



Pontificia Universidad Católica de Chile  
Escuela de Ingeniería  
Departamento de Ciencia de la Computación

# Clase 9

## Planificación

IIC2143 - Ingeniería de Software  
Sección 1

Rodrigo Saffie

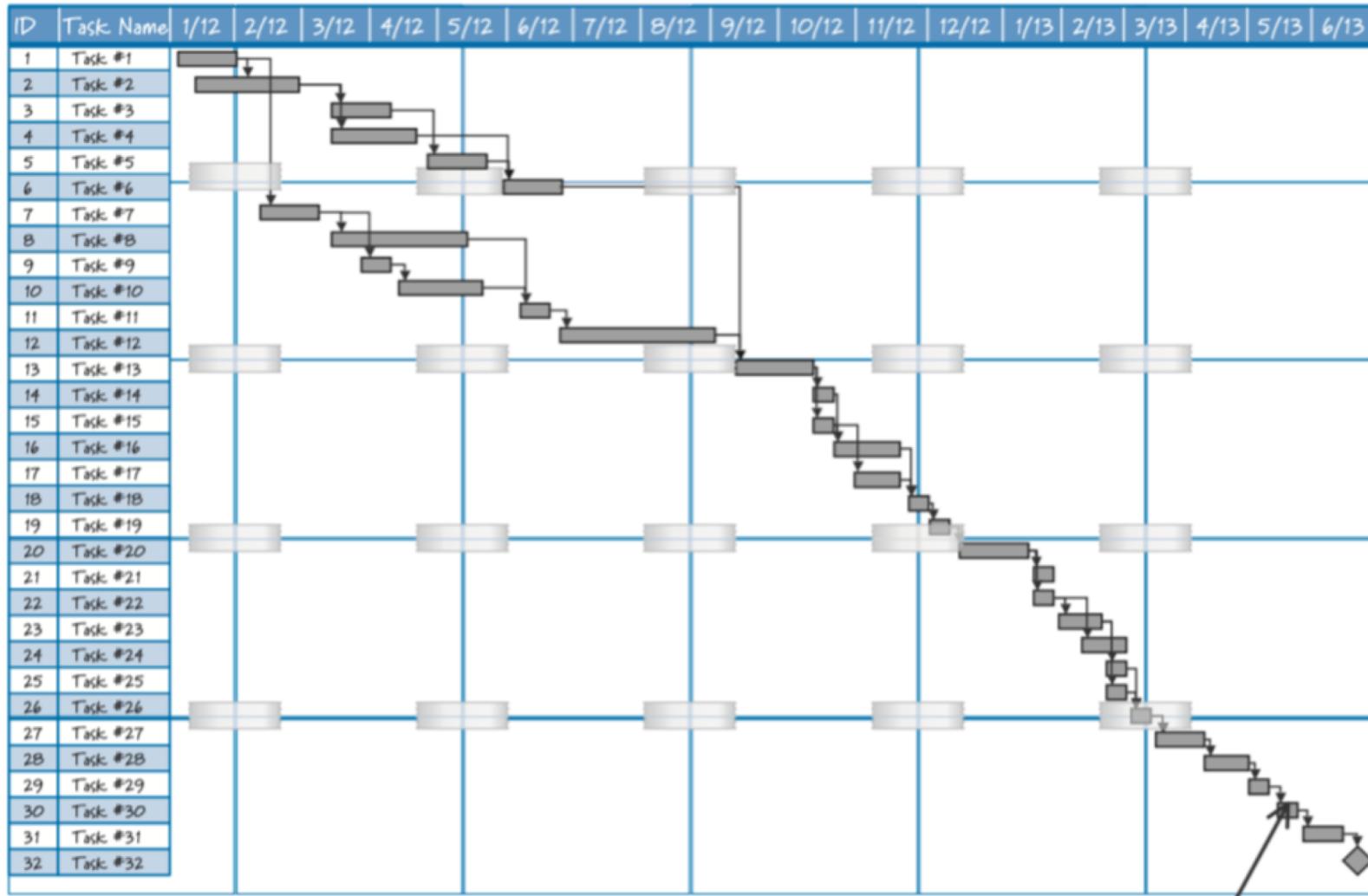
[rasaffie@uc.cl](mailto:rasaffie@uc.cl)

