¿Cómo va tu proyecto?" ¿No es ésta la primera pregunta que hace la gente? Es natural y necesario tener una idea de cómo avanza un proyecto. Algunas personas quieren saber si se terminará a tiempo. Otros han fijado la fecha de entrega y quieren saber cuánta funcionalidad tendrá el sistema. Algunas personas buscan una señal de alerta temprana de que es necesario realizar cambios. Todos quieren mirar hacia el futuro para asegurarse de que obtendrán lo que quieren.

Los gráficos de quemado son un método sencillo para monitorear el progreso del trabajo. Proporcionan una representación visual fácil de comprender del progreso del proyecto. Se pueden extrapolar visualmente para hacer predicciones sobre la fecha de entrega para un alcance fijo, o el alcance para una fecha de entrega fija, como se muestra en las figuras la y lb. Si esto

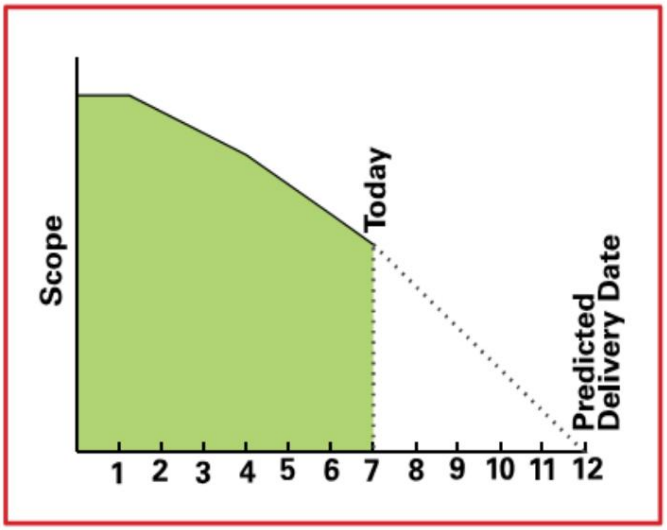


Figure 1a

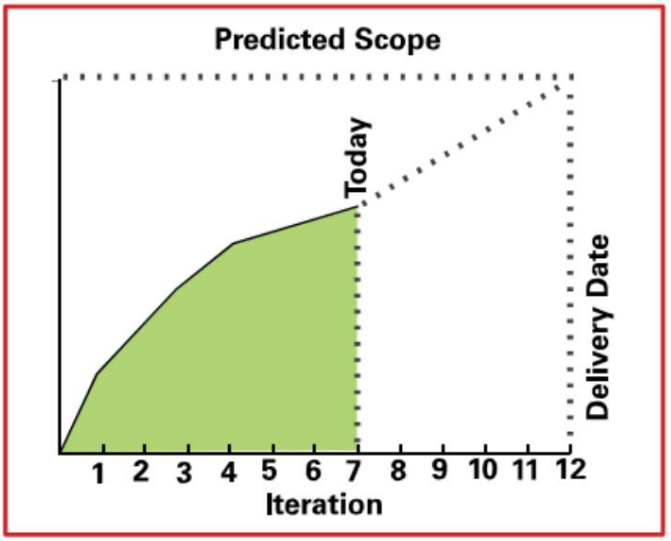


Figure 1b

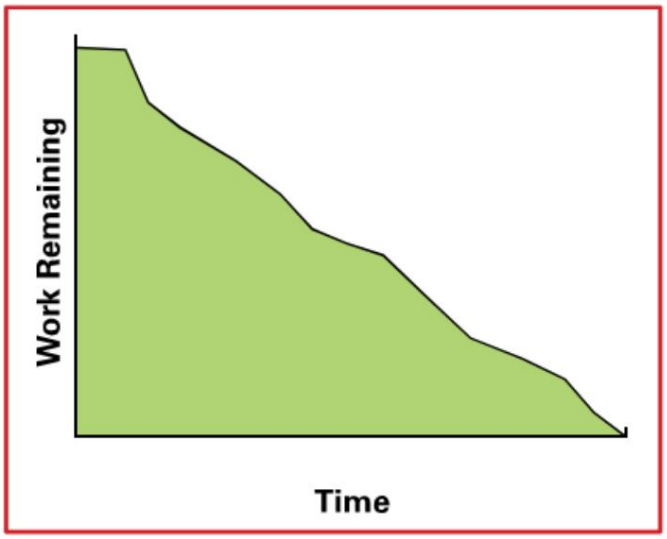


Figure 2

La extrapolación visual es difícil debido a la forma de la línea, entonces el gráfico nos dice otras cosas sobre cómo va el proyecto. Esto puede ser una dificultad para entregar software que funcione o una dificultad para decidir cuál debería ser el software. Examinaremos más de cerca algunos de estos escenarios un poco más adelante. Primero, examinaremos algunas de las variaciones de estos gráficos.

