



Ingeniería de software⁶

Yadran Eterovic S. (yadran@ing.puc.cl)

Planificación del proyecto: los desafíos

¿Cómo **planificamos, programamos y gestionamos** el trabajo que hay que hacer **sin tener un conocimiento completo** de lo que el cliente quiere, y lo que va a pasar en los próximos meses?

¿Cómo nos aseguramos de que haya una **comunicación y coordinación adecuada** entre los departamentos y equipos?

¿Cómo **dividimos el trabajo** y asignamos las partes interdependientes a los departamentos y a los equipos, y cómo **integramos las componentes producidas**?

Para planificar un proyecto, hay que tener buenas estimaciones del tiempo y esfuerzo necesarios

Las estimaciones de tiempo y esfuerzo de desarrollo se basan en las estimaciones de la magnitud del software que va a ser desarrollado:

- toman en cuenta factores tales como el tipo de software que se va a desarrollar y las características del equipo que va a hacer el desarrollo

Para los proyectos que siguen el proceso de cascada podemos usar el modelo de estimación COCOMO II

Para los proyectos que siguen procesos ágiles hay que usar estimaciones y planificación ágiles

En proyectos cascada, podemos usar la siguiente distribución de esfuerzo y tiempo por fase

Fase	Esfuerzo %	Tiempo %
Planes y requisitos	7 (2–15)	16–24 (2–30)
Diseño del producto	17	24–28
Programación	64–52	56–40
Diseño detallado	27–23	
Código y tests unitarios	37–29	
Integración y pruebas	19–31	20–32

adicional al 100% que representan las otras tres fases (propiamente de desarrollo)

X–Y: El % cambia de X% a Y% a medida que la magnitud del proyecto crece de 2 mil líneas de código (2 KSLOC) a 512 mil líneas de código (512 KSLOC)

En proyectos ágiles—en que se valora más responder al cambio—la planificación sigue siendo vital

Le da al equipo una visión clara de los objetivos del proyecto:

- apoyan la naturaleza colaborativa de “ágil,” ya que todos están en la misma página

Bien hecha, los planes resultantes no se vuelven obsoletos:

- definen el trabajo
- ayudan al equipo a tomar decisiones basadas en hechos

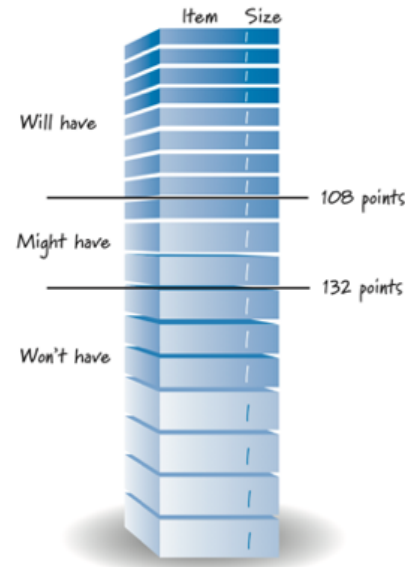
Se basa en Sprints y relatos de usuario:

- ... pero esto no significa que uno pueda ignorar el *big picture*

El Product Owner define el Backlog del Producto (entendiendo que Scrum es sólo uno de los enfoques ágiles)

Lista priorizada de requisitos—PBIs:

- todas las nuevas funcionalidades que hay que desarrollar, mejoras que hay que introducir, y correcciones de errores que hay que hacer al producto
- es *la* fuente del trabajo que hay que hacer
... pero no se espera que esté completamente definida desde un comienzo
- es dinámica y evoluciona—comienza como una lista de capacidades requeridas de alto nivel, y se vuelve más detallada a medida que el producto es construido, se obtiene *feedback*, y la gente identifica lo que se necesita para ser competitivo y útil



Cada PBI tiene un título, una descripción, una indicación de su valor potencial y como éste será validado, la cantidad probable de trabajo, y cualquier riesgo y dependencia asociado

El orden de los PBIs en el backlog indica el orden en que serán llevados a cabo: mientras más cerca está un PBI del tope de la lista, más importante es que sea implementado pronto, y más detallado tiene que estar

El Product Owner establece una *hoja de ruta* aproximada para el producto—permite visualizar las capacidades requeridas en momentos clave en el tiempo a lo largo de un horizonte de planificación de mediano a largo plazo:

en cuál *release* van a incluirse cuáles funcionalidades

el primer *release* normalmente corresponde al *producto viable mínimo* (MVP)

cada *release* subsiguiente agrega capacidades incrementalmente (hacia el producto completo que se ha visualizado)

para cada *release* hay que definir qué se va a medir —y cuál se espera que sea el resultado— una vez que el producto esté entregado

	Q3—año 1	Q4—año 1	Q1—año 2
Mercado	lanzamiento inicial	mejores resultados más plataformas	usuarios sofisticados
Funcionalidades	aprendizaje básico filtros básicos	aprendizaje mejorado consultas complejas	definir fuentes <i>learn by example</i>
Arquitectura	100,000 usuarios concu- rrentes vía Web	iOS Android	interfaz para servicios Web
Eventos	exposición en <i>social media</i>	conferencia de usuarios (revisr todo)	
<i>Releases</i>	1.0	2.0	3.0

El producto viable mínimo (MVP) nos permite emplear un enfoque basado en retroalimentación

Hay que identificar el trabajo crítico que tiene que hacer el usuario tipo de la aplicación

... y de ahí derivar la funcionalidad mínima que tiene que tener la aplicación para permitir que ese usuario tipo pueda hacer su trabajo:

- incluso, hay que considerar si cualquier funcionalidad se puede sacar de la aplicación y aun así lograr que ésta cumpla su propósito

... p.ej., en el método MoSCoW, cada ítem del Backlog se evalúa como ya sea *Must have*, *Should have*, *Could have* o *Won't have*, y el MVP consiste en los ítems *Must have*

Finalmente, hay que definir cómo se va a validar si el MVP es exitoso

El Dueño del Producto debe priorizar los ítemes del Backlog del Producto ...

