Ingeniería de Software I

Segundo Cuatrimestre de 2016

Clase 3b: Introducción a los métodos ágiles y Scrum

Buenos Aires, 18 de Agosto de 2016

Introducción – Agile Manifesto

Manifiesto por el Desarrollo Ágil de Software

Estamos descubriendo mejores maneras de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de esta experiencia hemos aprendido a valorar:

Individuos e interacciones	**sobre**	***procesos y herramientas***
Software que funciona	**sobre**	***documentación exhaustiva***
Colaboración con el cliente	**sobre**	***negociación de contratos***
Responder ante el cambio	**sobre**	***seguimiento de un plan***

Esto es, aunque los elementos a la derecha tienen valor, nosotros valoramos por encima de ellos los que están a la izquierda

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Dave Thomas
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland

http://www.agilemanifesto.org (2001)

Algunos conceptos

- Time boxing:
  - Priorizar duración sobre alcance
  - La reducción del alcance facilita mantener la calidad
  - La limitación estricta del tiempo estimula a mantener el foco
  - Se aplica a iteraciones, reuniones, tareas grupales o individuales

- Desarrollo incremental:
  - El sistema va creciendo como consecuencia de la integración de nuevas funcionalidades
  - Cada funcionalidad que se agrega está testeada en forma unitaria, y también se prueba integrada al resto de la aplicación

- Desarrollo iterativo:
  - El proyecto se divide en iteraciones, que para el equipo son mini-proyectos
  - El resultado de cada iteración debe ser la aplicación 'andando' con una porción de la funcionalidad requerida

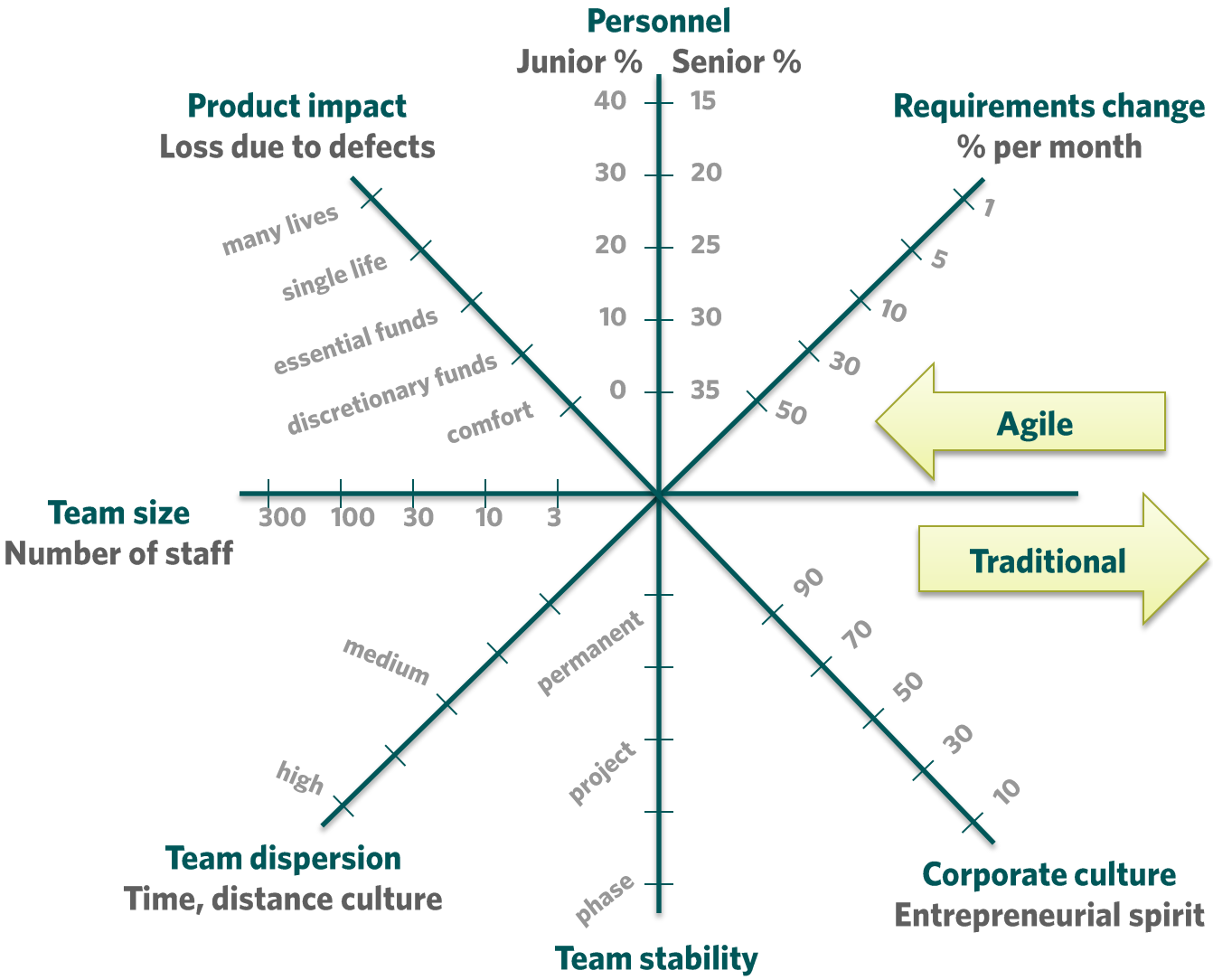
Principios aplicables al desarrollo ágil

