

2.2 Metodologías ágiles

Mike Cohn Henrik Kniberg
Ernesto Gafeuille
Juan Rodríguez Hortalá



1

Plan-driven development

- Plan-driven development = técnicas clásicas de IS
- Lo más adecuado para cierto tipo de proyectos SW (sistemas críticos de control de aviones o coches), donde
 - Se necesita un análisis completo del sistema
 - Los requisitos son muy estables



3

Referencias

- Sommerville, I. *Software Engineering International Edition. Ninth Edition*. Pearson, 2011
- *Extreme Programming: A gentle introduction*. <http://www.extremeprogramming.org/>
- Schwaber K., Sutherland J. *The Scrum Guide*. Scrum.org, 2010. <http://www.scrum.org/scrumguides/>
- Kniberg, H. *Scrum and XP from the Trenches: How we do Scrum*. InfoQ Enterprise Software Development Series, 2007. Disponible online en <http://www.infoq.com/minibooks/scrum-xp-from-the-trenches>



2

Plan-driven development

- Proyectos SW muy grandes y con larga vida: SW aeroespacial o del gobierno
 - Equipos de trabajo grandes, con miembros de varias empresas y dispersión geográfica
 - Tiempos de desarrollo muy largos. Ej: SW control de un avión => 10 años de proyecto desde el inicio de la especificación al desplegado
 - La sobrecarga en planificación, diseño y documentación está justificada por
 - necesidad de coordinar el trabajo de muchos subgrupos de varias empresas
 - sistemas críticos: calidad pq se juegan vidas humanas
 - se prevé un mantenimiento que durará muchos años e involucrará a mucha gente



4

Agile development

- Hoy en día los negocios operan en un entorno global que está en constante cambio

- El SW debe poder construirse rápidamente antes de que cambien los requisitos. Si no queda anticuado ya el mismo día de la entrega
- A menudo es imposible obtener un conjunto completo y estable de requisitos
- Al usar el pesado enfoque plan-driven en sistemas de negocios de tamaño pequeño o mediano
 - La sobrecarga de planificación/diseño/documentación es tan grande que domina el proceso de desarrollo
 - Se gasta más tiempo en decidir cómo se va a desarrollar el SW que en desarrollarlo y testearlo



5

Agile development

- Antecedentes: la necesidad de ser capaces de desarrollar sistemas con rapidez, y de disponer de procesos de desarrollo que puedan manejar el cambio se detectó hace tiempo
- 80s: desarrollo incremental y Rational Unified Process
- Finales de los 90: un paso más allá con los enfoques ágiles: DSDM, XP, Scrum



7

Failure

The Standish Group has studied over 40,000 projects in 10 years.

IT project success rate 1994: 15%

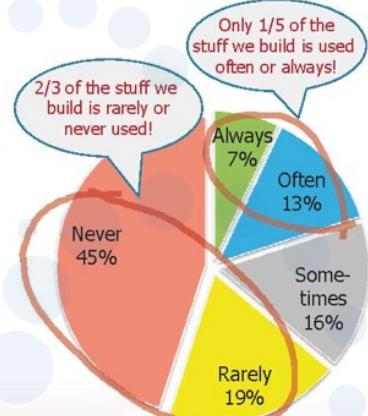
Average cost & time overrun: ≈170%

IT project success rate 2004: 34%

Average cost & time overrun: ≈70%



Features and functions used in a typical system



Sources:
<http://www.softwaremag.com/L.cfm?DocID=Newsletter/2004-01-15/Standish>
<http://www.infog.com/articles/Interview-Johnson-Standish-CHAO%202002>
Standish group study reported at XP2002 by Jim Johnson, Chairman

6



6

Agile development

- Procesos diseñados para producir rápidamente SW útil
 - El SW no se produce como una única unidad, sino como una serie de incrementos. Cada incremento añade una o varias funcionalidades al sistema SW
- Características comunes de las metodologías ágiles
 - Los procesos de especificación, diseño e implementación están entrelazados, no aislados como tradicionalmente
 - No existe una espec. de requisitos detallada: sólo se definen las características más importantes del sistema
 - Se minimiza la documentación de diseño o se genera automáticamente mediante herramientas



8

Agile development

- El sistema se desarrolla en una serie de versiones (releases)
 - Los usuarios y/o el cliente final se involucran en la especificación y evaluación de cada versión: proponen cambios y nuevos requisitos para versiones futuras
 - Esto proporciona un feedback rápido acerca de los requerimientos cambiantes
- Los incrementos son pequeños y las nuevas releases se entregan al cliente cada 2 o 3 semanas (Sommerville)
- Se minimiza la documentación de cualquier tipo: favorece la comunicación informal y personal frente a la escrita
- Cortar el esfuerzo empleado en burocracia de los procesos evitando hacer trabajo que tiene un valor a largo plazo dudoso y eliminando documentación que probablemente nunca se utilizará



Mountain Goat Software, LLC



9

Agile is like a homing missile

Assumptions:

- The customer discovers what he wants
- The developers discover how to build it
- Many things change along the way



11

Waterfall is like a cannonball

Assumptions:

- The customer knows what he wants
- The developers know how to build it
- Nothing will change along the way



26



10

Agile Manifesto

www.agilemanifesto.org

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.



12



Mountain Goat Software, LLC



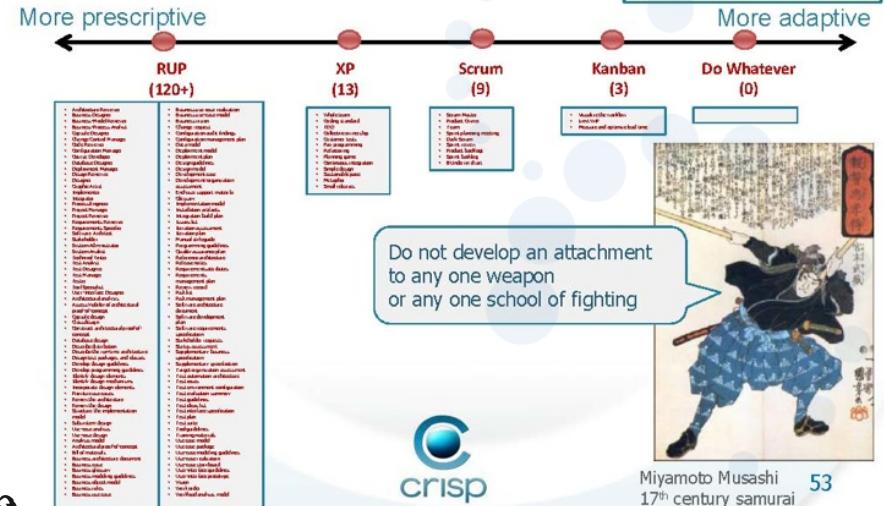
Metodologías ágiles

- Probablemente la más conocida sea extreme programming (XP)
 - Otros enfoques ágiles: Scrum, Crystal, Adaptive SW Development, DSDM y Feature Driven Development
 - El éxito de estos métodos ha conducido al desarrollo de ciertos métodos híbridos que mezclan el enfoque ágil con la IS clásica plan-driven: agile modelling e instanciaciones ágiles del Proceso Unificado de Rational
 - "XP as a minimal RUP process": <http://www.objectmentor.com/resources/articles/RUPvsXP.pdf>



13

Prescriptive vs Adaptive processes



14

Metodologías ágiles

- Todas estas metodologías ágiles están basadas en la noción de desarrollo y entrega incremental
 - Proponen diferentes formas de implementar esa noción
 - Todos comparten una serie de principios basados en el manifiesto ágil, por lo que tienen mucho en común

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is to provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.