... y así identificar los mejores candidatos para el próximo Sprint

Criterios:

- Valor para el negocio
- Deuda técnica
- Reducción temprana de riesgos
- Permitir hacer un trabajo que depende de otros trabajos
- Probar hipótesis
- Limpiar el Backlog de los ítemes con baja prioridad

Valor para el negocio

Significa diferentes cosas para diferentes organizaciones: p.ej., una empresa y una agencia de gobierno

Cada vez más, se toma en cuenta el impacto sobre las personas y el planeta, la calidad de servicio y la experiencia del cliente, además de la ganancia

El Dueño del Producto, el equipo de descubrimiento y otros *stake-holders* deberían acordar una definición del significado de *valor para el negocio*

... y las funcionalidades que más contribuyan a esos factores deberían ser priorizadas por sobre otras funcionalidades

Ejemplos de factores a considerar:

- aumentar ingresos
- proteger ingresos
- reducir costos
- mejorar la experiencia del cliente
- cumplir regulaciones y obligaciones sociales
- lograr una estrategia de mercado
- reclutar y mantener a la mejor gente, prepararlos desarrollarlos y apoyarlos

Todo equipo acumula deuda técnica

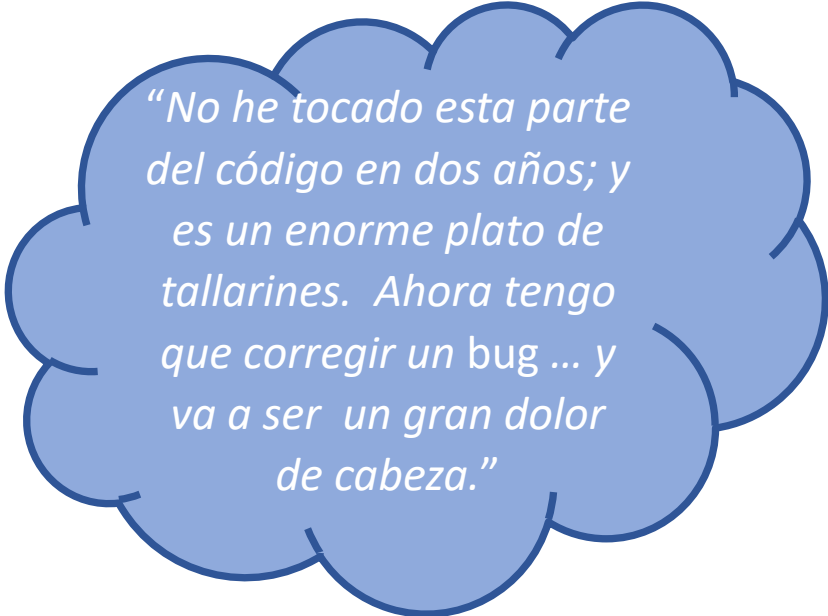
Los pequeños problemas en el código se van sumando a lo largo del tiempo

Mientras más tiempo persisten los problemas de diseño y programación, más se “componen” los problemas

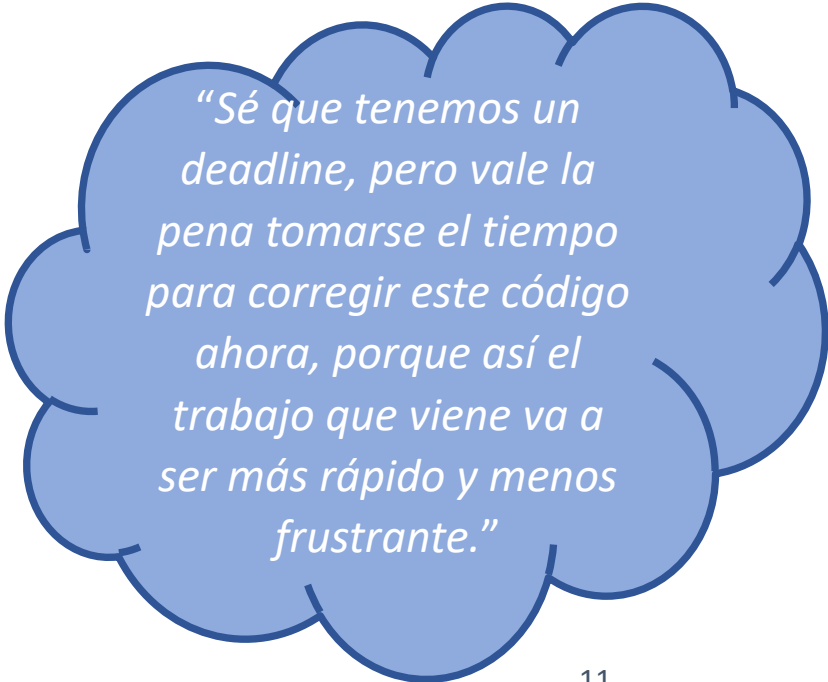
→ **deuda técnica:** código complejo con el cual es difícil trabajar

Los equipos XP pagan la deuda técnica en **cada ciclo semanal, agregando tiempo** al ciclo, para que los problemas no se empiecen a acumular:

- ¿la forma más eficaz? **refactorizar sin misericordia**



“No he tocado esta parte del código en dos años; y es un enorme plato de tallarines. Ahora tengo que corregir un bug ... y va a ser un gran dolor de cabeza.”



“Sé que tenemos un deadline, pero vale la pena tomarse el tiempo para corregir este código ahora, porque así el trabajo que viene va a ser más rápido y menos frustrante.”

