[Capítulo 5](#_bookmark2)

[Requisitos e historias de usuario](#_bookmark2)

En este capítulo discuto cómo los requisitos en un proyecto Scrum se manejan de manera diferente a los de un proyecto tradicional. Con este contexto, describo el papel de las historias de usuario como un formato común para representar elementos de valor empresarial. Me enfoco en qué son las historias de usuario, cómo pueden representar el valor comercial en múltiples niveles de abstracción y cómo determinar cuándo las historias de usuario son buenas. A continuación, describo cómo manejar los requisitos no funcionales y el trabajo de adquisición de conocimientos en un proyecto Scrum . Termino detallando dos técnicas para recopilar historias de usuario.

# [Visión general](#_bookmark2)

Scrum y el desarrollo secuencial de productos tratan los requisitos de manera muy diferente. Con el desarrollo secuencial de productos, los requisitos no son negociables, se detallan desde el principio y están destinados a ser independientes. En Scrum, los detalles de un requisito se negocian a través de conversaciones que ocurren continuamente durante el desarrollo y se desarrollan *justo a tiempo* y *lo suficiente para* que los equipos comiencen a construir la funcionalidad para respaldar ese requisito.

Con el desarrollo secuencial de productos, los requisitos se tratan de manera muy similar a como se tratan en la fabricación: son especificaciones requeridas y no negociables a las que debe ajustarse el producto. Estos requisitos se crean por adelantado y se entregan al grupo de desarrollo en forma de un documento muy detallado. Es el trabajo del grupo de desarrollo, entonces, producir un producto que se ajuste a los requisitos detallados.

Cuando se considera necesario un cambio del plan original, se gestiona a través de un proceso formal de control de cambios. Debido a que el objetivo es cumplir con las especificaciones, estas desviaciones son indeseables y costosas. Después de todo, es posible que gran parte del trabajo en curso (WIP), en forma de requisitos muy detallados (y todo el trabajo basado en ellos), deba cambiarse o descartarse.

Por el contrario, Scrum ve los requisitos como un grado importante de libertad que podemos manipular para cumplir con nuestros objetivos comerciales. Por ejemplo, si nos estamos quedando sin tiempo o dinero, podemos eliminar los requisitos de bajo valor. Si, durante el desarrollo, nueva información indica que la relación costo/beneficio de un requisito se ha vuelto significativamente menos favorable, podemos optar por eliminar el requisito del producto. Y si surge un nuevo requisito de alto valor, tenemos la capacidad de agregarlo al producto, tal vez descartando un requisito de menor valor para hacer espacio.

Probablemente todos hemos tenido la experiencia de escribir un documento de requisitos "completo" al comienzo del desarrollo, solo para descubrir más tarde que un documento importante

**79**

faltaba el requisito. Cuando descubrimos ese requisito faltante, la conversación probablemente sonó más o menos así:

Cliente: "Ahora que veo estas características integradas, me doy cuenta de que necesito esta otra característica que no está en el documento de requisitos".

Desarrolladores: "Si querías esa característica, ¿por qué no la especificaste desde el principio?"

Cliente: "Bueno, no me di cuenta de que necesitaba esa función hasta que vi que el producto se unía".

Desarrolladores: "Bueno, si hubieras pensado más y más sobre los requisitos desde el principio, habrías encontrado esa característica entonces en lugar de ahora".

El hecho es que, cuando se desarrollan productos innovadores, no se pueden crear requisitos o diseños completos por adelantado simplemente trabajando más tiempo y más duro. Siempre surgirán algunos requisitos y diseños una vez que el desarrollo del producto esté en marcha; Ninguna cantidad de trabajo inicial exhaustivo lo impedirá.

Por lo tanto, cuando usamos Scrum, no invertimos una gran cantidad de tiempo y dinero en desarrollar los detalles de un requisito por adelantado. Debido a que esperamos que los detalles cambien a medida que pasa el tiempo y aprendemos más sobre lo que estamos construyendo, evitamos invertir demasiado en requisitos que luego podríamos descartar. En lugar de compilar un gran inventario de requisitos detallados por adelantado, creamos marcadores de posición para los requisitos, denominados elementos de **trabajo pendiente del producto** (**PBI**). Cada elemento de la cartera de productos representa un valor comercial deseable (véase la figura 5.1).

Inicialmente, los elementos de la cartera de productos son grandes (representan grandes franjas de valor comercial) y hay muy pocos detalles asociados con ellos. Con el tiempo, hacemos fluir estos elementos del backlog del producto a través de una serie de conversaciones entre los interesados, el propietario del producto y el equipo de desarrollo, refinándolos en una colección de PBI más pequeños y detallados. Eventualmente, un elemento de trabajo pendiente del producto es lo suficientemente pequeño y detallado como para pasar a un sprint, donde se diseñará, construirá y probará. Sin embargo, incluso durante el sprint, se expondrán más detalles en las conversaciones entre el propietario del producto y el equipo de desarrollo.

Como discutiré en el Capítulo 6, el trabajo pendiente del producto es simplemente una instantánea de la colección actual de elementos del trabajo pendiente del producto y sus detalles asociados.

Aunque Scrum no especifica ningún formato estándar para estos elementos de trabajo pendiente del producto, muchos equipos representan PBI como casos de usuario. No tienes que hacerlo. Algunos equipos prefieren casos de uso, y otros optan por representar sus PBI en sus propios formatos personalizados. En este libro, empleo las historias de usuario como la representación principal de los elementos del backlog del producto. Discutiré los detalles de las historias de usuario más adelante en este capítulo. Incluso si eliges usar otra cosa, la discusión de las historias de usuario te resultará útil en

Comprender qué características querrá de cualquier otra representación.

**Cartera de productos a lo largo del tiempo**



**Leyenda**

**Tamaño del marcador de posición**

**Mayor**

**Menor**

**Cantidad de detalles**

**Un poco**

**Mucho**

**Apagado**

**FIGURA 5.1** Scrum utiliza marcadores de posición para los requisitos.

# [Uso de conversaciones](#_bookmark2)

Como vehículo de comunicación, los requisitos facilitan una comprensión compartida de lo que hay que construir. Permiten que las personas que entienden lo que debe ser creado comuniquen claramente sus deseos a las personas que tienen que crearlo.

