Le ayudamos a crear una cultura y un proceso de producto exitosos

8 de octubre de 2008

El nuevo backlog de historias de usuario es un mapa

Por qué el backlog de historias de usuario plano no funciona y cómo crear un mejor backlog que lo ayudará a explicar su sistema, priorizar y planificar sus lanzamientos de manera más efectiva.

Este es Gary .

Gary y yo trabajamos juntos durante un día para crear un mapa de historias de usuario, una versión mejorada de un backlog de producto. La creación de un mapa de historias de

Próximos eventos

SEP 23 — 23 de septiembre a las 8:00 a. m. - 26 de septiembre a las 12:30 p. m . PDT

Liderazgo apasionado de productos en línea y en directo: horario diurno en EE. UU., horario vespertino en Reino Unido y Europa: septiembre de 2024

OCT — 21 de octubre a las 12:00 p. m. - 24 de octubre a las 3:30 p. m . EDT

usuario nos ayuda a centrarnos en el panorama general, el producto en su conjunto, en lugar de centrarnos miopemente en una historia individual.

Cuando llegó el momento de priorizar, Gary lo hizo teniendo en cuenta todo el contexto del sistema. En el caso de Gary, originalmente se propuso crear Mimi (abreviatura de *Music Industry Marketing Interface)* . Sin embargo, cuando llegó el momento de priorizar, nos centramos en el miembro de la banda que promocionaba un concierto en un club. Era demasiado grande para una sola historia, demasiado grande para una "épica" en realidad... se necesitó la mayor parte del mapa para pensar en todos los detalles del diseño de un correo promocional, la creación de una lista de audiencia, el envío de la promoción y el seguimiento de las respuestas. Al final del ejercicio de mapeo, todas las demás cosas relacionadas con la música que Mimi podría haber hecho parecían una tarea difícil. Es más, crear un software comercial que facilitara y acelerara el trabajo de enviar correos parecía una buena idea. Y resultó que lo era.

Mimi se lanzó después de mucho sudor y esfuerzo por parte de Gary , Dave Hoover y la excelente gente de Obtiva . Al momento de escribir esto, Mimi envía cerca de cuatro millones de mensajes por mes. *La experiencia del usuario es una obra de arte* . Es una pieza de software atractiva. Y la lista de historias de usuario que creamos ese primer día de discusión todavía describe la misma visión de alto nivel de Mimi que vimos entonces, menos las historias que sacamos del lanzamiento, por supuesto.

Los backlogs planos no funcionan para mí

Una de las cosas más preocupantes, para mí, sobre la práctica común de desarrollo ágil es la idea del backlog de historias de usuario plano. Me parece una idea tonta. Bueno, no completamente tonta, en el sentido de que si voy a crear software de forma incremental en pequeñas partes, tendré que averiguar con qué parte empezar. El backlog coloca esas partes en orden de creación, desde el valor más alto hasta el valor más bajo, lo cual es genial si eres un gerente de proyectos, lo cual no es mi caso.

Estoy escribiendo esto desde el avión mientras regreso de una oficina de un cliente. Tuvimos lo que creo que fueron un par de días bastante exitosos de redacción de historias y planificación de lanzamiento. Me divertí y me alegró ver que el grupo realmente se involucró y se sintió bien con lo que se les ocurrió. He estado usando una práctica que [ahora] llamo mapeo de historias . Escribí por primera vez sobre este enfoque en el artículo " Cómo lo divides ". El artículo se publicó en enero de 2005, pero cuando lo escribí, ya había estado usando la práctica durante un par de años.

Años después, todavía me encanta ese enfoque… tanto que corro el riesgo de verlo como una " solución milagrosa ", lo que es una señal temprana de que me estoy volviendo un tonto dogmático. Sin embargo, veo que cada vez más personas llegan a enfoques similares, así que al menos sé que no estoy demasiado loco.

Permítame describir lo que está mal con la escritura de historias, luego, en general, qué es un mapa de historias y por qué resuelve mis problemas.

Artículos populares

El desarrollo de doble vía no es una vía dual

El desarrollo de Kanban simplificado en exceso

El nuevo backlog de historias de usuario es un mapa

Artículos recientes

Cómo identificar los productos digitales de su organización 1 de agosto de 2024

El backlog plano es una explicación deficiente de lo que hace un sistema

Organizar las historias de usuario en el orden en que las crearás no me ayuda a explicarles a los demás lo que hace el sistema. Intenta entregarles tu lista de historias de usuario a las partes interesadas o a los usuarios cuando te pregunten "¿qué hace el sistema que estás creando?".

