

Gestión de Proyectos Software

Scrum – Requisitos e historias de usuario



Contenidos

- Requisitos
- Historias de usuario
 - Las tres ces
 - Nivel de detalle
 - Los criterios INVEST
- Requisitos no funcionales y adquisición de conocimiento
- Conseguir historias



Requisitos



Requisitos

- En desarrollo tradicional, los requisitos se detallan al principio y se intenta que sean inmutables
- En Scrum se negocian continuamente durante el desarrollo y se entregan al equipo de desarrollo justo a tiempo y justo con lo necesario para que trabajen
- En desarrollo tradicional, un cambio de requisitos puede implicar un proceso formal de cambio
 - Por tanto se considera no deseable y caro
 - Sin embargo, es común que un cliente no se de cuenta de que quiere algo hasta bien avanzado el desarrollo
- En Scrum los requisitos son un grado de libertad que se puede manipular para alcanzar los objetivos de negocios
 - Si creas algo innovador, no hay manera de que puedas pensar en todos los requisitos por adelantado



Requisitos en Scrum

- En lugar de crear requisitos detallados al principio, se crean entradas (*items*) de la pila del producto (PBI – *product backlog items*)
 - Son huecos (*placeholders*) que más tarde rellenaremos con los requisitos detallados
 - Cada uno representa algo valioso y deseable
- Los PBI al principio son grandes y poco detallados
 - Con el tiempo se refinarán en PBI más detallados
 - Tarde o temprano serán lo bastante pequeños y detallados como para ser movidos a un sprint, donde se diseñarán, implementarán y probarán. Durante el sprint se detallarán aún más
- Aunque no es “obligatorio” en Scrum, los PBI se representan a menudo como historias de usuario (*user stories*)
 - Algunos equipos prefieren casos de uso u otras técnicas



Conversaciones

- Los requisitos facilitan una comprensión compartida de lo que hay que crear
 - El que entiende lo que quiere se lo comunica al que lo tiene que crear
- El desarrollo tradicional se basa en requisitos escritos y con vocación de permanencia
- Scrum se apoya más en conversaciones verbales frecuentes (comunicación bidireccional, *feedback* rápido, ancho de banda grande)
 - Esto no reemplaza a todos los documentos. La pila del producto es un “documento vivo”



Refinado progresivo

- En el desarrollo tradicional todos los requisitos están al mismo nivel de detalle al mismo tiempo
 - Hay que predecir todos los detalles al principio, que es cuando sabemos menos
 - Todos los requisitos se tratan igual, sin considerar su prioridad
 - Se crea un gran inventario de requisitos, que será caro de modificar o descartar (por el tiempo invertido en crearlo)
 - Se reduce la posibilidad de tener conversaciones para clarificar cosas porque los requisitos ya están “completos”
- En Scrum, los requisitos en los que se va a trabajar antes serán más pequeños y detallados que aquellos que no se abordarán en un tiempo
- El refinado progresivo desagrega grandes requisitos poco detallados en pequeños requisitos más detallados conforme hacen falta (*just-in-time*)



Historias de usuario



Historias de usuario

- Son un formato para expresar el valor deseado para muchos tipos de entradas en la pila del producto, especialmente características
 - Entendibles desde el punto de vista de negocios y técnico
- Estructuralmente simples, proporcionan **un punto de partida** para la conversación
- Se pueden escribir a distintos niveles de detalle y son fáciles de refinar progresivamente
- No son la única forma de representar entradas en la pila del producto
 - “Como usuario, me gustaría que el sistema no corrompiera la base de datos” no parece la forma más adecuada de describir eso
- Son una aproximación ligera que encaja bien con los principios ágiles



Las tres ces: *card,* *conversation,* *confirmation*



Tarjeta (*Card*)

- Muchas historias de usuario se escriben en tarjetas o en *post-its*
 - Pequeñas, para forzar la brevedad
- Capturan la esencia de un requisito y permiten que luego se desarrollen discusiones más detalladas
- Una plantilla común:
 - Como <clase de usuario>
 - Quiero <un objetivo>
 - Para <un beneficio>



