PLANIFICACIÓN Y ESTIMACIÓN ÁGILES

COHN

PARTE I
EL PROBLEMA Y LA META

ESTIMAR Y PLANIFICAR DE FORMA ÁGIL

- La estimación y la planificación deben ser ágiles. Sin estimaciones y planificación ágiles no tenemos proyectos ágiles.
- Planificar:
  - qué deberíamos construir y para cuándo
- Incluye:
  - cuán grande es (estimaciones)
  - cuándo se debe hacer y cuánto puedo tener hecho para cuándo (cronograma)
- Estimar y planificar son actividades de todo el equipo.

CAP. 1
EL PROPÓSITO DE PLANIFICAR

EL PROPÓSITO DE PLANIFICAR

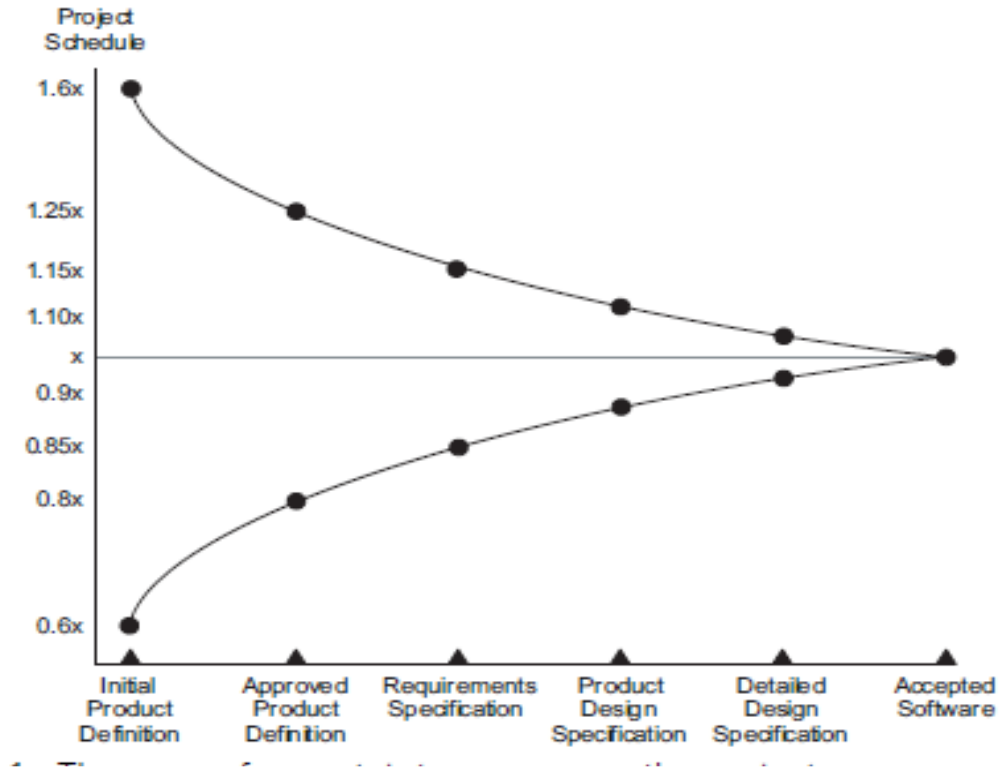
- Llegar, de forma iterativa, a una respuesta optimizada de la pregunta principal acerca del desarrollo de un nuevo producto: qué se debería construir.
- Es decir, qué capacidades debería exhibir el producto, en que plazo y con cuántos y cuáles recursos.
- Estimar y planificar son críticos para el éxito de cualquier proyecto de desarrollo de sw de cualquier tamaño e importancia.
- Los planes:
  - guían las decisiones de inversiones. p. ej.:
    - podríamos iniciar un proyecto si estimamos que llevará 6 meses y costará diez mil dólares,
    - pero lo rechazaríamos si pensáramos que llevaría 2 años y costaría 40 mil dólares.
  - nos ayudan a saber quién tiene que estar disponible para trabajar en el proyecto en un período dado.
  - nos permiten saber si un proyecto va en buen camino de entregar la funcionalidad que los usuarios necesitan y esperan.

ERRORES AL PLANIFICAR

- Planificar es difícil. Los planes a menudo salen mal.
- Los equipos responden a esto yéndose a uno de dos extremos:
  - no planifican nada o
  - sobreplanifican.
- El equipo que no planifica no puede responder las preguntas más básicas tales como: ¿cuándo terminarán? y ¿podemos programar la liberación del producto para junio?
- No podemos excusarnos de planificar porque sea difícil.

- El equipo que sobreplanifica se convence de que cualquier plan está bien. El plan puede ser más exhaustivo, pero no necesariamente más preciso o útil.

EL CONO DE INCERTIDUMBRE - BOEHM



- Las estimaciones en etapas tempranas de un proyecto son mucho menos certeras que las que se hacen más adelante en el proyecto.

EL CONO DE INCERTIDUMBRE - PMI

- Para el PMI, el cono de incertidumbre es asimétrico:
  - Un orden inicial de magnitud de la estimación entre un +75 % y un -25 % (o sea 175 % a 75 %).
  - La siguiente estimación es la estimación presupuestal (*budgetary estimate*), con un rango de +25 % a -10%.
  - Y la estimación final definitiva, con un rango de +10 % a -5 %.

RAZONES PARA PLANIFICAR

- Si estimar o planificar es difícil y es imposible lograr una estimación precisa hasta bien avanzado el proyecto, ¿para qué planificar?
- Razones de la organización:
  - Planificar campañas de marketing
  - Armar el cronograma para la liberación del producto
  - Capacitar usuarios internos
  - etc.
- En enfoques con planificación continua, planificar = búsqueda de valor.
- Planificar es buscar una solución óptima a la pregunta: ¿Qué deberíamos construir?
- Considerar:
  - *features*
  - recursos
  - cronograma

UNA BUENA PLANIFICACIÓN

- Una buena planificación ayuda responder la pregunta: ¿qué debemos construir?, puesto que
  - reduce los riesgos
  - reduce la incertidumbre
  - apoya una mejor toma de decisiones
  - establece confianza
  - transmite información

REDUCE RIESGOS

- Planificar aumenta la probabilidad de éxito del proyecto al proporcionar percepciones de los riesgos del proyecto.
- Las discusiones al estimar hacen surgir preguntas que exponen aspectos oscuros del proyecto. P. ej. si nos piden estimar cuánto llevara integrar el nuevo proyecto con un sistema legado existente del que nada sabemos, eso expondrá los *features* de integración como un riesgo potencial. Y lo podremos gestionar.

REDUCE LA INCERTIDUMBRE

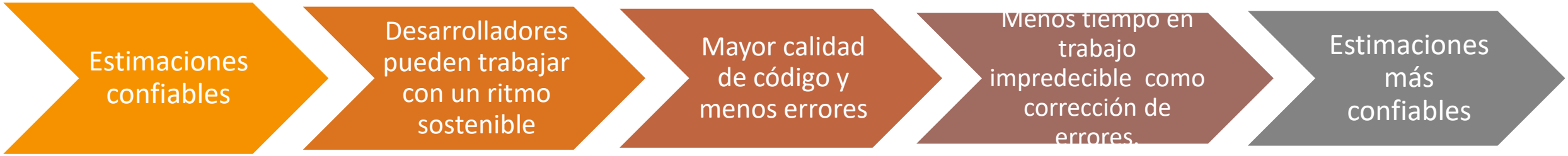
- Un proceso de planificación iterativo ayuda al equipo a refinar su visión del producto.
- El riesgo más crítico es el de desarrollar el producto equivocado. Un enfoque ágil de planificación puede reducirlo o eliminarlo.
- Un proyecto que entrega todos los *features* que estaban en el plan inicial no es necesariamente exitoso. Puede que, una vez empezado el proyecto, no valga la pena desarrollar un *feature* definido originalmente.

APOYA UNA MEJOR TOMA DE DECISIONES

- Las estimaciones de valor, cronograma, costo y recursos ayudan
  - a decidir si vale la pena hacer un proyecto particular o si comenzar o no el proyecto:
    - ¿será tan largo que perderemos la ventana de mercado?
    - ¿costará más de lo que aportará financieramente?
    - ¿necesita recursos clave que están comprometidos en otros proyectos?
  - a asegurarnos que estamos trabajando en los proyectos de mayor valor posible.
  - a tomar decisiones negociando entre funcionalidad, costo y plazo:
    - ¿vale la pena retrasar la release por este *feature*?
    - ¿Debemos contratar otro desarrollador para que este *feature* entre en la release?
    - ¿Debemos comprar esa herramienta de desarrollo?

ESTABLECE CONFIANZA

- La entrega frecuente confiable de los *features* prometidos crea confianza entre los desarrolladores y el cliente.
- Estimaciones confiables permiten entrega confiable.
- El cliente necesita las estimaciones para tomar decisiones de priorización y concesiones.
- Las estimaciones también ayudan al cliente a decidir cuánta funcionalidad desarrollar. En vez de invertir 20 días y obtener todo, quizás 10 días de esfuerzo darán un 80 % del beneficio.
- Los cliente son reacios a tomar este tipo de decisiones concesivas en etapas tempranas del proyecto, a menos que se haya probado que las estimaciones de los desarrolladores son dignas de confianza.



TRANSMITE INFORMACIÓN

- Un plan transmite las expectativas y describe una posibilidad de lo que puede ocurrir durante el curso del proyecto.
- El plan no garantiza un conjunto exacto de *features* en una fecha exacta a un costo especificado.
- El plan comunica y establece un conjunto básico de expectativas. Demasiado frecuentemente se lo reduce a una única fecha y se olvidan todas las asunciones y expectativas que llevaron a esa fecha.

¿QUÉ HACE QUE UN PLAN SEA BUENO?

- Un buen plan es uno suficientemente confiable como para ser usado como base para tomar decisiones sobre el producto y el proyecto. P. ej.
  - Si el producto se va a liberar en el segundo o en el primer semestre.
  - Decir que contendrá un conjunto determinado de funcionalidades.
  - Preparar materiales de marketing.
  - Agendar una campaña publicitaria.
  - Asignar recursos para asistir en la actualización de clientes clave, etc.
- El plan será útil mientras prediga lo que realmente pasa en el proyecto.
  - Si el desarrollo lleva 12 meses en lugar de los 6 planificados, no es un buen plan.
  - Si lleva 7 en lugar de los 6 estimados, está dentro del margen de error del PMI. Aunque el plan no es certero, fue incluso más útil si lo actualizaron durante el proyecto y si se supo con antelación del atraso en la entrega.

EN QUÉ CONSISTE LA PLANIFICACIÓN ÁGIL

- Se enfoca en la planificación y no en el plan
- Alienta el cambio
- Construye planes que puedan ser cambiados con facilidad
- Se realiza a lo largo del proyecto

FOCO EN LA PLANIFICACIÓN Y NO EN EL PLAN

- Se enfoca más en la planificación que en la creación del plan:
  - Planes = documentos o figuras, fotografías de cómo creemos que el proyecto se desarrollará en el futuro.
  - Planificar = una actividad.
- Busca un equilibrio entre
  - el esfuerzo y la inversión en planificación y
  - el saber que el plan será revisado a lo largo del proyecto.

ALIENTA EL CAMBIO

- «Un plan ágil es aquel que no solo estamos dispuestos, sino ansiosos por cambiar».
- Por qué cambiar:
  - porque aprendimos algo
  - para evitar una equivocación.
- P. ej.: aprendimos que
  - los usuarios quieren más de esta funcionalidad o menos de aquella
  - la usabilidad es más importante de lo que creíamos
  - programar en este nuevo lenguaje lleva más de lo esperado.
- Se puede evaluar el impacto financiero de cada cambio. SI vale la pena, se puede alterar el plan y el cronograma.

CONSTRUYE PLANES QUE PUEDAN SER CAMBIADOS CON FACILIDAD

- Necesitamos planes que puedan ser cambiados con facilidad.
- Por eso, la planificación es más importante que el plan:
  - El conocimiento ganado en la planificación persistirá más tiempo que el que dure un plan que será actualizado o cambiado por otro.
  - Un plan ágil es uno que es fácil de cambiar.
- Podemos cambiar el plan sin cambiar las fechas:
  - Eliminar una funcionalidad del alcance
  - Reducir el alcance de una funcionalidad
  - Agregar personal al proyecto
  - Etc.

SE REALIZA A LO LARGO DEL PROYECTO

- Dado que no podemos definir por completo un proyecto en su inicio, no debemos hacer toda la planificación en el inicio.
- La planificación ágil se hace de forma pareja a lo largo de todo el proyecto.
  - Planificación de la versión (*release*)
  - Planificación de la iteración

CAP. 2
POR QUÉ FALLA LA PLANIFICACIÓN

PROBLEMAS DE LOS ENFOQUES TRADICIONALES DE PLANIFICACIÓN

- Casi 2/3 de los proyectos se pasan de las estimaciones de costo por mucho (Lederer, 1992).
- 64 % de los *features* incluidos en los productos se usan rara vez o nunca (Standish, 2002).
- El proyecto promedio se pasa del cronograma en un 100 % (Standish, 2001).

CAUSAS DEL FRACASO DE LOS ENFOQUES TRADICIONALES DE PLANIFICACIÓN

1. Se planifica por actividad y no por *feature*
2. La multitarea
3. No se desarrollan los *features* por prioridad
4. No se toma en cuenta la incertidumbre
5. Las estimaciones se convierten en compromisos

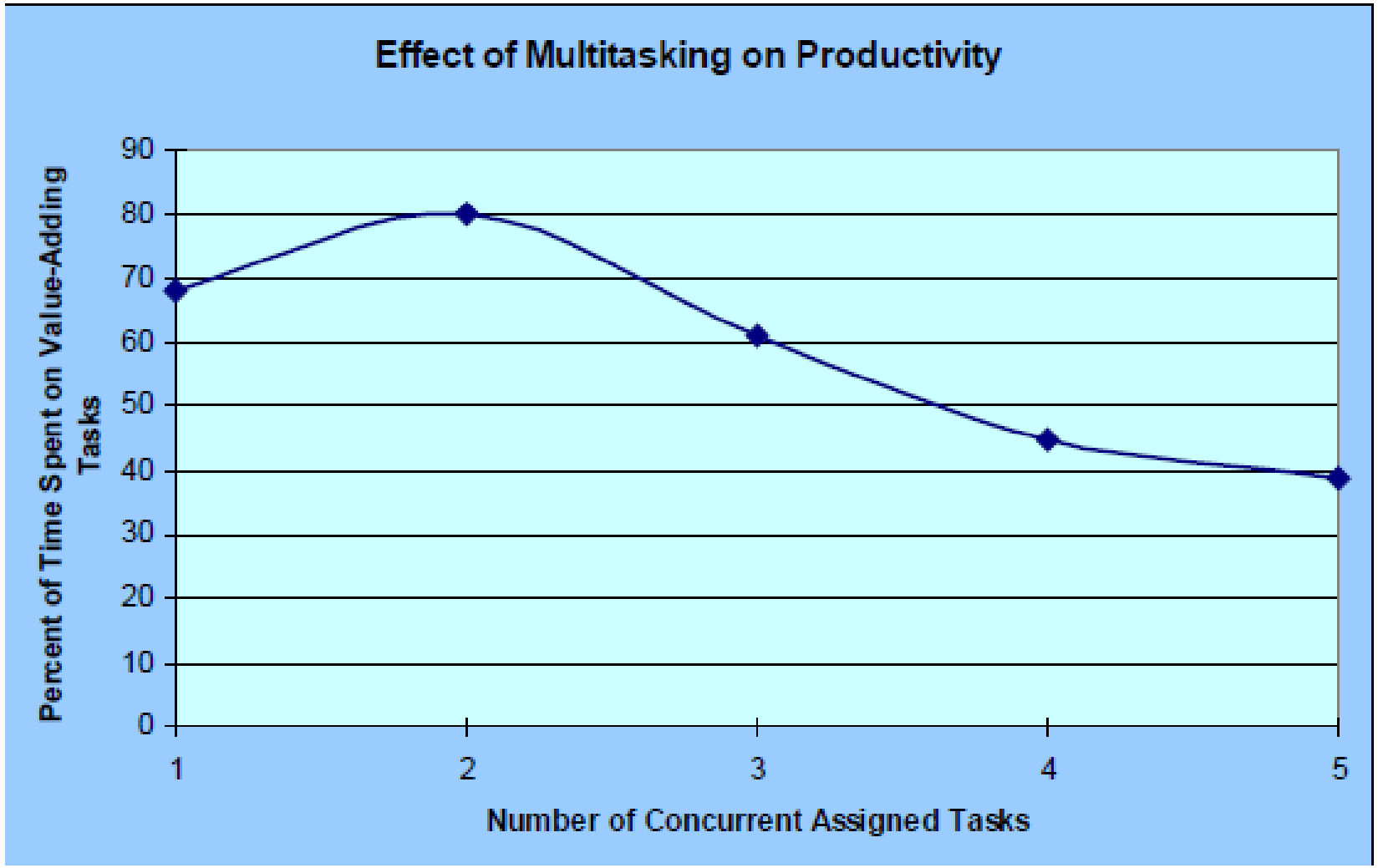
SE PLANIFICA POR ACTIVIDAD EN LUGAR DE POR *FEATURE*

- La planificación tradicional se enfoca en completar actividades (WBS, Gantt) y no en entregar *features*.
- Pero los clientes no obtienen valor de la compleción de actividades Los *features* son las verdaderas unidades de valor para el cliente.
- Al revisar un plan basado en actividades buscamos actividades faltantes y no *features* faltantes.
- Los planes basados en actividades a menudo se atrasan, debido a que
  - Las actividades no se terminan temprano.
  - El retraso se arrastra en el cronograma.
  - Las actividades no son independientes (la duración de una influye en la duración de otra)
- Eso lleva a que los equipos traten de ahorrar tiempo reduciendo la calidad
- o que restrinjan los cambios al producto, incluso si son cambios muy valiosos.