En mi opinión, la parte difícil del desarrollo de software es intentar comprender el sistema (el sistema completo). Una de las quejas más comunes que escucho de los equipos ágiles es que pierden la visión general (si es que alguna vez la tuvieron).

Una vez traté de explicarle a un compañero de trabajo lo que pensaba que estaba mal con una lista de tareas pendientes. La historia es algo así:



“Pasamos mucho tiempo trabajando con nuestros clientes. Nos esforzamos por entender sus objetivos, sus usuarios y las partes principales del sistema que podríamos construir. Luego, finalmente, llegamos a los detalles: las partes de la funcionalidad que nos gustaría construir. En mi cabeza, veo un árbol donde el tronco se construye a partir de los objetivos

Tipo de producto Lienzo 1 de agosto de 2024

Todo es un producto 1 de agosto de 2024

o los beneficios deseados que impulsan el sistema; las ramas grandes son los usuarios; las ramas pequeñas y las ramitas son las capacidades que necesitan; luego, finalmente, las hojas son las historias de usuario lo suficientemente pequeñas como para incluirlas en las iteraciones de desarrollo".

"Después de todo ese trabajo, después de establecer todo ese entendimiento compartido, siento que sacamos todas las hojas del árbol y las cargamos en una bolsa para hojas, y luego cortamos el árbol".

"Para mí, eso es un atraso plano: una bolsa de mantillo sin contexto".

Necesito ese contexto para poder contar realmente una historia sobre el sistema.

Para un sistema nuevo, el backlog plano es desalentador para ayudarme a determinar si he identificado todas las historias.

Si tengo una pila de fichas o una hoja de cálculo llena de historias de usuarios, puedo mirarlas y comentarlas durante horas... pero siempre me quedo con la persistente sensación de que hay algo que me estoy perdiendo. Por supuesto, sé que no puedo pensar en todo, pero me gustaría sentirme un poco más cómodo.

¿Con qué frecuencia se ve sorprendido al pasar por alto partes importantes de la funcionalidad al evaluar el alcance de un nuevo software?

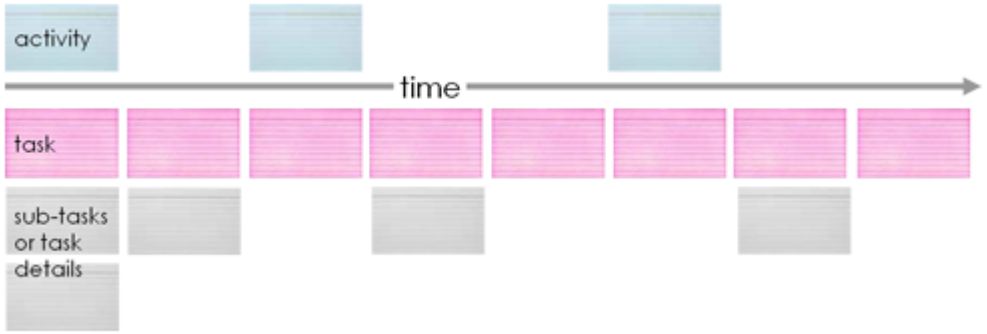
La planificación de lanzamientos con un backlog plano es difícil

En mi caso, en el típico proyecto en el que me involucro, la cantidad de historias en un backlog es de docenas como mínimo, a menudo más de cien. Trabajo duro, especialmente cuando estamos empezando, para mantener el nivel alto de las historias. Incluso entonces, es común terminar con unas 120 historias o más en una primera parte de un backlog. Revisar cada una de ellas y tomar una decisión sobre si entrar o salir es tedioso. Algunas de las horas más miserables de mi vida las he pasado sentado en reuniones con grupos que priorizan su backlog. Uuugh.

Construir un mapa de la historia

Organizar las historias de los usuarios en una forma útil (un mapa) ha funcionado bien para mí.

En realidad, es una idea sencilla. Un pequeño mapa de la historia podría verse así:



En la parte superior del mapa se encuentran las "grandes historias". Las llamo actividades de usuario (tomando prestado el término de expertos en UX como Larry Constantine y Don Norman).

