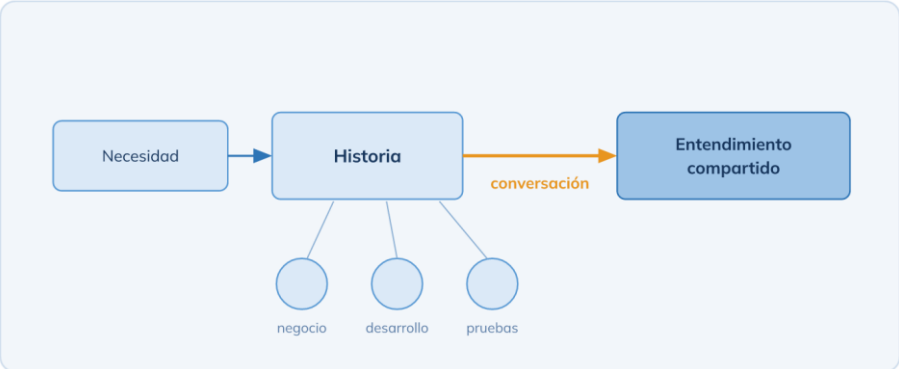
Scrum Manager
Uncovering better ways

Guía

Historias de Usuario

Desarrollo de los temas del State of the art



Actualizado a septiembre de 2026

Scrum Manager® · Skill Arena

Sobre este documento

Este documento desarrolla en profundidad los temas del mapa de referencia (State of the art) para trabajar con historias de usuario y con los demás elementos del Product Backlog, desde escribirlas cuando aportan y conversarlas hasta juzgar su calidad, dividirlas sin perder valor, mantener el contexto del conjunto y verificar que el resultado responde a la intención, también cuando quien redacta o quien implementa es una inteligencia artificial, que está disponible en: Skill Arena.

Úsalo como material de consulta para mantener y contrastar tu práctica con el conocimiento profesional actual.

Se complementa con la plataforma de entrenamiento y evaluación en Skill Arena. En el área Historias de Usuario puedes realizar pruebas de entrenamiento para contrastar y mejorar tu nivel de conocimiento y si lo deseas también puedes obtener un diploma que acredita curricularmente la solvencia y vanguardia profesional en esta área.

Estado del conocimiento

El conocimiento sobre las historias de usuario está asentado en su canon y se mueve muy deprisa en su frontera: las piezas que lo fijan tienen entre veinte y veinticinco años y siguen vigentes sin revisiones mayores, mientras una franja de práctica gana adopción sin ser universal (medir la calidad con criterios explícitos, el mapa de ejemplos, dimensionar por medida empírica, preservar el contexto de las conversaciones) y la capa de inteligencia artificial y agentes cambia en cuestión de meses. En los distintos apartados del documento, las etiquetas (ESTABLECIDO, EN CONSOLIDACIÓN, EMERGENTE) ayudan a identificar la madurez de cada concepto:

ESTABLECIDO consenso asentado; conocimiento que se da por necesario.

EN CONSOLIDACIÓN gana adopción con rapidez; aún no universal pero ya relevante.

EMERGENTE frontera reciente; alta relevancia y alta volatilidad.

Índice

Capítulos del desarrollo, en el orden del State of the art.

BLOQUE A · LA HISTORIA DE USUARIO: QUÉ ES Y QUÉ LUGAR OCUPA

Capítulo 1. La historia como promesa de conversación: origen, las tres C y la plantilla · ESTABLECIDO

De dónde viene la historia de usuario, qué la distingue de un requisito y por qué su valor está en la conversación que promete.

Introducción

Casi todo el mundo que trabaja en producto ha escrito una historia de usuario, y una parte considerable lo ha hecho creyendo que escribía un requisito breve. De ahí vienen la mayoría de los problemas que veremos en esta guía. En este primer capítulo volvemos al origen para recuperar lo que la técnica dice de sí misma, porque casi todos los errores que se cometen con ella consisten en olvidar para qué se inventó.

El recorrido es corto y conviene hacerlo con las fuentes en la mano. La historia nace en la programación extrema a finales de los años noventa, se ordena en 2001 con las tres C de Ron Jeffries y adopta poco después la plantilla que hoy se reconoce en cualquier herramienta. Veremos las tres piezas y, sobre todo, qué papel juega cada una.

1.1 De dónde viene y qué problema resolvía

Las historias de usuario aparecen en el proyecto C3 de Chrysler en 1997 y se describen por primera vez en *Extreme Programming Explained* (Kent Beck, 1999), donde son la unidad con la que cliente y equipo planifican el trabajo. Su función original no era documentar, sino **dar pie a una conversación en el momento oportuno**. Alistair Cockburn, que visitó aquel proyecto, acuñó la formulación que mejor la resume: una historia de usuario es una promesa de conversación.

El contraste con el requisito tradicional aclara la intención. Un requisito aspira a ser completo, aprobado y estable, y su valor está en que no haga falta preguntar nada más. Una historia aspira a lo contrario, a ser lo bastante pequeña e incompleta como para que merezca la pena hablar de ella antes de construirla. Mike Cohn lo enuncia sin rodeos: lo escrito es un recordatorio, y el requisito real emerge en la conversación, los ejemplos, las reglas, los bocetos, las pruebas y las decisiones.

Conviene también distinguirla de dos vecinos con los que se confunde. Un **caso de uso** describe un flujo completo con sus alternativas y sus excepciones, y es una herramienta valiosa cuando el flujo

importa (volveremos a él en el capítulo 24). Una **tarea** es una porción del trabajo técnico necesario para completar algo, no una porción de valor observable: «crear la tabla de usuarios en la base de datos» no es una historia, aunque alguien la haya escrito con su plantilla.

1.2 Las tres C: tarjeta, conversación y confirmación

En agosto de 2001, Ron Jeffries ordenó la técnica en tres partes que siguen siendo la mejor forma de explicarla. La **tarjeta** contiene «lo justo para identificar el requisito y recordar a todos de qué se trata», y por eso el soporte tradicional era una ficha de cartón, cuyo tamaño impedía escribir demasiado. La **conversación** es donde el requisito se comunica de verdad, mediante un intercambio de ideas y de opiniones entre quien tiene la necesidad y quienes la van a resolver. La **confirmación** son los ejemplos y las pruebas de aceptación que dirán si lo construido responde a la necesidad.

Las tres, no una. Una historia con tarjeta y sin conversación es una especificación disfrazada; con conversación y sin confirmación, un acuerdo que nadie podrá comprobar. El trabajo profesional consiste en sostener las tres.

Siete años después, Jeffries volvió sobre el asunto con una frase que conviene tener a mano cuando alguien pide «historias mejor escritas»: una historia de usuario no es un documento de requisitos, es simplemente algo que hay que hacer; y lo escrito es probablemente la parte menos importante, muy por detrás de pensar, comunicar y comprobar.

1.3 La plantilla de Connextra y sus límites

La plantilla «Como [tipo de usuario], quiero [algo], para [un beneficio]» se inventó en 2001 en Connextra, una empresa británica, y de ahí su nombre. Nació de un problema concreto: quienes escribían las peticiones eran de ventas y de marketing, anotaban la funcionalidad y se olvidaban de decir quién la necesitaba y por qué, de modo que el equipo tenía que localizar al interesado para poder conversar. La plantilla obliga a incluir esos dos datos.

El glosario de Agile Alliance la describe como unas ruedas de aprendizaje y advierte de que dedicar mucho esfuerzo a cumplirla no tiene demasiado sentido. Cohn, que la popularizó, señala hoy dos límites: el «como usuario» genérico, que a menudo indica que el elemento no encaja en el molde de historia, y el seguimiento mecánico de la fórmula. La tercera parte, el porqué, es la que más se descuida y la que más aporta, porque una necesidad con su beneficio explícito admite varias soluciones y permite juzgarlas.

En Vantia, la plataforma de gestión fiscal que acompaña los ejemplos de esta guía, la petición que llegó del área comercial fue «necesitamos un chatbot». Formulada así, no hay usuario, no hay necesidad y no hay porqué, solo una solución elegida por quien la pide. La misma cosa, después de conversar con los autónomos que abandonaban la plataforma de noche, se convirtió en algo muy distinto: cuando alguien se atasca con un trámite fuera del horario de su asesoría, quiere una explicación clara en el momento para poder terminar la gestión esa misma noche.

1.4 La unidad de trabajo y lo que Scrum pide

Alrededor de la historia se ha construido un vocabulario de tamaños que conviene usar con soltura y sin dogmatismo: **tema**, **épica**, **historia** y **tarea** son contenedores por volumen, cómodos para

organizar una conversación, y no categorías con propiedades formales. Volveremos sobre ellos en el capítulo 14, al hablar de cuándo dividir.

Lo que Scrum exige de un elemento del Product Backlog es mucho menos de lo que suele creerse: una descripción, un orden y un tamaño. Nada más. La forma de expresar ese elemento pertenece al oficio y se elige según el caso, y a eso dedicamos el bloque F entero. Empezar por aquí evita la discusión estéril de si «hay que» usar historias.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Explicar el origen de la historia de usuario y qué problema resolvía, citando sus fuentes.
- Distinguir una historia de un requisito tradicional, de un caso de uso y de una tarea.
- Enunciar las tres C y decir qué aporta cada una al entendimiento del trabajo.
- Usar la plantilla de Connextra cuando ayuda y prescindir de ella cuando estorba, sin perder el usuario ni el porqué.
- Reformular como necesidad con beneficio una petición que llega ya convertida en solución.

Errores y antipatrones frecuentes

Tratar la tarjeta como el entregable. Cuando el esfuerzo se concentra en redactar bien la tarjeta, la conversación se da por celebrada y los malentendidos llegan intactos a la revisión.

Escribir «como usuario, quiero…». Un usuario genérico no informa de nada y suele ser la señal de que no se sabe quién se beneficia; si de verdad no hay un usuario distinguible, el elemento probablemente no es una historia.

Rellenar el «para» con una tautología. «Para poder verlo en la pantalla» no es un beneficio; el porqué debe poder discutirse, y si no lo hay, conviene averiguarlo antes de construir.

Convertir tareas técnicas en historias. El BoK de Scrum Manager señala este como el error más frecuente en español: la tarea disfrazada de historia impide priorizar por valor y esconde a quién sirve el trabajo.

Para profundizar

· Jeffries — Card, Conversation, Confirmation

· Jeffries — How should user stories be written?

· Agile Alliance — User story template

· Cohn — User Stories (guía de referencia)

· Scrum Manager BoK — Historia de usuario

BLOQUE A · LA HISTORIA DE USUARIO: QUÉ ES Y QUÉ LUGAR OCUPA

Capítulo 2. Qué dicen los marcos: de la neutralidad de Scrum a la institucionalización de SAFe · ESTABLECIDO

Qué exige cada marco sobre las historias de usuario, para no defender como norma lo que es una elección del equipo.

Introducción

Buena parte de las discusiones sobre historias de usuario se resolverían leyendo lo que dicen los marcos, que es bastante menos de lo que se les atribuye. En este capítulo repasamos las posiciones con sus fuentes, porque conocerlas ahorra tiempo en las organizaciones donde alguien afirma que «Scrum obliga» a algo.

2.1 Scrum no las prescribe

La Guía de Scrum de 2020 no contiene la expresión «historia de usuario». Lo que pide de un elemento del Product Backlog es que tenga descripción, orden y tamaño, y describe el refinamiento como el acto de desmenuzar y precisar esos elementos. La versión de 2017 enumeraba tipos («funcionalidades, funciones, requisitos, mejoras y correcciones») y esa lista desapareció en 2020, lo que refuerza la lectura.

Scrum.org lo enunció sin ambigüedad en 2017, en un artículo cuyo título es ya la respuesta: Scrum no prescribe ni requiere las historias de usuario. LeSS es igual de explícito al decir que el Product Backlog no contiene historias, sino elementos. Scrum Alliance apunta en la misma dirección al describir el artefacto.

Consecuencia práctica. Si nadie obliga, la elección del formato es una decisión profesional que hay que saber justificar. Esa justificación es el contenido del bloque F.

2.2 SAFe y Disciplined Agile las formalizan

En el extremo opuesto, SAFe convierte la historia en un artefacto con reglas. Define la historia de usuario con la plantilla y con INVEST, fija que debe poder terminarse en pocos días y añade un segundo tipo, la **historia habilitadora** (*enabler story*), con cuatro variedades: exploración, arquitectura, infraestructura y cumplimiento normativo. Es una respuesta razonable al problema de coordinar muchos equipos, y conviene conocerla porque muchas organizaciones grandes trabajan así.

Disciplined Agile, dentro de PMI, describe la descomposición de funcionalidades en historias como una práctica con su procedimiento. El PMI-ACP incluye en su temario descomponer los elementos del backlog cuando haga falta, sin prescribir formato.

Kanban es neutral en cuanto a la forma de los elementos: su guía habla de elementos de trabajo y se concentra en el flujo, la visualización y las políticas explícitas. Esa neutralidad es coherente con el enfoque de esta guía, y en el capítulo 14 veremos qué aporta su idea de dimensionar los elementos según la expectativa de servicio.

2.3 La ingeniería de requisitos las trata como una técnica

El BABOK ágil del IIBA y el material de IREB sobre requisitos ágiles sitúan la historia donde le corresponde desde su punto de vista: una técnica de expresión de requisitos entre varias, con sus fortalezas y sus límites. El material de IREB recoge las tres C e INVEST, y subraya que las propiedades más útiles del acrónimo son la estimabilidad, el tamaño y la comprobabilidad.

Esta mirada aporta algo que la literatura ágil suele dar por sabido: existen contextos, como los sistemas con certificación regulatoria, donde la historia sola no basta y conviene combinarla con requisitos formales o con casos de uso. El capítulo 24 lo desarrolla con criterios de decisión.

2.4 El debate sobre el propio término

Hay una discusión abierta y sana sobre si conviene seguir llamando «historias» a los elementos del backlog. Quienes proponen abandonar el término argumentan que arrastra una confusión difícil de deshacer, porque hace pasar por norma una técnica y empuja a forzar la plantilla en trabajo que no la admite. Quienes lo defienden recuerdan que el nombre viene de cómo se usa la técnica, contando historias, y que el problema no está en la palabra.

La posición de Scrum Manager, que esta guía adopta, es intermedia y práctica: conservar el término por su valor comunicativo y su arraigo, enseñarlo con su origen y sus límites, y dejar claro desde el principio que un elemento del Product Backlog no tiene por qué ser una historia de usuario.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Decir qué exige y qué no exige la Guía de Scrum sobre las historias de usuario, con su texto.
- Explicar cómo formalizan la historia SAFe y Disciplined Agile, y qué son las historias habilitadoras.
- Situar la historia como una técnica de requisitos entre otras, según el BABOK ágil e IREB.
- Sostener en su organización que el formato del elemento del backlog es una decisión del equipo y no una norma del marco.

Errores y antipatrones frecuentes

Atribuir a Scrum lo que Scrum no dice. Es la fuente de discusiones interminables; basta el texto de la guía para cerrarlas, y conviene tenerlo localizado.

Aplicar las reglas de SAFe fuera de SAFe. El tamaño de pocos días o la tipología de habilitadores tienen sentido dentro de su marco de coordinación; trasplantarlas sin ese contexto añade burocracia sin beneficio.

Descartar los casos de uso por «no ser ágiles». Hay flujos y contextos regulados donde son la herramienta adecuada, y renunciar a ellos por adhesión al método es una decisión mal fundada.

Para profundizar

· Guía de Scrum 2020

· Scrum.org — Mito 4: el backlog no tiene que ser historias

· SAFe — Enablers

· LeSS — Product Backlog

· IREB — RE@Agile

BLOQUE A · LA HISTORIA DE USUARIO: QUÉ ES Y QUÉ LUGAR OCUPA

Capítulo 3. El entendimiento compartido como criterio de cuánto escribir · ESTABLECIDO

El criterio que decide cuánto conviene escribir en cada caso, y el hilo que recorre toda la skill.

Introducción

Si la historia es una promesa de conversación, queda una pregunta que nadie puede responder con una plantilla: cuánto hay que escribir. Escribir de más no crea entendimiento y consume tiempo; escribir de menos deja el trabajo a merced de la interpretación de cada cual. En este capítulo presentamos el criterio con el que se decide, que es también el hilo conductor de toda la guía.

3.1 Explícito, implícito y falso

Martin Glinz y Samuel Fricker propusieron un modelo del entendimiento compartido en ingeniería del software que da nombre a lo que la práctica ágil venía haciendo por intuición. El entendimiento puede ser **explícito**, cuando está escrito y puede comprobarse, o **implícito**, cuando vive en la cabeza de quienes trabajan juntos y se sostiene en el contexto que comparten. Y, en ambos casos, puede ser **verdadero** o **falso**.

El caso peligroso es el entendimiento implícito falso: todos creen estar de acuerdo, nadie ha comprobado nada y la discrepancia aparece cuando el trabajo ya está hecho. Es exactamente el fenómeno que hunde las historias «bien escritas» que nadie conversó. La conversación no elimina ese riesgo por sí sola, pero es lo único que lo hace visible a tiempo, porque obliga a que las interpretaciones distintas choquen.

El criterio. Se escribe lo que hace falta para que el entendimiento pase de implícito a explícito allí donde equivocarse sea caro. Ni más, ni en otro sitio.

3.2 Por qué la conversación es más rica que el documento

Cockburn ordenó los canales de comunicación por su riqueza: dos personas ante una pizarra transmiten más, y más rápido, que un documento revisado; el vídeo transmite más que el audio, y el audio más que el papel. La razón es que el canal rico permite preguntar, señalar, corregir y comprobar el entendimiento en el momento, mientras el documento solo permite leer.

De ahí la consigna de Gojko Adzic, que resume el oficio en cuatro palabras: contar historias, no escribirlas. No significa no escribir nada, sino que lo escrito sirve a la conversación y no la sustituye. Patton lo dice desde otro ángulo, recordando que compartir documentos no es compartir entendimiento.

3.3 Qué cambia con la inteligencia artificial

Durante veinticinco años, la restricción práctica que protegía este criterio fue el coste de escribir: nadie redactaba veinte historias con criterios detallados porque llevaba días. Ese coste ha desaparecido. Hoy, producir texto plausible sobre cualquier producto cuesta minutos, y el entendimiento compartido sigue costando lo mismo que siempre.

La consecuencia es que el criterio importa más, no menos. Cuando el volumen deja de ser una señal de esfuerzo, deja también de ser una señal de calidad, y el valor profesional se desplaza a tres tareas que la asistencia no hace por nadie: dar el contexto adecuado, criticar lo que devuelve y verificar que el resultado responde a la intención. Los bloques G y H desarrollan esas tres tareas.

3.4 El recorrido de esta guía

Con ese hilo, el recorrido queda ordenado. Los bloques A y B fijan qué es una historia y cómo se juzga su calidad y la del conjunto. El bloque C lleva la historia a la conversación y la cierra con reglas, ejemplos y criterios. El bloque D enseña a dividir sin perder valor, que es donde más equipos se atascan. El bloque E devuelve el contexto que la lista ordenada pierde. El bloque F responde a qué hacer cuando el trabajo no es una historia. Y los bloques G y H tratan la inteligencia artificial, primero como asistente del trabajo con historias y después como quien las implementa.

Todos los ejemplos concretos usan un mismo producto ficticio, **Vantia**, el de la guía troncal Product Owner 2.0: una plataforma donde miles de autónomos llevan su facturación, sus gastos y sus impuestos, vendida por licencias a las asesorías que los atienden. Su problema conocido es que casi un tercio de quienes la usan de noche la abandona antes del primer año, porque se atascan y no tienen a quién preguntar.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Explicar el entendimiento compartido con sus cuatro casos y reconocer el implícito falso en una situación real.
- Decidir cuánto escribir en una historia concreta con ese criterio, en lugar de con una plantilla o una lista fija.
- Argumentar por qué la conversación es un canal más rico que el documento, y cuándo aun así conviene escribir.
- Explicar qué cambia en el trabajo con historias cuando escribir deja de tener coste.

Errores y antipatrones frecuentes

Medir el refinamiento por el detalle escrito. Un backlog muy detallado puede tener menos entendimiento compartido que uno breve y conversado; el detalle no es una medida de preparación.

Buscar el acuerdo aparente. Preguntar «¿está claro?» produce asentimientos; hacer chocar interpretaciones con ejemplos concretos produce entendimiento.

Escribir para protegerse. La historia redactada como contrato para poder señalar culpables después convierte el documento en el objetivo y desactiva la conversación.

Para profundizar

· Glinz — Requirements Engineering, reflexión personal (con el modelo de 2015)

· Adzic y Evans — Fifty Quick Ideas to Improve Your User Stories

· Fowler — UserStory

· Skill Arena — Product Owner

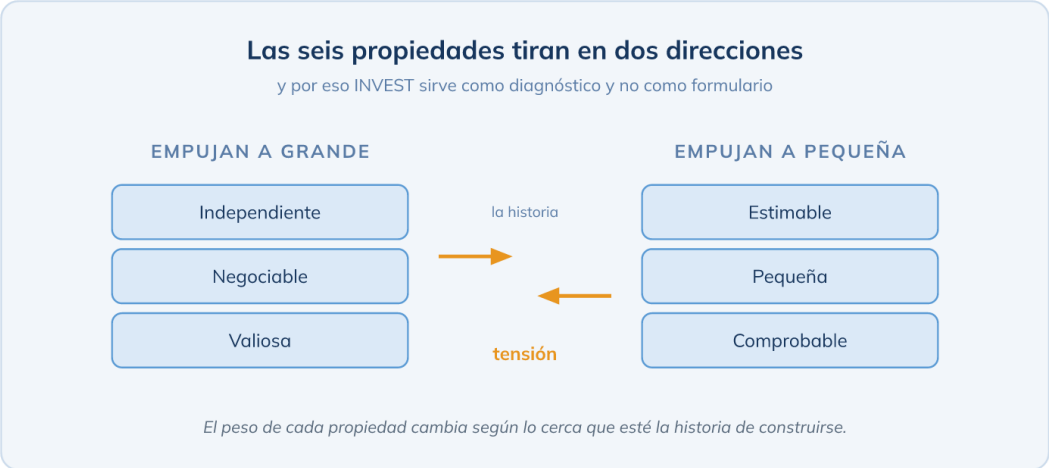
BLOQUE B · LA CALIDAD DE UNA HISTORIA Y DEL CONJUNTO

Capítulo 4. INVEST y su relectura: guía de trabajo, no lista de comprobación · ESTABLECIDO

Las seis propiedades, sus preguntas de diagnóstico, la tensión entre ellas y la revisión que hizo su propio autor.

Introducción

INVEST es el criterio de calidad más citado del oficio y también el más maltratado. Bill Wake lo publicó en 2003 como un acrónimo mnemotécnico para recordar seis propiedades deseables de una historia, y en muchas organizaciones ha terminado convertido en un formulario que alguien rellena antes de dejar entrar un elemento en el sprint. En este capítulo lo recuperamos como lo que es, un instrumento de diagnóstico para cuando una historia se resiste.



4.1 Las seis propiedades y su pregunta útil

Independiente. Wake pedía que las historias no se solapen en concepto y puedan ordenarse e implementarse en cualquier orden, y añadía la salvedad de que no siempre se consigue. La pregunta de diagnóstico es directa: ¿podríamos entregar esta y no la siguiente? Si la respuesta es no, hay una dependencia que conviene hacer visible antes de planificar.

Negociable. Una historia no es un contrato de funcionalidades; el detalle se construye entre quien tiene la necesidad y quien la resuelve. Pregunta: ¿queda algo que decidir juntos, o ya está todo cerrado por escrito?

Valiosa. Debe aportar valor a alguien identificable. Pregunta: ¿quién nota la diferencia cuando esto se entregue, y en qué? Si no hay respuesta, no es que la historia esté mal escrita, es que no está lista para construirse.

Estimable. No hace falta precisión, solo el conocimiento suficiente para poder ordenarla y planificarla. Pregunta: ¿sabemos bastante para decir si es grande o pequeña? Si no, lo que procede es investigar, no adivinar.

Pequeña. Aquí está el cambio de escala más llamativo, que veremos en 4.3. Pregunta: ¿cabe holgadamente en un ciclo, con sitio para conversarla y comprobarla?

Comprobable. Debe poder decirse qué observaremos para saber que está resuelta. Pregunta: ¿sabríamos escribir un ejemplo que la confirme? Si no, falta conversación, y el bloque C está dedicado a eso.

4.2 La tensión interna, que es lo que lo hace útil

Las seis propiedades no tiran en la misma dirección, y esa es precisamente su utilidad como instrumento. La independencia, la negociabilidad y el valor empujan hacia historias grandes, porque una porción de valor completo suele necesitar varias piezas. La estimabilidad, el tamaño y la comprobabilidad empujan hacia historias pequeñas. Un elemento que cumpliera las seis a la perfección sería una rareza.

Cómo se usa la tensión. El peso de cada propiedad cambia según lo cerca que esté la historia de construirse: en el horizonte lejano mandan el valor y la independencia; cuando entra en el sprint, mandan el tamaño y la comprobabilidad. Diagnosticar consiste en preguntarse qué propiedad falta *ahora*.

De ahí que INVEST funcione mal como puerta de entrada y bien como diagnóstico. Cuando una historia se resiste, el acrónimo dice por dónde mirar: si no es divisible sin perder valor, casi siempre lo que falta es el valor (lo veremos en el capítulo 17); si no es comprobable, faltan ejemplos; si no es estimable, falta información y probablemente un spike.

