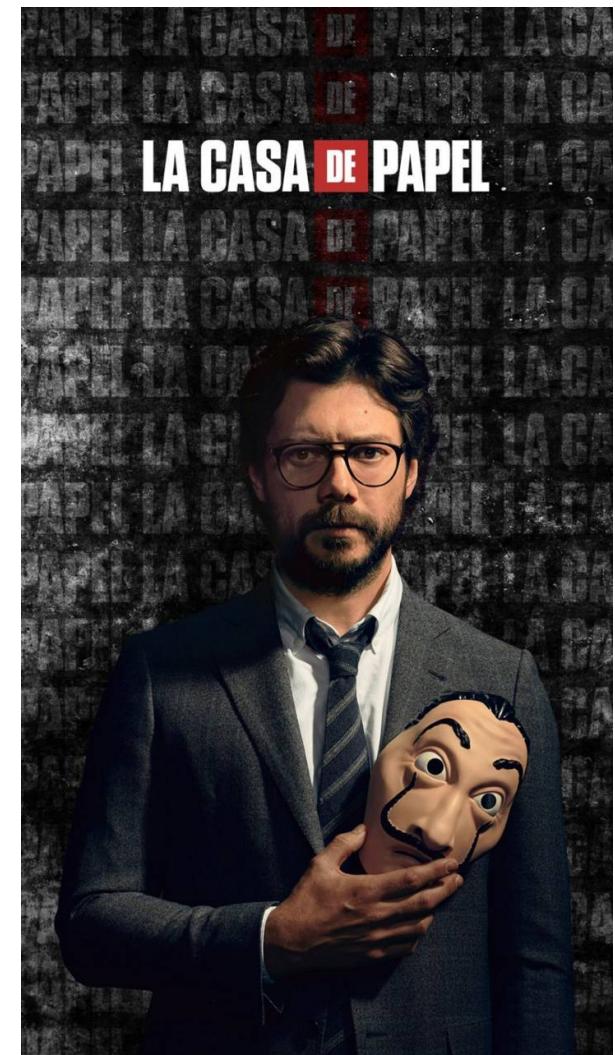


# Planificación del Proyecto

- ▶ Los proyectos complejos requieren una planificación cuidadosa
- ▶ Incluso los proyectos relativamente simples requieren ser planificados
- ▶ El desarrollo de un proyecto de software sin ninguna planificación suele terminar en un desastre



# Planeación en el Contexto Agil



- ▶ La analogía del descenso libre en ski
  - ▶ ¿ se puede planificar el decenso completo ?
  - ▶ ¿ qué se puede planificar ?

Cuando el terreno y el mapa no coinciden  
hay que creerle al terreno

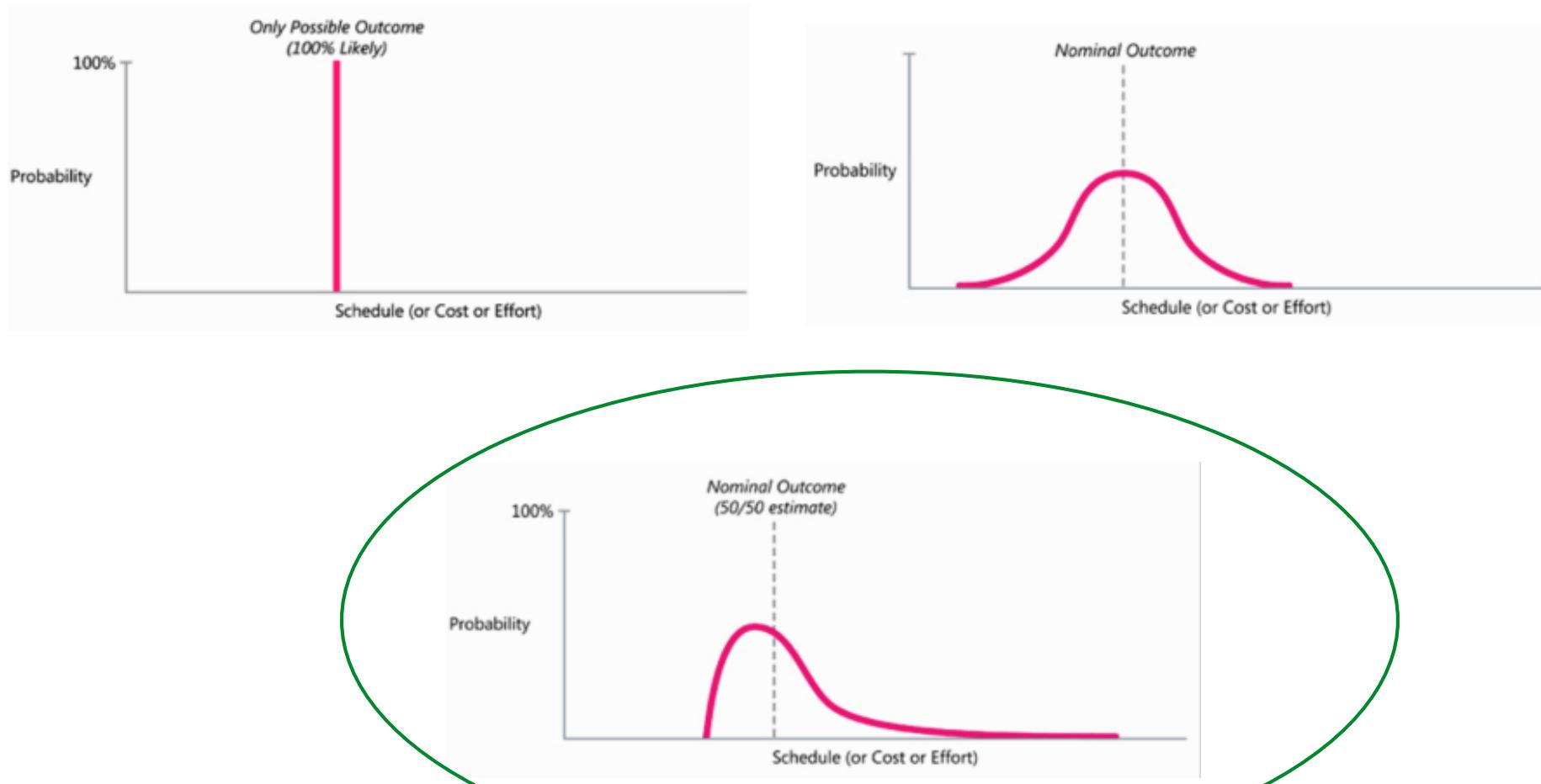


# Planificar requiere hacer estimaciones

- ▶ La problemática de hacer estimaciones es un tema bastante amplio que veremos más adelante con más detalle
- ▶ Lo básico
  - ▶ **Qué necesitamos estimar**
    - ▶ esfuerzo de desarrollo - semanas ingeniero
    - ▶ tiempo de desarrollo - semanas
  - ▶ **Que insumos tenemos**
    - ▶ tamaño del proyecto (lineas de código, relatos de usuario)
    - ▶ productividad del equipo (historia)