Planificación del *release* a partir de la visión y hoja de ruta del producto

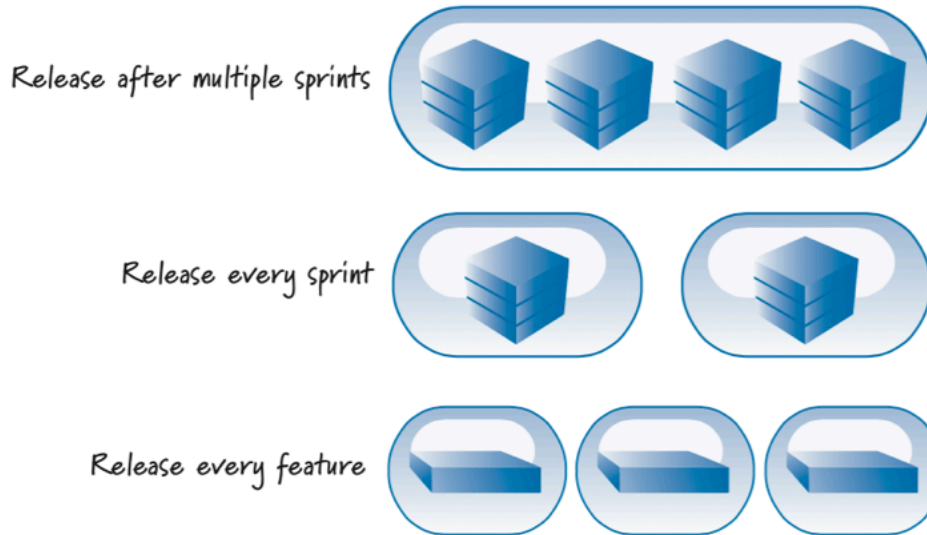
La visión del producto define el “norte” para el desarrollo o mejoramiento de un producto

La hoja de ruta del producto es una vista de alto nivel de cómo las funcionalidades del producto van a ir siendo entregadas (*released*) a lo largo del tiempo

Las organizaciones que usan Scrum deberían ser capaces de entregar incrementos del producto al final de cada Sprint:

- ... pero el Dueño del Producto puede preferir esperar hasta que haya suficientes funcionalidades que se puedan entregar juntas
- ... en ese caso, se suele combinar varios Sprints en un **release**

Posibles cadencias para el *release*



El release es planificado con respecto a una única restricción:

- ... ya sea para determinar cuánto alcance se puede entregar en un número fijo de Sprints (fecha fija)
... o cuántos Sprints van a ser necesarios para entregar un alcance determinado (alcance fijo)

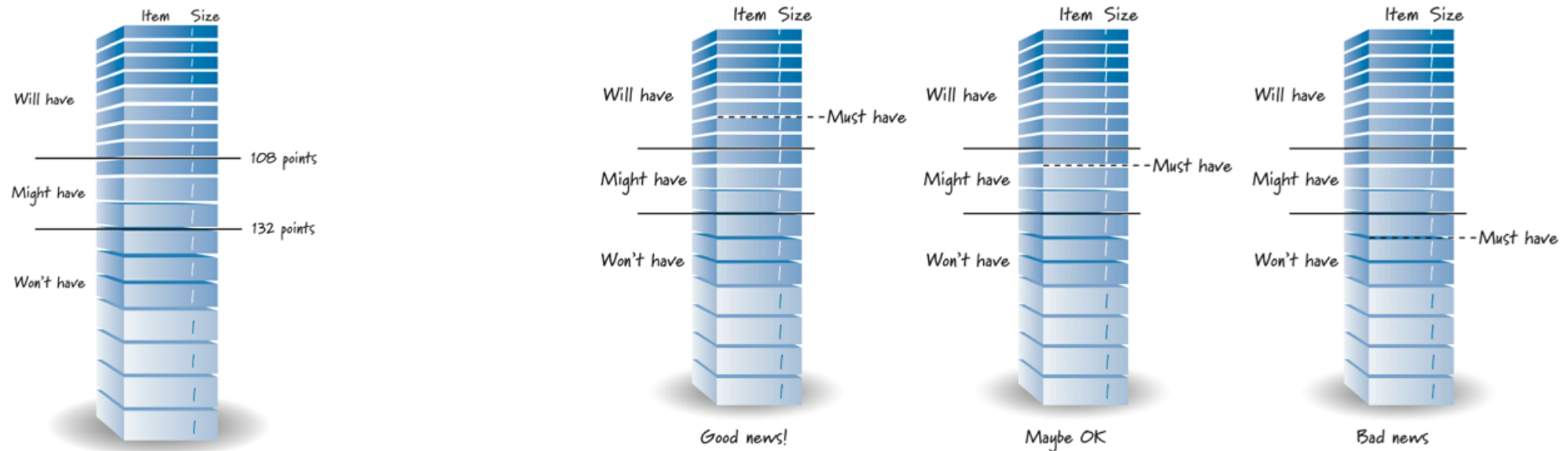
El proceso de planificación del *release*

- 1) Tomar los primeros ítems del Backlog del Producto
- 2) Para cada uno:
 - considerar qué se necesita para implementarlo
 - asignarle un tamaño
 - si el tamaño es más grande que *medium*, entonces dividirlo
- 3) Repetir el paso 2) hasta que el Sprint esté lleno
- 4) Pegar el primer ítem del Sprint en la pared

Si hay un número fijo de Sprints, entonces repetir los pasos 1) a 4) hasta que todos los Sprints del release tengan ítems

Si hay un alcance fijo, entonces repetir los pasos 1) a 4) hasta que todos los ítems del alcance hayan sido pegados en la pared

Planificación si el número de Sprints está fijo



Es conveniente que el MVP sea planificado para algo así como el 70% del tiempo del Sprint:

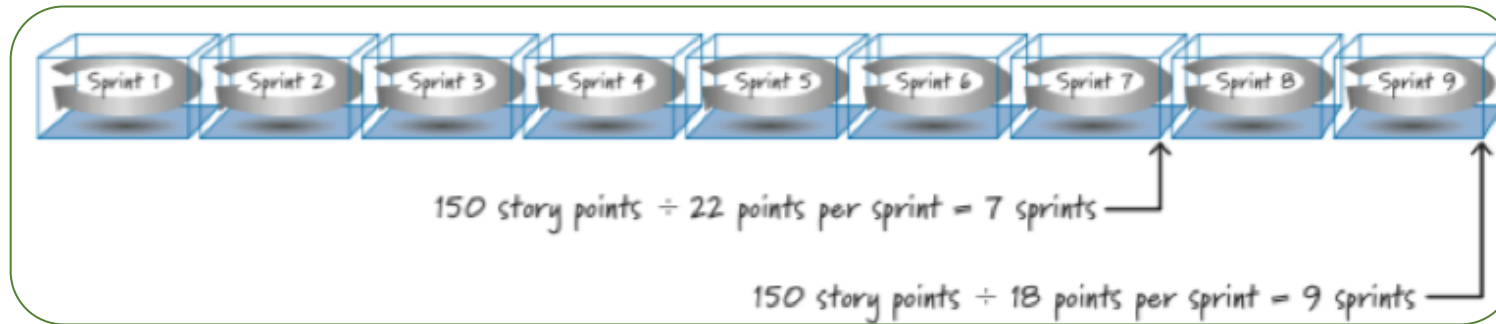
- pueden surgir otros “Must have”