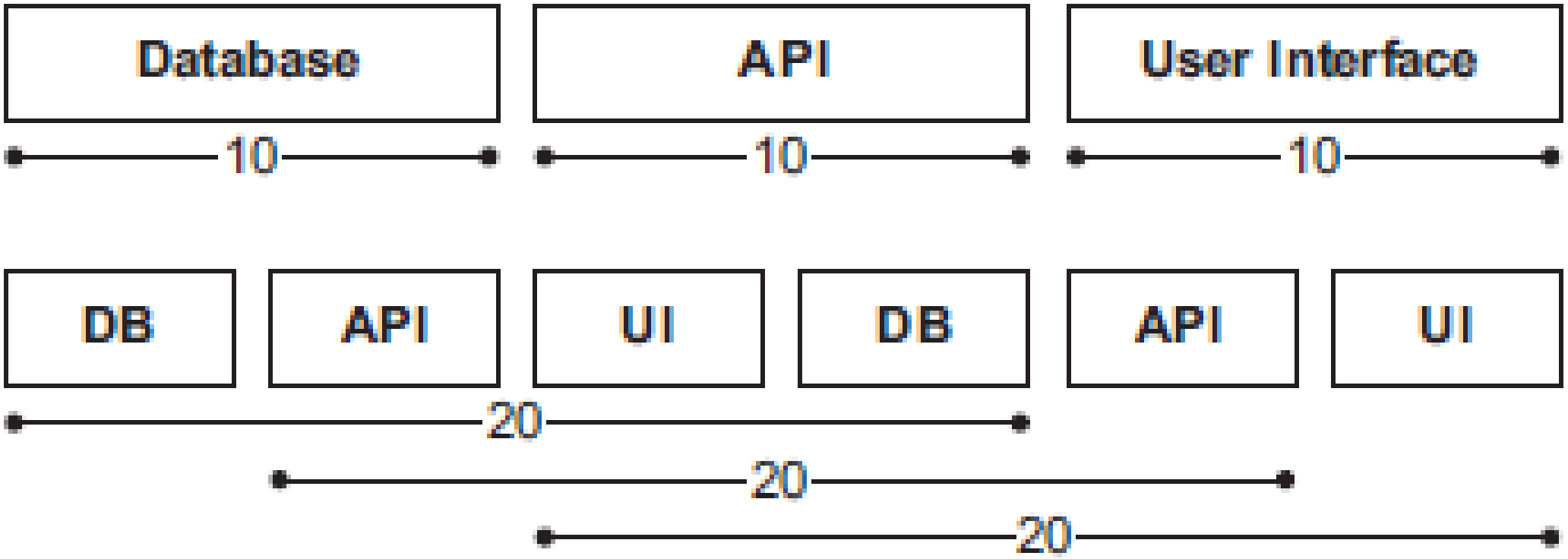
LA MULTITAREA CAUSA RETRASOS

- Los miembros del equipo ven la multitarea como una posible solución, pero finalmente se atrasan aún más debido a los costos ocultos de la multitarea.
- Cuando el proyecto va muy atrasado, inevitablemente se abandonan *features*.

EFECTO DE LA MULTITAREA SOBRE LA PRODUCTIVIDAD



EFECTO DE LA MULTITAREA EN LOS PLAZOS DE ENTREGA



NO SE DESARROLLAN LOS *FEATURES* POR PRIORIDAD

- Puesto que los *features* se desarrollan en el orden que los desarrolladores consideran más conveniente, los *features* que se dejan de lado no son necesariamente aquellos de menor valor para los usuarios.

NO SE TOMA EN CUENTA LA INCERTIDUMBRE

- Incertidumbre del producto:
  - pensamos que el primer análisis de requisitos es completo y perfecto y que el cliente no lo va a cambiar.
  - No se incluyen funcionalidades que se identificarán después de la creación del plan.
- Incertidumbre sobre cómo vamos a desarrollar el producto:
  - no podemos esperar identificar todas las actividades que se van a hacer durante el proyecto.
  - Las estimaciones son imprecisas.
  - Probablemente falten actividades en el plan.
- Todo esto aumenta la probabilidad de que
  - el proyecto se retrase
  - se dejen de lado *features* al final
  - se hagan concesiones inapropiadas relativas a la calidad.

SUGERENCIAS

- Para reducir la incertidumbre:
  - iteraciones cortas
  - mostrar o liberar funcionalidad al cliente cada pocas semanas
- Expresar la fecha de finalización del proyecto como un rango.

LAS ESTIMACIONES SE TOMAN COMO COMPROMISOS

- En cada estimación hay implícita una probabilidad de terminar el trabajo en el tiempo estimado.
- Muchas organizaciones confunden estimaciones con compromisos.
- Apenas un equipo expresa una estimación, se ven forzados a comprometerse con ella.

CAP. 3
UN ENFOQUE ÁGIL

UN ENFOQUE ÁGIL DE LOS PROYECTOS

- Los equipos ágiles
  - trabajan como un solo equipo
  - trabajan en iteraciones cortas
  - entregan algo al final de cada iteración
  - se enfocan en las prioridades del negocio
  - inspeccionan y se adaptan

TRABAJAR COMO UN SOLO EQUIPO

- Los equipos ágiles trabajan juntos como un equipo, pero incluyen roles que son asignados a personas específicas.
  - *product owner:* es responsable de la visión del producto y de priorizar los *features* en los que trabajará el equipo
  - cliente: es la persona que paga por el proyecto o que compra el sw
  - usuarios
  - desarrolladores (programadores, verificadores, analistas, administradores de bd, expertos en usabilidad, especialista técnicos, arquitecto, diseñadores, etc.)
  - gerente de proyecto

TRABAJAR EN ITERACIONES CORTAS

- Iteraciones cortas (generalmente de 1 a 4 semanas, pero hasta 3 meses).
- Todo el trabajos se hace concurrentemente dentro de cada iteración (requisitos, diseño, codificación, verificación, etc.).
- Tienen largo determinado (fijo o se define al comienzo de la iteración).

ENTREGAR ALGO AL FINAL DE CADA ITERACIÓN

- Entregar un producto codificado, verificado y potencialmente entregable al final de cada iteración.
- No se va a liberar necesariamente.
- Una única iteración no alcanza para completar funcionalidad que satisfaga los deseos del cliente → *release*
- Una *release* comprende una o más iteraciones para construir un conjunto de funcionalidad relacionada
- 2 a 6 meses.

FOCO EN LA PRIORIDADES DEL NEGOCIO

- Los *features* a desarrollar en cada iteración deben seleccionarse en base a la prioridades del negocio. Así se asegura que los *features* más importantes se desarrollen primero.
- Para dar más flexibilidad a la priorización, escribir los *features* de manera que se minimicen las dependencias técnicas.
- Foco en completar y entregar *features* de valor para el usuario y no en completar tareas aisladas. Una forma de lograr esto es trabajar con historias de usuario.
- Historias de usuario:
  - breve descripción de funcionalidad como la ven el usuario o el cliente.
  - Sin formato determinado, pero útil pensarlas como:
  - Como un <tipo de usuario>, quiero <capacidad> de forma tal que <valor de negocio>.
  - enfoque de requisitos *just-in-time:*
    - breve descripción de la historia de usuario en una tarjeta o en computadora
    - conversaciones entre desarrolladores y el *product owner* (comunicación verbal) siempre que se necesite.

INSPECCIONAR Y ADAPTARSE

- Adaptar los planes siempre que sea necesario.
- P. ej. cambiar prioridades de una iteración a otra, maximiza el ROI.

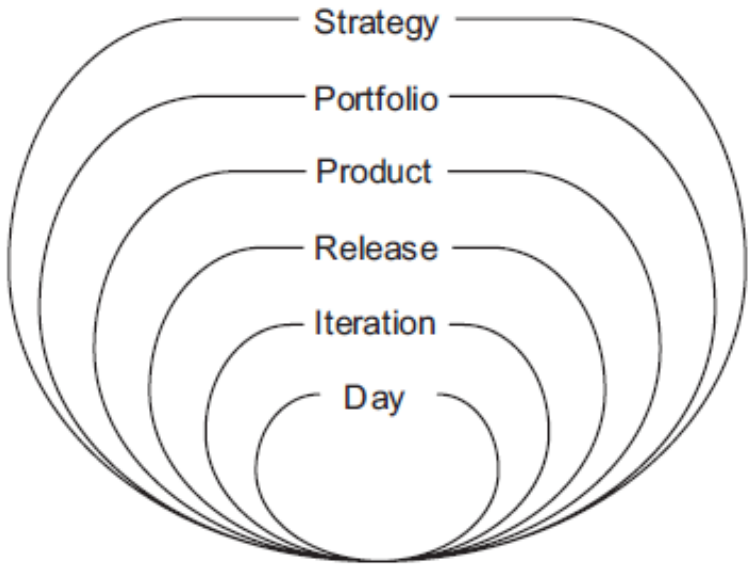
UN ENFOQUE ÁGIL A LA PLANIFICACIÓN

GENERACIÓN DE CONOCIMIENTO

- Los proyectos deberían ser vistos no como que ejecutan una secuencia de pasos, sino como que generar de forma rápida y confiable un flujo de nuevas capacidades y nuevos conocimientos útiles.
- Los proyectos generan dos tipos de conocimiento nuevo:
  - acerca del producto: nos ayuda a saber más de cómo debería ser el producto
  - acerca del proyecto: información sobre el equipo, las tecnologías usadas, los riesgos, etc.
- A menudo fallamos en reconocer y planificar para este nuevo conocimiento. Asumimos que sabemos todo lo necesario para crear un plan exacto.

MÚLTIPLES NIVELES DE PLANIFICACIÓN

- Los equipos ágiles planifican en 3 niveles:
  - **planificación de la *release***
    - El plan de la release abarca la duración de la *release* (típicamente de 3 a 6 meses).
  - **planificación de la iteración**
    - Un plan de la iteración abarca solo la duración de una iteración (típicamente 1 a 4 semanas).
  - **planificación diaria**
    - Un plan diario es el resultado de los compromisos que los miembros del equipo se hacen unos a otros durante la reunión diaria.



PLAN DE LA RELEASE

- Meta: determinar el alcance, el cronograma y los recursos para la release.
- Al inicio de la release, pero es actualizado al comienzo de cada iteración, tal que siempre refleje las expectativas actuales de lo que va a incluirse en la release.

PLAN DE LA ITERACIÓN

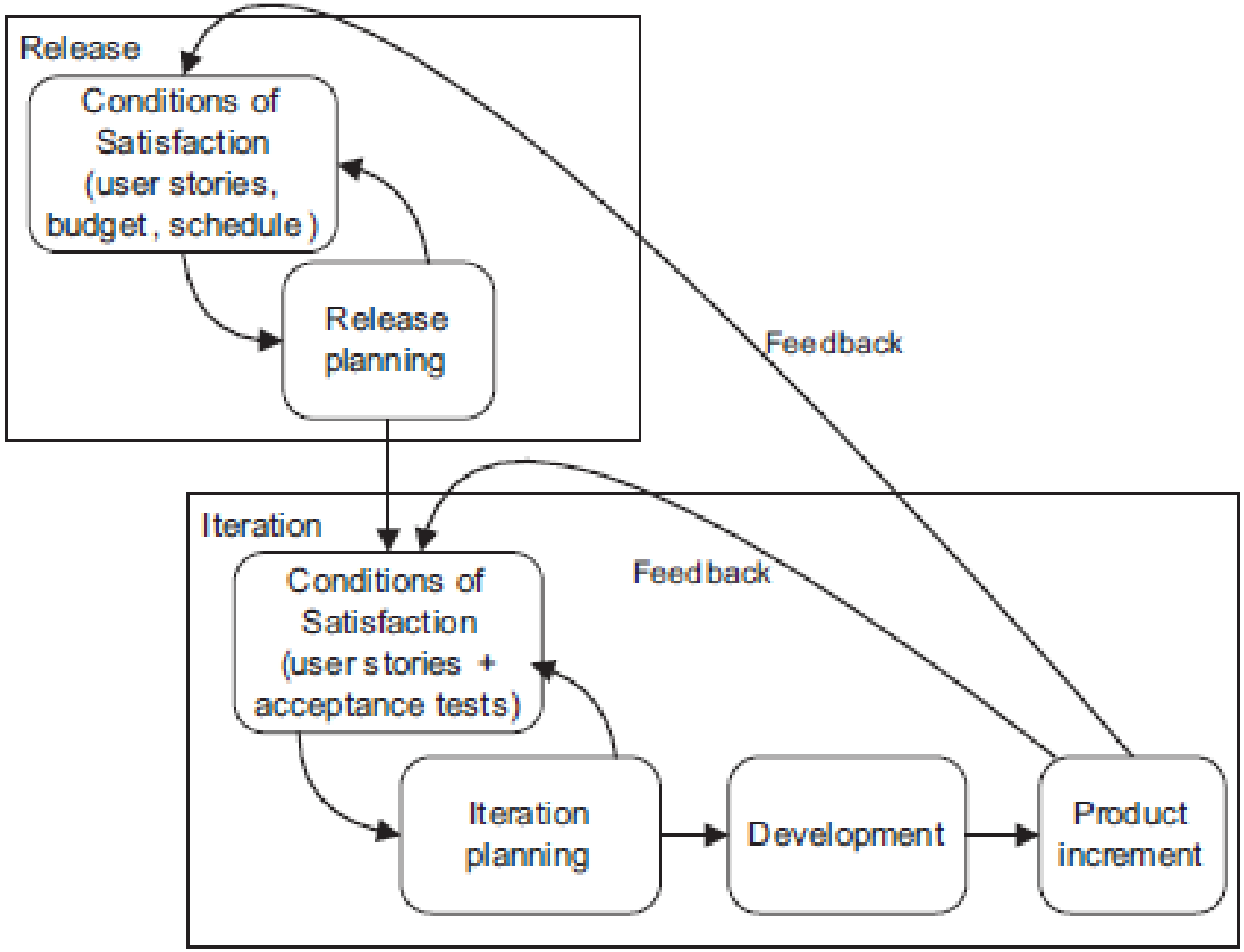
- Al comienzo de cada iteración.
- En base al trabajo terminado en la iteración anterior, el *product owner* determina la prioridad del trabajo a realizar en la nueva iteración.
- Se planifican la tareas necesarias para transformar un feature requerido en software operative y verificado.

PLAN DIARIO

- Reunión diaria de pie para coordinar el trabajo y sincronizar los esfuerzos diarios.
- Alcance: hasta el día siguiente.
- Foco en la planificación de tareas y en la coordinación de actividades individuales que llevarán a completar la tarea.

CONDICIONES DE SATISFACCIÓN

- Son los criterios que van a ser utilizados para evaluar el éxito del proyecto (criterios de éxito).
- Comprender las condiciones de satisfacción del *product owner* es esencial tanto en la planificación de la release como en la planificación de la iteración.
- Planificación de la release
  - todo el equipo identifica una forma de cumplir con las condiciones de satisfacción de la release, lo que incluye el alcance, el cronograma y los recursos (calidad no negociable).
  - Para lograrlo, puede que el *product owner* deba relajar una o varias condiciones de satisfacción.
- Planificación de la iteración
  - las condiciones de satisfacción son los nuevos *features* que se implementarán y los casos de prueba de alto nivel que demuestran que los *features* fueron implementados correctamente.


Release
Conditions of Satisfaction (user stories, budget , schedule)
Release planning
Feedback
Iteration
Conditions of Satisfaction (user stories + acceptance tests)
Feedback
Iteration planning
Development
Product increment

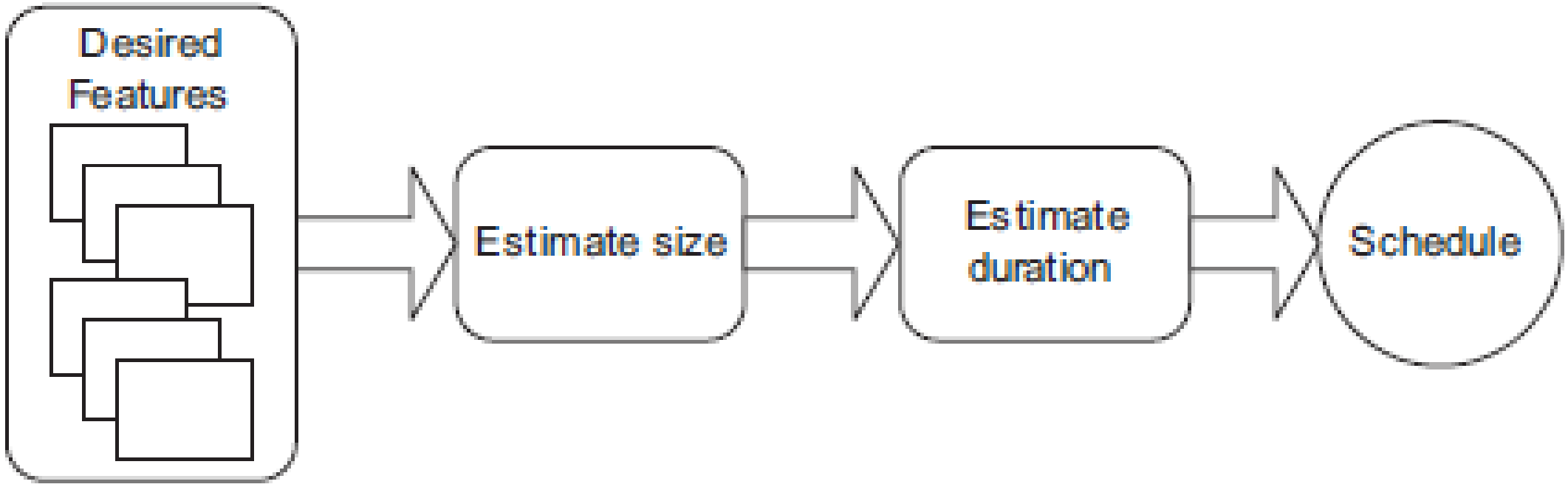
PARTE II
ESTIMACIÓN DEL TAMAÑO

UNIDADES

TÉCNICAS

REESTIMACIÓN

PROCESO DE ESTIMACIÓN DE LA DURACIÓN



CAP. 4
ESTIMACIÓN DEL TAMAÑO CON PUNTOS DE HISTORIA

PUNTOS DE HISTORIA: MEDIDA DE TAMAÑO RELATIVO

- Los puntos de historia son una **medida relativa** del **tamaño** de la historia de usuario:
  - Una historia de usuario estimada en 10 puntos de historia es dos veces más grande, compleja o riesgosa que una historia estimada en 5 puntos de historia, y la mitad que una estimada en 20.
- Una estimación en puntos de historia amalgama:
  - Esfuerzo en desarrollar el *feature*
  - La complejidad de desarrollarlo
  - El riesgo inherente, etc.
- Dos formas de empezar:
  - Por la más pequeña y asignarle 1.
  - Por una mediana y asignarle la mitad del rango.
- Sugerencia: rango 1-10.

EJERCICIO: PUNTOS DE PERRO

VELOCIDAD

- Es una medida de la tasa de avance del equipo por iteración.
- Al final de cada iteración, el equipo calcula la velocidad = Σ puntos de historia de cada historia completada en la iteración.
- La duración del proyecto se calcula (no se estima).
- Cantidad total de iteraciones = el total de puntos de historia / la velocidad del equipo.
- Se puede usar la velocidad del equipo en proyectos anteriores o se puede estimar.

CORRECCIÓN DE LA ESTIMACIÓN

- Se puede calcular la velocidad después de las primeras iteraciones.
- Calcular la velocidad permite corregir las estimaciones, sin reestimar el trabajo.
- Separa (se estiman de forma independiente):
  - La estimación del esfuerzo (tamaño)
  - La estimación de la duración

CAP. 5
ESTIMACIÓN EN DÍAS IDEALES

TIEMPO IDEAL VS. TIEMPO TRANSCURRIDO

- Tiempo ideal
  - Cantidad de tiempo que lleva una actividad sin actividades periféricas.
- Tiempo transcurrido
  - Cantidad de tiempo que transcurre en el reloj o en el calendario.

