este proceso no debe ser estático, si no que nuestro mapping puede y debe evolucionar en función de las necesidades del proyecto.

https://www.youtube.com/watch?v=k_4SchJgAI4

Technical DEBT

La deuda técnica es un concepto en el desarrollo de software que refleja el costo implícito del retrabajo adicional causado por elegir una solución fácil o limitada ahora en lugar de utilizar un mejor enfoque que tomaría más tiempo.

Hay autores que afirman que la deuda técnica empieza al momento que el primer entregable ve la luz.

Hoy en día, la deuda técnica se refiere tanto a los atajos que tomamos deliberadamente como a las muchas cosas malas que afectan a los sistemas de software. Éstos incluyen:

 Diseño no apto (malo): un diseño que alguna vez tuvo sentido pero que ya no lo hace, dados los cambios importantes en el negocio o las tecnologías que ahora usamos

 Defectos: problemas conocidos en el software que aún no hemos invertido tiempo en eliminar

 Pruebas insuficientes cobertura: áreas donde sabemos que deberíamos hacer más pruebas, pero no lo hacemos

 Pruebas manuales excesivas: pruebas a mano cuando realmente deberíamos tener pruebas automatizadas

 Mala integración y administración de versiones: realizar estas actividades de una manera que requiere mucho tiempo y es propenso a errores

 Falta de experiencia en la plataforma, por ejemplo, tenemos aplicaciones de mainframe escritas en COBOL pero ya no tenemos muchos programadores COBOL experimentados

 Y muchos más, porque el término deuda técnica actual se usa realmente como marcador de posición para un problema multidimensional

Además, existe una deuda técnica inevitable, que generalmente es impredecible. Por ejemplo, nuestra comprensión de lo que hace un buen diseño surge de hacer el trabajo de diseño y construir características valiosas para el usuario. No podemos predecir perfectamente por adelantado cómo nuestro producto y su diseño deberán evolucionar con el tiempo. Por lo

tanto, es posible que las decisiones de diseño e implementación que tomamos al principio deban cambiar a medida que cerramos bucles de aprendizaje importantes y adquirimos aprendizaje validado. Los cambios requeridos en las áreas afectadas son deudas técnicas inevitables.

Principales consecuencias de la deuda técnica

Mayor tiempo de entrega: asumir una deuda técnica significa tomar un préstamo hoy contra el tiempo requerido para hacer un trabajo futuro. Cuanto mayor sea la deuda hoy, más lenta será la velocidad mañana. Cuando la velocidad disminuye, lleva más tiempo entregar nuevas características y soluciones de productos a los clientes. Entonces, en presencia de una alta deuda técnica, el tiempo entre los entregables en realidad aumenta en lugar de disminuir. En mercados cada vez más competitivos, la deuda técnica está trabajando activamente en contra de nuestros mejores intereses.

Número significativo de defectos: los productos con una deuda técnica significativa se vuelven más complejos, lo que dificulta hacer las cosas correctamente. Los defectos compuestos pueden causar fallas críticas del producto con frecuencia alarmante. Estas fallas se convierten en una interrupción importante para el flujo normal del trabajo de desarrollo de valor agregado. Además, la sobrecarga de tener que manejar muchos defectos consume el tiempo disponible para producir funciones de valor agregado. En algún momento, comenzamos a ahogarnos, pero estamos tan ocupados pisando aguas llenas de defectos que no podemos ver cómo salir del desastre en el que nos encontramos.

Aumento de los costos de desarrollo y soporte: a medida que aumenta la deuda técnica, los costos de desarrollo y soporte comienzan a aumentar. Lo que solía ser simple y barato de hacer ahora es complicado y costoso. En presencia de niveles crecientes de deuda técnica, incluso los pequeños cambios se vuelven muy caros. Además, el aumento de los costos puede cambiar la economía de si proceder con una reparación de características o defectos. Una característica que podría construirse (o un defecto que podría repararse) a un bajo costo en presencia de una baja deuda técnica podría ser demasiado costosa en presencia de una alta deuda técnica. Como resultado del aumento de los costos, nuestros productos se vuelven menos adaptables al entorno en evolución en el que deben existir.

