Capítulo 1

Una visión general

Los requisitos de software son un problema de comunicación. Quienes desean lo nuevo...
El software (ya sea para usar o vender) debe comunicarse con quienes lo construirán.
El nuevo software. Para tener éxito, un proyecto se basa en la información de los jefes de
personas muy diferentes: por un lado están los clientes y usuarios y, a veces, los analistas, los expertos en
el dominio y otros que ven el software desde una perspectiva empresarial o
perspectiva organizacional; por otro lado está el equipo técnico.

Si cualquiera de las partes domina estas comunicaciones, el proyecto pierde. Cuando el
El lado comercial domina, exige funcionalidad y fechas con poca preocupación
Que los desarrolladores puedan cumplir ambos objetivos, o si comprenden exactamente lo que se necesita.
Cuando los desarrolladores dominan las comunicaciones, la jerga técnica reemplaza el lenguaje del
negocio y de los desarrolladores.
perder la oportunidad de aprender lo que se necesita escuchando.

Lo que necesitamos es una manera de trabajar juntos para que ninguna de las partes domine y
De modo que la cuestión emocional y política de la asignación de recursos
se convierte en un problema compartido. Los proyectos fracasan cuando el problema de la asignación de
recursos recae completamente en un solo lado. Si los desarrolladores asumen el problema (generalmente en
la forma de que les digan “No me importa cómo lo hagas, pero hazlo todo para junio”).
puede negociar la calidad por características adicionales, puede implementar solo parcialmente una
característica o puede tomar únicamente cualquiera de una serie de decisiones en las que los clientes
y los usuarios deben participar. Cuando los clientes y los usuarios asumen la carga de
En cuanto a la asignación de recursos, solemos ver una larga serie de discusiones al comienzo de cada...
un proyecto durante el cual se eliminan progresivamente características del proyecto.
Luego, cuando finalmente se entrega el software, tiene incluso menos funcionalidad.
que el conjunto reducido que se identificó.

A estas alturas, hemos aprendido que no podemos predecir con exactitud un proyecto de desarrollo de
software. A medida que los usuarios ven las primeras versiones del software, se les ocurren...
Las nuevas ideas y sus opiniones cambian. Debido a la intangibilidad del software,
La mayoría de los desarrolladores tienen notoriamente dificultades para estimar cuánto tiempo durarán las cosas.
Debido a estos y otros factores, no podemos diseñar un PERT perfecto.
Gráfico que muestra todo lo que se debe hacer en un proyecto.

Entonces ¿qué hacemos?
Tomamos decisiones basándonos en la información que tenemos a mano. Y lo hacemos.
A menudo. En lugar de tomar un conjunto único de decisiones al inicio de un proyecto, la toma de decisiones se distribuye a lo largo de su duración. Para ello, nos aseguramos de contar con un proceso que nos proporcione información lo antes posible y con la mayor frecuencia posible. Y aquí es donde entran en juego las historias de usuario.

¿Qué es una historia de usuario?

Una historia de usuario describe una funcionalidad valiosa para el usuario o comprador de un sistema o software. Las historias de usuario se componen de tres aspectos:

- una descripción escrita de la historia utilizada para la planificación y como recordatorio
- conversaciones sobre la historia que sirven para desarrollar los detalles de la historia
- pruebas que transmiten y documentan detalles y que pueden utilizarse para determinar cuando una historia está completa

Dado que las descripciones de las historias de usuario se escriben tradicionalmente a mano en tarjetas de papel, Ron Jeffries ha denominado estos tres aspectos con la maravillosa aliteración de Tarjeta, Conversación y Confirmación (Jeffries, 2001). La Tarjeta puede ser la manifestación más visible de una historia de usuario, pero no es la más importante.
Rachel Davies (2001) afirmó que las tarjetas «representan los requisitos del cliente en lugar de documentarlos ». Esta es la manera perfecta de pensar en las historias de usuario: si bien la tarjeta puede contener el texto de la historia, los detalles se elaboran en la conversación y se registran en la confirmación.

Como ejemplo de historia de usuario, consulte la Tarjeta de historia 1.1, que es una tarjeta de historia del hipotético sitio web de búsqueda y publicación de empleos BigMoneyJobs.

> Un usuario puede publicar su currículum en el sitio web.

■ Tarjeta de historia 1.1 Una historia de usuario inicial escrita en una tarjeta de notas.

Para mantener la coherencia, muchos de los ejemplos del resto de este libro se referirán al sitio web de BigMoneyJobs. Otros ejemplos de historias para BigMoneyJobs podrían incluir:

• Un usuario puede buscar empleos.

• Una empresa puede publicar nuevas ofertas de empleo.

• Un usuario puede limitar quién puede ver su currículum.

Debido a que las historias de usuario representan una funcionalidad que los usuarios valorarán, Los siguientes ejemplos no constituyen buenas historias de usuario para este sistema:

• El software estará escrito en C++.

• El programa se conectará a la base de datos a través de un pool de conexiones.

El primer ejemplo no es una buena historia de usuario para BigMoneyJobs, ya que a sus usuarios no les importaría el lenguaje de programación utilizado. Sin embargo, si se tratara de una interfaz de programación de aplicaciones, la usuaria de ese sistema (una programadora) podría perfectamente haber escrito que «el software estará escrito en C++».

La segunda historia no es una buena historia de usuario en este caso porque a los usuarios de este sistema no les importan los detalles técnicos de cómo la aplicación se conecta a la base de datos.

Quizás hayas leído estas historias y estés gritando: "¡Un momento! ¡Usar un pool de conexiones es un requisito en mi sistema!". Si es así, espera, la clave es que las historias se escriban de forma que el cliente las valore. Hay maneras de expresar historias como estas de forma que sean valiosas para el cliente. Veremos ejemplos en el Capítulo 2, "Cómo escribir historias".

¿Dónde están los detalles?

Una cosa es decir "Un usuario puede buscar trabajos". Otra muy distinta es poder empezar a programar y probar con solo eso como guía. ¿Dónde están los detalles? ¿Qué pasa con todas las preguntas sin respuesta como:

• ¿Qué valores pueden buscar los usuarios? ¿Estado? ¿Ciudad? ¿Puesto de trabajo? ¿Palabras clave?

• ¿El usuario debe ser miembro del sitio?

• ¿Se pueden guardar los parámetros de búsqueda?

• ¿Qué información se muestra para los trabajos que coinciden?

Muchos de estos detalles pueden expresarse como historias adicionales. De hecho, es mejor tener más historias que historias demasiado extensas. Por ejemplo, el sitio web completo de BigMoneyJobs probablemente se describe con estas dos historias:

- Un usuario puede buscar un trabajo.
- Una empresa puede publicar ofertas de empleo.

Claramente, estas dos historias son demasiado extensas para ser de mucha utilidad. El capítulo 2, "Escribiendo historias", aborda a fondo la cuestión del tamaño de las historias, pero como punto de partida, conviene contar con historias que puedan codificarse y probarse entre medio día y quizás dos semanas por uno o dos programadores. Si se interpretan con liberalidad, las dos historias anteriores podrían cubrir fácilmente la mayor parte del sitio web de BigMoneyJobs, por lo que cada una probablemente les llevará a la mayoría de los programadores más de una semana.