Si tienes una pala y estás esparciendo un montón de tierra, el tamaño del montón restante te muestra cuánto trabajo te queda por hacer. A medida que se quitan paladas de la parte superior de la pila y se distribuyen, la altura de la pila disminuye con el tiempo. Si trazamos la altura de la pila a lo largo del tiempo, veremos algo parecido a la figura 2.

Este es un ejemplo de un gráfico de quemado. En cada incremento de tiempo, trazamos el trabajo restante. Podemos ver el progreso hacia nuestro objetivo final, que es no tener trabajo restante. De hecho, "quemamos" la cantidad de trabajo planificada hasta que no queda nada. Si el ritmo de progreso es consistente, entonces podremos predecir fácilmente cuándo estará terminado el trabajo. Si tenemos una fecha límite fija, veremos si vamos a cumplirla.

Formas de medir el trabajo Hay varias formas de medir el trabajo. Para un físico, el trabajo se mide en julios. Si su gerente le pregunta cuánto trabajo debe hacer Leif en su proyecto y usted responde con un valor en julios... bueno, esperemos que su gerente tenga sentido del humor.

POR HORAS, DÍAS, SEMANAS

Una forma más común de medir el trabajo es con unidades de tiempo. Trabajo ocho horas por el salario de un día. Esta es una medida simple y fácil de entender. Lo que pasa es que no mide lo que crees que mide.

Supongamos que empiezo a trabajar a tiempo pero luego voy a tomar un café mientras espero que mi computadora se inicie. En la cafetería me encuentro con mi jefe y pasamos cuarenta y cinco minutos hablando sobre las políticas de la empresa. Después de eso, inicio sesión en mi correo electrónico para ponerme al día. Algunos de los correos electrónicos requieren respuestas detalladas y se necesitan un par de horas para recopilar la información. Ahora, manos a la obra. Inicio mi IDE, inicio mi servidor local, vuelvo a implementar el código que estoy escribiendo e inicio la aplicación. Pruebo el escenario que aborda mi historia de usuario actual. Ups, hora de almorzar. Inmediatamente después del almuerzo tengo una reunión de dos horas. Bien, ¿dónde lo dejé? Ah, sí, aquí está. Empiezo a codificar, pero noto que la gente se va. La jornada laboral terminó y he hecho ocho horas de "trabajo", pero no he logrado mucho.

Por supuesto, no todos los días son tan malos, pero el tiempo tiene una sorprendente tendencia a pasar desapercibido. No todas las horas corresponden a la misma cantidad de trabajo. No hace falta mucho para que horas (o días, semanas, meses) se conviertan en una forma muy inexacta de medir el trabajo.

La gente no es muy precisa al hacer predicciones, especialmente sobre el futuro. Si estamos haciendo una tarea repetitiva, como esparcir tierra de una pila grande, entonces podríamos estimar cuánto tiempo llevará esparcir el resto de la pila en función de la cantidad que hemos hecho hasta ahora. O, si estamos realizando una tarea que es esencialmente la misma que hemos realizado antes, usamos esa experiencia para estimar cuánto tiempo llevará esta instancia.

Con una tarea compleja como el desarrollo de software, tenemos problemas mayores. El trabajo no avanza a un ritmo constante. Estamos avanzando rápidamente y de repente nos encontramos con un problema que nos detiene en seco. Hay algo que no sabemos y tenemos que resolverlo antes de poder continuar. Esto arruina nuestra predicción de cuánto tiempo llevará la tarea. Sin ningún progreso aparente mientras estamos descubriendo, no podemos saber cuándo terminará el período de estancamiento.

Esto no significa que no puedas tener éxito estimando en horas y días. Al realizar estimaciones en unidades de tiempo y comparar los datos reales con las estimaciones, mejorará sus estimaciones. A nivel individual, eso tiene algún mérito. Pero con un grupo de personas, la posibilidad de llegar a ser realmente bueno en estimaciones disminuye, el costo de hacerlo aumenta y la recompensa es incierta. El objetivo de un equipo de desarrollo de software es crear software valioso. No está claro que una mejor estimación aumentará la cantidad o la calidad del software producido, pero sí está claro que la capacidad de estimación se verá alterada cada vez que haya un cambio en el equipo.

Los partidarios de Scrum recomiendan que el equipo estime cada tarea y luego reestime diariamente la cantidad de trabajo restante en cada tarea para calcular el total de horas de trabajo restantes. [1] Esto puede proporcionar más precisión, pero genera mucho trabajo general y no produce el software más rápido. Y, al consumir tiempo y energía, retrasa la entrega.

