Ingeniería de Software II

Ingeniería de Software II

Segundo Cuatrimestre de 2011

Clase 2: Introducción a los métodos ágiles y Scrum

Buenos Aires, 18 de Agosto de 2011

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

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)

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

¿Qué características tienen los métodos ágiles?

- Subconjunto de métodos que adoptan el Modelo de Ciclo de Vida iterativo e incremental
- Los requerimientos evolucionan a través de la colaboración entre equipos multifuncionales
- Procesos disciplinados con alta visibilidad que promueven la inspección y adaptación frecuente
- Liderazgo que alienta el trabajo en equipo, la auto-organización y la responsabilidad
- Prácticas que permiten la entrega rápida de software de alta calidad
- Alineación del desarrollo con las necesidades de los clientes y los objetivos de la empresa

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

Más características

- "Livianas"
- "Adaptativas" en forma empírica
- Rápidas, pero nunca apuradas
- Centradas en el cliente
- Empujan las decisiones hacia los niveles más bajos
- Favorecen la confianza, la honestidad y el coraje
- Favorecen la auto-organización y la polifuncionalidad
- Prefieren el trabajo en equipos reducidos y el "pairing"
- Fomentan la proximidad física de los miembros del equipo y la comunicación cara a cara u online, en lugar de escribir documentación o emails

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

Otros 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

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

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**: debe facilitarse a través de la reducción de su costo.
- **Shipping is a feature, and your product must have it**

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

Anti-principios

- It Works on my PC: no es una excusa, debe funcionar en todos los ambientes.
- 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
  - Groupthink: situaciones que desalientan el disenso
  - Bug Driven Development: los requerimientos se completan en forma de bugs

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

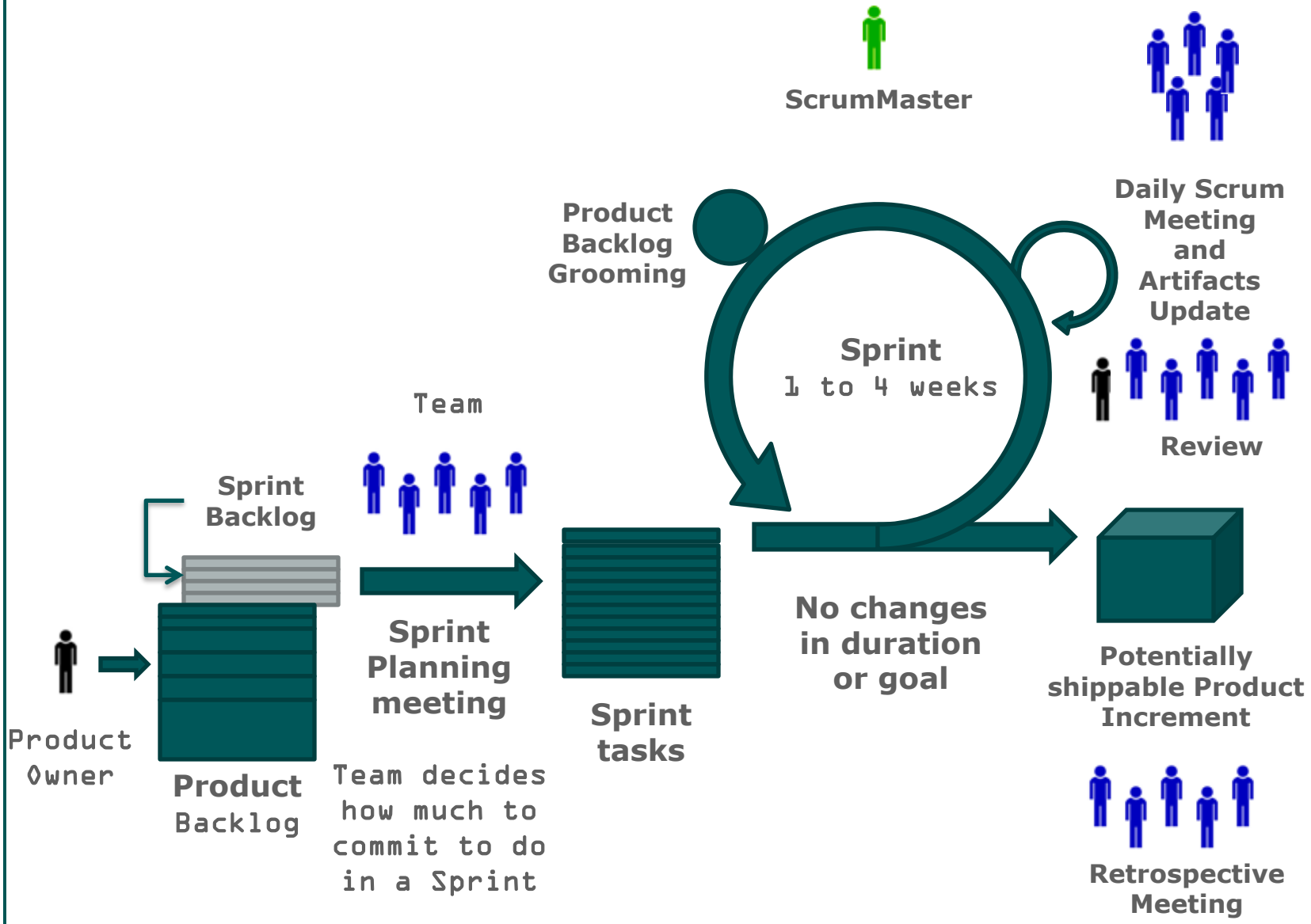
¿Qué es un scrum?