***Mientras mas temprana la estimación mayor es el error***

# Estimaciones son afirmaciones probabilísticas

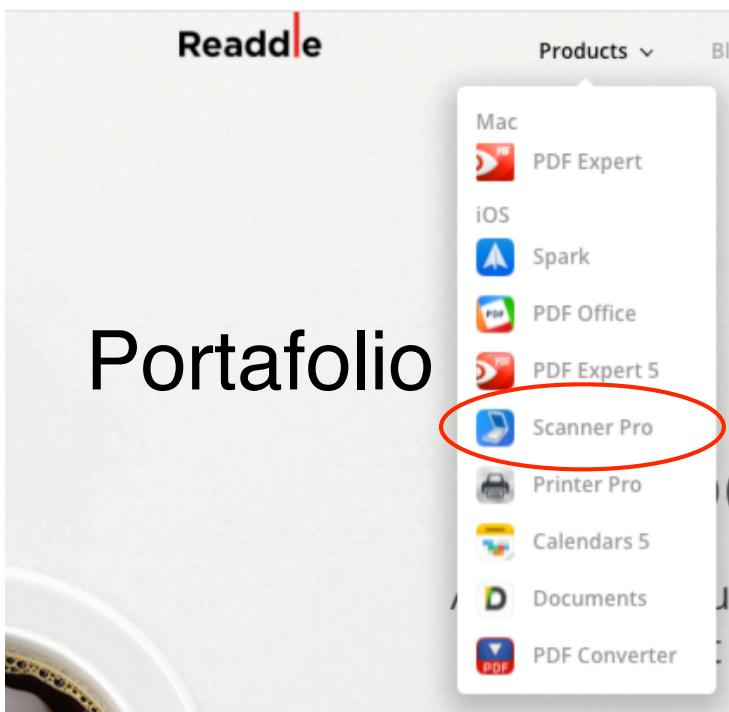


# Varios niveles de planeación

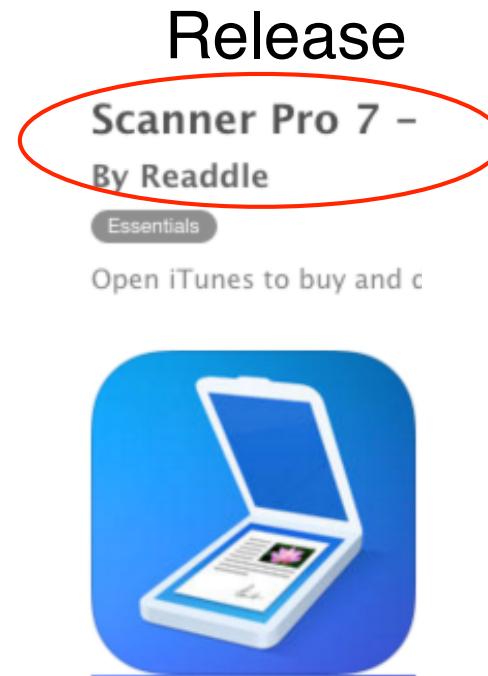
- ▶ **Iteración** - debe producir un producto potencialmente entregable
- ▶ **Release** - produce el entregable al fin del proceso iterativo
- ▶ **Producto** - plan de releases a lo largo de la vida del producto
- ▶ **Portafolio** - desarrollo de los diversos productos de la compañía

# Ejemplo

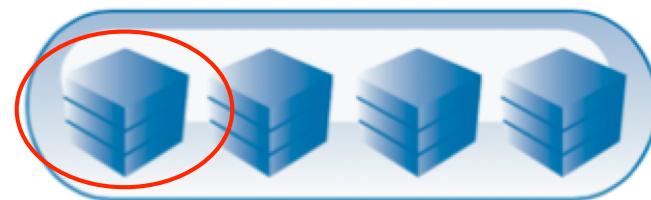
Portafolio



Producto



Iteración



# Aplicación a su Proyecto

- ▶ Planeación a nivel de Release (entrega final)
- ▶ Planeación a nivel de Iteración (cada sprint)

# Planeación a nivel de Release

- ▶ Salida de un sprint es potencialmente entregable
- ▶ Normalmente un release incluye un cierto número de sprints
- ▶ Organización decide qué y cuando entregar (qué es lo que va en el release)

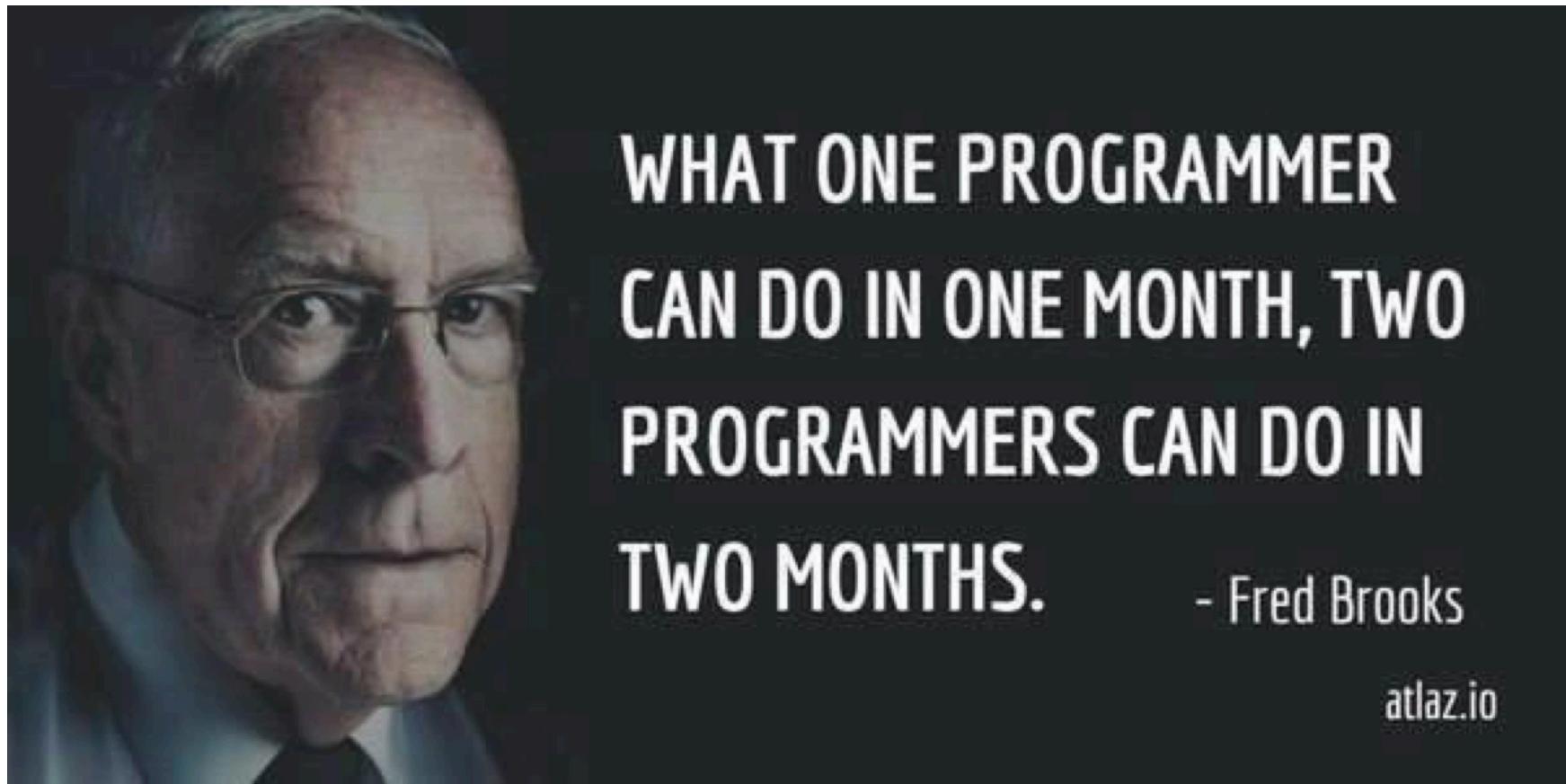
# Planeación se hace sujeta a restricciones