Por Story Points, recomiendo un enfoque diferente: dividir el trabajo en historias de usuarios, porciones pequeñas pero funcionales de la aplicación (consulte StickyNotes para obtener más información). Asigne a cada historia de usuario una estimación relativa simple. Por ejemplo, asigna un punto a cada una de las historias simples. Asigne dos puntos a las historias que parezcan dos veces más difíciles. Calcule su trabajo en puntos de la historia y realice un seguimiento del número de puntos de la historia que quedan por hacer. No se "obtiene crédito" por un punto de la historia hasta que esté completamente terminado: codificado, probado, aceptable para el cliente y listo para una posible implementación. Esto le brinda una indicación confiable del trabajo realizado (lo que realmente desea saber) en lugar del esfuerzo invertido.

No se preocupe por mejorar la precisión de sus estimaciones. En un proyecto grande, probablemente haya suficientes perturbaciones aleatorias como para que nunca alcances la precisión que deseas. Recuerde, nuestro resultado deseado es un software que funcione, no estimaciones precisas. Las estimaciones simplemente tienen que ser lo suficientemente consistentes como para utilizarlas con fines de planificación. Si bien nuestras estimaciones pueden estar desviadas en un 50 por ciento o incluso un 100 por ciento, si esperamos lograr la misma cantidad estimada en la próxima iteración que hicimos en esta iteración, estaremos en el objetivo.

Alcance variable

El gráfico de combustión simple utiliza el punto cero del eje y como línea de meta. Esto funciona bien siempre que el objetivo no cambie.

Si no mantiene constante el alcance del trabajo, de vez en cuando es probable que obtenga un gráfico de quemado como el que se muestra en la figura 3. ¿Qué pasó aquí? ¿Por qué la cantidad de trabajo restante aumentó en lugar de disminuir?
¿Se agregó más trabajo?

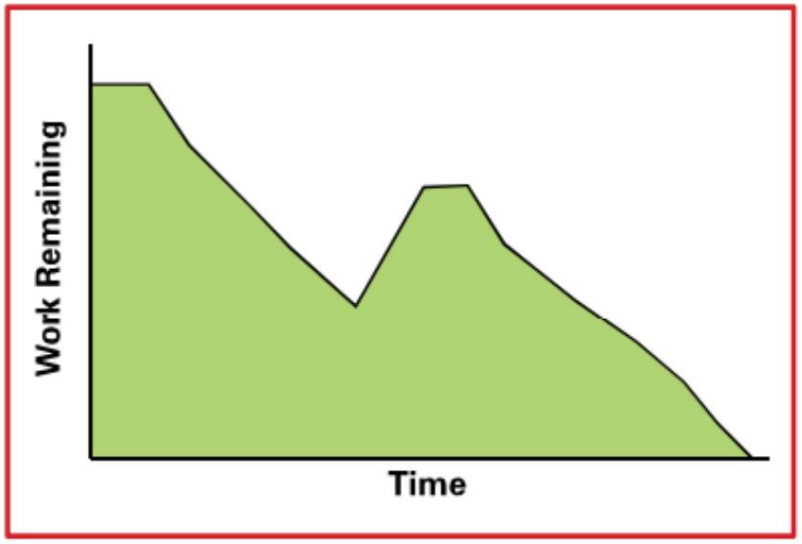


Figure 3

En una sola iteración, esto no debería suceder, pero a veces sucede. En un período de tiempo más largo, generalmente se espera que se agreguen nuevos trabajos.
¿Salieron a la luz tareas imprevistas? Si está quemando horas y reestimando el trabajo restante en cada tarea, como se recomienda frecuentemente en los libros y artículos de Scrum, entonces podría obtener tal cosa.
¿Una historia retrocedió? Quizás se pensó que estaba hecho, pero se descubrió un nuevo escenario y la historia se devolvió a los desarrolladores para que la desarrollaran más. O tal vez los evaluadores encontraron un error que se había escapado del desarrollo.

Cuando el gráfico de quemado se curva hacia arriba de esta manera, ya no se puede predecir cuándo estará terminado el trabajo. Si el alcance cambia, ya no podremos confiar en la línea de meta. Si la medida del trabajo restante resulta poco confiable, entonces estamos perdiendo la capacidad de predecir cuándo estará terminado el alcance actual del trabajo. No podemos extrapolar el gráfico de quemado a la línea de gol.

Burn down chart with a variable floor

Arde bajo cero.

Una forma de separar los cambios en el alcance y el progreso es utilizar un gráfico de evolución con un piso variable, como se muestra en la figura 4.