El desarrollo secuencial de productos depende en gran medida de los requisitos escritos, que parecen impresionantes, pero pueden malinterpretarse fácilmente. Recuerdo una conversación con un vicepresidente de gestión de productos de una empresa que visité. Le pregunté a esta persona, que dirigía a todos los analistas de negocios de la empresa, cómo manejaban los requisitos. Dijo a modo de ilustración: "El 1 de enero mi equipo entrega a la organización de ingeniería el documento de requisitos, y el 31 de diciembre nos presentamos y vemos lo que obtuvimos".

Le pregunté quién de su equipo estaría disponible durante el año para responder preguntas y aclarar los requisitos de los desarrolladores. Él dijo: "Nadie. Todo el tiempo que mi grupo tuvo que invertir en este proyecto se dedicó a escribir el documento de requisitos. Mis analistas están trabajando en los documentos de requisitos para otros proyectos. Pero no te preocupes, escribimos un buen documento, y cualquier pregunta que tengan los desarrolladores o evaluadores se puede responder leyendo detenidamente el documento".

Me parecía poco probable que no hubiera ambigüedades en su documento detallado de 150 páginas sobre el caso de uso de un nuevo sistema de registros médicos electrónicos. El inglés no es tan preciso; Incluso si lo fuera, la gente no es tan precisa con su escritura.

Una forma de asegurarse mejor de que se están construyendo las características deseadas es que las personas que saben lo que quieren tengan conversaciones oportunas con las personas que están diseñando, construyendo y probando esas características.

En Scrum, aprovechamos la conversación como una herramienta clave para garantizar que los requisitos se discutan y comuniquen adecuadamente. La comunicación verbal tiene la ventaja de ser de gran ancho de banda y proporcionar una retroalimentación rápida, lo que hace que sea más fácil y barato obtener una comprensión compartida. Además, las conversaciones permiten una comunicación bidireccional que puede generar ideas sobre problemas y oportunidades, discusiones que probablemente no surgirían de la lectura de un documento.

La conversación, sin embargo, es solo una herramienta. No sustituye a todos los documentos. En Scrum, el backlog del producto es un "documento vivo", disponible en todo momento durante el desarrollo del producto. Aquellos que aún desean o deben tener un documento de especificación de requisitos pueden crear uno en cualquier momento, simplemente recopilando los elementos del backlog del producto y todos sus detalles asociados en un documento con el formato que deseen.

# [Refinamiento progresivo](#_bookmark2)

Con el desarrollo secuencial de productos, todos los requisitos deben estar al mismo nivel de detalle al mismo tiempo. En particular, el documento de requisitos aprobado debe especificar todos y cada uno de los requisitos para que los equipos que realizan el trabajo de diseño, construcción y prueba puedan comprender cómo cumplir con las especificaciones. No quedan detalles por agregar.

Forzar todos los requisitos al mismo nivel de detalle al mismo tiempo tiene muchas desventajas:

* Debemos predecir todos estos detalles al principio del desarrollo del producto, cuando tenemos el menor conocimiento que jamás tendremos.
* Tratamos todos los requisitos de la misma manera, independientemente de su prioridad, lo que nos obliga a dedicar valiosos recursos hoy para crear detalles para requisitos que tal vez nunca construyamos.
* Creamos un gran inventario de requisitos que probablemente serán muy costosos de reelaborar o descartar cuando las cosas cambien.
* Reducimos la probabilidad de utilizar conversaciones para elaborar y aclarar los requisitos porque los requisitos ya están "completos".

Como se ilustra en la figura 5.1, cuando se utiliza Scrum, no todos los requisitos tienen que estar en el mismo nivel de detalle al mismo tiempo. Los requisitos en los que trabajaremos antes serán más pequeños y detallados que los que no trabajaremos durante algún tiempo. Empleamos una estrategia de **refinamiento progresivo** para desagregar, justo a tiempo, los requisitos grandes y ligeramente detallados en un conjunto de elementos más pequeños y detallados.

# [¿Qué son las historias de usuario?](#_bookmark2)

**Los casos de usuario** son un formato conveniente para expresar el valor empresarial deseado para muchos tipos de elementos de trabajo pendiente del producto, especialmente las características. Las historias de usuario se elaboran de manera que sean comprensibles tanto para los empresarios como para los técnicos. Son estructuralmente simples y proporcionan un excelente marcador de posición para una conversación. Además, se pueden escribir en varios niveles de granularidad y son fáciles de refinar progresivamente.

Por muy bien adaptadas a nuestras necesidades que puedan estar las historias de usuario, no considero que sean la única forma de representar los elementos del backlog del producto. Son simplemente un enfoque ligero que encaja muy bien con los principios ágiles básicos y nuestra necesidad de un marcador de posición eficiente y eficaz. Los utilizo como marcador de posición central al que adjuntaré cualquier otra información que crea que es relevante y útil para detallar un requisito. Si descubro que las historias de usuario son un ajuste forzado para una situación particular (como la representación de ciertos defectos), usaré otro enfoque. Por ejemplo, una vez vi a un equipo escribir la siguiente historia de usuario: "Como cliente, me gustaría que el sistema no corrompiera la base de datos". Creo que todos podemos estar de acuerdo en que una historia de usuario probablemente no sea la mejor manera de representar este problema. Tal vez sería más apropiado una simple referencia al defecto en el sistema de seguimiento de defectos .

Entonces, ¿qué son exactamente las historias de usuario? Ron Jeffries ofrece una forma sencilla pero eficaz de

pensemos en las historias de usuario (Jeffries, 2001). Las describe como las tres C: carta, conversación y confirmación.

### Tarjeta

La idea de la tarjeta es bastante simple. Originalmente, las personas escribían (y muchos todavía lo hacen) historias de usuario directamente en tarjetas de índice de 3 × 5 pulgadas o notas adhesivas (ver Figura 5.2).