Es conveniente usar puntos de relato y “ritmos” observados en la práctica

Planificación si el número de Sprints está fijo

Step	Description	Comments
1	Determine how many sprints are in this release.	If all sprint lengths are equal, this is simple calendar math because you know when the first sprint will start and you know the delivery date.
2	Groom the product backlog to a sufficient depth by creating, estimating the size of, and prioritizing product backlog items.	Because we are trying to determine which PBIs we can get by a fixed date, we need enough of them to plan out to that date.
3	Measure or estimate the team's velocity as a range.	Determine an average faster and an average slower velocity for the team (see Chapter 7).
4	Multiply the slower velocity by the number of sprints. Count down that number of points into the product backlog and draw a line.	This is the "will-have" line.
5	Multiply the faster velocity by the number of sprints. Count down that number of points into the product backlog and draw a second line.	This is the "might-have" line.

Planificación si el alcance está fijo



El conjunto de funcionalidades es poco transable, pero estamos dispuestos a postergar la entrega

Todas las funcionalidades en la lista *Must have*

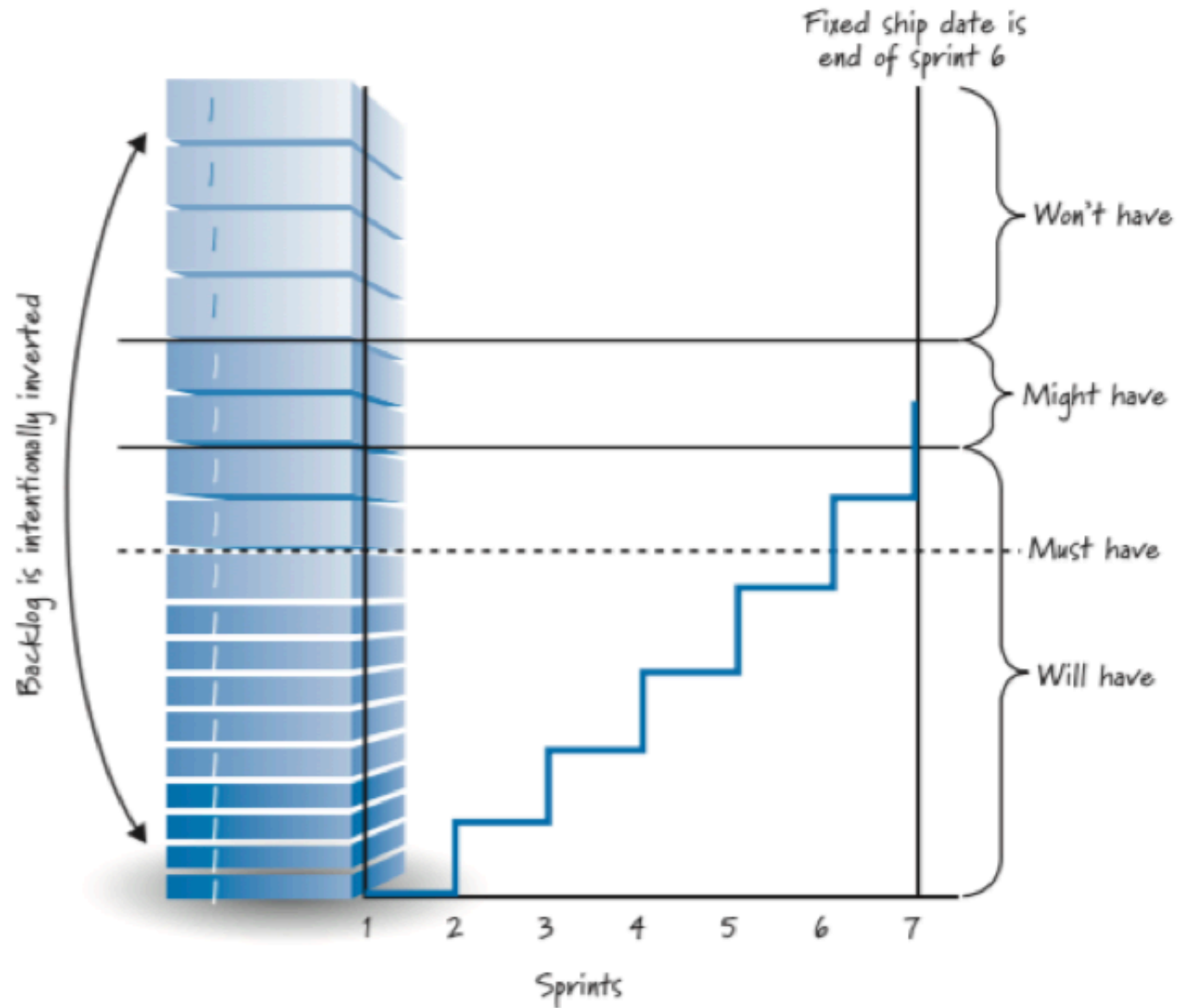
El número de Sprints puede ser variable

Puede cambiarse por una serie de releases con fecha fija

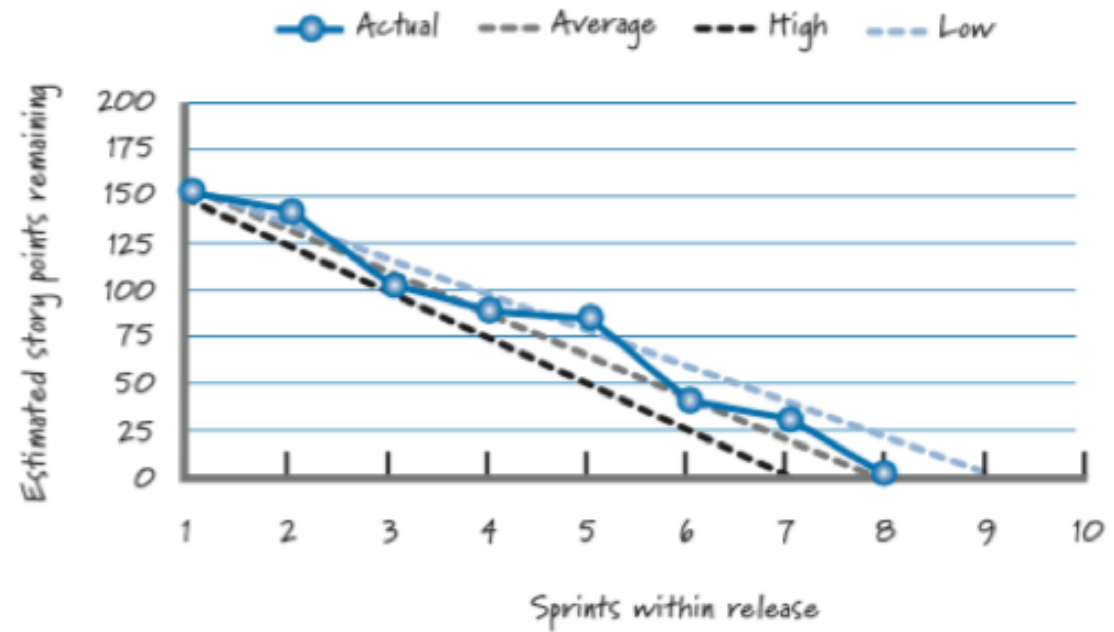
Planificación si el alcance está fijo

Step	Description	Comments
1	Groom the product backlog to include at least the PBIs we would like in this release by creating, estimating the size of, and prioritizing PBIs.	Because this is a fixed-scope release, we need to know which PBIs are in the fixed scope.
2	Determine the total size of the PBIs to be delivered in the release.	If we have a product backlog of estimated items, we simply sum the size estimates of all of the items we want in the release.
3	Measure or estimate the team's velocity as a range.	Determine an average faster and an average slower velocity for the team.
4	Divide the total size of the PBIs by the faster velocity and round up the answer to the next integer.	This will tell us the lowest number of sprints required to deliver the features.
5	Divide the total size of the PBIs by the slower velocity and round up the answer to the next integer.	This will tell us the highest number of sprints required to deliver the features.

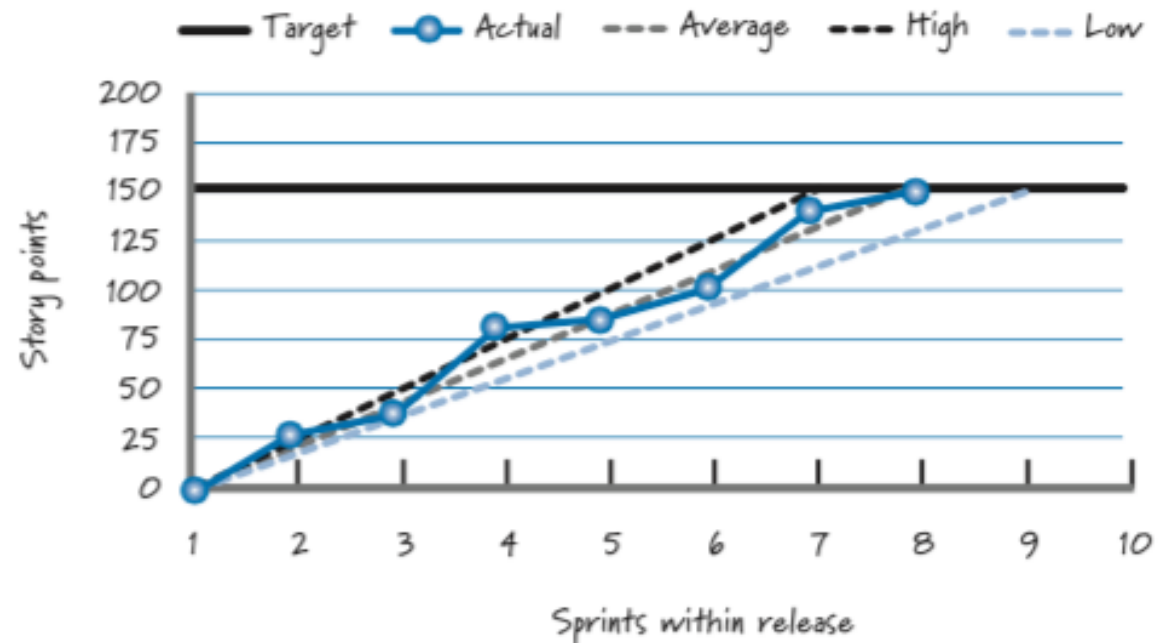
Comunicación
del progreso si
el número de
Sprints está fijo