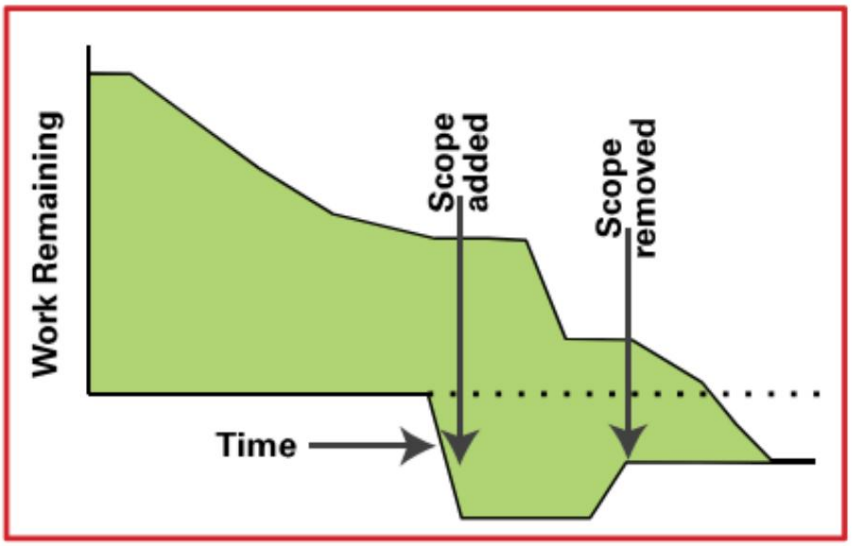


Figure 4

En otras palabras, el trabajo restante se traza como antes, pero la línea de meta puede variar desde el punto cero. Si se agrega alcance, la línea de meta se mueve por debajo del punto cero. Si se elimina el alcance, la línea de meta sube. Podemos ver nuestro progreso continuo, pero la intersección con la meta cambia. O, para completarlo antes, podemos reducir el alcance y acercar el objetivo (consulte las StickyNotes para obtener más información).

Esto me parece un poco complicado de dibujar, especialmente si el alcance cambia mucho. También presupone que conozcas la línea de meta desde el principio, lo cual me resulta difícil de hacer. Generalmente encuentro que la adición de nuevos puntos de la historia continúa hasta bien avanzado el desarrollo, si no hasta el final. Prefiero no usar este gráfico, pero la gente lo ha usado efectivamente cuando se cambia el alcance del trabajo durante una iteración. También prefiero no hacer eso, pero hay ocasiones en las que es apropiado: cuando el equipo ha subestimado y se está quedando sin cosas que hacer, el equipo ha sobreestimado y claramente necesita recortar el alcance, o ocurre una emergencia.

Burn UP

QUEMAR Una

buena manera de indicar el alcance de la variable es utilizar un gráfico de quemado. [2] Esto es como un gráfico de quemado al revés. En lugar de seguir el trabajo, un gráfico de quemado rastrea el trabajo completado. La línea de meta se puede mover y la diferencia entre ella y el trabajo completado le dará una estimación del trabajo restante. En el gráfico de quemado de la figura 5, puede ver la adición de alcance a lo largo del tiempo. Periódicamente, la meta da un paso hacia arriba.

Cada uno de los aumentos en la línea de gol puede representar la definición y estimación de una característica importante como historias. O bien, los aumentos pueden indicar un alcance que se suponía pero que ahora se hace explícito. El alcance del trabajo también se puede ajustar hacia abajo para cumplir con una fecha de lanzamiento particular o porque se está recortando la funcionalidad menos importante.

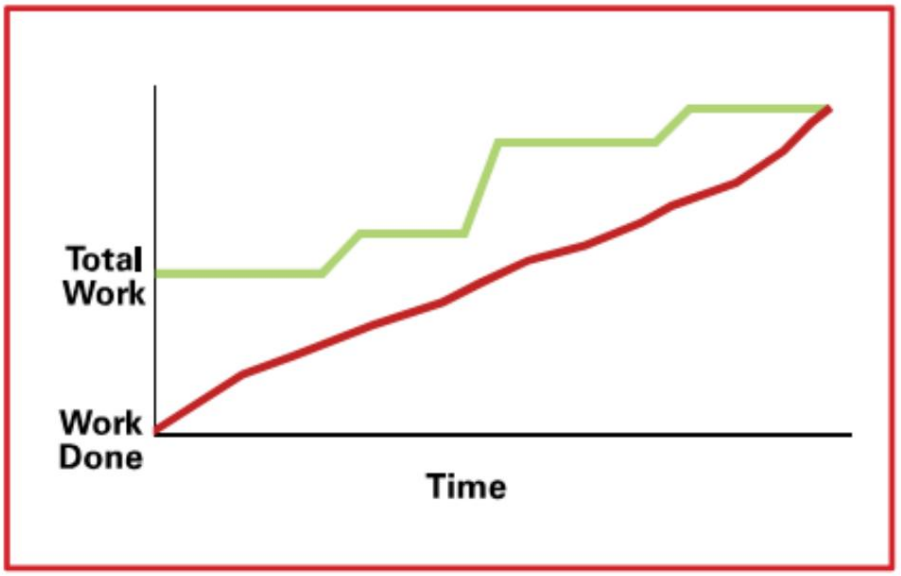


Figure 5

un burn up chart no coincide con la simplicidad de un burn down

Para un ámbito de trabajo fijo, un gráfico de quemado no coincide con la simplicidad de un gráfico de quemado. Pero para el alcance variable, como la mayoría de los proyectos duran un período de tiempo prolongado, el gráfico de avance brinda una indicación clara del progreso hasta el momento y una predicción visual de la fecha de finalización. Mi preferencia cuando hago desarrollo de software iterativo es usar un gráfico de avance (burn down) para una iteración, que tiene un lapso de tiempo corto y un alcance fijo, y un gráfico de avance para la planificación de lanzamientos u otra planificación de proyectos a largo plazo,

