Módulo 2. *Backlog*

A lo largo de este módulo hablaremos mucho de *backlog* y queremos comenzar introduciendo su significado: "es una lista emergente y ordenada de lo que se necesita para mejorar el producto" (Schwaber y Sutherland, 2020, https://bit.ly/3FPUAD4).

El *backlog* del producto es básicamente un listado de ítems (*product backlog ítems*, PBI) que representan las características del producto a construir. Esta lista de PBI es un elemento vivo y emergente, que cambia constantemente a medida que aprendes más y más acerca del producto. Es mantenida y ordenada por el *Product Owner* (PO) y es la única fuente del trabajo que hace el equipo Scrum. Todos los PBI que componen el *Product Backlog* tienen una razón de existir, y esa razón es cumplir con cierto objetivo de producto.

El *backlog* es una pila de trabajo, un conjunto de cosas que aún no están hechas, pero necesitan estar hechas. Podríamos traducirlo como *stock*, acumulación. En agilidad hablaremos mucho de la pila de producto, actividades, necesidades para construir productos (ya sea *software* o cualquiera de ellos). El *Product Owner,* al ser dueño del producto a construir, es, por lo tanto, dueño de ese *backlog*.

Lo primero que diremos sobre el rol del *Product Owner* es que forma parte del equipo Scrum. En particular, es el responsable del producto desde el punto de vista de negocio. Debe velar porque los incrementos que entrega el equipo sean del mayor valor posible en cada *sprint*. El PO es el encargado de la optimización y maximización del valor del producto (Alamio, 2021). Distinguiremos maximizar el valor del producto de gestionar el *product backlog*, dos responsabilidades del PO.

La responsabilidad del PO es garantizar que todos entiendan lo mismo del *Product Backlog*. Esto no significa documentar detalladamente los requerimientos, sino conversar y verificar durante el *sprint* que el incremento que se vaya creando cumple con las expectativas. Al mismo tiempo, coordina y facilita actividades que llamaremos refinamiento, donde junto a los *stakeholders* y los desarrolladores, se involucra activamente en la definición del producto a mediano plazo y el entendimiento en detalle a corto plazo.

Debe ser capaz de poder explicitar los objetivos del producto y comunicar a los *stakeholders* cómo se forma el valor del producto en el que están apostando. Dada la naturaleza colaborativa de Scrum, lo ideal sería que estos objetivos no sean algo impuesto a los demás, sino el resultado de una creación conjunta alrededor de una visión de producto.

Visión de producto

La visión del producto es la descripción del propósito del producto. Debería responder la pregunta: ¿para qué lo quieres construir? O ¿qué quieres lograr? Scrum no hace referencia a una visión de producto, por lo que no podemos decir que sea un artefacto de Scrum, aunque es muy recomendable contar con ella. Por ejemplo, la visión de Spotify es "Ser una plataforma cultural donde los creadores profesionales puedan liberarse de las limitaciones de su medio y donde todos puedan disfrutar de una experiencia artística inmersiva que nos permita sentir empatía entre nosotros y sentirnos parte de un todo mayor". La visión de Google es "proporcionar acceso a la información del mundo con un solo clic".

Las visiones, en general, no suelen medirse, sino que funcionan como horizontes ambiciosos, motivacionales, que guían nuestras prácticas. Para poder medir el progreso hacia la realización eventual de la visión de producto, debemos determinar una consecución de objetivos de producto.

Objetivos de producto

A partir de la visión, el PO establece la estrategia de creación del producto. En esa estrategia trazará los diferentes objetivos del producto a ir alcanzando. Los objetivos de producto sí son parte del *framework* de Scrum. Podríamos deducir entonces que cuando hablamos de objetivos de producto estamos hablando de diferentes hitos de negocio, medibles, que al encadenarse determinan la estrategia de producto hacia una visión. Tanto la visión, como la estrategia, como los objetivos emergen del trabajo colaborativo que involucra a *stakeholders* y miembros del equipo Scrum.

El objetivo del producto sí está definido dentro de Scrum. Es un hito futuro que deseamos lograr a través del producto, pero no mide alcance sino resultado. Por ejemplo, incrementar un 200 % las suscripciones de clientes de Asia a las categorías plus y prémium.

El camino hacia la visión de producto podrá contar con una serie de objetivos de producto diferentes a lo largo del tiempo. El equipo Scrum deberá alcanzar un objetivo (o abandonarlo) antes de perseguir el siguiente.

El PO es responsable de desarrollar y comunicar explícitamente los objetivos del producto. Dada la naturaleza colaborativa de Scrum, lo ideal sería que estos objetivos no sean algo impuesto a los demás, sino el resultado de una creación conjunta alrededor de una visión de producto.