Tarjeta - Ejemplo

“Como usuario típico, quiero ver críticas no sesgadas de restaurantes cercanos a una dirección, para poder decidir donde ir a cenar”



Conversación

- Los detalles de un requisito se sacan a la luz y se comunican en una conversación entre el equipo de desarrollo, el dueño del producto y los clientes, inversores etc.
 - La historia de usuario es una promesa de que se tendrá esa conversación
- Esa “conversación” no es un evento que sucede una vez. Es un diálogo **permanente**
 - Cuando se escribe la historia, cuando se refina, cuando se estima, cuando se planifica el sprint, y cuando se diseña, construye y prueba durante el sprint
- En buena parte son verbales, pero se suelen complementar con documentos
 - Pueden derivar en un borrador de interfaz de usuario, o una elaboración de ciertas reglas de negocio que se escriben, o referenciar algún documento (p.ej. un artículo) para consultar más adelante



Confirmación

- Una historia de usuario contiene unas condiciones de satisfacción
 - Criterios de aceptación que clarifican el comportamiento deseado
- Se pueden escribir por detrás de la tarjeta
- Pueden ser tests de aceptación de alto nivel
 - Que serán solo una mínima parte de los tests que el equipo de desarrollo tendrá para esa historia



Confirmación - Ejemplo

- Como usuario de la wiki, quiero subir ficheros para poder compartirlos con mis colegas
- Condiciones de satisfacción
 - Verificar con ficheros .txt y .doc
 - Verificar con ficheros .jpg, .gif y .png
 - Verificar con ficheros .mp4 de menos de 1GB
 - Verificar que no se pueden subir ficheros con DRM



Nivel de detalle de las historias



Nivel de detalle

- Es difícil planificar a alto nivel con historias de usuario del tamaño que se puede abordar en un sprint
 - Por fortuna podemos escribir historias de usuario con distintos niveles de abstracción
- Épicas (*epics*): historias de muchos meses, que incluyen uno o varios lanzamientos de producto
 - Visión global
- Características (*features*): demasiado grandes para un solo sprint
- Historias implementables (*implementable stories*): caben en un sprint
 - A veces se llaman *sprintable stories*



Nivel de detalle

- Ejemplo de épica: Como usuario típico, quiero entrenar el sistema sobre los tipos de productos que prefiero para que sepa como filtrar mejor mis búsquedas



Nivel de detalle

- A veces se usa el término tema (*theme*) para referirse a una colección de historias relacionadas
 - Son como la tarjeta resumen de un puñado de tarjetas que hemos atado juntas por estar relacionadas
- Las tareas (*tasks*) son el nivel que hay por debajo de las historias
 - No son historias
 - Típicamente las lleva a cabo una persona, o máximo dos
 - Típicamente cuestan horas
 - Especifican como construir algo y no el qué construir (para eso están las historias)



Escribir buenas historias: los criterios INVEST



Los criterios INVEST

- Los criterios INVEST se usan para evaluar si una historia de usuario cumple su propósito
 - *Independent*
 - *Negotiable*
 - *Valuable*
 - *Estimatable*
 - *Small* (del tamaño adecuado)
 - *Testable*



Independiente

- Las historias de usuario deberían ser independientes o, al menos, poco acopladas entre si
 - La interdependencia complica la estimación, la priorización y la planificación: p.ej. antes de trabajar en una historia, hay que desarrollar aquellas de las que depende...



¿Independientes?

- Como programador, quiero poder ver simultáneamente varias partes de mi código en el editor para poder analizar mejor sus interdependencias
- Como programador, quiero poder dividir el editor en *frames* para poder tener diferentes vistas simultáneas del código de la aplicación que desarrollo



Negociable

- Los detalles deberían ser negociables
- Las historias no son contratos; son formas de recordar que hay que tener las conversaciones donde se negociarán los detalles
 - Capturan la esencia de la funcionalidad deseada y por qué se desea. Dejan sitio para negociar los detalles



Valiosa

- Para algún cliente, usuario o ambos
 - Los clientes eligen el producto y pagan por él
 - Los usuarios lo usan
- Una historia que no es valiosa para ninguno no tiene sitio en la pila
 - Corolario: todas las historias de la pila deben ser valiosas



¿Valiosa?