Cuando una historia es demasiado extensa, a veces se denomina épica . Las épicas pueden dividirse en dos o más historias más pequeñas. Por ejemplo, la épica «Un usuario puede buscar trabajo» podría dividirse en las siguientes historias:

- Un usuario puede buscar empleos por atributos como ubicación, rango salarial, puesto de trabajo, nombre de la empresa y fecha en que se publicó el empleo.
- Un usuario puede ver información sobre cada trabajo que coincida con una búsqueda.
- Un usuario puede ver información detallada sobre una empresa que ha publicado una trabajo.

Sin embargo, no seguimos dividiendo las historias hasta que tengamos una que cubra hasta el último detalle. Por ejemplo, la historia «Un usuario puede ver información sobre cada empleo que coincida con una búsqueda» es muy razonable y realista.
No necesitamos dividirlo más en:

- Un usuario puede ver una descripción del trabajo.
- Un usuario puede ver el rango salarial de un trabajo.
- Un usuario puede ver la ubicación de un trabajo.

De manera similar, no es necesario ampliar la historia del usuario en los requisitos típicos. Documentación de mentos en un estilo como este:

4.6) Un usuario puede ver información sobre cada trabajo que coincida con una búsqueda.

4.6.1) Un usuario puede ver la descripción del trabajo.
4.6.2) Un usuario puede ver el rango salarial de un trabajo.
4.6.3) Un usuario puede ver la ubicación de un trabajo.

En lugar de escribir todos estos detalles como historias, la mejor estrategia es que el equipo de desarrollo y el cliente los discutan. Es decir, conversar sobre ellos en el momento en que se vuelven importantes.

No hay nada de malo en hacer algunas anotaciones en una tarjeta de historia según

Una discusión, como se muestra en la Tarjeta de Historia 1.2. Sin embargo, la conversación es la clave, no la nota en la tarjeta. Ni los desarrolladores ni el cliente pueden señalar la tarjeta tres meses después y decir: "Pero, ¿ves?, ahí lo dije". Las historias no son obligaciones contractuales. Como veremos, los acuerdos se documentan mediante pruebas que demuestran que una historia se ha desarrollado correctamente.

Los usuarios pueden ver información sobre cada trabajo que coincida con una búsqueda.

Marco dice mostrar descripción, salario y ubicación.

■ Tarjeta de historia 1.2 Una tarjeta de historia con una nota.

"¿Cuánto tiempo tiene que durar?"

Yo era el chico de literatura de la secundaria que siempre preguntaba: "¿Cuánto tiene que durar?" cada vez que nos mandaban escribir un trabajo. A los profesores nunca les gustó la pregunta, pero aun así creo que era justa porque me indicaba cuáles eran sus expectativas. Es igual de importante comprender las expectativas de los usuarios de un proyecto. Esas expectativas se reflejan mejor en las pruebas de aceptación.

Si usa tarjetas de notas de papel, puede voltearlas y anotar estas expectativas. Estas expectativas se escriben como recordatorios sobre cómo probar la historia, como se muestra en la Tarjeta de Historia 1.3. Si usa un sistema electrónico, probablemente tenga un espacio para anotar los recordatorios de la prueba de aceptación.

Pruébelo con una descripción de trabajo vacía.

Pruébelo con una descripción de trabajo realmente larga.

Pruébelo con un salario faltante.

Pruébelo con un salario de seis dígitos.

■ Tarjeta de historia 1.3 El reverso de una tarjeta de historia contiene recordatorios sobre cómo evaluar la historia.

Las descripciones de las pruebas deben ser breves e incompletas. Se pueden añadir o eliminar pruebas en cualquier momento. El objetivo es proporcionar información adicional sobre la historia para que los desarrolladores sepan cuándo han terminado. Así como las expectativas de mi profesor me resultaron útiles para saber cuándo había terminado de escribir sobre Moby Dick, a los desarrolladores les resulta útil conocer las expectativas del cliente para saber cuándo han terminado.

El equipo del cliente

En un proyecto ideal, contaríamos con una sola persona que prioriza el trabajo de los desarrolladores, responde a sus preguntas con total claridad, utiliza el software una vez terminado y escribe todas las historias. Esto casi siempre es demasiado, por lo que formamos un equipo de atención al cliente. Este equipo incluye a quienes se aseguran de que el software satisfaga las necesidades de los usuarios previstos. Esto significa que el equipo de atención al cliente puede incluir testers, un gerente de producto, usuarios reales y diseñadores de interacción.

¿Cómo será el proceso?

Un proyecto que utiliza historias tendrá una sensación y un ritmo diferentes a los que estás acostumbrado. Un proceso tradicional en cascada conlleva un ciclo de redacción de todos los requisitos, análisis de los mismos, diseño de una solución, codificación de la solución y, finalmente, pruebas. Con frecuencia, durante este tipo de proceso, los clientes y usuarios participan al principio en la redacción de los requisitos y al final en la aceptación del software, pero la participación de usuarios y clientes puede desaparecer casi por completo entre la redacción de los requisitos y la aceptación. A estas alturas, hemos aprendido que esto no funciona.

Lo primero que notará en un proyecto basado en historias es que los clientes y usuarios permanecen involucrados durante todo el proyecto. No se espera (¡ni se permite!) que desaparezcan a mitad del proyecto. Esto se aplica tanto si el equipo utiliza Programación Extrema (XP; consulte el Apéndice A, "Resumen de la Programación Extrema" para más información), una versión ágil del Proceso Unificado, un proceso ágil como Scrum (consulte el Capítulo 15, "Uso de Historias con Scrum") o un proceso ágil propio basado en historias.

Los clientes y los usuarios previstos del nuevo software deberían planificar un papel muy activo en la redacción de las historias de usuario, especialmente si utilizan XP. El proceso de redacción de historias se inicia mejor considerando los tipos de usuarios previstos.

sistema. Por ejemplo, si está creando un sitio web de reservas de viajes, puede
Tienen tipos de usuarios como viajeros frecuentes, planificadores de vacaciones, etc. El equipo de atención al cliente
debe incluir representantes de tantos de estos tipos de usuarios como sea posible.
práctico. Pero cuando no se puede, el modelado de roles del usuario puede ayudar. (Para más información sobre esto

(Para más información sobre el tema, consulte el Capítulo 3, “Modelado de roles de usuario”).

¿Por qué el cliente escribe las historias?

El equipo del cliente, en lugar de los desarrolladores, escribe las historias de usuario por dos razones principales. En primer lugar, cada historia debe estar escrita en el lenguaje del negocio, no... en jerga técnica, para que el equipo del cliente pueda priorizar las historias para su inclusión En iteraciones y lanzamientos. En segundo lugar, como principales visionarios del producto, el equipo del cliente es el más indicado para describir el comportamiento del producto.

Las historias iniciales de un proyecto a menudo se escriben en un taller de escritura de historias, pero
Se pueden escribir historias en cualquier momento del proyecto. Durante el taller de escritura de historias, todos
intercambian ideas sobre tantas historias como sea posible. Armados con un
Conjunto inicial de historias, los desarrolladores estiman el tamaño de cada una.