burn up

como se muestra. en las figuras 6a y 6b.

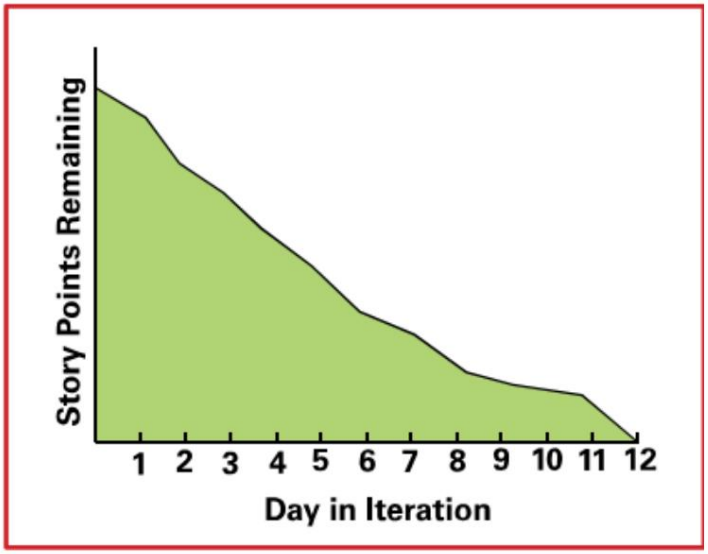


Figure 6a

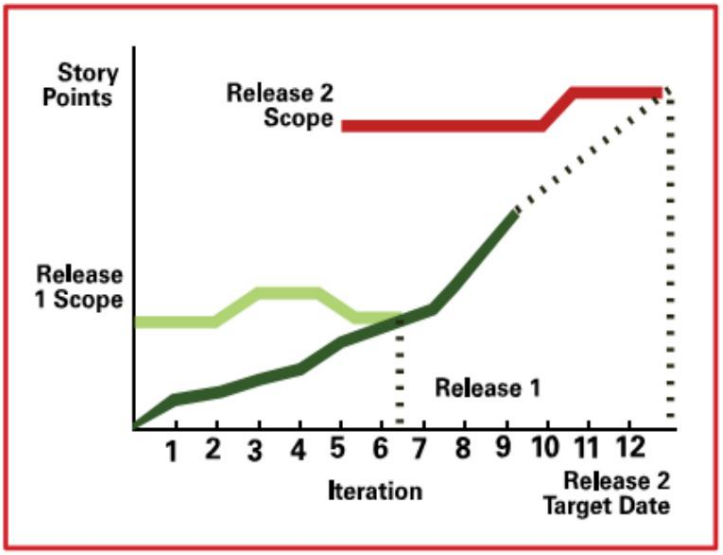


Figure 6b

Leer los gráficos Un gráfico de

quemado o quemado hace más que seguir el progreso. Al examinar el gráfico, podemos hacer inferencias sobre cómo avanza el trabajo. Las Figuras 7a a 7e muestran algunos ejemplos de gráficos de evolución que siguen el progreso en el desarrollo iterativo.

Demasiado TRABAJO La

Figura 7a muestra un gráfico de quemado que no llega a la línea de meta. Al final de la iteración, queda trabajo sin terminar. Esto puede ser un indicador de un problema inusual en esta iteración, pero si sucede con frecuencia, lo interpretaría como una señal de compromiso excesivo.

Con frecuencia, los equipos de desarrollo de software son demasiado optimistas acerca de sus capacidades. Saben cómo hacer las cosas que ven que deben hacerse. Muchas veces no recuerdan ni consideran las veces que se quedaron estancados, ya sea en un problema difícil o esperando alguna dependencia externa. Es posible que ignoren el tiempo que dedican a hacer otras cosas además de

creando código. Y cuando al final se quedan cortos, es posible que se comprometan a trabajar aún más durante la siguiente iteración para ponerse al día.