Previsibilidad disminuida: para un producto con altos niveles de deuda técnica, es casi imposible hacer cualquier tipo de predicción. Por ejemplo, las estimaciones se convierten en malas estimaciones incluso para los miembros del equipo más experimentados. Simplemente hay demasiada incertidumbre en torno a cuánto tiempo puede tomar algo cuando se trata de un producto endeudado. En consecuencia, nuestra capacidad para hacer compromisos y tener una expectativa razonable de cumplirlos se ve seriamente afectada

Bajo rendimiento: Lamentablemente, a medida que aumenta la deuda técnica, las personas esperan un rendimiento de desarrollo cada vez más bajo y, por lo tanto, reducen sus expectativas de lo que es posible. Por supuesto, las expectativas reducidas comienzan a

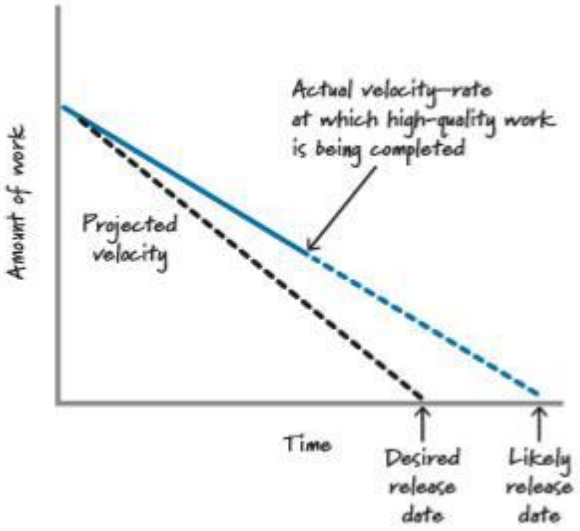
propagarse a través de la cadena de valor, lo que resulta en un rendimiento general más bajo en toda la organización.

Frustración universal: la desafortunada consecuencia humana de la alta deuda técnica es que todos en la cadena de valor se frustran. La acumulación de todos esos atajos pequeños pero molestos hace que el trabajo en el producto sea doloroso. Finalmente, la alegría en el desarrollo desaparece y se reemplaza con la rutina diaria de los problemas de lucha que nadie quiere (o debería tener que enfrentar). La gente se quema. Los miembros conocedores del equipo de desarrollo comienzan a partir para buscar oportunidades más gratificantes; y, como son los que están en la mejor posición para hacer algo sobre el problema de la deuda, su partida empeora las cosas para los que quedan. La moral desciende en espiral hacia abajo con intensidad creciente. La deuda técnica no le quita la alegría solo a las personas técnicas; tiene el mismo efecto en la gente de negocios. ¿Cuánto tiempo queremos seguir haciendo compromisos comerciales que no se pueden cumplir? ¿Y qué pasa con nuestros clientes pobres, que están tratando de administrar sus negocios además de nuestro producto endeudado? Ellos también se cansan rápidamente de las repetidas fallas del producto y de nuestra incapacidad para cumplir las promesas que hacemos. La confianza que una vez existió a través de la cadena de valor se reemplaza con frustración y resentimiento.

Disminución de la satisfacción del cliente: la satisfacción del cliente disminuirá a medida que aumente la frustración del cliente. Por lo tanto, el alcance del daño causado por la deuda técnica no está solo aislado para el equipo de desarrollo o incluso para la organización de desarrollo en su conjunto. Peor aún, las consecuencias de la deuda técnica pueden afectar sustancialmente a nuestros clientes y su percepción de nosotros.