Comunicación del progreso si el alcance está fijo



Release
burndown



Release
burnup

Planificación del Sprint

Participa todo el equipo

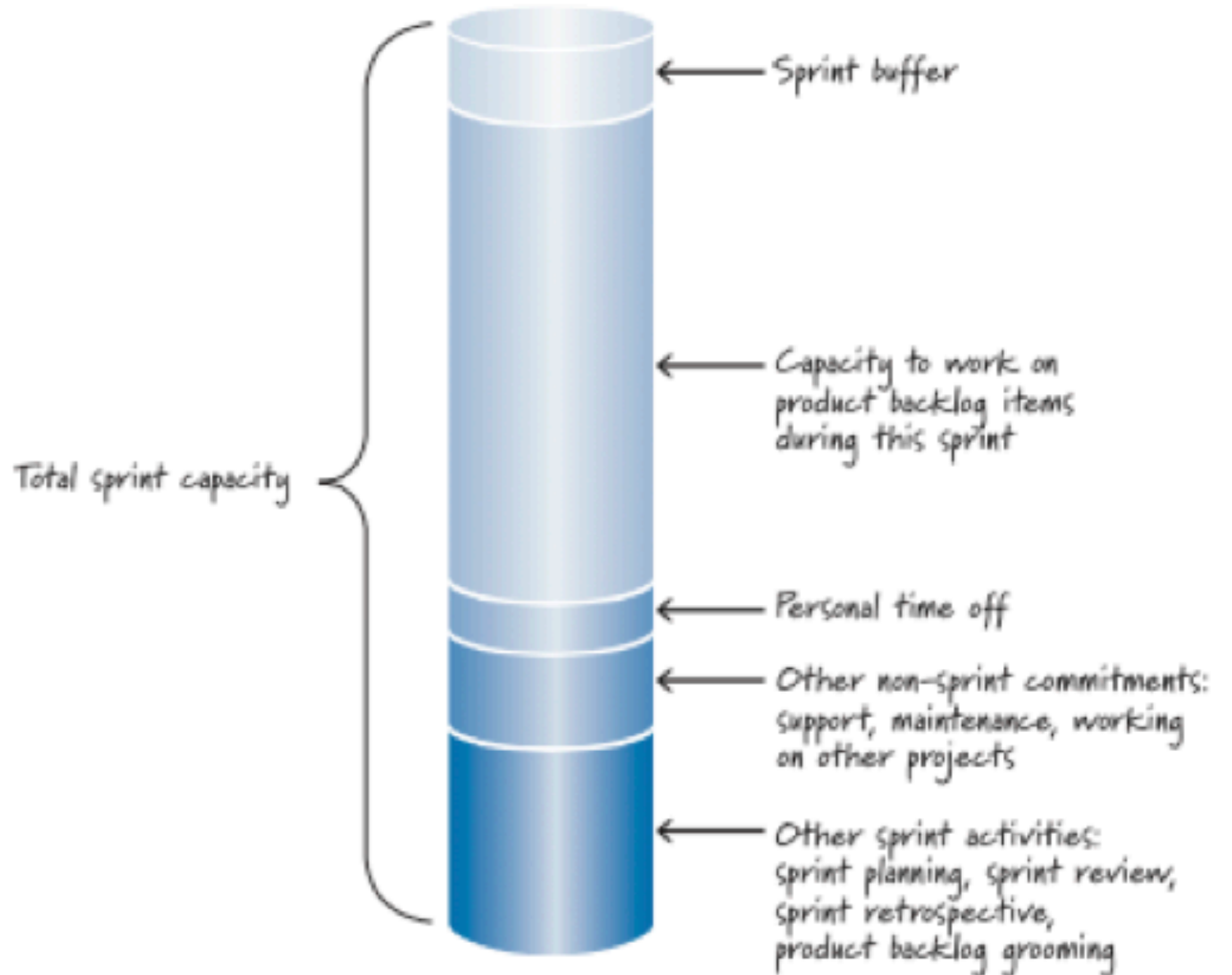
El Dueño del Producto presenta PBIs priorizados

Otros inputs necesarios incluyen velocidad de desarrollo, capacidad del equipo, restricciones