Un formato de plantilla común para escribir historias de usuario (como se muestra a la izquierda en la figura 5.2) es especificar una clase de usuarios (el rol de usuario), lo que esa clase de usuarios quiere lograr (el objetivo) y por qué los usuarios quieren lograr el objetivo (el beneficio) (Cohn 2004). La parte "para que" de una historia de usuario es opcional, pero a menos que el propósito de la



**Historia de usuario tfitle**

Como <rol de usuario> quiero <objetivo> para que <beneficie>.

Plantilla



**Buscar reseñas cerca de la dirección**

Como usuario típico, quiero ver reseñas imparciales de un restaurante cerca de una dirección para poder decidir dónde ir a cenar.

**FIGURA 5.2** Una plantilla y una tarjeta de historia de usuario

La historia es completamente obvia para todos, debemos incluirla con cada historia de usuario. El lado derecho de la Figura 5.2 muestra un ejemplo de una historia de usuario basada en esta plantilla.

La tarjeta no está pensada para capturar toda la información que conforma el requisito. De hecho, utilizamos deliberadamente tarjetas pequeñas con espacio limitado para promover la brevedad. Una tarjeta debe contener algunas oraciones que capturen la esencia o la intención de un requisito. Sirve como marcador de posición para discusiones más detalladas que tendrán lugar entre las partes interesadas, el propietario del producto y el equipo de desarrollo.

### Conversación

Los detalles de un requisito se exponen y comunican en una conversación entre el equipo de desarrollo, el propietario del producto y las partes interesadas. La historia de usuario es simplemente una promesa de tener esa conversación.

Digo "esa conversación", pero en realidad, la conversación no suele ser un evento único, sino más bien un diálogo continuo. Puede haber una conversación inicial cuando se escribe la historia de usuario, otra conversación cuando se refina, otra cuando se estima, otra durante la planificación del sprint (cuando el equipo se sumerge en los detalles del nivel de tarea) y, por último, conversaciones continuas mientras se diseña, construye y prueba la historia de usuario durante el sprint.

Uno de los beneficios de las historias de usuario es que cambian parte del enfoque de la escritura a las conversaciones. Estas conversaciones permiten una forma más rica de intercambiar información y colaborar para garantizar que todos expresen y entiendan los requisitos correctos.

Aunque las conversaciones son en gran medida verbales, pueden ser, y con frecuencia se complementan, con documentos. Las conversaciones pueden conducir a un boceto de la interfaz de usuario o a una elaboración de reglas de negocio que se escriben. Por ejemplo, visité una organización que estaba desarrollando software de imágenes médicas. Una de sus historias se muestra en la Figura 5.3.

Tenga en cuenta que la historia de usuario hace referencia a un artículo completo para leerlo y conversarlo en el futuro.

Por lo tanto, no vamos a desechar todos nuestros documentos en favor de las historias de usuario y sus tarjetas de historia asociadas. Las historias de usuario son simplemente un buen punto de partida para obtener la esencia inicial de lo que se desea y para proporcionar un recordatorio para discutir



**Visualización de Johnson de datos de resonancia magnética**

Como radiólogo, quiero visualizar la resonancia magnética

utilizando el nuevo algoritmo del Dr. Johnson.

Para más detalles, véase el informe de enero de 2007

Emisión de la Revista de Matemáticas páginas 110-118.

**FIGURA 5.3** Historia de usuario con datos adicionales adjuntos

requisitos con más detalle cuando corresponda. Sin embargo, las historias de usuario pueden y deben complementarse con cualquier otra información escrita que ayude a proporcionar claridad sobre lo que se desea.

### Confirmación

Una historia de usuario también contiene información de confirmación en forma de condiciones de satisfacción. Estos son criterios de aceptación que clarifican el comportamiento deseado. Son utilizados por el equipo de desarrollo para comprender mejor qué construir y probar, y por el propietario del producto para confirmar que la historia de usuario se ha implementado a su satisfacción.

Si el anverso de la tarjeta tiene una descripción de pocas líneas de la historia, el reverso de la tarjeta podría especificar las condiciones de satisfacción (véase la figura 5.4).

Estas condiciones de satisfacción pueden expresarse como pruebas de aceptación de alto nivel. Sin embargo, estas pruebas no serían las únicas que se ejecutan cuando se está desarrollando la historia. De hecho, para el puñado de pruebas de aceptación que están asociadas con una historia de usuario, el equipo tendrá muchas más pruebas (tal vez de 10 a 100 veces más) a un nivel técnico detallado que el propietario del producto ni siquiera conoce.

Las pruebas de aceptación asociadas con la historia existen por varias razones. En primer lugar, son una forma importante de captar y comunicar, desde la perspectiva del propietario del producto, cómo determinar si la historia se ha implementado correctamente.

Estas pruebas también pueden ser una forma útil de crear historias iniciales y refinarlas a medida que se conocen más detalles. Este enfoque a veces se denomina **especificación por ejemplo** o **desarrollo basado en pruebas de aceptación** (**ATTD**). La idea es bastante intuitiva. Las discusiones sobre las historias pueden centrarse, y con frecuencia lo hacen, en la definición de ejemplos específicos o comportamientos deseados. Por ejemplo, en la historia de "Cargar archivo" en la Figura 5.4, es probable que la conversación haya sido más o menos así:

Inicialmente, limitemos el tamaño de los archivos cargados a 1 GB o menos. Además, asegúrese de que podamos cargar correctamente archivos de texto y gráficos comunes. Y por razones legales, no podemos tener ningún archivo con restricciones de administración de derechos digitales (DRM) cargado en la wiki.



**Subir archivo**

Como usuario de la wiki, quiero subir un archivo a

la wiki para que pueda compartirla con

mis colegas.