15

XP y Scrum

- Distintas metodologías ágiles instancian estos principios de diversas maneras
- Nos centraremos en XP y Scrum
 - Xp se centra en prácticas de programación mientras que Scrum se centra en organización y gestión
 - Debido a esto estas dos metodologías se complementan y a menudo son utilizadas en combinación



16

XP

- Una de las metodologías ágiles más conocidas y utilizadas: en su totalidad o algunas de sus prácticas
- Recibe su nombre porque en XP se intenta llevar el desarrollo iterativo a niveles extremos
- En XP varias versiones nuevas de un sistema pueden ser desarrolladas por programadores diferentes, integradas y testeadas en un sólo día



17

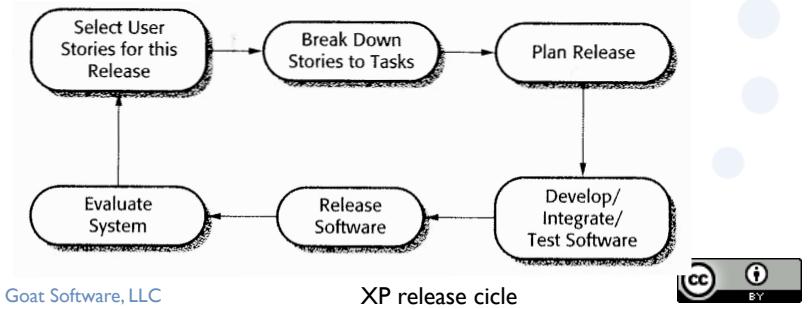
XP: prácticas

Principle or practice	Description
Incremental planning	Requirements are recorded on Story Cards and the Stories to be included in a release are determined by the time available and their relative priority. The developers break these Stories into development 'Tasks'. See Figures 3.5 and 3.6.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.
Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity.
On-site customer	A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

19

XP

- En XP los requisitos se expresan como escenarios llamados “historias de usuario” que se dividen en una serie de tareas y se implementan directamente
- La programación por parejas y el TDD son innovaciones de XP y prácticas fundamentales en XP
- Se hacen releases del sistema cada poco tiempo



XP release cycle



18

XP

- La especificación se basa en “historias de usuario” que desarrolla el cliente junto con los miembros del equipo
- “Planning game”: entre el cliente y el equipo las historias se dividen en tareas y se estima el esfuerzo requerido. El cliente prioriza las historias según su utilidad para el negocio, y así se decide cuales irán en el próximo incremento
- En el próximo incremento se pueden añadir historias, y modificar o eliminar historias existentes aún no implementadas, según sugiera el entorno de negocio (el mercado del cliente)



XP release cycle



20

Prescribing Medication

Kate is a doctor who wishes to prescribe medication for a patient attending a clinic. The patient record is already displayed on her computer so she clicks on the medication field and can select current medication', 'new medication' or 'formulary'.

If she selects 'current medication', the system asks her to check the dose. If she wants to change the dose, she enters the dose and then confirms the prescription.

If she chooses 'new medication', the system assumes that she knows which medication to prescribe. She types the first few letters of the drug name. The system displays a list of possible drugs starting with these letters. She chooses the required medication and the system responds by asking her to check that the medication selected is correct. She enters the dose and then confirms the prescription.

If she chooses 'formulary', the system displays a search box for the approved formulary. She can then search for the drug required. She selects a drug and is asked to check that the medication is correct. She enters the dose and then confirms the prescription.

The system always checks that the dose is within the approved range. If it isn't, Kate is asked to change the dose.

After Kate has confirmed the prescription, it will be displayed for checking. She either clicks 'OK' or 'Change'. If she clicks 'OK', the prescription is recorded on the audit database. If she clicks on 'Change', she reenters the 'Prescribing medication' process.



Ejemplo de historia

Mountain Goat Software, LLC



21

XP

- A veces se hacen iteraciones “spike” (de asentamiento), en las que no se implementa nueva funcionalidad sino que se dedican a la arquitectura del sistema o a documentar (siempre lo mínimo necesario)
- Siempre se cumple con las fechas límites de los incrementos, apoyándose en los tests automatizados para evitar regresiones
- No se diseña anticipando cambios que quizás no ocurran, sino que se emplea el diseño más simple para las historias que se soportan en un incremento concreto. La refactorización constante evita que el diseño se degenera con el tiempo



Mountain Goat Software, LLC

XP release cycle



23

Task 1: Change Dose of Prescribed Drug

Task 2: Formulary Selection

Task 3: Dose Checking

Dose checking is a safety precaution to check that the doctor has not prescribed a dangerously small or large dose.

Using the formulary ID for the generic drug name, look up the formulary and retrieve the recommended maximum and minimum dose.

Check the prescribed dose against the minimum and maximum. If outside the range, issue an error message saying that the dose is too high or too low. If within the range, enable the 'Confirm' button.

Tareas asociadas a la historia anterior



22

● Especificación de un test para la tarea 3

- Los tests no siempre son automatizables

Test 4: Dose Checking

Input:

1. A number in mg representing a single dose of the drug.
2. A number representing the number of single doses per day.

Tests:

1. Test for inputs where the single dose is correct but the frequency is too high.
2. Test for inputs where the single dose is too high and too low.
3. Test for inputs where the single dose × frequency is too high and too low.
4. Test for inputs where single dose × frequency is in the permitted range.

Output:

OK or error message indicating that the dose is outside the safe range.



24

Scrum: estamos perdiendo la carrera de relevos

"En enfoque de 'carrera de relevos' en el desarrollo de productos ... puede entrar en conflicto con los objetivos de máxima velocidad y flexibilidad. En su lugar, un enfoque holístico o estilo 'rugby' - donde un equipo intenta ir a la distancia como una unidad, pasando la pelota hacia adelante y hacia atrás -pueden servir mejor a los actuales requisitos competitivos".

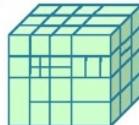
Hirotaka Takeuchi and Ikujiro Nonaka, "The New New Product Development Game", Harvard Business Review, January 1986.



25

Scrum in a nutshell

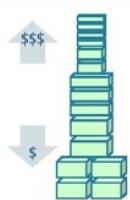
Split your product



Large group spending a long time building a big thing
Small team spending a little time building small thing
... but integrating regularly to see the whole



Optimize business value



Split time

January

4



Optimize process



Henrik Kniberg



27

Scrum en 100 palabras

- Scrum es un proceso ágil que nos permite centrarnos en ofrecer el más alto valor de negocio en el menor tiempo.
- Nos permite rápidamente y en repetidas ocasiones inspeccionar software real de trabajo (cada dos semanas o un mes).
- El negocio fija las prioridades. Los equipos se auto-organizan a fin de determinar la mejor manera de entregar las funcionalidades de más alta prioridad.
- Cada dos semanas o un mes, cualquiera puede ver el software real funcionando y decidir si liberarlo o seguir mejorándolo en otro sprint.



26

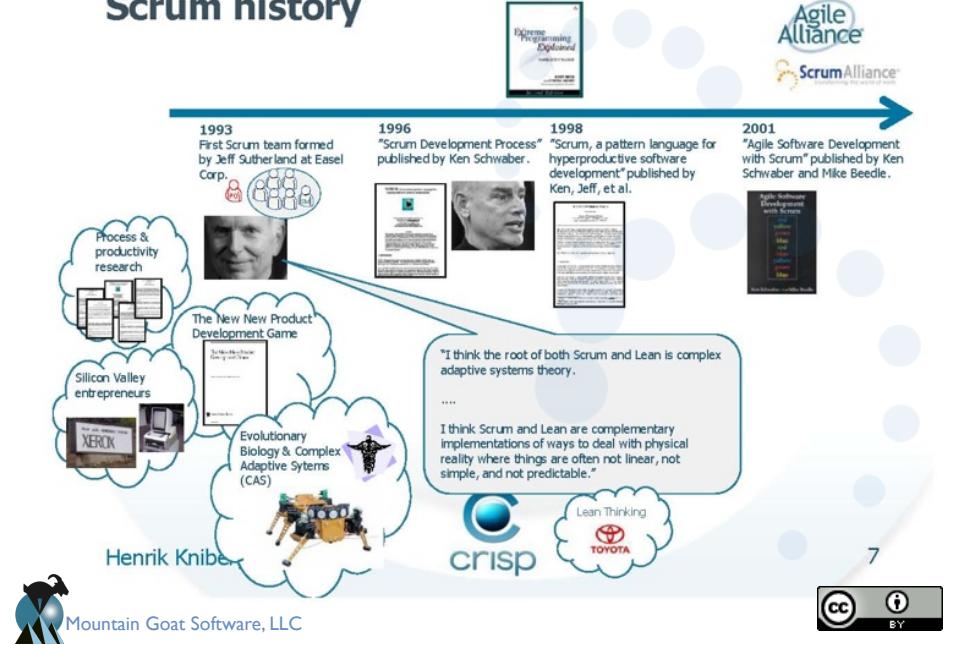
Orígenes de Scrum

- Jeff Sutherland
 - Scrums iniciales en Easel Corp en 1993
 - IDX 500 personas haciendo Scrum
- Ken Schwaber
 - ADM
 - Se presenta Scrum en OOPSLA 96 con Sutherland
 - Autor de tres libros sobre Scrum
- Mike Beedle
 - Patrones Scrum en PLOPD4
- Ken Schwaber and Mike Cohn
 - Fundaron conjuntamente la Scrum Alliance en 2002, inicialmente dentro de la Agile Alliance



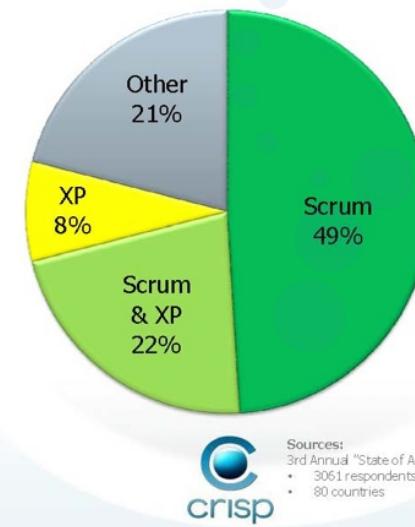
28

Scrum history



29

Scrum is the most popular agile process



Sources:
3rd Annual "State of Agile Development" Survey June-July 2008
• 3051 respondents
• 80 countries