*"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?

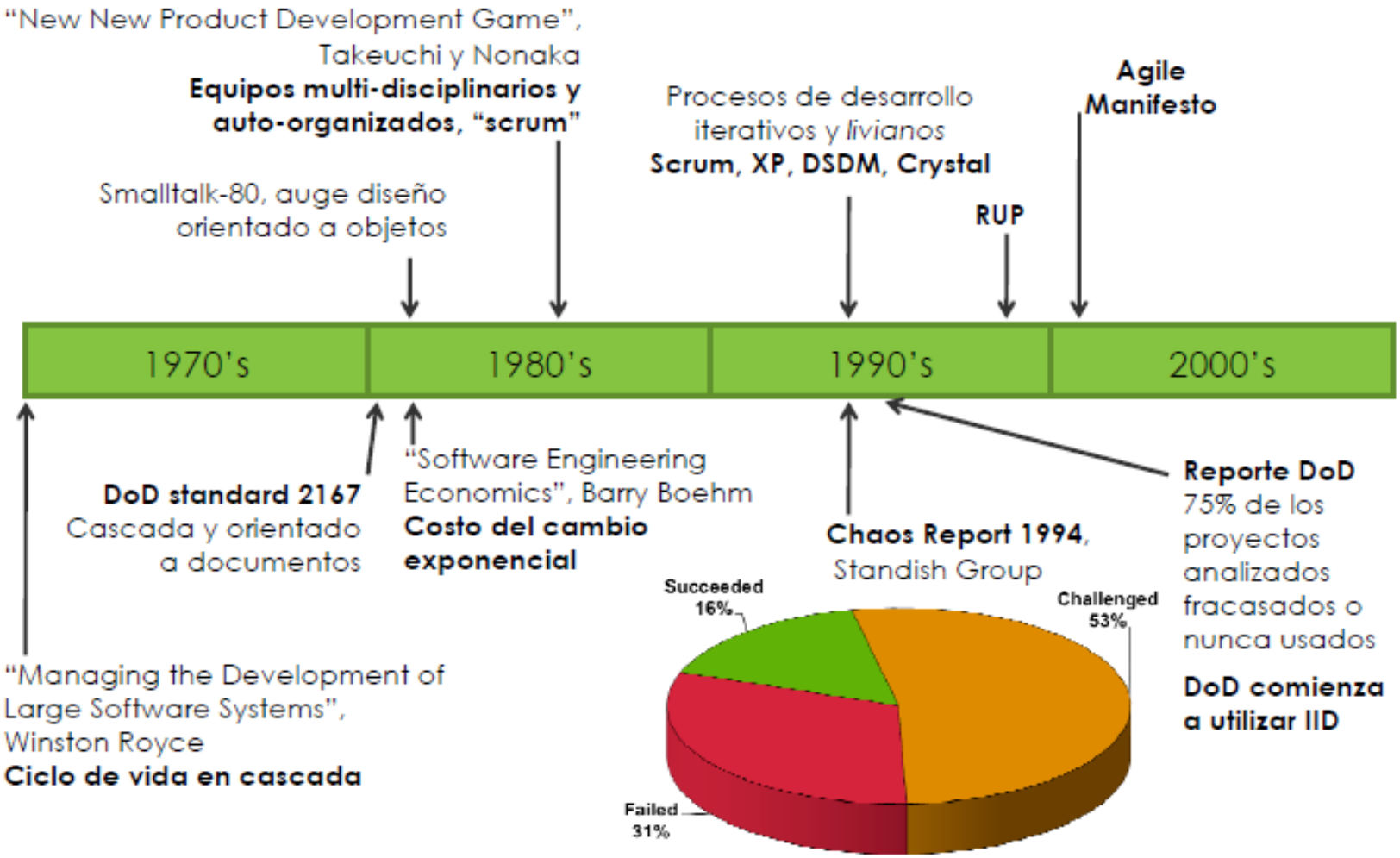
© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