4.3 La relectura del propio autor y el cambio de escala

Wake ha revisado su acrónimo en público durante dos décadas. Matizó la independencia en 2012, propuso leer la S como **escalable** en 2016 (una historia puede crecer o encogerse según el momento, y lo importante es que admita ese ajuste) y en 2021 publicó «All You Need is INVEST? No!», donde añade la propiedad de ser **de extremo a extremo**, es decir, que atraviese el sistema en lugar de quedarse en una capa. Esa última propiedad anticipa el corte vertical del capítulo 15.

El cambio de escala merece atención. El texto de 2003 dice que una historia representa «como máximo unas pocas semanas-persona de trabajo». SAFe habla hoy de historias que se terminan en pocos días o menos, y la práctica en equipos de producto ronda de medio día a dos días. No es que la definición haya cambiado: es que el tamaño que un equipo puede sostener con integración continua y despliegue frecuente es mucho menor que en 2003.

4.4 Lo que dice la evidencia sobre su uso

La encuesta de REFSQ de 2016, con 182 profesionales, ofrece tres datos que conviene conocer. El primero, que solo uno de cada cuatro equipos usa INVEST, mientras casi el 60 % usa la plantilla de Connextra y cerca del 40 % no sigue ninguna guía. El segundo, y más interesante: lo que mejora la calidad es **usar una plantilla**, no cuál se use. Y el tercero, que la utilidad de INVEST se concentra en los equipos que empiezan.

Los autores del estudio lo formulan como una recomendación que esta guía suscribe: INVEST no es una lista de verificación, sino una guía de trabajo. El BoK de Scrum Manager dice lo mismo con otras palabras, al describirlo como guía conversacional y no como formulario de calidad.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Diagnosticar una historia con las seis propiedades y decir cuál falta en cada caso.
- Explicar la tensión entre las propiedades y ajustar su peso según la cercanía de la historia al trabajo.
- Incorporar la relectura de Wake, con la lectura escalable del tamaño y la propiedad de extremo a extremo.
- Situar el tamaño recomendable en la escala actual y no en la de 2003.
- Rechazar con argumentos el uso de INVEST como puerta de entrada al sprint.

Errores y antipatrones frecuentes

Convertirlo en formulario. Un elemento que cumple las seis casillas puede seguir sin valor observable; el acrónimo no sustituye a la conversación que decide si merece la pena.

Exigir independencia absoluta. Wake ya avisó de que no siempre se logra; el trabajo profesional consiste en hacer visibles las dependencias, no en negarlas.

Confundir estimable con estimada. Que una historia admita una estimación no obliga a ponerle un número; el capítulo 14 trata el dimensionado empírico como alternativa.

Perseguir historias diminutas. Por debajo de cierto tamaño la pieza deja de ser una porción de valor y se convierte en un detalle de implementación disfrazado.

Para profundizar

· Wake — INVEST in Good Stories, and SMART Tasks

· Wake — All You Need is INVEST? No!

· Wake — Small/Scalable Stories

· Lucassen y otros — El uso de las historias en la práctica (REFSQ 2016)

· Scrum Manager BoK — INVEST

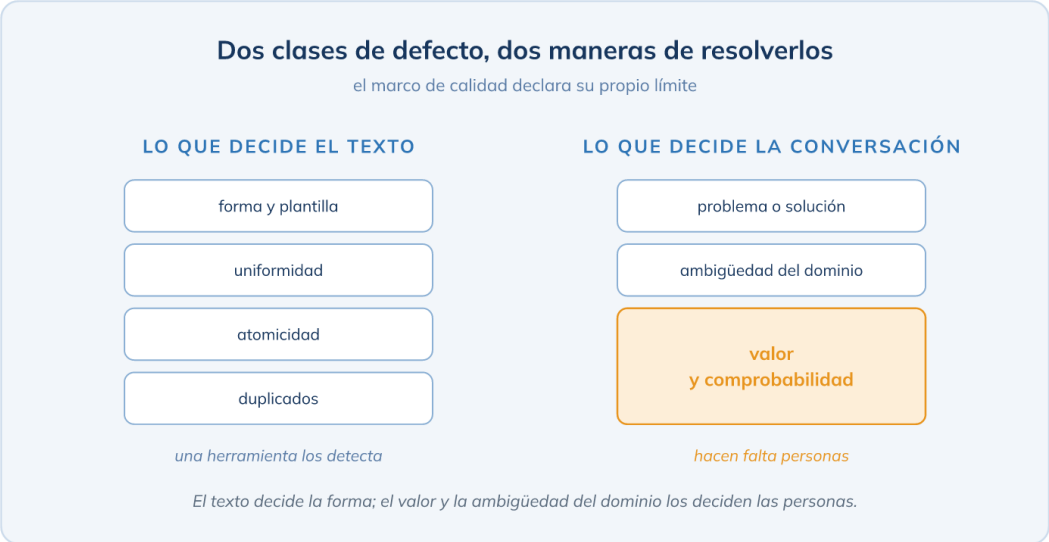
BLOQUE B · LA CALIDAD DE UNA HISTORIA Y DEL CONJUNTO

Capítulo 5. Defectos y ambigüedad: lo que decide el texto y lo que decide la conversación · EN CONSOLIDACIÓN

Qué defectos se detectan leyendo, cuáles exigen personas, y por qué una ambigüedad detectada es una oportunidad.

Introducción

Hay una distinción que ahorra mucho tiempo de refinamiento: no todos los defectos de una historia se resuelven de la misma manera. Unos se ven leyendo el texto y puede detectarlos una herramienta; otros solo aparecen cuando dos personas descubren que entendían cosas distintas. Saber en qué grupo está el defecto que tenemos delante decide quién debe resolverlo y cuánto va a costar.



5.1 El marco de calidad y su límite declarado

El marco **Quality User Story** (Lucassen, Dalpiaz, van der Werf y Brinkkemper, 2016) es el intento más riguroso de medir la calidad de una historia. Ordena trece criterios en tres categorías. Los **sintácticos** miran la forma: estar bien formada (con actor y acción), ser atómica y ser mínima. Los **semánticos** miran el contenido: ser conceptualmente sólida, estar orientada al problema y no a la solución, no ser ambigua y no entrar en conflicto con otras. Los **pragmáticos** miran el conjunto: frase completa, estimable, única, uniforme, independiente y completa.

Lo más valioso del marco es lo que declara dejar fuera. Sus autores excluyen expresamente el **valor** y la **comprobabilidad**, y explican por qué: no se deciden leyendo el texto. Esa frontera, dicha por quienes construyeron la herramienta de análisis, es el mejor argumento contra la idea de que la calidad de una historia pueda automatizarse por completo.

El reparto. Forma, uniformidad, atomicidad y duplicados: los detecta una herramienta o un modelo de lenguaje. Problema o solución, ambigüedad del dominio, valor y comprobabilidad: hacen falta personas que conozcan el producto.

5.2 Lo que se midió, y con qué resultado

El equipo aplicó su herramienta a 1.023 historias reales de 18 organizaciones. El 56 % tenía al menos un defecto, y los más frecuentes eran precisamente los mecánicos: falta de uniformidad, historias no mínimas (con detalle que sobra en la tarjeta) y no atómicas (dos necesidades en una). Es un dato útil porque señala dónde ayuda de verdad la automatización, y también su techo.

Trasladado al trabajo diario, sugiere un orden sensato: dejar que la revisión mecánica limpie la forma antes del refinamiento, para que la conversación se dedique a lo que solo la conversación resuelve. En el bloque G veremos que un modelo de lenguaje hace ese trabajo con buen resultado, y que ahí está su uso de mayor rendimiento y menor riesgo.

5.3 La ambigüedad tiene dos caras

La investigación sobre requisitos distingue varios tipos de ambigüedad: **léxica** (una palabra con más de un significado), **sintáctica** (una frase que admite dos análisis), **semántica** (una frase con dos interpretaciones en su contexto), **pragmática** (depende de quién la lea) más la **vaguedad** y la **generalidad**. Para el trabajo con historias, lo práctico es tener localizadas las construcciones de riesgo en español.

Las de la lista corta son: adjetivos valorativos sin medida (**adecuado**, **rápido**, **fácil**, **intuitivo**, **eficiente**), cuantificadores imprecisos (**varios**, **algunos**, **la mayoría**), la conjunción **y/o**, las cláusulas de escape (**según proceda**, **en la medida de lo posible**, **si es necesario**) y los verbos comodín (**gestionar**, **soportar**, **tratar**, **manejar**), que esconden un comportamiento sin describirlo. A esto se añaden los supuestos ocultos y el lenguaje demasiado técnico o demasiado genérico.

La segunda cara es la interesante. La investigación de Ferrari, Spoletini y Gnesi sobre entrevistas de requisitos muestra que la ambigüedad detectada es una **señal de conocimiento tácito**: donde una expresión admite dos lecturas suele haber una regla de dominio que nadie ha dicho porque le parece obvia. De ahí la regla operativa: no hay que eliminar por escrito toda la ambigüedad, sino localizar la que tenga alta probabilidad de malentendido y alto coste, formular la pregunta y decidir si se documenta o basta con acordarlo.

5.4 El ejemplo de Vantia

«Como autónomo, quiero que el asistente resuelva rápido mis dudas fiscales, para no perder tiempo.» La historia está bien formada, es atómica y tiene su porqué, de modo que pasaría cualquier revisión mecánica. Y sin embargo esconde tres ambigüedades que deciden el trabajo: **rápido** (¿un minuto?, ¿un segundo?), **resolver** (¿responder?, ¿responder con la referencia normativa?, ¿derivar al gestor cuando no sepa?) y **dudas fiscales** (¿cualquiera?, ¿solo las relacionadas con sus propios datos?).

Ninguna de las tres se resuelve leyendo. La primera se cierra con un número acordado; la segunda es una regla de negocio que en Vantia resultó ser crítica, porque una respuesta inventada es peor que reconocer el límite; y la tercera delimita el alcance. Las tres salieron de una conversación de veinte minutos, y las dos últimas se convirtieron en criterios de aceptación con sus ejemplos.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Clasificar un defecto de una historia como mecánico o conversacional y decidir quién lo resuelve.
- Aplicar los trece criterios del marco de calidad y explicar por qué el valor y la comprobabilidad quedan fuera.
- Localizar en un texto las construcciones de riesgo en español y formular la pregunta que las cierra.
- Decidir qué ambigüedades merecen eliminarse por escrito y cuáles basta con acordar en la conversación.
- Usar una ambigüedad detectada para aflorar una regla de dominio que nadie había enunciado.

Errores y antipatrones frecuentes

Pulir la forma y llamarlo refinamiento. Uniformar la redacción de cincuenta historias no cambia lo que el equipo entiende de ellas; es trabajo necesario, pero no es el refinamiento.

Perseguir la ambigüedad cero. Documentar cada matiz produce especificaciones que nadie lee y elimina la conversación que las haría innecesarias.

Confiar la detección de valor a una herramienta. Los propios autores del marco de calidad excluyen el valor porque no se decide con el texto; ninguna herramienta lo sabe.

Aceptar los verbos comodín. «Gestionar los gastos» puede significar cinco comportamientos distintos, y cada uno tiene un coste diferente; conviene desmenuzarlo antes de estimar.

Para profundizar

· Lucassen y otros — Quality User Story framework (REJ 2016)

· Lucassen y otros — Forging High-Quality User Stories (RE 2015)

· Femmer y otros — Requirements Smells

· Ferrari y otros — Ambigüedad y conocimiento tácito

· Berry y otros — Fuentes lingüísticas de ambigüedad

BLOQUE B · LA CALIDAD DE UNA HISTORIA Y DEL CONJUNTO

Capítulo 6. El catálogo de antipatrones de la historia · ESTABLECIDO

Los diez modos de fallo con nombre, cómo se reconoce cada uno y qué acción lo corrige.

Introducción

Los problemas de las historias se repiten tanto que tienen nombre. Conocer el catálogo ahorra la mitad de la discusión, porque cada patrón pide una acción distinta y la acción equivocada empeora las cosas: dividir una historia que en realidad está duplicada produce dos duplicados. En este capítulo recorremos los diez más frecuentes con su remedio.

6.1 Los antipatrones de la historia individual

Historia-tarea. Describe trabajo técnico sin decir a quién sirve («crear el índice de la tabla de facturas»). Se reconoce porque nadie fuera del equipo nota nada cuando se entrega. Remedio: buscar el beneficiario real o reclasificarla como elemento técnico, según el capítulo 23.

Usuario falso o genérico. «Como usuario, quiero…» o, peor, «como sistema, quiero…». Se reconoce porque el actor no permite discutir prioridad. Remedio: identificar el usuario concreto o admitir que el elemento no es una historia.

Solución disfrazada de necesidad. «Quiero un chatbot», «quiero un desplegable con las provincias». Se reconoce porque el «para» repite la solución con otras palabras. Remedio: excavar hasta el problema y volver a formular, que es lo que hizo Vantia con la petición del chatbot.

Historia sin porqué ni valor. Tiene actor y acción, y el beneficio es una tautología. Remedio: preguntar qué cambia para quien la pide, y si no hay respuesta, no entrar a construir.

Épica disfrazada de historia. Cabe en una frase, pero contiene semanas de trabajo («quiero gestionar mis impuestos»). Se reconoce por el verbo comodín y por la imposibilidad de dar un solo ejemplo que la confirme. Remedio: dividir con el bloque D.

6.2 Los antipatrones del conjunto

Fragmentación excesiva. Historias tan pequeñas que ninguna entrega valor por separado y hay que juntarlas todas para notar algo. Remedio: recombinar antes de volver a dividir, con un criterio de valor.

Duplicadas o contradictorias. Dos elementos que piden lo mismo con distintas palabras, o que se contradicen en una regla. Remedio: fusionar o resolver el conflicto en conversación, y volver a ordenar.

Obsoletas. Historias que respondían a un objetivo que ya no está vigente. Remedio: descartarlas sin ceremonia; el capítulo 7 trata la caducidad y la bancarrota del backlog.

Historia prescriptiva. Dicta la implementación o el diseño de la interfaz («con un menú lateral de tres pestañas»). Remedio: separar lo que es necesidad de lo que es propuesta, y devolver al equipo la decisión de cómo.

Historia tratada como contrato. Se cierra por escrito antes de conversarla y se invoca después para señalar incumplimientos. Remedio: es un problema de relación, no de redacción, y se corrige en el refinamiento (capítulo 8).

Cómo se usa el catálogo. En el refinamiento, nombrar el patrón en voz alta acorta la discusión y despersonaliza la crítica: se está diagnosticando un elemento del backlog, no la manera de escribir de un compañero.

6.3 De la detección a la acción

Los diez patrones se resuelven con seis acciones: **dividir**, **fusionar**, **reescribir**, **reclasificar** (dejar de tratarlo como historia), **posponer** y **descartar**. La correspondencia no es única, pero sí bastante estable: la épica disfrazada y la historia grande piden dividir; la fragmentación y las duplicadas, fusionar; el usuario falso y la solución disfrazada, reescribir; la historia-tarea, reclasificar; y las obsoletas, descartar.

El BoK de Scrum Manager señala como error frecuente en español el primero de la lista, escribir tareas técnicas en lugar de valor, y esa es también la experiencia de la mayoría de los equipos que empiezan. Conviene revisarlo antes que ningún otro.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Reconocer los diez antipatrones en un lote de historias reales y nombrarlos.
- Elegir la acción correctora que corresponde a cada patrón, sin aplicar la división como remedio universal.
- Distinguir una historia grande legítima de una épica disfrazada de historia.
- Usar el catálogo en el refinamiento como diagnóstico compartido y no como crítica personal.

Errores y antipatrones frecuentes

Dividir todo lo que no cabe. Si el problema es la duplicidad o la falta de valor, dividir multiplica el problema en lugar de resolverlo.

Confundir prescriptiva con detallada. Una historia puede llevar mucho detalle útil sobre reglas y ejemplos sin dictar la implementación; lo que estorba es la decisión técnica, no el detalle.

Guardar las obsoletas «por si acaso». Un backlog con elementos caducados obliga a releerlos en cada refinamiento y erosiona la confianza en el orden.

Para profundizar

· Pichler — Why User Stories Fail

· Pichler — 10 Tips for Writing Good User Stories

· Humanizing Work — ¿Todo el backlog tiene que ser historias?

· Scrum Manager BoK — Historia de usuario

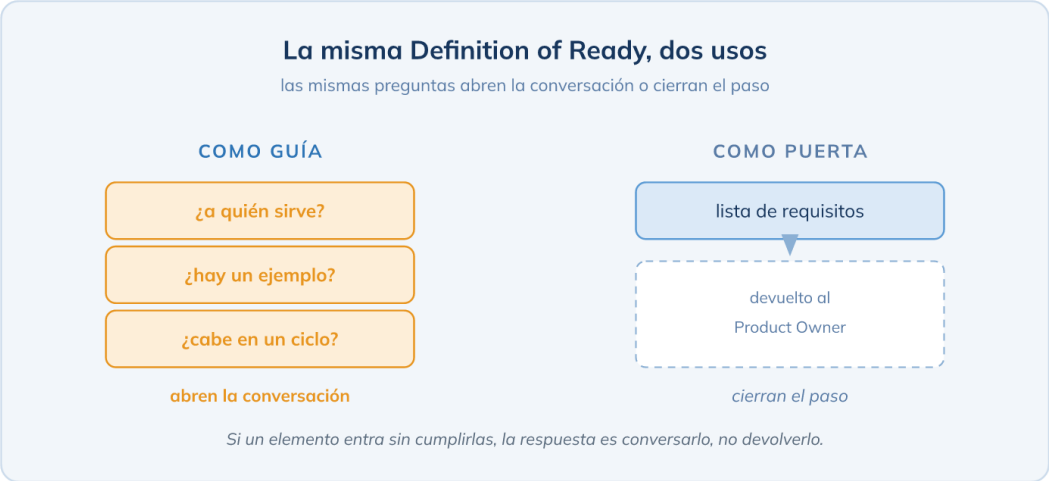
BLOQUE B · LA CALIDAD DE UNA HISTORIA Y DEL CONJUNTO

Capítulo 7. Salud del backlog y la Definition of Ready · ESTABLECIDO

Los criterios que no se ven en una historia aislada, la higiene del conjunto y cómo evitar que la Definition of Ready se convierta en una puerta.

Introducción

Un backlog puede estar lleno de historias impecables y ser inservible. La calidad del conjunto tiene criterios propios, y su deterioro es gradual: elementos que envejecen, duplicados que nadie detecta, detalle acumulado en cosas que nunca se harán y dependencias invisibles hasta que bloquean un sprint. En este capítulo vemos cómo se mide esa salud y cómo se mantiene.



7.1 Los criterios del conjunto

El marco de calidad del capítulo 5 aporta cinco criterios que solo tienen sentido mirando el backlog entero: **unicidad** (ningún elemento duplica a otro), **uniformidad** (formato razonablemente parecido, para poder leerlo de un tirón), **ausencia de conflictos** (dos elementos no piden reglas incompatibles), **independencia** (las dependencias están identificadas) y **completitud** (no falta una pieza sin la cual lo demás no se sostiene).

A ellos se añade el acrónimo **DEEP** de Roman Pichler, que describe el backlog sano: **detallado en proporción** (mucho detalle arriba, poco abajo), **estimado** (con el tamaño suficiente para ordenar), **emergente** (cambia con lo que se aprende) y **priorizado**. La primera propiedad es la que más se incumple, y su incumplimiento tiene una consecuencia costosa: se invierte esfuerzo en detallar lo que quizá nunca se construya.

7.2 Dependencias, edad y caducidad

Las dependencias de un backlog son de tres tipos: entre elementos (una historia necesita otra), con otros equipos y con terceros ajenos a la organización. La jerarquía para tratarlas es siempre la misma: **hacerlas visibles**, **eliminar las evitables** y **gestionar las inevitables**, en ese orden. El capítulo 17 vuelve sobre ellas, porque dividir mal las multiplica.

La edad de los elementos es un indicador infrautilizado. Un elemento que lleva meses en el backlog sin subir de orden está diciendo algo: o no importa, o nadie sabe cómo abordarlo, o responde a un objetivo abandonado. Poner fecha de revisión a los elementos y descartar sin ceremonia lo caducado mantiene el conjunto legible. Cuando el deterioro es general, existe una salida radical y saludable, declarar la **bancarrota del backlog**: archivar todo y reconstruir desde el objetivo vigente. Es menos costoso de lo que parece, porque lo importante vuelve a aparecer solo.

Requisitos no funcionales. El rendimiento, la accesibilidad o la seguridad casi nunca caben en una historia. Su sitio natural es una restricción transversal o un criterio de la Definition of Done, verificado en cada incremento. El capítulo 10 lo desarrolla.

7.3 La Definition of Ready y su deriva

La **Definition of Ready (DoR)** es un acuerdo del equipo sobre qué debe cumplir un elemento para poder empezar a trabajarlo. No forma parte de Scrum: la Guía de Scrum no la menciona y se limita a decir que los elementos que caben en un sprint se consideran listos para su selección. El BoK de Scrum Manager la documenta como práctica, con su variante para el trabajo asistido por inteligencia artificial.

Su deriva es conocida y la ha descrito Mike Cohn con crudeza: convertida en puerta, la DoR da al equipo un motivo formal para devolver trabajo y al Product Owner un formulario que rellenar, y sustituye la conversación por un trámite. Aparecen entonces las dos consecuencias típicas, un cuello de botella en el refinamiento y una relación de proveedor y cliente dentro del mismo equipo.

La forma sana de escribirla es como **guía de conversación**: unas pocas preguntas que ayudan a saber si merece la pena empezar (¿está claro a quién sirve?, ¿hay un ejemplo que la confirme?, ¿está identificada la dependencia?, ¿cabe en un ciclo?), acordadas por el equipo, revisables, y sin valor de veto. Si un elemento entra sin cumplirlas, la respuesta es conversarlo, no devolverlo.

7.4 Con inteligencia artificial

La revisión del conjunto es uno de los usos donde la asistencia rinde mejor, porque buena parte de los criterios de conjunto son mecánicos: detectar duplicados y solapes, señalar formatos dispares, encontrar reglas contradictorias entre elementos y listar los que llevan más tiempo sin moverse. Conviene pedirle esa revisión con los criterios explícitos, uno a uno, y quedarse con la lista de candidatos para decidir en conversación.

Lo que no debe delegarse es la consecuencia: descartar, fusionar o reordenar sigue siendo una decisión de producto con responsable identificable. El bloque G desarrolla ese reparto.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Evaluar la salud de un backlog con los criterios de conjunto y con DEEP.
- Identificar las dependencias de un backlog y tratarlas en el orden adecuado.
- Detectar elementos caducados y decidir cuándo procede declarar la bancarrota del backlog.
- Reconocer una Definition of Ready convertida en puerta y reformularla como guía de conversación.

- Situar los requisitos no funcionales fuera de la historia individual, como restricción o criterio de la Definition of Done.

Errores y antipatrones frecuentes

Detallar todo el backlog. El detalle en los elementos lejanos se desperdicia cuando el objetivo cambia, y además da una falsa sensación de control.

Usar la DoR para devolver trabajo. El equipo y el Product Owner comparten la responsabilidad del entendimiento; una puerta formal la rompe y crea dos bandos.

Contar el backlog en número de elementos. Un backlog de trescientos elementos no dice nada sobre su salud; lo que importa es el detalle en proporción, la ausencia de conflictos y la edad.

Meter los requisitos no funcionales como historias. «Como usuario, quiero que la página cargue rápido» no tiene beneficiario distinguible ni cabe en un sprint: es una restricción del sistema.

Para profundizar

· Pichler — Make the Product Backlog DEEP

· Cohn — Definition of Ready: por qué es peligrosa

· Scrum.org — ¿Por qué la DoR no está en la Guía de Scrum?

· ProductPlan — Cuándo declarar la bancarrota del backlog

· Scrum Manager BoK — A punto

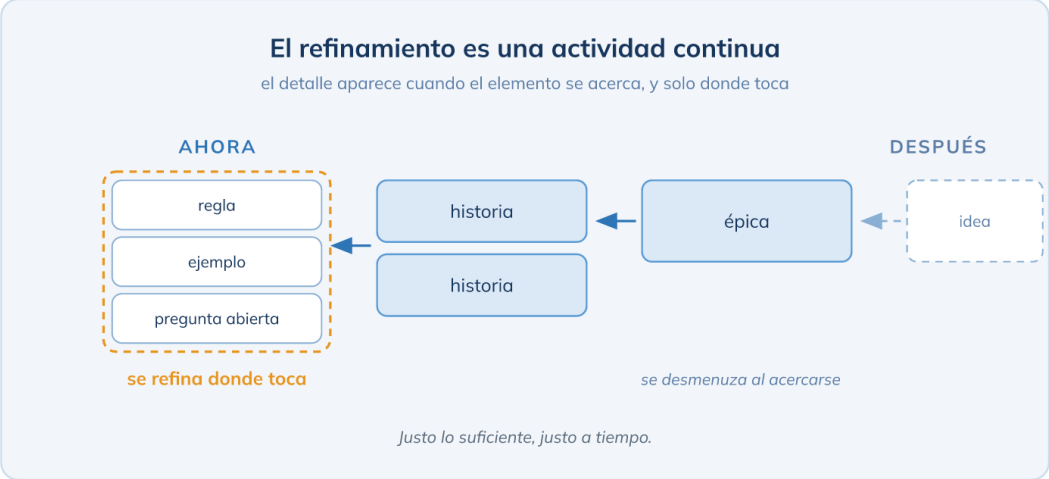
BLOQUE C · LA CONVERSACIÓN Y SU CONFIRMACIÓN

Capítulo 8. Refinamiento continuo: ni traspaso ni reunión · ESTABLECIDO

El refinamiento como actividad continua, sus dos degeneraciones clásicas y la versión nueva que trae la inteligencia artificial.