11

30

Scrum ha sido utilizado para:

- Microsoft
- Yahoo
- Google
- Electronic Arts
- High Moon Studios
- Lockheed Martin
- Philips
- Siemens
- Nokia
- Capital One
- BBC
- Intuit
- Intuit
- Nielsen Media
- First American Real Estate
- BMC Software
- Ipswich
- John Deere
- Lexis Nexis
- Sabre
- Salesforce.com
- Time Warner
- Turner Broadcasting
- Oce

- Software comercial
- Desarrollos internos
- Desarrollos bajo Contrato
- Proyectos Fixed-price
- Aplicaciones Financieras
- Aplicaciones certificadas ISO 9001
- Sistemas Embebidos
- Sistemas con requisitos 7x24 y 99.999% de disponibilidad
- Joint Strike Fighter

- Desarrollo de video juegos
- Sistemas críticos de soporte vital, aprobados por la FDA
- Software de control satelital
- Sitios Web
- Software para PDAs
- Teléfonos portátiles
- Aplicaciones de Network switching
- Aplicaciones de ISV
- Algunas de las más grandes aplicaciones en uso

31

4 pillars of Scrum



17

33



Empirical process control



Many small teams	Few large teams
Little testing	Lots of testing
Many small backlog items	Few large backlog items
Short sprints	Long sprints
Short planning horizon	Long planning horizon
... etc etc ...

Henrik Kniberg

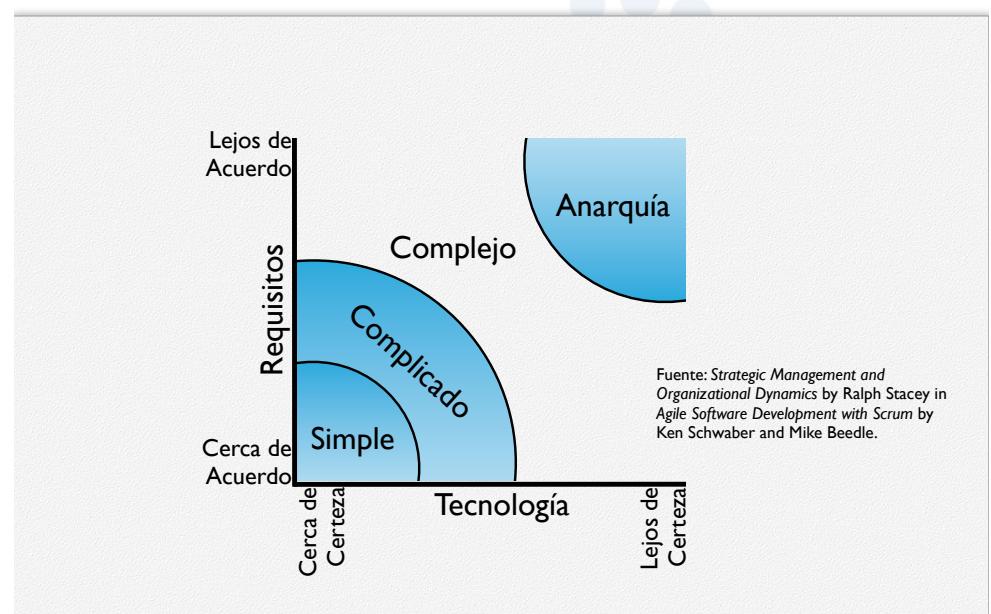
crisp

44



34

Nivel de ruido de un proyecto



Características

- Equipos auto-organizados
- El producto avanza en una serie de “Sprints” de dos semanas a un mes de duración
- Los requisitos son capturados como elementos de una lista de “Product Backlog”
- No hay prácticas de ingeniería prescritas
- Se basa en una serie de reglas que permiten crear un entorno ágil para la entrega de proyectos
- Uno de los “procesos ágiles”



35

36

Scrum

Return
Gift wrap
Cancel
Product Backlog



37



Scrum

Sprint
2-4 semanas

Return
Gift wrap
Cancel
Product Backlog



37



Scrum

Objetivo del Sprint
Gift wrap
Cancel
Product Backlog



37



Scrum

Sprint
2-4 semanas

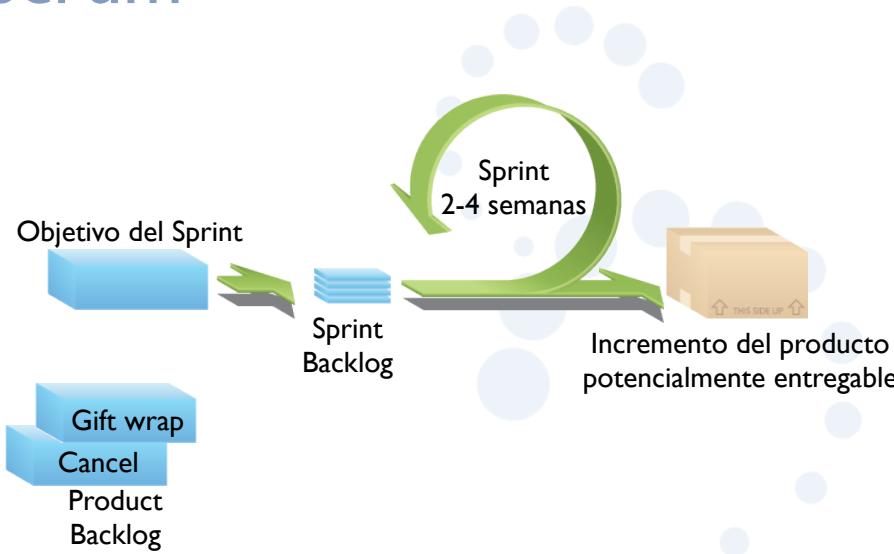
Objetivo del Sprint
Sprint Backlog
Gift wrap
Cancel
Product Backlog