TIEMPO IDEAL VS. TIEMPO TRANSCURRIDO

- Ejemplos
  - Un partido de fútbol
    - Tiempo ideal: 90 min.
    - Tiempo transcurrido: + tiempo extra + penales + entretiempo.
  - Un partido de básquetbol
    - Tiempo ideal: 40 min.
    - Tiempo transcurrido: + en los libres se para el reloj + 3 descansos + tiempos.
  - Un partido de tenis
    - Tiempo ideal: largo de cada punto / game / set
    - Tiempo transcurrido: espacio entre puntos + cambio de lado + suspensión por lluvia

TIEMPO IDEAL VS. TIEMPO TRANSCURRIDO

- Cada tipo de duración cumple un propósito. P. ej.:
  - el árbitro de un partido de fútbol usa tiempo ideal para pitar
  - el tiempo transcurrido se usa para calcular espacio de TV a reservar
- Es más fácil estimar tiempo ideal que tiempo transcurrido.

EL TIEMPO IDEAL Y EL DESARROLLO DE SOFTWARE

- El tiempo ideal =! tiempo transcurrido.
- Debido a sobrecarga natural en el trabajo cotidiano.

FACTORES QUE AFECTAN EL TIEMPO IDEAL

Apoyo a la release actual	Capacitación
Tiempo en que estuve enfermo	Correo electrónico
Reuniones	Revisiones y recorridas
Demostraciones	Entrevistas a candidatos
Problemas personales	Cambio de tareas
Llamadas por teléfono	Arreglo de errores en la release actual
Proyectos especiales	Revisiones de gestión

- Interrupciones:
  - Gerentes: cada 5 minutos
  - Desarrolladores: cada 15 minutos
- La multitarea agranda la brecha. Se pierde eficiencia al cambiar de una tarea a otra.

ESTIMAR EN DÍAS IDEALES O TRANSCURRIDOS

- Es más fácil estimar una historia de usuario en días ideales que en días transcurridos.
- Estimar en días transcurridos:
  - requiere considerar todas las interrupciones posibles.
- Al estimar en días ideales:
  - Consideramos solo cuánto tiempo llevará la historia.
  - Asumimos que
    - la historia estimada es en lo único en lo que trabajaremos
    - tendremos a mano todo lo necesario al comenzar
    - no habrá interrupciones
- Días ideales: estimación de tamaño, al igual que puntos de historia.

DÍAS IDEALES COMO MEDIDA DE TAMAÑO

- Días ideales = tiempo que llevara desarrollar, probar y aceptar una historia de usuario.
- No es necesario considerar *overhead* del contexto.
- El tiempo transcurrido diferirá, claro.
- Días ideales = estimación de tamaño (ídem puntos de historia).
- → puedo convertir una estimación en días ideales en una estimación de duración usando la velocidad (igual que con puntos de historia).

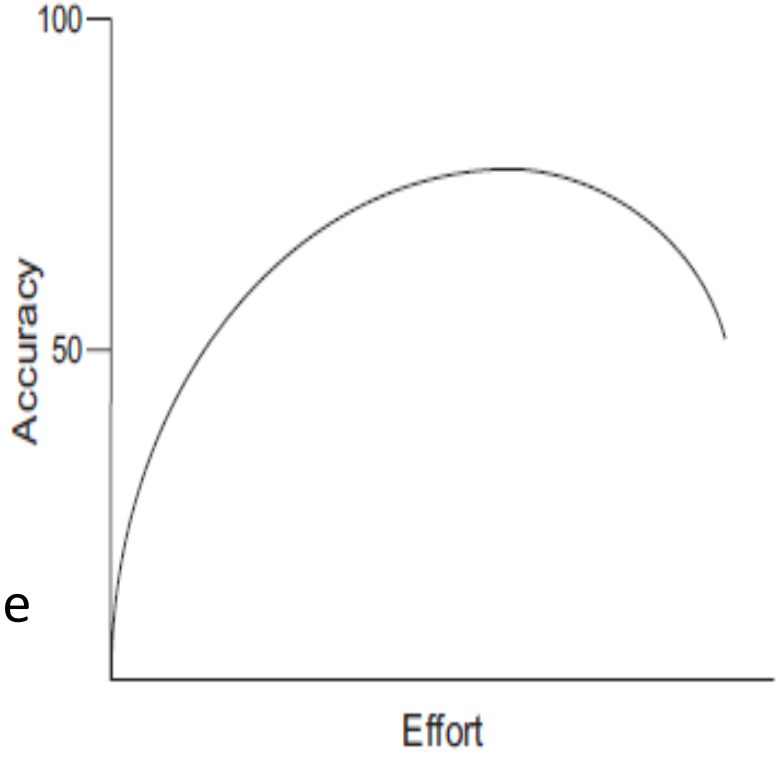
UNA ESTIMACIÓN, NO MUCHAS

- Asociar una única estimación a cada historia de usuario.
- No estimar lo que le llevará a cada rol por separado:
  - El foco en los roles individuales:
    - Cambia la mentalidad de «estamos en esto juntos».
    - Deberíamos seguir la velocidad y el trabajo remanente para cada rol.
  - Este esfuerzo adicional rara vez vale la pena, pero puede ser en alguna ocasión necesario.

CAP. 6.
TÉCNICAS PARA ESTIMAR

ESFUERZO DE ESTIMACIÓN Y PRECISIÓN

- Gastar más tiempo y esfuerzo en estimar no necesariamente aumenta la precisión de la estimación.
- P. ej. Lavado del auto.
- Nunca llego al 100 %.
- Con poco esfuerzo, mejoro mucho la estimación.
- La cantidad de esfuerzo que se pone en una estimación debería determinarse por el propósito de la estimación.



QUIÉN DEBE ESTIMAR

- La estimación debe hacerla el equipo y no una sola persona.
- A pesar de que es sabido que las mejores estimaciones las hacen los que harán el trabajo,
  - en un equipo ágil no se sabe por adelantado quién hará el trabajo.
  - Alguien más puede aportar algo importante.

LA ESCALA DE ESTIMACIÓN

ESCALAS

- Las estimaciones deben estar en una escala predefinida.
- Mantenerlas dentro de un orden de magnitud.
- Dos posibles escalas:
  - Fibonacci: 1, 2, 3, 5 y 8
    - La brecha en la secuencia aumenta a medida que los nros. aumentan
  - 1, 2, 4 y 8
    - Cada número es el doble del precedente.
- Estas escalas no lineales sirven porque reflejan la mayor incertidumbre asociada a estimaciones de unidades de trabajo más grandes.

INCLUSIÓN DEL 0 EN LA ESCALA

- Se puede incluir el 0 en la escala:
  - Poco probable que haya historias que no lleven nada de trabajo, pero asignar el 0 es útil:
    - Asignarle un nro. mayor que 0 a rasgos muy pequeños o triviales limitará el tamaño de los más grandes.
    - Si el trabajo está más cerca de 0 que de 1, no queremos que contribuya a la velocidad, porque costará más ganar ese punto en una iteración posterior. De lo contrario, bajará la velocidad.
  - 13 x 0 != 0
  - Alternativa: agrupar varias historias pequeñas y estimarlas como una sola unidad.

- Si se usan números grandes (1-100), identificar previamente las cifras a usar.
- No estimar una historia con 66 SP y otra con 67 SP.
- Falso nivel de precisión. No podemos distinguir una diferencia de 1,5 % en tamaño.
- Desde un punto de vista porcentual, las diferencias entre 1, 2 y 3 son mayores que entre 66 y 67.

GRANULARIDAD

- No siempre es posible estimar en un solo orden de magnitud porque implicaría escribir todas las historias con un alto nivel de granularidad.
- Historias de usuario
- Épicas:
  - Es una historia mucho más grande.
  - Sirve para los *features* que no estamos seguros de querer.
  - Sirve para *features* que no se desarrollarán en un futuro cercano.
- Temas
  - Es la combinación de un conjunto de historias de usuario relacionadas.
  - Se tratan como una unidad para la estimación y para la planificación de la *release*.
- Las estimaciones de épicas y temas serán más inciertas que las estimaciones de historias de usuario más específicas y pequeñas.

RECOMENDACIONES

- Las historias de usuario que se trabajarán en las próximas iteraciones:
  - deben ser lo suficientemente pequeñas como para ser completadas en una única iteración.
  - Estimarlas en un orden de magnitud, con una escala no linear de 1-10, tal como 1, 2, 3, 5 y 8 o 1, 2, 4, y 8.
- Las historias más grandes (épicas o temas) que no van a ser implementadas en las próximas iteraciones:
  - pueden ser estimadas en unidades tales como 13, 20, 40 y 100.

TÉCNICAS DE ESTIMACIÓN

- Juicio de expertos
- Analogía
- Desagregar

- *Planning poker*

JUICIO DE EXPERTOS

- Más fácil estimar tareas (en procesos orientados por planes) que unidades asociadas al uso del usuario (CU, historias de usuario) (en procesos ágiles), porque esta requieren una variedad de habilidades.

ANALOGÍA

- Comparar la estimación de una historia con otras. «Esta historia es el doble que aquella».
- Más fácil estimar tamaños relativos que absolutos.
- Triangulación:
  - No comparar una historia con una única línea base o referencia universal.
  - Comparar la historia con un grupo de las ya estimadas.

DESAGREGACIÓN

- Partir la historia en pedazos más pequeños y más fáciles de estimar.
- No exagerar. Problema:
  - Que me olvide de un pedazo (no tengo forma de verificar que están todos).
  - La sumatoria de estimaciones de tareas pequeñas puede dar un rango más grande que lo razonable.

PLANNING POKER

- En qué consiste
- La cantidad apropiada de discusión
- Estimar en subgrupos
- Cuándo jugar *planning poker*
- Por qué funciona

EN QUÉ CONSISTE

- Combina las tres técnicas.
- Participantes:
  - todos los desarrolladores del equipo = programadores, verificadores, ingenieros de BD, analistas, diseñadores de interacción con el usuario, etc.)
  - No más de 10. Si no, dividir en dos equipos.
  - El *product owner* participa, pero no estima.
- Procedimiento:
  - A cada estimador se le da un juego de tarjetas con la escala de estimación.
  - El moderador lee la descripción de un *feature* (puede ser cualquiera; generalmente es el *product owner* o un analista).
  - El *product owner* responder preguntas.
  - Cada estimador selecciona la tarjeta con su estimación. Todas las cartas se muestran al mismo tiempo.
  - Los estimadores con los nros. más bajo y más alto explican sus estimaciones.
  - Se discuten las estimaciones y el moderador toma notas para la implementación y la prueba. Se repite el proceso hasta que se alcanza un acuerdo.

LA CANTIDAD APROPIADA DE DISCUSIÓN

- Algo de discusión preliminar sobre el diseño es necesaria al estimar, pero no excesiva.
- Objetivo: llegar a una estimación valiosa con poco esfuerzo.
- Se puede usar un reloj de arena de dos minutos.

ESTIMACIÓN EN SUBGRUPOS MÁS PEQUEÑOS

- Se puede hacer con un subgrupo, con al menos tres estimadores.
- No es lo ideal, pero puede ser una opción razonable si hay mucho ítems a estimar (al comienzo del proyecto).
- Para asegurar la consistencia en la estimación de los equipos:
  - comenzar juntos durante 1 h estimando 10-20 historias;
  - cada equipo se lleva una copia de esas historias y sus estimaciones para usar de línea base para las demás estimaciones que le toque.

CUÁNDO ESTIMAR CON *PLANNING POKER*

- En 2 momentos diferentes:
  1. Antes que comience el proyecto o durante las primeras iteraciones, 2 o 3 reuniones de 1 a 3 h.
  2. Una reunión de estimación muy corta cerca el fin de cada iteración, para estimar las nuevas historias identificadas durante la iteración.

POR QUÉ FUNCIONA

1. Junta opiniones de múltiples expertos de varias disciplinas. «Las personas más competentes para resolver una tarea deberían estimarla».
2. El diálogo mejora la certeza de las estimaciones.
3. Promediar las estimaciones individuales, como se hace en discusiones en grupo, mejora los resultados.
4. Es divertido.

CAP. 7.
REESTIMACIÓN

CUÁNDO REESTIMAR

- Recordar que puntos de historia y días ideales son estimaciones de tamaño y complejidad de cada *feature*, no del tiempo que lleva implementarlo.
- Reestimar solo cuando se crea que el **tamaño relativo** de una o más historias ha **cambiado**.
- No importa si la estimaciones son más o menos correctas; solo si son consistentes.
- Nunca mezclar estimaciones revisadas y las originales.
- Si algunas historias están mal estimadas en su tamaño relativo, restimar la menor cantidad posible para realinear las estimaciones.

CUÁNDO NO REESTIMAR

Historias	Estimación inicial	Reestimación
Historia 1	3 PH	6 PH
Historia 2	3 PH	6 PH
Historia 3	3 PH	
Historia 4	3 PH	

- Todas de tamaño y complejidad equivalentes.
- Estimación inicial de velocidad: 12 PH por iteración.
- Final de la 1.a iteración: 1 y 2 completos (velocidad: 6 PH).
- Reestima 1 y 2 en 6 PH cada una. Velocidad: 12 PH.
- Debe reestimar 3 y 4.
- Se duplicó la velocidad, pero también se duplicó el trabajo restante → misma situación.
- No reestimar si no cambia el tamaño relativo.
- No reestimar cuando el avance no vaya tan rápido como esperábamos. Dejar que la velocidad se encargue de ajustar las inexactitudes de la estimación.

Historias	Estimación inicial (PH)	Reestimación		
		Escenario 1: no reestimar	Escenario 2: reestimar las historias terminadas	Escenario 3: reestimar cuando cambia el tamaño relativo
1	3		6	6
2	5			10
3	3			6
4	3			
5	3			
6	5		5	

- Historias 1, 2 y 3 despliegan gráficas.
- Plan de it. 1:
  - historias 1, 2 y 6.
  - Velocidad planeada: 13 PH.
- Real it. 1:
  - historias 1 y 6. Velocidad: 8 PH.
  - Fue más difícil hacer gráficas.
  - H1 debería haber sido 6 PH.
  - H2 y H3 también deberían ser el doble de lo estimado.

- Escenario 1: dejo todo así. velocidad = 8 PH, pero sé que no puedo hacer 2 y 3 en la misma iteración.
- Escenario 2: reestimo H1 -> velocidad anterior = 11 HP. Plan: H2, H3 y H4 (no voy a poder).
- Escenario 3: reestimar H1, H2 y H3. velocidad = 11 PH. Plan: H2.

CÓMO CONTAR [Y REESTIMAR] HISTORIAS PARCIALMENTE COMPLETAS

- Al final de la iteración:
  - si la historia está terminada (codificada, verificada y aceptada) se ganan los puntos; si falta algo, no ganan nada (0/100).
- Si al final de la iteración hay una iteración parcialmente completa, hay dos opciones:
  - Si se va a terminar en la iteración siguiente:
    - no acreditar ninguno de sus puntos y acreditarlos cuando se complete.
    - Habrá un desajuste en la velocidad en las dos iteraciones, pero despreciarlo.
  - Si no se va a terminar en la iteración siguiente:
    - Estimar y acreditar la parte completada.
    - Reestimar la parte remanente.
    - La suma no tiene por qué ser igual a la estimación inicial.
  - Mejor solución:
    - No tener historias incompletas.
    - Que las historias sean suficientemente pequeñas para que el crédito parcial no sea un problema.

CAP. 8
ELEGIR ENTRE PUNTOS DE HISTORIA Y DÍAS IDEALES

- Un equipo puede elegir estimar en puntos de historia o en días ideales. Cada uno tiene sus ventajas.

VENTAJAS DE ESTIMAR EN PUNTOS DE HISTORIA

- Ayudan a promover el comportamiento transversal en el equipo.
- Son una medida más pura del tamaño.
- Las estimaciones en puntos de historia no se deterioran. No necesitan reestimarse si el equipo mejora la tecnología o el dominio.
- Generalmente es más rápido estimar en puntos de historia que en días ideales.
- Pueden ser comparados entre miembros del equipo. Mis días ideales no son tus días ideales. Si uno piensa que algo llevará 4 días ideales y otro que llevara 1, ambos pueden tener razón y no hay base sobre la que discutir y establecer una única estimación.

VENTAJAS DE ESTIMAR EN DÍAS IDEALES

- Son más fáciles de explicar a gente ajena al equipo.
- Son más fáciles de estimar al comienzo.
- Facilitan las predicciones iniciales de velocidad.

RECOMENDACIÓN

- Usar puntos de historia.
- Preguntar: «¿Cuán grande es esta historia comparada con las otras que estimamos?» en lugar de «¿cuántos días ideales llevará?».

PARTE III
PLANIFICAR PARA AGREGAR VALOR

CAP. 9
PRIORIZAR TEMAS

FACTORES EN LA PRIORIZACIÓN

- Es necesario priorizar qué hacer primero. Considerar 4 factores:

1. El valor financiero de tener esos *features*
2. El costo de desarrollarlo [y mantenerlos]
3. La cantidad y el significado del aprendizaje y del nuevo conocimiento creado al desarrollar los *features*
4. Los riesgos eliminados al desarrollar los *features*

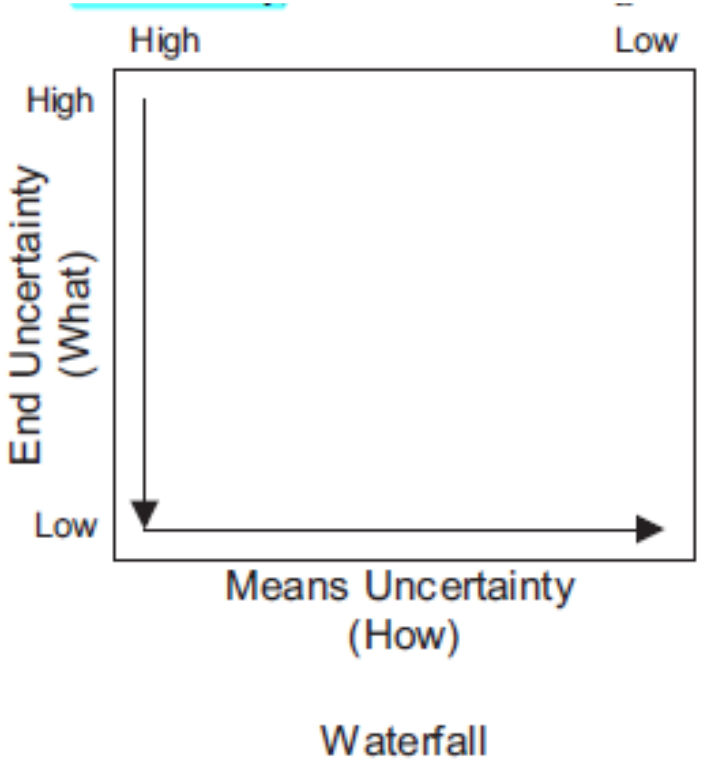
VALOR

- Priorizar según el valor para el negocio = valor financiero del tema: cuánto dinero la organización ganará o ahorrará al tener los nuevos features del tema.
- Estimar el impacto financiero en un período de tiempo (meses, trimestres o años).
- Como estimar el retorno financiero del tema es complejo, se puede usar medidas no financieras de deseabilidad.