Causas de la deuda técnica

Presión para cumplir una fecha límite: Tanto la deuda técnica estratégica como la ingenua a menudo se ven impulsadas por la presión empresarial para cumplir con una fecha límite inminente (ver Figura). La dimensión vertical representa la cantidad de trabajo que queremos realizar en una fecha de lanzamiento deseada (que se muestra en la dimensión horizontal). La línea entre la cantidad de trabajo y la fecha de lanzamiento deseada representa la velocidad proyectada constante a la que se debe completar el trabajo para cumplir con la fecha de lanzamiento deseada. Al trabajar a la velocidad proyectada, nuestro objetivo es completar características de alta calidad de manera oportuna y minimizar la acumulación de deuda técnica. Sin embargo, a medida que comenzamos a hacer el trabajo, la velocidad real necesaria para producir resultados de alta calidad es más lenta que la velocidad proyectada. Si continuamos produciendo resultados a la velocidad real, perderemos la fecha de lanzamiento deseada y terminaremos en la fecha de lanzamiento probable.

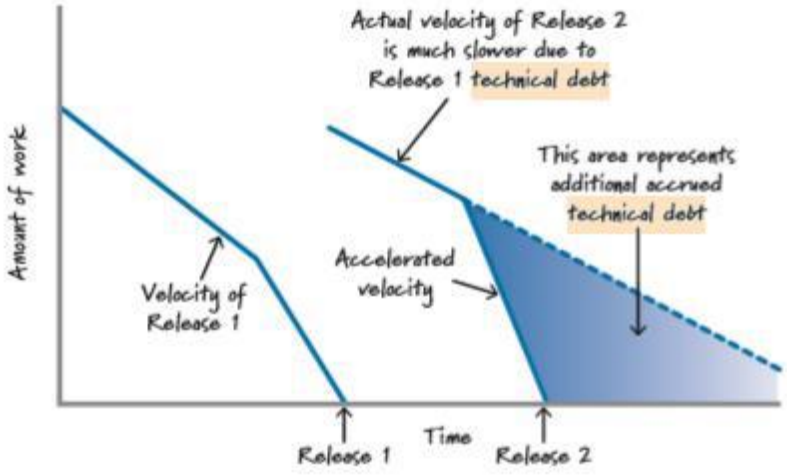


Intentando acelerar la velocidad falsamente: en este punto tenemos que tomar una decisión comercial. ¿Queremos reducir el alcance para cumplir con la fecha de lanzamiento deseada o deseamos agregar más tiempo al cronograma para acomodar una entrega en la fecha de lanzamiento probable? Desafortunadamente, en muchas circunstancias, la empresa rechaza ambas opciones y decreta que el equipo debe cumplir con la fecha de lanzamiento deseada con todas las características. En esta situación, se le dice al equipo que hace el trabajo que acelere su velocidad para alcanzar la fecha de lanzamiento deseada. Al trabajar a esta velocidad acelerada, el equipo tendrá que tomar decisiones deliberadas para asumir la deuda técnica (lo que significa que tendrán que tomar atajos para trabajar lo suficientemente rápido como para cumplir con la fecha de lanzamiento deseada). Quizás el diseño no sea tan bueno como debería ser, o se diferirán los tipos específicos de pruebas (quizás pruebas de carga). Como resultado, acumularemos deuda técnica, todo el trabajo que deberíamos haber hecho pero que no tuvimos tiempo de hacer.

Mito: Menos pruebas pueden acelerar la velocidad: un mito frecuente es que las pruebas son gastos generales adicionales, y al reducirlas, podemos acelerar la velocidad. La realidad es que reducir las pruebas aumentará la deuda y nos hará ir más despacio, porque los problemas pasarán desapercibidos hasta más tarde, cuando es mucho más tiempo consumirlos para solucionarlos. Los equipos experimentados entregan resultados de buena calidad más rápido y con menos deudas técnicas cuando las pruebas se integran fundamentalmente en el proceso de desarrollo. Estos equipos utilizan buenas prácticas técnicas, como el desarrollo basado en pruebas (TDD), donde el desarrollador escribe y automatiza una pequeña prueba de unidad antes de escribir el pequeño fragmento de código que hará pasar la prueba.