SCRUM – Gráfico del proceso



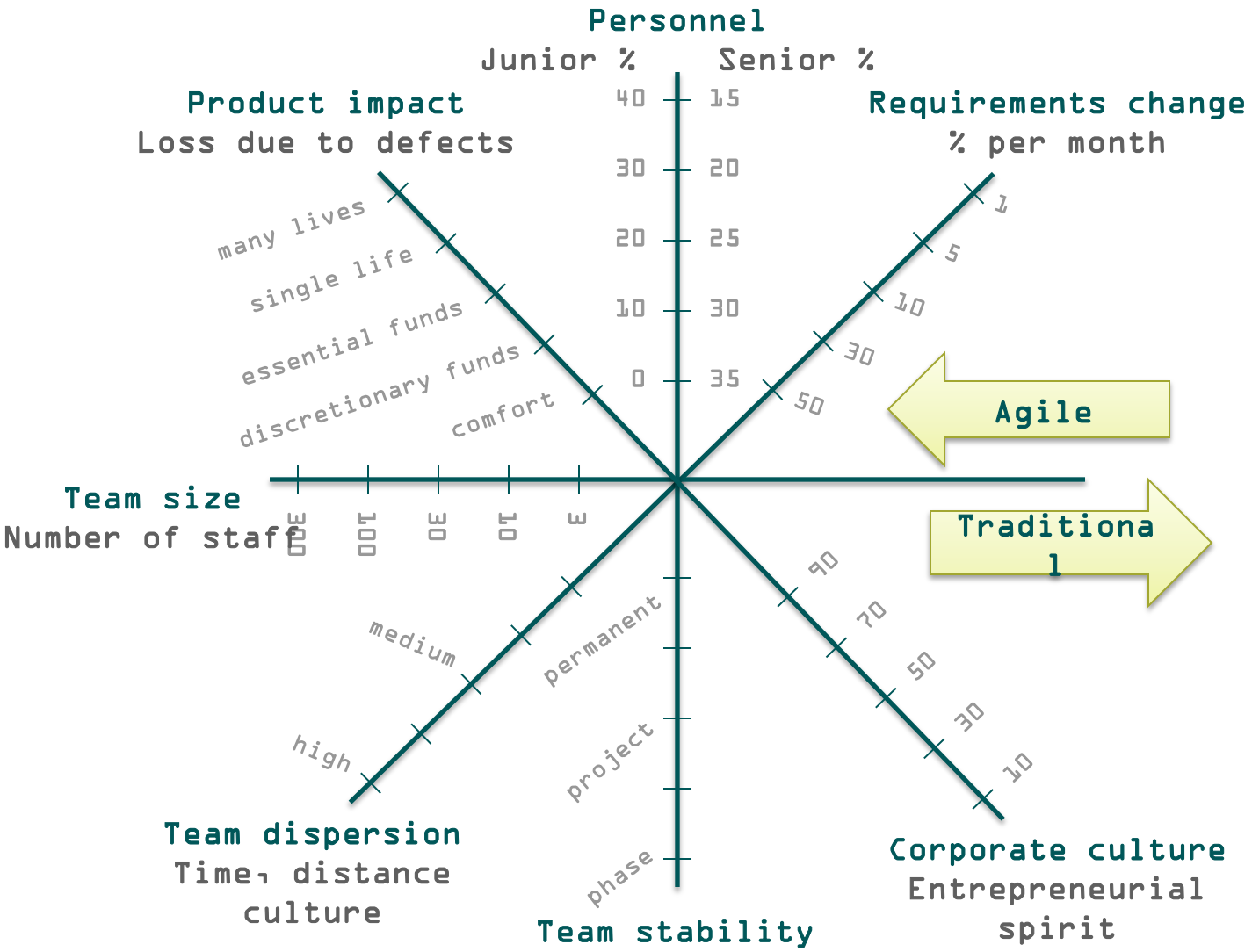
© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