- Como desarrollador, quiero migrar el sistema a la última versión de PostgreSQL para evitar seguir trabajando con una versión que ya no se mantiene
- Notad que es una “historia técnica”



Estimable

- Las historias deben ser estimables por el equipo
- Las estimaciones indican tamaño => coste y esfuerzo
- El tamaño de una historia es necesario
 - El dueño del producto lo necesita para priorizar
 - El equipo Scrum lo necesita para saber si la historia tiene que ser desagregada antes de ponerse con ella
- Si el equipo no es capaz de estimar una historia, o es demasiado grande o es demasiado ambigua
 - Habrá que partirla en historias más manejables
 - Si faltan conocimientos, habrá que explorar

Pequeña (del tamaño adecuado)

- El tamaño adecuado depende de cuando pensamos ponernos con ella
 - Si son para un sprint, pequeñas
 - En un sprint queremos trabajar en varias historias (con solo una hay demasiado riesgo de no terminarla)
- Una historia puede ser grande si planeamos trabajar en ella durante un año
 - Lo que sería una pérdida de tiempo sería ponernos a detallarla ahora mismo



Testeable

- De forma binaria: o pasa o no pasa
- Requiere buenos criterios de aceptación
- Única forma de saber con certeza si la historia se ha hecho o no
- Los tests ayudan a estimar el tamaño de una historia (suelen aportar detalles)



Testeable

- Algunas historias no se pueden testear
 - Una épica no tendrá tests asociados; no pasa nada, no la vamos a construir directamente
- Otras no podrán testearse de forma práctica
 - 99,99% de *uptime* en producción no se puede testear, es algo que hay que ir midiendo constantemente
 - Aún así, este criterio de aceptación es valioso porque guiará el diseño de la aplicación



¿Testeable?

- Como programador, quiero que los proyectos se compilen muy rápido para agilizar mi trabajo
 - Criterios de aceptación: un proyecto de 1000 LOC de Java debe compilarse en menos de 2 segundos
- ¿Es suficiente?



¿Testeable?

- Como programador, quiero que el buscador de código me haga sugerencias basadas en mis búsquedas previas en ese proyecto para agilizar mi trabajo
- ¿Qué criterios de aceptación necesitamos para que sea testeable?



Requisitos no funcionales y adquisición de conocimiento



Requisitos no funcionales

- Se pueden escribir como historias si ese formato nos resulta conveniente, pero no hace falta
- No van como entradas de la pila del producto porque son requisitos “globales”
 - Afectan al diseño y prueba de muchas (o todas) de las historias en la pila del producto
- Como norma general hay que intentar incluirlas en la definición de hecho del equipo
 - Así se comprueban en cada sprint para cada característica desarrollada



¿Entrada en la pila o definición de hecho?

- Ninguna acción del usuario tardará más de 1 segundo en proporcionarle una realimentación (aunque no sea necesariamente la respuesta que busca)



¿Entrada en la pila o definición de hecho?

- El usuario tendrá a su disposición combinaciones de teclas para poder invocar más rápidamente las funciones más comunes del programa



¿Entrada en la pila o definición de hecho?