9 de abril de 2018

# Planificación

- Luego de estimar y dimensionar los distintos parámetros de un proyecto se puede planificar

# La famosa Carta Gantt



# ¿Por qué no es buena idea?

- Dificulta el desarrollo ágil
  - Bastante trabajo para construirla y modificarla
- Rara vez se respeta el plan inicial
- Compromiso de tiempo por sobre todo el resto
- Puede insistir en mantener un plan que está fuera de la realidad

# Hitos de planificación de software

- Iteración: producto potencialmente entregable
- Lanzamiento (release): producto entregable al final del proceso iterativo
- Producto: desarrollo en a múltiples procesos iterativos
- Portafolio: varios productos gestionados por una compañía

# Planificación de un *release*

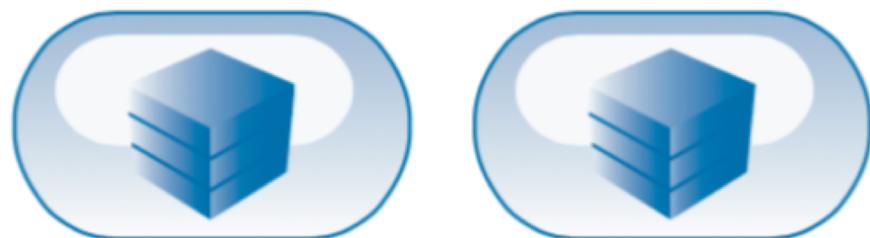
- Organización decide qué y cuándo entregar
- Se suelen combinar varios *sprints* para un *release*
- Dependiendo de la confianza en el proceso de desarrollo y las exigencias del negocio se puede implementar *continuous delivery*

# Planificación de un *release*

Release after multiple sprints



Release every sprint



Release every feature



# Restricciones: tiempo, alcance, presupuesto

- En *Scrum* se asume que no es posible planear con las 3 restricciones fijas
- El plan puede ajustarse si se deja una de estas variables flexible:
  - Ajuste por tiempo
  - Ajuste por alcance
  - Ajuste por presupuesto

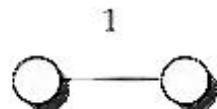
# Presupuesto variable

- Agregar más gente al proyecto no mejorará el tiempo de desarrollo
- Ley de Brooks:
  - “Aregar más gente a un proyecto con retraso lo atrasará aún más”
  - “Una mujer puede tener un bebé en nueve meses, pero nueve mujeres no pueden tener un bebé en un mes”

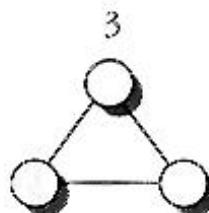
# Restricciones: presupuesto variable



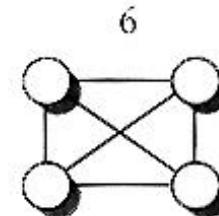
# Restricciones: presupuesto variable



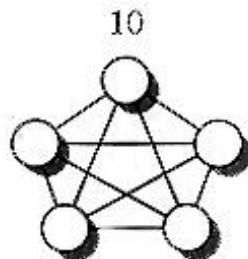
Communication path  
with two programmers



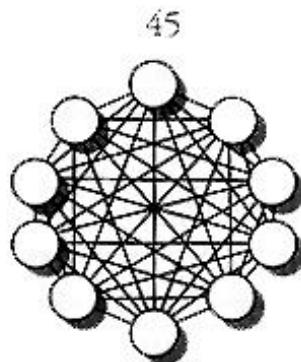
Communication paths  
with three programmers



Communication paths  
with four programmers



Communication paths  
with five programmers



Communication paths  
with ten programmers

# Restricciones:

## tiempo, alcance, presupuesto

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

# Ajuste por calidad

- Una práctica muy común

# ¿Qué es calidad?

David Garvin lo definió como 8 dimensiones (1984):

- **Vista transcendental:** la calidad se percibe, pero no se puede explicar
- **Vista del usuario:** la calidad en base a los objetivos del usuario final
- **Vista del productor:** la calidad según especificaciones del producto
- **Vista del producto:** la calidad en función de lo que hace el producto
- **Vista del valor:** la calidad en base a lo que está dispuesto a pagar un consumidor