SCRUM - Historia



© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

SCRUM - En qué proyectos es recomendable utilizarlo?



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

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

Scrum - Roles

Product Owner

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

Equipo

- Inter-disciplinario
- Auto-organizado

ScrumMaster

- Líder facilitador
- Coach

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

SCRUM - Roles

Product Owner (PO)

- Representa los intereses del cliente y comunica una visión del producto
- Es el responsable de maximizar el Retorno de Inversión, priorizando los requerimientos con mayor valor para el negocio
- Es el responsable de escribir los requerimientos como user stories
- Resume el input de los usuarios y el resto de los stakeholders en una vista única de los requerimientos del sistema, con prioridades definidas
- Responde las consultas del equipo sobre los requerimientos
- Acepta o rechaza el trabajo hecho por el equipo

Team

- Pilares fundamentales: Comunicación y Cooperación
- 7 +/- 2 miembros, full time, elenco estable
- Feature teams – incluye todos los skills necesarios para entregar el producto terminado: desarrolladores, diseñadores, testers, etc.
- Auto-organizado y auto-administrado
- Estima y define los alcances y límites de cada iteración
  - Son los Pigs, y el resto son chickens
- Puede haber varios equipos

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

SCRUM – Roles (cont.)

ScrumMaster (SM)

- Facilitador del equipo
  - Ayuda al equipo a aprender y usar Scrum y producir valor para el negocio
  - Remueve impedimentos y protege al team de interferencias externas
  - No es el manager del team ni es un PM – no asigna tareas
- Responsable de arrancar y mantener funcionando el proceso Scrum
- Puede ser full-time o part-time, según las necesidades del equipo
- Puede no tener formación de sistemas
- No puede ser el Product Owner

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

SCRUM – Producto y Product Backlog

Producto

- En Scrum, un Team desarrolla un producto definido por un Product Owner, usando un proceso adaptativo supervisado y asistido por el Scrum Master

Product Backlog

- Es una lista priorizada de todas las características que el Team debe darle al producto
- Debe expresar los requerimientos funcionales y no funcionales
- Puede incluir los defectos a corregir
- Es conveniente que incluya las características internas, no visibles al usuario (arquitectura, escalabilidad, uso de herramientas de desarrollo, etc).
- Cada item tiene un valor para el negocio estimado por el PO y un esfuerzo estimado por el Team
  - Los valores y los esfuerzos suelen estar limitados (serie de Fibonacci).
- La priorización natural es en orden decreciente de relación valor/esfuerzo
- Pueden agregarse o quitarse ítems, o cambiar las estimaciones o la prioridad en cualquier momento
- No hay un formato definido para los items – suelen ser User Stories
- Siempre hay un sólo Product Backlog