**Condiciones de Satisfacción**

Verificación con archivos .txt y .doc Verificación con archivos .jpg, .gif y .png Verificación con archivos .mp4 <= 1 GB Verificación sin archivos restringidos por DRM

**FIGURA 5.4** Condiciones de satisfacción de la historia de usuario

**TABLA 5.1** Ejemplo de prueba automatizada

|  |  |
| --- | --- |
| **Tamaño** | **Válido()** |
| 0 | Verdadero |
| 1,073,741,824 | Verdadero |
| 1,073,741,825 | Falso |

Si estuviéramos utilizando una herramienta como Fit o FitNesse, podríamos definir convenientemente estas pruebas en una tabla como la Tabla 5.1, que muestra ejemplos de diferentes tamaños de archivo y si son válidos o no.

Al elaborar ejemplos específicos como estos, podemos impulsar el proceso de creación y refinamiento de historias y tener pruebas de aceptación (automatizadas) disponibles para cada historia.

# [Nivel de detalle](#_bookmark3)

Las historias de usuario son un excelente vehículo para transportar elementos de valor para el cliente o usuario a través del flujo de creación de valor de Scrum. Sin embargo, si solo tenemos un tamaño de piso (el tamaño que cabría cómodamente en un sprint de corta duración), será difícil hacer una planificación de alto nivel y cosechar los beneficios del refinamiento progresivo.

Las historias pequeñas que se usan en el nivel de sprint son demasiado pequeñas y numerosas para respaldar la planificación de productos y lanzamientos de nivel superior. A estos niveles necesitamos menos elementos, menos detallados, más abstractos. De lo contrario, estaremos sumidos en un pantano de detalles en su mayoría irrelevantes. Imagínese tener 500 historias muy pequeñas y que le pidan que proporcione una descripción de nivel ejecutivo del producto propuesto para asegurar su financiación. O intenta priorizar entre esos 500 elementos realmente pequeños para definir la próxima versión.

Además, si solo hay un tamaño (pequeño) de historia, nos veremos obligados a definir todos los requisitos con un nivel de detalle muy fino mucho antes de lo que deberíamos. Tener solo historias pequeñas excluye el beneficio de refinar progresivamente los requisitos sobre una base justa y justo a tiempo.

Afortunadamente, las historias de usuario se pueden escribir para capturar las necesidades de los clientes y usuarios en varios niveles de abstracción (consulte la figura 5.5).

La figura 5.5 muestra historias en múltiples niveles de abstracción. Las más grandes serían las historias que tienen un tamaño de unos pocos a muchos meses y pueden abarcar una versión completa o varias versiones. Mucha gente se refiere a ellas como **epopeyas**, aludiendo a la idea de que son historias *del tamaño de El Señor de los Anillos* o *Guerra y Paz*. Las epopeyas son útiles porque ofrecen una visión general muy amplia y de alto nivel de lo que se desea (véase la figura 5.6).

Nunca convertiríamos una epopeya en un sprint para el desarrollo porque es demasiado grande y no muy detallada. En cambio, las epopeyas son excelentes marcadores de posición para un gran

**Meses tteeks**



**Días**

**Horas**

**Más grande que un lanzamiento Más grande que un sprint**

**Listo para el sprint**

**tfask**

**FIGURA 5.5** Jerarquía de abstracción de historias de usuario



**Preferencia tfraining Epic**

Como usuario típico, quiero entrenar el

sistema sobre qué tipos de productos y

revisiones de servicio que prefiero para que lo sepa

Qué características usar al filtrar reseñas en mi nombre.

**FIGURA 5.6** Ejemplo de epopeya

Colección de historias más detalladas que se crearán en un momento futuro apropiado. Ilustraré el uso de epopeyas durante la discusión de la planificación de productos en el Capítulo 17.

Las historias de siguiente tamaño en la Figura 5.5 son aquellas que a menudo tienen un tamaño del orden de semanas y, por lo tanto, son demasiado grandes para un solo sprint. Algunos equipos pueden llamar a estas **características**. Las formas más pequeñas de historias de usuario son aquellas a las que normalmente me refiero como **historias**. Para evitar cualquier confusión con epopeyas, características u otros elementos más grandes, que también son "historias", algunas personas llaman a estas historias  **historias sprintables** o **historias implementables** para indicar que tienen un tamaño del orden de días y, por lo tanto, lo suficientemente pequeñas como para caber en un sprint y ser implementadas. La figura 5.2 proporciona un ejemplo de un sprintable

cuento.

Algunos equipos también utilizan el término **tema** para referirse a una colección de historias relacionadas. Los temas proporcionan una forma conveniente de decir que un montón de historias tienen algo en común, como estar en la misma área funcional. En la Figura 5.7, el tema representa la colección de historias que proporcionarán los detalles de cómo realizar la capacitación en palabras clave.



**Palabra clave tfraining tfheme**

Como usuario típico, quiero entrenar el

sistema sobre qué palabras clave usar y cuándo

Filtrar reseñas para poder filtrar por palabras

que son importantes para mí.

**FIGURA 5.7** Tema de ejemplo

A menudo pienso en un tema como la tarjeta de resumen de un montón de tarjetas de notas apiladas con una banda elástica alrededor de ellas para indicar que son similares entre sí en un área que creemos que es importante.

Las tareas son la capa inferior a las historias, en las que suele trabajar una sola persona, o tal vez un par de personas. Por lo general, las tareas requieren horas para realizarse. Cuando vamos a la capa de tareas, estamos especificando *cómo* construir algo en lugar de *qué* construir (representado por epopeyas, características e historias). Las tareas no son historias, por lo que debemos evitar incluir detalles a nivel de tarea al escribir historias.

Es importante tener en cuenta que términos como *épica*, *largometraje*, *historia* y *tema* son solo etiquetas de conveniencia y no se comparten universalmente. Realmente no importa qué etiquetas uses, siempre y cuando las uses de manera constante. Lo que importa es reconocer que las historias pueden existir en múltiples niveles de abstracción, y que hacerlo apoya muy bien nuestros esfuerzos por planificar en múltiples niveles de abstracción y refinar progresivamente los elementos grandes en elementos pequeños a lo largo del tiempo.