- ▶ Tiempo - el producto tiene que estar lista en una fecha determinada
- ▶ Alcance - el producto tiene que incluir un set de prestaciones determinado
- ▶ Presupuesto - el desarrollo no puede superar los \$10.000.000

# La necesidad de ajustar

- ▶ Una estimación temprana por lo general tiende a subestimar el trabajo a realizar ...
- ▶ ... lo que hace difícil entregar la totalidad de las prestaciones en el tiempo planificado inicialmente
- ▶ En otras áreas de la ingeniería la solución para un proyecto atrasado es agregar mas mano de obra
- ▶ En proyectos de software eso NO FUNCIONA

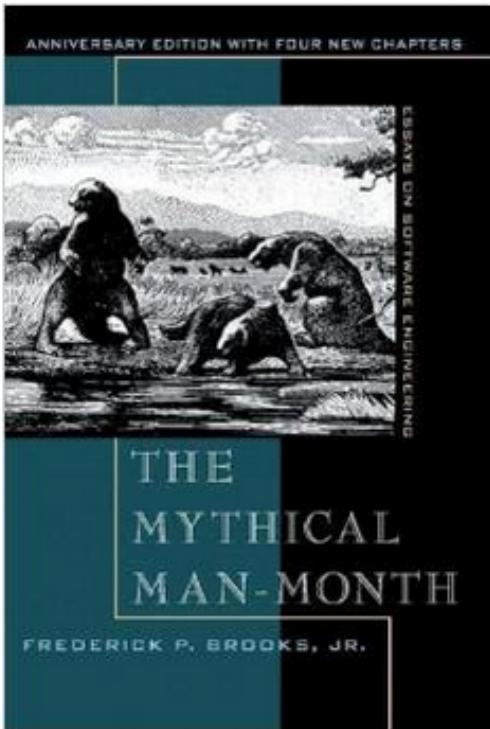
Dr. Fred Brooks, premio Turing 1999, autor del libro mas citado en Ing de Software



“Adding manpower to a late software project makes it later” (Brooks 1995)

“Nine women don’t make a baby in a month” (Brooks 1995)

# El libro más citado de la Ing de Software



**The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition (2nd Edition)**



**Fué mi profe !!!**

Mucha gente está llamando a mi libro "La biblia de la Ingeniería de Software". Debe ser porque

- Todo el mundo lo cita
- Algunos lo leen
- Muy poca gente sigue los consejos que allí aparecen

*Fred Brooks*

# ¿Qué hacemos entonces con un proyecto que está atrasado ?

- ▶ Aperramos no mas !
  - ▶ el equipo trabaja incansablemente día y noche tratando de agarrar el tren
  - ▶ producto suele terminar con muchos bugs
  - ▶ igual no se logra completar a tiempo
- ▶ Ajustamos en el camino
  - ▶ mientras antes mejor

# Tipo de Ajuste

- ▶ Ajuste de tiempo - se renegocia la fecha de entrega
  - ▶ no podremos entregar el 30 de Septiembre, necesitamos un mes más
- ▶ Ajuste por alcance - se renegocian las funcionalidades que van en el release
  - ▶ entregaremos el 30 de Septiembre como prometimos pero estas dos funcionalidades quedará para el siguiente release

# Combinaciones Posibles

Project Type	Scope	Date	Budget
Fixed everything (not recommended)	Fixed	Fixed	Fixed
Fixed scope and date (not recommended)	Fixed	Fixed	Flexible
Fixed scope	Fixed	Flexible	Fixed (not really)
Fixed date	Flexible	Fixed	Fixed

“Adding manpower to a late software project makes it later” (Brooks 1995)

“Nine women don’t make a baby in a month” (Brooks 1995)

# El tipo de ajuste que NO se debe hacer

- ▶ es el mas comúnmente usado: el ajuste por calidad
  - ▶ acortar trabajo de requisitos, diseño o **testing**
  - ▶ eliminar revisión de código
  - ▶ eliminar programación en pares
- ▶ es una pésima práctica
  - ▶ puede generar una tremenda **deuda técnica (technical backlog)**
  - ▶ puede quedar muy bajo las expectativas del usuario

# Deuda Técnica

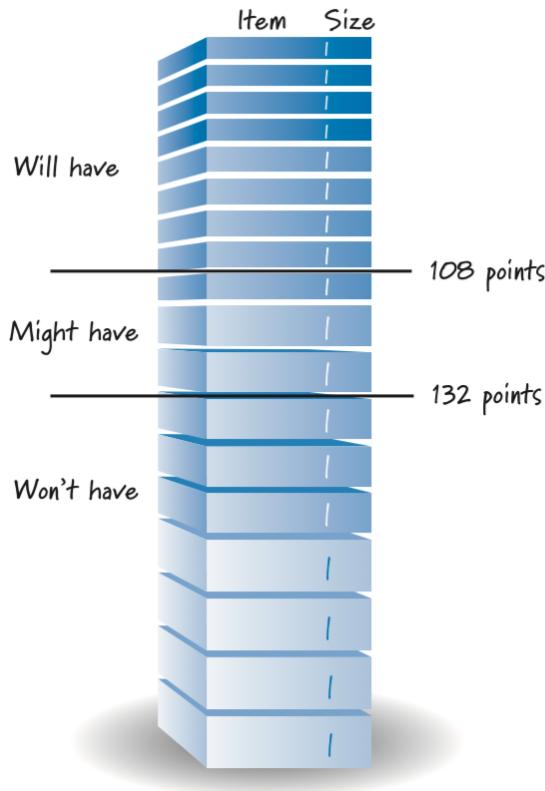
- ▶ La deuda técnica puede ser definida como las consecuencias de largo plazo de malas decisiones de diseño
- ▶ En este sentido es como cualquier otra deuda, hay que entender bien que se está incurriendo en ella y tener claro que se deberá pagar
- ▶ A veces vale la pena, por ejemplo para agilizar cosas urgentes que se deben cumplir en un determinado sprint
- ▶ Debido a que esas malas decisiones serán mas costosas de reparar en el futuro la deuda puede cobrar "intereses"

# Planeación en base a fecha fija

- ▶ Determinar el número de sprints (dividir el tiempo disponible por el largo del sprint)
- ▶ Preparar, estimar el esfuerzo (puntos de relato) y priorizar el backlog del producto
- ▶ Determinar velocidades promedio de desarrollo del equipo (mejor caso y peor caso)
- ▶ Con esas y el número de sprints calcular el número de puntos que se lograrán en el peor caso y el mejor caso
- ▶ Ellos corresponden al "will have" y al "might have")