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

SCRUM - Ejemplo de Product Backlog de Carrito de Compras

ID	Item	Prio	Value	Effort	Status
1	As a buyer, I want to place a book in a shopping cart - Details	1	7	5	Done
2	As a buyer, I want to remove a book from the shopping cart - Details	2	5	2	Done
5	Improve transaction performance - Details	3	4	13	In Sprint
7	Investigate solutions for speeding up credit card validation	4	4	20	In Sprint
6	Upgrade all servers to Apache 2.2.3	5	2	13	Ready
8	Diagnose and fix order processing script errors (JIRA SC-234)	6	3	3	Ready
3	As a shopper, I want to create and save a wish list - Details	7	2	40	Not Ready
4	As a shopper, I want to add items to my wish list - Details	8	2	20	Not Ready

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

SCRUM – User stories

- **Formato: As a [user role] I want to [goal] so that [benefit]**
  - Por ejemplo: Como comprador, quiero agregar libros al carrito de compras para elegir los libros que voy a comprar
  - Especifican qué hay que hacer, evitando especificar cómo hacerlo
  - Pueden agregarse detalles (breves)
  - Pueden tener attachments que definan el contexto
  - Conviene que incluyan los criterios de aceptación
  - Pueden estar en una herramienta, en Google Docs o en tarjetas
  - Se completa la info en una conversación
  - Escribir buenas stories suele ser lo más difícil al arrancar un proyecto con Scrum
  - Beneficios: ayuda a pensar desde la perspectiva del usuario

- **Modelo INVEST**
  - Independent – tan independiente de otras stories como sea posible
  - Negotiable – demasiados detalles inhiben la comunicación Team-PO-cliente
  - Valuable – deben tener valor para el usuario y/o para quien paga el desarrollo
  - Estimable – legibles para el Team y no demasiado grandes
  - Small – menos de 3 semanas-hombre
  - Testable – los criterios de aceptación deben ser tan objetivos como sea posible

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

SCRUM – User story en papel

#1 - As a buyer, I want to place a book in a shopping cart so I can choose the books that I will buy

Acceptance criteria:

1. User can select a book from the catalog
2. User can search the catalog by author or title.
3. If the book is already in the shopping cart, its quantity is increased.
4. If the book is not already in the shopping cart, it is added with a quantity of one.
5. After the operation, the shopping cart is displayed.

Effort: 5
Value: 7

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

SCRUM – Inicio de un proyecto

Antes de comenzar

- Conseguir el apoyo del sponsor para usar Scrum
- Capacitar en Scrum a todos los involucrados (PO, Team, cliente final, etc).
- Planificación:
  - Definir la duración inicial de los sprints (1 a 4 semanas)
  - El PO debe definir la visión inicial del producto
  - El PO debe crear el Product Backlog inicial y priorizar los ítems iniciales (al menos como para dos sprints)
    - Los ítems para los dos próximos sprints deben cumplir con INVEST
    - Los ítems menos prioritarios pueden ser sólo un título (epic)
  - El Team debe estimar los ítems más prioritarios
    - La estimación suele hacerse en forma relativa, en story points
  - Definición de los releases previstos y la funcionalidad que se espera entregar en cada uno
  - Evaluación de riesgos

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

SCRUM – En un Sprint

Qué es un sprint?

- Timeboxed iteration: no se extiende por ningún motivo, aunque no se haya terminado el trabajo prometido ni cumplido el objetivo
- El team decide cuáles ítems del PB entran en un sprint en función de la estimación
- Conviene que el sprint tenga un objetivo, como "implementación básica de débitos automáticos"
- El Scrum Master protege al equipo de interferencias durante el sprint
  - Se pueden agregar o cambiar stories sólo con el acuerdo del team
- Al final del Sprint el team revisa los resultados con los stakeholders y hace una demo
  - El resultado debería ser un incremento de funcionalidad potencialmente desplegable, integrado y testeado
- Luego el team inspecciona el proceso y lo adapta si es necesario