Una actividad es una especie de cosa importante que la gente hace, algo que tiene muchos pasos y no siempre tiene un flujo de trabajo preciso. Si estuviera construyendo un sistema de correo electrónico (algo que nunca sería tan tonto como para hacer), podría tener una actividad llamada: “administrar el correo electrónico”, “configurar servidores de correo electrónico” y “configurar respuestas de fuera de la oficina”.

Una historia para una “actividad” podría ser: **Como consultor, quiero administrar mi correo electrónico para poder mantenerme en contacto con clientes, colegas y amigos** .

Pero esa es una historia demasiado grande para incluirla en una iteración o un sprint.

Esa historia se descompone en otras historias como "enviar mensaje", "leer mensaje", "eliminar mensaje", "marcar mensaje como spam" y cosas así. Yo las llamo tareas de usuario (de nuevo, una palabra que usan los expertos en UX). Para mucha gente de Agile, las "tareas" se refieren a las cosas que hacen los desarrolladores para terminar las historias de usuario. En realidad, una tarea es algo que alguien hace para alcanzar un objetivo. Una tarea de usuario es lo que hacen los usuarios para alcanzar sus objetivos, las tareas de desarrollador son lo que hacen los desarrolladores para crear historias, las tareas Ant son lo que Ant hace para... bueno... hacer lo que sea que estés haciendo con Ant.

Simplemente organizo las cosas pequeñas debajo de las grandes en una especie de cuadrícula.

Siempre me imagino que el tiempo se mueve de izquierda a derecha (porque así es en mi mundo occidental. Mis homólogos de mundos de arriba hacia abajo o de derecha a izquierda pueden traducirlo según sea necesario). Por ejemplo, al organizar historias en el mapa, si una persona que usa el sistema normalmente hace una cosa después de otra,

entonces pondré la primera cosa a la izquierda y la última a la derecha. Hago esto con las cosas grandes y las pequeñas: las actividades y las tareas.

When teaching this, people often tell me "the users can perform these in any order. What order should I put them in?" I'll ask them to "explain to me what the system does at a high level – just tell me the activities." They then recite them to me. "That's the order" I say. In fact, the order you'd explain the behavior of the system in is the correct order. We're building a map that lets us tell a really big story about the system. Build the map in a way that helps you tell the story.

Keep your epics – but stop calling them that because it bothers me

The really big stories can be considered "epics" as Mike Cohn describes them. They're stories – just really big ones – too big to estimate and build. When an epic gets in your backlog, and it's time to discuss it in more detail, I often see people remove the epic from the backlog, decompose it, and replace the pieces they've identified. This is where I cringe. Recall my cutting down the tree and keeping the leaves in a mulch bag story above.

That big story was context. It was my simple way of thinking about the whole activity that people were doing. It's my quick way of explaining to others what the system is about.

And, I hate that word "epic." I haven't written Beowulf here. There's no hero with a magic weapon slaying a monster. It's just my user "managing email" – a relatively basic thing

from my user's perspective. At least the terms "activity" and "user task" give me some idea of what kinds of stories they are. That said, I'm not fond of the term "user story" either, but I've come to accept it. It beats the crap out of requirement. I still have trouble looking a C-level exec in the eye and explaining to him that "we've broken your system down into a series of epics and stories." Upon saying something like that I imagine he's hastily rethinking my daily consulting rate. All the sticky notes and index cards aren't helping either.

Walk your map to test it – to get the big picture

Given a fully assembled map, I have something I can hang on a wall, or lay out on a tabletop and actually have a discussion with. I can walk the map from beginning to end with a user, stakeholder, or developer and tell a story about the users of the system and what they're doing. I can skim along the top of the map, and just touch on the high points. I can dig down into the map to discuss the details.

Talking through the map with users and others helps me find things I've missed. I often hear "you've missed a couple steps here" from users when doing this.

I can annotate the map with pain points or opportunities. As I talk through the map with a user it's common to hear him say, "this here is really a problem with the system today."