La visión de producto establece el escenario futuro que se quiere lograr con el producto. Esta visión es típicamente utópica e inspiradora y determina la dirección,

pero difícilmente ayuda a medir el progreso. La visión de producto no es parte del *framework* de Scrum.

En función de lo descrito en el módulo 1, es importante resaltar que como PO garantizas el entendimiento del *Product Backlog*. El trabajo del *Product Owner* se desarrolla durante todo el día, todos los días del *sprint*. Siguiendo con esta descripción, tal como indicamos anteriormente, esto no significa documentar detalladamente los requerimientos, sino que, por el contrario, resulta fundamental conversar y verificar durante el *sprint* que el incremento que se vaya creando cumple con las expectativas.

Al mismo tiempo, coordina y facilita actividades que llamaremos refinamiento donde los *stakeholders*, los desarrolladores y el PO se involucran activamente en la definición del producto a mediano plazo y el entendimiento en detalle a corto plazo. Definitivamente, los ítems del *Product Backlog* a largo plazo no tiene ningún sentido que estén ordenados. A medida que el PO se acerca a ellos en el tiempo, en los refinamientos, los irá ordenando con mayor precisión.

Una forma en la se pueden ordenar los ítems del *Product Backlog* es según su aporte al objetivo del producto. Podemos entenderlo como la relevancia que un ítem tiene para el cumplimiento del objetivo del producto.

¿Ordenar o priorizar?

Cuando se habla de priorizar, una opción es acomodar el todo en tres grandes grupos: prioridad alta, media y baja. A diferencia de eso, los PBI de un *Product Backlog* deben estar en una fila, nunca compartiendo un mismo grupo. Priorizar es una forma de ordenar, pero no siempre lo que se encuentre primero luego de ordenar será lo prioritario desde el punto de vista del objetivo.

Digamos que el objetivo de construir una casa (producto) es mantenerme protegido de la lluvia del trópico: si priorizamos, pondríamos primero el techo, pues él nos cubrirá de la lluvia. Si ordenamos, no solo consideraremos el valor que aporte cada PBI, sino también, las dependencias, los riesgos y nuestro conocimiento adquirido. Entonces, pondríamos primero los cimientos y las paredes, pues estos sostendrán el techo.

El PO es quien determina el orden en el que se hace el trabajo. Todo lo que forma parte del *Product Backlog* está ordenado. El orden es muy preciso para aquellas cosas prioritarias que se transformarán en un incremento en el corto plazo. No tiene mucho sentido dotar de mucha precisión al orden de aquellas cosas que tienen menor prioridad, en las que se trabajará en el mediano plazo, ya que cualquier esfuerzo dedicado a ordenar con precisión estos ítems es muy probable que deba ser invertido nuevamente cuando se descubra que las prioridades deben cambiar a partir del *feedback* o el aprendizaje.

Definitivamente, los ítems del *Product Backlog* a largo plazo no tiene ningún sentido que estén ordenados. A medida que el PO se acerca a ellos en el tiempo, en los refinamientos, se irán ordenando con mayor precisión.

Todos los PBI del *Product Backlog* existen para lograr un cierto objetivo de producto. Dar un orden claro a los PBI es esencial porque justamente este orden determinará la estrategia de evolución del producto y las prioridades con las cuales los desarrolladores transformarán los PBI en incrementos de producto.

Este ordenamiento es responsabilidad exclusiva del *Product Owner* y, aunque todos dentro del equipo pueden hacer sugerencias o recomendaciones, es el *Product Owner* quien tiene la última palabra acerca del orden definitivo de los ítems del *Product Backlog*, teniendo en cuenta el contexto de negocio, el producto mismo y el mercado en el que está inserto.

Existen variadas técnicas de priorización de *backlog* con las cuales el PO es asistido y acompañado también por el *Scrum Master*. A continuación, repasaremos algunas de ellas.

Un modelo que puede ayudar a validar la importancia que cada PBI tiene para tu audiencia es el modelo KANO. El modelo KANO no es parte de Scrum, sino una recomendación, nos vamos a referir a los usuarios o consumidores como la "audiencia" de un producto o servicio. Con esta herramienta no podremos ordenar los PBI dado que, como dijimos anteriormente, no brinda posición específica para cada uno de ellos en el *Product Backlog*, lo más lejos que se podrá llegar es a agruparlos en cuatro grandes categorías. Es por eso que conviene utilizar el modelo KANO únicamente como una primera aproximación o como una validación de alto nivel de la alineación que hay entre el aporte esperado de cada PBI al objetivo de producto y las expectativas de tus usuarios o consumidores. La idea es agrupar las características de un producto en cuatro grandes grupos:

Características requeridas

Son todas aquellas características que la audiencia da por sentadas cuando se cumplen, pero les genera insatisfacción cuando el producto no las contempla.