# [INVIERTE en buenas historias](#_bookmark3)

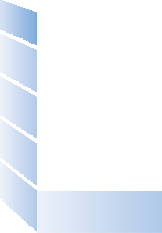
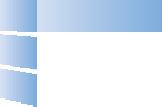
¿Cómo sabemos si los cuentos que hemos escrito son buenos cuentos? Bill Wake ha ofrecido seis criterios (resumidos por el acrónimo INVEST) que han demostrado ser útiles a la hora de evaluar si nuestras historias son aptas para el uso previsto o requieren algún trabajo adicional (Wake 2003).

Los criterios de **INVEST** son ***Independiente***, ***Negociable***, ***Valioso***, ***Estimable***, ***Pequeño*** (con el tamaño adecuado) y ***Comprobable***. Cuando combinamos la información derivada de la aplicación de cada criterio, obtenemos una imagen clara de los cambios adicionales que podríamos querer hacer en una historia, si es que los hay. Examinemos cada criterio.

### Independiente

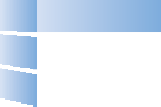
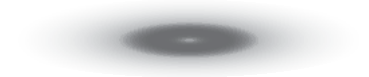
En la medida de lo posible, las historias de usuario deben ser *independientes* o, al menos, estar ligeramente acopladas entre sí. Las historias que exhiben un alto grado de interdependencia complican la estimación, la priorización y la planificación. Por ejemplo, en el lado izquierdo de la Figura 5.8, la historia #10 depende de muchas otras historias.

**Artículo**



**Dependencias**

**Historia #10**



**Artículo**

**Historia #9**

**Historia #10**

**Historia #11 Dependencias**

**FIGURA 5.8** Historias altamente dependientes

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, como se ilustra en el lado derecho de la figura 5.8. Tratar de determinar cómo priorizar todas estas historias y decidir en qué historias trabajar en un sprint sería difícil, por 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 inicial. En cambio, las historias son marcadores de posición para las conversaciones en las que 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 interesados y el equipo negocien los detalles.

Esta negociabilidad ayuda a todos los involucrados a evitar la mentalidad de "nosotros contra ellos", que es común con documentos detallados de requisitos iniciales. Cuando las historias son negociables, los desarrolladores no pueden decir: "Oye, si lo querías, deberías haberlo puesto en el documento", porque los detalles van a ser negociados con los desarrolladores. Y la gente de negocios realmente no puede decir: "Oye, obviamente

no entendió el documento de requisitos porque construyó la cosa equivocada", porque la gente de negocios estará en diálogo frecuente con los desarrolladores para asegurarse de que haya claridad compartida. Escribir historias negociables evita los problemas asociados con los requisitos detallados iniciales 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 sobre cómo. Cuando el *cómo* se vuelve innegociable, las oportunidades para que el equipo sea innovador disminuyen. El **despilfarro de innovación**  resultante podría tener consecuencias económicas devastadoras.

Sin embargo, hay ocasiones en las que  *la forma en que* se construye algo es realmente importante para el propietario del producto. Por ejemplo, puede haber una obligación reglamentaria de desarrollar una característica de una manera determinada, 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". Eso está bien; No todas las historias son totalmente negociables, pero la mayoría de las historias deberían serlo.

### Valioso

Las historias deben ser *valiosas* para un cliente*,* un usuario o ambos. Los clientes (o los que eligen) seleccionan y pagan el producto. Los usuarios realmente usan el producto. Si una historia no es valiosa para ninguno de los dos, no pertenece al backlog del producto. No puedo imaginarme diciendo: "La historia #10 no es valiosa para nadie, pero construyámosla de todos modos". Nosotros no haríamos eso. O bien reescribíamos la historia para hacerla valiosa para un cliente o usuario, o simplemente la descartábamos.

¿Qué hay de las 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 5.9?

El problema fundamental con las historias técnicas es que el propietario del producto puede no percibir ningún valor en ellas, lo que hace difícil, si no imposible, priorizarlas frente a las historias valiosas para el negocio. Para que exista una historia técnica, el propietario del producto debe entender por qué está pagando por ella y, por lo tanto, qué valor ofrecerá en última instancia.

|  |  |
| --- | --- |
| **Migrar a la nueva versión de Oracle** | |
| Como desarrollador, quiero migrar el archivo  sistema para trabajar con la última versión de  el DBMS de Oracle para que no estemos  operando en una versión que Oracle pronto se retirará. | |
|  |  |

**FIGURA 5.9** Ejemplo de historia técnica

En el caso de la historia "Migrar a la nueva versión de Oracle", es posible que el propietario del producto no entienda inicialmente por qué es valioso cambiar las bases de datos. Sin embargo, una vez que el equipo explica los riesgos de seguir desarrollando en una versión no compatible de una base de datos, el propietario del producto puede decidir que la migración de bases de datos es lo suficientemente valiosa como para aplazar la creación de algunas características nuevas hasta que se realice 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 concesiones informadas. Como resultado, esta historia técnica podría incluirse en el trabajo pendiente del producto.

Sin embargo, en la práctica, la mayoría de las historias técnicas (como la de la figura 5.10) no deben incluirse en el backlog del producto.

En cambio, este tipo de historias deben ser tareas asociadas con la realización de historias valiosas para el negocio. Si el equipo de desarrollo tiene una definición clara de lo que está hecho, no debería haber necesidad de escribir historias como estas, porque el trabajo está implícito en la definición de estar hecho.

El quid de los criterios *de valor* es que todas las historias en el backlog deben ser valiosas (vale la pena invertir en ellas) 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 para desarrollarlas que las historias más pequeñas).