De manera colaborativa, el equipo del cliente y los desarrolladores seleccionan una duración de iteración,
de quizás una a cuatro semanas. Se utilizará la misma duración de iteración para el
Duración del proyecto. Al final de cada iteración, los desarrolladores estarán
Responsable de entregar código totalmente utilizable para algún subconjunto de la aplicación.
El equipo del cliente permanece muy involucrado durante la iteración, hablando con
Los desarrolladores sobre las historias que se están desarrollando durante esa iteración. Durante
En la iteración, el equipo del cliente también especifica las pruebas y trabaja con los desarrolladores para
automatizarlas y ejecutarlas. Además, el equipo del cliente se asegura de que...
El proyecto avanza constantemente hacia la entrega del producto deseado.

Una vez que se haya seleccionado la longitud de iteración, los desarrolladores estimarán cómo
cuánto trabajo podrán realizar por iteración. A esto lo llamamos velocidad. El equipo...
La primera estimación de la velocidad será errónea porque no hay forma de saber la velocidad.
con antelación. Sin embargo, podemos utilizar la estimación inicial para hacer un boceto aproximado, o
plan de lanzamiento, de qué trabajo se realizará en cada iteración y cuántas iteraciones serán necesarias.

Para planificar un lanzamiento, clasificamos las historias en varias pilas, cada una de las cuales representa una
iteración. Cada pila contendrá un número determinado de historias, las estimaciones para
que suman solo la velocidad estimada. Las historias de mayor prioridad se colocan en la primera pila. Cuando esta
pila está llena, las siguientes historias de mayor prioridad...
van a una segunda pila (que representa la segunda iteración). Esto continúa hasta

Has creado tantas pilas que te has quedado sin tiempo para el proyecto, o hasta que las pilas representen una nueva versión deseable del producto. (Para más información sobre estos temas, consulta el Capítulo 9, «Planificación de una versión» y el Capítulo 10, «Planificación de una iteración»).

Antes del inicio de cada iteración, el equipo del cliente puede tomar decisiones a mitad de camino. Correcciones al plan. A medida que se completan las iteraciones, conocemos la velocidad real del equipo de desarrollo y podemos trabajar con ella en lugar de la velocidad estimada.
Esto significa que cada pila de historias podría necesitar ajustarse añadiendo o eliminando historias. Además, algunas historias resultarán mucho más fáciles de lo previsto, lo que significa que el equipo a veces querrá que se le asigne una historia adicional para esa iteración. Pero algunas historias serán más difíciles de lo previsto, lo que significa que parte del trabajo deberá trasladarse a iteraciones posteriores o fuera de ellas.
del lanzamiento en su totalidad.

Planificación de versiones e iteraciones

Una versión se compone de una o más iteraciones. La planificación de la versión se refiere a determinar un equilibrio entre el cronograma proyectado y el conjunto de funcionalidades deseado.
La planificación de la iteración se refiere a la selección de historias para su inclusión en esta iteración. Tanto el equipo del cliente como los desarrolladores participan en la planificación del lanzamiento y la iteración.

Para planificar un lanzamiento, el equipo del cliente comienza priorizando las historias. Mientras Al priorizar, querrán considerar:

- La conveniencia de la función para una amplia base de usuarios o clientes
- La conveniencia de la función para un pequeño número de usuarios o clientes importantes. clientes
- La cohesión de la historia en relación con otras historias. Por ejemplo, una historia de "alejamiento" puede no ser prioritaria por sí sola, pero puede tratarse como tal porque complementa a una historia de "acercamiento", que sí lo es.

Los desarrolladores tienen diferentes prioridades para muchas historias. Pueden sugerir que se cambie la prioridad de una historia en función de su riesgo técnico o porque complementa otra. El equipo del cliente escucha sus opiniones, pero prioriza las historias de la manera que maximiza el valor aportado a la organización.

No se pueden priorizar las historias sin considerar su costo. Mi prioridad para vacacionar el verano pasado fue Tahití hasta que consideré su costo. En ese momento...

Otras ubicaciones ascendieron en prioridad. El costo se considera en la priorización.
de cada historia. El costo de una historia es la estimación que le dan los desarrolladores.
A cada historia se le asigna una estimación en puntos de historia, que indica el tamaño y
complejidad de la historia en relación con otras historias. Por lo tanto, una historia estimada en cuatro
Se espera que los puntos de la historia tomen el doble de tiempo que una historia estimada en dos pisos.
agujas.

El plan de lanzamiento se construye asignando historias a las iteraciones del lanzamiento.
Los desarrolladores indican su velocidad esperada, que es el número de puntos de la historia.
Creen que completarán cada iteración. El cliente asigna historias.
a las iteraciones, asegurándose de que la cantidad de puntos de historia asignados a cada una
La iteración no excede la velocidad esperada del equipo.

A modo de ejemplo, supongamos que la Tabla 1.1 enumera todas las historias de su proyecto y
Se ordenan por orden de prioridad descendente. El equipo estima una velocidad de
trece puntos de historia por iteración. Las historias se asignarían a las iteraciones como
se muestra en la Tabla 1.2.

Tabla 1.1 Historias de muestra y sus costos.

Historia	Puntos de la historia
Historia A	3
Historia B	5
Historia C	5
Historia D	3
Historia E	1
Historia F	8
Historia G	5
Historia H	5
Historia I	5
Historia J	2

Debido a que el equipo espera una velocidad de trece, no se puede planificar ninguna iteración.
tener más de trece puntos de historia. Esto significa que el segundo y
Se planea que las terceras iteraciones tengan solo doce puntos de historia. No te preocupes por...
—la estimación rara vez es lo suficientemente precisa como para que esta diferencia importe, y si
Los desarrolladores van más rápido de lo planeado y pedirán una o dos historias más pequeñas.
Tenga en cuenta que, para la tercera iteración, el equipo del cliente ha elegido

incluir la Historia J sobre la Historia I, de mayor prioridad. Esto se debe a que la Historia I, a las cinco
Los puntos de la historia son en realidad demasiado grandes para incluirlos en la tercera iteración.

Tabla 1.2 Un plan de lanzamiento para las historias de la Tabla 1.1.

Iteración	Historias	Puntos de la historia
Iteración 1	A, B, C	13
Iteración 2	Re, Mi, Fa	12
Iteración 3	G, H, J	12
Iteración 4	I	5

Una alternativa a omitir temporalmente una historia grande y poner una más pequeña
Uno en su lugar en una iteración es dividir la historia grande en dos historias. Supongamos
que la Historia I de cinco puntos podría haberse dividido en la Historia Y (tres puntos) y
Historia Z (dos puntos). La historia Y contiene las partes más importantes del antiguo
Historia I y ahora puede encajar en la tercera iteración, como se muestra en la Tabla 1.3. Para obtener asesoramiento
Para saber cómo y cuándo dividir historias, consulte el Capítulo 2, “Cómo escribir historias”, y el Capítulo 3, “Cómo escribir historias”.
7, “Directrices para buenas historias”.

Tabla 1.3 División de una historia para crear un mejor plan de lanzamiento.