37

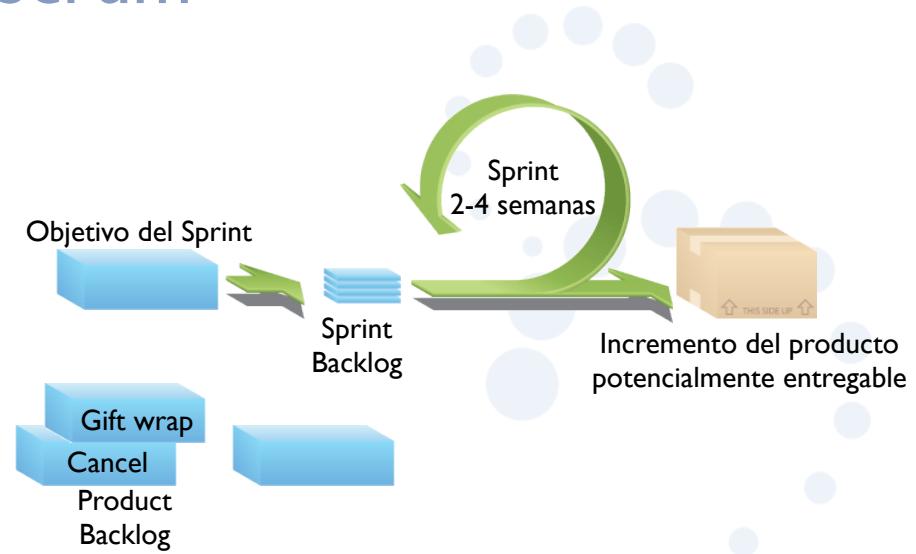


Scrum



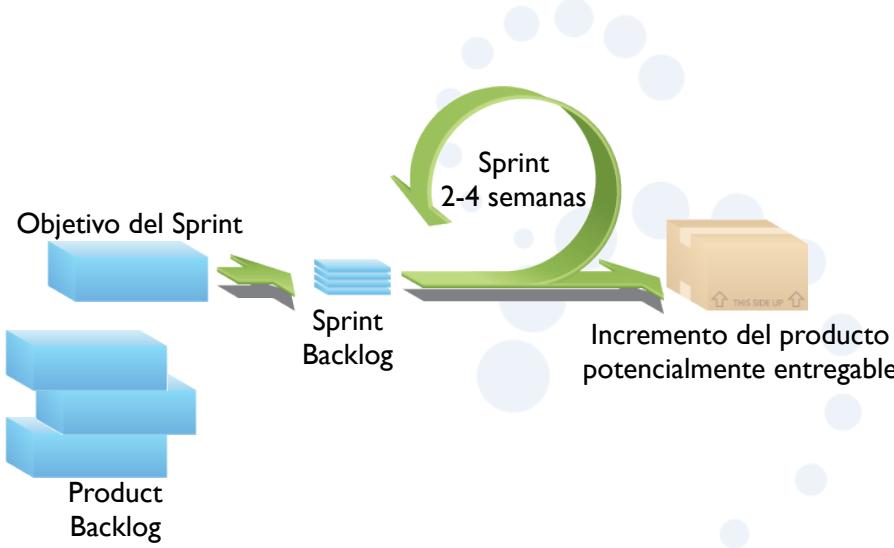
37

Scrum



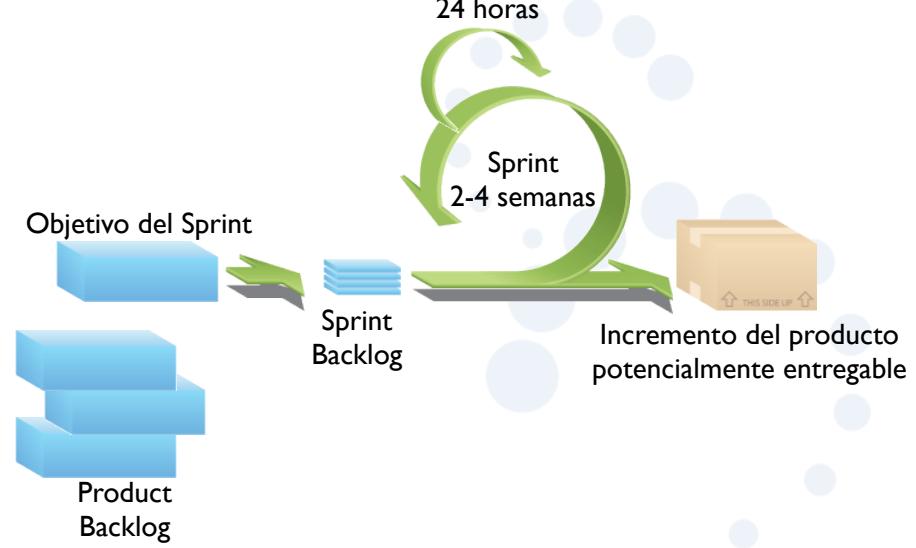
37

Scrum



37

Scrum



37

Poniendo todo junto

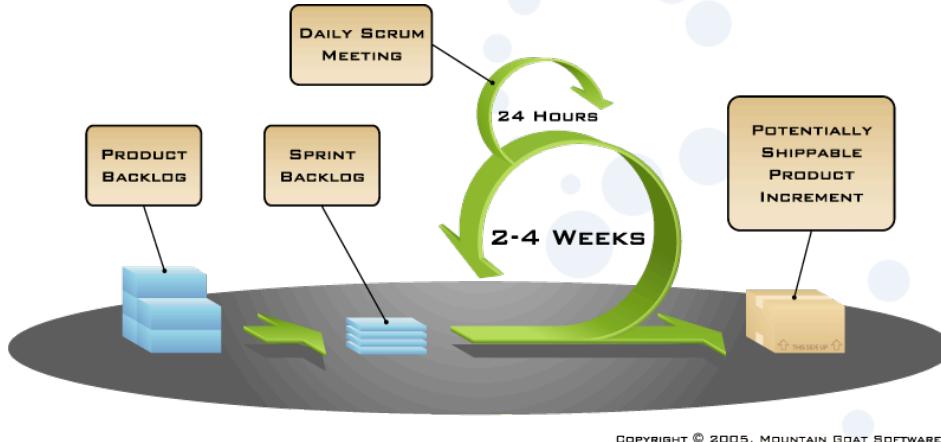


Imagen disponible en
www.mountaingoatsoftware.com/scrum



38

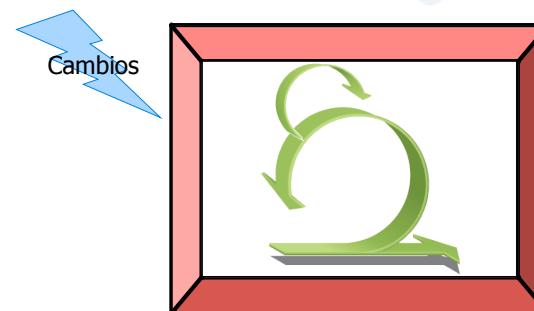
Sprints

- En Scrum los proyectos avanzan en una serie de “Sprints”
 - Análogo a las iteraciones en XP
- La duración típica es 2–4 semanas o a lo sumo un mes calendario
- La duración constante conduce a un mejor ritmo
- El producto es diseñado, codificado y testeado durante el Sprint



39

No hay cambios en un sprint



- Planee la duración del sprint en torno a cuánto tiempo usted puede comprometerse a mantener los cambios fuera del sprint



40



41

Scrum Framework

Roles

- Product owner
- ScrumMaster
- Team

Reuniones

- Release planning
- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artefactos

- Product backlog
- Sprint backlog
- Burndown charts

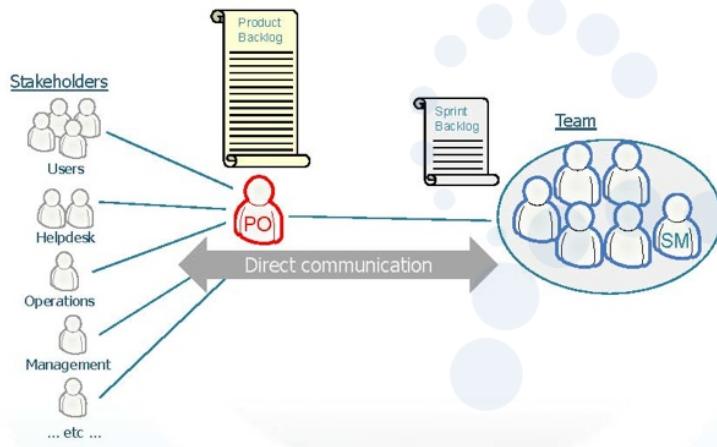


Mountain Goat Software, LLC



42

Scrum overview – structure



2009-08-23

Henrik Kniberg



18

Scrum framework

Roles

- Product owner
- ScrumMaster
- Team

- Release planning
- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum

Artefactos

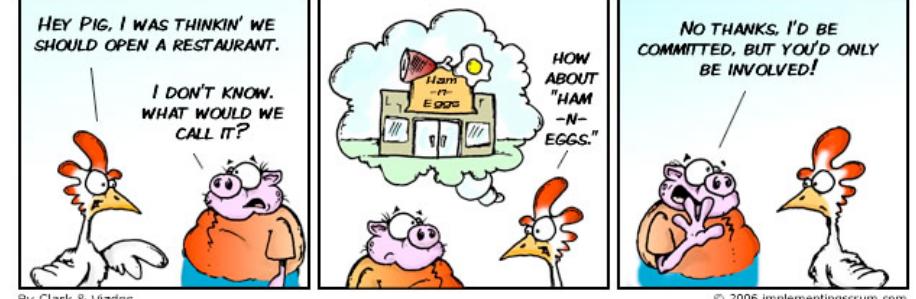
- Product backlog
- Sprint backlog
- Burndown charts



Mountain Goat Software, LLC



43



Mountain Goat Software, LLC



44



45



By Clark & Vizdos

- Las gallinas sólo están involucradas, pero los cerdos están realmente comprometidos
 - El Dueño del Producto es el cerdo del Product Backlog
 - El Scrum Master es el cerdo del proceso Scrum
 - El Equipo es el cerdo del trabajo del Sprint

Las gallinas no pueden decirle a los cerdos
cómo tienen que hacer su trabajo



45

El ScrumMaster

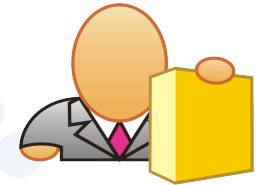


- Representa a la gestión del proyecto
- Responsable de promover los valores y prácticas de Scrum
- Elimina impedimentos
- Se asegura de que el equipo es completamente funcional y productivo
- Permite la estrecha cooperación en todos los roles y funciones
- Escudo del equipo de interferencias externas
- Debe permitir que el equipo de desarrollo se auto-organice



47

Product Owner



- Define las funcionalidades del producto
- Decide sobre las fechas y contenidos de los releases
- Es responsable por la rentabilidad del producto
- Prioriza funcionalidades de acuerdo al valor del mercado/negocio
- Ajusta el alcance y la prioridad de cada funcionalidad en cada sprint si es necesario
- Acepta o rechaza los resultados del trabajo del equipo



46

El Team



- Típicamente de 5 a 9 personas
- Multi-funcional:
 - Programadores, testers, analistas, diseñadores, etc.
- Cross-functional
 - Algunos miembros están especializados, pero las habilidades que comparte el equipo tienden a ser más importantes para conseguir el objetivo del sprint que las que no comparten
- Los miembros deben ser full-time
 - Puede haber excepciones (Ej.: Infraestructura, SCM, etc.)
- Los equipos son auto-organizativos
 - Idealmente, no existen títulos pero a veces se utilizan de acuerdo a la organización