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

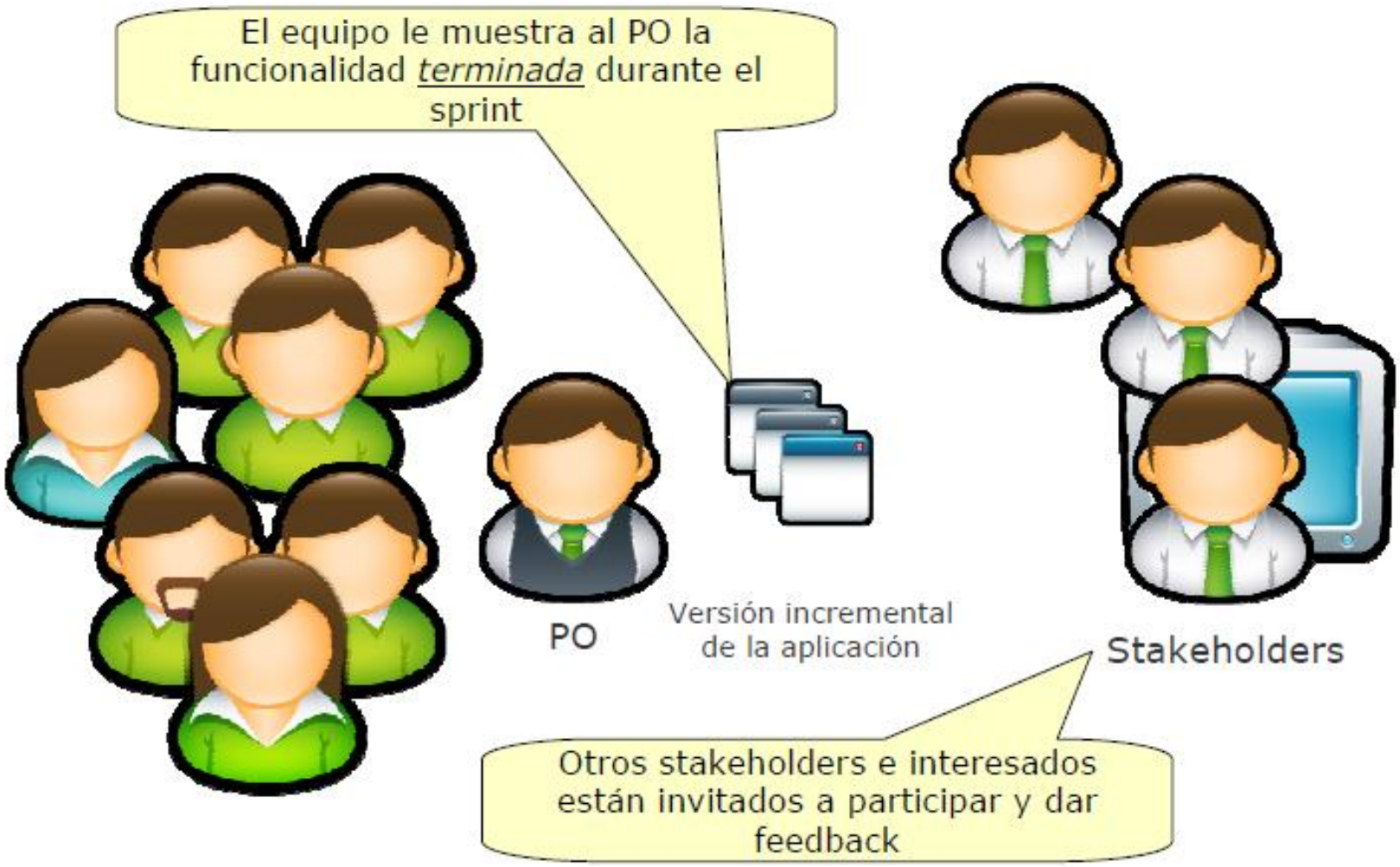
SCRUM – Planificación del Sprint

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

SCRUM – Reunión Diaria

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

SCRUM – Demo de un sprint (sprint review)