El Scrum Master debe ayudar a que el equipo incluya conjunto razonable de ítems

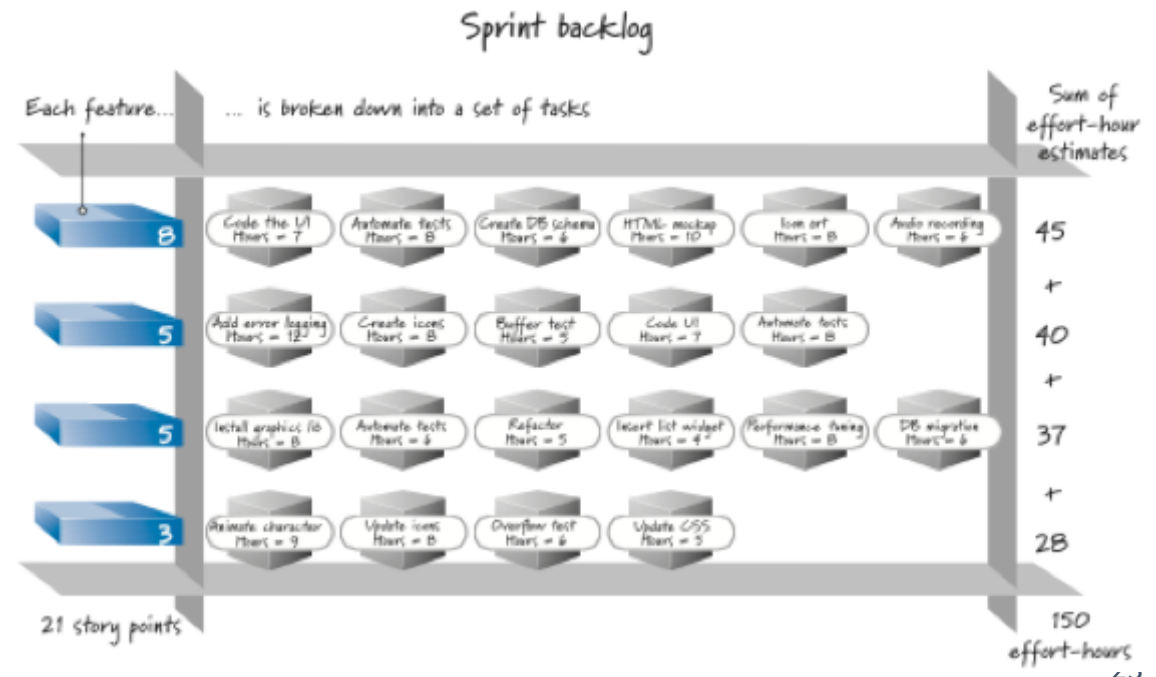


“Capacidad” de un Sprint

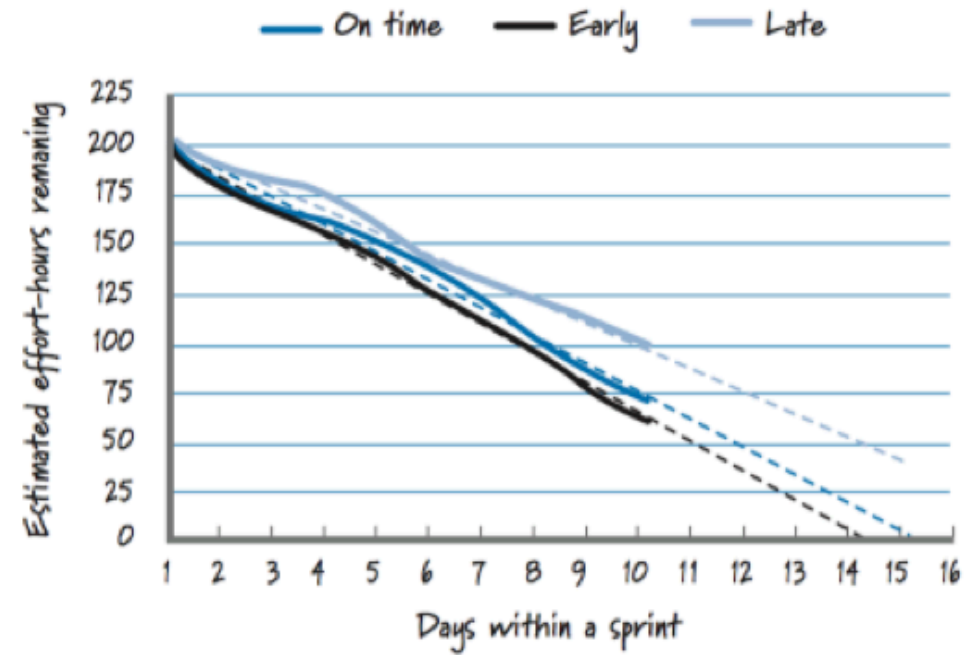
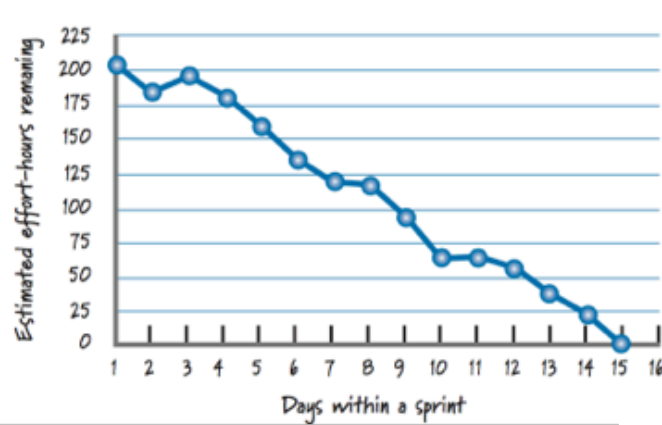


Ejemplo de planificación de un Sprint

Person	Days Available (Less Personal Time)	Days for Other Scrum Activities	Hours per Day	Available Effort-Hours
Jorge	10	2	4-7	32-56
Betty	8	2	5-6	30-36
Rajesh	8	2	4-6	24-36
Simon	9	2	2-3	14-21
Heidi	10	2	5-6	40-48
Total				140-197



Diagramas *burndown* para un Sprint



Recordemos un Sprint de principio a fin

Todo proyecto Scrum sigue el mismo patrón de conducta

... definido por una serie de **eventos** ...

... que siempre pasan en el mismo orden:

el *Sprint* (**E**)

la sesión de *Planificación del Sprint* (**A**)

la construcción del *backlog del Sprint* (**B**)

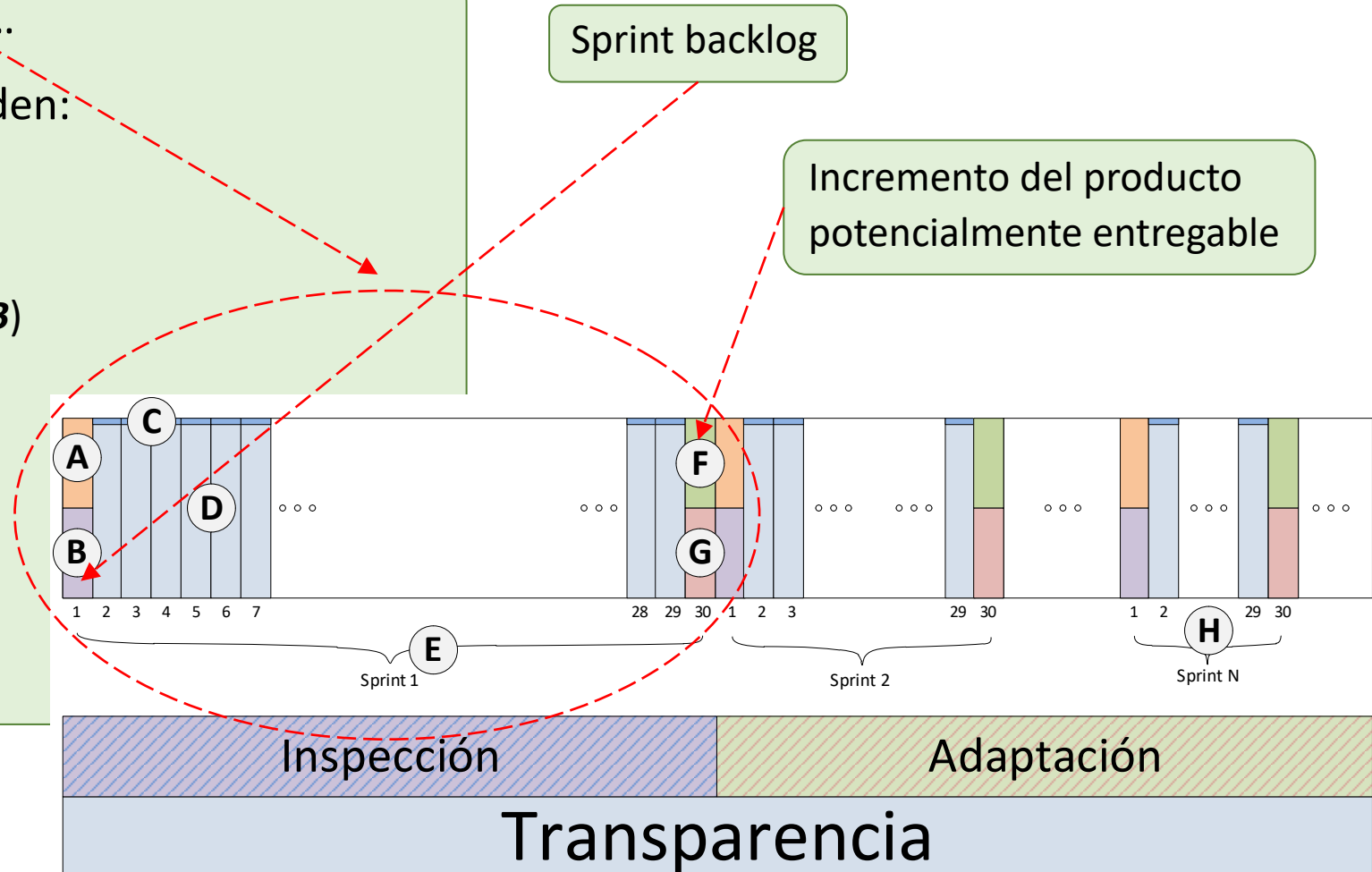
el *Scrum Diario* (**C**)