48

Scrum Framework

Roles

- Product owner
- ScrumMaster
- Team

Reuniones

- Release planning
- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artefactos

- Product backlog
- Sprint backlog
- Burndown charts



49

Timeboxing

Plan
(doomed to fail, but we don't know it yet)



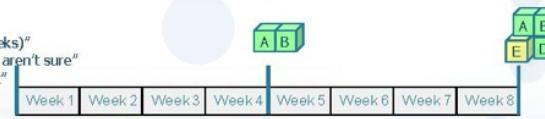
Traditional scenario

"We will deliver ABCD in 4 weeks"



Agile scenario

"We always deliver something every sprint (4 weeks)"
"We think we can finish ABCD in 1 sprint, but we aren't sure"
"We always deliver the most important items first"



Henrik Kniberg



29



51

Release Planning meeting

Propósito

- Establecer el objetivo del release (lanzamiento): alcance de la próxima versión del producto
 - Obtener una primera versión del Product Backlog: formulación poco detallada de las funcionalidades fundamentales
 - Establecer fecha de entrega y coste release
- Es opcional. Como se hará planificación just-in-time durante el sprint, puede y debe dedicarse poco esfuerzo a la planificación del release: 15-20% tiempo enfoques tradicionales



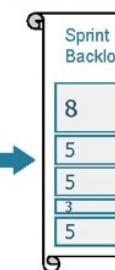
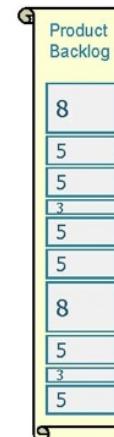
Release planning
meeting



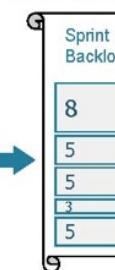
50

Measuring velocity

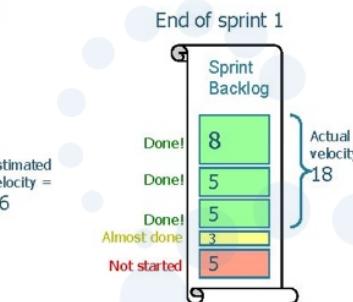
Beginning of sprint 1



Estimated velocity = 26



Actual velocity = 18



End of sprint 1



30



52

Agile estimating strategy

As a buyer
I want to save
car so that I can
shopping later

2 > 1

- **Don't estimate time.**
 - Estimate *relative size* of stories.
 - Measure velocity per sprint.
 - Derive release plan.
- **Estimates done by the people who are going to do the work.**
 - Not by the people who want the work done.
- **Estimate continuously during project, not all up front.**
- **Prefer verbal communication over detailed, written specifications.**
- **Avoid false precision**
 - Better to be roughly right than precisely wrong

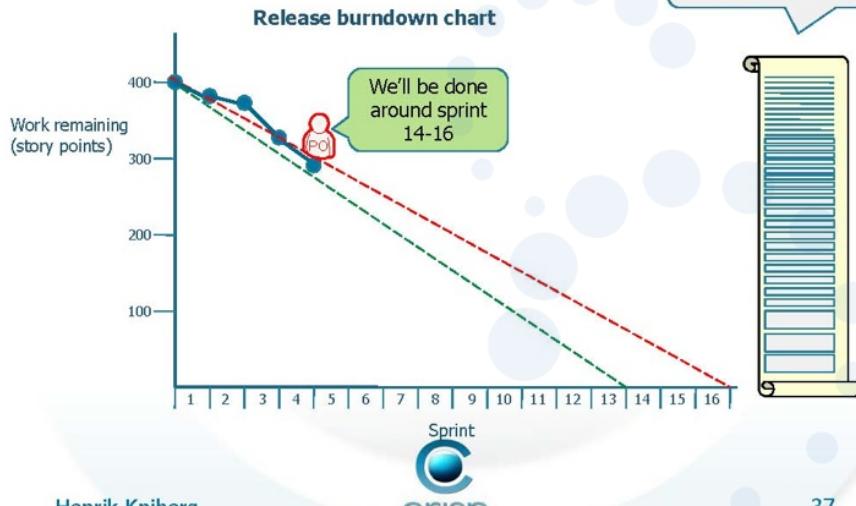


crisp



53

Release planning – fixed scope



37

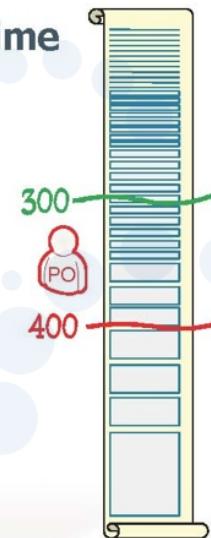


55

Release planning – fixed time

- Today is Aug 6
- Sprint length = 2 weeks
- Velocity = 30 - 40

What will be done
by X-mas?
(10 sprints)



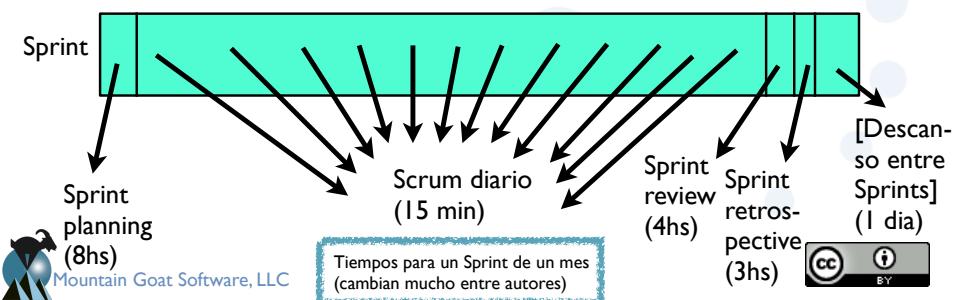
- 3 roles**
- Product owner
 - Scrum master
 - Team
- 3 artifacts**
- Product backlog
 - Sprint backlog
 - Sprint burndown
- 3 activities**
- Sprint planning
 - Daily scrum
 - Sprint review
 - Demo
 - Retrospective