Un ejemplo de esto sería un reproductor de música donde se entrecorta la transmisión del sonido. Los usuarios estarán insatisfechos mientras se interrumpa la música, pero el hecho de que no se entrecorte no les aumentará la satisfacción ni andarán contando por ahí lo bueno que es que la música se escuche de corrido. Dado que la mayoría de los usuarios de productos dan por sentado el cumplimiento de este tipo de características y las consideran como 'de mínima', es poco probable que las mencionen si se les pregunta qué cosas les gustaría ver en el producto.

Características buscadas

Una característica se cataloga como buscada cuando satisface a la audiencia si está presente en el producto y genera insatisfacción si falta. Estas son las características que por lo general se mencionan en los productos de consumo masivo o en las presentaciones corporativas de productos internos en las compañías. Siguiendo con el ejemplo del reproductor musical en línea, dos ejemplos de características buscadas podrían ser la capacidad de descargar la música y la posibilidad de escuchar sin anuncios. Por lo general, entra en esta categoría un segundo grupo importante de PBI con prioridades altas en el *Product Backlog*.

Características sorprendentes

Estas son características que causan satisfacción si están presentes, pero no generan insatisfacción si tu producto no cuenta con ellas debido a que la audiencia no las espera. En el ejemplo hipotético del reproductor musical en línea, una característica de este estilo es que puedan sincronizar diferentes dispositivos y comandar unos desde otros, por ejemplo, reproducir una canción en el computador, pero operando desde el teléfono móvil. Por lo general, es una categoría que agrupa PBI de baja prioridad.

Características no esperadas

Son aquellas que, de estar presentes en el producto, podrían causar insatisfacción en la audiencia y por eso no las espera. Imaginemos que el reproductor en línea tiene una característica mediante la cual envía un correo electrónico cada vez que se reproduce una canción. Pronto el usuario tendrá su casilla de correo repleta de *e-mails* que no aportan valor. Definitivamente, no debería haber PBI en el *Product Backlog* que califiquen en esta categoría.

Esta categorización de los ítems del *backlog*, también se trabaja con una herramienta denominada MoSCoW: *must, should, could, won´t*.

Tabla 1: MoSCoW

Tiene que estar (*Must*)	Debería estar (*Should*)	Podría estar (*Could*)	No es necesaria (*Won´t*)
			

Fuente: elaboración propia.

Otra forma de priorizar los PBI sería por el ROI (retorno de inversión), es decir, calcular el beneficio económico que se obtendrá en función del esfuerzo que se deba invertir. Esto, si bien es una simple fórmula matemática, tiene implícita la problemática de encontrar o conocer el valor económico ganado por la

incorporación de una determinada característica a un producto. Una vez identificada, el cálculo es relativamente simple:

ROI = beneficio/costo

El costo representa el esfuerzo necesario para la construcción de una determinada característica de un producto y el beneficio es el rédito económico obtenido por su incorporación.

Ordenado por importancia y riesgo

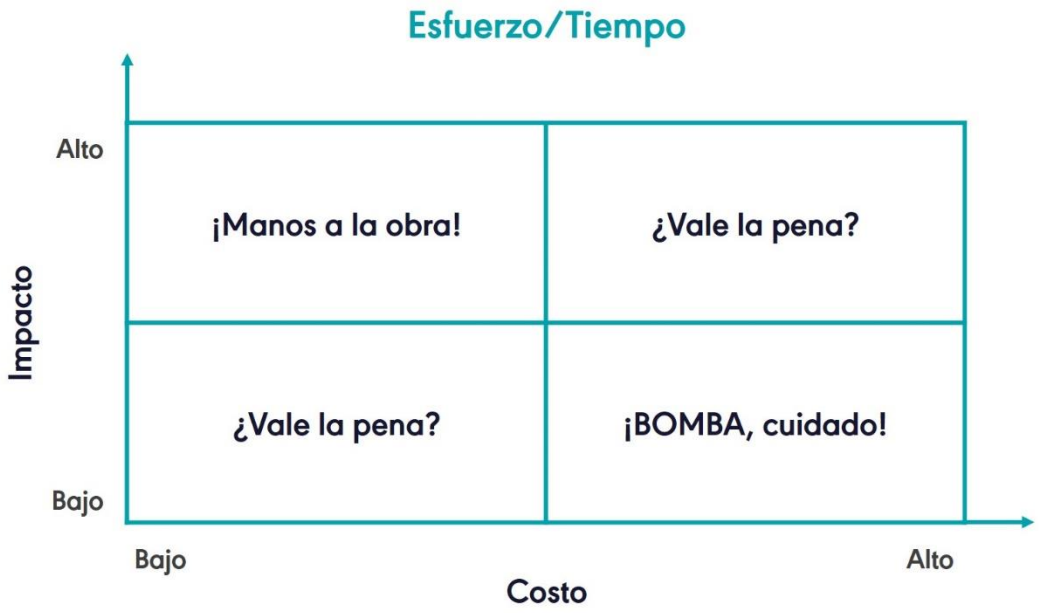
Ya sea que se ordenen los ítems del *Product Backlog* por aporte al objetivo del producto o por ROI, en cualquier caso, llamémosle "ordenar por importancia", estos pueden verse afectados por el nivel de riesgo asociado a cada uno de ellos.