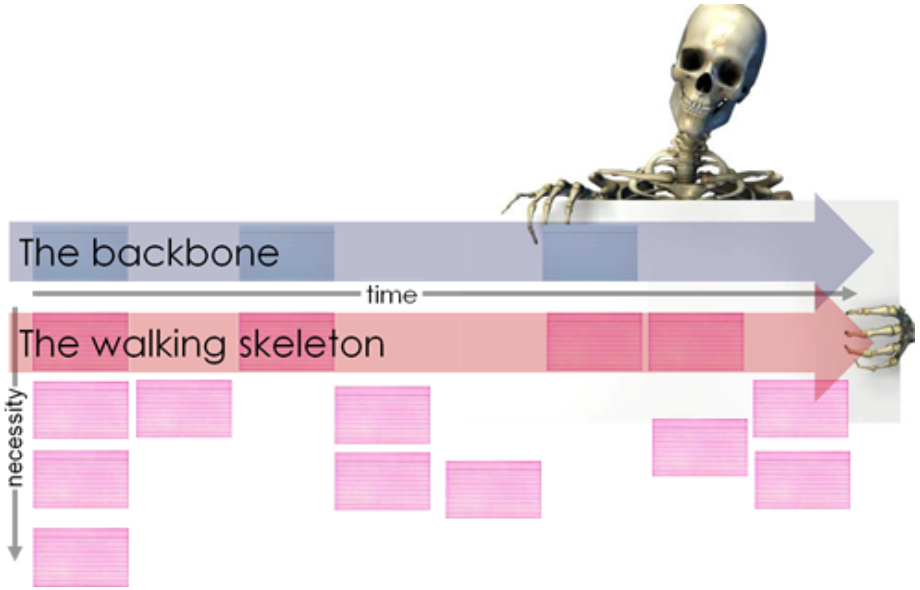
When working with users they often correct me. Yesterday I was working with a potential user of a new system – she was a compliance officer for a major company. I write down a process step as she said it and placed it on the table. I'd been using green cards for big things, yellow for smaller things. She quickly corrected me – "no, that should be a yellow card." In less than an hour's discussion she had caught onto the modeling approach and could, in very few words, supply me with a lot of information about the thing she was doing with the system. She knew yellow was smaller than green, and could tell me that the thing she was doing in the system wasn't as big as I thought it was.

Building and walking story maps leaves me more comfortable than ever before that I "get it" – that I haven't missed something big.

Your software has a backbone and a skeleton – and your map shows it

I find that the big things on the top of the story map look a little like vertebrae. And the cards hanging down look a little like ribs. Those big things on the top are often the essential capabilities the system needs to have. I refer to them as the "backbone" of the software. I stole this term from Dr. Dan Rawsthorne who might use the term slightly differently than I do.

When it comes time to prioritize stories, I don't prioritize the backbone. It just "is." I do prioritize the ribs – the stories hanging down from the backbone. Place them high to indicate they're absolutely necessary, lower to indicate they're less necessary. When you do this, you'll find that all the stories placed high on the story map describe the smallest possible system you could build that would give you end to end functionality. This is what Alistair Cockburn refers to as the "walking skeleton". I always try to build this first.



Plan using your backbone

When it's time to plan releases, it's usually not important to prioritize backbone items against each other. For instance if I were to build a high level backlog for a car it might look something like this:

- engine
- transmission
- brakes
- suspension
- …

It would be stupid to ask stakeholders to prioritize that: "what's more important, the engine or the transmission?" – or "we don't have enough time in this release, could we release without brakes and add them later?" These items are essential – and we'll need all of them to deliver a **minimum viable product** – and **MVP**.

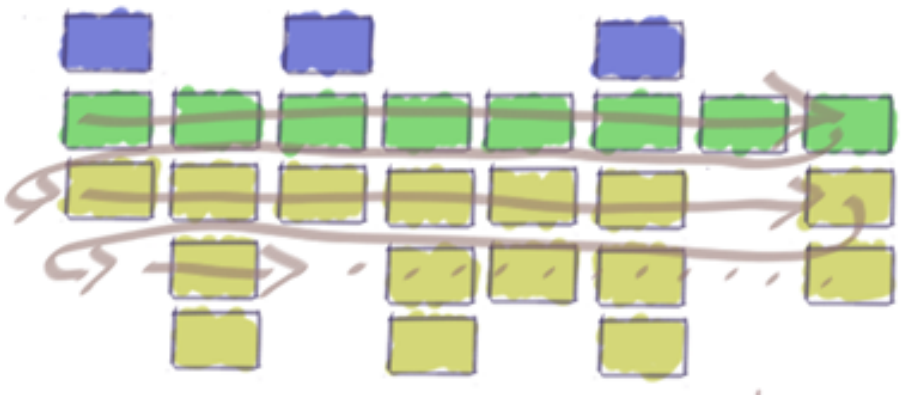
Where the prioritization comes in is below this level: 4-cylinder engine or 6-cylinder engine? brakes with anti-locking or brakes without? sport suspension or not? It's how we build up those backbone items – prioritize their characteristics that matters.