# Ejemplo

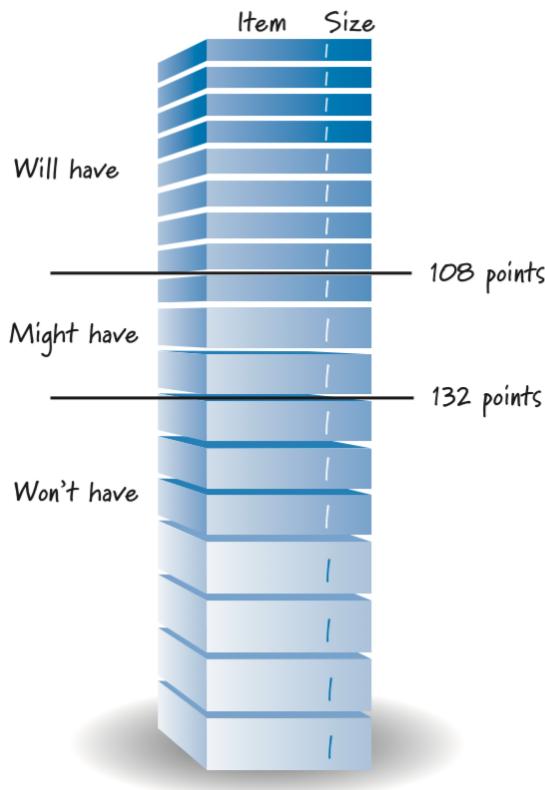
- ▶ Número de sprints: 4
- ▶ Velocidades: 27 - 38 sps/sprint
- ▶ worst case: 108
- ▶ best case: 132
- ▶ el primer corte está basado en el worst case (will have)
- ▶ el segundo corte está basado en el best case (might have)



# Minimum Releasable Features (MRF)

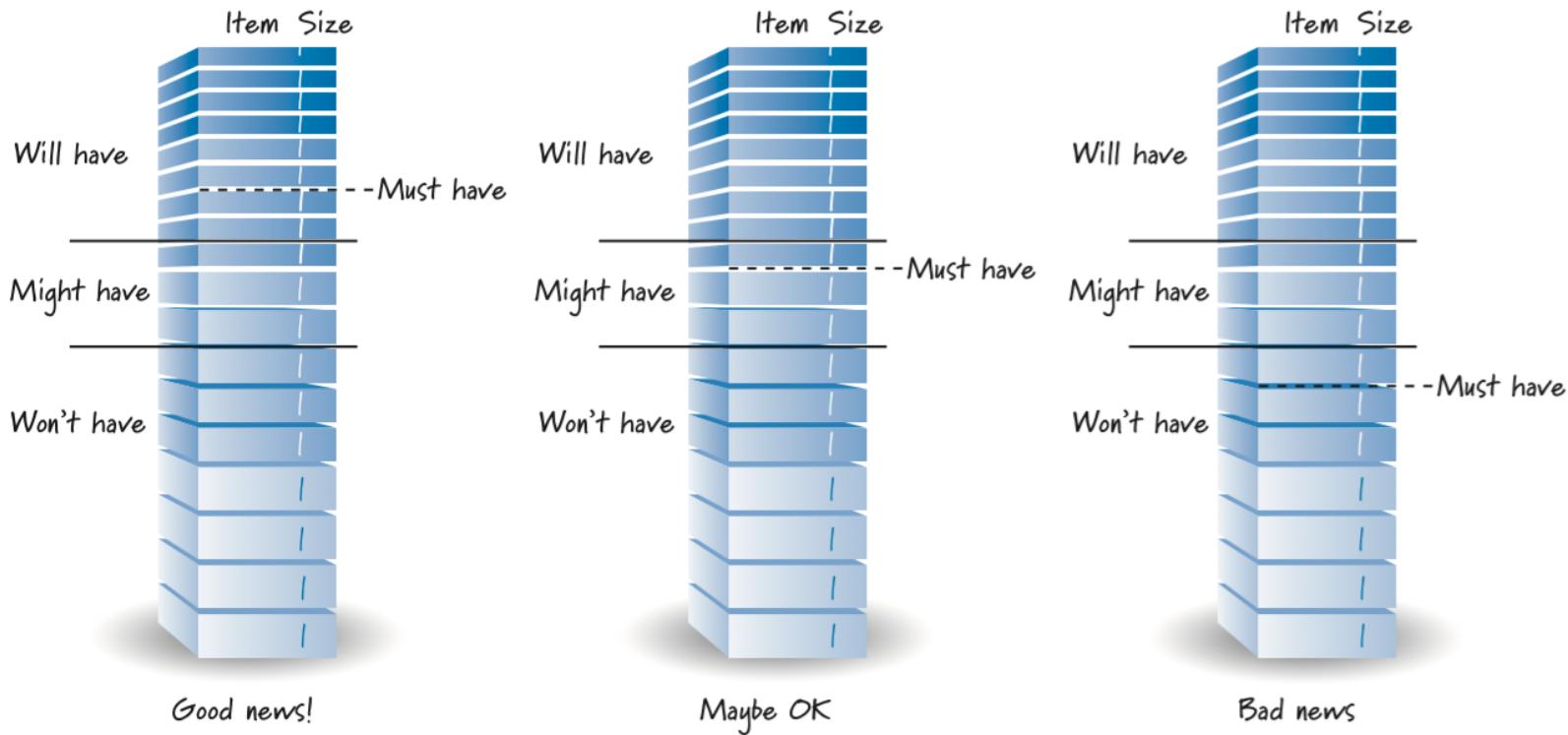
- ▶ set mínimo de “must have” features en el release
- ▶ responsabilidad del “product owner”
- ▶ una parte importante de planeación del release es reevaluar constantemente el MRF a medida que progresan los sprints

# Rango posible



- el PB está priorizado de arriba hacia abajo
- cada item ha sido evaluado en cuanto a esfuerzo (points)
- conocemos la productividad promedio (best case y worst case)
- el primer corte está basado en el worst case
- el segundo corte está basado en el best case
- es conveniente no dejar el MRF muy calzado con el tiempo (pueden surgir otros “must-have”)

# Ubicación del punto "must have"



# ¿Y qué demonios es un "story point" ?

- ▶ es la unidad de esfuerzo mas usada a nivel de release al usar desarrollo ágil
- ▶ es una forma de dimensionar en términos relativos el esfuerzo asociado a los distintos relatos
- ▶ escalas mas usadas
  - ▶ potencias de 2: 1, 2, 4, 8, ...
  - ▶ serie de fibbonacci : 1, 2, 3, 5, 8, ...

# Ojo: no confiar ciegamente en las escalas

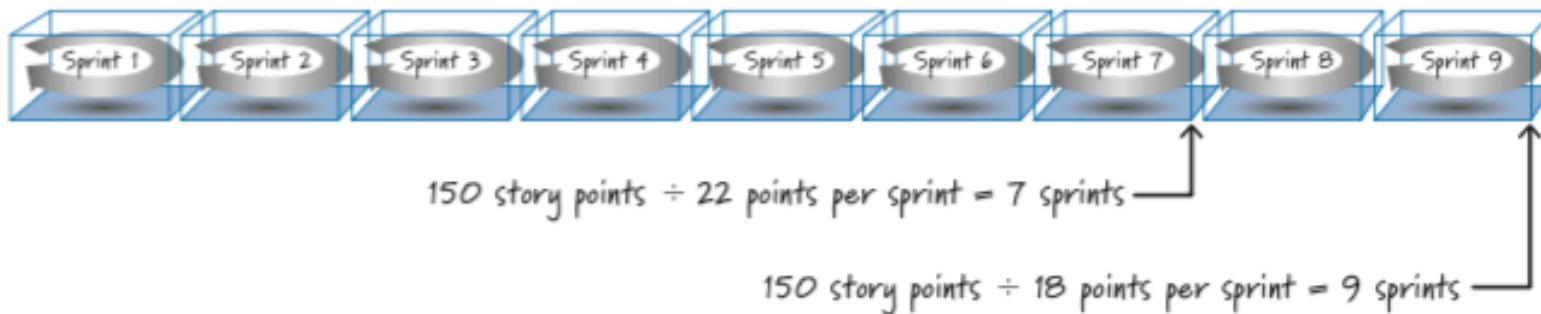
- ▶ Uso de una escala como por ejemplo 1, 2, 3, 5, 8 implica que un relato calificado con 5 debe involucrar esfuerzo 5 veces mayor que uno de 1
- ▶ Requiere disciplina al asignar los puntos al relato
- ▶ Partir con los mas simples (1) y luego pensar en forma relativa