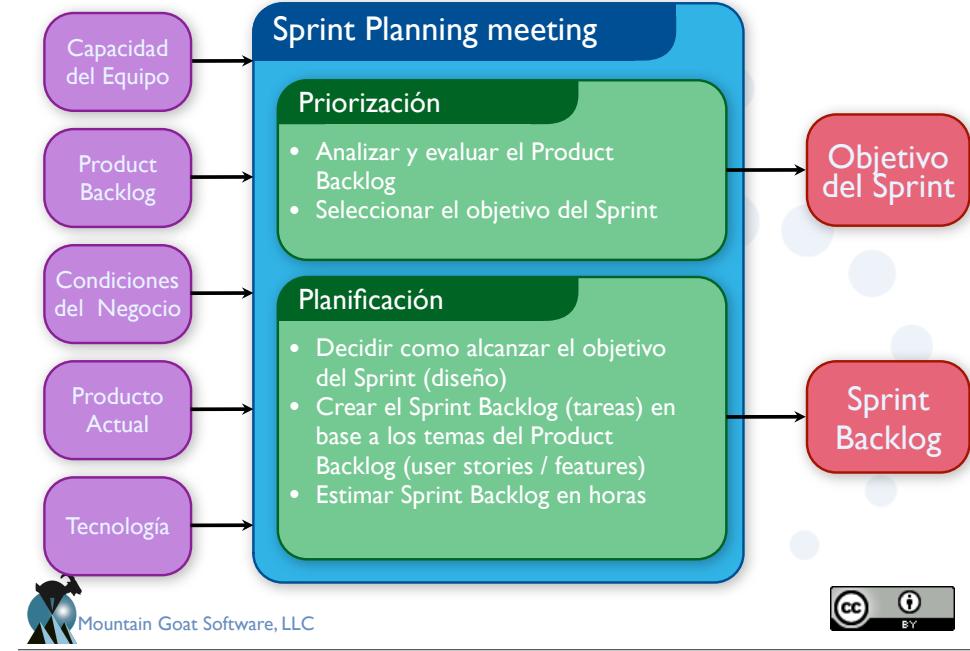
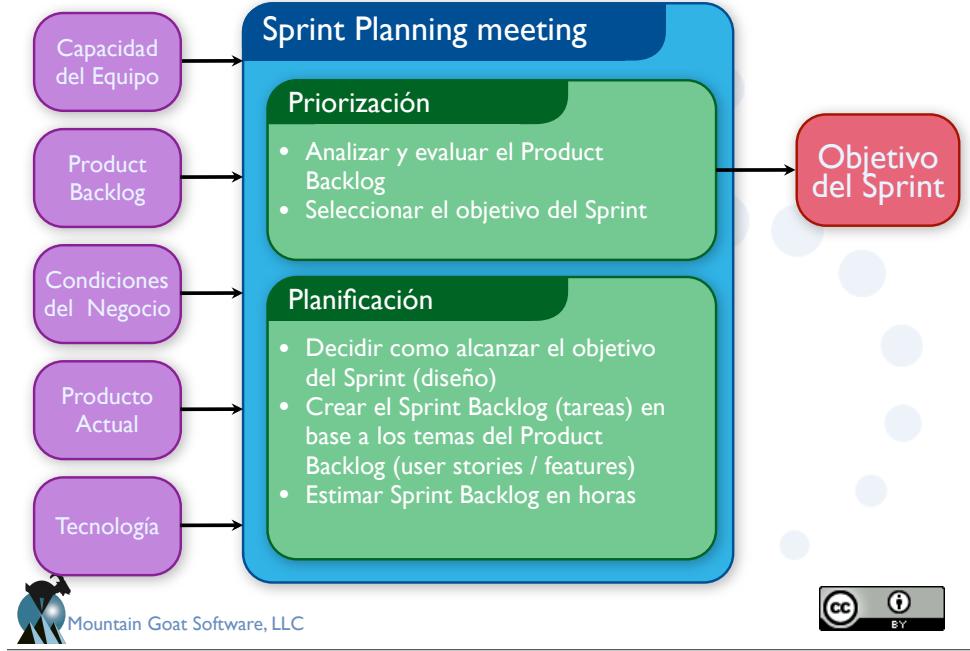
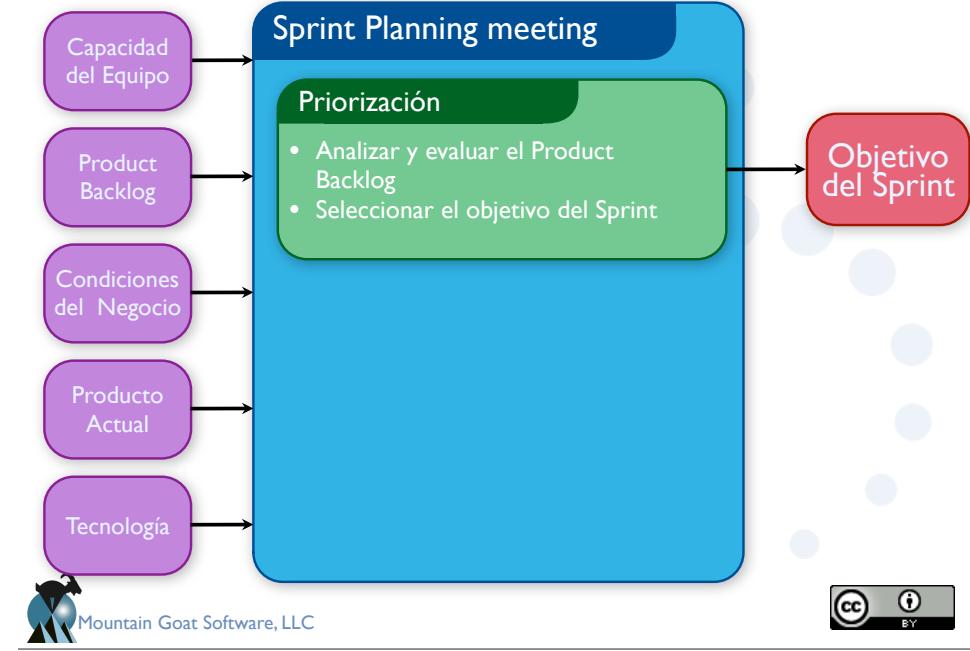
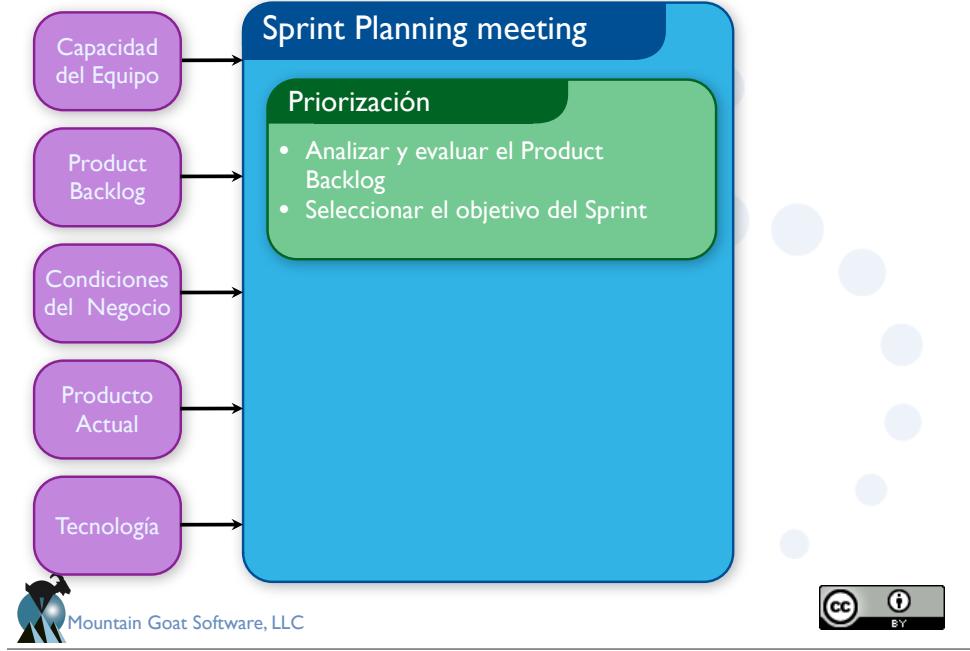
54

Sprint

- Un Sprint es una iteración
 - Todos los Sprints duran lo mismo
 - Durante un Sprint el objetivo del Sprint y la composición del equipo quedan fijos
- Cada Sprint debe producir un incremento del producto = una porción del producto potencialmente entregable



56



Planificación del Sprint

- El equipo selecciona las historias del Product Backlog que pueden comprometerse a completar
- Se crea el Sprint Backlog
 - Se identifican tareas y cada una es estimada (1-16 horas)
 - Realizado colaborativamente, no solo por el ScrumMaster
- El diseño de Alto Nivel es considerado

Historia (=elemento) del Product Backlog

COMO planificador
de vacaciones, YO
QUIERO ver fotos
de los hoteles.

Tareas correspondientes a la historia

- Codificar la capa intermedia (8 hs)
- Codificar la interfaz de usuario (4)
- Escribir los test fixtures (4)
- Codificar la clase foo (6)
- Actualizar test de performance (4)



58

Daily Scrum

- Parámetros
 - Diaria
 - Dura 15 minutos
 - Parados
- No para la solución de problemas
 - Todo el mundo está invitado
 - Sólo los miembros del equipo, ScrumMaster y Product Owner, pueden hablar
 - Ayuda a evitar otras reuniones innecesarias



59

Todos responden 3 preguntas

1
¿Qué hiciste ayer?

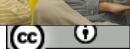
2
¿Qué vas a hacer hoy?

3
¿Hay obstáculos en tu camino?

- **No** es dar un status report al Scrum Master
- Se trata de compromisos delante de pares

Sprint review

- Tras finalizar el sprint, el equipo presenta lo realizado durante el sprint al resto de la empresa
- Normalmente adopta la forma de una demo de las nuevas características o la arquitectura subyacente
- Informal
 - Regla de 2 hs preparación
 - No usar diapositivas
- Todo el equipo participa
- Se invita a todo el mundo
- También llamada Sprint Demo



Sprint retrospective

- Después del Sprint Review, se echa un vistazo a lo que funciona y lo que no respecto al proceso. Es un postmortem del Sprint
- Normalmente 15 a 30 minutos
- Se realiza luego de cada sprint
- Todo el equipo participa
 - ScrumMaster
 - Product owner
 - Equipo
 - Posiblemente clientes y otros



Scrum framework

Roles

- Product owner
- ScrumMaster
- Team

Reuniones

- Release planning
- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum

Artefactos

- Product backlog
- Sprint backlog
- Burndown charts



62

Start / Stop / Continue

- Todo el equipo se reúne y discute lo que les gustaría:

Comenzar a hacer

Dejar de hacer

Continuar haciendo

Esto es sólo una
de las muchas
maneras de
hacer una
retrospectiva.