BOLSAS DE ARENA

Algunos equipos de desarrollo de software van en la otra dirección. Al no querer incumplir sus compromisos y decepcionar a sus partes interesadas, son conservadores en sus compromisos. La Figura 7b muestra un progreso constante y luego aflojar cuando se alcanza la meta parece asegurado. O bien, este gráfico podría significar que las historias posteriores se estimaron bajas en relación con el trabajo que requirieron. O podría significar que la deuda técnica contraída en las historias anteriores ralentizó el desarrollo de las posteriores. A menudo ocurre que un único gráfico de la iteración podría contar una o varias historias. Es posible que necesite consultar otra información para aclarar su comprensión.
Otra posibilidad: un equipo que nunca falta a su compromiso es un equipo que no traspasa los límites. Un equipo de alto rendimiento siempre debe comprobar dónde está la línea entre demasiado y muy poco trabajo. Esto es muy parecido a un marinero que se acerca más al viento hasta que la vela comienza a orzarse y luego se aleja para alcanzar la máxima velocidad. El viento y las olas siempre cambian, por lo que esta prueba debe realizarse de forma repetida y continua. Lo mismo ocurre con el desarrollo de software. Intente esforzarse por un poco más, pero tenga cuidado con las señales de que está tentado a tomar atajos o dejar algo sin terminar. Luego, retroceda un poco.

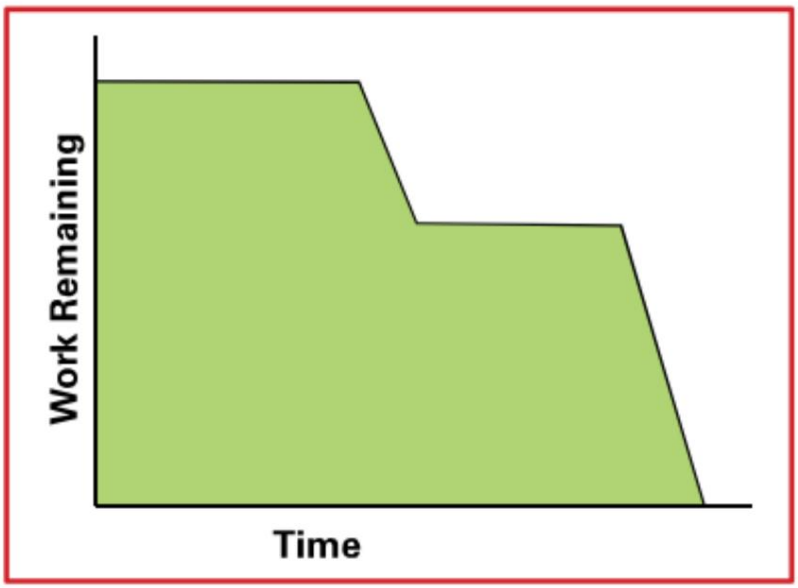


Figure 7c

LA REGLA DE TRES Gerald M.

Weinberg, en su libro The Secrets of Lonsulting, establece la Regla de Tres: Si no puedes pensar en tres cosas que podrían salir mal en tus planes, entonces hay algo mal en tu forma de pensar. . Esta regla es aplicable a mucho más que la planificación. Se aplica particularmente a la lectura de gráficos quemados.
Las posibilidades que ofrezco son comunes que he visto, pero se ofrecen sin ningún conocimiento del contexto de su proyecto. Necesitará discernir qué podrían decirle sus gráficos de quemado. Una vez que tengas una respuesta, intenta pensar en al menos dos más. Luego, armado con estas posibilidades, intenta comprobarlas para ver cuáles, si las hay, reflejan la realidad que te rodea.