# Planeación en base a alcance fijo

- ▶ Preparar, estimar el esfuerzo (puntos de relato) y priorizar el backlog del producto
- ▶ Seleccionar todos los PBIs que deben estar en el release (alcance fijo)
- ▶ Determinar velocidades promedio de desarrollo del equipo (mejor caso y peor caso)
- ▶ Determinar el número de sprints mínimo y máximo necesario (dividir el total de SPs por las velocidades mínima y máxima)
- ▶ Tenemos el rango de sprints necesarios y por lo tanto un tiempo de desarrollo necesario

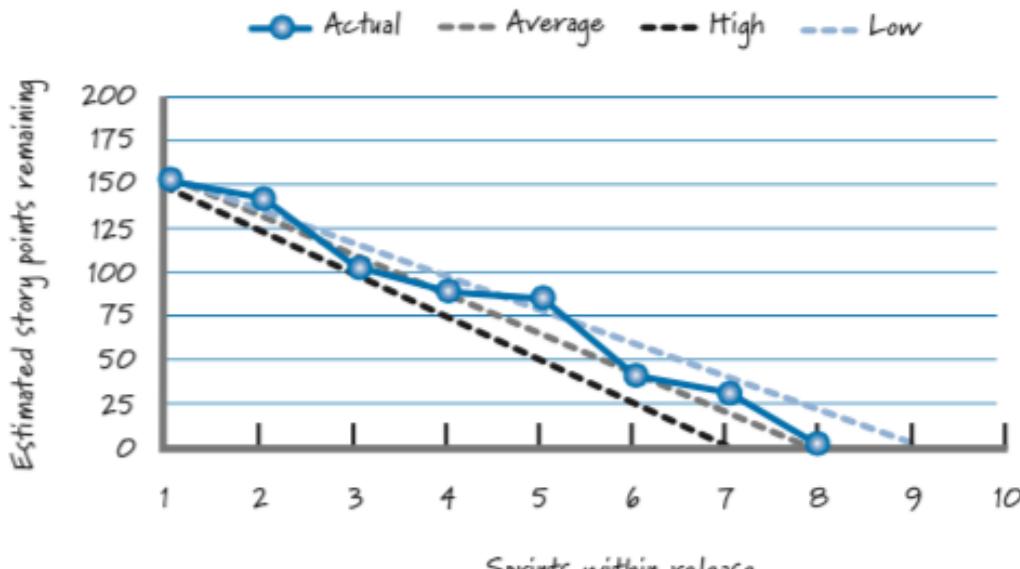
# Planeación en base a Alcance fijo

- ▶ set the features es poco transable pero estamos dispuestos a postergar la entrega
- ▶ todos los features en lista “must-have”
- ▶ número de sprints puede ser variable

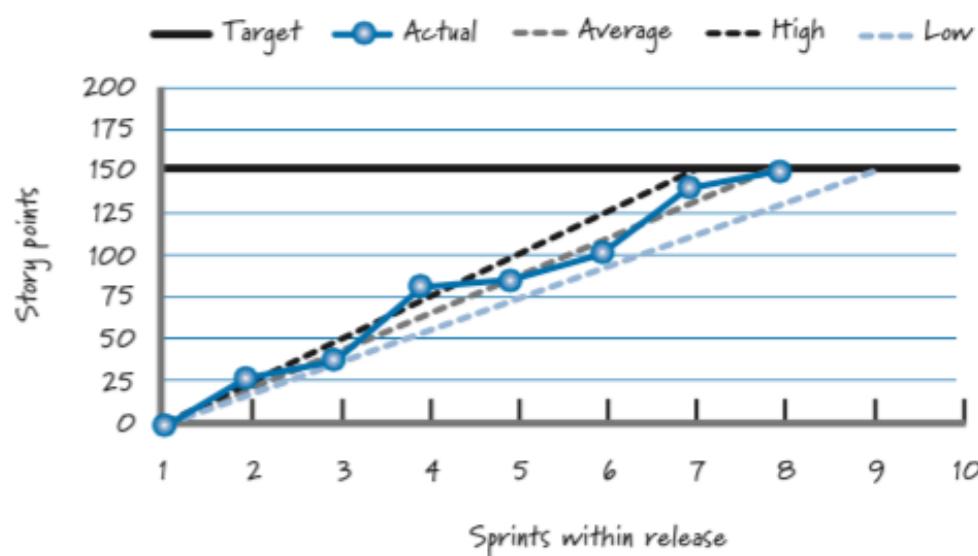


# Monitoreando el Progreso

## ▶ Alcance Fijo



Release burndown



Release burnup

# Por qué es necesario considerar *best case* y *worst case*

## 1. Estimación única

Feature	Estimated Days to Complete
Feature 1	1.5
Feature 2	1.5
Feature 3	2.0
Feature 4	0.5
Feature 5	0.5
Feature 6	0.25
Feature 7	2.0
Feature 8	1.0
Feature 9	0.75
Feature 10	1.25
<b>TOTAL</b>	<b>11.25</b>

Estimación de 11.25 días

# ¿ Exceso de Optimismo ?

2. El mismo equipo estima best case y worst case

Feature	Estimated Days to Complete	
	Best Case	Worst Case
Feature 1	1.25	2.0
Feature 2	1.5	2.5
Feature 3	2.0	3.0
Feature 4	0.75	2.0
Feature 5	0.5	1.25
Feature 6	0.25	0.5
Feature 7	1.5	2.5
Feature 8	1.0	1.5
Feature 9	0.5	1.0
Feature 10	1.25	2.0
<b>TOTAL</b>	<b>10.5<sup>1</sup></b>	<b>18.25</b>

Estimación de 11.25 días está  
mucho mas cerca del "best case"

Estimación más real sería de unos 14 días

# ¿ Como estimar las velocidades de desarrollo ?

- ▶ Examinando lo que ha pasado antes !
  - ▶ en la iteración anterior
  - ▶ en el release anterior
  - ▶ en el proyecto anterior (mismo equipo)
  - ▶ promedios de la compañía
  - ▶ promedios de la industria

# Por qué es necesario usar información histórica

- ▶ Permite contrarrestar tendencia natural al optimismo en un proyecto nuevo
  - ▶ esta vez no se nos irá el jefe del proyecto
  - ▶ esta vez tenemos mejor tecnología
  - ▶ esta vez no perderemos tiempo al comienzo
- ▶ Estimación basada en historia nos dice que ocurrirá mas o menos lo mismo