- El usuario podrá modificar las combinaciones de teclas que invocan las funciones más comunes del programa



Historias de adquisición de conocimiento

- Necesitamos aprender algo nuevo, así que exploramos
 - Prototipos, pruebas de concepto, experimentos, estudios...
- Es una forma de “comprar” información
 - Llevar a cabo la historia tiene un coste
- Se pueden representar perfectamente como historias de usuario
- Pueden ser entradas en la pila del producto



Historias de adquisición de conocimiento

- Como desarrollador, quiero prototipar dos alternativas para el nuevo motor de filtrado para saber cual es mejor a largo plazo
- Condiciones de satisfacción
 - Hacer tests de velocidad en los 2 prototipos
 - Hacer tests de escalado en los 2 prototipos
 - Escribir un breve informe describiendo los experimentos, los resultados y las recomendaciones



Historias de adquisición de conocimiento

- Hay que estimar el coste (prototipar no es gratis)
- Si va a costarnos un sprint en el que trabajará todo el equipo, ya sabemos los € que serán
- Luego estimamos los € de la información que obtendremos: el coste de tomar una decisión equivocada por no contar con esta información
- El dueño del producto compara, y decide la prioridad de esta historia en la pila



Calcula el coste

- Estimamos que la exploración que necesitamos para elegir entre el componente A y el componente B es de 1000€
- Si elegimos mal, el coste de empezar de nuevo es de 10000€
- ¿Hacemos la exploración?



Calcula el coste

- Estimamos que la exploración que necesitamos para elegir entre el componente A y el componente B es de 1000€
- Si elegimos mal, el coste de empezar de nuevo es de 1500€
- ¿Hacemos la exploración?



Calcula el coste

- Estimamos que la exploración que necesitamos para elegir entre el componente A, el componente B y el componente C es de 1750€
- Si elegimos mal, el coste es de 1500€
 - Asumimos que el peor caso es que elegimos mal, pagamos el coste, volvemos a elegir mal y volvemos a pagar el coste
- ¿Hacemos la exploración?



Conseguir historias



Conseguir historias

- Preguntar a los usuarios es difícil
 - Incluso si saben lo que quieren y lo saben expresar (que es difícil), pueden cambiar luego de opinión
- Es mejor hacerles parte de un equipo que determina qué construir y constantemente revisa lo que se está construyendo



Taller de escritura de historias de usuario

- Pensar colectivamente en el valor de negocio que se desea
 - El equipo Scrum, junto a clientes, inversores...
- Crear historias de usuario para lo que el producto o servicio tiene que hacer
- De unas horas a unos días



Taller de escritura de historias de usuario

- Generalmente con un foco específico
 - P.ej. historias para la próxima versión del producto
- Si es a principio de proyecto es útil pensar roles de usuario
 - Los del “Como <rol de usuario x> ...”
- Y asociar los roles con personas prototipo
 - Con nombre, descripción... Luego los nombres pueden usarse en las historias de usuario
- Las historias pueden pensarse *top-down* (partiendo de una épica y luego desagregando), *bottom-up*, o con mezcla de ambas



Mapeo de historias

- Esta técnica descompone la actividad de usuarios a alto nivel en tareas detallada
- Proporciona una vista bidimensional de las historias, que puede ser un buen complemento a la vista unidimensional de la pila del producto
- Aunque no se use formalmente, la idea de los flujos de trabajo que siguen los usuarios puede ayudar a entender el sistema



Epic
Comprar
Producto

Epic

Secuencia de uso o workflow (en el tiempo) →

Prioridad ↓

Theme
Buscar
Producto

Theme
Gestionar
carro de
compra

Theme

Theme

Theme

story
Buscar
por
nombre

story
Añadir al
carro

story

story

story

story
Buscar
por
color

story
Borrar
del
carro

story

story

story
Buscar
por
autor

story
Calcular
precio
del carro

Epic
Comprar
Producto

Epic

Theme
Buscar
Producto

Theme
Gestionar
carro de
compra

Theme



Theme



Theme

story

Buscar
por
nombre

story

Añadir al
carro



story

story



story



story

Buscar
por
color

story
Borrar
del
carro



story



story

story

Buscar
por
autor

story

Calcular
precio
del carro

Cada tema se descompone en historias implementables ordenadas por deseabilidad (no por prioridad; aún no las hemos estimado). No todas las historias de un tema tienen que ser implementadas en la misma versión

Bibliografía

- Kenneth S. Rubin. *Essential Scrum. A practical guide to the most popular agile process*
 - Chapter 5 (Requirements and User Stories)