HISTORIAS DEMASIADO

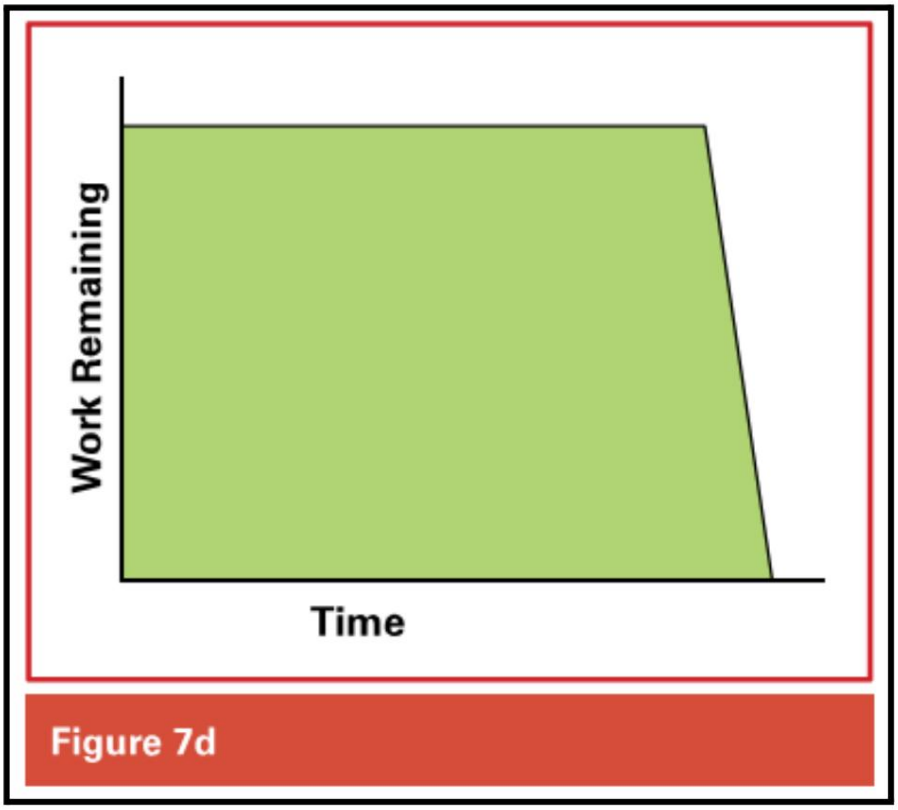
GRANDES La Figura 7c muestra una imagen en la que no hay indicios de progreso durante mucho tiempo; Luego, de repente, hay una gran parte cumplida de una vez. La causa más probable de esto son las “grandes historias”. A menudo, los equipos acostumbrados a un ciclo de vida de proyectos en serie (con definición de requisitos, diseño, implementación y luego pruebas) tienen dificultades para dividir el trabajo en historias pequeñas pero valiosas. En lugar de ello, crean “minicascadas” en porciones más grandes de trabajo y desglosan el trabajo por capas arquitectónicas en lugar de por funcionalidad. Como resultado, agrupan una gran cantidad de trabajo en una sola unidad. Trabajar en grandes porciones hace que sea difícil ver un progreso real. Algunas personas intentan contrarrestar esto estimando el progreso de trabajos inacabados. Es fácil engañarse acerca de la canti

progreso, sin embargo. Existe el peligro de crear la antigua situación de haber realizado el 90 por ciento del trabajo pero el 10 por ciento restante requiere el 90 por ciento del tiempo. Es mucho más confiable juzgar el progreso por la funcionalidad que se puede probar fácilmente para ver si está completa o no. Me gusta ver el progreso todos los días. De lo contrario, creo que necesito buscar un cuello de botella que detenga el progreso. Este es sólo un pequeño ejemplo de cómo los desarrolladores "se vuelven oscuros" y nadie más sabe lo que está sucediendo.

Las historias extensas también impiden que el propietario del producto dirija eficazmente el desarrollo del producto. Una historia así puede contener muchos pequeños detalles que “es bueno tener” en lugar de “imprescindibles”. Si todos se agrupan con funcionalidades imprescindibles, el propietario del producto debe aceptar o rechazar la historia por completo. Esto significará poner estos elementos interesantes en el producto antes que los imprescindibles en otras historias.

Lo que realmente funciona, una vez que aprendes a hacerlo, es crear fragmentos muy finos de funcionalidad. Los llamo cortes verticales porque atraviesan las capas arquitectónicas desde la interfaz de usuario hasta la base de datos. (Esta es la descripción típica de capas para sistemas empresariales, pero otros tipos de software tendrán capas diferentes). A esto se le ha llamado “un esqueleto andante” [3] o “disparar balas trazadoras a través de la aplicación”. [4]

Cada fina astilla no hace mucho, pero funciona en todos los sentidos. Además de estas porciones de funcionalidad, puede haber historias que solo modifiquen la funcionalidad existente.



”El trabajo inacabado tiene un costo (el trabajo realizado hasta ahora) pero ningún valor (no podemos entregárselo al cliente). Es mucho mejor terminar un trabajo pequeño que empezar uno grande”.

GRAN TRABAJO EN PROGRESO La Figura

7d podría estar diciéndonos que hay una gran historia, el peor de los casos de “historias demasiado grandes”. O podría estar diciéndonos que todas las historias se están trabajando simultáneamente. Cualquiera de estos es un indicador de que se está realizando mucho trabajo al mismo tiempo. Otra alternativa, que los desarrolladores estuvieran haciendo otro trabajo en lugar de las historias solicitadas, también produciría un gráfico quemado como este.

¿Cuál es el problema con esto? Un problema es que no estás seguro de completar las historias. ¿Qué pasa si llega al final de la iteración y todas las historias están “90 por ciento terminadas”, pero no hay ningún valor comercial funcional para entregar al propietario del producto?

El trabajo inacabado tiene un coste (el trabajo realizado hasta el momento) pero no tiene valor (no podemos entregárselo al cliente). Es mucho mejor terminar una pequeña cantidad de trabajo que comenzar una gran cantidad. Al dividir la historia en porciones pequeñas y funcionales, podemos ver el progreso que se está realizando. Si luego empezamos a trabajar en todas esas pequeñas historias a la vez, perdemos esa ventaja. Es mejor que el equipo se afane en cada historia para terminarla y luego comenzar con la siguiente. El objetivo debe ser tener la menor cantidad de historias en progreso a la vez en las que puedas trabajar productivamente.