La deuda construida sobre deuda: la deuda técnica futura se acumula rápidamente sobre la deuda técnica existente. Y, a medida que la deuda técnica comienza a acumularse, comienzan a aparecer consecuencias económicamente perjudiciales. La Figura ilustra las consecuencias de construir la Versión 2 sobre la deuda técnica de la Versión 1. En la Figura 8.6, la velocidad real durante la Versión 2 es más lenta que en la Versión 1. Está claro que a esta velocidad

volveremos a perder La fecha de lanzamiento prevista. Y, una vez más, el negocio insiste en que el equipo cumpla con la fecha de lanzamiento deseada con todas las características.



Como resultado, acumulamos aún más deuda técnica. Si este patrón continúa, eventualmente la línea de velocidad podría volverse horizontal. Este sería un estado donde la deuda técnica en el sistema es tan alta que nuestra velocidad efectiva es cero. El resultado es el tipo de producto en el que nos aterra realizar cualquier cambio, porque un pequeño cambio en un área podría causar que otras 18 cosas se rompan en lo que parecen ser áreas totalmente no relacionadas del producto. Peor aún, no hay forma de predecir que esas 18 cosas específicas se romperían. Y, por supuesto, no tenemos un marco de prueba apreciable que nos ayude a determinar cuándo se rompen, pero, no se preocupe, ¡nuestros clientes seguramente nos lo harán saber! Una vez que nos encontramos en una situación con una alta deuda técnica, todas las elecciones se convierten en malas elecciones:

 No hacer nada y el problema empeora.

 Realice inversiones cada vez mayores en la reducción de la deuda técnica que puede consumir más y más de nuestros valiosos recursos de desarrollo de productos.

 Declare bancarrota técnica, retire la deuda técnica y reemplace el producto en deuda con un nuevo producto con el costo y riesgo total de desarrollar un nuevo producto.

Con opciones como estas en el horizonte, es fundamental que gestionemos adecuadamente nuestra deuda técnica antes de que se salga de control.

Gestionar la acumulación de deuda técnica

Utilice buenas prácticas técnicas: el primer enfoque para gestionar la acumulación de deuda técnica es dejar de agregar deuda ingenua a nuestros productos. El uso de buenas prácticas técnicas es un excelente punto de partida. Aunque Scrum no define formalmente las prácticas técnicas, cada equipo exitoso de Scrum que he visto emplea prácticas como el diseño simple,

el desarrollo basado en pruebas, la integración continua, las pruebas automatizadas, la refactorización, etc. La comprensión y el uso proactivo de estas prácticas ayudará a los equipos a dejar de agregar muchas formas de deuda ingenua a sus productos. En el caso de la deuda técnica acumulada, la refactorización del código es una herramienta importante para pagarla. La refactorización es una técnica disciplinada para reestructurar un cuerpo de código existente, alterando su estructura interna sin cambiar su comportamiento externo. En otras palabras, limpiamos debajo del capó, pero desde la perspectiva del cliente, el producto sigue funcionando igual. Al refactorizar, nos esforzamos por reducir la complejidad al tiempo que mejoramos la capacidad de mantenimiento y la extensibilidad. El resultado de la refactorización es facilitar el trabajo en cuestión (el equivalente a reducir los pagos de intereses). Cunningham (2011) explica los beneficios de la refactorización con el ejemplo:

. . . el cliente está dispuesto a pagar por una nueva función; la función no encaja; reorganice el código para que encaje; ahora la característica es fácil de implementar. Esto podría llamarse refactorización justo a tiempo. Explicaría esto a la gerencia de la siguiente manera: esperamos tener un lugar en nuestro software para cada nueva solicitud. Pero a veces no tenemos un lugar para una función, por lo que primero tenemos que hacer el lugar y luego implementar la función. . . .

Use una definición fuerte de hecho: el trabajo que deberíamos haber realizado cuando se creó una característica, pero que terminó aplazando hasta un momento posterior, es una causa importante de deuda técnica. Al usar Scrum, queremos una definición sólida de hecho para ayudar a guiar al equipo a una solución de bajo o sin deuda al final de cada sprint. Mientras más técnicamente abarcamos nuestra lista de verificación de definición de hecho, es menos probable que acumulemos deudas técnicas. Y, muchas veces el costo de pagar la deuda técnica que pasa de una definición débil de hecho es sustancialmente mayor que abordarlo durante el sprint. Operar sin una definición sólida de done es como otorgar una licencia para acumular deuda técnica.