When you prioritize a story map, you'll move cards or stickies up and down to indicate high or low. Lately I take a long strip of masking tape and line off the story map – creating horizontal swim lanes for each release. Then I move the stories up and down into each lane, and even vary their height in the lane.

Keep your map displayed to communicate the big picture

Building a story map helps you initially understand the functionality. Ask yourself when in your project do you not need to understand your functionality any longer? I know it's not a fair question, but I do find some folks that spend a great deal of time building this sort of thing to understand the problem, then tossing it all out in favor of putting stories into a flat-backlog. Cutting down the tree and loading the leaves into a mulch bag.

I find a story map hung as an information radiator becomes a constant point of discussion about the product we're building. When the project is running, it becomes our sprint or iteration planning board. We identify or mark off stories to build in the next iteration directly on the map. During the iteration we'll place just the stories we're working on into a task wall to managing their development – but the story map lives on the planning wall reminding us what the big picture is, and how far we've come.



When we're building software incrementally, story by story, we'll choose them from the story map left to right, and top to bottom. We'll slowly move across the backbone, and down through the priorities of each rib. We're slowly building up the system not a feature at a time, but rather by building up all major features a little at a time. That way we never release a car without brakes.

A different backbone may be in order for adding features to an existing product

When adding features to an existing product, it already has a backbone – and sufficient functionality to have released. I find it useful to stilly identify the backbone to help me get context – to help me see where new features are being placed.

On some projects adding just a few features to a large existing product, it's been difficult to talk people into building up a story map for just a few features. In these cases, I'll simply prioritize the new features, then for each feature build a little story map to prioritize the user stories that make up that features. Each little story map may have 10 or so cards – but they're still arranged left to right, and top to bottom so we can focus on building a little-tiny-walking-skeleton of the features as early as possible.

It's a pattern – not an innovation

Aunque traté de no hacerlo, me indigné cuando escuché a alguien describirme este concepto porque una importante consultoría ahora lo estaba enseñando como una mejor práctica para gestionar los retrasos. "¡Me robaron la idea!", pensé. Pero luego tuve que recordarme a mí mismo que había visto a muchas otras personas haciendo cosas muy similares. Sé que cuando comencé en ThoughtWorks hace muchos años, no había estado trabajando allí mucho tiempo cuando me encontré con Luke Barrett , que había estado construyendo casi exactamente el mismo modelo en colaboración con los usuarios de la misma manera que yo.

Siempre me acuerdo de la prueba de fuego para encontrar un patrón. Si le explicas a alguien un concepto y te dice "¡qué idea tan genial!", no es un patrón. Pero si te dice "¡nosotros también estamos haciendo algo así!", sí lo es. Lo he visto con tanta frecuencia que creo que es un patrón.

Cuando doy clases, me gusta mostrar el modelo mental de Indi Young, que organiza la investigación de usuarios en un flujo similar de izquierda a derecha, con ideas de funciones ubicadas junto al comportamiento del usuario al que son relevantes. Me gusta la cuadrícula de análisis de tareas de Todd Warfel. Es como una vértebra de un mapa de historias con mucho contexto interesante y todas las historias de usuarios ordenadamente ordenadas debajo. Todd usa el color para indicar la prioridad, no la altura.

Obtenga más información sobre los mapas de historias

Si desea obtener más información sobre este tema, considere asistir a una de las conferencias en las que enseñaré esta práctica. Este año, estaré en UI13 de Spool en octubre, en SD Best Practices 2008 en noviembre y en Agile Development Practices 2008 de Better Software en Orlando. Una clase de medio día debería brindarle todo lo que necesita saber para comenzar.

Si estás impaciente, puedes intentar averiguarlo a partir de las diapositivas de mi presentación . Y vuelve a mi artículo original sobre cómo crear mapas de historias .

← Cómo darle sentido a la lista de pendientes de historias de usuario

El desarrollo de Kanban simplificado en exceso →