# Ejemplo: Proyecto de 180 story points

---

## **Data for Iteration 1**

27 story points delivered

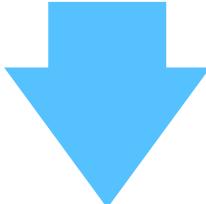
12 staff weeks expended

3 calendar weeks expended

## **Preliminary Calibration**

Effort = 27 story points ÷ 12 staff weeks = 2.25 story points/staff week

Schedule = 27 story points ÷ 3 calendar weeks = 9 story points/calender week



---

## **Data for Iteration 1**

### **Assumptions (from Preliminary Calibration)**

Effort = 2.25 story points/staff week

Schedule = 9 story points/calender week

Project size = 180 story points

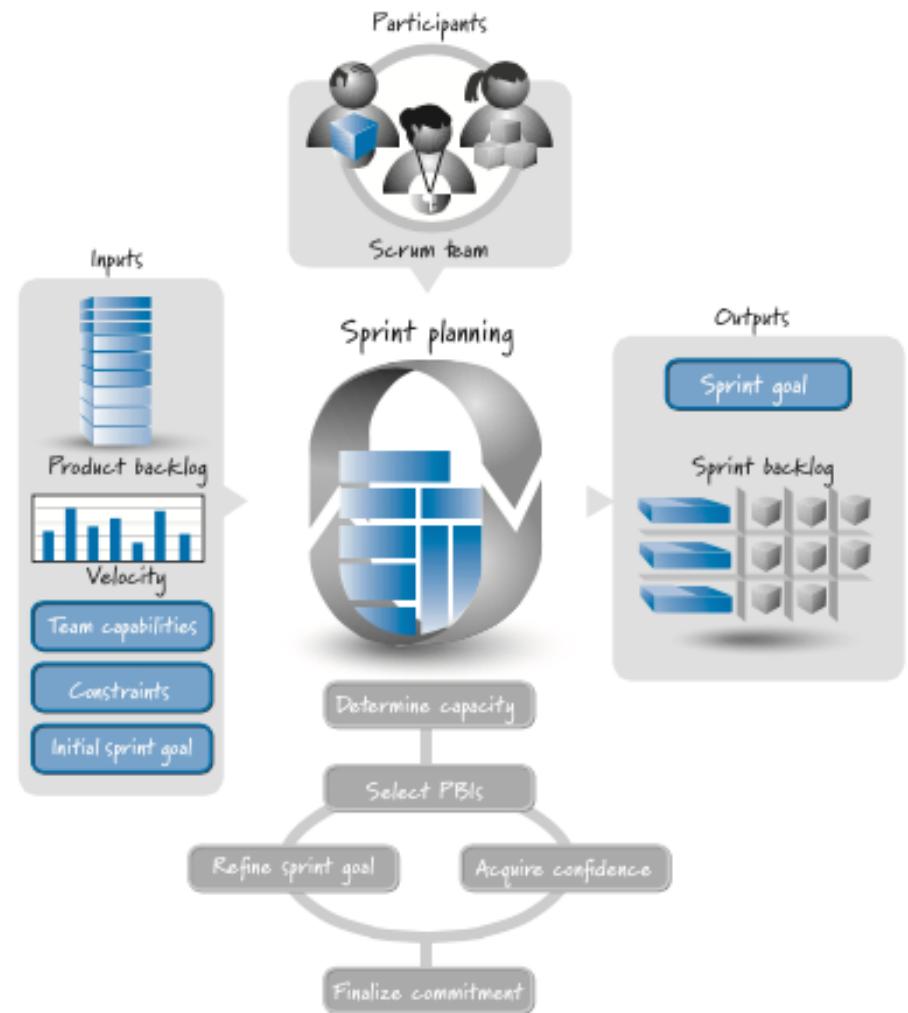
### **Preliminary Whole-Project Estimate**

Effort = 180 story points ÷ 2.25 story points/staff week = 80 staff weeks

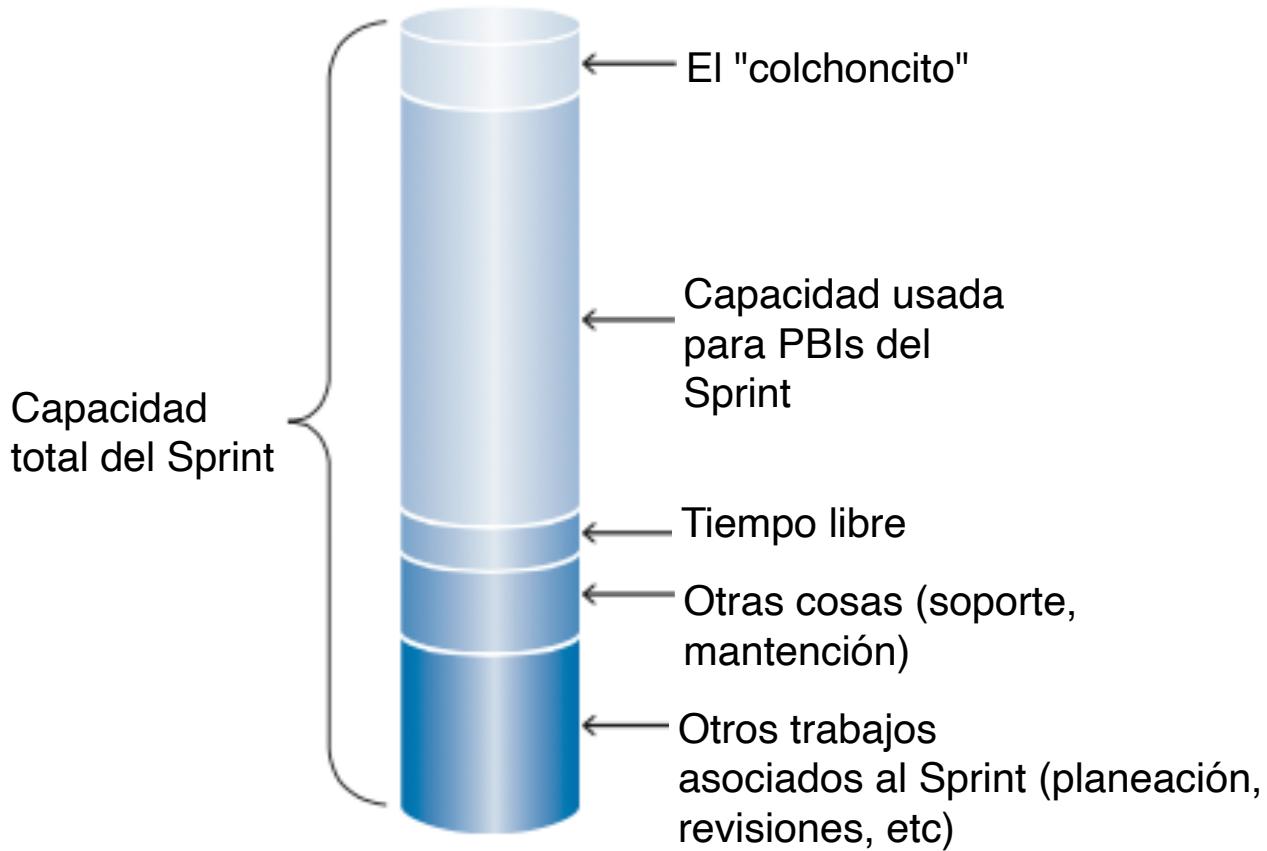
Schedule = 180 story points ÷ 9 story points/calender week = 20 calendar weeks

# Planeación a nivel de Sprint

- ▶ participa todo el equipo
- ▶ product owner presenta PBIs priorizados
- ▶ otros inputs necesarios incluyen velocidad de desarrollo, capacidad del equipo, restricciones
- ▶ ScrumMaster debe ayudar a que el equipo incluya set razonable de items