No toda la deuda técnica debe ser reparada

Ejemplos:

- Producto cerca del final de la vida útil

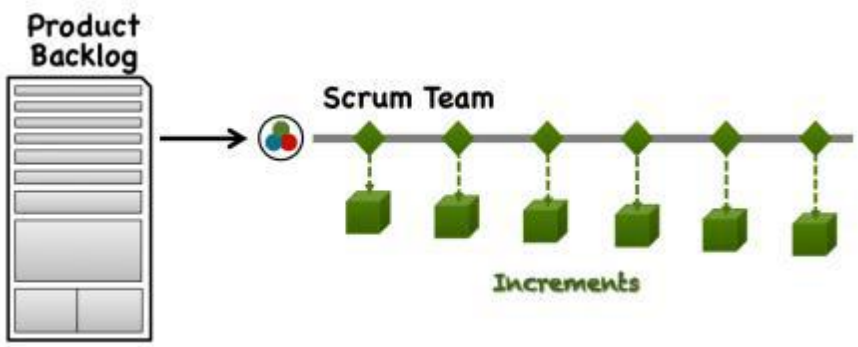
- Prototipo desechable

- Producto construido para una vida corta

Scaled Professional Scrum

Scrum escalado

Una sola instancia de Scrum tiene un equipo Scrum que funciona desde una cartera de productos. El equipo corre contra elementos seleccionados de la Lista de Producto y crea un Incremento de producto liberable en el final de cada Sprint.



Crear software liberable cada 30 días, o menos, es un desafío, incluso con un equipo. Requiere disciplina y rigor, un enfoque extremo en prácticas de desarrollo, personas, comunicación y colaboración y eliminación de impedimentos que limitan el progreso y la creatividad.

La creación de software integrado y liberable con múltiples equipos Scrum se convierte fácilmente en una gestión y drama técnico, dada la necesidad de sofisticación adicional; en técnicas, herramientas, integración, y colaboración.

Las razones para escalar al nivel de múltiples equipos que construyen conjuntamente un sistema incluyen:

 El deseo de completar más funcionalidades dentro de un período de tiempo determinado.

 El deseo de completar la funcionalidad prevista más rápidamente.

 Resolver una escasez temporal de habilidades y personas especializadas al acomodar a las personas, para uno o más Sprints, en equipos Scrum específicos.

 Cada combinación de razones anteriores.

Los líderes de un esfuerzo de escalamiento deben poder responder preguntas esenciales:

 ¿Con qué frecuencia se debe liberar el trabajo?

 ¿Qué técnicas se utilizarán para integrar el trabajo a esa frecuencia?

 ¿Qué se hará para medir y gestionar el trabajo y dicha integración?

 ¿En qué gastos generales se incurre para lograr esta integración y entrega?

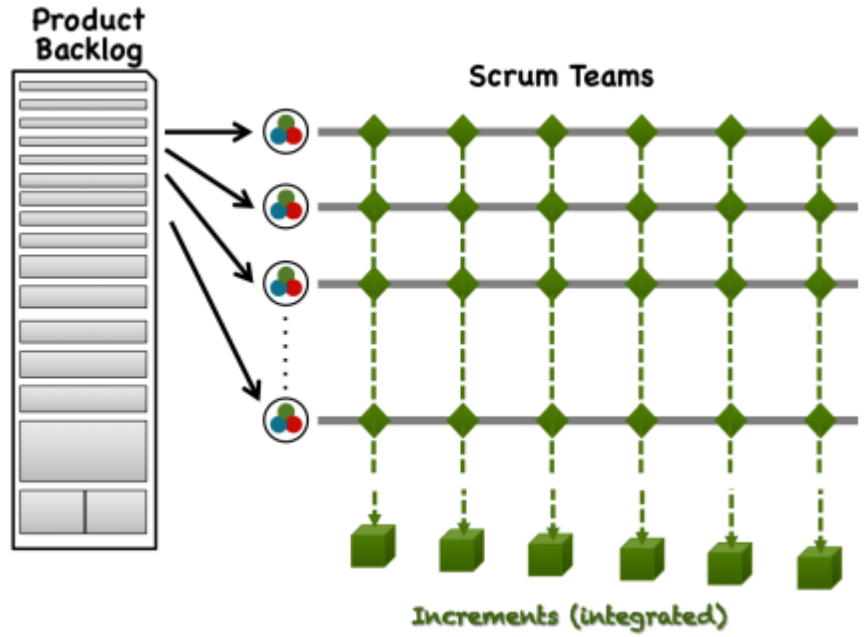
 ¿Pueden los costos y beneficios de la entrega justificarse por un aumento en el valor?