Iteración	Historias	Puntos de la historia
Iteración 1	A, B, C	13
Iteración 2	Re, Mi, Fa	12
Iteración 3	G, H, Y	13
Iteración 4	J, Z	4

¿Qué son las pruebas de aceptación?

La prueba de aceptación es el proceso de verificar que las historias se desarrollaron de tal manera
que cada uno funciona exactamente como el equipo del cliente esperaba que funcionara. Una vez
comienza una iteración, los desarrolladores comienzan a codificar y el equipo del cliente comienza
Especificar pruebas. Dependiendo de la competencia técnica de los miembros del equipo del cliente, esto puede
significar cualquier cosa, desde escribir pruebas en el reverso de la tarjeta de historia hasta
Introducir las pruebas en una herramienta de pruebas automatizada. Un tester dedicado y cualificado
Debería incluirse en el equipo del cliente para las tareas más técnicas.

Las pruebas deben escribirse lo más temprano posible en una iteración (o incluso ligeramente antes).
Antes de la iteración, si te sientes cómodo haciendo una pequeña conjetura sobre lo que habrá

La próxima iteración). Escribir pruebas con antelación es extremadamente útil porque las suposiciones y expectativas del equipo del cliente se comunican con mayor antelación a los desarrolladores. Por ejemplo, supongamos que escribe la historia "Un usuario puede pagar los artículos de su carrito de compras con tarjeta de crédito". Luego, escribe estas sencillas pruebas en el reverso de esa tarjeta de historia:

- Prueba con Visa, MasterCard y American Express (aprobado).
- Prueba con Diner's Club (reprobado).
- Prueba con tarjeta débito Visa (pase).
- Pruebe con los números de identificación de tarjetas buenos, malos y faltantes del reverso de la tarjeta.
- Prueba con tarjetas vencidas.
- Prueba con diferentes montos de compra (incluyendo uno que supere el límite de la tarjeta).

Estas pruebas capturan las expectativas de que el sistema manejará Visa, MasterCard y American Express y no permitirá compras con otras tarjetas.
Al entregar estas pruebas al programador con anticipación, el equipo del cliente no solo ha expresado sus expectativas, sino que también podría haberle recordado una situación que de otro modo habría olvidado. Por ejemplo, podría haber olvidado considerar las tarjetas caducadas. Anotarlo como prueba en el reverso de la tarjeta antes de empezar a programar le ahorrará tiempo. Para más información sobre cómo escribir pruebas de aceptación para historias, consulte el capítulo 6, "Pruebas de aceptación en historias de usuario".

¿Por qué cambiar?

Llegados a este punto, quizás te preguntes por qué cambiar. ¿Por qué escribir tarjetas de historias y mantener todas estas conversaciones? ¿Por qué no seguir escribiendo documentos de requisitos o casos de uso? Las historias de usuario ofrecen varias ventajas sobre enfoques alternativos. Encontrarás más detalles en el Capítulo 13, "¿Por qué usar historias de usuario?". Algunas de las razones son:

- Las historias de usuario enfatizan la comunicación verbal en lugar de la escrita.
- Las historias de usuario son comprensibles tanto para usted como para los desarrolladores.
- Las historias de usuario tienen el tamaño adecuado para la planificación.
- Las historias de usuario funcionan para el desarrollo iterativo.

• Las historias de usuario incentivan a posponer los detalles hasta tener la mejor comprensión posible de lo que realmente se necesita.

Debido a que las historias de usuario desplazan el énfasis hacia la conversación y lo alejan de la escritura, las decisiones importantes no se capturan en documentos que probablemente no se lean. En cambio, los aspectos importantes de las historias se capturan en pruebas de aceptación automatizadas y se ejecutan con frecuencia. Además, evitamos documentos escritos confusos con afirmaciones como:

> El sistema debe almacenar una dirección y un número de teléfono comercial o un número de teléfono móvil.

¿Qué significa eso? Podría significar que el sistema debe almacenar uno de estos:

> (Dirección y teléfono de trabajo) o teléfono móvil Dirección
> y (teléfono de trabajo o teléfono móvil)

Debido a que las historias de usuario están libres de jerga técnica (recuerde, el equipo del cliente las escribe), son comprensibles tanto para los desarrolladores como para el equipo del cliente.

Cada historia de usuario representa una funcionalidad discreta; es decir, algo que un usuario probablemente haría en un solo entorno. Esto hace que las historias de usuario sean una herramienta de planificación adecuada. Se puede evaluar el valor de cambiar las historias entre versiones mucho mejor que el impacto de omitir una o más declaraciones del tipo «El sistema deberá…».

Un proceso iterativo es aquel que avanza mediante refinamientos sucesivos. Un equipo de desarrollo realiza un primer análisis de un sistema, sabiendo que está incompleto o presenta deficiencias en algunas (quizás muchas) áreas. Luego, refinan esas áreas sucesivamente hasta que el producto es satisfactorio. Con cada iteración, el software se mejora mediante la adición de mayor detalle. Las historias funcionan bien para el desarrollo iterativo porque también es posible iterar sobre ellas. Para una característica que se desea con el tiempo, pero que no es importante en este momento, primero se puede escribir una historia extensa (una epopeya). Cuando se esté listo para incorporar esa historia al sistema, se puede refinar fragmentándola y reemplazándola con historias más cortas que sean más fáciles de manejar.

Es esta capacidad de iterar sobre un conjunto de historias lo que permite que estas fomenten la postergación de los detalles. Dado que hoy podemos escribir una épica provisional, no es necesario escribir historias sobre partes de un sistema hasta que se acerque su desarrollo. Postergar los detalles es importante porque nos permite no dedicar tiempo a pensar en una nueva característica hasta estar seguros de que es necesaria.

Las historias nos disuaden de fingir que podemos saberlo todo y escribirlo todo de antemano. En cambio, fomentan un proceso mediante el cual el software se perfecciona iterativamente, basándose en conversaciones entre el equipo del cliente y los desarrolladores.

Resumen

- Una tarjeta de historia contiene una breve descripción de las funciones valoradas por el usuario o el cliente.
  cionalidad.
- Una tarjeta de historia es la parte visible de una historia, pero las partes importantes son las
  Conversaciones entre el cliente y los desarrolladores sobre la historia.
- El equipo del cliente incluye a quienes se aseguran de que el software cumpla con las expectativas.
  Las necesidades de los usuarios previstos. Esto puede incluir probadores, un gerente de producto,
  usuarios reales y diseñadores de interacción.
- El equipo del cliente escribe las tarjetas de historia porque están en la mejor posición para expresar las
  características deseadas y porque luego deben poder
  Trabajar los detalles de la historia con los desarrolladores y priorizar las historias.
- Las historias se priorizan según su valor para la organización.
- Los lanzamientos y las iteraciones se planifican colocando historias en iteraciones.
- La velocidad es la cantidad de trabajo que los desarrolladores pueden completar en una iteración.
  ción.
- La suma de las estimaciones de las historias colocadas en una iteración no puede exceder
  la velocidad que los desarrolladores pronostican para esa iteración.
- Si una historia no cabe en una iteración, puedes dividirla en dos o más.
  Historias más pequeñas.
- Las pruebas de aceptación validan que una historia se ha desarrollado con la funcionalidad que el equipo
  del cliente tenía en mente cuando escribió la historia.
- Vale la pena usar las historias de usuario porque enfatizan la comunicación verbal, pueden ser entendidas
  por igual por usted y los desarrolladores y pueden usarse para
  planificar iteraciones, trabajar bien dentro de un proceso de desarrollo iterativo,
  y porque incentivan la postergación de los detalles.