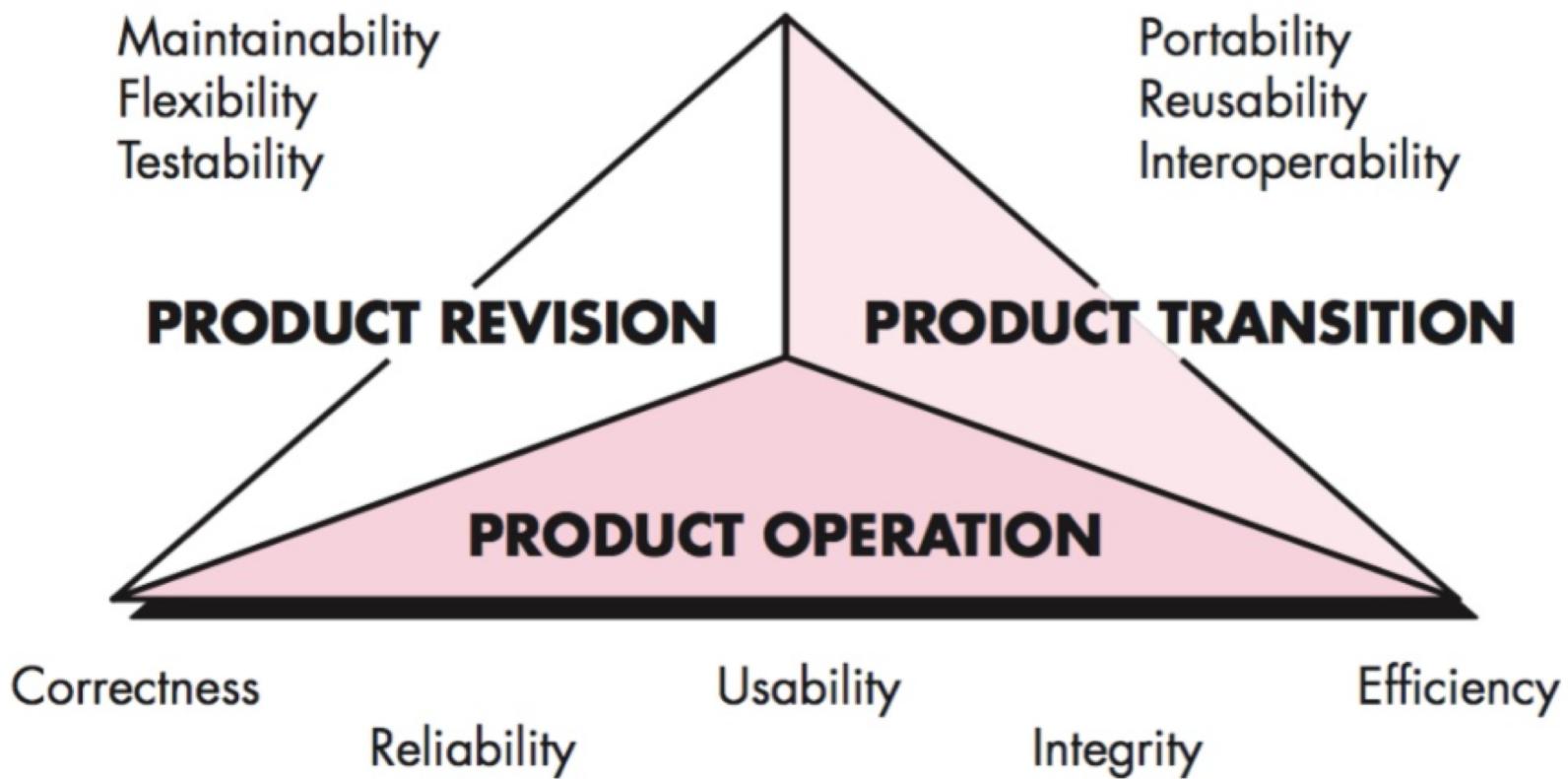
# ¿Qué es calidad?