 ¿Cómo se reduce sistemáticamente el costo?

Scaled Professional Scrum

Desde el exterior, una implementación a escala de Scrum se ve exactamente como Scrum (singular). Un producto la cartera de pedidos proporciona información para un Sprint. Al final del Sprint, un Incremento de producto liberable es disponible. Una implementación de Scrum escalada es diferente en que, para uno o más Sprints, el Producto

El propietario emplea a más de un equipo Scrum. El número de equipos Scrum puede ser constante y su la composición puede permanecer igual o no.



Scrum escalado no es "Scrum" si no todos los atributos del elemento más pequeño, es decir, un Scrum singular, el equipo que emplea Scrum de manera profesional se encuentra en la suma de todos los elementos. Se aplican los valores básicos, principios, artefactos, roles y reuniones de Scrum, ya sea que Scrum sea singular o escalado. Estos elementos esenciales siguen siendo inviolables para controlar el riesgo, generar creatividad, y creando transparencia. El objetivo principal sigue siendo que el software de trabajo se pueda liberar sin dependencias no resueltas

Sin embargo, cuando más equipos Scrum trabajan juntos en una sola cartera de productos, el número de interacciones, complejidades y eventos no lineales es mucho mayor. Esto tiene un costo. La relación entre el número de equipos Scrum, el aumento en el costo y el aumento en la productividad no es lineal. Los propietarios de productos que emplean múltiples equipos Scrum deben sopesar cuidadosamente los beneficios versus Los costos adicionales incurridos.

Cuando más de un equipo Scrum usa Scrum para desarrollar el mismo software, que se conoce como "Scrum escalado". Cuando el empirismo y la creación de conocimiento ascendente se emplea a escala, transparencia y la excelencia técnica se adoptan, y los valores subyacentes

de Scrum y los principios se promulgan, es "Scrum profesional escalado". Solamente Scrum profesional escala los beneficios de Scrum. Mecánico o Scrum aficionado (algunos dicen "Scrum zombi"), caracterizado por la falta de estándares de ingeniería adecuados y falta de integración, no escala, como dependencias excesivas permanecen sin resolver, de forma permanente.

La creación de una implementación a escala de Scrum profesional requiere esfuerzos concertados de varios niveles para beneficiarse del empirismo de Scrum y aumentar la agilidad de la organización, la capacidad de no solo responder al cambio, pero aprovechar de manera oportunista la incertidumbre y la turbulencia.

El marco Scaled Professional Scrum proporciona una base desde la cual puede crecer la escala, de alguna manera que se pueden abordar los desafíos únicos de organizaciones y situaciones únicas. Técnicas para organizar y seleccionar elementos de la Lista de Producto, resolver dependencias e integrar el trabajo, y se incluyen la creación de incrementos "Listo". La empresa reutiliza y aumenta las inversiones realizadas en capacitación de personal, gerencia y equipos en Scrum.

Al igual que la ley de Conway establece que la interacción del software es un reflejo de la comunicación de los programadores que escribieron el código, la estructura del desarrollo de productos a escala se reflejará en la estructura, la calidad y el valor del software desarrollado.