63

Product Backlog

- Los requisitos
- Una lista de todos los trabajos deseados en el proyecto
- Idealmente cada tema tiene valor para el usuarios o el cliente
- Priorizada por el Product Owner
- Repriorizada al comienzo de cada Sprint



Este es el
product backlog



64

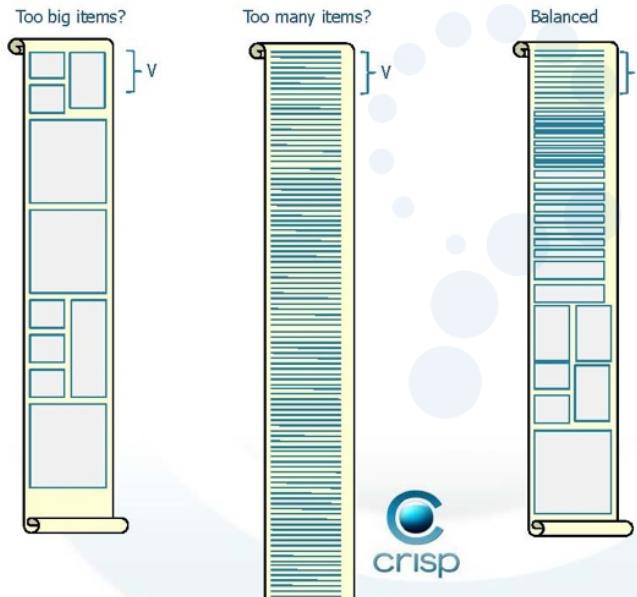
Ejemplo de Product Backlog

Backlog item	Estimación
Permitir que un invitado a hacer una reserva.	3
Como invitado, quiero cancelar una reserva.	5
Como invitado, quiero cambiar las fechas de una reserva.	3
Como un empleado de hotel, puedo ejecutar informes de los ingresos por habitación disponible	8
Mejorar el manejo de excepciones	8
...	30
...	50

Mountain Goat Software, LLC

66

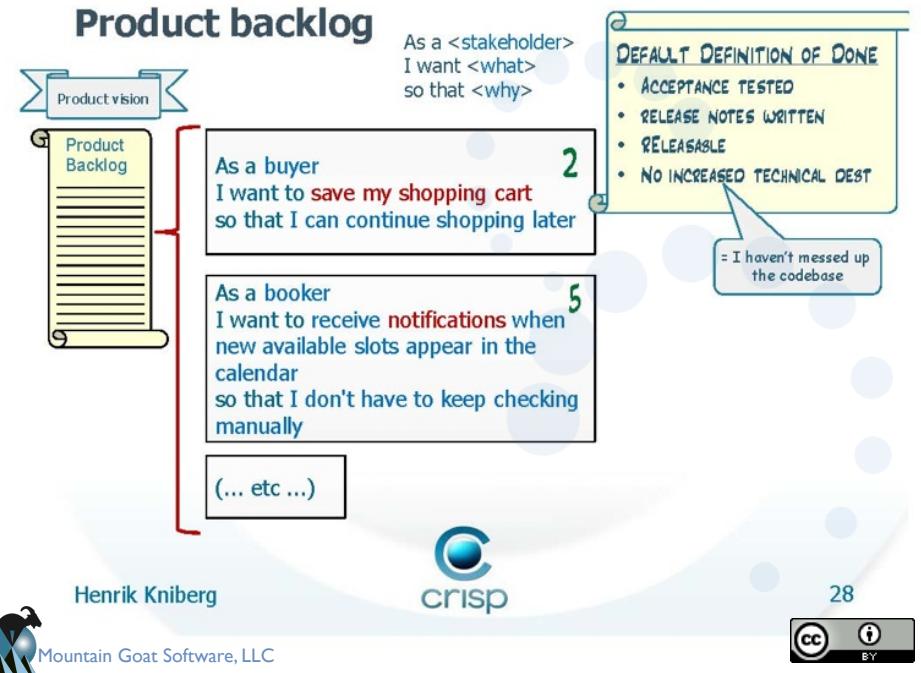
Balance the product backlog



Mountain Goat Software, LLC

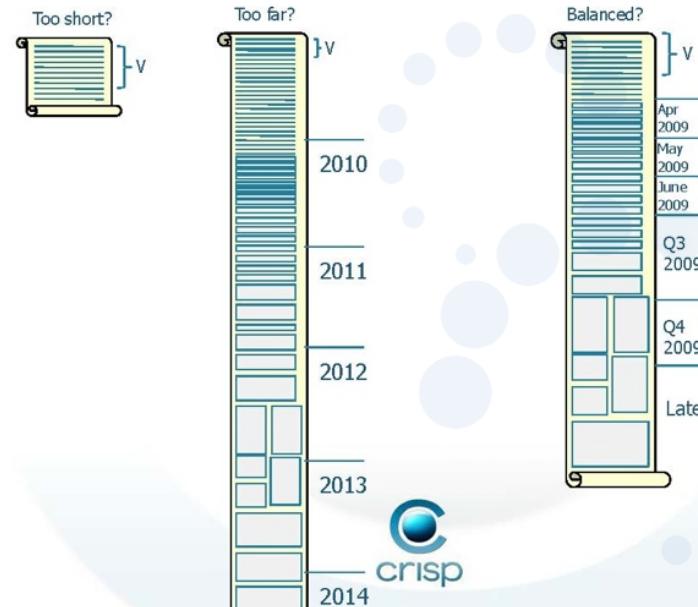
38

Product backlog



67

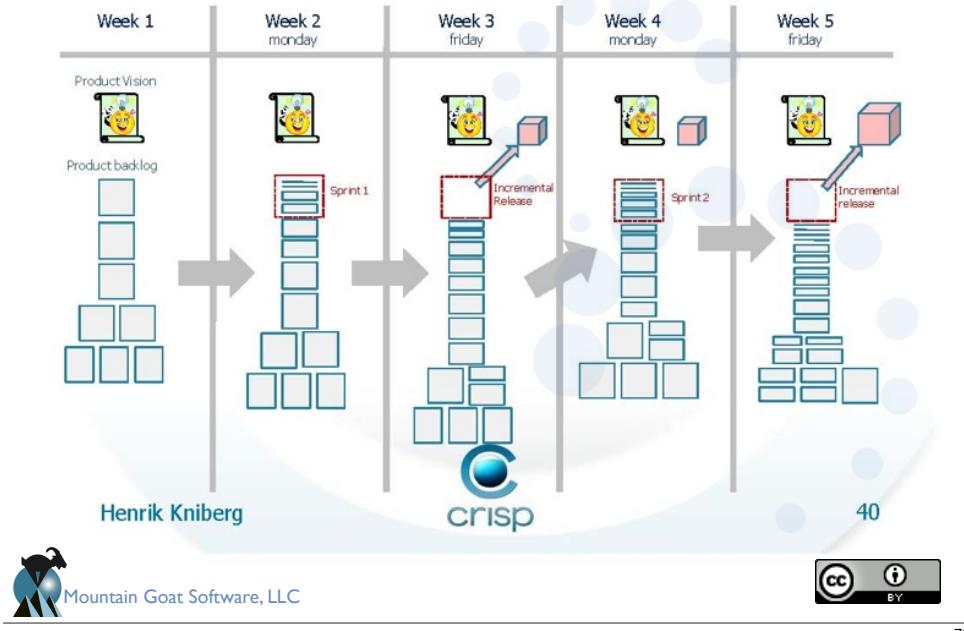
Choose the right planning horizon



Mountain Goat Software, LLC

39

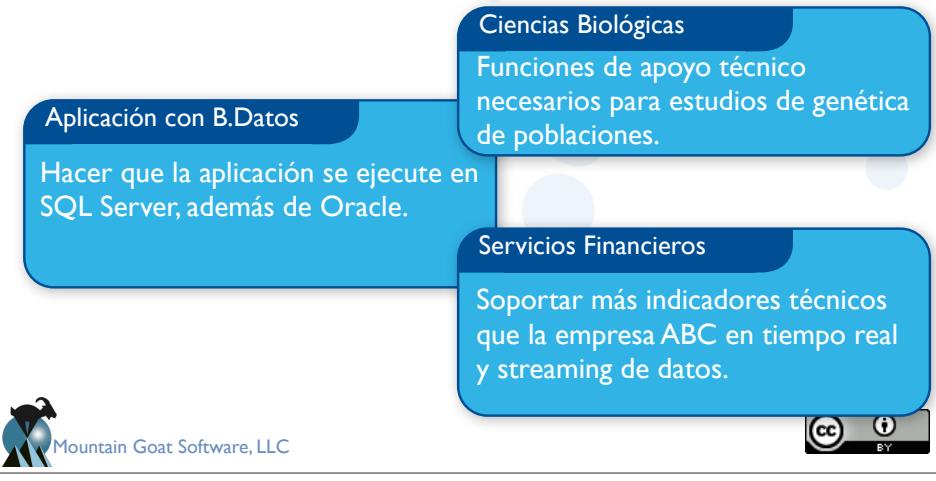
Example: evolution of a product backlog



70

El objetivo del Sprint

- Una breve declaración de cual será el foco del trabajo durante el sprint



71

Ejemplo de Sprint Backlog

- Los individuos eligen las tareas
- El trabajo nunca es asignado
- La estimación del trabajo restante es actualizada diariamente
- Cualquier miembro del equipo puede añadir, borrar o cambiar el Sprint Backlog
- El trabajo para el Sprint emerge
- Si el trabajo no está claro, definir un tema del Sprint Backlog con una mayor cantidad de tiempo y subdividirla luego.
- Actualizar el trabajo restante a medida de que más se conoce