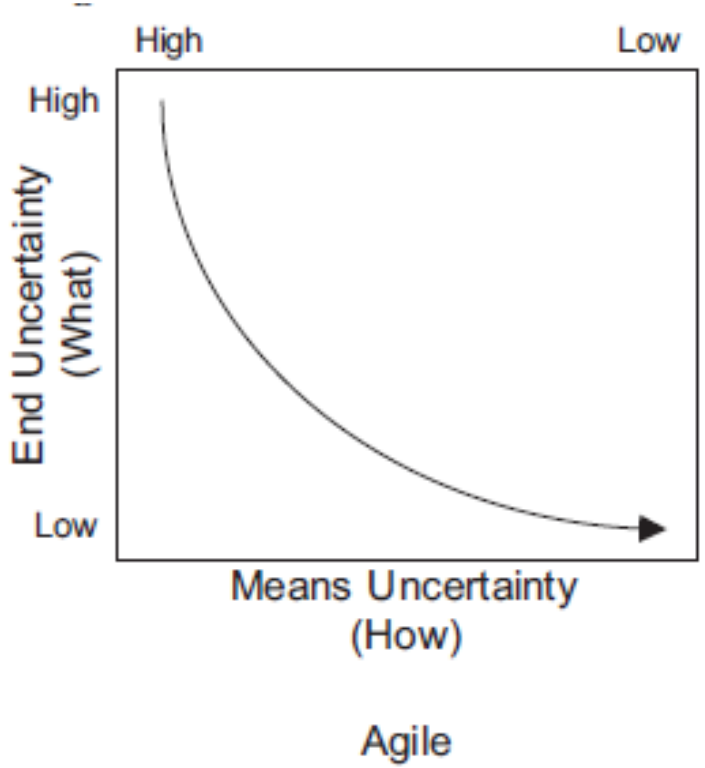
COSTO

- El costo puede cambiar con el tiempo.
- La mejor forma de reducir el costo del cambio es implementar la *feature* lo más tarde posible, cuando no hay más margen para cambiar.
- Hacer una conversión de puntos de historia a dinero:
  - sumar salarios en un período
  - ver puntos de historia implementados.

CONOCIMIENTO



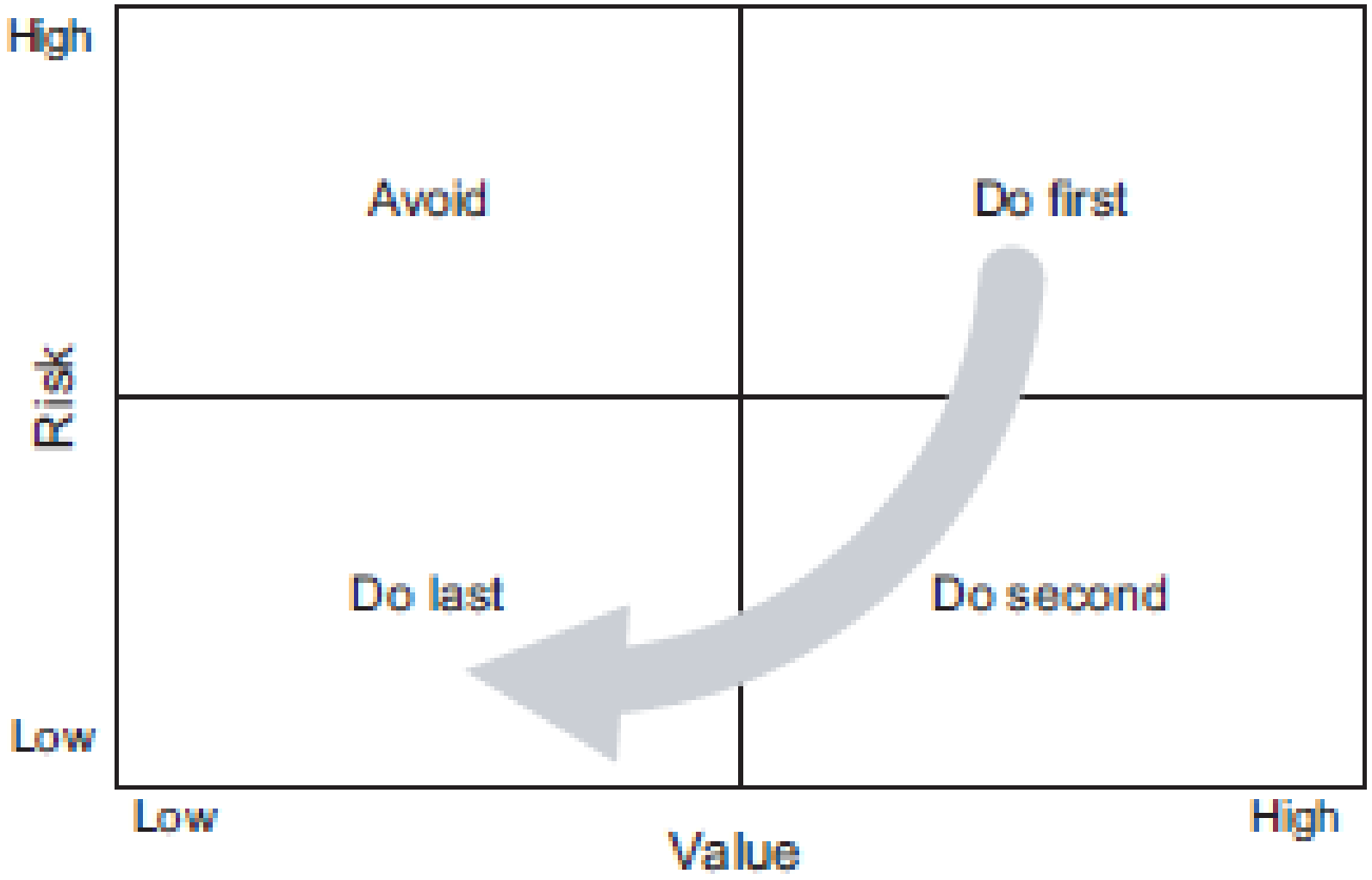
Waterfall



Agile

- *end uncertainty*: incertidumbre sobre que *features* deberá tener el producto.
- *means uncertainty*: incertidumbre de cómo construirlo.
- Se reducen adquiriendo más conocimiento sobre el producto y sobre el proyecto.

RIESGO



COMBINACIÓN DE LOS CUATRO FACTORES

- Orden inicial: los *features* con tasa valor/costo del tema más alta
- Mover los temas hacía arriba o abajo en este orden basándose en riesgo y conocimiento.

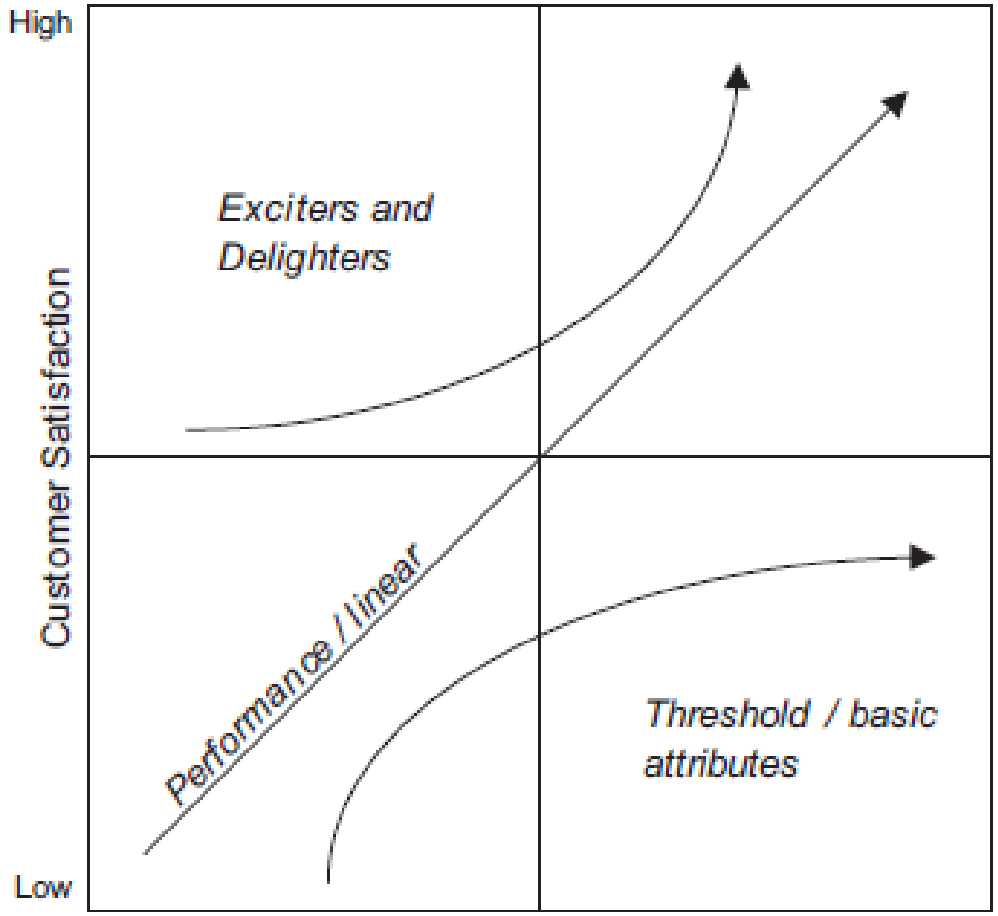
CAP. 11
PRIORIZAR LA DESEABILIDAD

DOS ENFOQUES:
ANÁLISIS KANO Y PESOS RELATIVOS

EL MODELO KANO DE SATISFACCIÓN DEL CLIENTE

- Los *features* se separan en *threshold* o *must-have features*, *linear features*, y *exciters and delighters.*
- *Threshold features*: los que tienen que estar para que el producto será exitoso.
- Features lineales: «cuanto más, mejor».
  - La satisfacción del cliente se correlaciona linealmente con la cantidad de ese *feature*.
  - El precio del producto se relaciona con estos atributos lineales.
- *Exciters* y *delighters*: *features* que proporcionan gran satisfacción
  - a menudo agregan un precio premium al producto.
  - Su falta no hará decrecer la satisfacción por debajo de neutral.
  - «necesidades desconocidas»: los clientes a menudos no saben que los necesitan hasta que los ven.

MODELO KANO DE SATISFACCIÓN DEL CLIENTE



- Una vez que se implementó cierta cantidad de un features must-have, la satisfacción del cliente no se puede aumentar con más cantidad de ese features.
- Relación directa en enre features lineales y la satisfacción del cliente.
- La satisfacción del cliente se eleva incluso con una implementación parcial de un exciter o delighter.

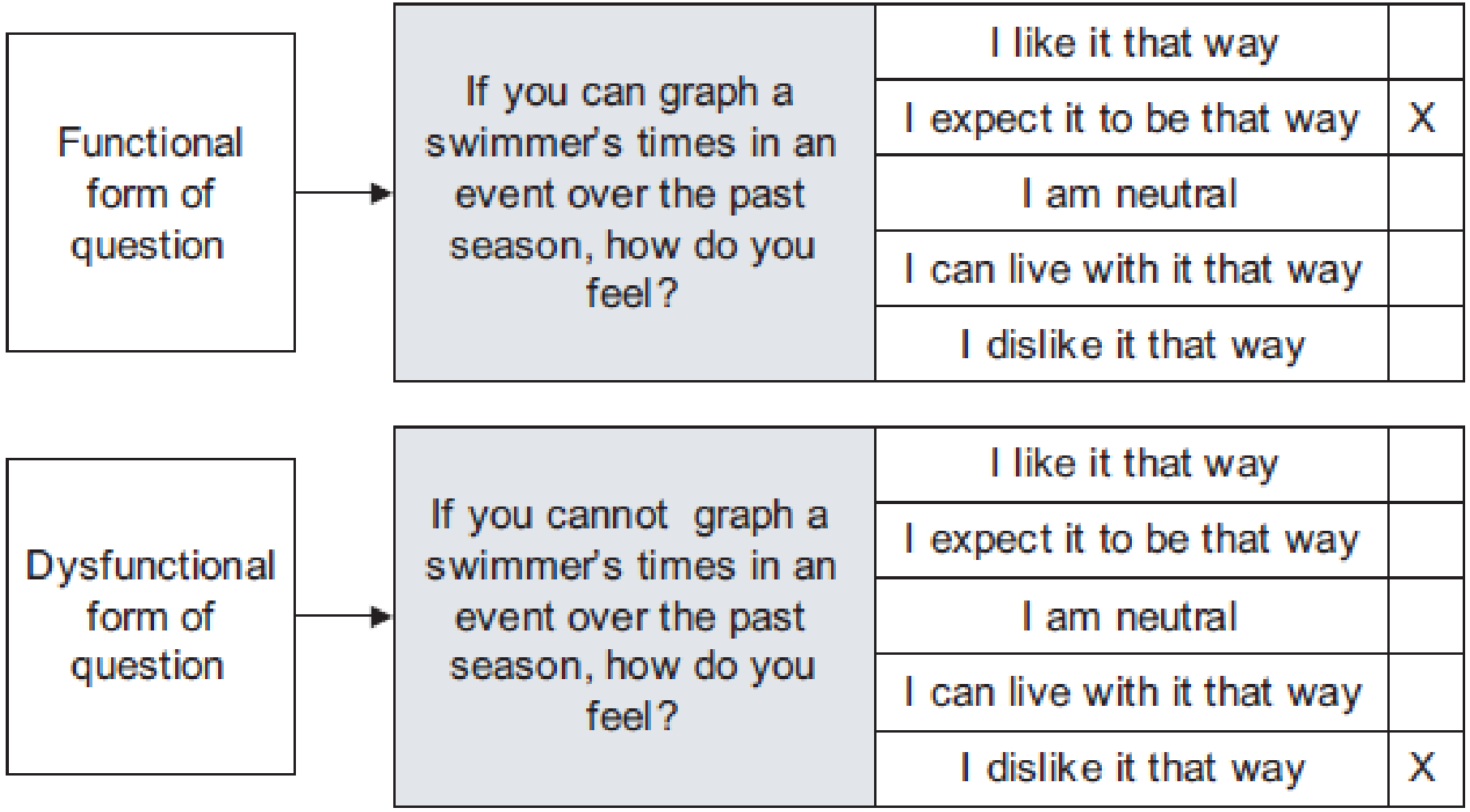
PRIORIZACIÓN EN EL DESARROLLO

- *Threshold features*, antes de que el producto sea liberado.
- Tantos *features* lineales como sea posible.
- Si da el tiempo, unos pocos *delighters*.

- Atención que los *features* tienden a descender en el diagrama Kano: lo que era un *delighter* ahora es un *feature* lineal y en breve se convertirá en un *must have*. P. ej. internet en el cuarto.

EVALUACIÓN DE LOS TEMAS

- Se separan los *features* preguntándole a los potenciales usuarios (20-30) dos preguntas:
  - Cómo se sentirían si el *feature* estuviera presente.
  - Cómo se sentirían si el *feature* no estuviera presente.


Functional form of question
If you can graph a swimmer's times in an event over the past season, how do you feel?
I like it that way
I expect it to be that way
X
I am neutral
I can live with it that way
I dislike it that way
Dysfunctional form of question
If you cannot graph a swimmer's times in an event over the past season, how do you feel?
I like it that way
I expect it to be that way
I am neutral
I can live with it that way
I dislike it that way
X

Customer Requirements → / ↓		Dysfunctional Question				
		Like	Expect	Neutral	Live with	Dislike
Functional Question	Like	Q	E	E	E	L
	Expect	R	I	I	I	M
	Neutral	R	I	I	I	M
	Live with	R	I	I	I	M
	Dislike	R	R	R	R	Q

M Must-have
L Linear
E Exciter
R Reverse
Q Questionable
I Indifferent

CLASIFICACIÓN FINAL

Theme	E	L	M	I	R	Q	Category
Graph event times	18.4	**43.8**	22.8	12.8	1.7	0.5	Linear
Can upload photos	8.3	30.9	**54.3**	4.2	1.4	0.9	Must-have
Post autobiographical profile	**39.1**	14.8	**36.6**	8.2	0.2	1.1	Exciter Must-have

- Sumar las respuestas de todos los usuarios .
- Las respuestas pueden ser inconsistente.
- Si hay dos valores altos, indica que diferentes usuarios tienen diferentes expectativas.

PESO RELATIVO

- Enfoque para evaluar los beneficios de implementar un *feature*, las sanciones por no implementarlo y el costo, en un único valor que representa la prioridad del *feature*.
- Se basa en juicio de expertos.
- El equipo, guiado por el *product owner*, analiza cada *feature* de la release:
  - el beneficio de implementarlo
  - la penalización de no implementarlo
  - ambas son medidas relativas (escala 1-9)
  - se les puede agregar peso

Feature	Relative Benefit	Relative Penalty	Total Value	Value %	Estimate	Cost %	Priority
Graph event times	8	6	14	42	32	53	0.79
Can upload photos	9	2	11	33	21	34	0.97
Post autobiographical profile	3	5	8	25	8	13	1.92
Total	**20**	**13**	**33**	**100**	**61**	**100**	

Dividir value % / cost %

CAP. 12
DIVIDIR HISTORIAS DE USUARIO

CUÁNDO DIVIDIR UNA HISTORIA DE USUARIO

- Puede ser útil dividir
  - una historia que no entra en ninguna iteración porque es demasiado grande
  - una historia que es demasiado grande para entrar en el tiempo restante en la iteración que se está planificando.
  - una historia grande si se necesita proporcionar una estimación más exacta de la que puede hacerse de la historia grande.

CÓMO DIVIDIR UNA HISTORIA DE USUARIO

- Se puede dividir una historia según el tipo de datos que manejará.
- También se puede dividir una historia según las fronteras operacionales, es decir, en base a sus operaciones inherentes. Es frecuente dividir historias según las operaciones comunes CRUD (Create, Read, Update, Delete).
- Dividir historias de prioridad mezclada. Muchas historias describen dos o más necesidades. Si estas necesidades tienen diferente prioridad, divida las historias de esa manera.
- Evite dividir una historia en las tareas de desarrollo necesarias para implementar el *feature*.

CÓMO ACHICAR UNA HISTORIA

- Se puede achicar una historia removiendo aspectos transversales tales como seguridad, logging, manejo de errores, etc.
- También se puede achicar una historia ignorando los objetivos de desempeño durante la iteración en la que se implementa la funcionalidad. Se puede crear una historia para el objetivo de desempeño y satisfacerla en un iteración posterior.

EVITAR LA TENTACIÓN DE CAMBIOS RELACIONADOS

- Evite la tentación de agrandar una historia ya de por sí grande al incluir cambios relacionados que no son necesarios para la entrega de la historia de usuario.

COMBINAR HISTORIAS

- A veces es apropiado combinar historias de usuario —en especial en el caso de corrección de errores— que podrían ser demasiado pequeñas por sí mismas.