Preguntas

1.1 ¿Cuáles son las tres partes de una historia de usuario?

1.2 ¿Quién forma parte del equipo del cliente?

1.3 ¿Cuáles de las siguientes no son buenas historias? ¿Por qué?

a El usuario puede ejecutar el sistema en Windows XP y Linux.

b Todos los gráficos y tablas se realizarán utilizando una biblioteca de terceros.

c El usuario puede deshacer hasta cincuenta comandos.

El software se lanzará el 30 de junio.

e El software estará escrito en Java.

f El usuario puede seleccionar su país de una lista desplegable.

g El sistema utilizará Log4J para registrar todos los mensajes de error en un archivo.

h Se le solicitará al usuario que guarde su trabajo si no lo ha guardado durante 15 minutos.

i El usuario puede seleccionar la función “Exportar a XML”.

j El usuario puede exportar datos a XML.

1.4 ¿Qué ventajas tienen las conversaciones de requisitos sobre los documentos de requisitos?

1.5 ¿Por qué querrías escribir pruebas en el reverso de una tarjeta de historia?

Capítulo 2

Escribiendo historias

En este capítulo nos centramos en la escritura de historias. Para crear buenas historias, nos centramos en seis atributos. Una buena historia es:

- Independiente
- Negociable
- Valioso para los usuarios o clientes
- Estimable
- Pequeño
- Comprobable

Bill Wake, autor de Extreme Programming Explored and Refactoring Workbook, ha sugerido el acrónimo INVEST para estos seis atributos (Wake 2003a).

Independiente

En la medida de lo posible, se debe tener cuidado de no introducir dependencias entre historias. Estas dependencias generan problemas de priorización y planificación. Por ejemplo, supongamos que el cliente ha seleccionado como alta prioridad una historia que depende de otra de baja prioridad. Las dependencias entre historias también pueden dificultar la estimación considerablemente. Por ejemplo, supongamos que trabajamos en el sitio web de BigMoneyJobs y necesitamos escribir historias sobre cómo las empresas pueden financiar las vacantes que publican en nuestro sitio. Podríamos escribir estas historias:

1. Una empresa puede pagar una oferta de empleo con una tarjeta Visa.
2. Una empresa puede pagar una oferta de empleo con una MasterCard.

3. Una empresa puede pagar una oferta de empleo con una tarjeta American Express.

Supongamos que los desarrolladores estiman que se necesitarán tres días para admitir el primer tipo de tarjeta de crédito (independientemente de cuál sea) y un día para cada uno de los dos. Con historias tan dependientes como estas, no se sabe qué estimación asignar a cada historia: ¿a cuál se le debe asignar la estimación de tres días?

Cuando se presenta este tipo de dependencia, hay dos formas de evitarlo:

• Combine las historias dependientes en una historia más grande pero independiente

• Encuentra una forma diferente de dividir las historias.

Combinar las historias sobre los diferentes tipos de tarjetas de crédito en una sola historia extensa ("Una empresa puede pagar una oferta de trabajo con tarjeta de crédito") funciona bien en este caso, ya que la historia combinada solo dura cinco días. Si la historia combinada es mucho más larga, un mejor enfoque suele ser encontrar una dimensión diferente para dividir las historias. Si las estimaciones para estas historias hubieran sido más largas, una división alternativa sería:

1. Un cliente puede pagar con un tipo de tarjeta de crédito.

2. Un cliente puede pagar con dos tipos adicionales de tarjetas de crédito.

Si no quieres combinar las historias y no encuentras una buena forma de dividirlas, siempre puedes adoptar el enfoque simple de poner dos estimaciones en la tarjeta: una estimación si la historia se termina antes que la otra historia, y una estimación más baja si se termina después.

Negociable

Las historias son negociables. No son contratos escritos ni requisitos que el software deba implementar. Las tarjetas de historia son descripciones breves de la funcionalidad, cuyos detalles se negociarán en una conversación entre el cliente y el equipo de desarrollo. Dado que las tarjetas de historia son recordatorios para una conversación, más que requisitos completamente detallados, no es necesario que incluyan todos los detalles relevantes. Sin embargo, si al escribir la historia se conocen algunos detalles importantes, deben incluirse como anotaciones en la tarjeta de historia, como se muestra en la Tarjeta de Historia 2.1. El reto reside en aprender a incluir los detalles justos.

La Tarjeta de Historia 2.1 funciona bien porque proporciona la cantidad adecuada de información al desarrollador y al cliente que hablarán sobre la historia. Cuando un desarrollador...

> Una empresa puede pagar una oferta de empleo con tarjeta de crédito.
>
> Nota: Aceptamos Visa, MasterCard y American Express.
> Considere Discover.

■ Tarjeta de historia 2.1 Una tarjeta de historia con notas que brindan detalles adicionales.

Cuando el operador comience a codificar esta historia, se le recordará que ya se ha decidido aceptar las tres tarjetas principales y podrá preguntar al cliente si se ha decidido aceptar las tarjetas Discover. Las notas en la tarjeta ayudan al desarrollador y al cliente a reanudar la conversación donde la dejaron. Idealmente, la conversación se puede reanudar así de fácil, independientemente de si son el mismo desarrollador y el mismo cliente quienes la reanudan.

Utilice esto como guía al agregar detalles a las historias.

Por otro lado, considere una historia con demasiadas notas, como se muestra en la Tarjeta de Historia 2.2. Esta historia tiene demasiados detalles ("Recopilar el mes y la fecha de vencimiento de la tarjeta") y, además, combina lo que probablemente debería ser una historia independiente ("El sistema puede almacenar un número de tarjeta para uso futuro").

> Una empresa puede pagar una oferta de empleo con tarjeta de crédito.
> Nota: Aceptamos Visa, MasterCard y American Express.
> Considere Discover. Para compras superiores a $100, solicite el número de identificación de la tarjeta que se encuentra en el reverso. El sistema puede determinar el tipo de tarjeta con solo leer los dos primeros dígitos. El sistema puede almacenar el número de tarjeta para uso futuro. Registre el mes y la fecha de vencimiento de la tarjeta.

■ Tarjeta de historia 2.2 Una tarjeta de historia con demasiados detalles.

Trabajar con historias como la Tarjeta de Historia 2.2 es muy difícil. La mayoría de los lectores de este tipo de historias asocian erróneamente el detalle adicional con la precisión adicional.
Sin embargo, en muchos casos, especificar detalles demasiado pronto solo genera más trabajo. Por ejemplo, si dos desarrolladores discuten y estiman una historia que simplemente dice "una empresa puede pagar una oferta de trabajo con tarjeta de crédito", no olvidarán que su discusión es algo abstracta. Faltan demasiados detalles para...