De esta manera, se debería aprovechar la construcción iterativa y evolutiva de Scrum para mitigar riesgos en forma implícita: construyendo primero aquellos PBI con mayor riesgo asociado y dejando los que poseen menor riesgo para etapas posteriores.

Se recomienda que los PBI de baja importancia y alto riesgo sean evitados.

Esto suele trabajarse con la matriz de costo/impacto:

Figura 1: Matriz de costo/impacto



Fuente: elaboración propia.

Alcance variable

Dado que no es posible conocer de manera anticipada el producto que se debe construir para alcanzar los objetivos, es de esperar que se vaya aprendiendo de los clientes, descubriendo el negocio y validando los supuestos. Scrum es un viaje de descubrimiento que el equipo emprende junto a sus clientes y, bajo este escenario, el *Product Backlog* debe ser un elemento vivo que se adapta constantemente, al ritmo de tu aprendizaje y del *feedback* frecuente.

Si bien, tradicionalmente, el alcance se ha intentado fijar desde el comienzo, y así manejar el costo y el tiempo como los elementos variables, desde la agilidad, esta ecuación se invierte: el tiempo y el costo son los componentes fijos del proyecto, mientras que el alcance es el componente variable, dado que es lo que no conocemos de forma anticipada.

El principio de Pareto

Wilfredo Pareto nació en 1848 en Italia, donde creció formando parte de la clase media alta. Fue un reconocido ingeniero, sociólogo, economista, político y filósofo. Uno de sus estudios más reveladores, en aquella época, dejó al descubierto que el 80 % de las tierras de Italia pertenecían el 20 % de la población. A partir de ese descubrimiento, varios matemáticos y economistas derivaron esas observaciones y las verificaron en otros ámbitos. Uno de ellos fue Joseph Juran, quien en 1941 planteó el principio de Pareto (o regla del 80/20) aplicado a la calidad: el 80 % de los efectos son producidos por el 20 % de las causas. Esta ley también se conoció como el principio de los pocos vitales (el 20 % principal que genera el 80 % más importante) y los muchos triviales (el 80 % restante que genera el 20 % remanente).

Aplicando este principio al desarrollo de productos, encontramos que aproximadamente el 20 % de las características de un producto resuelve el 80 % de la necesidad que le da origen. Y, de manera recursiva, el 20 % del 80 % restante de las características, resuelven el 80 % del 20 % restante de la necesidad.

Manejo de contingencias

Aprovechando que el alcance es variable y que todo lo que se construye está ordenado en el *Product Backlog* según su aporte al objetivo del producto, vamos a utilizar los PBI menos prioritarios como la contingencia frente a imprevistos. Esto quiere decir que, al respetar tiempo y costo, el alcance de menor prioridad sería el que pagaría el precio de retrasos o desvíos. Para que este enfoque sea eficaz, es fundamental la labor del *Product Owner* y su habilidad para facilitar el descubrimiento de las prioridades por parte de todos los involucrados.

Una vez que el *backlog* se encuentra priorizado y ordenado, lo que sigue es refinar los ítems primeros.

"El refinamiento del *Product Backlog* es el acto de dividir y definir aún más los PBI en elementos más pequeños y precisos" (Schwaber y Sutherland, 2020, https://bit.ly/3FPUAD4).

El refinamiento del *Product Backlog* no es un evento de Scrum sino una actividad. Esto significa que en el *framework* no hay una reunión específica dedicada a ello. Esta actividad es coordinada de forma autogestionada por el equipo Scrum, quienes deciden cuándo y durante cuánto tiempo realizan el refinamiento del *Product Backlog*.

Durante el refinamiento revisas los PBI tentativos de *sprints* futuros, pero no de muchos *sprints* futuros porque sería una anticipación sin haber validado necesidades reales e invirtiendo demasiado esfuerzo en PBI de baja prioridad con posibilidad de ser descartados. Por eso lo ideal es refinar no más de tres o, como mucho, cuatro *sprints* futuros.