PARTE IV

ARMAR EL CRONOGRAMA

CAP. 13
ASPECTOS BÁSICOS
DE PLANIFICAR UNA RELEASE

EL PLAN DE LA *RELEASE*

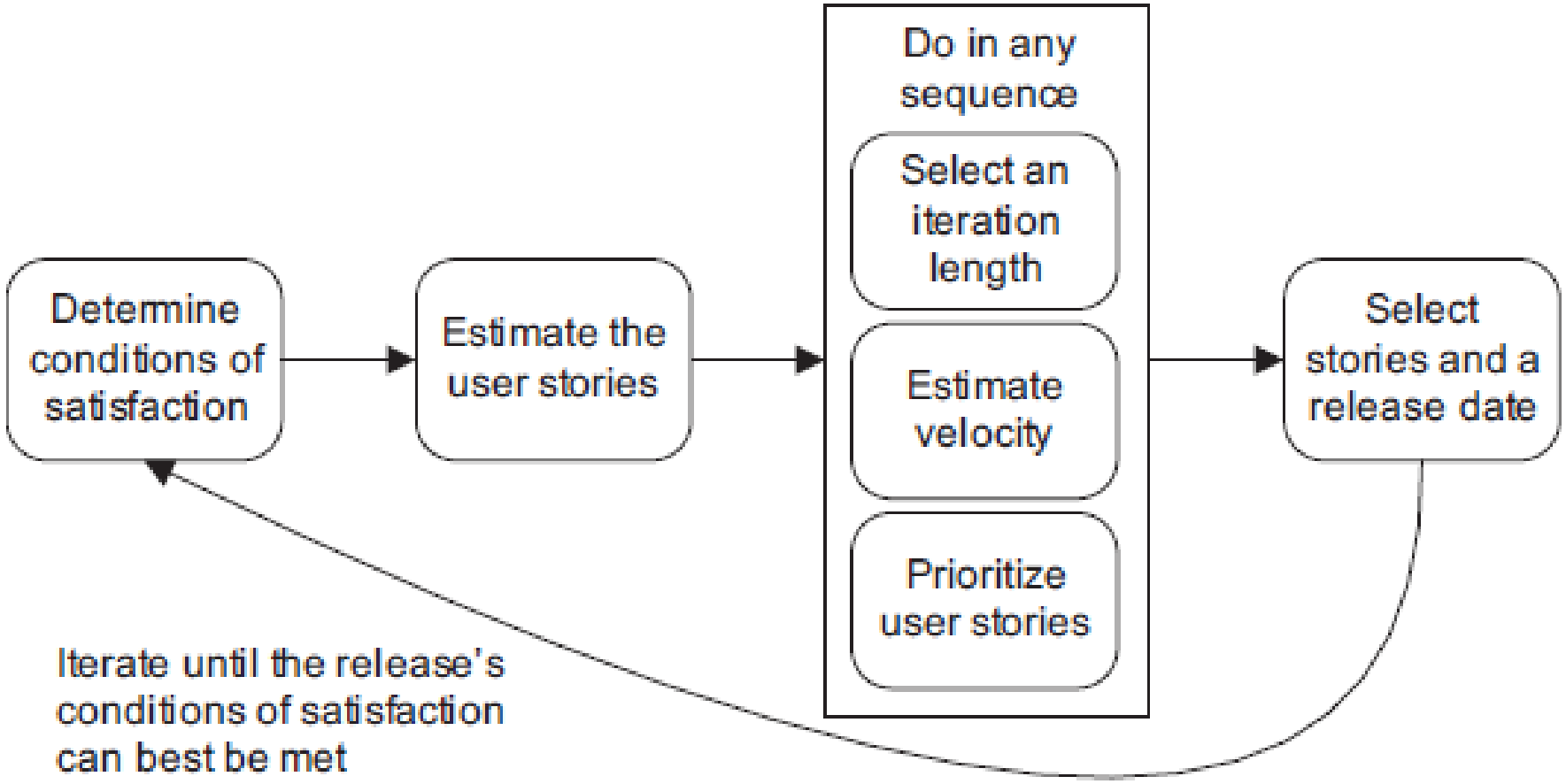
PLANIFICAR UNA *RELEASE*

- Planificar una *release* es el proceso de crear un plan de alto nivel que cubre un período más largo que una iteración.
- En general, una release abarca de 3 a 6 meses, con 3-12 iteraciones, pero no es inusual tener *releases* más largas o más cortas, dependiendo del tipo de software.
- Una release es importante porque:
  - el equipo y el *product owner* deciden cuánto deben desarrollar y cuánto demorarán en tener un producto liberable. Cuanto antes, la organización puede empezar a ganar ROI.
  - qué se va a desarrollar y el plazo es información para otras actividades de planificación estratégica.
  - Sirve de poste indicador hacia el cual avanzar. Sin el concepto de release, se sucederían las iteraciones indefinidamente. Un plan de release proporciona el contexto que permite que las iteraciones se combinen en un todo satisfactorio.

EL PLAN DE LA *RELEASE*

- La planificación de la *release* es un proceso iterativo que comienza identificando el producto, las condiciones de satisfacción del *product owner* para el proyecto.
- Esta generalmente incluyen metas para el cronograma, el alcance y los recursos. Uno es preminente:
  - *date-driven*
  - *feature-driven*
- Si no se puede planificar un proyecto que cumplan con el conjunto de condiciones de satisfacción iniciales, se repite el proceso de planificación para ver si se puede cumplir con un conjunto menor de condiciones. (Quizás la funcionalidad deseada puede ser entregada más tarde o por un equipo más grande).

PASOS PARA PLANIFICAR UNA *RELEASE*



PROCESO DE PLANIFICACIÓN DE LA RELEASE

- Condiciones de satisfacción:
  - Comienza identificando las condiciones de satisfacción del *product owner* para el proyecto.
  - Generalmente incluyen metas para el cronograma, el alcance y los recursos. Uno es preminente:
    - date-driven
    - feature-driven
- Seleccionar el largo de la iteración
- Estimar la velocidad:
  - usar la velocidad más reciente del equipo
  - si la tecnología o el dominio del negocio cambian dramáticamente, estimar otra

PROCESO DE PLANIFICACIÓN DE LA RELEASE

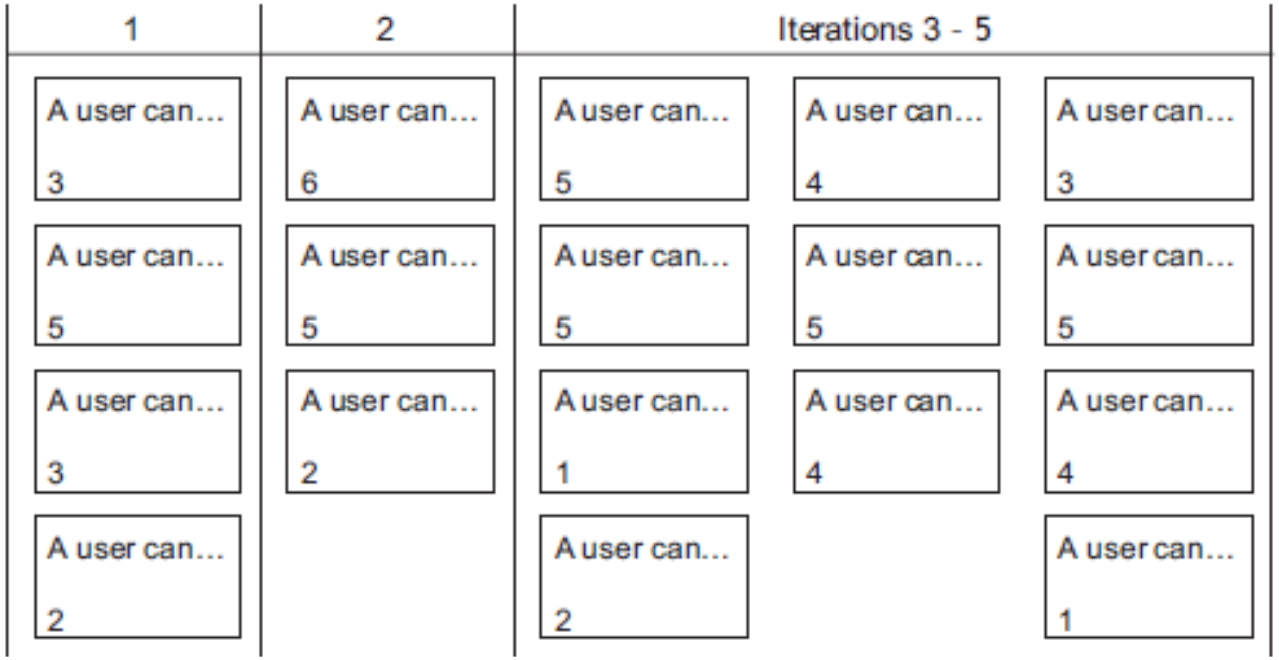
- Priorizar las historias de usuario
- Seleccionar las historias y una fecha
  - Si el proyecto es *feature-driven:*
    - Cantidad de iteraciones = $\sum$ estimaciones de todos los *features* / velocidad esperada.
  - Si el proyecto es *date-driven:*
    - Determinar la cantidad de iteraciones mirando el calendario.
    - Cantidad de iteraciones * velocidad esperada = cantidad de PH o DI que entrarán en la release.

ESTIMAR LAS HISTORIAS DE USUARIO

- Estime cada nuevo *feature* que tiene una posibilidad razonable de ser incluido en la *release*.

NIVEL DE DETALLE DEL PLAN

- Un plan de *release* no necesita describir con exactitud qué se hará durante cada iteración.
- En gral., alcanza con identificar las historias en las que se va a trabajar en las dos primera iteraciones, y dejar la priorización de la demás historias para iteraciones específicas posteriores.



- Si no se puede planificar un proyecto que cumplan con el conjunto de condiciones de satisfacción iniciales, se repite el proceso de planificación para ver si se puede cumplir con un conjunto menor de condiciones. (Quizás la funcionalidad deseada puede ser entregada más tarde o por un equipo más grande).

ACTUALIZACIÓN DEL PLAN DE LA *RELEASE*

- Una vez que se ha creado el plan de la *release*, no hay que dejarlo colgado en la pared.
- Actualizarlo con cierta frecuencia:
  - Fija (si velocidad regular, 4 a 6 semanas).
  - Al comienzo de cada iteración.

CAP. 14
PLANIFICACIÓN DE LA ITERACIÓN

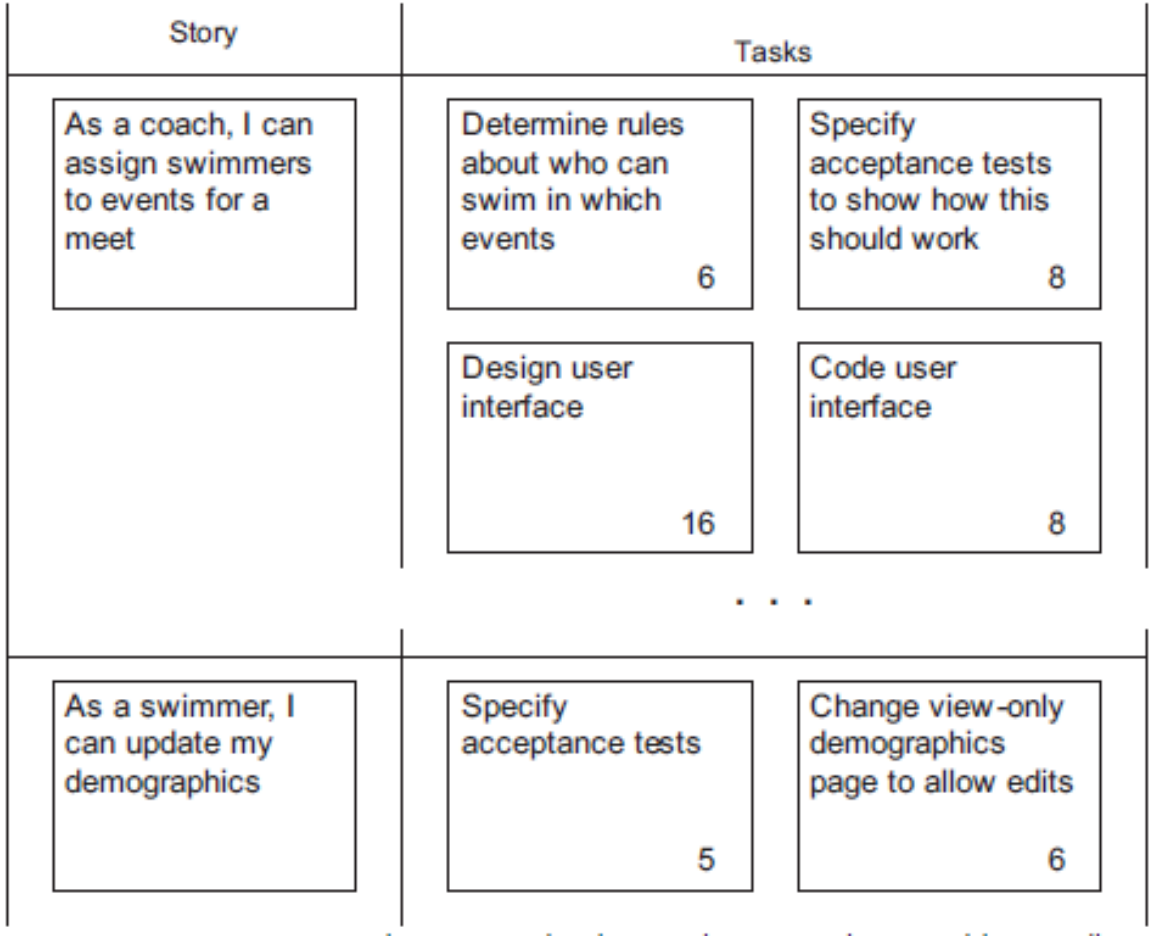
PLANIFICACIÓN DE LA ITERACIÓN

- El plan de la iteración se crea en una reunión de planificación de la iteración. Deberían estar presentes todos los involucrados.
- Para cada historia, determinar las tareas a realizar.

PLANILLA

Row	User Story / Task	Hours
1	As a coach, I can assign swimmers to events for a meet	
2	Determine rules about who can swim in which events	6
3	Specify acceptance tests to show how this should work	8
4	Design user interface	16
5	Code user interface	8
6	Add tables and stored procedures to database	6
7	Automate tests	6
8	As a swimmer, I can update my demographics	
9	Specify acceptance tests	5
10	Change view-only demographics page to allow edits	6
11	...	

TARJETAS



- Más democrático y colaborativo.

LAS TAREAS NO SE ASIGNAN DURANTE LA PLANIFICACIÓN DE LA ITERACIÓN

- No se asignan las tareas a personas específicas.
- Las personas no toman tareas hasta que la iteración comienza y generalmente no más de una o dos relacionadas al mismo tiempo.

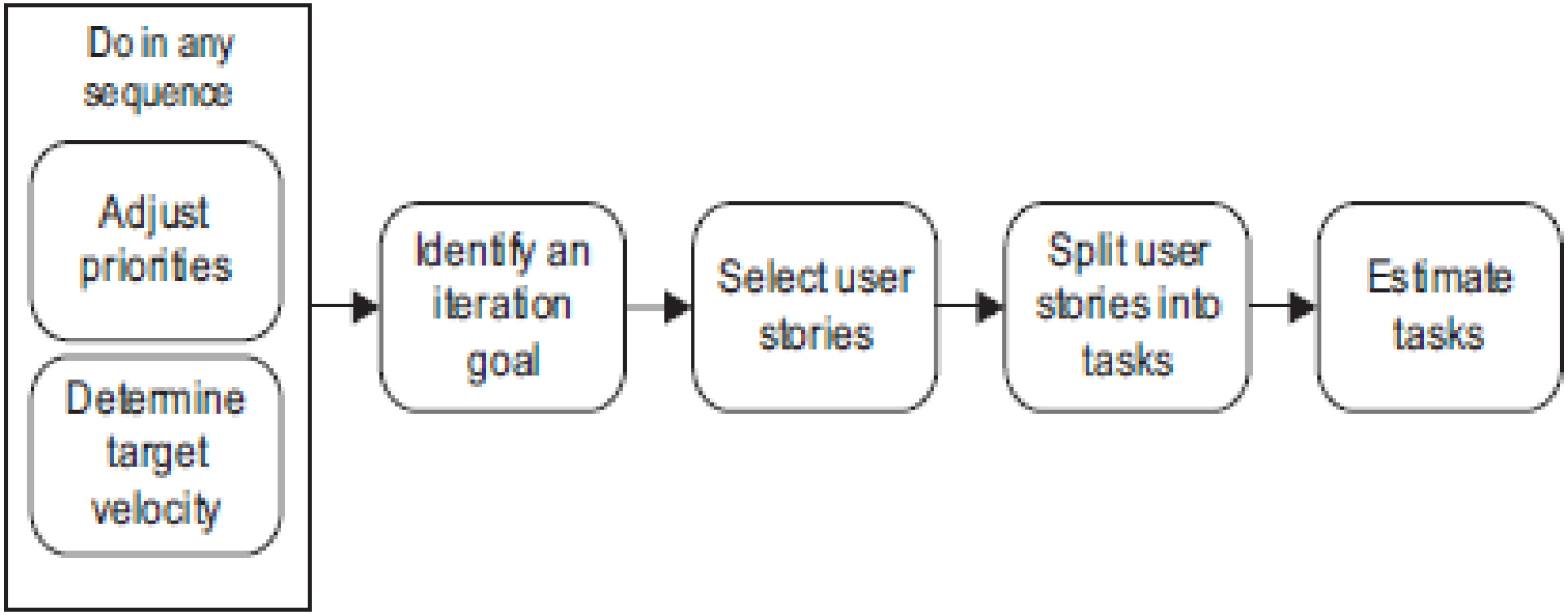
DIFERENCIAS ENTRE LA PLANIFICACIÓN DE LA *RELEASE* Y LA PLANIFICACIÓN DE LA ITERACIÓN

- A diferencia del plan de la *release*, un plan de la iteración mira el trabajo específico de la iteración con más en detalle.
- El plan de la release tiene un horizonte de 3 a 9 meses. El plan de la iteración solo mira una única iteración.
- Las historias del release plan se descomponen en tareas en el plan de la iteración.
- Las historias se estiman en PH o DI. Cada tarea se estima en horas ideales.

ENFOQUES PARA PLANIFICAR UNA ITERACIÓN

- Hay dos enfoques generales para planificar una iteración:
  - *velocity-driven*
  - *commitment-driven*
- Ambos comparten muchos de los mismos pasos y crean a menudo el mismo plan de la iteración.

PLANIFICACIÓN DE LA ITERACIÓN GUIADA POR LA VELOCIDAD



PLANIFICACIÓN DE LA ITERACIÓN GUIADA POR LA VELOCIDAD. PASOS

1. Ajustar las prioridades.
2. Identificar la velocidad deseada para la próxima iteración.
3. Seleccionar una meta de la iteración: descripción general de lo que se quiere lograr.
4. Seleccionar las historias con mayor prioridad que sustentan esa meta. La suma de sus estimaciones debe igualar la velocidad deseada.
5. Dividir cada historia en tareas.
6. Estimar cada tarea.

AJUSTAR LAS PRIORIDADES

- En la reunión de revisión de la iteración, al final de la iteración (idealmente).
- Sugerencia: reunirse aparte algunos días antes del comienzo de la iteración. Si no, no da el tiempo para tener las reuniones de revisión y de planificación de la próxima iteración el mismo día.