Introducción

El refinamiento es la actividad donde se decide casi todo lo que esta guía enseña, y también donde se pierde. Sus dos degeneraciones son opuestas y equivalentes en el daño: alguien que refina solo y entrega el resultado, o una reunión periódica donde el equipo rellena campos. En este capítulo vemos qué dice Scrum, qué patrones de organización funcionan y cómo se reconoce cada degeneración.



8.1 Qué dice Scrum y qué no

La Guía de Scrum define el refinamiento del Product Backlog como el acto de desmenuzar y precisar los elementos en otros más pequeños y definidos, y añade que es una actividad continua. No lo describe como un evento, no fija duración, no dice quién asiste y no lo hace obligatorio. Scrum.org lleva años desmontando el mito contrario, porque la práctica mayoritaria lo convirtió en una reunión semanal.

El BoK de Scrum Manager fija la referencia en español y aporta dos datos útiles: la proporción de capacidad que suele dedicarse (en torno al 5 %, aunque conviene medirlo en el propio equipo) y el rechazo del término *grooming*, retirado por sus connotaciones.

Actividad, no evento. Que el refinamiento sea continuo no significa que no haya sesiones. Significa que las sesiones sirven a la conversación que hace falta ahora, con quien hace falta, y no a un calendario.

8.2 El traspaso y sus disfraces

El **traspaso** (*handoff*) es la degeneración más silenciosa, porque parece eficiencia. Se reconoce en tres disfraces. El primero, el Product Owner que refina solo y entrega elementos cerrados; el equipo

recibe trabajo, no participa en definirlo, y las preguntas aparecen durante el sprint. El segundo, el «autor de historias», alguien tan bueno redactando que los demás dejan de intentarlo, con el efecto de que su esfuerzo desincentiva la participación. El tercero, los criterios de aceptación entregados como contrato, que convierten la conversación en negociación.

La inteligencia artificial ha añadido una cuarta versión, más rápida y más difícil de detectar: producto genera historias con un modelo, desarrollo las interpreta con otro modelo, y nadie ha hablado con nadie. El resultado tiene mejor acabado que nunca y el mismo entendimiento compartido que un correo reenviado. Volveremos sobre ello en el capítulo 27.

El antídoto no es organizativo, es de responsabilidad: el entendimiento del trabajo es responsabilidad compartida. Quien tiene la necesidad responde de que se entienda; quien la va a resolver responde de preguntar lo que no entiende.

8.3 Patrones de organización que funcionan

Horizonte corto. Se refina lo que está cerca de construirse, uno o dos ciclos por delante, y no todo el backlog. El detalle en proporción del capítulo 7 se sostiene solo con esta disciplina.

Grupos pequeños. Una conversación de tres o cuatro personas avanza; una de diez se convierte en una presentación. Las estructuras que reparten el trabajo en parejas o tríos y luego integran (del tipo «primero solo, después en parejas, después todos») funcionan mejor que la mesa redonda.

Divergir y converger. Primero se abren interpretaciones, preguntas y casos límite; después se cierra lo que hay que cerrar. Mezclar las dos fases produce la sensación de que el grupo discute sin avanzar.

Trabajo silencioso primero. Cinco minutos de lectura y anotación individual antes de hablar evitan que la primera opinión ancle a todo el grupo.

Timeboxes por historia. Poner límite a la conversación de cada elemento (el Example Mapping del capítulo 11 propone unos veinticinco minutos) obliga a decidir si está lista, si hay que partirla o si hay que investigar.

8.4 Los antipatrones con nombre

Refinamiento como reunión. Convocatoria fija, todo el equipo, orden del día largo y ningún cierre. Señal: nadie sabe decir qué se decidió.

Entrada de datos. La sesión se dedica a rellenar campos de la herramienta. Señal: el resultado se mide en elementos actualizados.

Criterios como contrato. Los criterios se cierran antes de conversar y se invocan después. Señal: la palabra «acordamos» aparece en las revisiones con tono de reclamación.

Cortes horizontales. El refinamiento produce piezas por capa técnica. Señal: ninguna de las piezas puede entregarse por separado. El capítulo 15 lo trata en detalle.

Dependencias técnicas que gobiernan el orden. El backlog se ordena por lo que es cómodo construir y no por valor. Señal: el orden no se parece a la estrategia del producto.

En Vantia, la primera versión del refinamiento del asistente fiscal cayó en el tercero: el Product Owner llevó los criterios redactados y la sesión se dedicó a discutir su redacción. Al reorganizarla con

un mapa de ejemplos, los mismos veinte minutos produjeron dos reglas nuevas y una pregunta que nadie había formulado, la de qué hacer cuando el asistente no encuentra apoyo en la normativa.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Explicar qué dice la Guía de Scrum sobre el refinamiento y qué le atribuye la práctica sin fundamento.
- Organizar el refinamiento como actividad continua, con horizonte corto y grupos pequeños.
- Reconocer los cuatro disfraces del traspaso, incluido el que introduce la asistencia de inteligencia artificial.
- Nombrar los cinco antipatrones del refinamiento a partir de sus señales.
- Facilitar una conversación de refinamiento con trabajo individual previo y límite de tiempo por elemento.

Errores y antipatrones frecuentes

Refinar todo el backlog. El detalle en los elementos lejanos caduca antes de usarse; refinar lejos es la forma más común de trabajar mucho sin avanzar.

Convocar a todo el equipo a cada conversación. La conversación productiva necesita las perspectivas necesarias, no la asistencia completa; el capítulo 9 lo desarrolla.

Medir el refinamiento por elementos actualizados. Lo que hay que medir es si el equipo puede empezar sin preguntas de bloqueo, y eso no se ve en la herramienta.

Refinar sin decidir. Una conversación que no termina en «está lista», «hay que partirla» o «hay que investigar» habrá que repetirla entera.

Para profundizar

· Guía de Scrum 2020

· Scrum Manager BoK — Mantenimiento de la pila del producto

· Scrum.org — ¿Qué es una historia de usuario?

· Ferrari y otros — Ambigüedad y conocimiento tácito

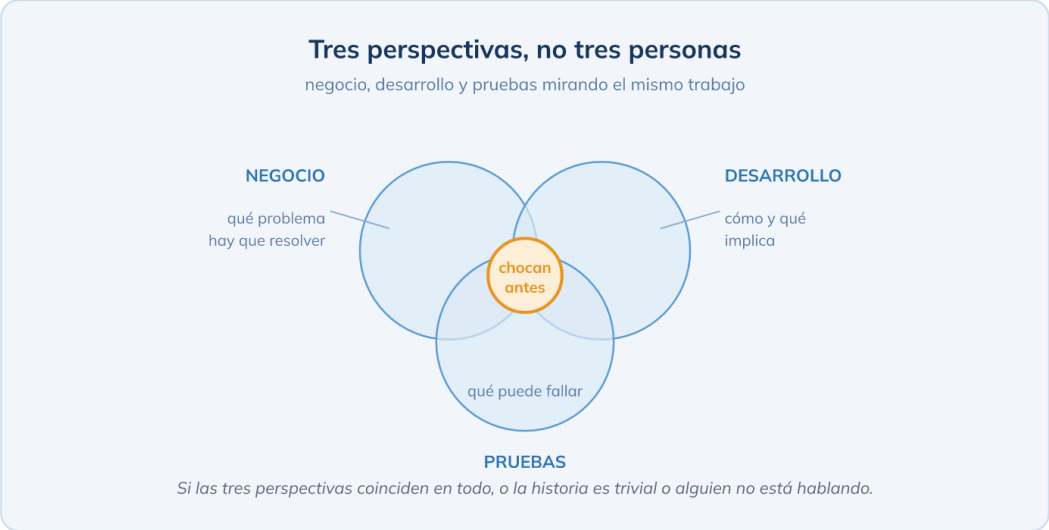
BLOQUE C · LA CONVERSACIÓN Y SU CONFIRMACIÓN

Capítulo 9. Quién conversa y en qué formato: Three Amigos y talleres de escritura · ESTABLECIDO

Las tres perspectivas necesarias, el taller de escritura y el criterio que envía la incertidumbre alta a un spike.

Introducción

Una conversación de refinamiento funciona cuando están presentes las perspectivas que van a chocar. Si falta alguna, el choque se produce igual, solo que más tarde y con código de por medio. En este capítulo vemos la práctica que nombra ese requisito, el formato de taller que la ordena y la escala que decide cuándo lo que hace falta no es conversar, sino investigar.



9.1 Three Amigos: tres perspectivas

George Dinwiddie propuso el nombre en 2009, en un artículo cuya pregunta de partida era qué hacer si no se automatizan las pruebas de aceptación. La respuesta fue reunir tres puntos de vista antes de construir: **negocio** (qué problema hay que resolver), **desarrollo** (cómo podría resolverse y qué implica) y **pruebas** (qué podría salir mal y cómo lo sabremos). Cuando el trabajo lo pide, se añaden diseño, operaciones, seguridad o cumplimiento normativo.

El matiz que más se pierde está en el nombre: son tres **perspectivas**, no tres **personas**. En un equipo pequeño, una misma persona puede aportar dos; en un trabajo delicado, la perspectiva de pruebas puede necesitar dos especialistas. La práctica circula también con otros nombres, *Triad* (Ken Pugh) y *Power of Three* (Lisa Crispin y Janet Gregory).

Para qué sirve de verdad. Su función es hacer que las interpretaciones distintas de la misma frase choquen antes de que cueste dinero, más que repartir tareas. Si las tres perspectivas coinciden en todo, o la historia es trivial o alguien no está hablando.

Sus errores están documentados y son cuatro: limitarlo literalmente a tres personas, ampliarlo a todo el equipo (con lo que vuelve a ser una reunión), convertirlo en un evento formal recurrente y llegar con los escenarios ya cerrados, que es la versión conversacional del traspaso.

9.2 El taller de escritura de historias

Cuando hay que abordar un conjunto y no una historia, el formato adecuado es el taller de escritura. Una agenda que funciona tiene siete pasos: repasar los elementos prioritarios y su objetivo; identificar usuarios y resultados esperados; aflorar supuestos y preguntas ocultas; reescribir lo que no se sostiene; partir lo que no cabe; esbozar criterios de aceptación con ejemplos; y cerrar con próximos pasos y responsables.

El taller no es una sesión de redacción colectiva, que produce prosa lenta y de comité. Es una sesión de decisión: qué entra, qué se parte, qué se investiga y qué se descarta. Lo escrito puede quedar en manos de una persona después, con el acuerdo tomado.

Para una historia concreta, el formato de veinticinco minutos del capítulo 11, Example Mapping, es más eficaz que cualquier taller largo, y conviene dominar los dos.

9.3 Preguntas antes que respuestas: el descubrimiento deliberado

Dan North llamó **descubrimiento deliberado** (*deliberate discovery*) a la disciplina de buscar activamente lo que no se sabe, en lugar de esperar que aparezca. Su argumento es que la ignorancia es el mayor coste del desarrollo, y que hay una variedad especialmente cara, la **ignorancia de segundo orden**: no saber que no se sabe.

Liz Keogh convirtió eso en una escala práctica de complejidad, que va del extremo «todos lo hemos hecho antes» al extremo «nadie lo ha hecho nunca», con peldaños intermedios («alguien de la organización lo ha hecho», «alguien en el mundo lo ha hecho», «alguien lo ha intentado»). La escala se usa para decidir qué hacer con cada historia: en los peldaños bajos, conversar y construir; en los altos, comprar información con un **spike** o un experimento en lugar de seguir conversando.

Es un criterio que ahorra sesiones enteras. Cuando el grupo lleva veinte minutos especulando, lo que falta no es más conversación: es un dato que nadie tiene.

9.4 Aplazar la decisión mientras las opciones estén abiertas

La última pieza de este capítulo es de temporización. Mientras las opciones sigan abiertas y el coste de esperar sea bajo, decidir tarde produce mejores decisiones, porque llegan con más información. Eso vale para el detalle de una historia, para su corte y para los criterios: cerrarlos en cuanto se enuncian es la manera más segura de fijar supuestos que aún no se han comprobado.

El límite está en el bloqueo: aplazar una decisión que impide avanzar no es prudencia, es indecisión. La regla operativa consiste en identificar la última fecha razonable para decidir y llegar a ella con las opciones vivas.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Convocar una conversación de refinamiento con las perspectivas necesarias y justificar quién falta o quién sobra.

- Facilitar un taller de escritura de historias con su agenda y cerrarlo con decisiones y responsables.
- Situar una historia en la escala de complejidad y decidir si procede conversar, investigar con un spike o experimentar.
- Reconocer los cuatro errores frecuentes de los Three Amigos.
- Identificar la última fecha razonable para cerrar una decisión y sostener el aplazamiento hasta entonces.

Errores y antipatrones frecuentes

Convertirlo en un rito de tres personas fijas. La composición depende del trabajo; congelarla convierte la práctica en un comité y deja fuera perspectivas necesarias.

Llegar con los escenarios cerrados. Si quien convoca trae ya la respuesta, la sesión sirve para validar, no para descubrir, y las tres perspectivas dejan de aportar.

Especular en lugar de investigar. Cuando la incertidumbre es de segundo orden, ninguna conversación la resuelve: hay que comprar información.

Cerrar los criterios en la primera sesión. Los criterios acordados demasiado pronto fijan supuestos sin comprobar y luego resulta caro cambiarlos.

Para profundizar

· Agile Alliance — Three Amigos

· Keogh — Estimating Complexity

· Extreme Programming — Create a Spike Solution

· Cohn — User Stories (guía de referencia)

BLOQUE C · LA CONVERSACIÓN Y SU CONFIRMACIÓN

Capítulo 10. Criterios de aceptación: formatos, límites y confusiones · ESTABLECIDO

Qué son, qué formatos admiten, en qué se distinguen de la Definition of Done y cómo se confirma un comportamiento probabilístico.

Introducción

Los criterios de aceptación son la tercera C, la confirmación, puesta por escrito. Son también la parte de la historia que más se confunde con otras cosas: con la Definition of Done, con los requisitos no funcionales y, en las organizaciones con mala relación entre negocio y desarrollo, con un contrato. En este capítulo los ponemos en su sitio.

10.1 Qué son y qué formatos admiten

Un criterio de aceptación es una **condición observable** que dirá que la necesidad está resuelta. Observable es la palabra clave: si no puede comprobarse mirando el comportamiento del producto, no es un criterio.

Hay cuatro formatos de uso corriente, y conviene elegir según el caso. La **lista de condiciones** es la más simple y sirve para la mayoría. El **escenario** «Dado que / Cuando / Entonces» ordena bien los casos con precondición, acción y resultado, y es el que popularizó el desarrollo guiado por el comportamiento. Las **reglas con ejemplos** funcionan mejor cuando el comportamiento depende de varias condiciones, y son el formato natural del Example Mapping. La **tabla** es imbatible cuando hay combinaciones de datos y resultados.

Elegir el formato. Si el escenario obliga a escribir seis pasos y una precondición artificial, el formato equivocado es el escenario. La forma sirve a la conversación, no al revés.

10.2 Criterios de aceptación y Definition of Done

Es la confusión más extendida y tiene una respuesta clara. Los **criterios de aceptación** son específicos de un elemento y dicen qué debe hacer *esto*. La **Definition of Done (DoD)** es común a todo el incremento y dice qué significa terminar *cualquier cosa* en este equipo (pruebas, revisión, documentación, accesibilidad, seguridad, despliegue). Scrum exige la segunda y no menciona los primeros.

Mezclarlas produce dos daños simultáneos: criterios que repiten en cada historia lo que ya exige la DoD, y una DoD que cambia con cada historia y deja de ser un acuerdo estable. La regla práctica: si algo debe cumplirse en todo lo que se entrega, va a la DoD; si es propio de esta necesidad, es un criterio.

Los **requisitos no funcionales** se reparten con el mismo criterio. El rendimiento general, la accesibilidad o el cifrado pertenecen a la DoD o a una restricción transversal. Un umbral específico de esta funcionalidad, en cambio, sí es un criterio de aceptación de esta historia.

10.3 Cuántos criterios y qué señala un exceso

No hay un número, pero sí una señal. Cuando una historia acumula muchos criterios (digamos, más de siete u ocho), casi siempre está diciendo una de estas tres cosas: que en realidad contiene varias historias, que se están escribiendo como criterios cosas que pertenecen a la DoD, o que alguien está intentando cerrar por escrito lo que debería conversarse.

Conviene además distinguir el criterio de su ejemplo. «Toda respuesta incluye la referencia normativa consultable» es un criterio; «al preguntar por la deducción de una comida de trabajo, la respuesta cita el artículo aplicable» es un ejemplo que lo confirma. Los criterios se mantienen pocos y canónicos; los ejemplos pueden ser varios. En el capítulo 29 veremos que esta distinción se vuelve crítica cuando el trabajo lo ejecuta un agente.

10.4 Cuando el comportamiento es probabilístico

Hay funcionalidades que no admiten un criterio binario porque su comportamiento es estadístico, y las respuestas de un asistente basado en un modelo de lenguaje son el ejemplo del día. «El asistente responde correctamente» no es comprobable como criterio de una historia, porque no existe una respuesta única.

La forma de confirmarlo es distinta: se define una **rúbrica** de lo que hace buena a una respuesta, se construye un **conjunto de evaluación** con casos que pasan, casos que fallan y casos límite, y el criterio pasa a ser un **umbral** sobre ese conjunto. En Vantia, la rúbrica del asistente fiscal tiene cuatro puntos (corrección según la normativa, referencia consultable, comprensible para quien no es gestor, e informar sin suplantar el asesoramiento del gestor), y el conjunto incluye un caso límite deliberado: una consulta dudosa donde la respuesta correcta no es un veredicto, sino explicar el criterio de la norma y derivar al gestor.

El instrumental completo de evaluación de funcionalidades con inteligencia artificial pertenece a la formación troncal de Product Owner y a la skill de IA aplicada al trabajo ágil; aquí basta con saber que el criterio de aceptación cambia de forma y no de función.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Escribir criterios de aceptación observables y elegir el formato que corresponde a cada caso.
- Separar con criterio lo que va a los criterios de aceptación y lo que va a la Definition of Done.
- Situar los requisitos no funcionales en el sitio adecuado según su alcance.
- Interpretar un exceso de criterios como señal de división pendiente o de conversación evitada.
- Formular la confirmación de una funcionalidad probabilística con rúbrica, conjunto de evaluación y umbral.

Errores y antipatrones frecuentes

Repetir la DoD en cada historia. Añade ruido a cada elemento y desactiva el acuerdo común; cuando algo vale para todo, su sitio es la Definition of Done.

Escribir criterios que dictan la interfaz. «Un botón azul arriba a la derecha» no es una condición observable de la necesidad, es una decisión de diseño colocada en el sitio equivocado.

Criterios sin un solo ejemplo. Un criterio abstracto sobrevive a interpretaciones opuestas; el ejemplo concreto es lo que hace chocar las interpretaciones.

Cubrir solo el camino feliz. Los casos límite y los negativos son donde se esconden las reglas de negocio que nadie ha dicho.

Pedir un criterio binario a un comportamiento probabilístico. Produce discusiones inacabables en la revisión; lo que hace falta es un umbral sobre un conjunto acordado.

Para profundizar

· Agile Alliance — Given-When-Then

· Scrum Manager BoK — Criterios de aceptación

· Scrum Manager BoK — Definición de hecho

· Cohn — Requisitos no funcionales como historias

· Skill Arena — Product Owner

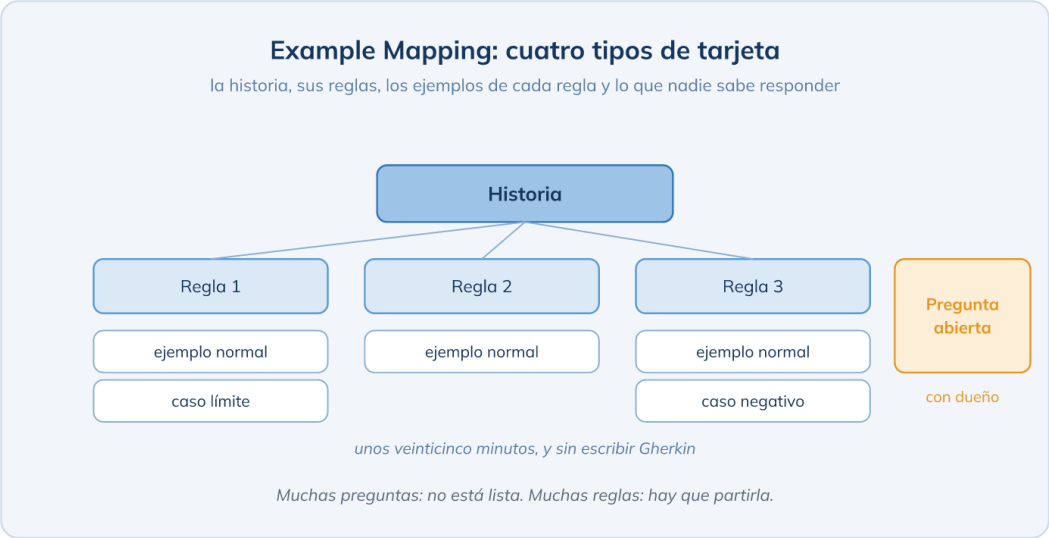
BLOQUE C · LA CONVERSACIÓN Y SU CONFIRMACIÓN

Capítulo 11. Reglas, ejemplos y Example Mapping · EN CONSOLIDACIÓN

Cómo se encuentran las reglas de una historia, qué ejemplos hacen falta y cómo se lee el mapa que resulta.

Introducción

Este capítulo contiene, probablemente, la técnica con mejor relación entre esfuerzo y resultado de toda la guía. Example Mapping ordena la conversación sobre una historia en veinticinco minutos y produce cuatro cosas a la vez: sus reglas, los ejemplos que las fijan, las preguntas que nadie sabe responder y un diagnóstico sobre si la historia está lista.



11.1 Reglas de negocio, explícitas e implícitas

Una **regla de negocio** es una condición que el comportamiento debe respetar siempre. Algunas están escritas en alguna parte (la normativa, un contrato, una política de la empresa); otras viven en la cabeza de quien conoce el dominio y no se dicen porque parecen obvias. Las segundas son las que hunden las entregas.

Junto a las reglas hay que aflorar sus **excepciones** y sus **condiciones**: cuándo no se aplica la regla, y qué hace el sistema entonces. La pregunta que mejor funciona para encontrarlas es una sola y conviene memorizarla: **¿hay algún contexto en el que esto sea distinto?**

11.2 Ejemplos que ilustran, no que agotan

Un **ejemplo** es un caso concreto, con datos concretos, que muestra la regla en acción. Su función es hacer visible el desacuerdo: dos personas pueden asentir ante una regla abstracta y discrepar en cuanto se pone un dato.

Hacen falta tres clases de ejemplo, y en ese orden: el **caso normal**, el **caso límite** (el valor en la frontera, el máximo, el cero, el vacío) y el **caso negativo** (lo que no debe ocurrir, o el error esperado). Lo que no hace falta es agotar las combinaciones: los ejemplos ilustran para que el equipo entienda la regla, y la exhaustividad pertenece al diseño de pruebas.

Ilustrar frente a agotar. Tres ejemplos bien elegidos enseñan la regla; veinte ejemplos la esconden y convierten la conversación en una revisión de tabla.

11.3 Example Mapping, paso a paso

Matt Wynne publicó la técnica en diciembre de 2015 y su protocolo es deliberadamente simple. Se usan cuatro colores de tarjeta: **amarilla** para la historia, **azules** para sus reglas, **verdes** para los ejemplos de cada regla y **rojas** para las preguntas que nadie sabe responder. Se coloca la amarilla arriba, y el grupo va poniendo reglas debajo, ejemplos bajo cada regla y preguntas a un lado, en silencio o hablando, según haga falta.

La duración declarada es de unos veinticinco minutos para una historia bien entendida y bien dimensionada, y esa cifra es parte de la técnica: si no se cierra, el propio hecho es información. Y hay una prohibición expresa: no se escribe Gherkin en la sesión. La formulación precisa viene después, y mezclarla con el descubrimiento paraliza la conversación en discusiones de sintaxis.

Conviene practicarla con frecuencia. Wynne recomienda hacerla cada dos días con las historias próximas, en lugar de reservarla para las difíciles: es en las que parecían fáciles donde suele aparecer la regla que faltaba.

11.4 Leer el mapa: cuatro diagnósticos

El mapa resultante se lee como un instrumento de medida, y esto es lo que lo hace valioso más allá de la conversación:

Muchas tarjetas rojas (preguntas): la historia no está lista. No hay que estimarla ni meterla en el sprint; hay que buscar respuestas, y quizá un spike.

Muchas tarjetas azules (reglas): la historia es demasiado grande. Cada regla o grupo de reglas es un candidato a historia propia, y el bloque D dice cómo cortarla.

Una regla con muchísimos ejemplos: probablemente no es una regla, sino varias mal enunciadas. Conviene desmenuzarla.

Pocas reglas, ejemplos claros y ninguna pregunta: la historia está lista, y además ya tiene escritos sus criterios de aceptación.

En Vantia, el mapa del asistente fiscal produjo tres reglas (responder con apoyo en la normativa y en los datos del propio usuario; incluir siempre la referencia consultable; reconocer el límite y derivar al gestor cuando no haya apoyo), cinco ejemplos y una pregunta roja que nadie pudo cerrar en la sesión: qué ocurre cuando el autónomo insiste tras una derivación. Esa pregunta se convirtió en el segundo elemento del backlog.