- **DRY**: Don't Repeat Yourself, no crear código, herramientas o infraestructura duplicadas aún a costa de algún esfuerzo adicional.
- **KISS**: Keep It Simple!, evitar la complejidad no esencial, aún a costa de algún esfuerzo adicional
- **Do The Simplest Thing That Could Possibly Work**: evitar la anticipación injustificada y las generalizaciones prematuras
- **YAGNI**: You Ain't Gonna Need It, idem
- **DOGBITE**: Do it, Or it's Gonna Bite you In The End, algunas cosas sí hay que hacerlas con anticipación (y suelen ser las menos atractivas).
- **SOC**: Separation of Concerns, evitar el solapamiento de funcionalidad entre las diversas características o módulos de un programa.
- **Done-Done-Done**: ser sincero sobre el estado Terminado de una tarea o unidad de entrega.
- **Refactoring**
- **Shipping is a feature, and your product must have it**

Anti-principios

- Antipattern: patrón organizacional, metodológico o de diseño, popular pero contraproducente
  - BDUF: Big Design Up Front, tiende a oponerse a YAGNI y KISS
  - Optimización prematura: sin tener métricas que la justifiquen
  - Bug Driven Development: los requerimientos se completan en forma de bugs

SCRUM - En qué proyectos es recomendable utilizarlo?



Fuente: Right-Sizing Agile Development, webinar by Steve McConnell

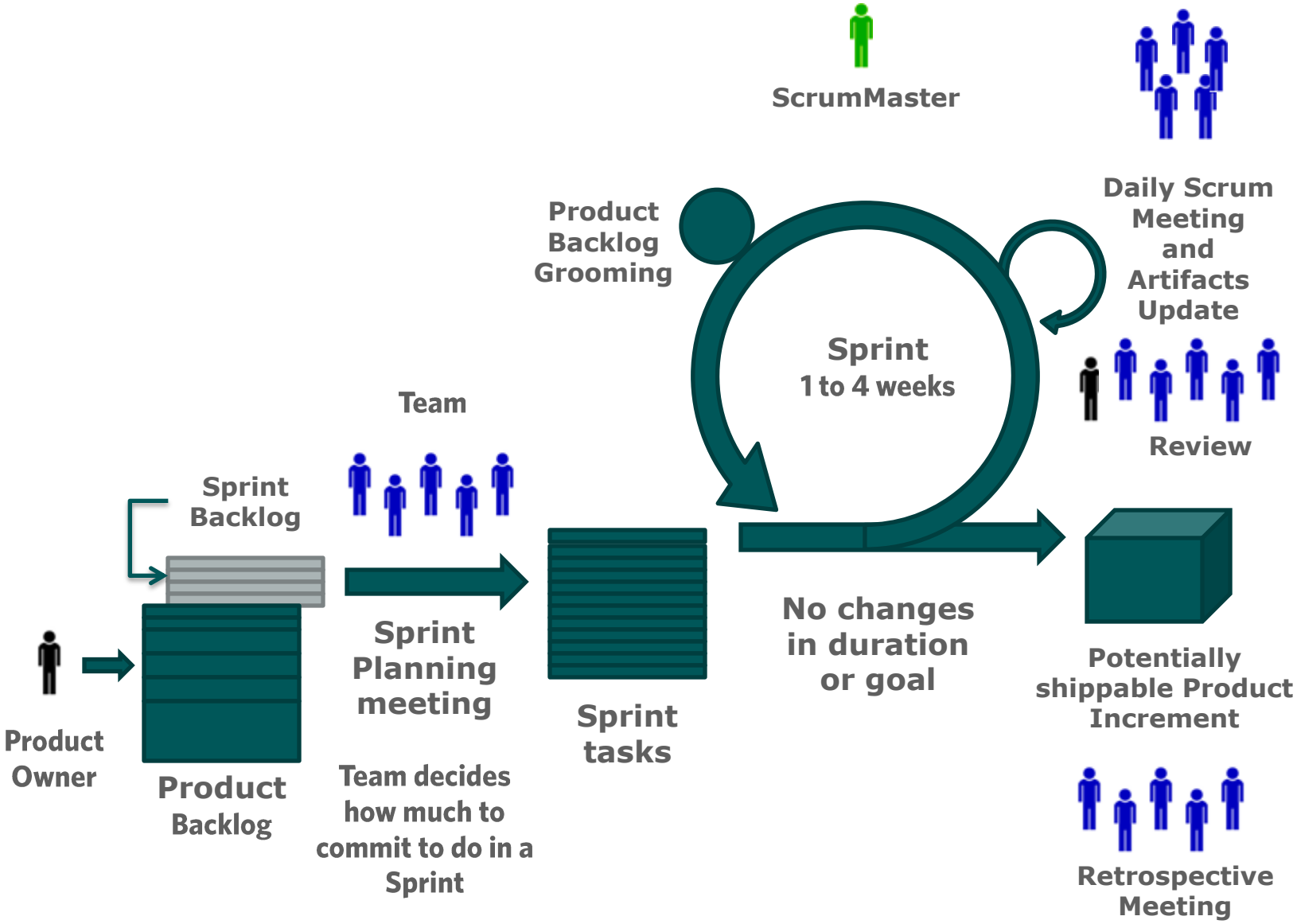
Scrum: ¿Qué es?