Se recomienda invitar a los actores claves de la audiencia a participar del refinamiento. Estos podrían ser *stakeholders*, usuarios, clientes, consumidores, etc. Esos PBI, aunque refinados, siguen siendo tentativos y podrían cambiar en orden y necesidad, por lo que no hay garantías que se lleven a cabo en los *sprints* previstos, por eso los llamo "tentativos".

Un *Product Backlog* eficiente

Cuando hablamos de eficiencia, hablamos de obtener el mayor beneficio con el menor esfuerzo posible. Este concepto llevado al *Product Backlog* significa invertir el esfuerzo de exploración y profundización de la manera más inteligente posible para evitar retrabajos y desperdicios. Por esto, fomentamos un *Product Backlog* donde sus PBI más prioritarios están expresados con un nivel de detalle mucho mayor que los PBI de menor prioridad, los cuales están descritos a un nivel más alto, ya que son los más susceptibles de ser alterados o reemplazados.

PBI e historias de usuario

Los ítems del *Product Backlog* (PBI) son los elementos que componen el *Product Backlog*. Al no ser prescriptivo, Scrum no indica cómo redactarlos ni gestionarlos, por lo que se han utilizado muchas técnicas y herramientas para hacerlo, por ejemplo, casos de uso, requerimientos, escenarios, historias de usuario, épicas, *bugs*, *spikes*, etc.

Sea cual sea el enfoque que se utilice para describir los PBI, hay al menos tres aspectos que no deben faltar:

- Una descripción de lo que se espera.
- Una estimación del esfuerzo necesario para crearlo.
- Un orden con respecto al resto de los PBI.

Ahora bien, el formato más utilizado para los PBI son las historias de usuario de *Extreme Programming*.

La forma en la que se comunican los requerimientos a los equipos que construyen productos tiene mucho de la ingeniería civil. Al ser esta una industria que se desarrolla principalmente en ambientes complicados con problemas bastante predecibles, no es para nada descabellado que, a la hora de construir puentes, edificios, fábricas, etc., se elaboren requerimientos extremadamente detallados como planos estructurales, diseños de instalaciones eléctricas y sanitarias, etc. para que luego un equipo de constructores repliquen en la realidad, con materiales estables y de comportamiento predecible, lo que ven en esas especificaciones.

La cuestión es que, en ambientes complejos, donde los constructores no tienen que replicar lo que ven en los planos, sino transformar supuestos de alto nivel de abstracción en un producto nuevo, este modelo no funciona.

En definitiva, las historias de usuario surgieron como una respuesta a una situación habitual en los proyectos de desarrollo de *software*. Los clientes o especialistas de negocio se comunican con los equipos de desarrollo a través de extensos documentos conocidos como especificaciones funcionales. A su vez, las especificaciones funcionales son la documentación de supuestos y están sujetas a interpretaciones, lo que causa malentendidos y que finalmente el *software* construido no se corresponda con la realidad esperada.

Una de las principales razones por las cuales la utilización de especificaciones detalladas como medio de comunicación no conduce a resultados satisfactorios es porque solo cubre una porción mínima (7 %) del espectro de la comunicación humana: el contenido. Según Albert Mehrabian, la comunicación humana se compone de tres partes:

- En un 7 %: el contenido (las palabras, lo dicho).
- En un 38 %: el tono de la voz.

- En un 55 %: las expresiones faciales.