Calidad según Pressman (2009):

“Un desarrollo de *software* efectivo, aplicado de una manera que crea un producto útil que provee valor cuantificable para aquellos que lo producen y aquellos que lo utilizan.”

# ¿Qué es calidad?

Factores de Calidad [McCall, 1977]:



# Ajuste por calidad

- Una práctica muy común
- Pésima idea:
  - Puede estar bajo las expectativas del usuario
  - Puede generar una tremenda deuda técnica

# Deuda técnica (*technical debt*)

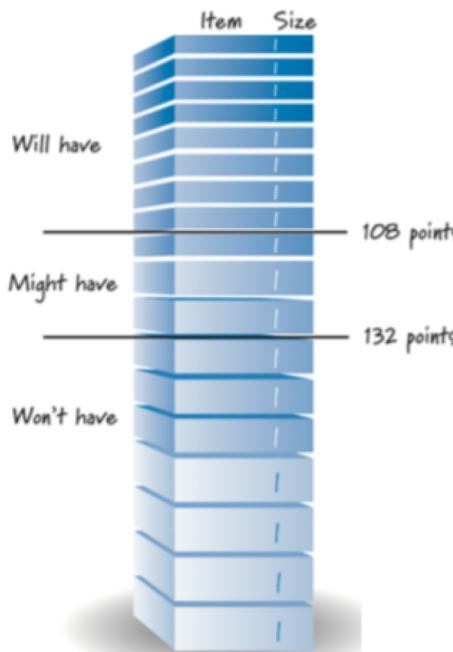
- Es un concepto para reflejar la implicancia en costo adicional de rehacer trabajo, debido a elegir una solución simple y fácil sobre una que tomaría más tiempo.

Technical Debt Quadrant		
	Reckless	Prudent
Deliberate	"We don't have time for design"	"We must ship now and deal with consequences (later)"
Inadvertent	"What's Layering?"	"Now we know how we should have done it"

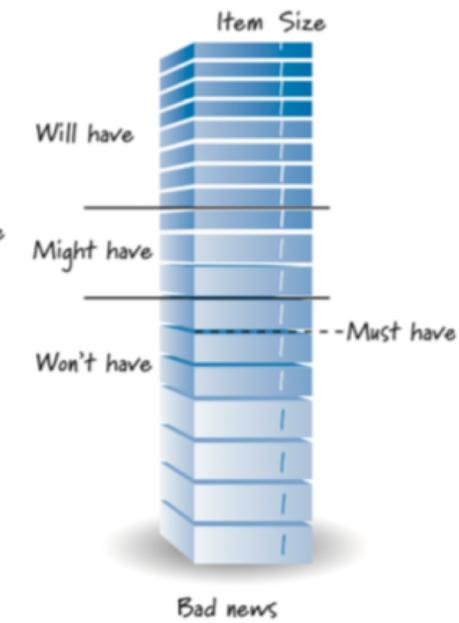
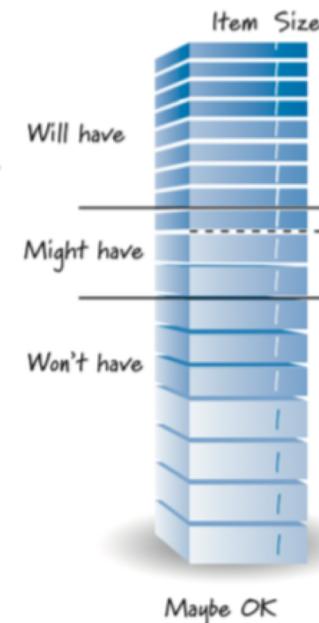
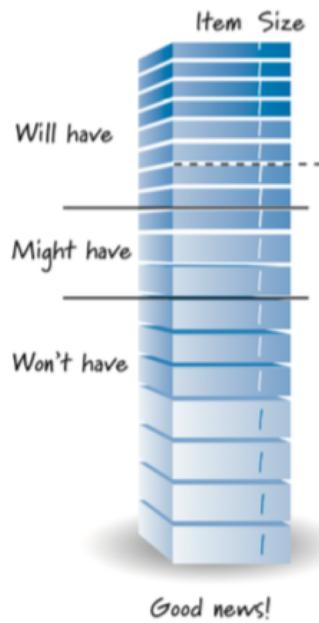
# ¿Cómo planificar un *release*?