la *Revisión del Sprint* (**F**)

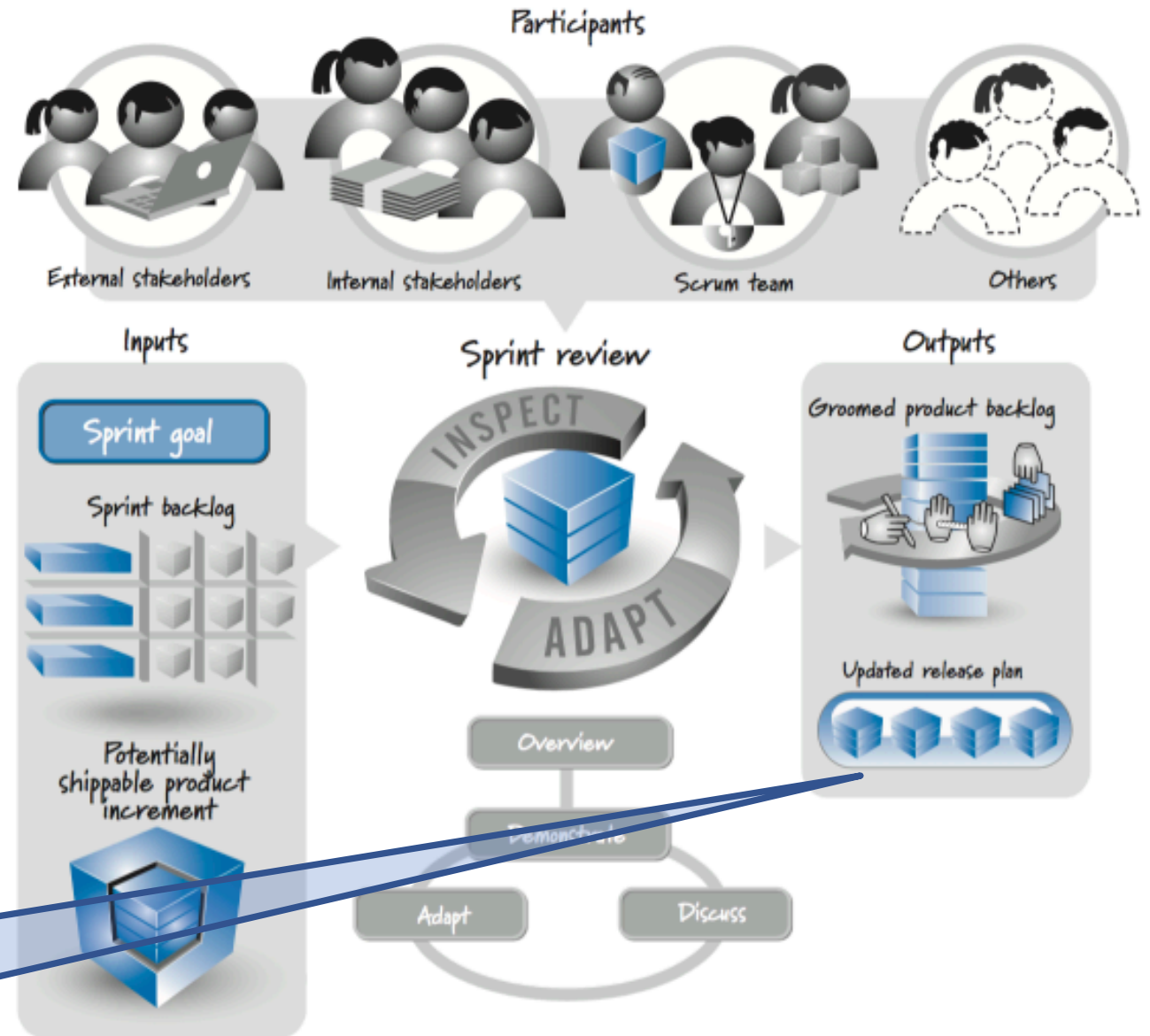
la *Retrospectiva del Sprint* (**G**)

resto del día laboral (**D**)

próximos Sprints (**H**)



Outputs de la Revisión del Sprint (Sprint Review)



<https://kanbanize.com/agile/project-management/planning>

<https://monday.com/blog/project-management/agile-planning/>