DETERMINAR LA VELOCIDAD DESEADA

- La velocidad de la iteración más reciente.
- O tomar el promedio de las últimas 3 iteraciones.
- Si son nuevos, hacer un pronóstico de la velocidad.

REUNIONES

- Incluirlas en la estimación.
- También la preparación
- Una sola tarea por reunión, no una para cada miembro de equipo.

CORRECCIÓN DE ERRORES

- Meta: corregir todos los errores en la iteración en la que fueron descubiertos.
- O bien la estimación para la codificación incluye el tiempo de corregir errores.
- O bien armar una tarea separada y estimarla (preferible). No considerarla complete hasta que pasen todas las pruebas.
- Un defecto encontrado más adelante o no corregido se trata como una historia de usuario.

SPIKES

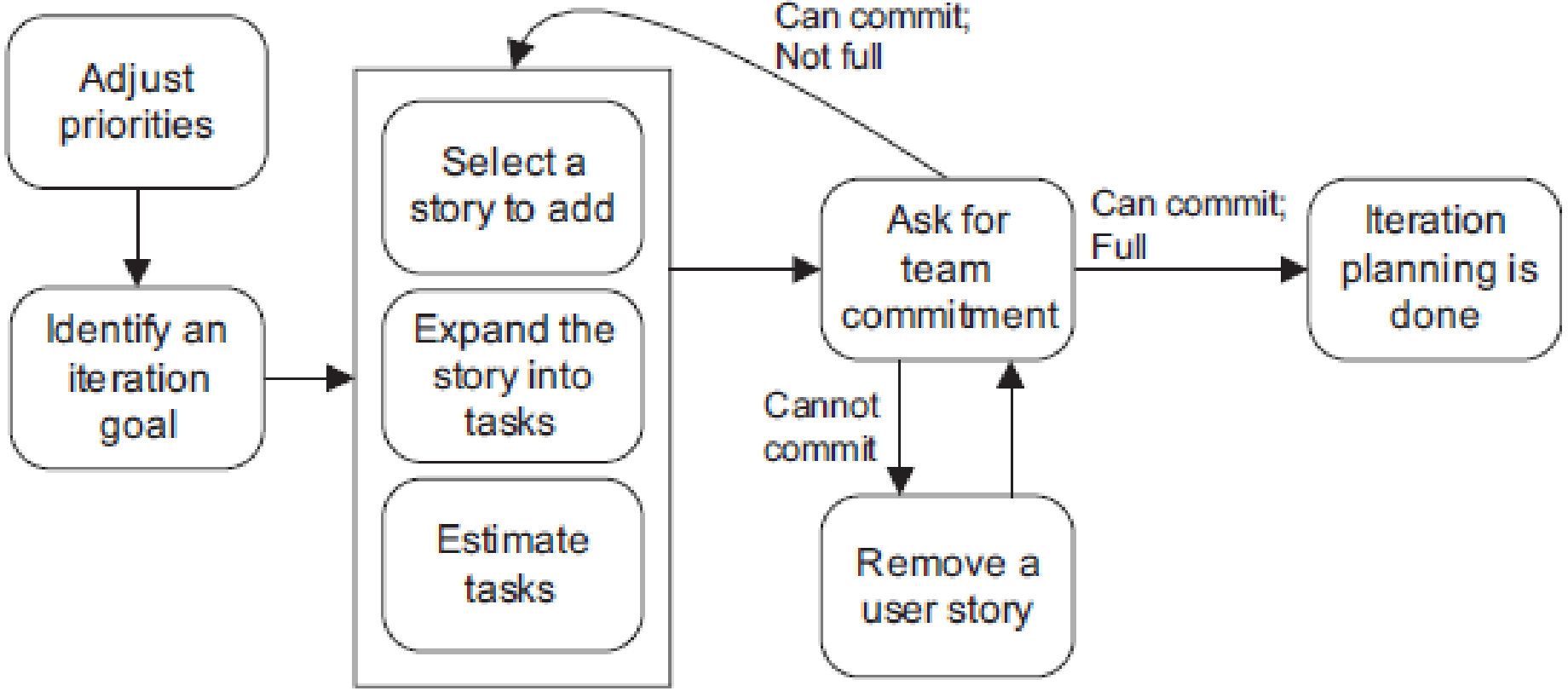
- Una *spike* es una tarea incluida en el plan de la iteración que se hace para ganar conocimiento o responder una pregunta.

ESTIMACIÓN DE LAS TAREAS

- En tiempo ideal.
- Tarea del equipo:
  - las tareas no son asignadas a una persona específica.
  - aunque pensemos que lo va a hacer una persona determinada, los demás podemos contribuir.
  - oír cuánto va a llevar ayuda a identificar malentendidos sobre la historia o la tarea.
  - Cuando la estimación se hace colaborativamente, hay menor renuencia a admitir que falló.
- El tamaño de cada tarea debería ser tal que cada desarrollador pueda terminar en promedio una por día.

PLANIFICACIÓN DE LA ITERACIÓN GUIADA POR EL COMPROMISO

- En lugar de crear un plan que intente completar tantas PH o DI como fueron completados en la iteración precedente, agregar historias a la iteración una a una hasta que no se pueden comprometer a más.


Adjust priorities
Identify an iteration goal
Select a story to add
Expand the story into tasks
Estimate tasks
Ask for team commitment
Can commit; Not full
Can commit; Full
Cannot commit
Remove a user story
Iteration planning is done

RECOMENDACIÓN

- Usar velocidad para la planificación de la release y usar compromiso para la planificación de la iteración.
- Razones:
  - velocidad tiene baja granularidad para iteraciones muy cortas.
  - se deberían completar de 20-30 historias por iteración para que los errores se compensen. Pocos equipos completan tantas historias (3-12).

CAP. 15
SELECCIÓN DE LA DURACIÓN DE LA ITERACIÓN

DURACIÓN DE LA ITERACIÓN

- La mayoría de los equipos ágiles trabajan en iteraciones de 1-4 semanas.
- No hay una duración de la iteración que le sirva a todos los equipos. Cada equipo debería considerar su situación única y elegir la duración que les sirva.

FACTORES EN LA SELECCIÓN DE LA DURACIÓN DE LA ITERACIÓN

- Factores que influyen en esta decisión:
  - El largo de la release en la que estamos trabajando.
  - La cantidad de incertidumbre
  - La facilidad para obtener feedback
  - Cuánto tiempo pueden mantenerse incambiadas las prioridades
  - La disposición a seguir sin feedback
  - El costo adicional de iterar
  - La necesidad de mantener un sentido de urgencia

CAP. 16
ESTIMACIÓN DE LA VELOCIDAD

- Hay 3 formas de estimar la velocidad:
  1. Primero se pueden usar promedios históricos, si se tienen. Sin embargo, antes de usar promedios históricos, se debería considerar si hay cambios significativos en el equipo, la naturaleza del proyecto, la tecnología, etc.
  2. Segundo, se puede diferir estimar la velocidad hasta que se hayan corrido algunas iteraciones. Esta es generalmente la mejor opción.
  3. Se puede pronosticar la velocidad refinando algunas historias en tareas y viendo cuánto entraría en una iteración. Este proceso es muy similar a la planificación de la iteración.
- Independientemente de cuál enfoque se use, las estimaciones de la velocidad deben ser dadas en un rango que refleje la incertidumbre inherente a la estimación. El cono de incertidumbre puede servir para ver el tamaño del rango a usar.

SEGUIMIENTO Y COMUNICACIÓN

CAP. 19
SEGUIMIENTO DEL PLAN DE LA *RELEASE*

SEGUIMIENTO DE LA RELEASE

- Plan de la release:

  «En los próximos **4 meses** y **8 iteraciones** de **2 semanas**
  completaremos aprox. **240 puntos de historia**
  (o días ideales) de trabajo.»

- Queremos evaluar dónde estamos:
  - Avance (trabajo completado)
  - Si hay cambios en el alcance
  - Si hay que reestimar

MEDIDA DE AVANCE: LA VELOCIDAD

- La velocidad mide la cantidad de trabajo completado en cada iteración.
- Completado:
  - Código bien escrito y bien factorizado
  - Chequeado
  - Limpio
  - Que cumple los estándares de codificación
  - Que pasó todas las pruebas.
- Se mide en puntos de historia o días ideales.
- Se calcula con regla todo o nada (0/100):
  - Si una historia está totalmente terminada, se suma todo el valor estimado.
  - Si está parcialmente completa, no se cuenta.

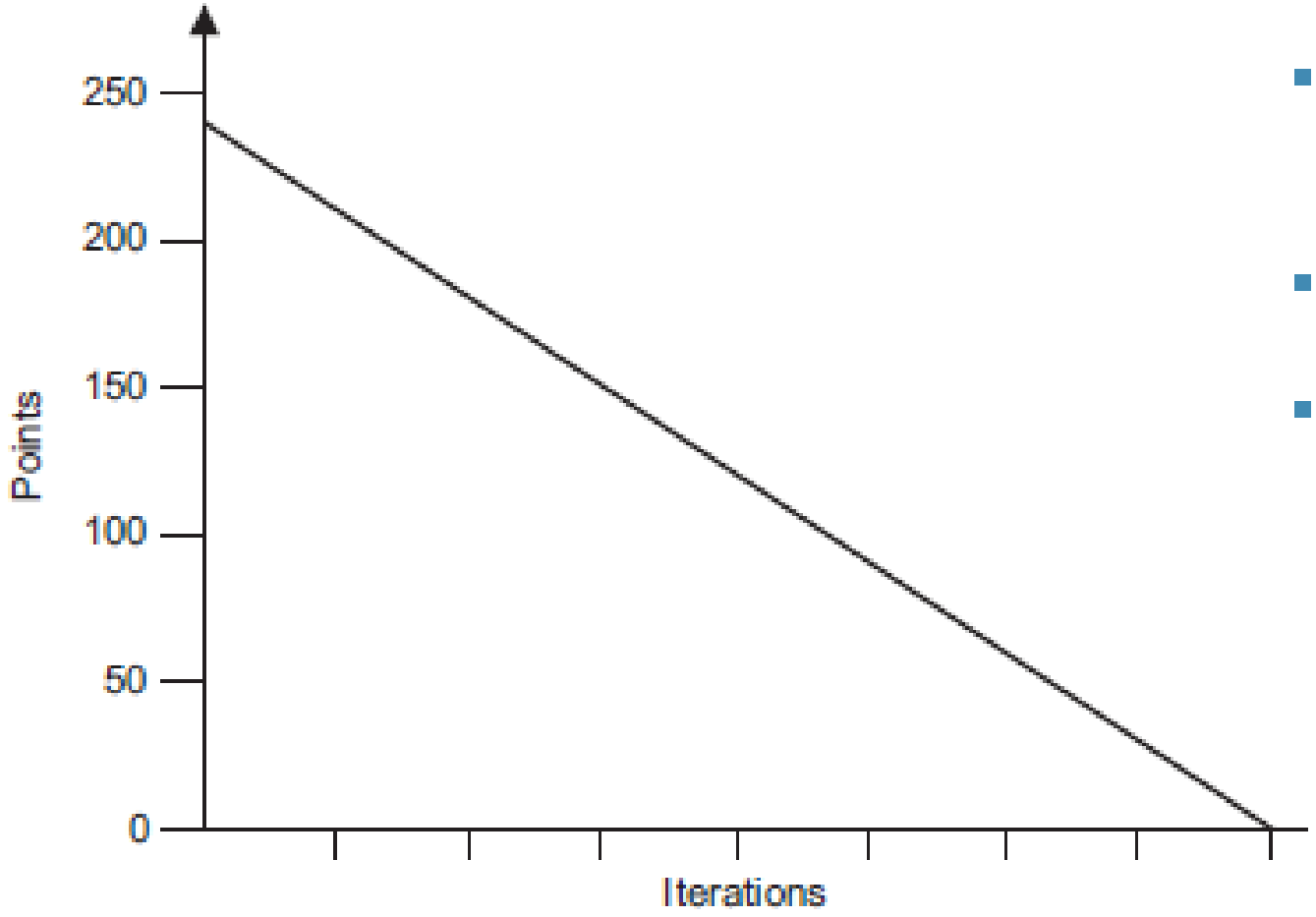
HISTORIAS INCOMPLETAS

- Trabajo incompleto:
  1. difícil de medir.
  2. Rompe la confianza entre el cliente y el equipo de desarrollo.
     - Sacarla de la iteración o partirla y sacar parte.
     - Cambiar criterios de aceptación
  3. Aumenta el trabajo en curso →
     - disminuye el tiempo de entrega
     - Retrasa la retroalimentación
- ¿Por qué?
  - Estimación por debajo de lo real
  - Pusimos demasiadas historias en la iteración

GRÁFICA *BURN-DOWN* DE LA *RELEASE*

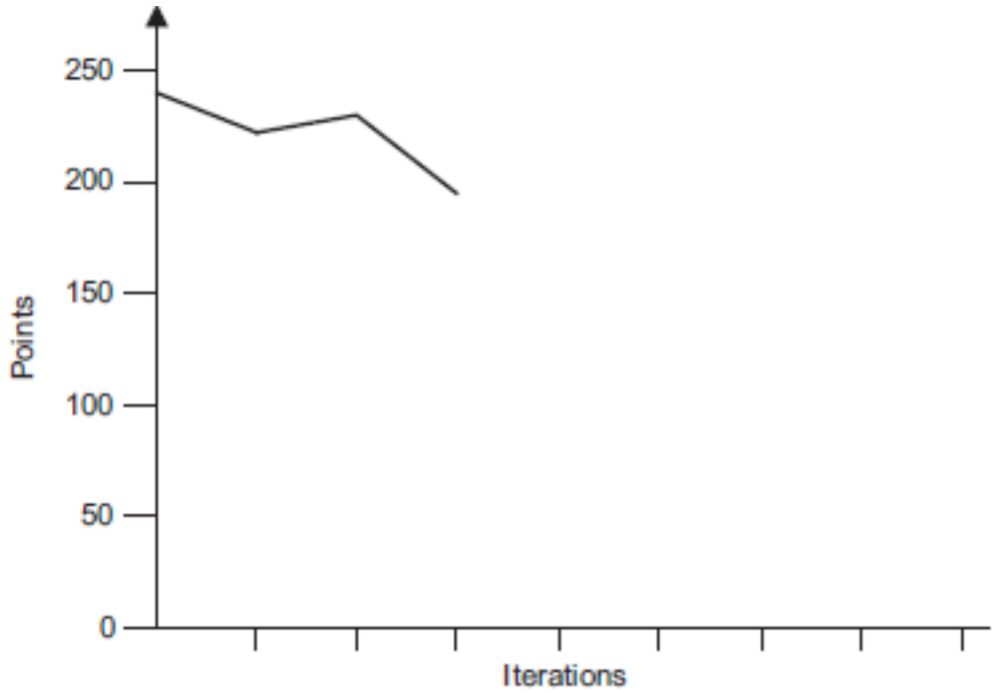
- Muestra la cantidad de puntos de historia o de días ideales que faltan en el proyecto al comienzo de cada iteración.
- Cifras importantes:
  - Trabajo pendiente en la release
  - Tasa de avance del grupo (= velocidad)

GRÁFICA *BURN-DOWN* DE LA *RELEASE*



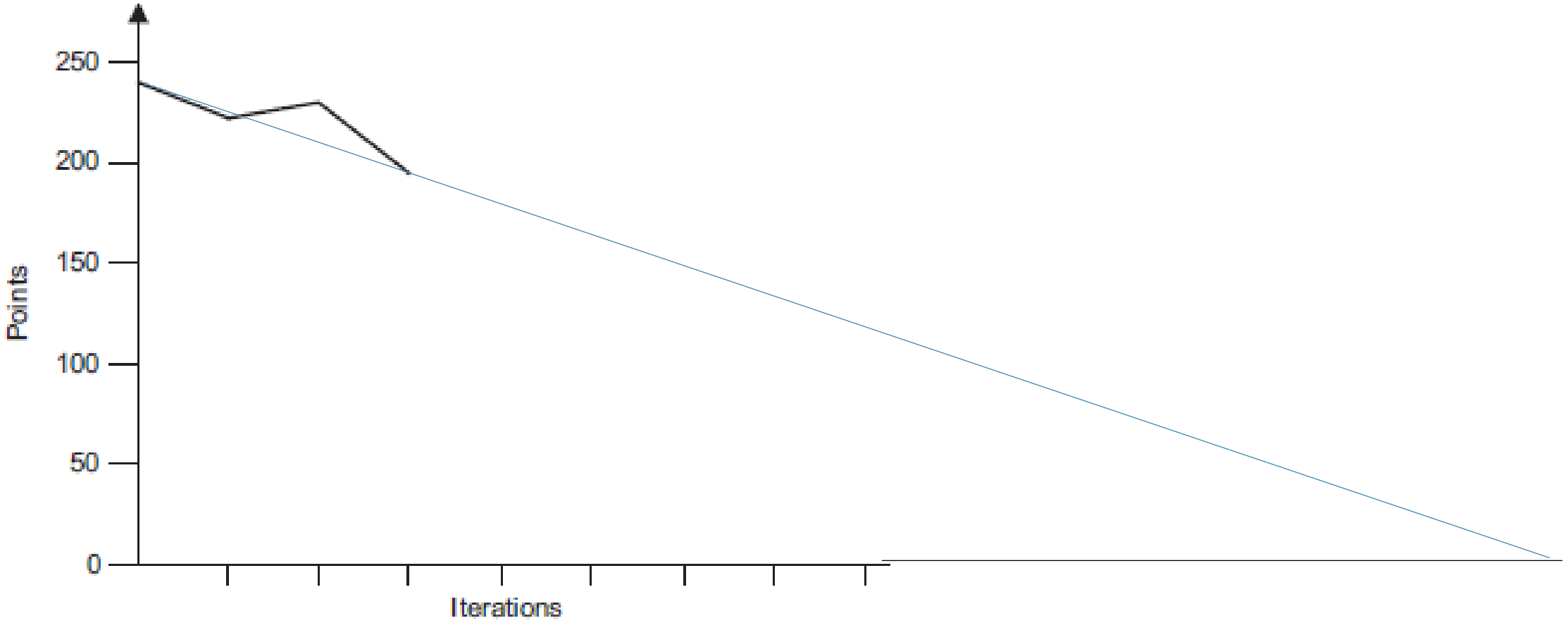
- Total: 240 puntos de historia
- 8 iteraciones
- VE = 30 P. H.