## Minimum Releasable Features (MRF):

- Conjunto mínimo de funcionalidades para un *release*
- Responsabilidad del *product owner*
- Al planificar siempre se debe ser consciente de los MRF del proyecto

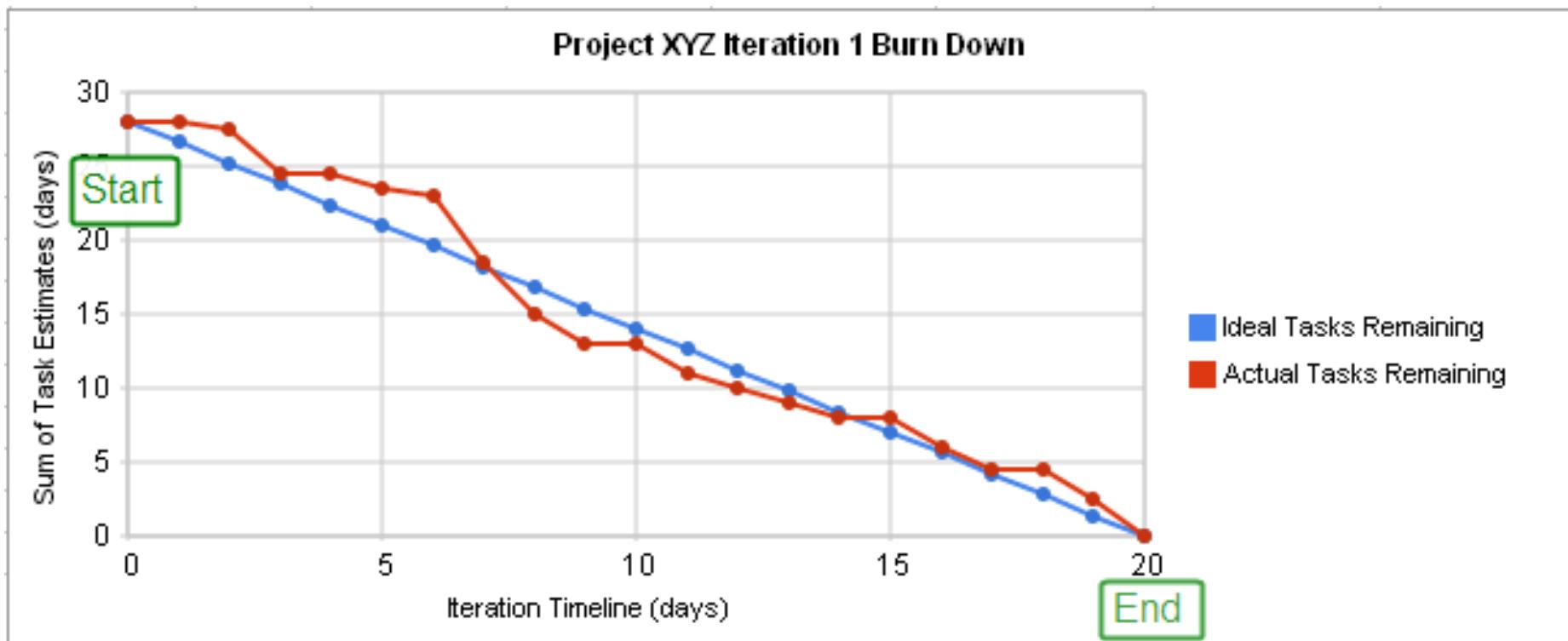


- es conveniente no dejar el MRF muy calzado con el tiempo (máximo 70%)
  - pueden surgir otros “must-have”
- usar story points y ritmos observados



# Comunicando el progreso

- En SCRUM se suele utilizar un *burn down chart*



# Planificación 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



Pontificia Universidad Católica de Chile  
Escuela de Ingeniería  
Departamento de Ciencia de la Computación

# Clase 9

## Planificación

IIC2143 - Ingeniería de Software  
Sección 1

Rodrigo Saffie

[rasaffie@uc.cl](mailto:rasaffie@uc.cl)

9 de abril de 2018