¿Qué es un scrum?

*"Un scrum es un **agrupamiento** (formación fija) en Rugby. 8 integrantes de cada equipo, llamados "delanteros", se enfrentan agrupados para tratar de obtener la pelota, que es introducida por uno de los equipos en el centro de la formación."*

Pero… ¿qué tiene que ver con la Ingeniería de Software?

SCRUM – Gráfico del proceso



Scrum - Roles

Product Owner

- Representa al cliente o usuario final
- Define prioridades para el negocio

Equipo

- Inter-disciplinario
- Auto-organizado

ScrumMaster

- Líder facilitador
- Coach

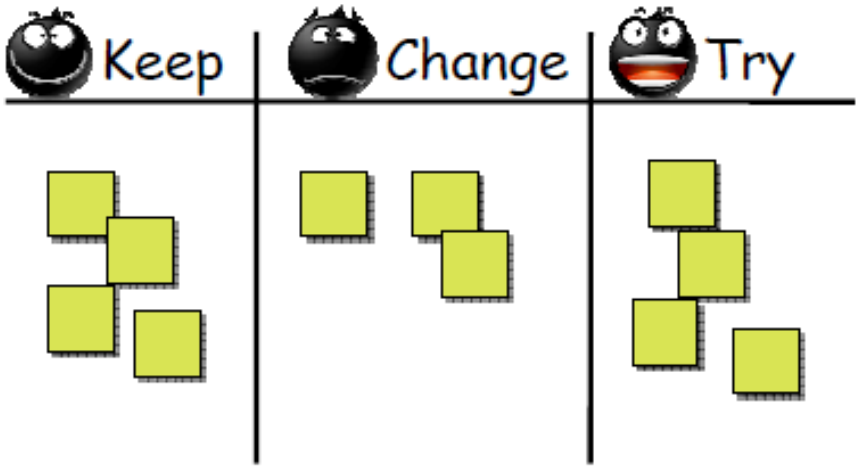
Estimaciones y Formato del Plan

- Las estimaciones con Scrum se hacen de manera "relativa" (por ejemplo de 1 a 5), a través de una especie de "piedra, papel y tijera"
- Se busca consenso
- El plan generado en la reunión tiene sólo dos niveles: stories y tareas
- Se usa un "taskboard", donde las tareas están divididas en "Planificadas, En curso, Terminadas"

Scrum - Retrospectiva

El equipo se reune después de cada iteración para evaluar y mejorar sus métodos y la interacción del equipo

1 Brainstorming



2 Elegir algo concreto para mejorar

Debe ser *"SMART"*:

- Specific
- Measurable
- Attainable
- Relevant
- Timely

Reportes En Scrum

Velocity = cantidad de story points que un equipo hace para un producto en un mes

Product Burndown chart

- Da una indicación de que tan rápido el equipo está terminando los requerimientos del product backlog en cada sprint

Story o task burndown chart

- Da una indicación de que tan rápido el equipo está terminando stories o "bajando horas" en un sprint.
- Muestran comportamientos "disfuncionales"