Esto puede llevar a que consideren erróneamente su conversación como definitiva o su estimación como precisa. Sin embargo, al añadir tantos detalles como en la Tarjeta de Historia 2.2, las conversaciones sobre la historia tienden a parecer mucho más concretas y reales. Esto puede llevar a la creencia errónea de que las tarjetas de historia reflejan todos los detalles y que no es necesario seguir discutiendo la historia con el cliente.

Si pensamos en la tarjeta de historia como un recordatorio para que el desarrollador y el cliente tengan una conversación, entonces es útil pensar en la tarjeta de historia como si contuviera:

- una frase o dos que actúen como recordatorios para mantener la conversación
- notas sobre cuestiones que se resolverán durante la conversación

Los detalles ya determinados mediante conversaciones se convierten en pruebas. Las pruebas se pueden anotar en el reverso de la tarjeta de historia si se utilizan tarjetas de notas o en cualquier sistema electrónico que se utilice. Las Tarjetas de Historia 2.3 y 2.4 muestran cómo el exceso de detalle de la Tarjeta de Historia 2.2 puede convertirse en pruebas, dejando solo las notas de la conversación como parte del anverso de la tarjeta. De esta manera, el anverso de una tarjeta de historia contiene la historia y notas sobre preguntas abiertas, mientras que el reverso contiene detalles sobre la historia en forma de pruebas que comprobarán si funciona como se esperaba.

> Una empresa puede pagar una oferta de empleo con tarjeta de crédito.
>
> Nota: ¿Aceptaremos tarjetas Discover?
>
> Nota para la interfaz de usuario: no tiene un campo para el tipo de tarjeta (se puede derivar de los primeros dos dígitos de la tarjeta).

■ Tarjeta de historia 2.3 El frente revisado de una tarjeta de historia con solo la historia y las preguntas para ser discutido.

Valioso para los compradores o usuarios

Es tentador decir algo como "Cada historia debe ser valorada por los usuarios". Pero sería un error. Muchos proyectos incluyen historias que no son valoradas por los usuarios. Teniendo en cuenta la distinción entre usuario (quien usa el software) y comprador (quien compra el software), supongamos que un equipo de desarrollo está desarrollando software que se implementará en un...

Prueba con Visa, MasterCard y American Express (aprueba).

Prueba con Diner's Club (falla).

Prueba con números de identificación de tarjetas buenos, malos y faltantes.

Prueba con tarjetas caducadas.

Prueba con más de $100 y menos de $100.

■ Tarjeta de historia 2.4 Los detalles que implican casos de prueba están separados de la historia en sí. Aquí se muestran en el reverso de la tarjeta de la historia.

Gran base de usuarios, quizás 5000 ordenadores en una sola empresa. El comprador de un producto como ese podría estar muy preocupado de que cada uno de los 5000 ordenadores utilice la misma configuración de software. Esto podría dar lugar a una situación como "Toda la información de configuración se lee desde una ubicación central". A los usuarios no les importa dónde se almacena la información de configuración, pero a los compradores sí.

De manera similar, historias como la siguiente podrían ser valoradas por los compradores contemporáneos. chapado comprando el producto pero que no sería valorado por los usuarios reales:

- Durante todo el proceso de desarrollo, el equipo de desarrollo producirá documentación adecuada para una auditoría ISO 9001.

- El equipo de desarrollo producirá el software de acuerdo con CMM Nivel 3.

Lo que quieres evitar son historias que solo valoran los desarrolladores. Para Por ejemplo, evite historias como estas:

- Todas las conexiones a la base de datos se realizan a través de un pool de conexiones.

- Todo el manejo y registro de errores se realiza a través de un conjunto de clases comunes.

Tal como están escritas, estas historias se centran en la tecnología y las ventajas para los programadores. Es muy posible que las ideas subyacentes sean buenas, pero deberían redactarse de forma que los beneficios para los clientes o usuarios sean evidentes. Esto permitirá al cliente priorizar inteligentemente estas historias en el cronograma de desarrollo. Mejores variaciones de estas historias podrían ser las siguientes:

- Hasta cincuenta usuarios deberían poder utilizar la aplicación con una licencia de base de datos de cinco usuarios.

- Todos los errores se presentan al usuario y se registran de manera consistente.

De la misma manera que conviene evitar las suposiciones sobre la interfaz de usuario en las historias, también conviene evitar las suposiciones tecnológicas. Por ejemplo, en las historias revisadas anteriores se ha eliminado el uso implícito de un pool de conexiones y un conjunto de clases de gestión de errores.

La mejor manera de garantizar que cada historia sea valiosa para el cliente o los usuarios es que el cliente la escriba. Los clientes a menudo se sienten incómodos con

Esto al principio, probablemente porque los desarrolladores los han entrenado para pensar que todo lo que escriben puede serles reprochable más adelante. ("Bueno, el documento de requisitos no decía eso..."). La mayoría de los clientes empiezan a escribir historias ellos mismos una vez que se familiarizan con el concepto de que las tarjetas de historias son recordatorios para hablar más tarde, en lugar de compromisos formales o descripciones de una funcionalidad específica.

Estimable

Es importante que los desarrolladores puedan estimar (o al menos estimar) el tamaño de una historia o el tiempo que tomará convertirla en código funcional. Hay tres razones comunes por las que una historia puede no ser estimable:

1. Los desarrolladores carecen de conocimiento del dominio.

2. Los desarrolladores carecen de conocimientos técnicos.

3. La historia es demasiado grande.

En primer lugar, los desarrolladores pueden carecer de conocimientos del dominio. Si no comprenden una historia tal como está escrita, deberían comentarla con el cliente que la escribió. No es necesario comprender todos los detalles de una historia, pero sí es necesario que los desarrolladores tengan una comprensión general de ella.

En segundo lugar, una historia puede no ser estimable porque los desarrolladores no comprenden la tecnología involucrada. Por ejemplo, en un proyecto Java se nos pidió que proporcionáramos una interfaz CORBA al sistema. Ningún miembro del equipo lo había hecho, así que no había forma de estimar la tarea. La solución en este caso es enviar a uno o más desarrolladores a lo que la Programación Extrema denomina un pico, que consiste en un breve experimento para aprender sobre un área de la aplicación. Durante el pico, los desarrolladores aprenden lo suficiente como para poder estimar la tarea. Al pico en sí siempre se le asigna un tiempo máximo definido (llamado time-box), lo que nos permite estimarlo. De esta manera, una historia no estimable se convierte en dos historias: un pico rápido para recopilar información y luego una historia para realizar el trabajo real.

Finalmente, es posible que los desarrolladores no puedan estimar una historia si es demasiado extensa. Por ejemplo, para el sitio web BigMoneyJobs, la historia "Un solicitante de empleo puede encontrar trabajo" es demasiado extensa. Para estimarla, los desarrolladores deberán desagregarla en historias más pequeñas y constituyentes.

Falta de conocimiento del dominio.

Como ejemplo de la necesidad de un mayor conocimiento del dominio, estábamos desarrollando un sitio web para la atención médica a largo plazo de enfermedades crónicas. El cliente (una enfermera altamente cualificada) escribió una historia que decía: "A los nuevos usuarios se les realiza una prueba de detección de diabetes". Los desarrolladores no estaban seguros de qué significaba eso y podría haber abarcado desde un simple cuestionario web hasta el envío de un formulario a los nuevos usuarios para una prueba física en casa, como se hizo con el producto de la empresa para pacientes con asma. Los desarrolladores se reunieron con el cliente y descubrieron que estaba pensando en un formulario web sencillo con unas pocas preguntas.