11.5 Variantes que conviene conocer

Feature Mapping (John Ferguson Smart) añade el recorrido del usuario y las tareas, y va bien cuando la historia tiene un flujo con varios pasos. **OOPSI** (Jenny Martin) recorre resultados, salidas, procesos, escenarios y datos de entrada, y sirve para encuadrar un conjunto antes de bajar a las historias. Ambas son primas de Example Mapping y comparten su idea central: descubrir con ejemplos concretos antes de formular.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Encontrar las reglas de negocio de una historia, con sus excepciones, usando la pregunta que localiza los contextos distintos.
- Elegir ejemplos normales, límite y negativos, y distinguir ilustrar de agotar.
- Facilitar un Example Mapping con sus cuatro tipos de tarjeta y su límite de tiempo.
- Leer el mapa resultante y decidir si la historia está lista, hay que partirla o hay que investigar.
- Convertir las reglas y los ejemplos del mapa en criterios de aceptación sin reescribirlos.

Errores y antipatrones frecuentes

Escribir Gherkin en la sesión de descubrimiento. La discusión se desplaza a la sintaxis y las reglas que faltaban no aparecen; la formulación va después.

Reservar la técnica para las historias difíciles. Las que parecen fáciles son las que esconden la regla no dicha, y son las más baratas de mapear.

Ignorar las tarjetas rojas. Una pregunta sin dueño y sin plazo reaparece en mitad del sprint convertida en bloqueo.

Buscar la exhaustividad. Un mapa con treinta ejemplos deja de ser un instrumento de conversación y se convierte en una especificación que nadie revisa.

Para profundizar

· Wynne — Introducing Example Mapping

· Smart — Feature Mapping

· Adzic — Specification by Example (recursos)

· Cucumber — Referencia de Gherkin

BLOQUE C · LA CONVERSACIÓN Y SU CONFIRMACIÓN

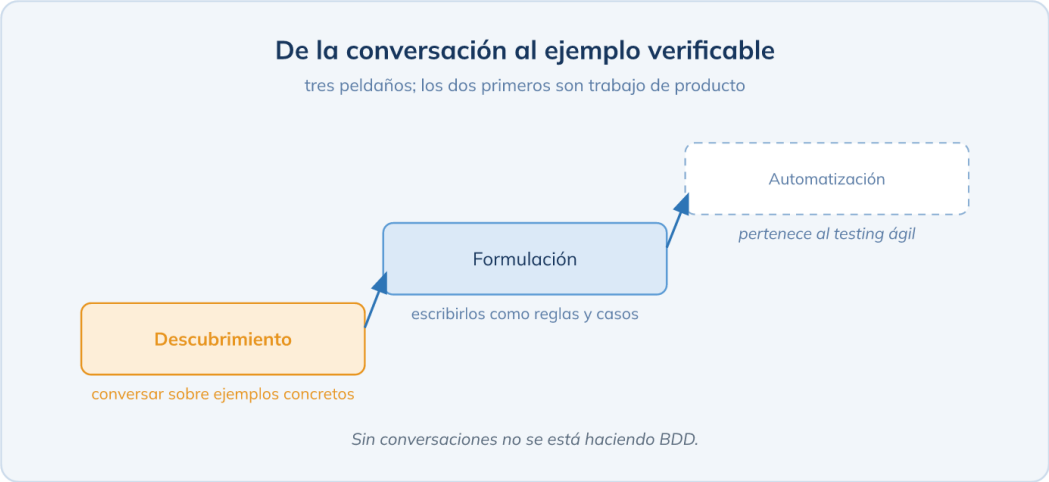
Capítulo 12. De los ejemplos a la verificabilidad: BDD sin herramienta ·

ESTABLECIDO

La escalera de descubrimiento, formulación y automatización, y por qué los dos primeros peldaños son trabajo de producto.

Introducción

El desarrollo guiado por el comportamiento (*behaviour-driven development*, BDD) sufre el malentendido inverso al de las historias: se le confunde con una herramienta de pruebas. En este capítulo recuperamos su parte de producto, que son sus dos primeros peldaños, y dejamos el tercero donde corresponde.



12.1 Tres peldaños y quién sube cada uno

La comunidad de BDD ordena la práctica en tres actividades. **Descubrimiento**: conversar con ejemplos concretos para entender el comportamiento buscado (es el capítulo 11). **Formulación**: escribir esos ejemplos de forma que cualquiera los entienda y sirvan como criterio. **Automatización**: convertirlos en pruebas que se ejecutan.

Los dos primeros peldaños son trabajo de producto y no requieren ninguna herramienta: una pizarra y una conversación bastan. El tercero es trabajo de ingeniería y pertenece al testing ágil, con su propio instrumental. Esta guía enseña los dos primeros y remite el tercero.

El orden importa. Empezar por la automatización produce cientos de escenarios que nadie conversó, cuyo mantenimiento cuesta más de lo que aportan. Es el modo de fallo clásico de BDD.

12.2 Lo que dicen sus autores

Conviene citarlos porque son inequívocos. Liz Keogh: si no se están teniendo conversaciones, no se está haciendo BDD en absoluto. Aslak Hellesøy, creador de Cucumber: BDD no es automatización de pruebas. Dan North, que acuñó el término, lo presentó como una manera de conversar sobre el comportamiento esperado usando ejemplos, y su artículo «What's in a Story?» sigue siendo la mejor introducción breve.

El BoK de Scrum Manager recoge esa lectura y sitúa a North en su origen. La conclusión práctica para el trabajo con historias es que el vocabulario «Dado que / Cuando / Entonces» es útil como forma de ordenar un ejemplo, y su valor no depende de que después se automatice.

12.3 Especificación con ejemplos y documentación viva

Gojko Adzic sistematizó en *Specification by Example* la idea de usar ejemplos acordados como especificación del comportamiento. Su efecto secundario más valioso aparece cuando esos ejemplos se automatizan: la documentación deja de envejecer, porque una descripción que se ejecuta y falla avisa de que ha dejado de ser verdad. Es lo que se llama **documentación viva**.

Para el trabajo de producto, el punto interesante es el anterior: los ejemplos acordados son ya la especificación, estén automatizados o no. Y ahí conecta con el capítulo 13, porque son también lo único que merece la pena conservar de una conversación de refinamiento.

12.4 Escribir escenarios legibles

Cuando se formula un ejemplo como escenario, hay dos estilos y uno funciona mejor. El **estilo imperativo** describe los pasos de la interfaz («cuando hago clic en Nueva factura, cuando escribo 100 en el campo importe, cuando pulso Guardar…»): se rompe con cada cambio de interfaz y no dice qué comportamiento se busca. El **estilo declarativo** describe la intención («cuando emito una factura de 100 euros a un cliente de otro país de la Unión Europea»): sobrevive a los cambios y se lee.

Tres reglas prácticas: pocas líneas por escenario, un solo comportamiento en cada uno, y datos concretos en lugar de descripciones («un cliente intracomunitario» es peor que «un cliente con NIF de Portugal»). Los escenarios así escritos sirven de criterio de aceptación sin reescritura, y son el punto de partida de la verificación mecánica del capítulo 29.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Explicar los tres peldaños del desarrollo guiado por el comportamiento y decir cuáles son trabajo de producto.
- Formular un ejemplo acordado como escenario legible, en estilo declarativo y con datos concretos.
- Distinguir un escenario que sirve de criterio de aceptación de uno que solo describe una interfaz.
- Argumentar por qué la conversación es la condición del método, y no la herramienta.
- Explicar qué es la documentación viva y qué la mantiene verdadera.

Errores y antipatrones frecuentes

Empezar por la herramienta. Instalar el marco de automatización antes de haber conversado produce escenarios sin dueño y una carga de mantenimiento que acaba abandonada.

Escribir escenarios imperativos. Los pasos de interfaz convierten cada rediseño en una tanda de escenarios roto sin que el comportamiento haya cambiado.

Traducir cada criterio a un escenario. No todo criterio necesita esa forma; forzarla añade precondiciones artificiales y alarga la lectura.

Llamar BDD a la automatización. Es el malentendido que sus propios autores llevan quince años corrigiendo, y arrastra al equipo a saltarse el descubrimiento.

Para profundizar

· North — What's in a Story?

· Keogh — A Third Conversational Pattern in BDD

· Rose — Historias de usuario y BDD

· Scrum Manager BoK — Behavior-Driven Development

· Adzic — Specification by Example (recursos)

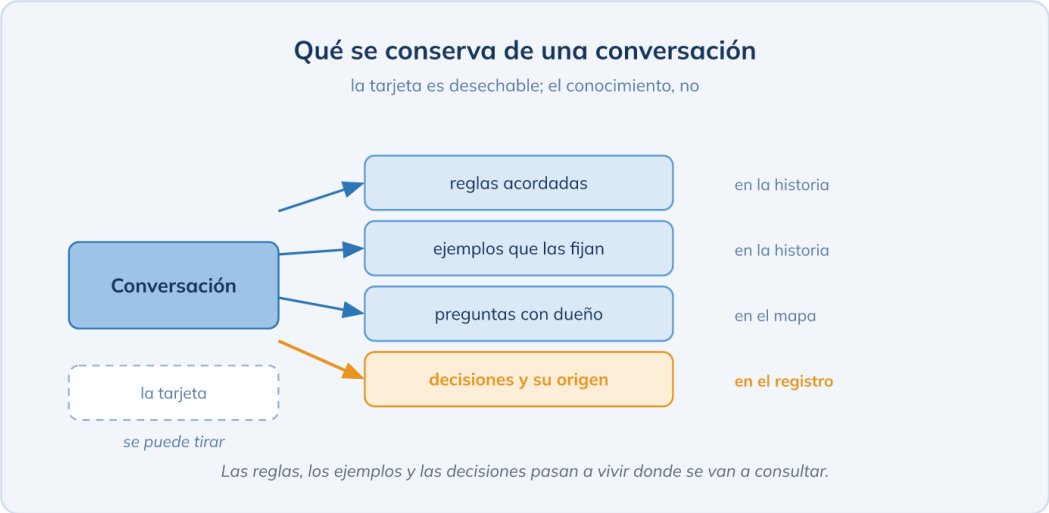
BLOQUE C · LA CONVERSACIÓN Y SU CONFIRMACIÓN

Capítulo 13. Preservar el contexto de la conversación · EN CONSOLIDACIÓN

Qué merece conservarse de un refinamiento, dónde ponerlo y cómo se resuelve la tensión con desechar las historias entregadas.

Introducción

Si el valor de una historia está en la conversación, queda un problema práctico: la conversación se olvida. Seis meses después, nadie recuerda por qué el asistente deriva al gestor en los casos dudosos, y la regla se cuestiona de nuevo o, peor, se cambia sin saber lo que costó acordarla. En este capítulo vemos qué conservar y dónde.



13.1 Cuatro cosas merecen conservarse

Las reglas acordadas. Son conocimiento de dominio y sobreviven a la historia que las descubrió.

Los ejemplos que las fijan. Sin ellos, la regla vuelve a admitir interpretaciones.

Las preguntas abiertas, con dueño. Una pregunta sin responsable es una pregunta perdida.

Las decisiones y su procedencia. Qué se decidió, con qué evidencia, qué alternativas se descartaron y quién responde.

Lo que no merece conservarse es casi todo lo demás: el relato de la sesión, las opiniones intermedias y las versiones descartadas de la redacción.

13.2 Dónde ponerlo

En la propia historia van las reglas y los ejemplos que la confirman, porque ahí es donde se van a leer.

En el mapa de ejemplos queda el registro de la conversación, si se conserva la foto del mapa con sus tarjetas rojas pendientes.

En la documentación viva quedan los ejemplos automatizados, que además avisan cuando dejan de ser verdad (capítulo 12).

En un registro de decisiones de producto van las decisiones que trascienden a una historia. Esta última pieza es una **propuesta de Scrum Manager**, y conviene decirlo: la traemos por analogía con los registros de decisiones de arquitectura, que la ingeniería de software usa desde 2011 y que consisten en media página por decisión, con contexto, opciones, decisión y consecuencias. La guía troncal Product Owner 2.0 incluye un ejemplo completo de ese registro aplicado a producto.

Media página por decisión. El formato importa menos que la disciplina: registrar solo las decisiones que costó tomar, y en el mismo sitio siempre. Un registro que crece sin criterio no se consulta.

13.3 La tensión con desechar las historias entregadas

La tradición ágil recomienda tirar las tarjetas cuando la historia se entrega, y tiene razón: mantener un archivo de historias completadas produce un cementerio que nadie consulta y que da una falsa sensación de trazabilidad.

La tensión se resuelve distinguiendo dos cosas. **La tarjeta es desechable**; era un recordatorio para una conversación que ya ocurrió. **El conocimiento no lo es**: las reglas, los ejemplos y las decisiones pasan a vivir donde se van a consultar, que no es el backlog. Con ese reparto, tirar la historia no pierde nada.

13.4 El contexto que solo existe en un chat

La asistencia de inteligencia artificial ha creado un lugar nuevo donde el contexto se pierde: la conversación privada con el modelo. Quien redacta una historia con ayuda suele construir por el camino un contexto valioso (supuestos que asumió, alternativas que descartó, preguntas que el modelo planteó), y ese contexto no llega al equipo, porque lo que llega es la historia acabada.

La práctica que lo corrige es sencilla y conviene adoptarla desde el principio: llevar al refinamiento **los supuestos y las preguntas**, no solo el resultado. Dicho de otro modo, si la conversación con el modelo produjo cinco decisiones, el equipo debe ver las cinco, aunque el texto final no las mencione. El capítulo 26 trata este asunto como parte de la procedencia.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Decidir qué se conserva de una conversación de refinamiento y qué se descarta.
- Colocar cada pieza conservada en el sitio donde se va a consultar.
- Escribir un registro de decisiones de producto de media página, con su procedencia.
- Explicar por qué la tarjeta es desechable y el conocimiento no.
- Hacer visible al equipo el contexto construido en una conversación privada con un modelo de lenguaje.

Errores y antipatrones frecuentes

Conservar la historia entera «por trazabilidad». Produce un archivo que nadie lee y que sustituye al conocimiento realmente reutilizable, que son las reglas y las decisiones.

Registrar todas las decisiones. Un registro con cientos de entradas deja de consultarse; solo merecen entrada las decisiones que costó tomar y que alguien va a cuestionar.

Dejar las preguntas sin dueño. Una tarjeta roja sin responsable ni fecha vuelve a aparecer en el sprint como bloqueo.

Presentar el resultado sin los supuestos. El equipo hereda decisiones que no sabe que se tomaron, y las descubre cuando alguna resulta equivocada.

Para profundizar

· Adzic — Specification by Example y documentación viva

· Wynne — el mapa de ejemplos como registro de la conversación

· Skill Arena — Product Owner

· Scrum Manager BoK — IA como teammate

BLOQUE D · DIVIDIR MANTENIENDO EL VALOR

Capítulo 14. Por qué, cuánto y cuándo dividir · ESTABLECIDO

La razón de fondo del lote pequeño, el tamaño medido en lugar de heredado, y el momento en que conviene partir.

Introducción

Dividir es la habilidad que más se pide en el trabajo con historias y la que menos se enseña. Antes de ver cómo se hace, conviene tener claro por qué se hace, porque el equipo que divide sin entender la razón acaba produciendo piezas pequeñas que no sirven para nada. En este capítulo tratamos el porqué, el cuánto y el cuándo; los tres capítulos siguientes, el cómo.

14.1 Por qué el lote pequeño

La razón no es la comodidad de la planificación, sino el aprendizaje. Un lote pequeño acorta el tiempo entre decidir y saber si la decisión era buena, y ese ciclo es lo único que convierte la entrega en conocimiento. Donald Reinertsen lo sistematizó para el desarrollo de producto: los lotes grandes acumulan riesgo, retrasan el descubrimiento de los errores y esconden lo que no aporta.

El programa DORA lo sostiene con datos año tras año, y sitúa el trabajo en lotes pequeños entre las capacidades asociadas al rendimiento de las organizaciones. Para el trabajo con historias, la consecuencia práctica son tres beneficios concretos: retroalimentación más temprana, menos riesgo por entrega y la posibilidad de **despriorizar o descartar** una parte cuando resulta que no importa. Ese tercer beneficio es el que más se olvida y el que mejor distingue un corte bueno de uno malo.

La prueba del descarte. Un corte que no permite renunciar a ninguna de sus piezas no ha dividido nada: ha repartido el mismo trabajo en más tarjetas.

14.2 Cuánto: de la banda heredada a la medida propia

Las cifras que circulan son una banda de heurísticas de distintas épocas: el texto original de INVEST hablaba de unas pocas semanas-persona; Cohn y la escuela de Humanizing Work manejan la idea de que quepan entre seis y diez historias en un sprint; SAFe habla de pocos días o menos; y la práctica en equipos de producto con despliegue frecuente ronda de medio día a dos días.

En lugar de adoptar una de esas cifras, conviene sustituirlas por la medida del propio equipo. La comunidad Kanban ha separado explícitamente dos ideas que suelen confundirse: **dimensionar según la expectativa de servicio** (*right-sizing*), que consiste en comprobar si un elemento cabe en el plazo que el equipo se ha comprometido a sostener, e **igualar tamaños** (*same-sizing*), que consiste en trocear todo hasta que mida lo mismo. La primera es útil; la segunda es un ritual.

La medida propia se obtiene mirando lo ya entregado: cuánto tardaron los últimos treinta elementos desde que se empezaron hasta que se terminaron. Con esa distribución, la pregunta ante una historia nueva deja de ser «cuántos puntos tiene» y pasa a ser «¿se parece a las que terminamos en menos de tres días?».

14.3 El límite inferior

Hay un tamaño por debajo del cual dividir empeora las cosas. Cuando la pieza ya no puede enunciar qué cambia para alguien, ha dejado de ser una porción de valor y se ha convertido en un detalle de implementación con formato de historia. La señal es doble: nadie fuera del equipo nota la entrega, y el coste de gestionar la tarjeta (refinar, ordenar, revisar) empieza a parecerse al de hacerla.

Cuando eso pasa, la respuesta es **recombinar**: juntar las piezas en un elemento que sí exprese un cambio observable y volver a dividirlo con otro criterio. Lo veremos en el capítulo 17.

14.4 Épica, tema y funcionalidad: contenedores, no categorías

El vocabulario de tamaños es útil para conversar y peligroso cuando se toma por una taxonomía. Una **épica** es un elemento grande que habrá que partir; un **tema** agrupa elementos relacionados por asunto; una **funcionalidad** (*feature*) es una capacidad del producto que suele contener varias historias. Las herramientas los convierten en tipos con reglas y jerarquías obligatorias, y ahí empieza el problema del capítulo 18.

Cohn lo resume bien: son etiquetas de conveniencia, y la pregunta que importa es si cabe en un ciclo y si expresa valor, más que de qué tipo es el elemento. El BoK de Scrum Manager documenta la jerarquía tema, épica, historia y tarea como convención, y conviene usarla en ese espíritu.

14.5 Cuándo dividir

El momento es **uno o dos ciclos antes** de que la historia llegue a la cima del backlog. Dividir antes es trabajo perdido cuando el objetivo cambia; dividir en la planificación es tarde, porque la conversación que descubre el corte no cabe en ese hueco.

El antipatrón opuesto es dividir todo el backlog por adelantado, que produce cientos de piezas pequeñas ordenadas por comodidad de construcción y sin relación visible con el valor. Es la forma más eficaz de perder el relato del producto, y el bloque E está dedicado a recuperarlo.

Conviene distinguir también dos operaciones que se confunden: **dividir en historias** (partir valor en porciones de valor, que es lo de este bloque) y **descomponer en tareas** dentro del sprint (repartir el trabajo técnico de una historia, que es asunto del equipo de desarrollo). La segunda no produce elementos del backlog.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Explicar con evidencia por qué el lote pequeño mejora el aprendizaje y reduce el riesgo.
- Obtener la medida de tamaño del propio equipo a partir de lo ya entregado, en lugar de adoptar una cifra heredada.
- Distinguir dimensionar según la expectativa de servicio de igualar tamaños.
- Reconocer el límite inferior de la división y recombinar cuando se ha pasado.
- Decidir el momento de dividir y distinguir dividir en historias de descomponer en tareas.

Errores y antipatrones frecuentes

Dividir para que quepa en el sprint. El tamaño es una consecuencia del corte, no su objetivo; cortar por capacidad produce piezas sin valor propio.

Igualar todos los tamaños. Convierte la división en un trámite y desperdicia la información que da la variedad de tamaños reales.

Dividir todo el backlog por adelantado. El detalle caduca, se pierde el relato y se invierte esfuerzo en piezas que nunca se construirán.

Llamar historias a las tareas de una historia. Infla el backlog con elementos sin beneficiario y hace que el orden deje de expresar valor.

Para profundizar

· DORA — Working in small batches

· ProKanban — Right-sizing no es igualar tamaños

· Kanban Guides — Open Guide to Kanban

· Cohn — Épicas, funcionalidades y temas

· Scrum Manager BoK — Epic

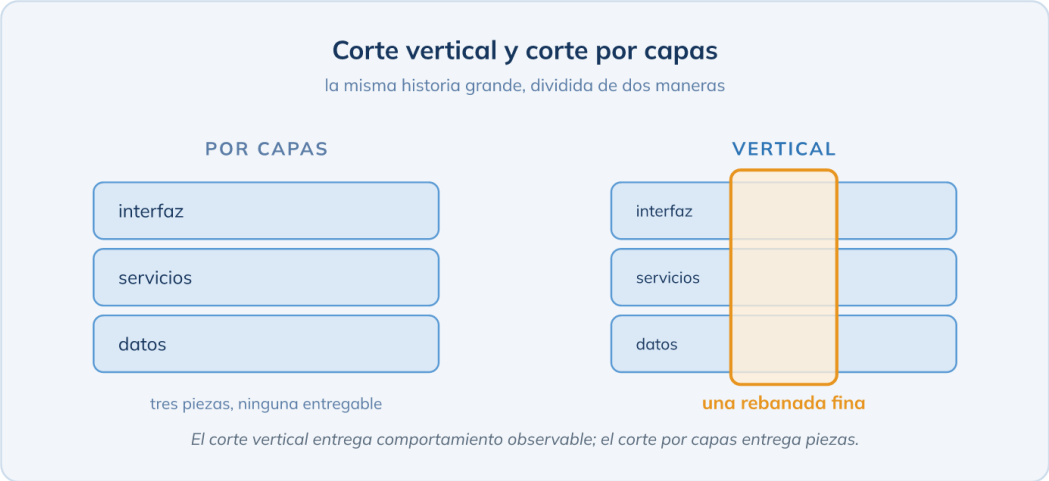
BLOQUE D · DIVIDIR MANTENIENDO EL VALOR

Capítulo 15. Corte vertical y cortes técnicos legítimos · ESTABLECIDO

El corte que atraviesa el sistema frente al corte por capas, y los tres cortes técnicos que sí están justificados.

Introducción

El error de división más extendido tiene nombre y forma reconocible: cortar por capas técnicas. Produce tres piezas (interfaz, servicios, datos) de las que ninguna puede entregarse ni valorarse por separado, y obliga a completar las tres para que alguien note algo. En este capítulo vemos la alternativa y los casos en que un corte técnico sí está justificado.



15.1 El corte vertical

Un **corte vertical** entrega un cambio de comportamiento observable y atraviesa todas las capas que haga falta para conseguirlo: si necesita interfaz, servicio y datos, los tres van dentro. La rebanada puede ser finísima (un solo caso, un solo tipo de dato, un solo camino) y sigue siendo vertical mientras alguien pueda usarla y notarla.

La ventaja no es estética. Una rebanada vertical se puede desplegar, medir, discutir con quien la pidió y descartar; tres capas horizontales solo se pueden mostrar en una demostración interna. El capítulo 4 anticipaba esta propiedad con la letra que Wake añadió a INVEST, la de ser de extremo a extremo.

Cómo se reconoce. Basta una pregunta: ¿alguien de fuera del equipo podría probar esto y decir si le sirve? Si la respuesta es no, el corte es horizontal, aunque la tarjeta tenga plantilla de historia.

15.2 El esqueleto andante y la bala trazadora

Hay una excepción legítima y bien documentada para la primera rebanada de un producto o de una funcionalidad grande. El **esqueleto andante** (*walking skeleton*), de Alistair Cockburn, es una implementación mínima que ejecuta una función de punta a punta, atravesando todos los

componentes con lo mínimo imprescindible. Su valor está en que integra la arquitectura antes de que haya código que reescribir.

La **bala trazadora** (*tracer bullet*), de Hunt y Thomas, es la misma idea desde la perspectiva del código: construir el camino completo con una funcionalidad reducida para comprobar que las piezas encajan y para tener por dónde disparar después.

Gojko Adzic añadió un correctivo práctico que conviene conocer, porque el esqueleto puro suele acabar en una demostración que no sirve a nadie: ponerle muletas. Es decir, completar la rebanada con lo que haga falta (datos cargados a mano, un proceso manual detrás, una pantalla provisional) para que alguien pueda **usarla de verdad** y no solo verla funcionar.

15.3 Los tres cortes técnicos legítimos

El spike. Cuando la incertidumbre es de conocimiento y no de alcance, procede comprar información con un trabajo acotado en tiempo y con una pregunta explícita. Es un último recurso, no un acompañante de cada historia: un spike por cada elemento es señal de que el refinamiento no está haciendo su trabajo.

La historia habilitadora. Trabajo que prepara el terreno para varias historias posteriores (una migración, una integración, un cambio de infraestructura). SAFe la institucionaliza con cuatro tipos, y fuera de SAFe conviene enunciarla en claro, con su beneficiario y su justificación económica.