Planning Poker

Planning Poker es una técnica basada en el consenso para estimar el esfuerzo. Las personas conocedoras (los expertos) programadas para trabajar en un PBI participan en una discusión intensa para exponer suposiciones, adquirir una comprensión compartida y evaluar el PBI. Planning Poker genera estimaciones de tamaño relativo al agrupar o agrupar con precisión elementos de tamaño similar. El equipo aprovecha su historial de estimación de PBI establecido para estimar más fácilmente el próximo conjunto de PBI.

Para realizar Planning Poker, el equipo debe decidir qué escala o secuencia de números usará para asignar las estimaciones. Debido a que nuestro objetivo es ser correctos y no demasiado precisos, preferimos no usar todos los números. En cambio, preferimos una escala de tamaños con más números en el extremo pequeño del rango y menos números más espaciados en el extremo grande del rango. La escala más utilizada es la propuesta por Mike Cohn, basada en parte en una secuencia de Fibonacci modificada: 1, 2, 3, 5, 8, 13, 20, 40 y 100. Se basa una escala alternativa que utilizan algunos equipos. En potencias de 2: 1, 2, 4, 8, 16, 32,. . . . Cuando se usa este tipo de escala, agrupamos o agrupamos PBI de tamaño similar y les asignamos el mismo número en la escala. Para ilustrar este concepto, digamos que trabajamos en la oficina de correos y necesitamos agrupar paquetes de tamaño similar en el mismo contenedor. Cuando recibimos un paquete, debemos decidir en qué contenedor colocar el paquete. Ahora,

no todos los paquetes en el mismo contenedor tienen o serán idénticamente la misma forma física, tamaño o peso, por lo que debemos examinar los paquetes que están actualmente en los contenedores para que podamos encontrar el contenedor que mejor se ajuste al paquete que estamos estimando. Una vez que encontramos el contenedor coincidente más cercano, colocamos el paquete en el contenedor y pasamos al siguiente paquete. Obviamente, cuantos más paquetes coloquemos en los contenedores, más fácil debería ser dimensionar y agrupar futuros paquetes porque tendremos más puntos de comparación. Para evitar ser demasiado precisos, no tenemos un "contenedor 4" (si estamos usando una escala basada en la secuencia de Fibonacci). Por lo tanto, cuando obtenemos un paquete que consideramos que es más grande que un 2 pero más pequeño que un 8, debemos colocarlo en el "contenedor 3" o en el "contenedor 5".

Las reglas de Planning Poker son las siguientes:

1. El propietario del producto selecciona un PBI para estimar y lee el artículo al equipo.

2. Los miembros del equipo de desarrollo discuten el tema y hacen preguntas aclaratorias al propietario del producto, quien responde las preguntas.

3. Cada estimador selecciona en privado una tarjeta que representa su estimación.

4. Una vez que cada estimador ha hecho una selección privada, todas las estimaciones privadas se exponen simultáneamente a todos los estimadores.

5. Si todos seleccionan la misma tarjeta, tenemos consenso, y ese número de consenso se convierte en la estimación de PBI.

6. Si las estimaciones no son las mismas, los miembros del equipo participan en una discusión enfocada para exponer suposiciones y malentendidos. Por lo general, comenzamos pidiendo a los estimadores altos y bajos que expliquen o justifiquen sus estimaciones.

7. Después de la discusión, volvemos al paso 3 y repetimos hasta llegar a un consenso.

En Planning Poker no tomamos promedios ni usamos ningún número que no esté en la escala / cartas. El objetivo no es comprometer, sino que el equipo de desarrollo llegue a un consenso sobre la estimación del tamaño (esfuerzo) general de la historia desde la perspectiva del equipo. Por lo general, este consenso se puede lograr dentro de dos o tres rondas de votación, durante las cuales la discusión enfocada de los miembros del equipo ayuda a obtener una comprensión compartida de la historia.

Bibliografía:

- La guía de Scrum: Las reglas del juego (Ken Schwaber y Jeff Sutherland) ;
- Essential Scrum A practical Guide to the most popular agile process (Kenneth S. Rubin)