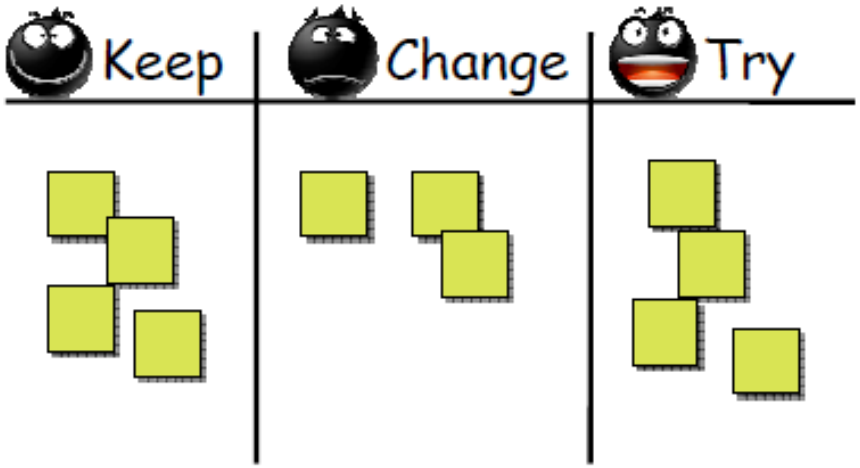


© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

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

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

SCRUM – Pizarra low-tech

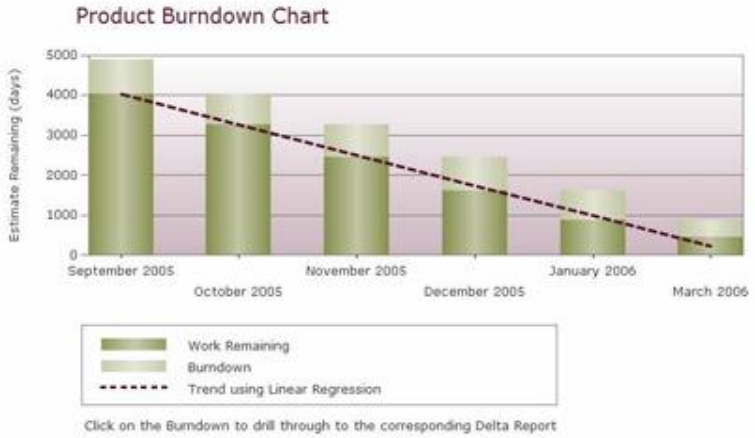
© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

Reportes En Scrum

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

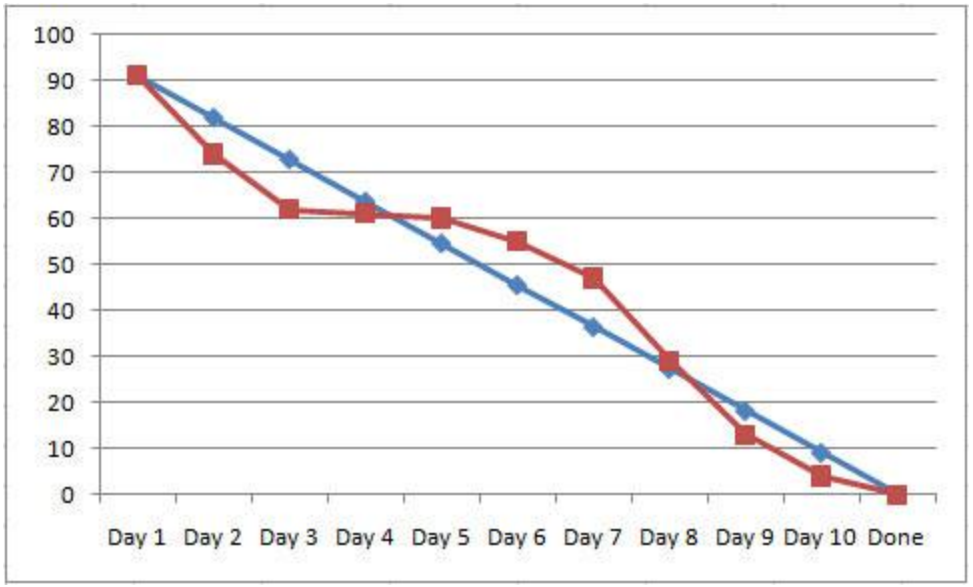
Product Burndown chart

- 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"



© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

SCRUM – Pizarra (cont.)