Aunque son demasiado extensas para estimarlas con fiabilidad, a veces resulta útil escribir epopeyas como «Un solicitante de empleo puede encontrar trabajo», ya que sirven como marcadores o recordatorios sobre partes importantes de un sistema que deben analizarse. Si decide omitir temporalmente partes extensas de un sistema, considere escribir una o dos epopeyas que las cubran. A la epopeya se le puede asignar una estimación amplia y aproximada.

Pequeño

Como Ricitos de Oro en busca de una cama cómoda, algunas historias pueden ser demasiado largas, otras demasiado pequeñas y otras pueden ser perfectas. El tamaño de una historia sí importa, porque si son demasiado largas o demasiado pequeñas, no se pueden usar para planificar. Las epopeyas son difíciles de manejar porque suelen contener varias historias. Por ejemplo, en un sistema de reservas de viajes, «Un usuario puede planificar unas vacaciones» es una epopeya. Planificar unas vacaciones es una funcionalidad importante para un sistema de reservas de viajes, pero implica muchas tareas. La epopeya debe dividirse en historias más pequeñas. La determinación final de si una historia tiene el tamaño adecuado se basa en el equipo, sus capacidades y las tecnologías utilizadas.

Historias divididas

Las epopeyas suelen caer en una de dos categorías:

- La historia compuesta
- La compleja historia

Una historia compuesta es una epopeya que comprende varias historias más cortas. Por ejemplo, el sistema BigMoneyJobs podría incluir la historia "Un usuario puede publicar su currículum". Durante la planificación inicial del sistema, esta historia podría ser apropiada. Pero cuando los desarrolladores hablan con el cliente, descubren que "publicar su currículum" en realidad significa:

- que un currículum puede incluir educación, trabajos anteriores, historial salarial, publicaciones, ciones, presentaciones, servicio comunitario y un objetivo
- que los usuarios puedan marcar currículums como inactivos
- que los usuarios puedan tener varios currículums
- que los usuarios puedan editar currículums
- que los usuarios puedan eliminar currículums

Dependiendo de cuánto tiempo tome desarrollarlos, cada uno podría convertirse en una historia única. Sin embargo, esto podría tomar una epopeya y llevarla demasiado lejos en la dirección opuesta, convirtiéndola en una serie de historias demasiado pequeñas. Por ejemplo, dependiendo de las tecnologías utilizadas y del tamaño y la habilidad del equipo, historias como estas generalmente serán demasiado pequeñas:

- Un solicitante de empleo puede ingresar una fecha para cada ingreso al servicio comunitario en un reanudar.
- Un solicitante de empleo puede editar la fecha de cada entrada de servicio comunitario en un reanudar.
- Un solicitante de empleo puede ingresar un rango de fechas para cada trabajo anterior en un currículum.
- Un solicitante de empleo puede editar el rango de fechas de cada trabajo anterior en un currículum.

En general, una mejor solución es agrupar las historias más pequeñas de la siguiente manera:

- Un usuario puede crear currículums que incluyan educación, trabajos anteriores, historial salarial, publicaciones, presentaciones, servicio comunitario y un objetivo.
- Un usuario puede editar un currículum.
- Un usuario puede eliminar un currículum.

• Un usuario puede tener varios currículums.

• Un usuario puede activar y desactivar currículums.

Normalmente existen muchas maneras de desagregar una historia compuesta. La desagregación previa se basa en crear, editar y eliminar, que es la más común. Esto funciona bien si la historia creada es lo suficientemente pequeña como para que pueda mantenerse como una sola historia. Una alternativa es desagregar según los límites de los datos. Para ello, considere cada componente de un currículum como si se añadiera y editara individualmente. Esto da lugar a una desagregación completamente diferente:

• Un usuario puede agregar y editar información educativa.

• Un usuario puede agregar y editar información del historial de trabajo.

• Un usuario puede agregar y editar información del historial salarial.

• Un usuario puede agregar y editar publicaciones.

• Un usuario puede agregar y editar presentaciones.

• Un usuario puede agregar y editar el servicio comunitario.

• Un usuario puede agregar y editar un objetivo.

Etcétera.

A diferencia de la historia compuesta, la historia compleja es una historia de usuario inherentemente extensa y difícil de desagregar en un conjunto de historias constituyentes. Si una historia es compleja debido a la incertidumbre asociada, se puede dividir en dos: una de investigación y otra de desarrollo de la nueva función. Por ejemplo, supongamos que a los desarrolladores se les da la historia "Una empresa puede pagar una oferta de trabajo con tarjeta de crédito", pero ninguno de ellos ha procesado tarjetas de crédito anteriormente. Pueden optar por dividir las historias de la siguiente manera:

• Investigar el procesamiento de tarjetas de crédito a través de la web.

• Un usuario puede pagar con tarjeta de crédito.

En este caso, la primera historia enviará a uno o más desarrolladores a un pico de aprendizaje. Cuando las historias complejas se dividen de esta manera, siempre se debe definir un límite de tiempo para la historia de investigación, o pico de aprendizaje. Incluso si la historia no se puede estimar con una precisión razonable, es posible definir el tiempo máximo que se dedicará al aprendizaje.

Las historias complejas también son comunes al desarrollar algoritmos nuevos o ampliarlos. Un equipo de una empresa de biotecnología tenía una historia para añadir extensiones novedosas.

a un enfoque estadístico estándar llamado maximización de expectativas. La compleja historia se reescribió en dos: la primera para investigar y determinar la viabilidad de extender la maximización de expectativas; la segunda para añadir esa funcionalidad al producto. En situaciones como esta, es difícil estimar la duración de la investigación.

Considere colocar el pico en una iteración diferente

Siempre que sea posible, conviene incluir la historia de investigación en una iteración y las demás historias en una o más iteraciones posteriores. Normalmente, solo se puede estimar la historia de investigación. Incluir las demás historias no estimables en la misma iteración implica un nivel de incertidumbre mayor de lo normal sobre cuánto se puede lograr en esa iteración.

La principal ventaja de desglosar una historia que no se puede estimar es que permite al cliente priorizar la investigación por separado de la nueva funcionalidad. Si el cliente solo tiene que priorizar la historia compleja ("Añadir nuevas extensiones a la maximización de expectativas estándar") y una estimación para la historia, podría priorizarla basándose en la suposición errónea de que la nueva funcionalidad se entregará aproximadamente en ese plazo. Si, en cambio, el cliente tiene una historia de investigación y de pico ("Investigar y determinar la viabilidad de ampliar la maximización de expectativas") y una historia funcional ("Extender la maximización de expectativas"), debe elegir entre añadir la historia de investigación que no añade ninguna funcionalidad nueva en esta iteración y quizás otra historia que sí la aporte.