Conocer el tamaño de una historia proporciona información procesable al equipo de Scrum. El propietario del producto, por ejemplo, necesita conocer el costo de una historia para determinar su prioridad final en el backlog del producto. El equipo Scrum, por otro lado, puede determinar a partir del tamaño de la historia si se requiere un refinamiento adicional o una desagregación. Una historia grande en la que planeamos trabajar pronto tendrá que dividirse en un conjunto de historias más pequeñas.

|  |  |
| --- | --- |
| **Compilaciones automáticas** | |
| Como desarrollador, quiero que las compilaciones  se ejecuta automáticamente cuando registro el código  para que se detecten los errores de regresión  cuando se introducen. | |
|  |  |

**FIGURA 5.10** Historia técnica indeseable

Si el equipo no puede dimensionar una historia, la historia es demasiado grande o ambigua para ser dimensionada, o el equipo no tiene suficiente conocimiento para estimar un tamaño. Si es demasiado grande, el equipo tendrá que trabajar con el propietario del producto para dividirlo en historias más manejables. Si el equipo carece de conocimientos, se necesitará algún tipo de actividad exploratoria para adquirir la información (discutiré este tema en breve).

### Tamaño adecuado (pequeño)

Las historias deben tener *el tamaño adecuado* para el momento en que planeamos trabajar en ellas. Las historias trabajadas en los sprints deben ser *pequeñas*. Si vamos a hacer un sprint de varias semanas, queremos trabajar en varias historias que tengan cada una 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 el hecho de que una historia sea grande no significa que sea mala. Digamos que tenemos una historia de tamaño épico en la que no planeamos trabajar hasta dentro de un año. Podría decirse que esa historia tiene el tamaño adecuado para cuando planeamos trabajar en ella. De hecho, si hoy pasáramos tiempo dividiendo esa epopeya en una colección de historias más pequeñas, fácilmente podría ser una completa pérdida de 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 a su tamaño. Hay que tener en cuenta *cuándo* se va a trabajar la historia a la hora de aplicar este criterio.

### Comprobables

Las historias deben ser *comprobables* de forma binaria, es decir, deben superar o no superar las pruebas asociadas. Ser testeable significa tener buenos criterios de aceptación (relacionados con las condiciones de satisfacción) asociados a la historia, que es el aspecto de "confirmación" de una historia de usuario que he comentado antes.

Sin criterios comprobables, ¿cómo sabríamos si la historia está terminada al final del sprint? Además, debido a que estas pruebas con frecuencia proporcionan detalles importantes de la historia, es posible que sean necesarias antes de que el equipo pueda estimar la historia.

Puede que no siempre sea necesario o posible probar una historia. Por ejemplo, es probable que las historias de tamaño épico no tengan pruebas asociadas, ni las necesiten (no construimos directamente las epopeyas).

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 se trate de 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, es posible que no haya un conjunto de pruebas que se puedan ejecutar cuando el sistema se ponga en producción que puedan demostrar que se ha cumplido este nivel de tiempo de actividad, pero el requisito sigue siendo valioso, ya que impulsará el diseño.

|  |  |
| --- | --- |
| **Internacionalización** | |
| Como usuario quiero una interfaz en inglés,  una lengua romance y una lengua compleja  de modo que exista una alta probabilidad estadística  que funcionará en los 70 requeridos | |
| Idiomas. |  |

|  |  |
| --- | --- |
| **Compatibilidad con el navegador tteb** | |
| El sistema debe ser compatible con IE8, IE9, Firefox 6, | |
| Firefox 7, Safari 5 y Chrome 15. | |
|  | |
|  | |
|  | |
|  |  |

**FIGURA 5.11** Requisitos no funcionales



# [No funcional Requisitos](#_bookmark3)

**Los requisitos no funcionales** representan restricciones a nivel del sistema. Con frecuencia escribo requisitos no funcionales como historias de usuario (véase el lado izquierdo de la Figura 5.11), pero no me siento obligado a hacerlo, especialmente si me parece incómodo o más conveniente escribirlos en un formato diferente (lado derecho de la Figura 5.11).

Como restricciones a nivel del sistema, los requisitos no funcionales son importantes porque afectan al diseño y las pruebas de la mayoría o de todas las historias del trabajo pendiente del producto. Por ejemplo, tener un requisito no funcional de "Soporte de navegador web" (lado derecho de la Figura 5.11) sería común en cualquier proyecto de sitio web. Cuando el equipo desarrolla las características del sitio web, debe asegurarse de que las características 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 "Compatibilidad con navegadores web" en la definición de finalización, el equipo tendrá que probar las nuevas características añadidas en el sprint con todos los navegadores enumerados. Si no funciona con todos ellos, la historia no está terminada.

Recomiendo que los equipos traten de incluir la mayor cantidad posible de requisitos no funcionales en sus definiciones de hecho. Esperar para probar los requisitos no funcionales hasta el final del esfuerzo de desarrollo difiere la obtención de una retroalimentación rápida sobre las características críticas de rendimiento del sistema.

# [Historias de adquisición de conocimiento](#_bookmark3)

A veces necesitamos crear un elemento de backlog del producto que se centre en la adquisición de conocimientos. Tal vez no tenemos suficiente conocimiento explotable sobre el producto o el proceso de construcción del producto para avanzar. Así que, como comenté en el capítulo 3, tenemos que explorar. Dicha exploración se conoce por muchos nombres: *prototipo*, *prueba de concepto*, *experimento*, *estudio*, *pico*, etc. Todas son básicamente actividades de exploración que implican la compra de información.

A menudo empleo una historia de usuario como marcador de posición para el trabajo de exploración (véase la figura 5.12).

En el ejemplo, el equipo desea evaluar dos arquitecturas posibles para el nuevo motor de filtrado. Propone crear prototipos de ambas arquitecturas y luego ejecutar pruebas de velocidad, escala y tipo contra ambos prototipos. El producto de la actividad de creación de prototipos será un breve memorándum que describa los experimentos que se realizaron, los resultados que se obtuvieron y la recomendación del equipo sobre cómo proceder.

Esta historia específica de adquisición de conocimientos parece una historia técnica, y como dije antes, el valor comercial de cualquier historia técnica tiene que ser justificable para el propietario del producto. Debido a que los propietarios de productos piensan en términos económicos, debe haber una justificación económica para hacer este trabajo de creación de prototipos. Es probable que exista un argumento técnico convincente para hacer una historia de adquisición de conocimientos, ya que el equipo está típicamente bloqueado para avanzar hasta que tenga el conocimiento producido por la historia. La pregunta para el equipo de Scrum es si el valor de la información adquirida excede el costo de obtenerla.