El esqueleto andante, ya visto, para la primera rebanada de algo grande.

Lo que estos tres tienen en común, y lo que los distingue del corte por capas, es que se enuncian con honestidad: nadie finge que sean valor de usuario. El capítulo 23 desarrolla cómo escribirlos sin usuarios falsos.

15.4 El debate abierto: por dónde empezar

Cuando hay que construir una rebanada vertical, queda una decisión con dos escuelas y sin ganador claro. **Interfaz primero**, con datos simulados o procesos manuales detrás: permite validar con usuarios muy pronto y descubrir que la necesidad era otra, a costa de trabajo que habrá que sustituir. **Servicios primero**, con la funcionalidad oculta tras un conmutador (*feature flag*): asegura los cimientos y evita retrabajo, a costa de tardar más en aprender del usuario.

La elección depende de qué riesgo domine. Si el riesgo mayor es no saber si la solución sirve, conviene empezar por lo que se puede mostrar. Si el riesgo mayor es técnico, conviene empezar por lo que se puede probar. En Vantia, el equipo eligió lo primero de forma extrema: antes de construir el asistente, montó un **Mago de Oz** con gestores respondiendo de noche tras una interfaz de asistente, y en dos semanas obtuvo el catálogo real de consultas, que reordenó el backlog entero.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Distinguir un corte vertical de un corte por capas con la pregunta del usuario que lo prueba.
- Construir la primera rebanada de una funcionalidad grande como esqueleto andante, con las muletas necesarias para que sirva.
- Justificar un spike con su pregunta y su límite de tiempo, y no convertirlo en rutina.

- Enunciar una historia habilitadora con su beneficiario y su razón económica.
- Elegir entre empezar por la interfaz o por los servicios según el riesgo que domine.

Errores y antipatrones frecuentes

Cortar por capas y llamarlo dividir. Es el primero de los cinco errores de división que documenta Cohn: produce piezas que nadie puede valorar y obliga a entregarlas todas.

Entregar el esqueleto sin muletas. Una rebanada que solo se puede demostrar internamente no produce aprendizaje real de usuario.

Un spike por historia. Convierte la investigación en un peaje y esconde que el refinamiento no está resolviendo la incertidumbre.

Historias técnicas con usuario inventado. «Como servidor, quiero…» no comunica nada y hace imposible priorizar; el capítulo 23 da las alternativas honestas.

Para profundizar

· Adzic — Forget the walking skeleton, put it on crutches

· Hunt y Thomas — Tracer bullets and prototypes

· Humanizing Work — Dealing with technical stories

· Cohn — Cinco errores de división

· SAFe — Enablers

BLOQUE D · DIVIDIR MANTENIENDO EL VALOR

Capítulo 16. El catálogo de patrones de división · ESTABLECIDO

Trece patrones con su procedencia y la señal de la historia que sugiere cada uno.

Introducción

Los patrones de división son el instrumental del oficio, y su valor está menos en la lista que en saber qué señal de la historia sugiere cada uno. En este capítulo los recorremos agrupados por esa señal, con su procedencia, y al final los cruzamos con las estrategias del manual básico de Scrum Manager que esta guía sustituye, para que quien venga de ahí encuentre su equivalencia.

16.1 Cuando la historia describe un flujo

Pasos del flujo de trabajo. Una historia que recorre varios pasos («solicitar, revisar, aprobar, notificar») admite partirse por pasos, con una condición que casi todo el mundo incumple: la primera rebanada debe atravesar el flujo completo de la forma más simple posible, y las siguientes van engordando los pasos. Partir por pasos entregando el primero completo produce piezas inservibles, y por eso Humanizing Work tituló su artículo sobre el asunto diciendo que la mayoría lo hace mal.

Caminos e interfaces. Si hay varios caminos alternativos (con tarjeta o por transferencia; desde web o desde móvil), cada camino es una rebanada. Es el patrón más productivo cuando la historia esconde un «o».

16.2 Cuando la historia esconde variedad

Variaciones de reglas de negocio. Una regla con excepciones («salvo para clientes intracomunitarios») admite partirse en la regla general primero y las excepciones después. La señal es la palabra «salvo», «excepto» o «depende de».

Variaciones de datos. Si el comportamiento cambia según el tipo de dato (un tipo de factura, un régimen fiscal, un idioma), cada variedad es una rebanada. La señal son las enumeraciones.

Operaciones. Las cuatro operaciones sobre una entidad (crear, ver, modificar, borrar) rara vez tienen el mismo valor: casi siempre una de ellas resuelve el problema y las otras son comodidad. La señal es el verbo comodín «gestionar».

Segmento de usuarios. Servir primero a un segmento (los autónomos nocturnos de Vantia, y no a todos los usuarios) permite aprender antes con menos alcance. La señal es un usuario descrito en plural genérico.

16.3 Cuando la complejidad no está repartida

Simple primero, complejo después. Se implementa el caso central y se aplazan los casos raros. La señal es un porcentaje: el 80 % de los casos son iguales.

Esfuerzo mayor aislado. Cuando una parte concreta concentra casi todo el trabajo, conviene separarla para poder entregar el resto y decidir después si merece la pena. La señal es una estimación dominada por un solo elemento.

Cualidades del sistema aplazadas. Primero funciona, después funciona rápido (o accesible, o a escala). Aplazar la cualidad es legítimo si se hace explícito y con fecha; el riesgo es que el aplazamiento se convierta en permanente.

Capacidad y volumen. Empezar con un límite (diez facturas, cien usuarios) y ampliarlo después. La señal es un requisito de volumen que multiplica la complejidad.

16.4 Cuando lo que falta es información o valor

Manual antes que automatizado. Resolver primero con una persona detrás y automatizar después, cuando se sabe qué automatizar. Es el patrón del Mago de Oz de Vantia.

Aprender antes que ganar. Adzic lo formula como separar el aprendizaje de la ganancia: si la historia mezcla las dos cosas, conviene entregar primero la parte que enseña algo, aunque no produzca ingresos, y después la que los produce.

Extraer un spike. Cuando la incertidumbre impide dimensionar, se extrae la pregunta como trabajo acotado. Es el último recurso del capítulo 15.

SPIDR, para recordar cinco. Mike Cohn agrupó los cinco patrones que más rinden en un acrónimo cómodo: **S**pike, **P**ath (caminos), **I**nterface (interfaces), **D**ata (datos) y **R**ules (reglas). Sirve como primera pasada antes de recurrir al catálogo completo.

16.5 Equivalencia con el manual básico de Scrum Manager

Quien haya trabajado con el manual básico «Historias de usuario» de Scrum Manager reconocerá aquí sus once estrategias, formuladas con otro nombre. La correspondencia es directa: la división por pasos del flujo de trabajo, por reglas de negocio, por tipos de datos o parámetros, por operaciones y por roles se corresponden con los patrones de 16.1 y 16.2; la división por camino feliz y camino alternativo es un caso de variación de reglas y de caminos; la división por opciones o plataformas de entrada es el patrón de interfaces; la división por casos o escenarios de prueba equivale a variaciones de datos guiadas por los ejemplos del capítulo 11; la de optimizar ahora o más

tarde es el aplazamiento de cualidades; la de compatibilidad de navegador, un caso particular de interfaces; y la estrategia cero, construir un spike, es el patrón de extraer un spike.

El catálogo unificado que presentamos aquí es una **síntesis editorial** sobre patrones establecidos uno a uno, y conviene decirlo con la advertencia del glosario de Agile Alliance: sobre cómo dividir historias hay bastante menos consenso del que parece. Por eso el capítulo siguiente dedica su espacio al criterio de elección, que es lo que de verdad distingue a quien sabe dividir.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Reconocer en una historia la señal que sugiere cada patrón de división.
- Aplicar el patrón de pasos del flujo con una primera rebanada que atraviese el flujo completo.
- Usar SPIDR como primera pasada y recurrir al catálogo completo cuando no basta.
- Explicar la procedencia de los patrones y su grado de consenso.
- Traducir las estrategias del manual básico de Scrum Manager a los patrones canónicos.

Errores y antipatrones frecuentes

Partir por pasos entregando el primero completo. Produce una pieza que no atraviesa el flujo y de la que nadie puede decir si sirve; la primera rebanada debe llegar al final, aunque sea de forma mínima.

Dividir «gestionar» en cuatro operaciones iguales. Casi nunca tienen el mismo valor: conviene entregar la que resuelve el problema y decidir después sobre las demás.

Aplazar cualidades sin fecha. El aplazamiento sin compromiso se convierte en deuda permanente y reaparece como incidencia.

Memorizar el catálogo. Lo que hay que reconocer es la señal en la historia; la lista sin señales produce cortes arbitrarios.

Para profundizar

· Humanizing Work — Guide to Splitting User Stories

· Humanizing Work — Por qué casi todos dividen mal los flujos

· Wake — Twenty Ways to Split Stories

· Cohn — SPIDR

· Agile Alliance — Story Splitting

BLOQUE D · DIVIDIR MANTENIENDO EL VALOR

Capítulo 17. Del patrón al corte: procedimiento, elección y dependencias · EN CONSOLIDACIÓN

El procedimiento completo, las dos reglas que juzgan un corte, el meta-patrón para historias que se resisten y las dependencias que un corte introduce.

Introducción

Conocer los patrones no basta: hay que saber cuál aplicar, cómo juzgar los cortes que salen y qué hacer cuando una historia se resiste a todos. En este capítulo cerramos el bloque con el procedimiento, el criterio de elección y el efecto que un corte tiene sobre las dependencias.

17.1 Antes de partir: preparar la historia

La precondición del procedimiento es contraintuitiva y viene de Humanizing Work: antes de dividir, la historia debe cumplir INVEST **salvo la letra S**, la del tamaño. Es decir, debe ser independiente, negociable, valiosa, estimable y comprobable, y solo fallar por grande.

Cuando falla otra propiedad, dividir no arregla nada. Y la que falla casi siempre es la misma, el **valor**: la historia es grande porque en realidad no expresa una porción de valor, sino un conjunto de trabajo técnico. En ese caso la operación correcta es la inversa, **recombinar** hasta encontrar el cambio observable que el conjunto produce, y dividir ese.

Lo que suele faltar es el valor. Humanizing Work lo dice con veinte años de talleres detrás: cuando una historia se resiste a la división, la propiedad que falta es la V, no la S.

17.2 Aplicar los patrones y generar candidatos

El procedimiento tiene tres movimientos. Primero, **identificar la señal** que la historia ofrece (un flujo, un «salvo», una enumeración, un verbo comodín, un porcentaje). Segundo, **aplicar dos o tres patrones** compatibles con esa señal, sin quedarse en el primero: la práctica muestra que el segundo o tercer corte suele ser mejor que el primero. Tercero, **comparar los candidatos** con los criterios de 17.3.

Es trabajo de grupo y de pizarra, no de herramienta. Cinco minutos con tres personas producen dos o tres cortes candidatos; media hora con la herramienta produce uno, y suele ser el horizontal.

17.3 Juzgar un corte: los criterios y la prueba del falso corte

Un corte candidato se juzga con seis criterios, y no hace falta que los cumpla todos: **conserva valor** (cada pieza expresa un cambio observable), **es verificable** (se puede escribir un ejemplo que la confirme), **permite despriorizar** (alguna pieza podría no hacerse), **da piezas de tamaño parecido**, **enseña algo** (alguna pieza reduce incertidumbre) y **es suficientemente independiente**.

Las dos reglas que más discriminan son la tercera y la cuarta, y conviene aplicarlas siempre. Y hay una prueba que descarta los falsos cortes en un segundo: **si todas las piezas son obligatorias para completar la historia grande, no se ha cortado nada**. Se ha repartido el trabajo en más tarjetas, que es el resultado más frecuente de las divisiones hechas por presión de planificación.

17.4 El meta-patrón para historias que se resisten

Cuando ningún patrón encaja, Humanizing Work propone un procedimiento de tres pasos que funciona con una frecuencia sorprendente: **encontrar la complejidad central** (qué parte concreta hace difícil la historia), **identificar las variaciones** alrededor de esa complejidad, y **reducirlas a una** para la primera rebanada. Lo que era «gestionar la presentación trimestral con todos los regímenes» se convierte en «presentar el trimestre de un autónomo en régimen general», y las demás variaciones se convierten en rebanadas posteriores.

Existe además el **método de la hamburguesa** de Gojko Adzic, especialmente útil cuando el equipo solo sabe pensar en capas: se enumeran las capas técnicas (el pan y los ingredientes), se listan para cada una las opciones de calidad de menor a mayor, y se elige un nivel mínimo de cada capa para formar la primera rebanada completa. Es una manera de usar el pensamiento por capas para llegar a un corte vertical.

El BoK de Scrum Manager documenta este método; conviene saber que su página **invierte los términos vertical y horizontal** respecto a la convención del oficio, y que la corrección está pendiente.

17.5 Ir a lo grande antes de ir a lo pequeño

Hay un caso frecuente en el que la historia «indivisible» resulta ser una tarea: si al intentar partirla todas las piezas son técnicas, probablemente el elemento nunca fue una historia. La salida es subir un nivel, encontrar la funcionalidad de la que forma parte, y dividir esa. Suena costoso y suele ahorrar la mitad del trabajo.

17.6 Las dependencias que introduce un corte

Todo corte crea dependencias, y el de pasos del flujo las crea especialmente: si la rebanada dos necesita la uno, ya no se pueden reordenar. Eso no invalida el corte, pero obliga a tratarlas con la jerarquía del capítulo 7: **hacerlas visibles** (anotarlas en el elemento, no en la cabeza de alguien), **eliminar las evitables** (a veces basta un dato provisional o un conmutador) y **gestionar las inevitables** (ordenar en consecuencia y avisar a quien corresponda).

La razón de tomárselas en serio es de flujo: cada dependencia no gestionada reduce la probabilidad de terminar a tiempo, porque introduce esperas que no dependen del equipo. La comunidad de gestión de flujo insiste en reducirlas antes que gestionarlas, y ese es el orden correcto.

17.7 Practicar: Elephant Carpaccio

La división se aprende dividiendo. El ejercicio canónico es **Elephant Carpaccio**, de Alistair Cockburn, con la guía de facilitación de Henrik Kniberg: partir una funcionalidad pequeña en veinte rebanadas verticales de unos minutos cada una. Humanizing Work sostiene que con unas tres horas de práctica deliberada se alcanza la fluidez, lo que hace de este ejercicio una de las inversiones más rentables de la formación de un equipo.

En Vantia, el ejercicio se hizo con el aviso de plazo fiscal en riesgo, el quinto elemento del backlog. La primera rebanada fue avisar por correo a los autónomos de un solo régimen y un solo plazo, sin panel y sin preferencias, y bastó para descubrir que casi la mitad de los avisos llegaban a quien ya lo tenía presentado, lo que reordenó el resto de las rebanadas.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Comprobar la precondición de INVEST salvo el tamaño antes de dividir, y recombinar cuando lo que falta es el valor.
- Generar dos o tres cortes candidatos a partir de la señal de la historia.
- Juzgar los candidatos con los seis criterios y descartar los falsos cortes con la prueba de la obligatoriedad.
- Aplicar el meta-patrón de complejidad central y variaciones a una historia que se resiste.
- Usar el método de la hamburguesa para llevar a un corte vertical a un equipo que piensa en capas.
- Identificar las dependencias que introduce un corte y tratarlas en el orden adecuado.

Errores y antipatrones frecuentes

Dividir una historia sin valor. Produce piezas que tampoco lo tienen; primero hay que recombinar hasta encontrar el cambio observable.

Quedarse en el primer corte. El segundo o el tercero suelen ser mejores, y compararlos cuesta cinco minutos.

Aceptar un corte con todas las piezas obligatorias. No es una división, es un reparto; nada podrá despriorizarse y el riesgo del lote grande sigue intacto.

Dejar las dependencias en la cabeza de alguien. Reaparecen como esperas en mitad del sprint y nadie las había previsto.

Para profundizar

· Humanizing Work — ¿Historias indivisibles?

· Humanizing Work — Story Splitting Q&A

· Adzic — El método de la hamburguesa

· Cockburn — Elephant Carpaccio

· Kniberg — Guía de facilitación de Elephant Carpaccio

BLOQUE E · EL MAPA DE HISTORIAS Y EL CONTEXTO DEL BACKLOG

Capítulo 18. Lo que se pierde en un backlog plano · EN CONSOLIDACIÓN

Qué información no cabe en una lista ordenada, qué estructuras se han propuesto y por qué el debate sigue abierto.

Introducción

Un backlog bien dividido tiene un efecto secundario incómodo: cuanto mejor se divide, más piezas hay, y más difícil resulta ver de qué producto forman parte. La lista ordenada es un instrumento excelente para decidir qué va primero y un instrumento pésimo para entender el conjunto. En este capítulo vemos exactamente qué se pierde, antes de recuperarlo en el capítulo 19.

18.1 Cuatro cosas que la lista no sostiene

La narrativa. Una lista no dice qué hace una persona con el producto de principio a fin. Con doscientos elementos ordenados por prioridad, nadie puede reconstruir el recorrido de un autónomo de Vantia desde que emite su primera factura hasta que presenta el trimestre.

El contexto. Cada elemento aparece solo, sin la actividad a la que pertenece, así que su importancia hay que recordarla o volver a discutirla.

Las relaciones. Que dos historias sirvan a la misma actividad, o que una sea la variante de otra, no se ve en una lista; se ve cuando alguien lo recuerda.

La composición de una entrega. La pregunta más útil de la planificación de producto (qué conjunto de historias, entregadas juntas, produce un cambio con sentido) no se puede responder mirando una columna ordenada.

El diagnóstico es antiguo y sigue vigente. Jeff Patton lo formuló en 2008 diciendo que el nuevo backlog de historias es un mapa. Casi veinte años después, la mayoría de los equipos sigue trabajando con la columna.

18.2 Las estructuras alternativas propuestas

Las jerarquías de la herramienta. Épica, funcionalidad, historia, subtarea. Es la solución más extendida y su problema es que ordena carpetas: dice de qué tipo es un elemento y no qué hace una persona con el producto.

El tablero de backlog (Roman Pichler): una cuadrícula con temas en un eje y horizonte en otro, que da algo de contexto sin pretender contar el relato.

El árbol de oportunidades (Teresa Torres): objetivo, oportunidades, soluciones y experimentos. No es una estructura del backlog, sino del descubrimiento, y es la mejor manera de justificar por qué un elemento existe.

El mapa de impacto (Gojko Adzic): objetivo, actores, impactos y entregables, que conecta el trabajo con el resultado buscado.

El backlog de oportunidades (Marty Cagan): separar las oportunidades por explorar del trabajo comprometido, para que no compitan en la misma lista.

18.3 La defensa de la lista plana

Conviene tomar en serio el argumento contrario, porque es sólido. Quien defiende la lista plana con etiquetas sostiene que la jerarquía introduce trabajo de mantenimiento, invita a planificar de arriba abajo y hace que los elementos hereden importancia de su contenedor en lugar de ganarla por su valor. Añade que ordenar una lista obliga a decidir, mientras un árbol permite aplazar la decisión indefinidamente.

La posición de esta guía es que las dos cosas son compatibles y sirven a propósitos distintos: **el orden se decide en una lista** (una sola dimensión, sin empates) y **el contexto se ve en un mapa** (dos dimensiones, relato y detalle). Lo que no funciona es pedirle a una estructura las dos cosas.

18.4 Qué hacer si la herramienta obliga

En la mayoría de las organizaciones, la herramienta ya impone una jerarquía y no se puede cambiar. Tres medidas prácticas evitan lo peor. La primera, **no confundir el contenedor con el valor**: una épica no es prioritaria porque contenga muchas historias. La segunda, **mantener el mapa fuera de la herramienta** si dentro no cabe, con la condición de actualizarlo (capítulo 20). La tercera, **usar etiquetas para el relato** (la actividad a la que pertenece cada elemento) en lugar de crear niveles nuevos.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Enunciar qué información se pierde en un backlog plano y ponerlo en un ejemplo real.
- Situar las estructuras alternativas y decir a qué pregunta responde cada una.
- Sostener el argumento a favor de la lista plana y explicar el reparto entre lista y mapa.
- Convivir con la jerarquía impuesta por una herramienta sin dejar que el contenedor decida el valor.

Errores y antipatrones frecuentes

Priorizar por contenedor. Una épica grande arrastra elementos sin valor propio y desplaza a otros que sí lo tienen.

Sustituir la lista por el árbol. El árbol permite no decidir; el orden de una sola dimensión obliga a hacerlo, y esa obligación es su utilidad.

Crear niveles nuevos en la herramienta. Cada nivel añade mantenimiento y da la ilusión de estructura sin aportar relato.

Para profundizar

· Patton — The new user story backlog is a map

· Bastow — Por qué la jerarquía del backlog está rota

· Hinshelwood — ¿Debería tu backlog ser plano?

· Pichler — El tablero de backlog

· Cagan — El backlog de oportunidades

BLOQUE E · EL MAPA DE HISTORIAS Y EL CONTEXTO DEL BACKLOG

Capítulo 19. El mapa de historias: anatomía y construcción · ESTABLECIDO

Las piezas del mapa con el vocabulario de Patton, los cinco pasos para construirlo y los errores que documenta su autor.

Introducción

El mapa de historias (*user story mapping*) recupera en dos dimensiones lo que la lista pierde: en el eje horizontal, el relato de lo que hace una persona; en el vertical, el detalle por orden de necesidad. Es la técnica de este bloque y una de las más rentables del oficio, porque se aprende en una sesión y se usa durante años.

19.1 Las piezas y su vocabulario

Jeff Patton fijó el vocabulario y conviene usarlo con precisión. Los **usuarios** son quienes hacen cosas con el producto. Las **actividades** son los grandes bloques de lo que hacen (facturar, anotar gastos, presentar impuestos). Las **tareas de usuario** son pasos concretos, y Patton insiste en que se escriban como **frases verbales cortas** («emitir factura», «subir ticket»), no como sustantivos ni como historias completas.

Las actividades y las tareas principales, colocadas en su orden narrativo, forman la **espina dorsal** (*backbone*) del mapa: la línea superior que se lee de izquierda a derecha y cuenta el relato completo. Debajo de cada tarea cuelgan las **costillas**, el detalle y las variantes, ordenadas de arriba abajo por necesidad.

Hay una distinción más, los **niveles de objetivo**: no todas las tareas están al mismo nivel de granularidad, y el mapa admite subir o bajar según lo que se quiera ver. Confundir niveles es la causa más común de mapas ilegibles.

El mapa no es un diagrama de flujo. El eje horizontal es narrativo, no temporal estricto: cuenta la historia de uso en el orden en que se entiende mejor, no la secuencia exacta de pulsaciones.

19.2 El mapa del ahora y el mapa del después

Patton distingue dos usos con consecuencias muy distintas. El **mapa del ahora** documenta cómo se hace hoy el trabajo, con producto o sin él, y se construye con datos: entrevistas, analítica, observación. Sirve para encontrar dónde duele.

El **mapa del después** describe cómo será, y es una **hipótesis** hasta que se libera y se mide. Tratarlo como un plan es el error que convierte el mapa en un documento de proyecto: se dibuja entero, se aprueba y se ejecuta, y para cuando llega la primera entrega nadie recuerda que era una conjetura.

En Vantia, el mapa del ahora del trabajo nocturno mostró algo que el backlog escondía: el autónomo que se atasca a las once de la noche no abandona la plataforma en ese momento, sino dos o tres intentos después, y en la mitad de los casos el atasco está en la clasificación de un gasto, no en la factura. Ese dato reordenó el mapa del después.

19.3 Los cinco pasos para construirlo

Encuadrar. Acordar de qué producto, de qué usuarios y de qué problema se está hablando, y qué queda fuera. Sin este paso, el mapa crece sin límite.

Mapear el panorama. Recorrer la actividad completa «una milla de ancho y una pulgada de fondo»: todas las tareas principales, ninguna en detalle. Aquí se descubre lo que faltaba.

Explorar. Bajar al detalle solo donde hay incertidumbre o valor: variantes, casos límite, usuarios distintos, lo que puede fallar.

Cortar entregas viables. Trazar líneas horizontales que separan conjuntos entregables, cada uno con su resultado esperado (capítulo 20).

Cortar la estrategia de desarrollo. Dentro de la primera entrega, decidir por dónde empezar, normalmente con un esqueleto que atraviese el relato (capítulo 15).

La facilitación sigue las reglas del capítulo 9: grupo pequeño con las perspectivas necesarias, trabajo individual antes de hablar, y límite de tiempo por fase. Dos horas bastan para un primer mapa útil.

19.4 Descomposición progresiva y producto existente

El mapa se refina donde toca, igual que el backlog: mucho detalle en la primera entrega, casi ninguno en las siguientes. Un mapa con el mismo nivel de detalle de punta a punta es un mapa que ha consumido esfuerzo en lo que quizá no se haga.

En un producto existente hay un atajo que ahorra semanas: **mapear solo lo nuevo** y el tramo del relato que se ve afectado, en lugar de reconstruir el mapa completo del producto. El mapa exhaustivo de un producto maduro es un ejercicio caro cuyo resultado envejece antes de terminarse.

19.5 Los cinco errores que documenta Patton

Perder el relato. El mapa se convierte en una lista de funcionalidades agrupadas, sin recorrido que se pueda leer.

Ahogarse en el detalle. El grupo baja al detalle en la primera actividad y no llega al final del panorama.

Obsesionarse con las variantes. Se recogen todas las alternativas posibles antes de saber cuáles importan.