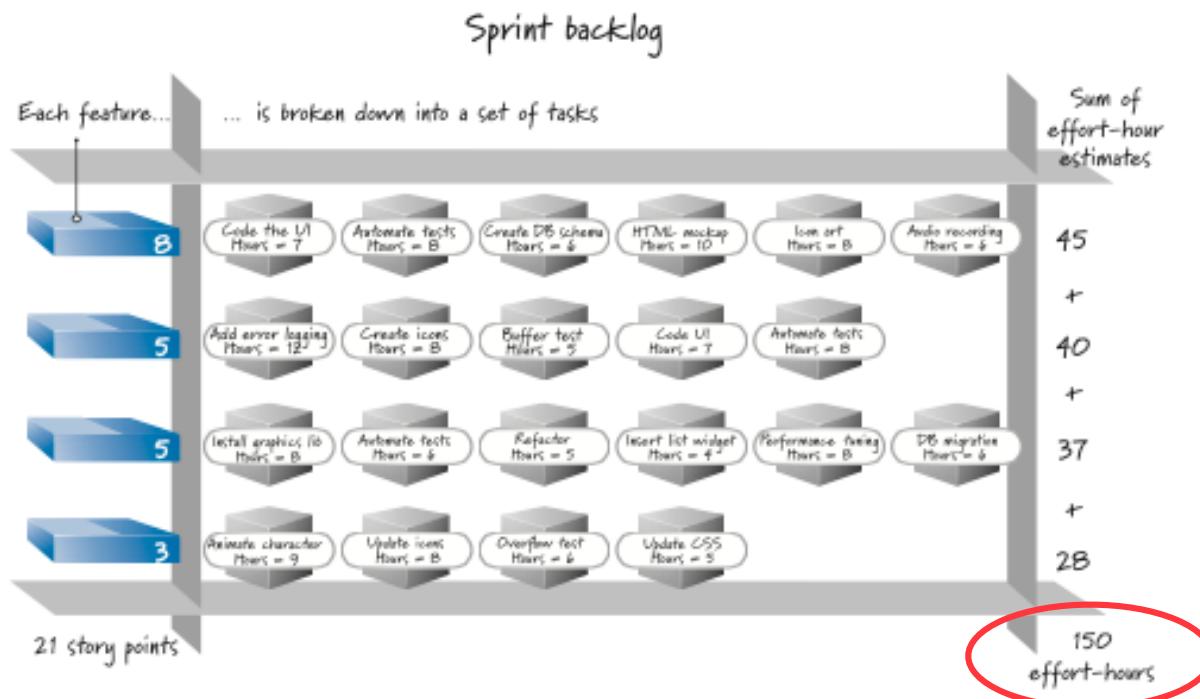


# Capacidad del equipo vs capacidad real disponible para el Sprint

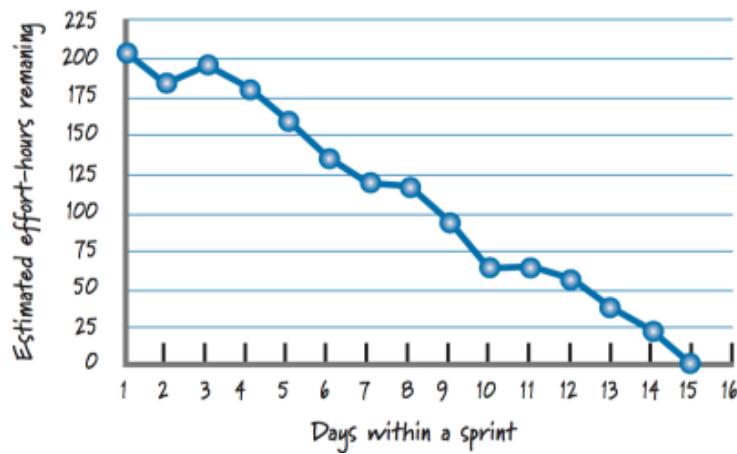


# Ejemplo de Planificación de un Sprint

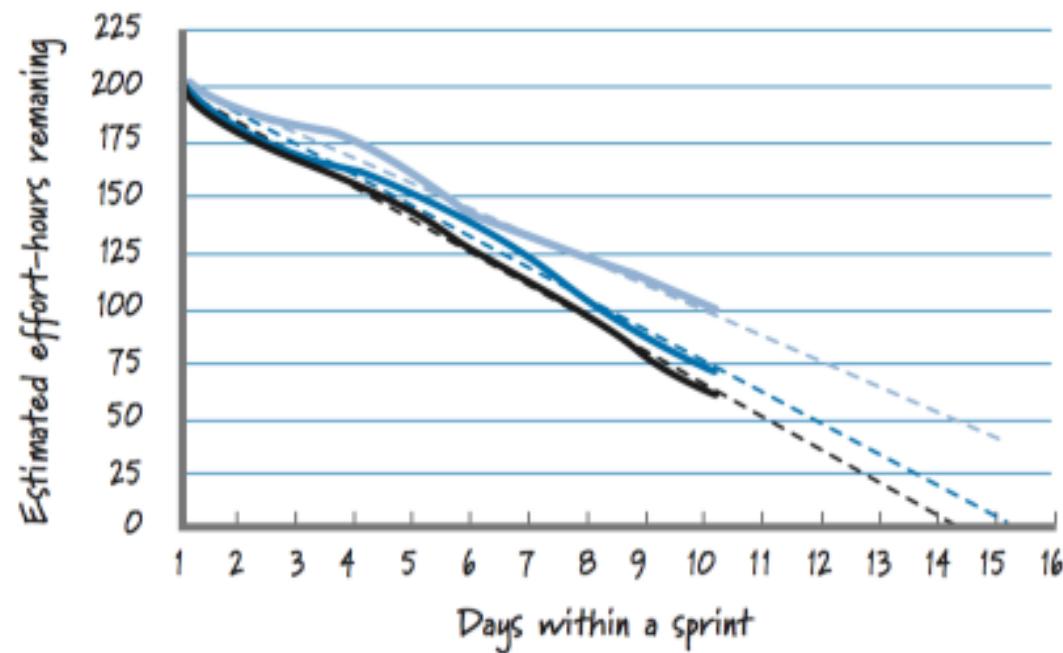
Person	Days Available (Less Personal Time)	Days for Other Scrum Activities	Hours per Day	Available Effort-Hours
Jorge	10	2	4–7	32–56
Betty	8	2	5–6	30–36
Rajesh	8	2	4–6	24–36
Simon	9	2	2–3	14–21
Heidi	10	2	5–6	40–48
Total				140–197