Así es como un equipo de Scrum podría abordar la respuesta a esa pregunta. En primer lugar, necesitamos conocer el coste de la creación de prototipos. Ningún buen propietario de producto autorizará la exploración ilimitada. Es posible que el equipo no pueda responder a preguntas particulares hasta que se haya tomado una decisión arquitectónica, pero debe ser capaz de responder a la pregunta de cuánto esfuerzo quiere dedicar a comprar la información necesaria para tomar la decisión arquitectónica. Por lo tanto, le pedimos al equipo que dimensione la historia de creación de prototipos.

Digamos que la estimación del tamaño indica que todo el equipo tendría que trabajar en la historia para un sprint. Sabemos quién está en el equipo y la duración del sprint, por lo que también sabemos el costo de adquirir la información. (Digamos que cuesta $ 10K). Ahora necesitamos conocer el valor de la información.

Esta es una forma en que podríamos estimar el valor. Imagina que lanzo una moneda al aire. Si sale cara, haremos la arquitectura A; si sale cruz, haremos la arquitectura B. Ahora, le pido al equipo que calcule el costo de equivocarse. Por ejemplo, si lanzo la moneda al aire y sale cara y comenzamos a construir características comerciales sobre la arquitectura A, y la arquitectura A resulta ser el enfoque equivocado, ¿cuál sería el costo de deshacer



|  |  |
| --- | --- |
| **Evaluación de la arquitectura del motor de filtrado** | |
| Como desarrollador, quiero crear dos prototipos  Alternativas para el nuevo motor de filtrado  para que sepa cuál es un mejor  Elección a largo plazo. | |
|  |  |



**Condiciones de Satisfacción**

Realice una prueba de velocidad en ambos prototipos. Realice una prueba a escala en ambos prototipos. Realice una prueba de tipo en ambos prototipos.

Escribe un breve memorándum que describa los experimentos, los resultados y las recomendaciones.

**FIGURA 5.12** Historia de la adquisición de conocimientos

la mala decisión y reconstruir todo encima de la arquitectura B? Digamos que el equipo estima que el costo es de $500K.

Ahora tenemos suficiente información para tomar una decisión económica sensata. ¿Estamos dispuestos a gastar $10K para comprar información que tiene un valor esperado de $250K (la mitad de las veces que lanzamos la moneda al aire estaríamos en lo cierto)? Claro, parece una decisión comercial sensata. Ahora el propietario del producto puede justificar por qué esta historia está en el backlog. Como ilustración final del uso de la economía para justificar las historias de adquisición de conocimiento, alteremos los números. ¿Qué pasa si la respuesta del equipo a "¿Cuánto costaría si nos equivocáramos?" es de 15.000 dólares? En este caso sería una mala decisión hacer la historia de prototipos. ¿Por qué gastar $10K para comprar información que tiene un valor esperado de $7.5K? Sería mejor que simplemente lanzáramos la moneda al aire (o hiciéramos una suposición informada) y, si nos equivocamos, simplemente rehiciéramos el trabajo usando la otra arquitectura. En realidad, dadas las tecnologías actuales en constante avance, este escenario no es tan descabellado como puede parecer. Es un ejemplo de lo que algunas personas llaman una  **estrategia de falla rápida** (intente algo, obtenga alimentación rápida-

y inspeccionar y adaptar rápidamente).

# [Recopilación de historias](#_bookmark3)

¿Cómo surgen las historias de usuario? Los enfoques tradicionales para la recopilación de requisitos implican preguntar a los usuarios qué quieren. Nunca he tenido mucho éxito con ese enfoque. En mi experiencia, los usuarios son mucho mejores críticos que autores.

Por lo tanto, si le preguntas a un usuario "¿Qué quieres?", puede que pueda o no responder. Incluso si ella responde a la pregunta y construimos exactamente lo que pidió, puede decir: "Sí, me diste exactamente lo que pedí , y ahora que lo veo, quiero algo diferente". Estoy seguro de que todos hemos tenido una experiencia así.

Un mejor enfoque es involucrar a los usuarios como parte del equipo que determina qué construir y revisa constantemente lo que se está construyendo. Para promover este nivel de participación, muchas organizaciones prefieren emplear talleres de escritura de historias de usuario como medio principal para generar al menos el conjunto inicial de historias de usuario. Algunos también emplean el mapeo de historias para organizar y proporcionar un contexto centrado en el usuario a sus historias . Describiré brevemente cada técnica.

### Taller de Escritura de Historias de Usuario

El objetivo de un taller de redacción de **historias de usuario** es hacer una lluvia de ideas colectivas sobre el valor comercial deseado y crear marcadores de posición de historias de usuario para lo que se supone que debe hacer el producto o servicio.

El taller suele incluir al propietario del producto, al ScrumMaster y al equipo de desarrollo, junto con las partes interesadas internas y externas. La mayoría de los talleres duran desde unas pocas horas hasta unos pocos días. Pocas veces los he visto durar más, ni creo que deban hacerlo. El objetivo no es generar un conjunto completo y completo de usuarios

historias por adelantado (similar a una especificación completa de requisitos en un proyecto de desarrollo secuencial). En cambio, el taller suele tener un enfoque específico. Por ejemplo, con frecuencia realizo un taller junto con la planificación inicial del lanzamiento para generar un conjunto de historias candidatas para el próximo lanzamiento (consulte el Capítulo 18 para obtener más detalles).

Si es el primer taller, suelo empezar realizando un análisis de roles de usuario. El objetivo es determinar la colección de roles de usuario que se pueden usar para rellenar la parte de rol  **de usuario** de nuestras historias ("Como <rol de usuario>, quiero..."). Por supuesto, la gente de marketing o de investigación de mercados podría haber creado una buena definición de nuestros usuarios en una actividad separada antes del taller de escritura de historias.