Mapear todo el producto. Un mapa de todo, siempre desactualizado, en lugar de un mapa del trabajo que se tiene delante.

Cortar entregas sin resultado. Rebanadas horizontales que agrupan lo que cabe en el equipo, sin decir qué cambia para nadie. Es el asunto del capítulo siguiente.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Construir un mapa de historias con el vocabulario correcto, con su espina dorsal y sus costillas.
- Distinguir el mapa del ahora del mapa del después, y tratar el segundo como hipótesis.
- Facilitar los cinco pasos, con el panorama antes del detalle.
- Refinar el mapa por proximidad y mapear solo lo nuevo en un producto existente.
- Reconocer los cinco errores frecuentes en un mapa ya construido.

Errores y antipatrones frecuentes

Escribir historias completas en la espina dorsal. La línea narrativa se lee de un vistazo con frases verbales cortas; con historias enteras deja de leerse.

Mapear todo el producto «para tenerlo». El coste es alto, el resultado envejece y nadie lo consulta; conviene mapear el trabajo que se tiene delante.

Mezclar niveles de objetivo. Tareas de granularidad muy distinta en la misma fila hacen el mapa ilegible y falsean el relato.

Tratar el mapa del después como un plan aprobado. Es una hipótesis; convertirlo en compromiso reintroduce el proyecto en el trabajo de producto.

Para profundizar

· Patton — Story Map Concepts (referencia rápida)

· Patton — The new user story backlog is a map

· Patton — Five common story mapping mistakes

· Scrum Manager BoK — Mapa de historias

BLOQUE E · EL MAPA DE HISTORIAS Y EL CONTEXTO DEL BACKLOG

Capítulo 20. Cortar por resultado y mantener el mapa vivo · EN CONSOLIDACIÓN

Los tres cortes del mapa, cómo se reconoce un corte hecho por capacidad y qué mantiene el mapa útil después del taller.

Introducción

Un mapa sirve para decidir qué se entrega junto, y ahí es donde la mayoría de los equipos vuelve a las viejas costumbres: agrupan lo que cabe en el próximo sprint y llaman a eso una entrega. En este capítulo vemos los tres cortes que Patton distingue, con su criterio, y qué hace falta para que el mapa siga siendo verdad dentro de tres meses.

20.1 Los tres cortes

La entrega con resultado. Un conjunto de historias que, entregadas juntas, producen un cambio que puede enunciarse y medirse. La condición es doble: un resultado esperado y una métrica que lo compruebe. En Vantia, la primera entrega del asistente fue «los autónomos que se atascan de noche obtienen una respuesta útil en el momento», con el abandono nocturno como medida.

El experimento o producto mínimo viable. Una rebanada más pequeña cuyo objetivo no es entregar valor sostenido, sino comprobar una hipótesis. Patton define el producto mínimo viable de una forma que conviene citar porque desplaza el criterio: el producto más pequeño que alcanza los resultados buscados. No el más pequeño que se puede construir.

La estrategia de desarrollo. Dentro de la primera entrega, el orden de construcción, que normalmente empieza por un esqueleto que atraviesa el relato completo (capítulo 15) y luego engorda por donde más se aprende.

Tres cortes, tres preguntas. La entrega responde a «¿qué cambia para alguien?»; el experimento, a «¿qué queremos averiguar?»; la estrategia, a «¿por dónde empezamos?». Confundirlas produce entregas que no cambian nada y experimentos que tardan un trimestre.

20.2 Cómo se reconoce un corte por capacidad

El corte hecho por capacidad del equipo tiene una señal inequívoca: **no se puede enunciar qué cambia para nadie**. Si al preguntar «¿qué será distinto cuando esto se entregue?» la respuesta habla de lo construido («estará el módulo de gastos») y no de lo que alguien podrá hacer, el corte es por capacidad.

La segunda señal es que la rebanada contiene piezas de actividades distintas sin relación entre sí, elegidas porque encajaban en el hueco. La tercera, que no hay métrica asociada, o que la métrica es de avance («porcentaje de historias completadas») y no de resultado.

Corregirlo no obliga a rehacer el mapa: basta preguntarse, para cada rebanada, qué usuario notará qué cosa, y mover las piezas hasta que la respuesta exista.

20.3 Del mapa a la conversación y a los criterios

El mapa no sustituye a nada de lo visto en el bloque C: lo alimenta. Cada tarea de la primera entrega es candidata a historia, y cada historia entra al taller del capítulo 9 o al Example Mapping del capítulo 11, donde adquiere sus reglas, sus ejemplos y sus criterios.

El recorrido completo, entonces, es este: el descubrimiento produce oportunidades; el mapa las ordena en un relato y corta entregas con resultado; la conversación convierte cada tarea en una historia con sus ejemplos; y la división ajusta el tamaño donde hace falta. Cada pieza de esta guía ocupa un lugar en esa cadena.

20.4 Mantener el mapa vivo

Un mapa que no se actualiza deja de ser un instrumento y se convierte en decoración. Tres prácticas lo mantienen útil. La primera, **actualizarlo con lo aprendido** después de cada entrega: lo que se descubrió, lo que se descartó, lo que apareció. La segunda, **usarlo en las conversaciones** donde se decide (la revisión con stakeholders, el refinamiento del siguiente tramo); un mapa que no se abre no se mantiene. La tercera, **dejar visible su fecha y su estado**, distinguiendo lo documentado de lo hipotético.

Cuando el mapa se abandona tras el taller inicial, lo que ocurre es que el equipo vuelve a la lista plana con el coste del taller ya pagado. Es el modo de fallo más frecuente y el asunto del capítulo 21.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Cortar un mapa en entregas con su resultado esperado y su métrica.
- Distinguir la entrega con resultado, el experimento y la estrategia de desarrollo.
- Aplicar la definición de producto mínimo viable por resultados, y no por tamaño de lo construido.
- Reconocer un corte hecho por capacidad con sus tres señales y corregirlo.
- Mantener el mapa vivo con las tres prácticas, y distinguir en él lo documentado de lo hipotético.

Errores y antipatrones frecuentes

Llamar entrega a lo que cabe en el sprint. Sin resultado enunciable, la rebanada no permite decidir si mereció la pena, y la revisión se convierte en un recuento de trabajo.

Definir el producto mínimo viable por su tamaño. El criterio es el resultado buscado; el «más pequeño que se puede construir» suele no alcanzar ninguno.

Medir la entrega con métricas de avance. El porcentaje completado no dice nada sobre el cambio producido en quien usa el producto.

Abandonar el mapa tras el taller. El coste del taller ya está pagado; no actualizarlo desperdicia la inversión y devuelve al equipo a la lista plana.

Para profundizar

· Patton — It's All in How You Slice It

· Kniberg — Making sense of MVP

· Patton — Dual Track Development is not Duel Track

· Torres — Resultados frente a salidas

BLOQUE E · EL MAPA DE HISTORIAS Y EL CONTEXTO DEL BACKLOG

Capítulo 21. Mapas vecinos y el mapa dentro de la herramienta · EN CONSOLIDACIÓN

Cuatro técnicas que se confunden, cómo se encadenan, y los tres modos de fallo al llevar el mapa a una herramienta.

Introducción

Alrededor del mapa de historias hay tres técnicas con nombre parecido y propósito distinto, y elegir mal cuesta una sesión entera. En este capítulo las distinguimos, mostramos cómo se encadenan y cerramos el bloque con lo que ocurre cuando el mapa entra en una herramienta de gestión.

21.1 Cuatro técnicas, cuatro preguntas

Mapa de historias. Pregunta: ¿qué hace una persona con el producto, de principio a fin, y qué conjunto de eso entregamos primero? Su material son tareas de usuario.

Mapa del recorrido del usuario (*journey map*). Pregunta: ¿cómo vive una persona ese recorrido, con sus emociones, sus puntos de dolor y sus canales? Su material son etapas y experiencias, y su uso natural es el descubrimiento.

Mapa de impacto (*impact mapping*). Pregunta: ¿qué comportamientos de qué actores nos acercan al objetivo, y qué podríamos construir para provocarlos? Su material son objetivo, actores, impactos y entregables.

Árbol de oportunidades (*opportunity solution tree*). Pregunta: ¿qué oportunidades hay bajo este objetivo y qué soluciones podrían atenderlas? Su material son oportunidades halladas en la investigación.

La confusión típica. Usar el mapa de historias para hacer descubrimiento (y quedarse sin datos) o el mapa del recorrido para planificar entregas (y quedarse sin tareas). Cada uno responde bien a su pregunta.

21.2 Cómo se encadenan

El encadenamiento natural va del descubrimiento a la entrega. La investigación con clientes produce oportunidades, que el **árbol** ordena bajo el objetivo. El **mapa de impacto** conecta el objetivo con los comportamientos que hay que provocar. El **recorrido del usuario** muestra cómo se vive hoy la actividad y dónde duele. Y con eso, el **mapa de historias** organiza el relato y corta la primera entrega.

Las dos primeras técnicas pertenecen al descubrimiento y tienen su propia skill en Scrum Manager, igual que la investigación con clientes; aquí se citan para saber cuándo cambiar de herramienta.

Hay un vecino más que conviene conocer: el **Product Backlog Building** de Paulo Caroli, que produce un backlog inicial a partir de una inception. Es el punto de contacto con la skill de Agile Inception, y su material de salida es exactamente lo que este bloque enseña a estructurar.

21.3 El mapa dentro de la herramienta: tres modos de fallo

La jerarquía sustituye a los dos ejes. La herramienta ofrece épica, funcionalidad e historia, y el equipo cree que eso es un mapa. No lo es: una jerarquía tiene un solo eje (contenencia) y el mapa tiene dos (relato y detalle). Se pierde precisamente lo que se quería recuperar.

El mapa gigante sincronizado. Con una extensión que sincroniza el mapa con el backlog, aparece la tentación de mapear el producto entero. El resultado es un mural de cientos de tarjetas que nadie lee y cuya actualización consume horas.

El mapa físico abandonado. Se hace el taller con notas en la pared, se sacan fotos y no se vuelve a tocar. A las pocas semanas el mapa contradice al backlog y nadie sabe cuál manda.

Las tres se evitan con la misma decisión: elegir **un solo lugar** donde el mapa es verdad, mantenerlo del tamaño del trabajo que se tiene delante y actualizarlo en las conversaciones donde se decide.

21.4 Con inteligencia artificial

Las herramientas de mapeo ofrecen ya generar el mapa a partir de una descripción del producto, y lo hacen con soltura: producen actividades y tareas plausibles en segundos. El problema es el del bloque G en su forma más pura: un mapa del **ahora** generado sin datos no documenta nada, porque no procede de ninguna observación; y un mapa del **después** generado sin evidencia es una hipótesis de la máquina, no del equipo.

El uso que sí rinde es el simétrico al del capítulo 25: pedirle que **critique** un mapa hecho por el equipo. Encuentra huecos en el relato, tareas que faltan entre dos pasos, variantes no consideradas y rebanadas sin resultado enunciable, y todo eso es material excelente para la siguiente sesión.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Elegir entre mapa de historias, recorrido del usuario, mapa de impacto y árbol de oportunidades según la pregunta que hay que responder.
- Encadenar las técnicas del descubrimiento a la entrega y saber cuándo cambiar de una a otra.
- Reconocer los tres modos de fallo del mapa en una herramienta y decidir dónde vive la verdad.
- Usar la asistencia para criticar un mapa propio en lugar de para generarlo sin datos.

Errores y antipatrones frecuentes

Confundir la jerarquía con el mapa. Se pierde el relato, que es justamente lo que el mapa aporta sobre la lista.

Mapear el producto entero en la herramienta. El mural resultante no se lee y su mantenimiento compite con el trabajo real.

Dos fuentes de verdad. Un mapa en la pared y un backlog en la herramienta divergen en semanas y obligan a decidir cuál manda en el peor momento.

Generar el mapa del ahora con un modelo. El mapa del ahora se documenta con datos; generado, solo refleja lo que el modelo ha visto en otros productos.

Para profundizar

· Adzic — Impact Mapping

· Nielsen Norman Group — Journey Mapping 101

· Torres — Opportunity Solution Trees

· Caroli — Product Backlog Building

· Skill Arena — Agile Inception

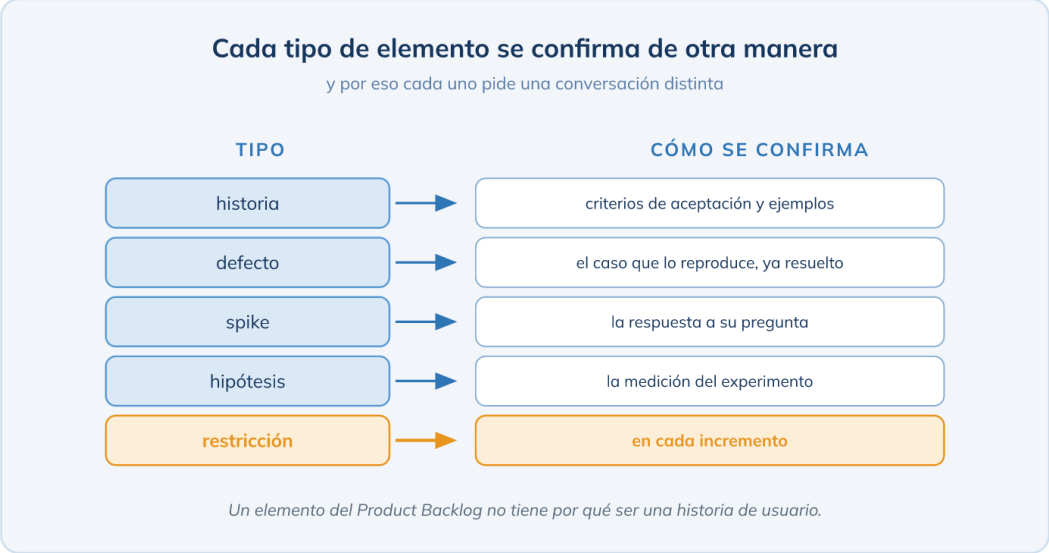
BLOQUE F · NO TODO ES UNA HISTORIA

Capítulo 22. Elemento del Product Backlog frente a historia · ESTABLECIDO

Qué tipos de elemento conviven en un backlog real y cómo se confirma cada uno.

Introducción

Un backlog real no contiene solo historias, y nunca lo hizo. Contiene defectos, investigaciones, trabajo técnico, restricciones, obligaciones normativas e hipótesis por validar. Forzar todo eso al molde de historia produce las ficciones del capítulo 6. En este capítulo ordenamos el catálogo de tipos y, sobre todo, el modo de confirmar cada uno.



22.1 Lo único que Scrum pide

Conviene repetirlo aquí porque es la base del bloque: un elemento del Product Backlog necesita **descripción, orden y tamaño**. La Guía de Scrum de 2017 enumeraba tipos (funcionalidades, funciones, requisitos, mejoras y correcciones) y la versión de 2020 eliminó esa lista, dejando la forma en manos del equipo.

La guía troncal Product Owner 2.0 de Scrum Manager lo enuncia en la misma dirección: un elemento del backlog no tiene por qué ser una historia de usuario; la historia es una herramienta, no una obligación. Esa frase resume el bloque entero.

22.2 El catálogo de tipos y su confirmación

Historia de usuario. Se confirma con criterios de aceptación y sus ejemplos (capítulos 10 y 11).

Defecto. Se confirma reproduciendo el fallo y comprobando que deja de ocurrir. Necesita el caso que lo reproduce, no una plantilla de historia.

Spike. Se confirma con la respuesta a su pregunta y con la decisión que permite tomar. Necesita pregunta explícita y límite de tiempo, y no produce funcionalidad.

Trabajo técnico o habilitador. Se confirma con el criterio técnico acordado y con la constatación de que habilita lo que decía habilitar (capítulo 23).

Restricción o requisito no funcional. Se confirma **en cada incremento**, no una vez: forma parte de la Definition of Done o de una restricción transversal (capítulo 10).

Cumplimiento normativo. Se confirma con la evidencia que exige la norma, que suele incluir trazabilidad y aprobación de un responsable identificable.

Hipótesis con su resultado. Se confirma con la medición del experimento, y su resultado puede ser descartarla. El instrumental completo pertenece a la formación troncal de Product Owner.

Por qué importa el modo de confirmación. Es lo que decide qué conversación hace falta. Un spike pide acordar la pregunta; una restricción, acordar el umbral; una hipótesis, acordar qué medida la refuta. Tratarlos todos como historias hace que la conversación equivocada se celebre con todos.

22.3 Todos en la misma lista, ordenados por valor

Que los tipos se confirmen de forma distinta no significa que vivan en listas separadas. Compiten por el mismo tiempo del equipo, y por tanto se ordenan juntos. Un defecto que hace abandonar a los usuarios va antes que una funcionalidad nueva, y una investigación que desbloquea tres historias puede ir antes que las tres.

El fragmento del backlog de Vantia lo ilustra: junto a las historias del asistente conviven la derivación al gestor (que salió de una conversación de refinamiento), el refuerzo de dos materias débiles (que salió del análisis de errores del piloto) y un panel para gestores (que pidió un stakeholder). Cada uno se confirma de una manera y todos están en la misma columna, con una anotación de su procedencia.

22.4 Qué exige el vocabulario del BoK

El BoK de Scrum Manager mantiene páginas para las piezas de este catálogo (pila del producto, requisitos ágiles, unidad de trabajo, tarea, épica, tema) y conviene usar su léxico en español al escribir para la organización. Dos precisiones que ayudan: **unidad de trabajo** es el término genérico cuando no se quiere prejuzgar el tipo, y **tarea** designa el trabajo técnico interno de una historia, no un elemento del backlog.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Enumerar los tipos de elemento que conviven en un backlog y decir cómo se confirma cada uno.
- Decidir el tipo de un elemento antes de decidir su formato.
- Ordenar tipos distintos en una misma lista por su valor, sin crear listas paralelas.
- Usar el léxico del BoK de Scrum Manager al escribir para la organización.

Errores y antipatrones frecuentes

Convertir cada defecto en historia. El defecto se confirma reproduciendo el fallo; envolverlo en «como usuario, quiero que no falle» añade ruido y esconde el caso que lo reproduce.

Spikes sin pregunta. Un spike sin pregunta explícita se convierte en tiempo de investigación indefinida sin decisión asociada.

Listas paralelas por tipo. Si los tipos no compiten en la misma lista, el orden deja de reflejar el valor y las decisiones se toman por separado.

Confirmar una restricción una sola vez. El rendimiento o la accesibilidad se degradan con cada cambio; su sitio es la Definition of Done, verificada en cada incremento.

Para profundizar

· Guía de Scrum 2020

· Scrum.org — Mito 4: el backlog no tiene que ser historias

· Scrum Manager BoK — Pila del producto

· Scrum Manager BoK — Requisitos ágiles

· Skill Arena — Product Owner

BLOQUE F · NO TODO ES UNA HISTORIA

Capítulo 23. Trabajo técnico sin usuario falso · ESTABLECIDO

Las tres salidas honestas para el trabajo sin usuario visible, y cómo se distingue la historia técnica de la tarea disfrazada.

Introducción

«Como servidor, quiero un índice en la tabla de facturas, para responder más rápido.» Esta frase, o alguna parecida, existe en casi todos los backlog del mundo, y no comunica nada. En este capítulo vemos las tres maneras honestas de tratar el trabajo que no tiene un usuario visible, y cómo se decide entre ellas.

23.1 Por qué aparece el usuario falso

El usuario falso no es un descuido de redacción: es la consecuencia de una norma mal entendida. Cuando la organización exige que todo elemento del backlog tenga forma de historia, el equipo cumple la forma y renuncia al contenido. La responsabilidad, entonces, es de la norma.

El daño es doble. Por un lado, la historia no permite priorizar, porque no dice a quién sirve ni cuánto vale. Por otro, oculta la conversación que sí hacía falta, que era técnica y con criterio técnico.

23.2 Las tres salidas honestas

Encontrar el beneficiario real y su valor. Muchas veces existe y no se ha buscado. El índice de la tabla de facturas no sirve al servidor: sirve al autónomo que espera cuatro segundos cada vez que abre su listado, y el valor se puede enunciar en abandono o en consultas al soporte. Cuando el beneficiario aparece, el elemento vuelve a ser una historia legítima.

Enunciarlo en claro como trabajo técnico. Si no hay beneficiario distinguible, se escribe lo que es, con su justificación económica: «reindexar el almacén de facturas para sostener el crecimiento previsto del próximo año». Cabe llamarlo elemento técnico, funcionalidad al estilo de FDD, spike o habilitador, según el caso, y ninguna de esas formas necesita plantilla.

Convertirlo en restricción o criterio de la Definition of Done. Cuando el trabajo expresa una cualidad que debe cumplirse siempre (tiempo de respuesta, accesibilidad, cifrado), su sitio no es un elemento del backlog que se hace una vez, sino un criterio que se verifica en cada incremento.

La salida deshonesta. Inventar un usuario («como sistema…», «como base de datos…») cumple el formulario y desactiva las tres decisiones que había que tomar: quién se beneficia, cuánto vale y cómo se comprueba.

23.3 Historia técnica honesta frente a tarea disfrazada

La distinción es la que más se usa en el refinamiento, y tiene una prueba sencilla. Una **historia técnica honesta** produce un cambio en el sistema que alguien identificable necesita y que se puede priorizar frente a otras cosas: se puede posponer, se puede descartar y se puede explicar a quien no es técnico. Una **tarea disfrazada** es un paso necesario de otra cosa: no se puede posponer sin bloquear la historia a la que sirve, y su sitio es dentro de ella.

Humanizing Work publica el criterio con ejemplos y añade una guía útil: si el elemento no tiene sentido fuera del contexto de otro elemento, es una tarea. La consecuencia práctica es que el backlog se limpia mucho al aplicar esta prueba: buena parte de los elementos técnicos que lo abarrotan son tareas de historias que ya están en la lista.

23.4 Deuda técnica y su priorización

La deuda técnica merece una mención propia porque su tratamiento como elemento del backlog es una fuente conocida de discusiones. Enunciada como trabajo técnico sin beneficiario ni consecuencia, siempre pierde frente a una funcionalidad nueva. Enunciada con su efecto observable (cuánto ralentiza cada cambio, cuántas incidencias produce, qué riesgo introduce), compite en igualdad.

La formulación que funciona es la del **interés**: en lugar de «hay que refactorizar el módulo de facturación», «cada cambio en facturación cuesta el doble y produce una incidencia de cada tres, y eso seguirá así mientras no se reordene». Con esa formulación, la decisión es de producto y no una concesión al equipo técnico.

23.5 El matiz de los equipos de plataforma

Hay un caso donde el «usuario técnico» no es una ficción: los equipos cuyo producto es una plataforma interna. Ahí quien desarrolla es un cliente real, con necesidades reales y alternativas (puede construirse su propia solución o usar otra), y las historias con desarrolladores como usuarios son legítimas y necesarias.

La literatura de topologías de equipos ha ordenado ese caso, y su criterio para la plataforma es exigente: debe ser la **plataforma viable más pequeña** que resuelve el problema de sus equipos usuarios, con una relación de servicio explícita. En ese contexto, todo lo que enseña esta guía se aplica igual, con el desarrollador como usuario y su experiencia como valor.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Elegir entre las tres salidas honestas ante un elemento sin usuario visible.
- Buscar el beneficiario real de un trabajo técnico y enunciar su valor en términos observables.
- Distinguir una historia técnica honesta de una tarea disfrazada con la prueba del contexto.
- Formular la deuda técnica con su interés, de modo que compita en igualdad en el orden del backlog.
- Reconocer el caso de los equipos de plataforma, donde el desarrollador es un usuario real.

Errores y antipatrones frecuentes

Inventar un usuario técnico. Cumple el formulario y desactiva la priorización; además impide la conversación técnica que sí hacía falta.

Meter las tareas en el backlog. Infla la lista con elementos que no se pueden posponer y hace que el orden deje de significar nada.

Pedir un porcentaje fijo para deuda técnica. Convierte una decisión de producto en una cuota negociada; con el interés bien enunciado, la deuda gana su sitio por valor.

Tratar la plataforma interna como trabajo sin cliente. Sus usuarios son identificables y exigentes, y su experiencia se puede medir igual que la de cualquier usuario final.

Para profundizar

· Humanizing Work — Dealing with technical stories

· SAFe — Enablers

· Cohn — No todo tiene que ser una historia: funcionalidades de FDD

· Team Topologies — conceptos clave

· Scrum Manager BoK — Unidad de trabajo

BLOQUE F · NO TODO ES UNA HISTORIA

Capítulo 24. Representaciones alternativas y cómo elegir el formato · EN CONSOLIDACIÓN

Los formatos disponibles con su autor y su uso, y las cinco preguntas que deciden cuál conviene.

Introducción

Este capítulo es el que cierra el bloque y, en cierto modo, la parte clásica de la guía: el catálogo de formas de expresar trabajo y el criterio para elegir entre ellas. Quien lo domina deja de discutir sobre plantillas y empieza a decidir con argumentos.

24.1 La job story

Alan Klement propuso en 2013 sustituir la plantilla clásica por otra centrada en la situación: **«Cuando [situación], quiero [motivación], para [resultado esperado]»**. Su argumento es que el perfil del usuario informa menos que el contexto en que aparece la necesidad, y que la plantilla clásica invita a inventar personas.

La experiencia de Intercom, que la popularizó, y la revisión posterior de sus autores coinciden en algo importante: la job story **complementa** y no sustituye. Va mejor cuando la situación manda (el autónomo a las once de la noche, con el plazo mañana) y peor cuando lo que importa es distinguir a dos tipos de usuario con necesidades opuestas.