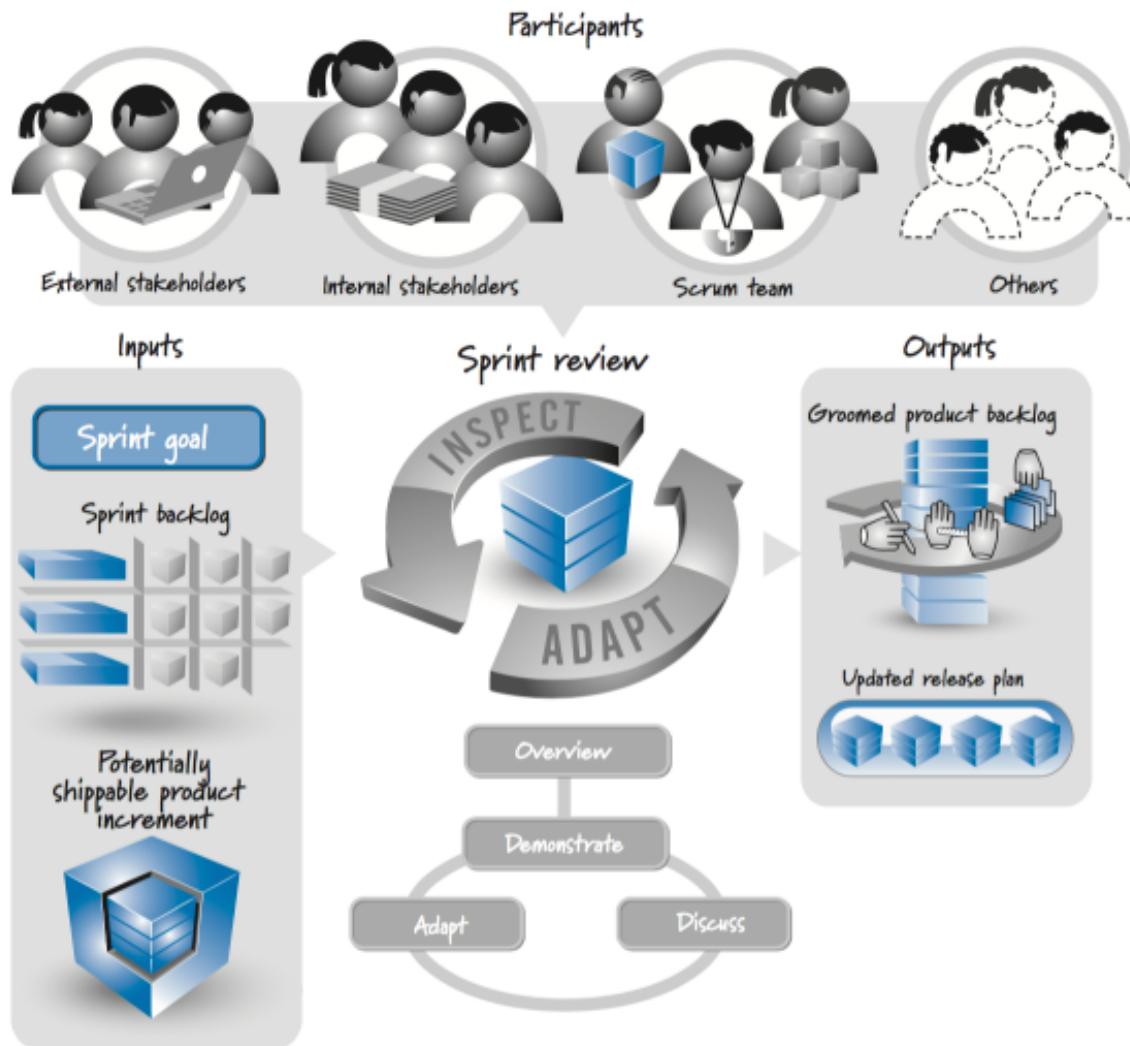
# Sprint burndown charts



— On time   — Early   — Late



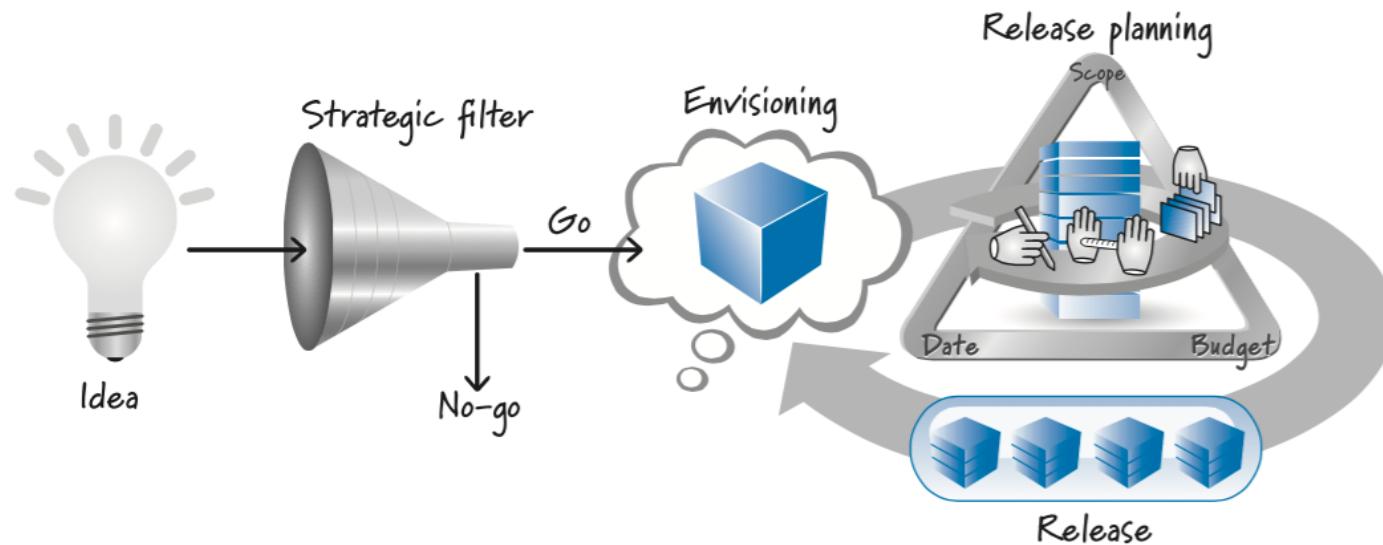
# Output del Sprint Review



nuevo product backlog

ajuste al plan para el release del producto

# Planeación del Producto



- ▶ Visión de producto genera un backlog inicial (PBI)
- ▶ Proceso de planeación genera una "hoja de ruta" para los futuros releases

# La hoja de ruta del producto (SR4U)

N		Q3—Year 1	Q4—Year 1	Q1—Year 2
Market map		Initial launch	Better results More platforms	Sophisticated users
Feature/benefit map		Basic learning Basic filtering	Improved learning Complex queries	Define sources Learn by example
Architecture map		100K concurrent web users	iOS and Android	Web services interface
Market events		Social Media Expo	Review Everything User Conference	
Release schedule		1.0	2.0	3.0

# Planeación de Portafolio

- ▶ cuando organización maneja más de un producto
- ▶ en qué backlog items del portafolio trabajar y en qué orden
  - ▶ ¿ mejorar producto A o extender producto B primero ?
- ▶ participantes deben tener una visión amplia del negocio

# Planeación Multinivel

Level	Horizon	Who	Focus	Deliverables
Portfolio	Possibly a year or more	Stakeholders and product owners	Managing a portfolio of products	Portfolio backlog and collection of in-process products
Product (envisioning)	Up to many months or longer	Product owner, stakeholders	Vision and product evolution over time	Product vision, roadmap, and high-level features
Release	Three (or fewer) to nine months	Entire Scrum team, stakeholders	Continuously balance customer value and overall quality against the constraints of scope, schedule, and budget	Release plan
Sprint	Every iteration (one week to one calendar month)	Entire Scrum team	What features to deliver in the next sprint	Sprint goal and sprint backlog
Daily	Every day	ScrumMaster, development team	How to complete committed features	Inspection of current progress and adaptation of how best to organize the upcoming day's work