- Nunca es una línea recta.
- Varía por:
  - Malas estimaciones
  - Cambios en las estimaciones
  - Cambios en el alcance



  - Puede mostrar un *burn-up* durante la iteración. Esto significa:
    - O aumentaron las estimaciones
    - O aumentó el alcance

PREDICCIÓN

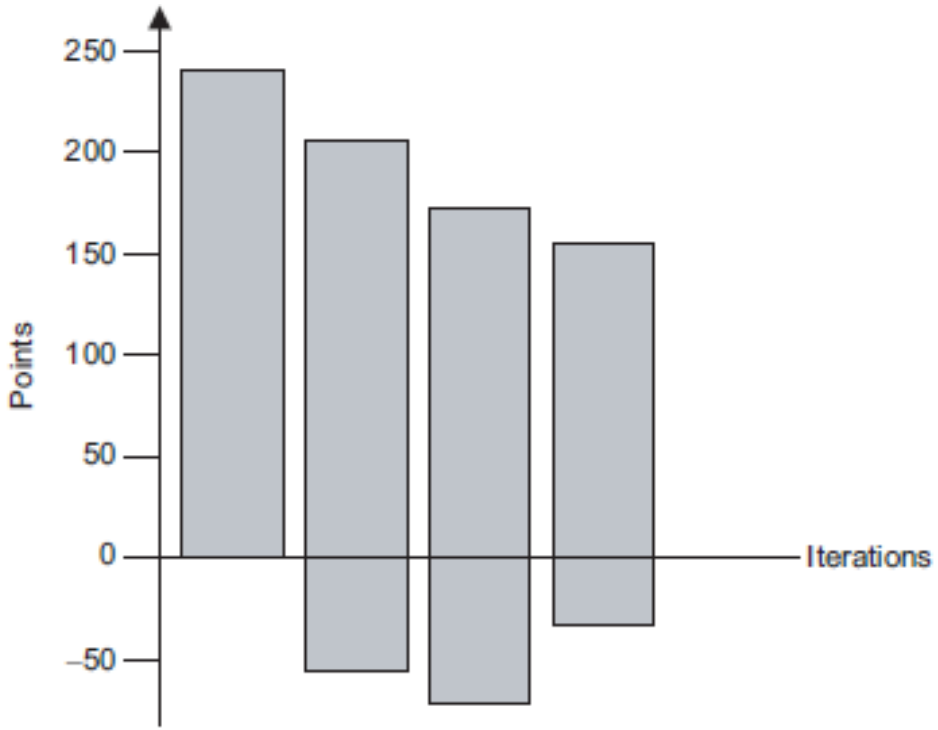


- Va a llevar más tiempo que el previsto.
- Nos indica dónde terminará si los factores no cambian.

GRÁFICA DE BARRAS *BURN-DOWN* DE LA *RELEASE*

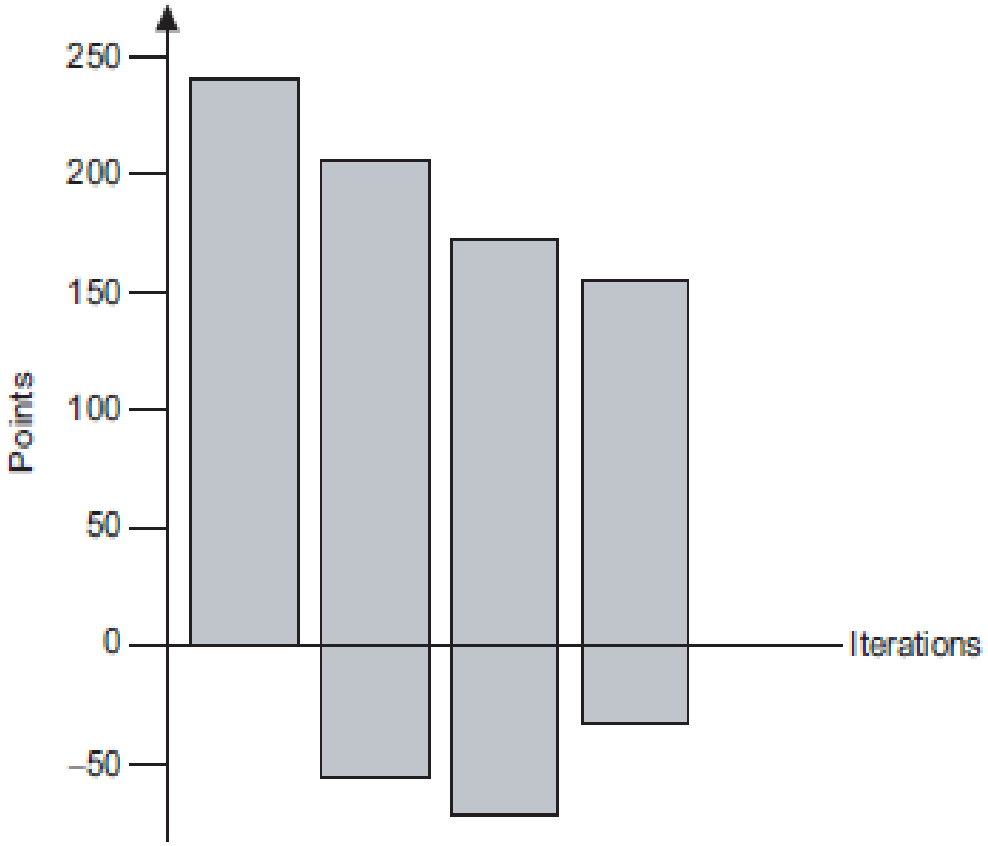
- Separa el avance de los cambios en el alcance de la release.

GRÁFICA DE BARRAS *BURN-DOWN*



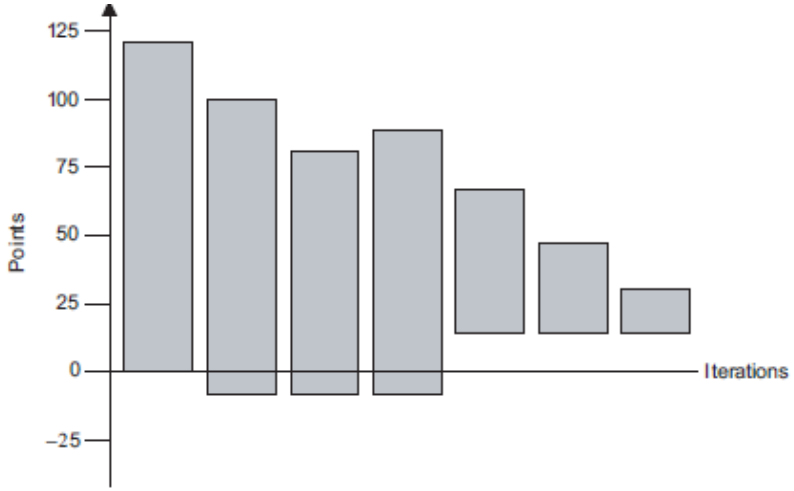
- Los cambios en el alcance se muestran bajando la barra por debajo del eje de las x.
- Cuatro reglas:
  - Se baja el tope superior cuando se complete trabajo.
  - Se baja o sube el tope cuando se reestima.
  - Se baja el fondo cuando se agrega alcance.
  - Se sube el fondo cuando se disminuye el alcance.

EJEMPLO - INTERPRETACIONES



1. La velocidad del equipo es la estimada. Se ve en el tope de las dos primeras barras.
2. Se agregó mucho trabajo. Se ve en el fondo de la 2.a barra.
3. Se ha agregado más trabajo del que ha sido completado. Dependerá de si luego la tendencia se confirma, o de cuán importante sea la fecha inicial del a release.
4. La cantidad total de trabajo pendiente en la release es mayor que cuando el proyecto empezó, porque el largo de la barra 2 es mayor que el de la barra 1.
5. En la 3.a iteración se agregó alcance, pero menos que la vez anterior.
6. En la 3.a iteración la velocidad bajó de 30 a 20, por
   - Se subestimaron algunas historias de las hechas
   - Un integrante enfermo o de licencia
   - Se reestimaron historias del alcance pendiente.

EJERCICIO

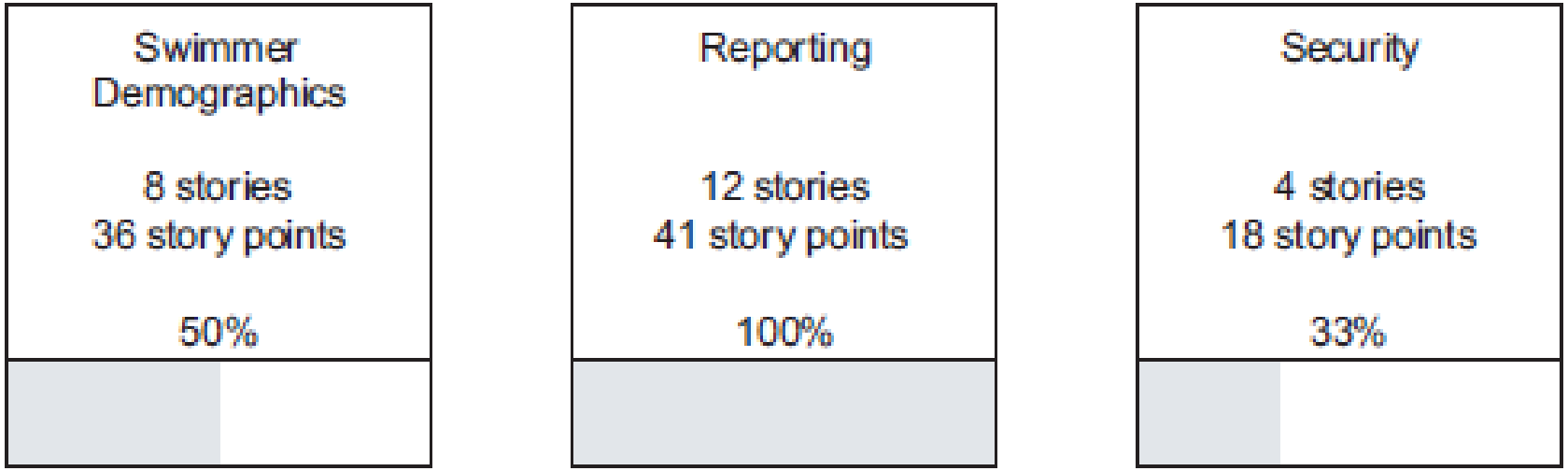


- ¿Velocidad en cada iteración?
- ¿Agregó el product owner trabajo? ¿En qué iteración?
- ¿Qué pasó durante la tercera iteración?
- ¿Redujo el product owner el alcance de la release? ¿Cuándo?

GRÁFICA DE LOTE DE ESTACIONAMIENTO

GRÁFICA DE LOTE DE ESTACIONAMIENTO

- Colocar una caja rectangular para cada tema de la release, con:
  - El nombre del tema
  - La cantidad de historias del tema
  - La cantidad de puntos de historia o días ideales para esas historias
  - El porcentaje de puntos de historia completas.



GRÁFICA DE LOTE DE ESTACIONAMIENTO

- No podemos decir cuáles historias están completas.
- Las cajas se pueden colorear para indicar si un tema
  - está completo,
  - está en tiempo,
  - necesita atención, o
  - está significativamente atrasado.
- Es útil para mostrar una vista de alto nivel del avance del equipo en la implementación de los temas planificados de la release.
- Se puede mostrar en una sola página.

CAP. 20
SEGUIMIENTO DEL PLAN DE LA ITERACIÓN

EL PANEL DE TAREAS

- Es un pizarrón blanco, o un panel de corcho o un espacio en en la pared.
- Ayuda al equipo a organizar su trabajo y a visualizar el trabajo pendiente.
- Las columnas están etiquetadas.
- Los miembros mueven las tarjetas de tareas entre las columnas a medida que el trabajo avanza.

EL PANEL DE TAREAS

- Los integrantes no se autoasignan o son asignados trabajo hasta que estén prontos para hacerlo.
- El panel de tareas permite que todos vean en qué tareas se está trabajando y cuáles están disponibles para tomar.

Story	To Do	Tests Ready	In Process	To Verify	Hours
As a user, I can… 5	Code the... 8 Code the... 5 Test the... 6	√	Code the... SC 6 Code the... DC 4	Code the... LC 4	33
As a user, I can… 2	Code the... 8 Code the... 5				13
As a user, I can… 3	Code the... 3 Code the... 6	√	Code the... MC 4		13

FILAS

Story	To Do	Tests Ready	In Process	To Verify	Hours
As a user, I can… 5	Code the... 8 Code the... 5 Test the... 6	√	Code the... SC 6 Code the... DC 4	Code the... LC 4	33
As a user, I can… 2	Code the... 8 Code the... 5				13
As a user, I can… 3	Code the... 3 Code the... 6	√	Code the... MC 4		13

- Se colocan las tarjetas de historia y las tarjetas de tareas.
- Una fila por cada historia.

COLUMNAS

Story	To Do	Tests Ready	In Process	To Verify	Hours
As a user, I can… 5	Code the... 8 Code the... 5 Test the... 6	√	Code the... SC 6 Code the... DC 4	Code the... LC 4	33
As a user, I can… 2	Code the... 8 Code the... 5				13
As a user, I can… 3	Code the... 3 Code the... 6	√	Code the... MC 4		13

1. Historia
   - Contiene la tarjeta de historia.
   - Y la cantidad de puntos de historia.

COLUMNAS

Story	To Do	Tests Ready	In Process	To Verify	Hours
As a user, I can… 5	Code the... 8 Code the... 5 Test the... 6	√	Code the... SC 6 Code the... DC 4	Code the... LC 4	33
As a user, I can… 2	Code the... 8 Code the... 5				13
As a user, I can… 3	Code the... 3 Code the... 6	√	Code the... MC 4		13

2. Para hacer
   - Contiene todas las tarjetas de tarea para implementar la historia.
   - Cada una con su estimación.
   - Ancho de la columna puede ser más grande que una tarjeta.

COLUMNAS

Story	To Do	Tests Ready	In Process	To Verify	Hours
As a user, I can… 5	Code the... 8 Code the... 5 Test the... 6	√	Code the... SC 6 Code the... DC 4	Code the... LC 4	33
As a user, I can… 2	Code the... 8 Code the... 5				13
As a user, I can… 3	Code the... 3 Code the... 6	√	Code the... MC 4		13

3. Pruebas especificadas
   - Indica si las pruebas de aceptación están prontas para esa historia.
   - Test-driven development:
     - Escribir una prueba de falla antes de codificar.
     - Diseñar pruebas de aceptación de alto nivel para el feature antes de codificar.
     - Las condiciones de satisfacción de cada historia (identificadas en la planificación de la iteración) son pruebas de aceptación de la historia de alto nivel.
   - La columna se chequea si las pruebas de aceptación de alto nivel de la historia están especificadas.
   - No mover tarjetas a la columna En curso, a menos que las pruebas estén especificadas.

COLUMNAS

Story	To Do	Tests Ready	In Process	To Verify	Hours
As a user, I can… 5	Code the... 8 Code the... 5 Test the... 6	√	Code the... SC 6 Code the... DC 4	Code the... LC 4	33
As a user, I can… 2	Code the... 8 Code the... 5				13
As a user, I can… 3	Code the... 3 Code the... 6	√	Code the... MC 4		13

4. En curso
   - Contiene las tarjetas que han sido tomadas. Cada desarrollador toma una tarjeta de la columna Para hacer, pone sus iniciales y la mueve a la columna En curso.
   - A medida que va quedando libre.
   - No tomar más de una tarjeta al mismo tiempo. Eso ayuda a mantener un flujo de trabajo consistente y reduce el costo de cambiar de contexto entre múltiples tareas. → ancho de la columna no mayor a una tarjeta.

COLUMNAS

Story	To Do	Tests Ready	In Process	To Verify	Hours
As a user, I can… 5	Code the... 8 Code the... 5 Test the... 6	√	Code the... SC 6 Code the... DC 4	Code the... LC 4	33
As a user, I can… 2	Code the... 8 Code the... 5				13
As a user, I can… 3	Code the... 3 Code the... 6	√	Code the... MC 4		13

4. Para verificar
   - Típicamente hay una tarjeta para la tarea de verificar cada historia.
   - Se puede tomar cuando la tarea de codificar la historia llega a la columna Done.
   - Pero si no hay tarjeta o si el implementador quiere que alguien la revise, la coloca en esta columna.

COLUMNAS

Story	To Do	Tests Ready	In Process	To Verify	Hours
As a user, I can… 5	Code the... 8 Code the... 5 Test the... 6	√	Code the... SC 6 Code the... DC 4	Code the... LC 4	33
As a user, I can… 2	Code the... 8 Code the... 5				13
As a user, I can… 3	Code the... 3 Code the... 6	√	Code the... MC 4		13

5. Terminada
   - Cuando se terminó la tarea

COLUMNAS

Story	To Do	Tests Ready	In Process	To Verify	Hours
As a user, I can… 5	Code the... 8 Code the... 5 Test the... 6	√	Code the... SC 6 Code the... DC 4	Code the... LC 4	33
As a user, I can… 2	Code the... 8 Code the... 5				13
As a user, I can… 3	Code the... 3 Code the... 6	√	Code the... MC 4		13

6. Horas

- Suma de las horas de trabajo restantes para la historia.
- Los desarrolladores pueden en cualquier momento
  - cambiar la estimación (tachar y colocar una nueva) de cualquier tarea
  - Romper una tarjeta de tarea y reemplazarla por otras dos o tres, cada una con su propia estimación
- Cada mañana sumar las horas de cada columna y usarlas para la gráfica *burn-down* de la iteración.