El elemento del backlog de Vantia que hemos ido citando está escrito así, y por eso funciona: «cuando me atasco de noche con un trámite y no tengo a quién preguntar, quiero una explicación clara en el momento, para terminar la gestión esa misma noche y no perder las ganas».

24.2 El caso de uso y su rebanada

El **caso de uso** describe un objetivo del actor con su flujo principal y sus alternativas, y sigue siendo la mejor herramienta cuando el flujo es lo complicado: muchas ramas, muchas excepciones, o consecuencias caras si se olvida una.

Su versión ágil, **Use-Case 2.0** (Jacobson, Spence y Kerr), resuelve la objeción del documento monolítico con las **rebanadas de caso de uso** (*use-case slices*): se identifica el caso de uso completo para tener el mapa, y se entrega por rebanadas, cada una con su historia y sus pruebas. Es el punto donde el caso de uso y la historia dejan de ser alternativas.

Situaciones donde conviene: sistemas con normativa que exige trazabilidad, equipos distribuidos que no pueden conversar a diario, dominios donde el coste del error es alto. El manual básico de Scrum Manager al que esta guía sustituye ya ofrecía esa tabla de contextos, y sigue siendo válida.

24.3 Funcionalidad, hipótesis, escenario y notación estructurada

La funcionalidad al estilo FDD. Palmer y Felsing propusieron una sintaxis para el trabajo sin usuario cercano: acción, resultado y objeto («calcular el total de retenciones de una factura»). Cohn la recomienda expresamente para APIs y componentes internos.

La hipótesis con su resultado. Para lo que hay que validar antes de construir: «creemos que [este cambio] producirá [este resultado]; lo sabremos cuando veamos [esta medida]». Su instrumental completo pertenece a la formación de Product Owner y a la skill de descubrimiento.

El escenario Gherkin. Es una **confirmación**, no una tarjeta. Escribir el elemento del backlog directamente como escenario confunde el qué con el cómo se comprueba, y produce backlogs ilegibles.

La notación EARS. Alistair Mavin propuso cinco plantillas para requisitos con condición («cuando…, el sistema deberá…», «mientras…», «si…, entonces…»). Es útil cuando el requisito lo va a leer una máquina o cuando hay que evitar toda ambigüedad, y por eso reaparece en el bloque H, en las especificaciones para agentes.

El pitch y el documento de diseño. Ryan Singer (*Shape Up*) propone el pitch: problema, apetito, solución esbozada, riesgos y lo que queda fuera. Los documentos de diseño de Google cumplen una función parecida para decisiones técnicas. Ambos sirven cuando la decisión necesita argumento y no cabe en una tarjeta.

24.4 Las cinco preguntas que deciden

Ante un elemento, cinco preguntas bastan para elegir el formato:

¿Quién se beneficia? Si hay un usuario identificable con una necesidad, historia. Si el beneficiario es el propio equipo o el sistema, trabajo técnico enunciado en claro.

¿Qué se ignora? Si la incertidumbre es alta, hipótesis o spike antes que cualquier formato de solución.

¿Qué conversación hace falta? Si hay que acordar reglas y ejemplos, historia con Example Mapping. Si hay que acordar un umbral, restricción. Si hay que acordar un flujo, caso de uso.

¿Quién lo va a leer? Si lo lee el equipo que conversa a diario, basta poco texto. Si lo lee un auditor, otro equipo o un agente, hace falta precisión y notación.

¿Qué cuesta equivocarse? Cuanto más caro sea el error, más explícito el formato.

La regla que resume el bloque. El formato sirve a la conversación que hace falta tener. Elegido al revés, la conversación se adapta al formato, y entonces se conversa de lo que el formato permite decir.

24.5 Saber no escribir una historia

El ejercicio con el que conviene cerrar el bloque es de reclasificación: tomar un lote de diez o quince elementos reales del backlog propio y decidir, con las cinco preguntas, qué es cada uno. El resultado típico es revelador: entre un tercio y la mitad no son historias, y buena parte no debería estar en el backlog.

Y hay un caso final que merece nombre propio: cuando la historia **añade ruido**. Un cambio de un texto legal, la actualización de un tipo impositivo o la corrección de una errata no necesitan usuario, motivación ni beneficio. «Actualizar el tipo de IVA reducido al 5 % desde el 1 de enero» es un elemento perfecto tal como está escrito.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Elegir el formato de un elemento con las cinco preguntas y justificar la elección.
- Escribir una job story y decir cuándo conviene frente a la plantilla clásica.
- Usar rebanadas de caso de uso cuando el flujo tiene muchas alternativas o el error es caro.
- Expresar trabajo sin usuario cercano con la sintaxis de funcionalidad, y lo incierto como hipótesis.
- Reconocer los elementos que no necesitan ningún formato y dejarlos como están.
- Reclasificar un lote de elementos mal formulados y decidir cuáles no deberían estar en el backlog.

Errores y antipatrones frecuentes

Sustituir todas las historias por job stories. Cambiar de plantilla universal no resuelve nada: el problema era la universalidad, no la plantilla.

Escribir el elemento como escenario Gherkin. El escenario confirma, no describe; usado como tarjeta produce backlogs que nadie puede ordenar.

Descartar los casos de uso por prejuicio. En dominios con muchas ramas o con trazabilidad exigida son la herramienta adecuada, y su versión por rebanadas es plenamente compatible con la entrega incremental.

Envolver en plantilla lo que no la necesita. Un cambio normativo escrito como historia añade tres líneas de ficción a algo que ya estaba claro.

Para profundizar

· Klement — Replacing the user story with the job story

· Adams — Cómo inventamos las job stories (Intercom)

· Jacobson, Spence y Kerr — Use-Case 2.0

· Mavin — Notación EARS

· Singer — Shape Up: escribir el pitch

· Ubl — Design Docs at Google

BLOQUE G · LA INTELIGENCIA ARTIFICIAL EN EL TRABAJO CON HISTORIAS

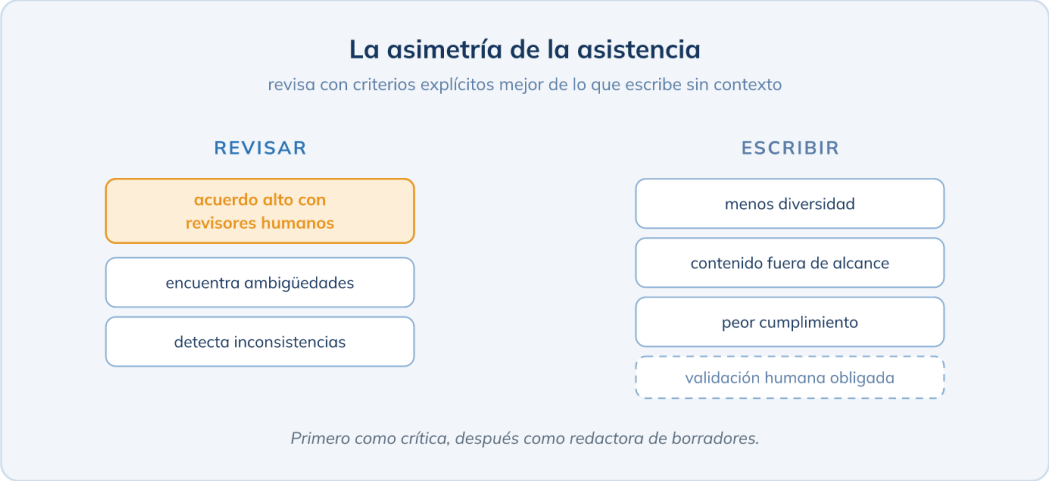
Capítulo 25. La asimetría: la inteligencia artificial revisa mejor de lo que escribe · EN CONSOLIDACIÓN

Qué dice la evidencia sobre evaluar frente a generar historias, y el orden de uso que se deriva de ella.

Introducción

El primer impulso de casi todo el mundo, al tener un modelo de lenguaje a mano, es pedirle historias. Es el uso de menor rendimiento y de mayor riesgo, y la evidencia lo dice con bastante claridad. En este capítulo vemos qué se ha medido, qué resultado da y qué orden de trabajo se deduce.

Conviene empezar con una advertencia sobre la naturaleza de lo que sigue. Buena parte de la investigación disponible es reciente, de laboratorio y con muestras pequeñas; hay poca evidencia de campo. Cuando algo sea propuesta de Scrum Manager y no hallazgo contrastado, lo diremos.



25.1 Como evaluadora, con criterios explícitos

El trabajo de Ronanki, Cabrero-Daniel y Berger (2023) comparó la evaluación de calidad de historias hecha por un modelo con la de revisores humanos, y encontró un acuerdo alto, en torno al 70-75 %, que subía por encima del 85 % tomando la mejor de tres respuestas. La investigación de Quattrocchi y otros (2025) confirma la parte de evaluación: con criterios claros, el modelo evalúa bien.

En detección de ambigüedades e inconsistencias, los trabajos comparativos con herramientas basadas en reglas dan ventaja al modelo, que encuentra problemas que las reglas no cubren, incluidos algunos que el análisis manual pasa por alto. Es coherente con el reparto del capítulo 5: lo que se decide leyendo puede detectarse mecánicamente, y un modelo de lenguaje es hoy el mejor detector disponible.

La condición: criterios explícitos. El modelo evalúa bien cuando se le dan los criterios (INVEST, los trece del marco de calidad, la lista de palabras de riesgo, las reglas del equipo). Sin criterios, produce opiniones de manual.

25.2 Como autora, sin contexto

El mismo trabajo de Quattrocchi encontró lo contrario al generar: las historias producidas tienen calidad superficial comparable, pero son **menos diversas**, cumplen **peor** los criterios de calidad y contienen **más elementos fuera de alcance**. El proyecto ALAS, con agentes especializados en mejorar historias, elevó la satisfacción de los equipos, y su conclusión incluye una condición explícita: requiere validación manual del Product Owner.

El estudio de campo más informativo es el de Steghöfer (2026), con Product Owners que ya trabajaban así. Sus dos hallazgos son de sentido común y por eso valen: la salida resulta **demasiado genérica porque falta el contexto**, y **no cita el material de origen**, de modo que verificarla cuesta casi lo mismo que haberla escrito.

La conclusión no es que no se use para redactar. Es que redactar es el paso donde menos aporta y el que más verificación exige.

25.3 El orden de uso que se deriva

Con esa asimetría, el orden razonable es el inverso al que casi todo el mundo aplica.

Primero como crítica. Pasarle lo que el equipo ya ha escrito, con los criterios explícitos: INVEST, ambigüedades, duplicados y solapes en el conjunto, coherencia entre elementos, casos límite que faltan. El resultado es una lista de candidatos para decidir en conversación.

Después como redactora de borradores. Con contexto suficiente (capítulo 26), para producir un primer texto que el equipo corrige, variantes de una misma historia, propuestas de criterios y de ejemplos, o cortes candidatos que después se juzgan con los criterios del capítulo 17.

Nunca como fuente de conocimiento del usuario. El BoK de Scrum Manager lo dice en una frase que conviene recordar: la inteligencia artificial no comprende usuarios reales; genera historias que suenan bien porque ha visto miles de ejemplos parecidos, sin experiencia directa con los usuarios de este producto.

25.4 Lo que ya sostiene Scrum Manager

Esta lectura no es nueva en la casa. La skill de Agile Inception ya sostiene que la asistencia es mejor revisora que autora del backlog, y el BoK recoge el mismo criterio en sus páginas de historia de usuario y de criterios de aceptación, con su límite explícito. La guía troncal Product Owner 2.0 lo aplica al trabajo del rol y añade el aviso pertinente: una historia impecablemente redactada que nadie ha conversado es una especificación disfrazada, con todos sus malentendidos intactos.

En el mercado han aparecido herramientas que asumen precisamente el papel de crítica, como el revisor de historias de Mountain Goat Software, que evalúa claridad del resultado, foco del alcance, comprobabilidad y encaje en el backlog. Es una señal de que la asimetría empieza a reconocerse también en el producto.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Explicar la asimetría entre evaluar y generar historias con la evidencia que la sostiene.
- Usar la asistencia como crítica con criterios explícitos y obtener una lista de candidatos.

- Decidir en qué momento del trabajo conviene pedir un borrador y con qué verificación.
- Rechazar el uso de la asistencia como fuente de conocimiento sobre los usuarios del producto.
- Situar el coste de verificación como parte del cálculo de la ganancia.

Errores y antipatrones frecuentes

Pedir «veinte historias» de entrada. Produce un lote genérico y poco diverso cuya verificación consume la ganancia; además fija un alcance que nadie ha decidido.

Criticar sin criterios. Sin criterios explícitos, la revisión devuelve consejos de manual en lugar de defectos concretos de estas historias.

Aceptar la crítica sin filtro. La lista es de candidatos, no de decisiones: descartar, fusionar o reescribir sigue siendo trabajo de producto.

Confundir acabado con calidad. Un texto bien redactado no indica que el contenido sea correcto, y esa confusión tiene nombre en el capítulo 27.

Para profundizar

· Quattrocchi y otros — ¿Pueden los LLM generar historias y evaluar su calidad?

· Ronanki y otros — ChatGPT en la evaluación de calidad de historias

· Zhang y otros — ALAS: agentes para mejorar historias

· Steghöfer — Faster than the Team, Faster than the Customer

· Scrum Manager BoK — Historia de usuario

BLOQUE G · LA INTELIGENCIA ARTIFICIAL EN EL TRABAJO CON HISTORIAS

Capítulo 26. Contexto, invención y procedencia · EMERGENTE

Qué contexto cambia el resultado, qué se inventa un modelo, cómo se verifica barato y cómo se conserva la procedencia.

Introducción

Si el capítulo anterior decide **cuándo** usar la asistencia, este decide **con qué**. La diferencia entre una salida genérica y una útil está casi toda en el contexto que se le da, y la diferencia entre una salida útil y un problema está en saber qué parte se ha inventado. En este capítulo tratamos las dos cosas, y cerramos con la procedencia.

26.1 Qué contexto cambia el resultado

El contexto que de verdad mejora una historia generada es concreto y limitado: la **evidencia del descubrimiento** (qué se ha observado, con qué datos), la **épica o el objetivo** al que sirve, las **personas reales** con sus situaciones (no arquetipos genéricos), las **reglas de dominio ya conocidas** y las **decisiones ya tomadas** que no se van a reabrir.

En Vantia, ese contexto ocupa poco más de una página: el problema del abandono nocturno con su cifra, el objetivo vigente, la descripción del autónomo que trabaja de noche y del gestor de la asesoría, las tres reglas del asistente y la decisión de que una respuesta inventada es peor que reconocer el límite. Con esa página delante, la salida cambia de naturaleza.

Contexto persistente, no repetido. Lo estable del producto (visión, objetivo, personas, reglas, glosario) conviene tenerlo en un documento reutilizable que se adjunta a cada tarea, y no reescrito en cada petición. Esa disciplina reaparece en el capítulo 28 como capas de contexto.

26.2 Qué no se pega

Hay material que no debe entrar en una conversación con un modelo, y conviene tenerlo decidido antes de necesitarlo: **datos personales de clientes** (nombres, identificadores fiscales, direcciones,

contenido de sus documentos), **notas crudas de entrevistas** con datos identificables, credenciales, y cualquier material sujeto a confidencialidad contractual.

La regla operativa es sencilla: se comparte lo que se podría enseñar en una presentación interna sin anonimizar nada. Cuando hace falta el caso real, se anonimiza primero. El instrumental de privacidad y de uso responsable pertenece a la skill de IA aplicada al trabajo ágil, que trata la cuestión con el detalle que merece.

26.3 La taxonomía de lo que se inventa

Usuarios y necesidades sintéticas. El modelo produce perfiles verosímiles que no corresponden a nadie. Nielsen Norman Group ha documentado el problema de los usuarios sintéticos con una conclusión clara: son útiles para practicar, no para decidir.

Reglas plausibles pero falsas. Las más peligrosas, porque suenan a dominio: umbrales, plazos, excepciones normativas. En un producto fiscal, esta categoría puede producir afirmaciones incorrectas sobre la propia normativa.

Criterios sin evidencia. Aparecen números (tres segundos, 95 %) que nadie ha decidido y que después se defienden como si fueran acuerdos.

Falso consenso. El texto presenta como acordado lo que solo es una interpretación posible, y al equipo le llega en forma de decisión tomada.

Las revisiones sistemáticas del área confirman que la mayor parte de la investigación está en fase temprana y documentan la frecuencia con que aparece material inventado, en una proporción que ninguna organización debería asumir sin verificación.

26.4 Cómo verificar barato

La verificación completa es costosa y no siempre necesaria; hay tres técnicas de coste bajo que atrapan la mayor parte de los problemas.

Exigir la fuente de cada criterio. Pedir, junto a cada criterio o regla, de dónde sale (una entrevista, una decisión previa, la normativa, o «propuesto por mí»). Lo que no tiene fuente queda marcado como supuesto.

Marcar los supuestos explícitamente. Que la salida distinga lo que sabe de lo que ha inferido. Las herramientas de especificación del bloque H usan una marca literal para esto, y la práctica es trasladable a cualquier historia.

Contrastar con quien conoce el dominio. Media hora del gestor de la asesoría vale más que dos horas de revisión propia. Y es el único filtro que detecta las reglas plausibles pero falsas.

El coste de estas tres técnicas es real y hay que contarlo: parte de la ganancia de la generación se va en verificar. Reconocerlo es lo que distingue una adopción profesional de un entusiasmo.

26.5 Procedencia y trazabilidad

Con asistencia en el proceso, la historia deja de tener un solo autor, y conviene que eso se vea. La distinción mínima útil tiene tres categorías: lo **aportado** (evidencia y decisiones del equipo), lo

inferido (lo que el modelo ha deducido del contexto) y lo **generado** (lo que ha producido sin apoyo), más el dato de **quién lo ha validado**.

No hace falta un sistema: basta una línea al pie del elemento. En Vantia, el fragmento de backlog ordenado incluye una columna de origen precisamente por esto, y en ella conviven «entrevistas con autónomos», «conversación de refinamiento», «propuesta del equipo» y «análisis automático del uso». Que el origen esté visible permite discutir el elemento con la confianza que corresponde a cada procedencia.

Hay además una razón externa. El artículo 50 del Reglamento europeo de inteligencia artificial, aplicable desde agosto de 2026, establece obligaciones de transparencia para el contenido generado, con una exención cuando existe **revisión humana o control editorial y un responsable identificable** del contenido. Para el trabajo con historias, esa condición coincide con la buena práctica: alguien responde de cada elemento del backlog. El detalle normativo pertenece a otras skills; aquí basta saber que la trazabilidad ya no es solo una cortesía profesional.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Preparar el contexto de producto que hace útil una salida asistida, y mantenerlo como activo reutilizable.
- Decidir qué material no entra en una conversación con un modelo y anonimizar cuando hace falta.
- Reconocer las cuatro clases de invención en un material generado.
- Aplicar las tres técnicas de verificación barata y contar su coste en el cálculo de la ganancia.
- Registrar la procedencia de un elemento distinguiendo lo aportado, lo inferido y lo generado, con su responsable.

Errores y antipatrones frecuentes

Pedir sin contexto y culpar al resultado. Sin evidencia ni reglas del dominio, la salida solo puede ser genérica; el defecto está en la petición.

Repetir el contexto en cada conversación. Además de costoso, produce versiones divergentes del mismo contexto; conviene mantenerlo en un documento único.

Pegar notas crudas de entrevistas. Contienen datos identificables y rara vez hacen falta: lo que aporta es la evidencia destilada, no la transcripción.

Aceptar números que nadie decidió. Los umbrales inventados se convierten en compromisos en cuanto entran en un criterio de aceptación.

Presentar como acordado lo inferido. El equipo hereda decisiones que no sabe que existen, y las descubre cuando alguna resulta equivocada.

Para profundizar

· Steghöfer — Faster than the Team, Faster than the Customer

· Cheng y otros — IA generativa en ingeniería de requisitos (revisión)

· Nielsen Norman Group — Usuarios sintéticos

· Reglamento europeo de IA, artículo 50

· Skill Arena — IA aplicada al trabajo ágil

BLOQUE G · LA INTELIGENCIA ARTIFICIAL EN EL TRABAJO CON HISTORIAS

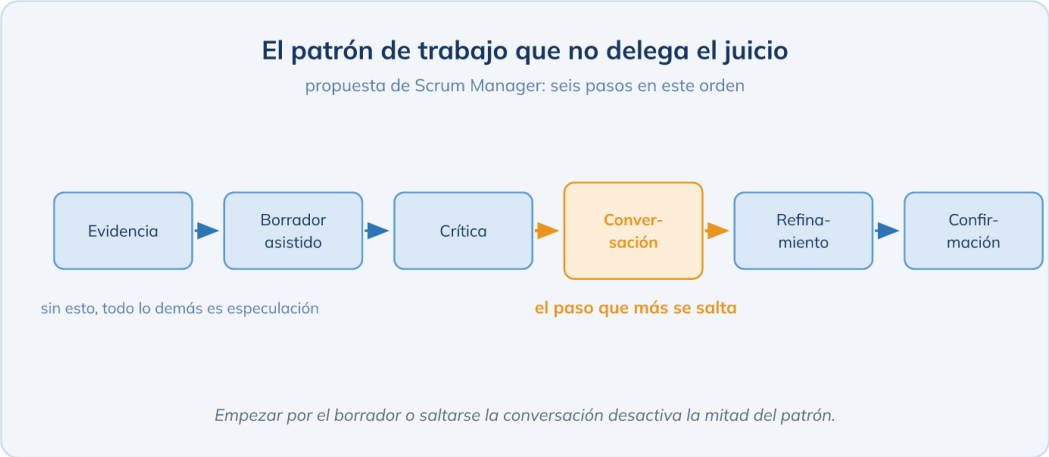
Capítulo 27. Riesgos y el patrón de trabajo que no delega el juicio ·

EMERGENTE

Los cinco riesgos con su evidencia, el patrón de seis pasos de Scrum Manager y el papel que le queda al profesional.

Introducción

Los riesgos de trabajar con asistencia no son los que suelen citarse. Que el modelo se equivoque se detecta; lo que cuesta detectar es que sus errores lleguen con buen acabado, en gran cantidad y en el momento en que menos se revisa. En este capítulo los ordenamos con la evidencia disponible y proponemos un patrón de trabajo que los contiene.



27.1 Los cinco riesgos

La invención con apariencia de solvencia. Ya vista en el capítulo 26: reglas, criterios y usuarios que no existen, redactados como si existieran.

El sesgo de precisión. El acabado del texto concede autoridad al contenido. Humanizing Work lo ha documentado en gestión de producto: una historia con criterios bien formateados y números redondos se cuestiona menos que la misma historia escrita a mano, aunque su fundamento sea peor.

La homogeneización. Los trabajos sobre creatividad asistida encuentran que las propuestas individuales mejoran y el conjunto se parece más entre sí. Conviene decirlo con precisión: esa evidencia viene de otros dominios (escritura creativa, generación de ideas) y su traslado al trabajo con historias es una **inferencia razonable, no un hallazgo medido**.

La sobreproducción de backlog. Generar cuesta menos que decidir, así que el backlog crece más deprisa que la capacidad de ordenarlo, y el resultado es el deterioro del capítulo 7 a mayor velocidad.

La sustitución de la conversación por el chat. El riesgo mayor, porque afecta a la razón de ser de la técnica: si la historia llega redactada y con criterios, la conversación parece innecesaria y se salta.

El sesgo de automatización. La investigación clásica sobre automatización (Parasuraman y Manzey) sostiene algo incómodo: la tendencia a confiar en el sistema automatizado **no se corrige con formación ni con advertencias**. Lo que la contiene es el diseño del proceso, no la prudencia individual.

27.2 El patrón de trabajo de Scrum Manager

Frente al antipatrón de «pedir historias y llamar backlog al resultado», proponemos un patrón de seis pasos. Es una **propuesta de Scrum Manager**, construida sobre la evidencia de los capítulos anteriores y sobre la práctica de la casa, no un consenso del oficio:

1. Evidencia. Se parte de lo observado: entrevistas, datos de uso, incidencias, resultados de experimentos. Sin este paso, todo lo demás es especulación con buen formato.

2. Borrador asistido. Con el contexto del capítulo 26, se pide un primer texto: historias candidatas, criterios, ejemplos, cortes.

3. Crítica. Se pide una revisión con criterios explícitos, del borrador propio y del generado, y se obtiene una lista de defectos y de huecos.

4. Conversación. El material entra en el refinamiento con las perspectivas necesarias, incluidos los supuestos y las preguntas abiertas, no solo el resultado.

5. Refinamiento. Se decide: qué entra, qué se parte, qué se investiga, qué se descarta, y se fijan las reglas y los ejemplos acordados.

6. Confirmación. Se cierran los criterios observables y se registra la procedencia y las decisiones (capítulo 13).

El orden importa más que los nombres. Los dos errores frecuentes son empezar por el paso 2 (sin evidencia) y saltarse el 4 (sin conversación), y cada uno desactiva la mitad del patrón.

27.3 El papel que le queda al profesional

Con la redacción abaratada, el trabajo de producto se concentra en seis tareas que la asistencia no hace por nadie: **dar contexto** (que exige conocer el producto y su evidencia), **preguntar** (que exige saber qué falta), **contrastar** (con quien conoce el dominio y con los datos), **encontrar el corte** (que exige el criterio del bloque D), **decidir** (que exige responsabilidad) y **verificar la intención** (que exige saber qué se buscaba).

Dicho de otro modo: el oficio no se desplaza hacia la escritura mejorada, sino hacia el juicio. Esa es la tesis de esta guía y la razón de que su parte clásica ocupe seis bloques de ocho.