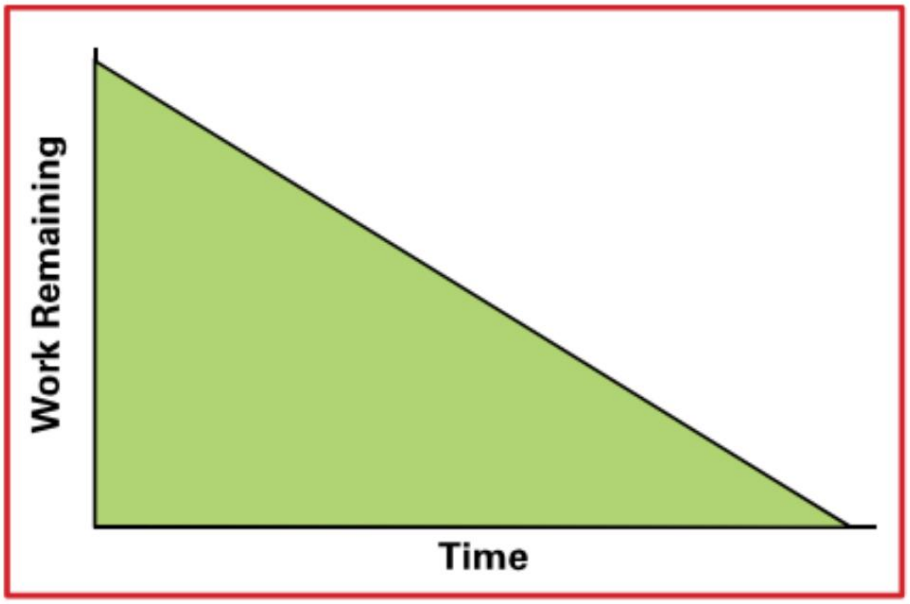


Figure 7e

COCINA LOS LIBROS

Uno de los gráficos de quemado más insidiosos es aquel en el que la línea de progreso es una flecha recta de principio a fin, como se muestra en la figura 7e. Si bien esto puede suceder ocasionalmente, lo más probable es que signifique que alguien está manipulando los libros. Es posible que esté informando del progreso que percibe que la gente quiere ver en lugar de una medida honesta de cómo avanza el proyecto. Esto es especialmente fácil de hacer cuando se realiza un seguimiento del progreso en horas de tarea en lugar de puntos de la historia. También se puede hacer tomando atajos en las historias y afirmando que ya están terminadas cuando el código todavía es un desastre internamente.

¿Por qué sucedería esto? Puede haber una serie de factores contribuyentes:

- La persona (o el software) que dibuja el gráfico de evolución puede marcar una línea para poder decir fácilmente si el desarrollo se está quedando atrás o adelantándose a la tasa promedio esperada. Otros pueden interpretar esa línea como la tasa "ideal".

- El equipo puede sentir que está siendo juzgado según el gráfico, por lo que los miembros del equipo quieren que se vea bien. • Puede ser que el progreso se esté midiendo más por el esfuerzo que por los logros. Los sistemas que calculan automáticamente el tiempo restante de una tarea son propensos a esto.

Cuando el foco se convierte en un gráfico de evolución "bonito", se pierde la capacidad de utilizarlo para comprender y gestionar el desarrollo.

Es su elección Los gráficos

de grabación y grabación ofrecen una variedad considerable en las formas en que pueden mostrar el progreso para que pueda entenderse de un vistazo. Son altamente adaptables a su contexto: puede realizar un seguimiento del trabajo con cualquier medida que tenga sentido en su entorno. Puede realizar un seguimiento durante períodos de tiempo cortos o largos, o ambos. ¿Tiene un alcance fijo? Entonces una quema es una buena opción. ¿Tiene un alcance bien comprendido con ajustes relativamente pequeños? Sería apropiado utilizar el gráfico de quemado de piso variable o un gráfico de quemado. ¿Está trabajando en un flujo continuo y continúa agregando trabajo pendiente a medida que se desarrolla? Entonces, un gráfico de quemado es probablemente la mejor opción.

El punto importante es que su gráfico de quemaduras debe reflejar una realidad objetiva, no deseos y esperanzas. Debe construirse con datos medidos, no estimados. Publique su gráfico en un lugar destacado donde lo pueda consultar todos los días. De esta manera, puede indicarle de un vistazo si es probable que sus deseos y esperanzas se hagan realidad. Si le dice algo más, le dará las pistas que necesita para tomar alguna medida correctiva.