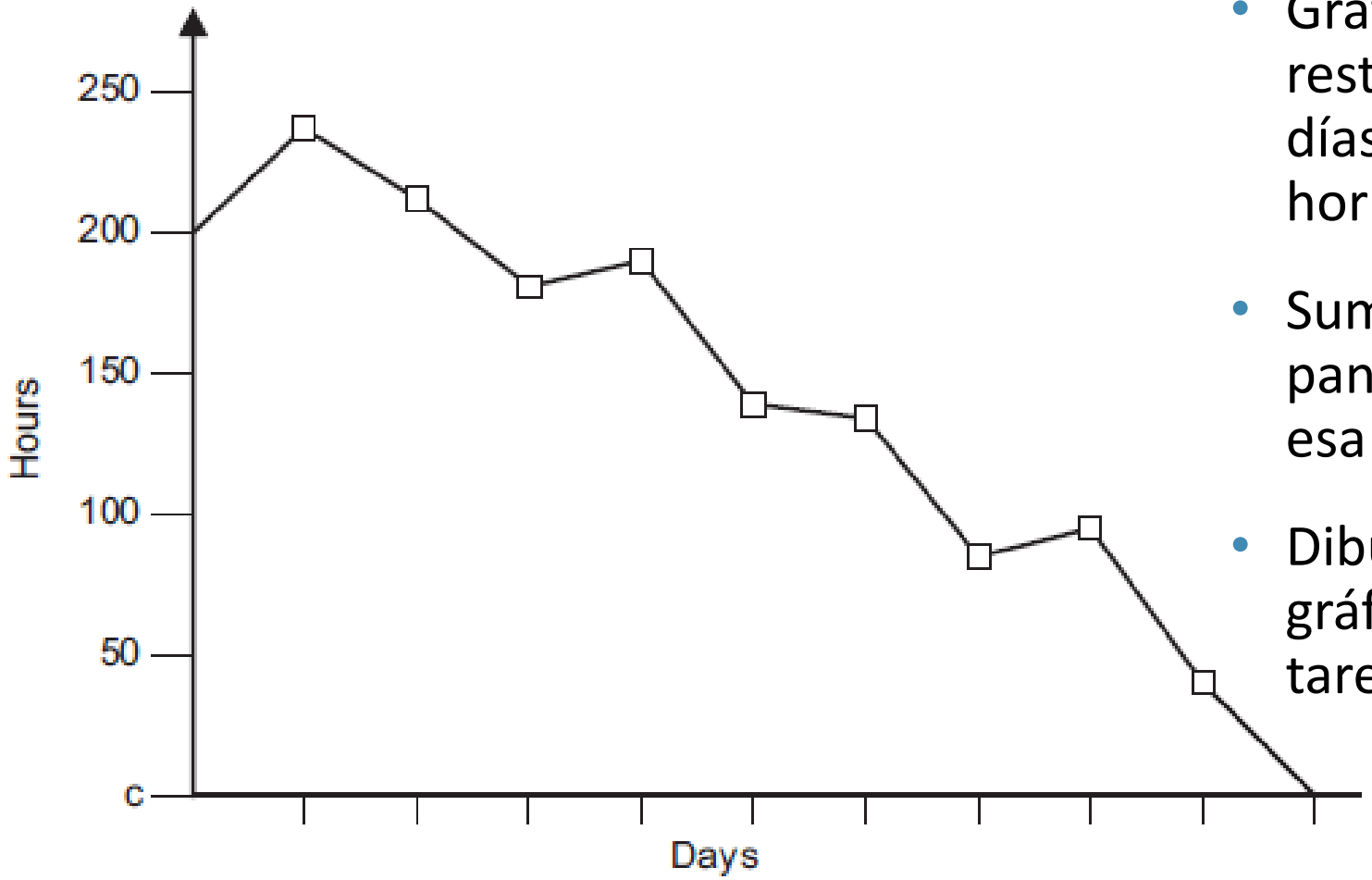
SEGUIMIENTO DE *BUGS* EN UN PANEL DE TAREAS

- Al comienzo de la iteración:
  - Se seleccionan los bugs a corregir.
  - Escribir una tarjeta de tarea para cada bug con su estimación.
  - Se colocan todas en una única fila.
  - El product owner decide qué porcentaje de la iteración se dedica a corrección de bugs.
- Defectos descubiertos en la iteración de alta prioridad:
  - Se estima
  - Se agregan al tablero
  - Se quita de la columna To Do una cantidad de trabajo equivalente.

GRÁFICA *BURN-DOWN* DE LA ITERACIÓN

- Es similar a la gráfica *burn-down* de la *release*, pero solo hace el seguimiento del trabajo en la iteración en curso.
- Grafica la cantidad de horas restantes por día.

- Si iteraciones de una semana:
  - No es útil: para cuando se vean las tendencias, la iteración ya habrá terminado.
- Iteraciones de 2 semanas o más:
  - Es útil.



- Grafica cantidad de horas restantes en el eje vertical y los días de la iteración en el eje horizontal.
- Sumar todas las horas en el panel una vez al día y colocar esa cifra en la gráfica.
- Dibujar o agregar en papel la gráfica diariamente al panel de tareas.

SEGUIMIENTO DEL ESFUERZO REALIZADO

- Es más útil saber cuánto falta que cuánto se hizo.
- Saber cuánto se dedicó no siempre ayuda a mejorar la estimación.

VELOCIDAD INDIVIDUAL

- Es la cantidad de puntos de historia o días ideales completados por un integrante.
- No hacer seguimiento de la velocidad individual, sino de la del equipo.
- Seguir la velocidad individual no incentiva el trabajo en equipo.

CAP. 21
COMUNICACIÓN DE LOS PLANES

COMUNICACIÓN DEL PLAN

- La comunicación de estimaciones y planes debe ser frecuenta, honesta y en ambas direcciones.

LA COMUNICACIÓN DEL PLAN

- La comunicación de plan incluye:
  - Fecha
  - Cantidad de iteraciones
- Agregar:
  - Grado de confianza en al estimación
    - Estoy 90 % seguro de que completaremos la funcionalidad planificada para el 31 de julio.
  - Rango de posibles fechas
    - Según nuestras estimaciones del tamaño y rendimiento del equipo, el proyecto llevará de 3 a 4 meses.
  - Ambos

DIAGRAMA DE GANTT POR *FEATURES*

- Un diagrama de Gantt por *features*:
  - Muestra desglose de *features* y no el WBS.
  - Cada feature abarca toda la iteración.
  - No hay asignación de recursos porque todo el equipo es responsable.
  - Sí hay asignación de equipos si hay múltiples equipos.

ID	Description	Start	End	Chart
1	**Iteration 1**	July 1	July 14	
2	As a user, I want ...	July 1	July 14	
3	As a user, I want ...	July 1	July 14	
4	As a user, I want ...	July 1	July 14	
5	**Iteration 2**	July 15	July 29	
6	As a user, I want ...	July 15	July 29	
7	As a user, I want ...	July 15	July 29	
8	As a user, I want ...	July 15	July 29	
9	As a user, I want ...	July 15	July 29	

COMUNICACIÓN DEL AVANCE

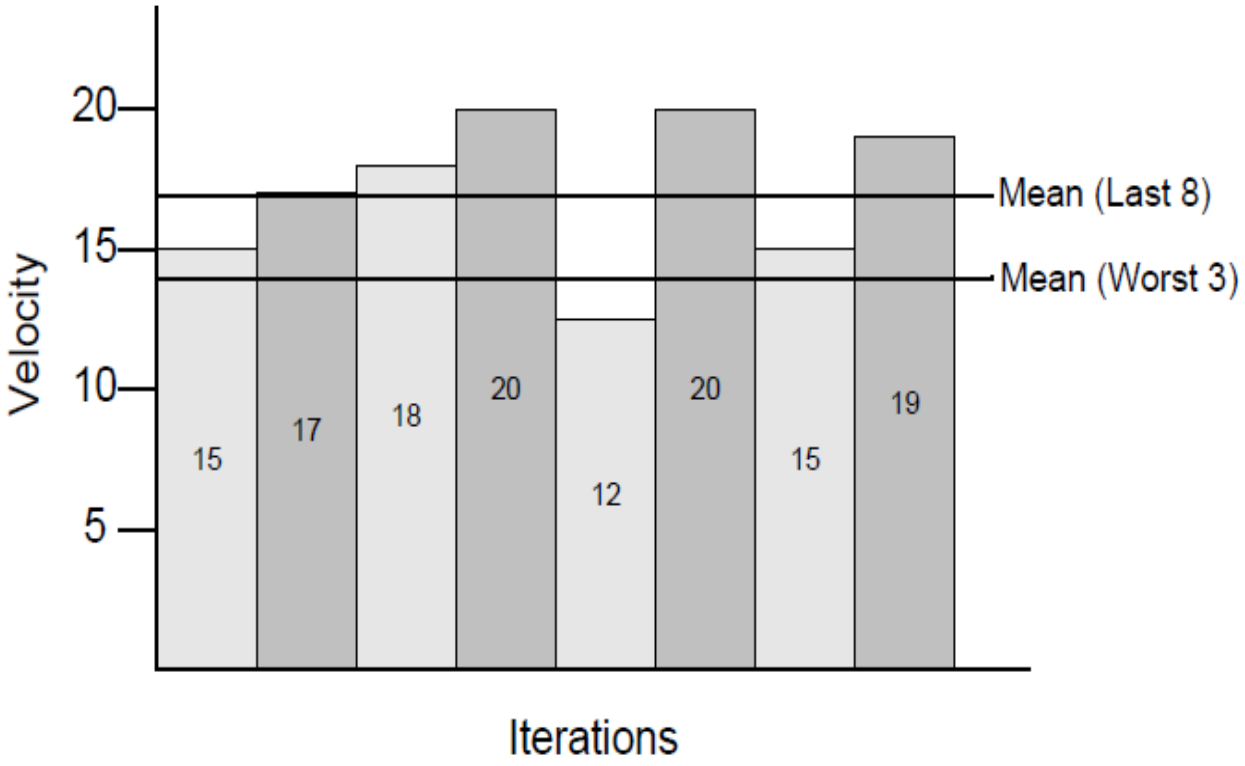
GRÁFICAS *BURN-DOWN*

- Las gráficas *burn-down* son el principal método para comunicar el avance.
- Son una función de la cantidad de trabajo pendiente y la velocidad del equipo.

PREDICCIÓN DE LA CANTIDAD DE ITERACIONES RESTANTES

- Para predecir la cantidad de iteraciones restantes dividir la cantidad de puntos de historia pendientes entre la velocidad del equipo.
- P. ej. 100 puntos pendientes. VE = 15. Iteraciones restantes = 100/15 = 6,66 → 7
- Sin embargo, la velocidad es fluctuante → usar un rango de velocidades para predecir la cantidad de iteraciones faltantes.
- Además, gráfica que muestre la velocidad del equipo por iteración.
- Pensar en un rango de velocidades.
- Usar:
  - La velocidad de la última iteración (muestra lo que acaba de pasar)
  - El promedio de las 8 iteraciones previas (muestra un promedio a largo plazo)
  - El promedio de las tres iteraciones con velocidad más baja de entre las últimas 8 (peor caso)

EJEMPLO



- La velocidad de la última iteración = 19
- Promedio de velocidad durante últimas 8 iteraciones = 17
- Promedio de las 3 velocidades más bajas en las 8 últimas iteraciones = 14

PREDICCIONES

Description	Velocity	Iterations	Total Points
Worst three	14	5	70
Last eight	17	5	85
Most recent	19	5	95

- El product owner podrá tener la certeza e que al menos obtendrá 70 puntos de historia adicionales durante las 5 iteraciones siguientes.
- No debería comprometerse con nada más allá de 85 puntos de historia.

TENDENCIAS DE LA VELOCIDAD

- Si se nota una tendencia a una velocidad en aumento:
  - no hacer nada al respecto.
- Si la velocidad parece estar disminuyendo:
  - buscar y eliminar la causa del declive
  - No planificar en una velocidad descendiendo

INFORME DE FIN DE ITERACIÓN

- Es útil
  - para difundir la información actual y
  - como documento histórico para usar en el futuro.
- Hacer el resumen lleva 30 m por iteración.

CONTENIDO DEL INFORME DE FIN DE LA ITERACIÓN

1. Contexto
   - Fechas
     - Fecha de inicio de la iteración
     - Fecha de fin de la iteración
     - Cantidad de días de la iteración
   - Personal
     - Personal disponible durante la iteración
     - Cantidad de días de trabajo planificados (puede variar de iteración en iteración por licencias)
     - cantidad de días de trabajo reales (puede variar por enfermedad o días planificados de licencia que no se tomaron)
2. Métricas
   - Resultados de lo construido por día:
     - Cantidad de pruebas, si todas fueron exitosas
     - Cantidad de pruebas fallidas, si hubo una fallida
   - Gráfica *burn-down* de la iteración
   - Velocidad
3. Contenidos y evaluaciones
   - Resultados:
     - Historias, resultados, P. H. estimados, P. H. ganados
   - Revisión de la iteración:
     - Fecha y hora
     - Ítems identificados.

CONTEXTO

- Fechas

First Day of Iteration	September 1
Last Day of Iteration	September 14
Number of Working Days	9

- Personal

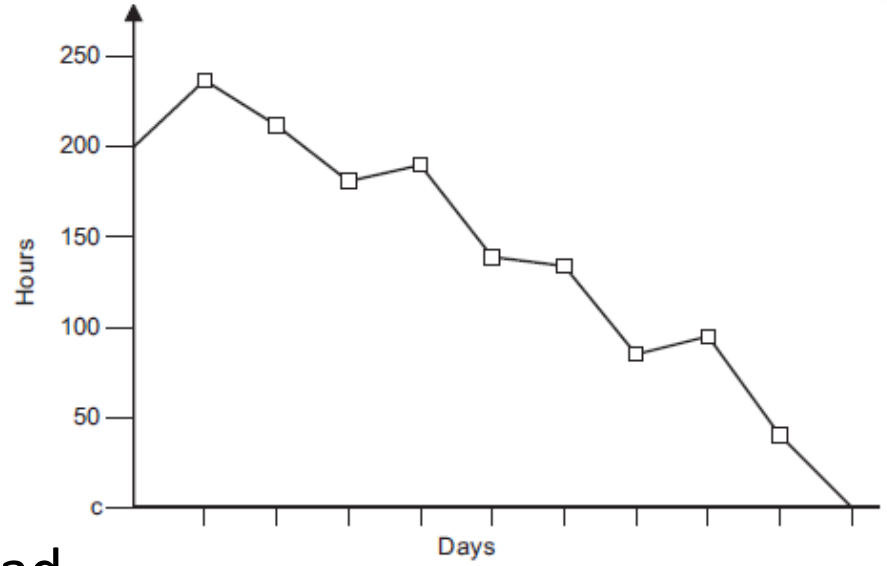
Name	**Days Planned**	**Days Worked**
Carina	9	9
Vadim	9	7
Sasha	8	9
Dmitri	9	9
Total	**35**	**35**

MÉTRICAS

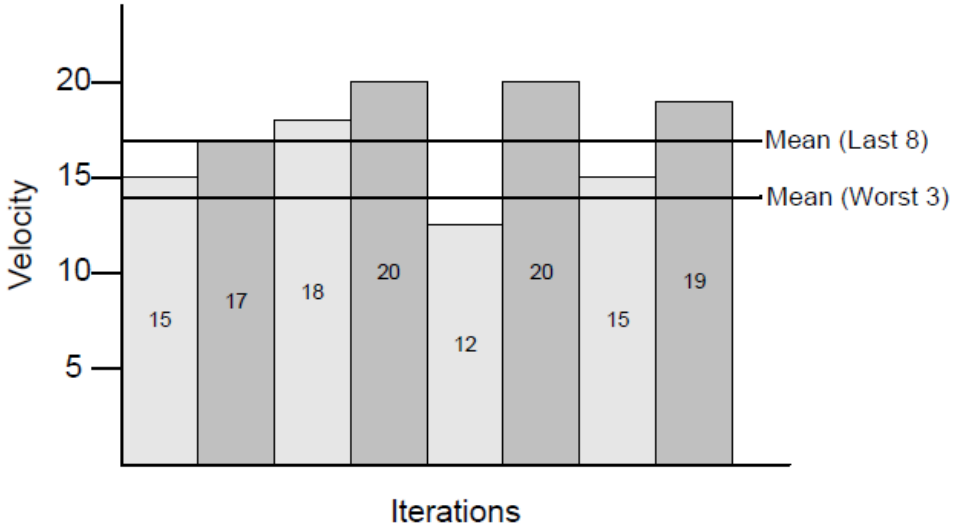
- Resultados de lo construido por día:

Metric	Status
Monday, June 6	Build failed
Tuesday, June 7	812 unit tests passed; 1431 FitNesse tests passed; 104 user interface tests passed
Wednesday, June 8	826 unit tests passed; 1433 FitNesse tests passed; 104 user interface tests passed
Thursday, June 9	1 unit test failed
Friday, June 10	841 unit tests passed; 1442 FitNesse tests passed; 105 user interface tests passed
…	…

- Gráfica *burn-down* de la iteración



Velocidad



CONTENIDOS Y EVALUACIONES

- Resultados:

Story	Result	Points Planned	Points Earned
As a coach, I can enter the names and demographic information for all swimmers on my team.	Finished	8	8
As a coach, I can define practice sessions.	Finished; bigger than planned	8	8
As a swimmer, I can see all of my times for a specific event.	Finished	3	3
As a swimmer, I can update my demographics information.	Started; not finished	1	0
Total		**20**	**19**

- Revisión de la iteración:
- La revisión de la iteración tuvo lugar el 15 de setiembre a las 9 h.
- Se identificaron los siguientes ítems:

Action Item	Responsible
Mark pointed out that the session planning logs used by most coaches he knows list information in the opposite order we show on the screen. We need to run those screens by some coaches.	Carina
Gary suggested we add a story about graphing a swimmer's results in an event.	Carina

POR QUÉ SIRVE LA PLANIFICACIÓN ÁGIL

CAP. 22
POR QUÉ SIRVE LA PLANIFICACIÓN ÁGIL

12 GUÍAS PARA LA ESTIMACIÓN Y PLANIFICACIÓN ÁGILES

1. Involucra a todo el equipo.
2. Planifica en diferentes niveles (*release*, iteración, planes diarios). Cada plan cubre un horizonte de tiempo distinto, un nivel de precisión diferente y sirve para un propósito único.
3. Usa diferentes unidades para las estimaciones de tamaño (puntos de historia) y duración (transforma PH en duración usando la velocidad).

12 GUÍAS PARA LA ESTIMACIÓN Y PLANIFICACIÓN ÁGILES

4. Expresa la incertidumbre en la funcionalidad o en la fecha, en el plan de la *release*.
   - Si la cantidad de funcionalidad está fija, expresa la incertidumbre con un rango de fechas. Terminaremos en el primer cuatrimestre; terminaremos en 7-10 iteraciones.
   - Si la fecha está fija, expresa la incertidumbre acerca de la funcionalidad exacta que se entregará. P. je. Terminaremos para el 31 de diciembre y el producto incluirá al menos estos *features*, pero probablemente no más de estos otros *features*.
   - Usa unidades más grandes (iteraciones, meses, cuatrimestres) cuanto mayor la incertidumbre.

12 GUÍAS PARA LA ESTIMACIÓN Y PLANIFICACIÓN ÁGILES

5. Replanifica a menudo.
   - Revisa la relevancia del plan de la *release* en cada iteración.
6. Haz el seguimiento y comunica el avance.
   - Publica regularmente indicadores de avance simples y comprensibles.
7. Reconoce la importancia del aprendizaje
   - Actualiza los planes para incluir el conocimiento nuevo:
     - Si aprendes más de las necesidades del cliente, agrega nuevos *features*.
     - Si aprendes más de las tecnologías o cuán bien estamos trabajando, ajusta las expectativas de la tasa de avance.

12 GUÍAS PARA LA ESTIMACIÓN Y PLANIFICACIÓN ÁGILES

8. Planifica *features* del tamaño correcto:
   - La funcionalidad que se implementarán en las próximas iteraciones deberá descomponerse en historias de usuario pequeñas (que lleven entre 1-2 días, no más de 10).
   - Estimar dentro de un orden de magnitud.
   - Historias que puedan ser desarrolladas en una iteración.
   - Al crear el plan de la release para más de 2 o 3 meses, escribir épicas o estimar el trabajo al nivel de temas, para evitar descomponer historias grandes con demasiada antelación.

12 GUÍAS PARA LA ESTIMACIÓN Y PLANIFICACIÓN ÁGILES

9. Prioriza las *features*, de modo que el orden optimice el valor total del proyecto. Considera
   - valor,
   - costo,
   - el aprendizaje (del producto o del esfuerzo) y
   - la reducción del riesgo.

12 GUÍAS PARA LA ESTIMACIÓN Y PLANIFICACIÓN ÁGILES

10. Basa tus estimaciones y planes en hechos (0-100 %)
11. Deja holgura. No planifiques usar el 100 % del tiempo del equipo.
12. Coordina equipos mediante la planificación anticipatoria escalonada. Asigna features específicos a iteraciones específicas, para plenificar y acomodar las dependencias entre los equipos.