Contenido

- Las tareas suelen agruparse en columnas "Not Yet Started", "In Progress", "Completed" y un sector "Blocked"
- El Sprint Burndown Chart es muy útil para saber cómo va el sprint

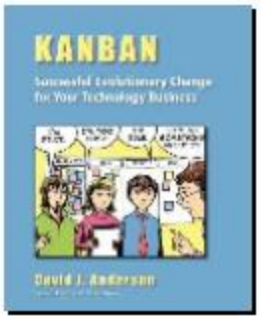
Actualización

- Conviene que la actualización sea inmediata, al tomar una tarea o al terminarla
  - Si un miembro del equipo trabaja en forma remota, debe pedirle a alguien que esté en la oficina que actualice la pizarra
- Cada tarea lleva su estimación de horas restantes, que debe ser actualizada antes o durante el Daily
- Al final del día también deben actualizarse las horas restantes para poder actualizar el Burndown Chart
  - Las horas restantes de una tarea pueden crecer
- Si surge una tarea inesperada se agrega a la pizarra, con su estimación si es posible
  - Nadie debe trabajar en tareas que no estén en la pizarra
- Las tareas que se bloquean se pasan al sector "Blocked" con un Post-It rojo que explica el motivo del bloqueo

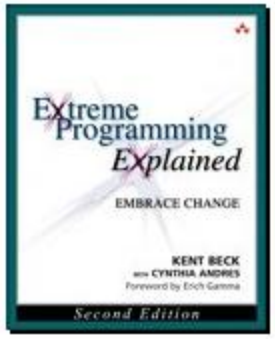
© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

Más información

Agile Project Management with Scrum.
Ken Schwaber, 2004.



Kanban.
David Anderson, 2010.

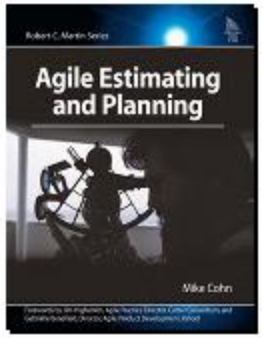


Extreme Programming Explained: Embrace Change.
Kent Beck, 2004.

Implementing Lean Software Development: From Concept to Cash.
M. Poppendieck, T. Poppendieck, 2006.

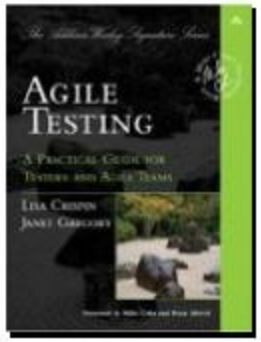
© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

Más información

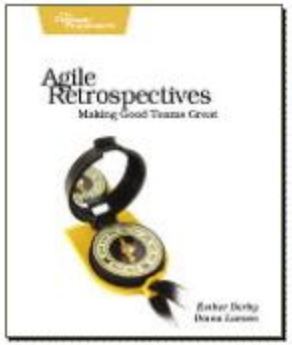


Agile Estimating and Planning.
Mike Cohn, 2005.

The Art of Agile Development.
James Shore, 2007.



Agile Testing: A Practical Guide for Testers and Agile Teams.
L. Crispin, J. Gregory, 2009.



Agile Retrospectives: Making Good Teams Great.
E. Derby, D. Larsen, 2006.

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

Aún más información

- **User Stories Applied: For Agile Software Development.** Mike Cohn, 2004
- **Agile & Iterative Development, A Manager's Guide.** Craig Larman, 2002
- **Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum.** C. Larman, B. Vodde, 2008
- **Succeeding with Agile: Software Development Using Scrum.** Mike Cohn, 2009
- **Software by Numbers: Low-Risk, High-Return Development.** M. Denne, J. Cleland-Huang, 2003

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011

Recursos on-line

- Comunidad ágil latinoamericana: www.agiles.org
- Scrum: www.scrumalliance.org
- Agile Alliance: www.agilealliance.org
- Manifiesto ágil: www.agilemanifesto.org
- Grupo ScrumDevelopment (inglés):
- http://groups.yahoo.com/group/scrumdevelopment/
- Foro latinoamericano (español):
- http://tech.groups.yahoo.com/group/foro-agiles/

© Cátedra de Ingeniería de Software II – FCEN – UBA, 2011