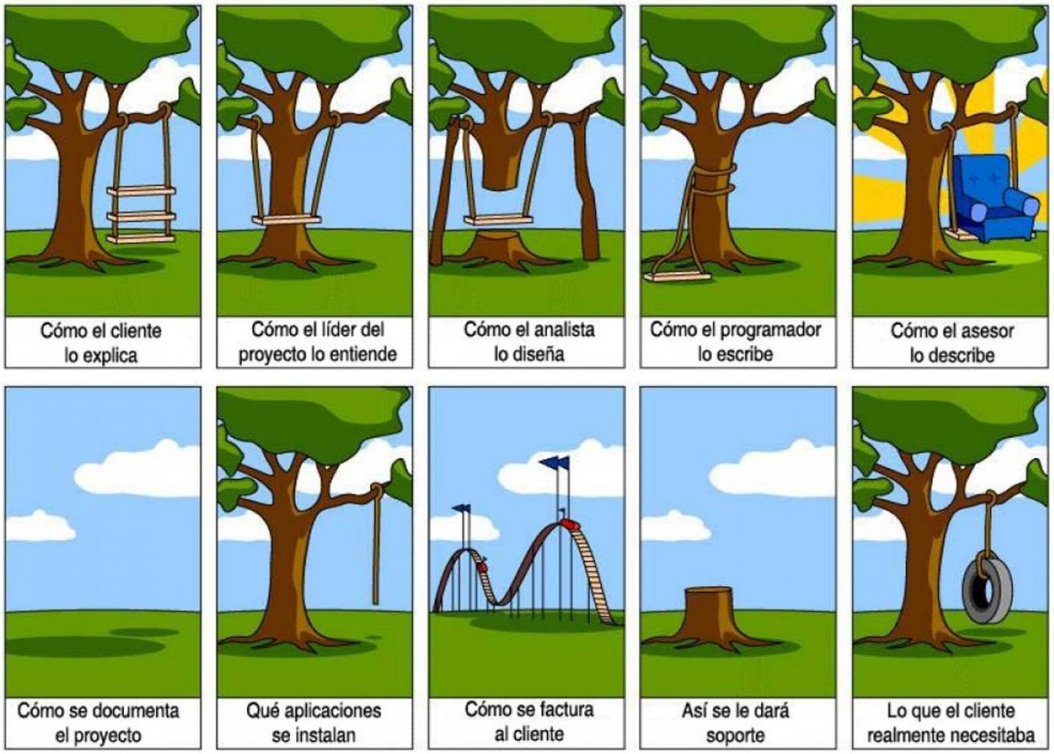
Por esto se concluye que, para tener una comunicación sólida, completa, es necesario el contacto cara-a-cara entre los interlocutores. En un esfuerzo orientado a que esas conversaciones existan, podemos decir que las historias de usuario son especificaciones funcionales que invitan a la conversación para que el detalle sea consecuencia de esta última y no un reemplazo.

Una historia de usuario se compone de tres elementos, también conocidos como "las tres C" (Jeffries, 2001) de las historias de usuario:

- *Card* (tarjeta). Toda historia de usuario debe poder describirse en una tarjeta/ficha de papel pequeña. Si una historia de usuario no puede describirse en ese tamaño, es una señal de que estamos traspasando las fronteras y comunicando demasiada información que debería compartirse cara a cara.
- Conversación. Toda historia de usuario debe tener una conversación con los *stakeholders*. Una comunicación cara a cara que intercambia no solo información sino también pensamientos, opiniones y sentimientos.
- Confirmación. Toda historia de usuario debe estar lo suficientemente explicada para que el equipo de desarrollo sepa qué es lo que debe construir y qué es lo que se espera para darla como terminada. Esto se conoce también como criterios de aceptación.

Este esfuerzo por hacernos entender con las historias de usuario responde a tantos malos entendidos que encontramos en nuestras conversaciones. Esta imagen te hará pensar en alguna experiencia personal sin duda.

Figura 2: Historias de usuarios



Fuente: [imagen sin título sobre historias de usuarios], (s. f.). https://bit.ly/3VWaEZk

Redacción de una historia de usuario

Mike Cohn sugiere una determinada forma de redactar historias de usuario bajo el siguiente formato:

- Como (rol) necesito (funcionalidad) para (beneficio)

Los beneficios de este tipo de redacción son, principalmente:

Primera persona

La redacción en primera persona de la historia de usuario invita a quien la lee a ponerse en el lugar del usuario.

Orden

Tener esta estructura para redactar la historia de usuario ayuda al *Product Owner* a ordenar el *Product Backlog*. Si el *Product Backlog* es un conjunto de ítems como "Permitir crear un evento tentativo", "Confirmar un evento tentativo", "Notificar al responsable de logística", "Ver el estado de inscripciones", etc. el *Product Owner* debe trabajar más para comprender cuál es la característica de producto, quien se beneficia y cuál es el valor de esta.

Propósito

Conocer el propósito de una característica del producto permite al equipo Scrum plantear alternativas que cumplan con el mismo propósito en el caso de que el costo de la característica esperada sea alto o su construcción no sea viable.

Se recomienda que toda historia de usuario cumpla con seis características que se pueden recordar bajo la regla mnemotécnica "INVEST" (Wakw, 2003).

Figura 3: Invest

I	No debe depende de otras para poder completar el objetivo de la historia. Las dependencias pueden reducirse combinándolas en una o dividiéndolas diferente.
N	Deben ser negociables ya que sus detalles serán acordados con el cliente o el usuario durante la fase de conversación.
V	Deben aportar algún valor al usuario.
E	El equipo de ser capaz de poder estimar la necesidad expresada.
S	Deberían poder englobar trabajo traducible en el periodo del *Sprint*.
T	Si el cliente o usuario no sabe cómo probar la historia significa que no es del todo clara o que no es valiosa.