Combinando historias

A veces, las historias son demasiado pequeñas. Una historia demasiado pequeña suele ser aquella que el desarrollador no quiere escribir ni estimar, ya que hacerlo podría llevar más tiempo que realizar el cambio. Los informes de errores y los cambios en la interfaz de usuario son ejemplos comunes de historias que suelen ser demasiado pequeñas. Un buen enfoque para historias pequeñas, común en los equipos de Programación Extrema, es combinarlas en historias más grandes que representan desde medio día hasta varios días de trabajo. A la historia combinada se le asigna un nombre y luego se programa y se trabaja en ella como cualquier otra historia.

Por ejemplo, supongamos que un proyecto tiene cinco errores y una solicitud para cambiar algunos colores en la pantalla de búsqueda. Los desarrolladores estiman el trabajo total requerido.

Y toda la colección se trata como una sola historia. Si has optado por usar fichas de papel, puedes hacerlo grapándolas con una cartulina.

Comprobable

Las historias deben escribirse de forma que sean testables. Superar las pruebas demuestra que se han desarrollado correctamente. Si la historia no se puede probar, ¿cómo pueden los desarrolladores saber cuándo han terminado de codificar?

Las historias no comprobables suelen aparecer en el caso de requisitos no funcionales, que son requisitos sobre el software, pero no directamente sobre su funcionalidad. Por ejemplo, considere estas historias no funcionales:

- El usuario debe encontrar el software fácil de utilizar.
- Un usuario nunca debe tener que esperar mucho tiempo para que aparezca una pantalla.

Tal como están escritas, estas historias no son testeables. Siempre que sea posible, las pruebas deben automatizarse. Esto significa buscar una automatización del 99 %, no del 10 %. Casi siempre se puede automatizar más de lo que se cree. Cuando un producto se desarrolla de forma incremental, las cosas pueden cambiar muy rápidamente y el código que funcionaba ayer puede dejar de funcionar hoy. Se necesitan pruebas automatizadas que detecten esto lo antes posible.

Existe un subconjunto muy pequeño de pruebas que no se pueden automatizar de manera realista. Por ejemplo, una historia de usuario que dice "Un usuario novato puede completar flujos de trabajo comunes sin capacitación" se puede probar, pero no se puede automatizar. Para probar esta historia, probablemente será necesario que un experto en factores humanos diseñe una prueba que incluya la observación de una muestra aleatoria de usuarios novatos representativos. Este tipo de prueba puede ser larga y costosa, pero la historia es comprobable y podría ser adecuada para algunos productos.

La idea de que «un usuario nunca tiene que esperar mucho para que aparezca una pantalla» no es comprobable porque dice «nunca» y no define qué significa «esperar mucho». Demostrar que algo nunca sucede es imposible. Un objetivo mucho más sencillo y razonable es demostrar que algo rara vez sucede.

Esta historia podría haberse escrito como "Las nuevas pantallas aparecen en dos segundos en el 95% de los casos". Y, mejor aún, se puede escribir una prueba automatizada.

diez para comprobarlo.

Resumen

Idealmente, las historias son independientes entre sí. Esto no siempre es posible, pero en la medida en que lo sea, las historias deben escribirse de forma que puedan desarrollarse en cualquier orden.

- Los detalles de una historia se negocian entre el usuario y los desarrolladores.

Las historias deben redactarse de forma que su valor para los usuarios o el cliente quede claro. La mejor manera de lograrlo es que el cliente las escriba.

- Las historias pueden estar anotadas con detalles, pero demasiados detalles oscurecen el significado de la historia y pueden dar la impresión de que no es necesaria ninguna conversación entre los desarrolladores y el cliente.
- Una de las mejores maneras de anotar una historia es escribir casos de prueba para la historia.
- Si son demasiado grandes, las historias compuestas y complejas pueden dividirse en varias Por ejemplo, historias más pequeñas.
- Si son demasiado pequeñas, se pueden combinar varias historias pequeñas en una grande. historia de ger
- Las historias deben poder probarse.

Responsabilidades del desarrollador

- Usted es responsable de ayudar al cliente a escribir historias que sean promesas de conversación en lugar de especificaciones detalladas, que tengan valor para los usuarios o el cliente, que sean independientes, que se puedan probar y que tengan el tamaño adecuado.
- Si siente la tentación de pedir una historia sobre el uso de una tecnología o una pieza de infraestructura, usted es responsable de describir la necesidad en términos de su valor para los usuarios o el cliente.

Responsabilidades del cliente

- Usted es responsable de escribir historias que sean promesas de conversación en lugar de especificaciones detalladas, que tengan valor para los usuarios o para usted mismo, que sean independientes, que se puedan probar y que tengan el tamaño adecuado.

Preguntas

2.1 Para las siguientes historias, indique si son buenas o no. Si no, ¿por qué?

a Un usuario puede dominar rápidamente el sistema.

b Un usuario puede editar la dirección en un currículum.

c Un usuario puede agregar, editar y eliminar varios currículums.

El sistema puede calcular aproximaciones de puntos de silla para distribuciones de formas cuadráticas en variables normales.

e Todos los errores de tiempo de ejecución se registran de manera consistente.

2.2 Divida esta epopeya en historias de componentes de tamaño apropiado: “Un usuario puede crear y cambiar agentes de búsqueda de empleo automatizados”.

Esta página se dejó en blanco intencionalmente

Capítulo 3

Modelado de roles de usuario

En muchos proyectos, las historias se escriben como si solo hubiera un tipo de usuario. Todas las historias se escriben desde la perspectiva de ese tipo de usuario. Esta simplificación es una falacia y puede llevar a un equipo a pasar por alto historias para usuarios que no se ajustan al molde general del tipo de usuario principal del sistema. Las disciplinas del diseño centrado en el uso (Constantine y Lockwood, 1999) y el diseño de interacción (Cooper, 1999) nos enseñan los beneficios de identificar los roles y personajes de los usuarios antes de escribir historias. En este capítulo, analizaremos los roles de usuario, el modelado de roles, los mapas de roles de usuario y los personajes, y mostraremos cómo estos pasos iniciales conducen a mejores historias y un mejor software.

Roles de usuario1

Supongamos que estamos creando el sitio de búsqueda y publicación de empleos BigMoneyJobs. Este tipo de sitio tendrá muchos tipos de usuarios diferentes. Cuando hablamos de historias de usuario , ¿de quién hablamos? ¿Hablamos de Ashish, que tiene un trabajo pero siempre está buscando uno mejor? ¿Hablamos de Laura, una recién graduada que busca su primer trabajo profesional? ¿Hablamos de Allan, que ha decidido aceptar cualquier trabajo que le permita mudarse a Maui y practicar windsurf todas las tardes? ¿O hablamos de Scott, que no odia su trabajo, pero se ha dado cuenta de que es hora de cambiar de trabajo? Quizás estemos hablando de Kindra, que fue despedida hace seis meses y buscaba un buen trabajo, pero ahora acepta cualquier cosa en el noreste de Estados Unidos.

¿O deberíamos pensar que el usuario proviene de una de las empresas que publican las ofertas de empleo? Quizás sea Mario, que trabaja en recursos humanos y publica...

1. Gran parte del análisis de los roles de usuario en este capítulo se basa en el trabajo de Larry Constantine y Lucy Lockwood. Puede encontrar más información sobre el modelado de roles de usuario en su sitio web www.foruse.com o en Software for Use (1999).