La literatura de producto apunta en la misma dirección desde ángulos distintos, con propuestas como el Product Owner orquestador y con avisos sobre el efecto de la asistencia en la calidad de las decisiones. Ninguna de esas propuestas está asentada, y conviene seguirlas con atención crítica.

27.4 El coste que no se ve

Hay un último efecto documentado que conviene tener presente porque es de largo plazo: la investigación sobre uso de asistencia y pensamiento crítico apunta a una reducción del esfuerzo

cognitivo y de la confianza en el propio juicio cuando la asistencia se usa de forma sostenida. Son autoinformes y no permiten inferir causalidad, así que no conviene exagerarlos.

La lectura prudente es de diseño del trabajo: conviene reservar deliberadamente el juicio para las decisiones que importan (qué entra, cómo se corta, qué se descarta) y automatizar sin escrúpulos lo mecánico (uniformar, detectar duplicados, formatear). Delegar en el orden inverso es lo que produce equipos que redactan mucho y deciden mal.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Enumerar los cinco riesgos y distinguir los que tienen evidencia medida de los que son inferencia.
- Explicar por qué el sesgo de automatización no se corrige con formación y qué lo contiene.
- Aplicar el patrón de seis pasos y reconocer los dos errores frecuentes de orden.
- Enunciar las seis tareas del profesional que la asistencia no sustituye.
- Repartir el trabajo entre lo que conviene automatizar y lo que conviene reservar al juicio.

Errores y antipatrones frecuentes

Confiar en el acabado. El formato impecable no informa sobre el fundamento, y además desalienta la crítica de quien revisa.

Generar backlog en masa. El conjunto crece más deprisa de lo que puede ordenarse y el deterioro se acelera; conviene generar poco y decidir mucho.

Saltarse la conversación porque el texto está completo. Es el riesgo que más daña, porque elimina justamente lo que hace útil a una historia.

Presentar el resultado sin los supuestos. El equipo hereda decisiones invisibles; llevar al refinamiento las preguntas abiertas cuesta un minuto.

Automatizar el juicio y hacer a mano lo mecánico. Es el reparto inverso al razonable: lo mecánico es donde la asistencia rinde sin riesgo.

Para profundizar

· Parasuraman y Manzey — Complacencia y sesgo en el uso de la automatización

· Humanizing Work — Sesgo de precisión y IA en gestión de producto

· Humanizing Work — Dos trampas de la IA en el backlog

· Lee y otros — El impacto de la IA generativa en el pensamiento crítico

· Skill Arena — IA aplicada al trabajo ágil

BLOQUE H · LA HISTORIA COMO ENTRADA DE UN FLUJO CON AGENTES

Capítulo 28. De la historia a la spec: la intención como fuente de verdad

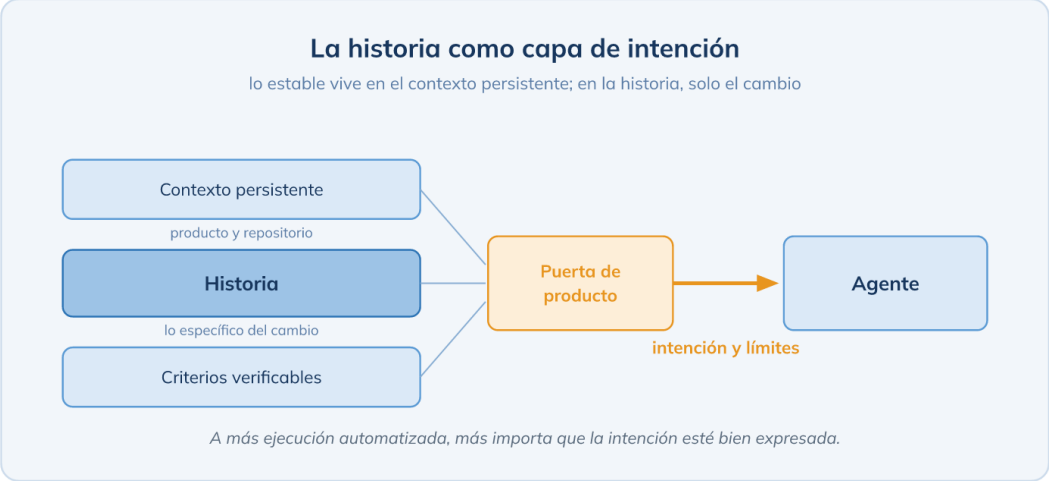
· EMERGENTE

Qué hacen hoy las herramientas de desarrollo con agentes, qué relación guarda la spec con la historia y cómo se reparte el contexto.

Introducción

Este último bloque trata lo que cambia cuando quien implementa una historia es, en todo o en parte, un agente de inteligencia artificial. Es la materia más reciente de la guía, con herramientas de meses de vida y posiciones enfrentadas, así que la trataremos con las etiquetas de evidencia por delante y sin dar por asentado lo que no lo está.

Conviene decir de entrada qué no vamos a tratar aquí. El flujo completo del desarrollo guiado por especificaciones, con sus fases, sus puertas y sus herramientas, tiene skill propia en Scrum Manager, igual que la convivencia entre historias y specs según el riesgo. Este bloque mira solo una cosa: qué le pasa a la historia, y qué debe hacer producto.



28.1 Qué hacen hoy las herramientas

Las herramientas de desarrollo con agentes trabajan sobre un artefacto intermedio, la **especificación** (*spec*), y conviene saber qué contiene, porque no es lo que muchos suponen. La documentación de Kiro (Amazon, julio de 2025) describe especificaciones que incluyen historias de usuario con sus criterios de aceptación en notación estructurada, y un tipo específico para corrección de defectos que declara explícitamente el comportamiento que **no debe cambiar**.

La plantilla de GitHub Spec Kit (septiembre de 2025, versión 1.0 en agosto de 2026) es aún más elocuente: contiene historias priorizadas y marcadas como comprobables por separado, escenarios de aceptación, casos límite, supuestos declarados y una marca literal para lo que falta aclarar. Es decir, todo lo que esta guía enseña, ordenado en un documento.

La spec contiene a la historia. No la sustituye. Lo que ocurre es que el trabajo de producto (intención, reglas, ejemplos, límites) pasa a ser la parte crítica de un documento que antes era

técnico. El BoK de Scrum Manager lo fija así: el desarrollo guiado por especificaciones no sustituye a las historias de usuario, las complementa.

28.2 La doctrina y sus críticos

La formulación fuerte de la tendencia dice que la especificación es el nuevo código y que la intención es la fuente de verdad: si el código se puede regenerar desde la intención, lo valioso es la intención bien expresada. Es una idea sugerente y con consecuencias evidentes para el trabajo de producto.

Tiene críticos serios y conviene conocerlos. La objeción principal es que un flujo que exige escribir la especificación completa antes de construir reintroduce el ciclo en cascada con otro nombre, con su misma debilidad: la especificación detallada de lo que aún no se entiende. Los análisis de campo de Thoughtworks son más matizados y señalan que estas herramientas pueden ser un mazo para partir una nuez, con especificaciones desproporcionadas para cambios pequeños.

La posición prudente, y la que esta guía adopta, es tratarla como **tendencia en evaluación**: conviene conocer el formato porque ya llega a los equipos, conviene aportar desde producto lo que solo producto sabe, y conviene no rehacer el proceso de trabajo alrededor de una herramienta de meses.

28.3 Las capas de contexto

Hay un hallazgo práctico que sí está bien establecido y que cambia cómo se escribe una historia en este escenario: el contexto se organiza en capas, y repetir lo estable en cada petición **degrada** el resultado.

Capa persistente. Lo que no cambia entre tareas: qué es el producto, quiénes son sus usuarios, el glosario del dominio, las convenciones del repositorio, las decisiones vigentes. Vive en los ficheros de contexto que las herramientas leen automáticamente y se mantiene como un activo del equipo.

Capa de la tarea. Solo lo específico de este cambio: la intención, las reglas que aplican aquí, los ejemplos, los límites y las dudas. Es lo que va en la historia.

La razón de esa separación es medida: el cumplimiento de instrucciones se degrada a medida que crece su número, de modo que una historia que repite el contexto del producto compite consigo misma. Es exactamente el mismo principio que la tarjeta con «lo justo para recordar», veinticinco años después y por otro motivo.

28.4 Qué significa esto para el trabajo con historias

Tres consecuencias, y ninguna obliga a cambiar de método. La primera, que el detalle de la historia deja de ser opcional en el tramo que el agente va a ejecutar: no porque haya que escribir más, sino porque hay que escribir **lo que antes se decía hablando**. La segunda, que la conversación no desaparece, se adelanta: ocurre antes de escribir la historia, y su resultado es lo que se escribe. La tercera, que la verificación gana peso, y de eso trata el capítulo 30.

En Vantia, la primera historia que el equipo pasó a un agente fue el aviso de plazo fiscal en riesgo. La historia que había servido para conversar con el equipo (una frase, tres criterios) resultó insuficiente para el agente en un punto concreto y previsible: no decía qué **no** debía cambiar. El agente reescribió la plantilla de correo que ya usaba el resto de la plataforma.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Explicar qué contiene una especificación de las herramientas actuales y qué papel juega la historia dentro.
- Situar el desarrollo guiado por especificaciones como tendencia en evaluación, con su doctrina y sus críticas.
- Repartir el contexto entre la capa persistente y la de la tarea, y justificar por qué no se repite.
- Anticipar qué parte de lo que antes se conversaba hay que escribir cuando ejecuta un agente.

Errores y antipatrones frecuentes

Escribir la especificación completa por adelantado. Es la crítica fundada a esta tendencia: especificar en detalle lo que aún no se entiende reproduce el problema del ciclo en cascada.

Repetir el contexto del producto en cada historia. Degrada el cumplimiento y multiplica las versiones divergentes del mismo contexto.

Rehacer el proceso alrededor de la herramienta. Las herramientas tienen meses; el criterio de producto tiene décadas, y es lo que hay que aportar.

Suponer que la spec sustituye a la conversación. La conversación se adelanta, no se elimina: su resultado es lo que la especificación recoge.

Para profundizar

· GitHub Spec Kit — desarrollo guiado por especificaciones

· Kiro — especificaciones de funcionalidad

· Böckeler — Understanding spec-driven development

· Zaninotto — La revancha del ciclo en cascada

· Scrum Manager BoK — Spec-Driven Development

· Skill Arena — SDD en equipos ágiles

BLOQUE H · LA HISTORIA COMO ENTRADA DE UN FLUJO CON AGENTES

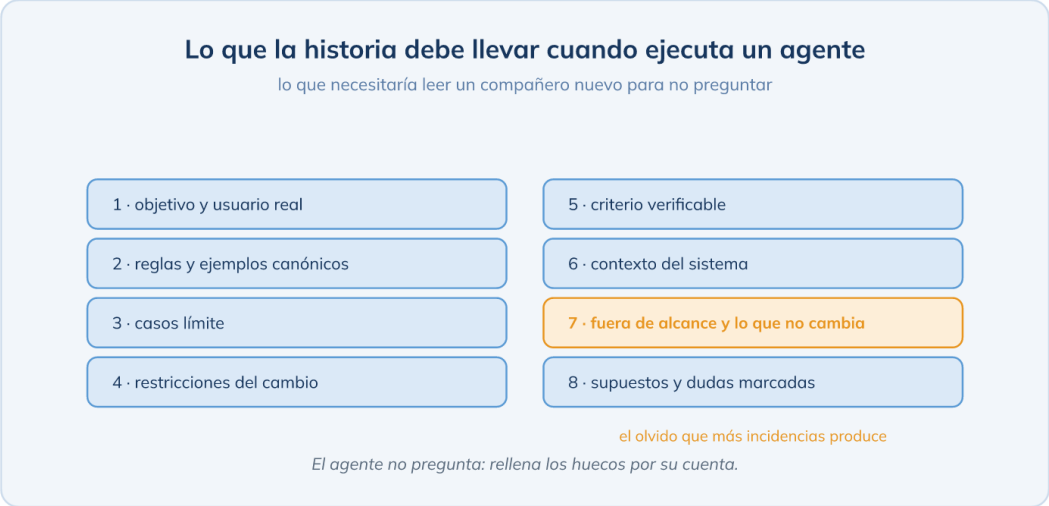
Capítulo 29. Lo que el agente necesita y el criterio que se verifica solo ·

EMERGENTE

Los ocho elementos que la historia debe llevar, las dudas como entregable y la relectura de INVEST bajo agentes.

Introducción

Hay una regla que resume este capítulo y conviene tenerla presente antes del detalle: la historia debe contener lo que necesitaría leer **un compañero nuevo y competente** para hacer la tarea sin preguntar. Ni más, porque el exceso degrada; ni menos, porque el agente no pregunta.



29.1 Los ocho elementos

1. Objetivo y usuario real. Para qué se hace y para quién, con el mismo criterio del capítulo 1. Sin esto, el agente optimiza lo que puede medir.

2. Reglas y ejemplos canónicos. Las reglas que aplican a este cambio y dos o tres ejemplos que las fijan, no una batería exhaustiva.

3. Casos límite. Los valores en la frontera y los casos negativos que importan (capítulo 11).

4. Restricciones y requisitos no funcionales del cambio. Los que aplican aquí, no los generales del sistema (que están en la capa persistente).

5. Criterio de éxito verificable. Cómo se comprueba, preferiblemente de forma mecánica (29.3).

6. Contexto del sistema. Qué ficheros, componentes o patrones existentes son relevantes, y dónde mirar un ejemplo parecido ya resuelto.

7. Fuera de alcance y comportamiento que no debe cambiar. El elemento más olvidado y el que más incidencias evita. Es lo que faltó en la historia del aviso de plazos de Vantia.

8. Supuestos declarados y dudas marcadas. Lo que se ha asumido y lo que no se sabe, señalado como tal.

Las dudas son un entregable. El agente no pregunta: rellena los huecos por su cuenta y de forma inconsistente. Trabajos recientes sobre agentes de programación confirman que, ante enunciados ambiguos, no piden aclaración. Por eso una duda marcada vale más que una decisión inventada.

29.2 Pocos criterios y canónicos

La tentación natural es cargar la historia de criterios, y es contraproducente por una razón medida: el cumplimiento conjunto de instrucciones **cae** a medida que crece su número. Con veinte criterios, la probabilidad de que se respeten todos es baja, y además no hay manera de saber cuál se ha ignorado.

La práctica que funciona es la del capítulo 10 llevada al extremo: pocos criterios, canónicos (que fijen la regla, no que enumeren casos) y verificables. Los casos particulares van como ejemplos, que ilustran sin multiplicar instrucciones.

Las guías de las propias herramientas coinciden. La documentación del agente de programación de GitHub pide criterios completos sobre qué es una buena solución; la de Claude Code insiste en declarar lo que queda fuera de alcance y en no dar por bueno lo que no se puede verificar; y la de Devin acota el tamaño de la tarea a unas pocas horas de trabajo, que es la escala del capítulo 14.

29.3 Del criterio observable al test

Un criterio de aceptación bien escrito es la semilla de la verificación mecánica, y ese es el punto donde el trabajo del bloque C rinde más en este escenario. La cadena es directa: la regla acordada y su ejemplo concreto se convierten en un escenario declarativo (capítulo 12), y el escenario se convierte en una prueba que el agente puede ejecutar para comprobarse a sí mismo.

La investigación sobre generación de pruebas de aceptación con modelos de lenguaje da resultados prometedores en cuanto a utilidad de los escenarios generados, con la salvedad esperable de que una parte necesita regenerarse o descartarse. Es decir: la generación ayuda, y la aprobación del escenario sigue siendo humana, porque el escenario **es** el criterio.

Para funcionalidades probabilísticas se aplica lo del capítulo 10: rúbrica, conjunto de evaluación y umbral. Es la única forma de que un criterio de aceptación sea verificable cuando no hay una respuesta única correcta.

29.4 INVEST cuando ejecuta un agente

El acrónimo del capítulo 4 sigue sirviendo, con desplazamientos que proponemos como **relectura de Scrum Manager**, no como consenso del oficio:

Independiente y **pequeña** se refuerzan. Con ejecución rápida y coste bajo por intento, la pieza pequeña e independiente es aún más ventajosa: se verifica antes y se descarta sin duelo.

Comprobable pasa a **verificable por máquina**. No basta con poder escribir un ejemplo: conviene que exista una comprobación ejecutable, porque es lo que permite delegar la ejecución sin delegar el criterio.

Negociable se adelanta. La negociación deja de ocurrir durante la implementación (el agente no negocia) y se traslada a la conversación previa y a la puerta del capítulo 30. Aquí está el debate

abierto que conviene declarar: adelantar la negociación se parece a fijar el detalle demasiado pronto, que es precisamente lo que la técnica quería evitar. No hay respuesta asentada, y el criterio prudente es adelantar solo lo que el agente necesita para no inventar.

Valiosa y **estimable** no cambian de naturaleza, aunque el dimensionado se vuelve más importante: las tareas que caben en pocas horas de trabajo del agente son las que mejor funcionan, y eso empuja hacia el corte fino del bloque D.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Preparar una historia con los ocho elementos y comprobar que no falta el fuera de alcance.
- Marcar las dudas como entregable en lugar de resolverlas por conjetura.
- Escribir pocos criterios canónicos y trasladar los casos particulares a ejemplos.
- Convertir un criterio observable en un escenario declarativo y en una comprobación ejecutable.
- Releer INVEST en este escenario y explicar qué desplazamiento tiene cada propiedad.
- Reconocer el debate abierto sobre adelantar la negociación y adoptar el criterio prudente.

Errores y antipatrones frecuentes

Omitir lo que no debe cambiar. Es el olvido que produce más incidencias: el agente reescribe lo que había, porque nadie dijo que estuviera bien.

Cargar la historia de criterios. El cumplimiento conjunto se degrada con el número, y no se sabe cuál se ha ignorado.

Resolver las dudas por conjetura para «no bloquear». Una conjetura escrita se convierte en requisito; una duda marcada se convierte en conversación.

Repetir el contexto del sistema en cada historia. Su sitio es la capa persistente; repetido, compite con lo específico del cambio.

Delegar la aprobación de los escenarios. El escenario es el criterio de aceptación: aprobarlo es una decisión de producto, aunque lo haya redactado un modelo.

Para profundizar

· Spec Kit — plantilla de especificación

· GitHub — obtener buenos resultados del agente de codificación

· Claude Code — buenas prácticas

· Ferreira y otros — generación de pruebas de aceptación con LLM

· Informe sobre la maldición de las instrucciones

· Kiro — buenas prácticas de especificación

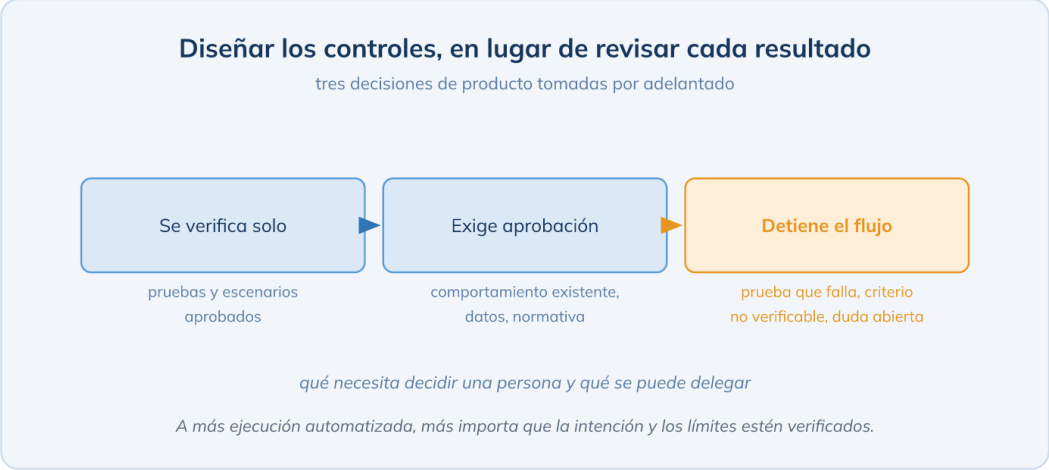
BLOQUE H · LA HISTORIA COMO ENTRADA DE UN FLUJO CON AGENTES

Capítulo 30. La puerta de requisitos desde producto · EMERGENTE

Qué revisa producto antes de que un agente arranque, cómo se diseña el control del bucle y qué dice la evidencia sobre la productividad.

Introducción

Cerramos la guía con la parte que corresponde a producto cuando la ejecución está automatizada: una puerta ligera antes de que el agente empiece, y un papel nuevo que consiste en diseñar los controles del proceso en lugar de revisar cada resultado.



30.1 Los seis puntos de la puerta

La puerta no es una Definition of Ready con otro nombre (el capítulo 7 explica por qué esa deriva es dañina): son seis comprobaciones que se hacen en dos minutos, y su función es evitar el trabajo automatizado sobre una intención equivocada.

1. La intención es la real, no la atribuida. Que lo que pide la historia sea lo que se quiere conseguir, y no la solución que alguien supuso.

2. Están las restricciones y lo que no debe cambiar. El punto más olvidado del capítulo 29.

3. Los criterios son observables. Y, a poder ser, verificables mecánicamente.

4. El alcance es manejable. Del tamaño que el agente completa con fiabilidad, que hoy es de pocas horas de trabajo.

5. El qué no lleva decisiones técnicas dentro. Salvo las que son restricciones reales del sistema.

6. Las dudas están marcadas. Y las que bloquean, resueltas antes de arrancar.

La pregunta que ordena el refinamiento. Dave West la formuló con precisión para este escenario: qué necesita decidir una persona y qué se puede delegar. Es la pregunta con la que conviene repasar cualquier historia antes de pasarla a un agente.

30.2 Sobre el bucle, no dentro del bucle

Con volumen de ejecución automatizada, revisar cada resultado deja de ser viable, y el papel se desplaza a **diseñar los controles**: qué se verifica automáticamente, qué requiere aprobación humana, qué se puede deshacer sin coste y qué no, y en qué punto se detiene el flujo si algo no cuadra.

Aplicado al trabajo con historias, eso significa decidir tres cosas por adelantado. Qué comprobaciones acompañan a cada historia (pruebas, escenarios aprobados, revisión de una persona). Qué cambios exigen aprobación explícita antes de integrarse (los que afectan a comportamiento existente, a datos de clientes o a obligaciones normativas). Y qué señal detiene el proceso (una prueba que falla, un criterio no verificable, una duda que reaparece).

Es una responsabilidad de producto, no de ingeniería, porque las tres decisiones son sobre riesgo y valor, no sobre implementación.

30.3 Los riesgos de este escenario

Más documentación sin más entendimiento. El riesgo mayor: especificaciones extensas que nadie ha conversado, con el mismo malentendido de siempre y mejor presentación.

Supuestos automatizados. Un supuesto no marcado se convierte en comportamiento implementado, y se descubre cuando alguien lo usa.

Propagación rápida de errores. Un criterio mal escrito se ejecuta en minutos y en muchos sitios; el coste de un malentendido ya no lo amortigua la lentitud del proceso.

Falsa precisión. El texto estructurado y las notaciones formales dan apariencia de exactitud a decisiones que nadie ha tomado.

30.4 Lo que dice la evidencia sobre la productividad

Aquí conviene ser especialmente cuidadoso, porque la exageración es la norma. El estudio de METR (julio de 2025) midió a desarrolladores expertos trabajando en sus propios repositorios con y sin asistencia, y encontró que **tardaron un 19 % más** con asistencia, mientras ellos creían haber ido más rápido. Es un estudio con una muestra pequeña y en un contexto concreto, y no autoriza a concluir que la asistencia no sirva; sí autoriza a desconfiar de la percepción como medida.

El informe DORA de 2025 aporta el panorama: la adopción es casi universal (en torno al 90 %) y la confianza es mucho menor (cerca de un tercio declara poca o ninguna). Y ofrece un dato útil para esta guía: entre las capacidades que asocia al rendimiento con inteligencia artificial están el **trabajo en lotes pequeños** y el **foco en el usuario**, que son, exactamente, el contenido de los bloques D y A.

La conclusión que esta guía sostiene, y con la que cierra: a más ejecución automatizada, más importa que la intención y los límites estén bien expresados y verificados. La técnica que enseña este material no queda obsoleta con los agentes; queda en el centro.

Lo que debes saber hacer

Al dominar este tema, un profesional es capaz de:

- Repasar una historia con los seis puntos de la puerta antes de pasarla a un agente.

- Formular la pregunta que reparte lo que decide una persona y lo que se puede delegar.
- Diseñar los controles del proceso: qué se verifica solo, qué exige aprobación y qué detiene el flujo.
- Reconocer los cuatro riesgos de este escenario y las prácticas que los contienen.
- Interpretar con prudencia la evidencia sobre productividad y desconfiar de la percepción como medida.

Errores y antipatrones frecuentes

Convertir la puerta en una Definition of Ready. Reaparece la deriva del capítulo 7: un formulario de veto en lugar de dos minutos de comprobación.

Revisar cada resultado. No escala y desplaza el trabajo de producto a la inspección; lo que hay que diseñar son los controles.

Fiarse de la sensación de velocidad. La evidencia disponible muestra que la percepción puede ir en dirección contraria a la medida.

Especificar más para entender mejor. El entendimiento compartido no crece con la extensión del documento, y este es el hilo con el que empezó la guía.

Para profundizar

· METR — estudio con desarrolladores expertos

· DORA — informe 2025

· DORA — capacidades

· Böckeler — humanos y agentes en el bucle

· Scrum.org — ¿Está tu Product Backlog listo para los agentes?

· Skill Arena — Scrum en equipos con IA