Fuente: elaboración propia.

Independientes (I)

Las historias de usuario deben ser independientes de forma tal que no se superpongan y que puedan planificarse y desarrollarse en cualquier orden.

Muchas veces esta característica no puede cumplirse para el 100 % de las historias. El objetivo que debemos perseguir es preguntarnos y cuestionarnos en cada historia de usuario si hemos hecho todo lo posible para que esta sea independiente del resto.

Negociable (N)

Una buena historia de usuario es negociable. No es un contrato explícito por el cual se debe entregar todo-o-nada. Por el contrario, el alcance de las historias (sus criterios de aceptación) podrían ser variables: pueden incrementarse o eliminarse con el correr del desarrollo y en función de la inspección del producto. En el caso de que uno o varios criterios de aceptación se extraigan de una historia de usuario, estos se transformarán en una o varias historias de usuario nuevas.

Valorable (V)

Una historia de usuario debe ser valorable por el *Product Owner*. Los desarrolladores pueden tener actividades técnicas como parte del *Product Backlog*, pero para que puedan ser consideradas una historia de usuario, deben ser enmarcadas de forma tal que el *Product Owner* las considere importantes, caso contrario, no deberían formar parte del *Product Backlog*.

En general, esta característica representa un desafío a la hora de dividir historias de usuario. Bill Wake propone pensar en una historia de usuario como si fuese una torta de múltiples capas, por ejemplo: una capa de persistencia, una capa de negocio, una capa de presentación, etc. Cuando dividamos esa historia de usuario, lo que vamos a estar sirviendo es una parte de esa "torta" y el objetivo debería ser darle al *Product Owner* la esencia de la "torta" completa, y la mejor manera de hacerlo es cortando una rodaja vertical de esta "torta" a través de todas las capas. Los desarrolladores tenemos una inclinación especial de trabajar en una capa a la vez hasta completarla, pero una capa de persistencia de datos completa y terminada tiene muy poco o ningún valor para el *Product Owner* si no hay una capa de negocio y de presentación.

Estimable (E)

Una historia de usuario debería ser estimable. Mike Cohn (2003), identifica tres razones principales por las cuales una historia de usuario no podría estimarse:

1. La historia de usuario es demasiado grande. En este caso la solución sería dividir la historia de usuario en historias más pequeñas que sean estimables.
2. Falta de conocimiento funcional. En este caso la historia de usuario vuelve al *Product Owner* para bajar en detalle la historia o inclusive (y recomendable) tener una conversación con el equipo de desarrollo.

3. Falta de conocimiento técnico. Muchas veces el equipo de desarrollo no tiene el conocimiento técnico suficiente para realizar la estimación. En estos casos el equipo de desarrollo puede dividir la historia en 1) un *time-box* conocido como *spike* que le permita investigar la solución y proveer una estimación más certera y 2) la funcionalidad a desarrollar como parte de la historia en sí misma.

Pequeña (*Small*)

Toda historia de usuario debe ser lo suficientemente pequeña de forma tal que permita ser estimada por los desarrolladores. Si bien no es una medida explícita, tener entre 6 y 8 historias de usuario por *sprint* es una buena señal de tamaño.

Las descripciones de las historias de usuario también deberían ser pequeñas, y escribirlas en fichas pequeñas ayuda a que eso suceda.

Verificable (Testable)

Una buena historia de usuario es verificable. Se espera que el equipo Scrum no solo pueda describir la expectativa que los *stakeholders* tienen, sino que también logren verificarla (probarla).

Principio de la mejora continua

La mejora continua es la práctica de evaluar constantemente tus acciones a la luz de tus resultados, identificando puntos de mejora y experimentación, de forma sistemática, con el objetivo de tener cada vez mejores resultados. Para llevar adelante esa mejora, entonces, es necesario inspeccionar y adaptar.

Para ilustrar diferentes tipos de mejoras que podrías llevar adelante cuando no te satisfacen los resultados, vamos a introducir un modelo ideado por Echeverría y Pizarro, quienes lo han llamado "OSAR", como referencia al siguiente esquema:

(O)bservador (S)istema (A)cciones (R)esultados

El modelo OSAR se lee de derecha a izquierda, comenzando por los resultados porque sostiene la premisa que nuestras interpretaciones y acciones se evalúan a partir de los resultados que obtenemos. Es decir, la única explicación es el resultado.

Según este modelo, cuando evaluamos los resultados y no estamos conforme con ellos, tenemos una gran oportunidad de aprendizaje. Este aprendizaje se puede dar en dos órdenes diferentes. Los veremos a continuación.