También podemos tener **personajes**, que son individuos prototípicos que representan las características centrales de un rol. Por ejemplo, "Lilly", junto con su descripción asociada, podría ser el personaje correspondiente al papel de la jugadora de siete a nueve años de un videojuego de niñas. Una vez que Lilly esté definida, escribiríamos historias con Lilly en la posición de rol de usuario, en lugar de un rol más abstracto como "Jugadora Joven". Por ejemplo, "Como Lilly, quiero elegir entre muchos vestidos diferentes para poder personalizar mi avatar a mi gusto".

Durante el taller no existe una forma estándar de generar historias de usuario. Algunos equipos prefieren trabajar de arriba hacia abajo y otros prefieren trabajar de abajo hacia arriba. El enfoque de arriba hacia abajo implica que el equipo comience con una historia grande (como una epopeya) y luego concentre sus esfuerzos en generar una colección razonable de historias más pequeñas asociadas con la epopeya.

Una alternativa es trabajar de abajo hacia arriba y comenzar de inmediato a hacer una lluvia de ideas sobre las historias que están asociadas con la próxima versión de un sistema existente. No hay un enfoque correcto o incorrecto; Utilice cualquier enfoque que funcione bien, o cambie de enfoque para obtener lo mejor de ambos.

### Mapeo de historias

**El mapeo de historias** es una técnica popularizada por Jeff Patton (Patton, 2009) que adopta una perspectiva centrada en el usuario para generar un conjunto de historias de usuario. La idea básica es descomponer la actividad del usuario de alto nivel en un flujo de trabajo que se pueda descomponer aún más en un conjunto de tareas detalladas (consulte la figura 5.13).

Patton utiliza términos como *actividad*, *tarea* y *subtarea* para describir la jerarquía dentro de un story map. Para ser coherente con la terminología que presenté anteriormente, utilizo *épica*, *tema* e *historia sprintable*.

En el nivel más alto se encuentran las epopeyas, que representan las grandes actividades de valor económico medible para el usuario, por ejemplo, la epopeya "Comprar un producto".

A continuación, pensamos en la secuencia o flujo de trabajo común de las tareas del usuario que componen la epopeya (representada por temas, colecciones de historias relacionadas). Establecemos los temas a lo largo de una línea de tiempo, donde los temas del flujo de trabajo que naturalmente ocurrirían antes se colocan a la izquierda de los que ocurrirían más tarde. Por ejemplo, el tema "Buscar producto" estaría a la izquierda del tema "Administrar carrito de compras".



Búsqueda por ISBN

**Cuento**

**Prioridad**



**Epopeya**

Comprar un producto



**Epopeya**

**ttorkflow o secuencia de uso (a lo largo del tiempo)**



**Cuento**

**Cuento**

**Cuento**

**tfheme**

**tfheme**



Buscar productos Administrar compras

Carro

**Cuento**

Buscar por nombre Añadir a la cesta

Búsqueda por color Quitar de la cesta

**Cuento**

Buscar por autor Precio del carrito

**Cuento**

**Cuento**

**Cuento**

**Cuento**

**Cuento**

**Cuento**

**tfheme**

**tfheme**

**tfheme**

**FIGURA 5.13** Mapa de la historia

Luego, cada tema se descompone en un conjunto de historias implementables que se organizan verticalmente en orden de prioridad (realmente deseable porque es poco probable que las historias se hayan estimado aún y no podemos saber realmente la prioridad final hasta que sepamos el costo). No es necesario que todas las historias de un tema se incluyan en la misma versión. Por ejemplo, es posible que la historia de "Búsqueda por color" no esté programada para el primer lanzamiento, mientras que la historia de "Búsqueda por nombre" probablemente sí lo esté.

El mapeo de historias combina los conceptos de diseño centrado en el usuario con la descomposición de historias. Los buenos story maps muestran un flujo de actividades desde la perspectiva de los usuarios y proporcionan un contexto para comprender las historias individuales y su relación con unidades más grandes de valor para el cliente.

Incluso si no haces un mapeo formal de historias, la idea de usar flujos de trabajo me parece útil durante mis talleres de escritura de historias. Centran la discusión en la escritura de historias en el contexto de la entrega de un flujo de trabajo completo de valor para el usuario.

Al tener el contexto del flujo de trabajo, es más fácil para nosotros determinar si hemos pasado por alto alguna historia importante asociada con el flujo de trabajo.

Una diferencia entre los talleres tradicionales de escritura de historias y el mapeo de historias es que durante el taller nos enfocamos principalmente en generar historias y no tanto en priorizarlas (la posición vertical de las historias implementables dentro de un mapa de historias). Por lo tanto, podríamos utilizar el story mapping como complemento del taller, como una técnica para ayudar a visualizar la priorización de las historias. Los story maps proporcionan una vista bidimensional de un trabajo pendiente del producto en lugar de la representación lineal (unidimensional) tradicional del trabajo pendiente del producto.

# [Cierre](#_bookmark3)

En este capítulo discutí cómo los requisitos se tratan de manera diferente en un proyecto de Scrum que en un proyecto de desarrollo secuencial tradicional. En un esfuerzo de desarrollo que utiliza Scrum, creamos marcadores de posición para los requisitos denominados elementos de trabajo pendiente del producto. Estos elementos se expresan con frecuencia como historias de usuario y fluyen a través del proceso de Scrum con un enfoque distinto en las conversaciones como una forma de elaborar los detalles de los requisitos. También empleamos una estrategia de refinar progresivamente historias más grandes y menos detalladas en historias más pequeñas y detalladas justo a tiempo.

A continuación, presenté formalmente las historias de usuario describiéndolas en el contexto de "tarjeta, conversación y confirmación". A continuación, hablé de cómo se pueden utilizar las historias de usuario para representar el valor empresarial en múltiples niveles de abstracción. A continuación, expliqué cómo los criterios INVEST son útiles para determinar si tenemos buenas historias de usuario. A continuación, presenté formas de hacer frente a los requisitos no funcionales y a las actividades de adquisición de conocimientos. Concluí con una discusión sobre cómo recopilar historias de usuarios, centrada en talleres de escritura de historias de usuarios y mapeo de historias. En el próximo capítulo hablaré del backlog del producto.