Aprendizaje de primer orden

Este orden de aprendizaje sucede al modificar nuestras acciones en busca de nuevos resultados. Probamos hacer algo mejor a lo realizado hasta ese momento. De esta manera, aprendemos nuevas formas de hacer las mismas cosas y desarrollamos nuestras competencias.

En el caso típico de los requerimientos, cuando la contraparte no los comprende como nosotros esperamos, lo que hacemos es mejorar la forma de especificar. Especificamos diferente, con mayor profundidad, creamos escenarios, incluimos actores, sumamos diagramas, etc.

Luego, con esa acción mejorada, esperamos un resultado diferente. Si el resultado no es lo esperado, volvemos a mejorar la acción. Y así sucesivamente.

Aprendizaje de segundo orden

Muchas veces el aprendizaje de primer orden no es suficiente para producir un cambio significativo en los resultados. En este caso, aparece la posibilidad de un aprendizaje de segundo orden en el cual el cambio se da a nivel del observador y no con respecto a las acciones.

De esa manera, nos constituimos en un nuevo observador, diferente al que éramos y vemos la realidad de una nueva manera. Al ser un nuevo observador tienes la capacidad de ver nuevas acciones que, hasta ese momento, no existían en el ámbito de lo posible. Al ver la realidad de manera diferente es posible hacer cosas nuevas.

El resultado de este aprendizaje de segundo orden fue dejar de mejorar la forma de especificar y, así como suena, empeorarla. Hacerla tan deficiente que no quede otra opción más que juntarse a conversar para entender la problemática y lo que se espera. Por eso se dice que las historias de usuario son, ni más ni menos, que el recordatorio de una conversación pendiente.

Conclusión sobre estimaciones en Scrum

Muchas teorías y enfoques convergen en las siguientes características sobre estimaciones en equipos Scrum. No tiene sentido presentar estimaciones certeras, ya que su probabilidad de ocurrencia es extremadamente baja por el alto nivel de incertidumbre. Intentar bajar dicha incertidumbre mediante el análisis puede llevarnos al análisis-parálisis. Para evitar esto debemos estimar a alto nivel con un elevado grado de probabilidad, actuar rápidamente, aprender de nuestras acciones y refinar el *Product Backlog* frecuentemente. Este enfoque se conoce también como *Rolling Wave Planning* o elaboración progresiva. La mejor estimación es la que proveen los desarrolladores. Será mucho más realista que la estimación provista por un experto ajeno al equipo. Las estimaciones son de inversión tolerada y no de costo esperado. El tiempo y el costo son fijos, el alcance es variable. Las métricas principales son de *outcome* en vez de *output*.

Outcomes vs. *outputs*

¿Cómo hace Scrum para mitigar el riesgo que conlleva la falta de precisión en las estimaciones? Lo hace apoyándose en dos aspectos: privilegiando los *outcomes* sobre los *outputs* y entregando frecuentemente un incremento terminado. Los ***outcomes*** son los resultados que la empresa o producto quiere o necesita lograr. En el caso de Scrum serían los objetivos de producto y objetivos de *sprint*. Los ***outputs*** son las acciones o elementos que contribuyen a lograr esos *outcomes*, por ejemplo, el plan del *sprint* y las tareas que lo componen. Medir un *output* es medir el hacer, por ejemplo, la cantidad de PBI o puntos de historia entregados. Medir un *outcome* es medir el resultado de lo hecho, por ejemplo, el grado de cumplimiento de un objetivo de negocio. Si estamos construyendo características para incrementar las ventas, entonces sería medir el incremento en las ventas. Al medir los *outcomes* (incremento en las ventas) en vez de los *outputs* (cantidad de puntos de historia entregados), podríamos saber si el producto que se está construyendo resuelve las necesidades. Al entregar frecuentemente un incremento terminado podremos medir los *outcomes* de forma temprana. Al aportar el mayor valor posible en cada incremento debido al ordenamiento del *Product Backlog* se estará mitigando implícitamente los riesgos, tanto en la entrega de alcance como en el aporte de valor.

Referencias

[Imagen sin título sobre historias de usuarios], (s. f.). https://www.researchgate.net/figure/Figura-2-El-arbol-de-los-proyectos-segun-Project-Cartoon-2006_fig2_334756539

Alaimo, M. (2021). *Scrum y algo más*. Mtn Labs.

Cohn, M. (2003). *User Stories Applied*. Addison-Wesley Professional.

Jeffries, R. (2001). *Essential XP: Card, Conversation, Confirmation*. https://ronjeffries.com/xprog/articles/expcardconversationconfirmation/

Schwaber, K. y Sutherland, J. (2020). *La Guía Scrum*. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-European.pdf

Wakw, B. (2003). *INVEST in Good Stories, and SMART Tasks*. https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/