Tareas	L	M	M	J	V
Codificar UI	8				
Codificar negocio	16				
Testear negocio	8				
Escribir ayuda online	12				
Escribir la clase foo	8				

72



73

Ejemplo de Sprint Backlog

Tareas	L	M	M	J	V
Codificar UI	8	4			
Codificar negocio	16	12			
Testear negocio	8	16			
Escribir ayuda online	12				
Escribir la clase foo	8	8			

Ejemplo de Sprint Backlog

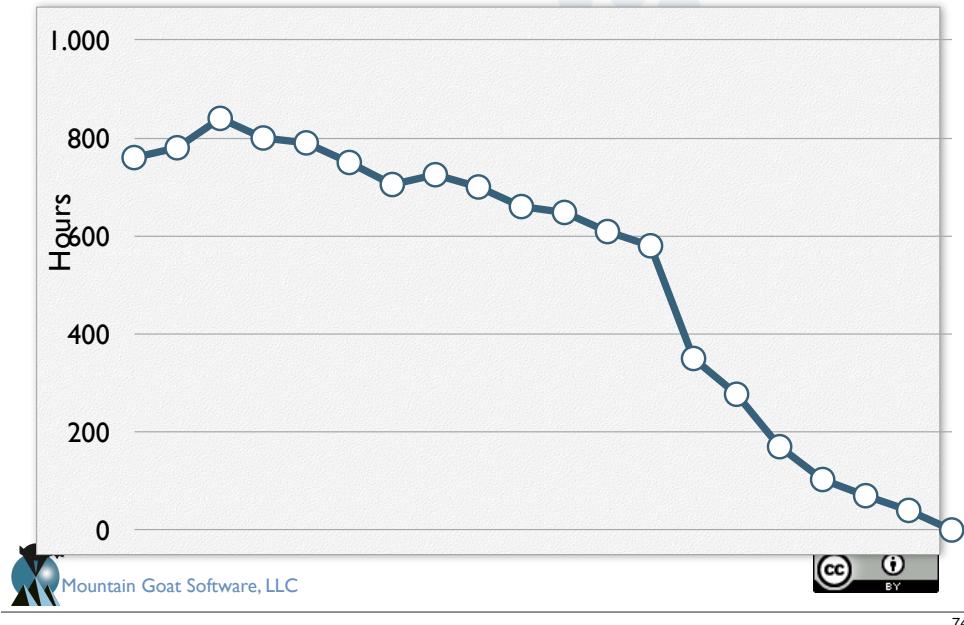
Tareas	L	M	M	J	V
Codificar UI	8	4	8		
Codificar negocio	16	12	10	4	
Testear negocio	8	16	16	11	
Escribir ayuda online	12				
Escribir la clase foo	8	8	8	8	
Agregar error logging					8

Ejemplo de Sprint Backlog

Tareas	L	M	M	J	V
Codificar UI	8	4	8		
Codificar negocio	16	12	10	4	
Testear negocio	8	16	16	11	
Escribir ayuda online	12				
Escribir la clase foo	8	8	8	8	
Agregar error logging			8	4	

Tareas	L	M	M	J	V
Codificar UI	8	4	8		
Codificar negocio	16	12	10	4	
Testear negocio	8	16	16	11	8
Escribir ayuda online	12				
Escribir la clase foo	8	8	8	8	8
Agregar error logging				8	4

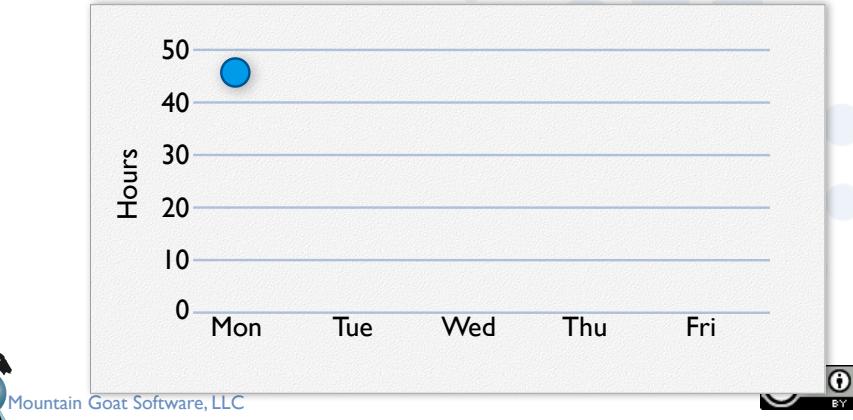
Un Sprint Burndown Chart



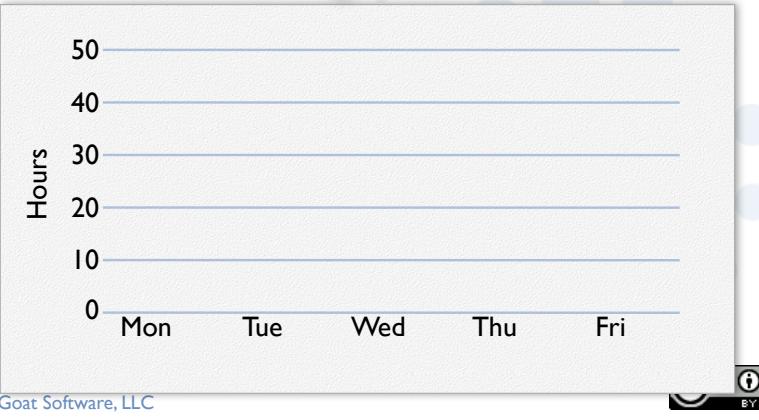
Tareas	L	M	M	J	V
Codificar UI	8				
Codificar Negocio	16				
Testear Negocio	8				
Escribir ayuda online	12				



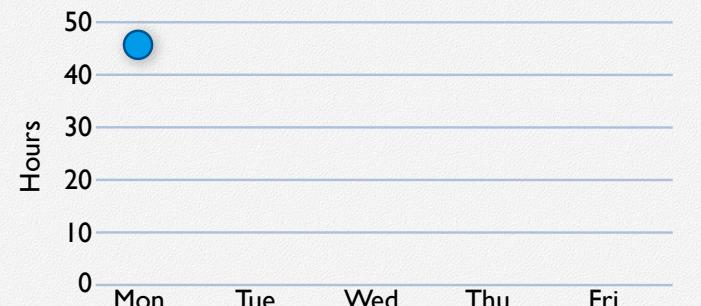
Tareas	L	M	M	J	V
Codificar UI	8				
Codificar Negocio	16				
Testear Negocio	8				
Escribir ayuda online	12				



Tareas	L	M	M	J	V
Codificar UI	8				
Codificar Negocio	16				
Testear Negocio	8				
Escribir ayuda online	12				



Tareas	L	M	M	J	V
Codificar UI	8	4			
Codificar Negocio	16	12			
Testear Negocio	8	16			
Escribir ayuda online	12				



75

Tareas	L	M	M	J	V
Codificar UI	8	4			
Codificar Negocio	16	12			
Testear Negocio	8	16			
Escribir ayuda online	12				



75

Tareas	L	M	M	J	V
Codificar UI	8	4			
Codificar Negocio	16	12			
Testear Negocio	8	16			
Escribir ayuda online	12				



75

Tareas	L	M	M	J	V
Codificar UI	8	4	8		
Codificar Negocio	16	12	10	7	
Testear Negocio	8	16	16	11	8
Escribir ayuda online	12				



75

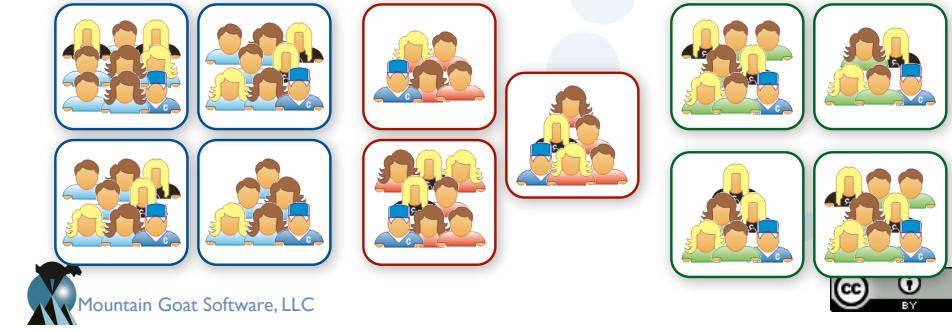
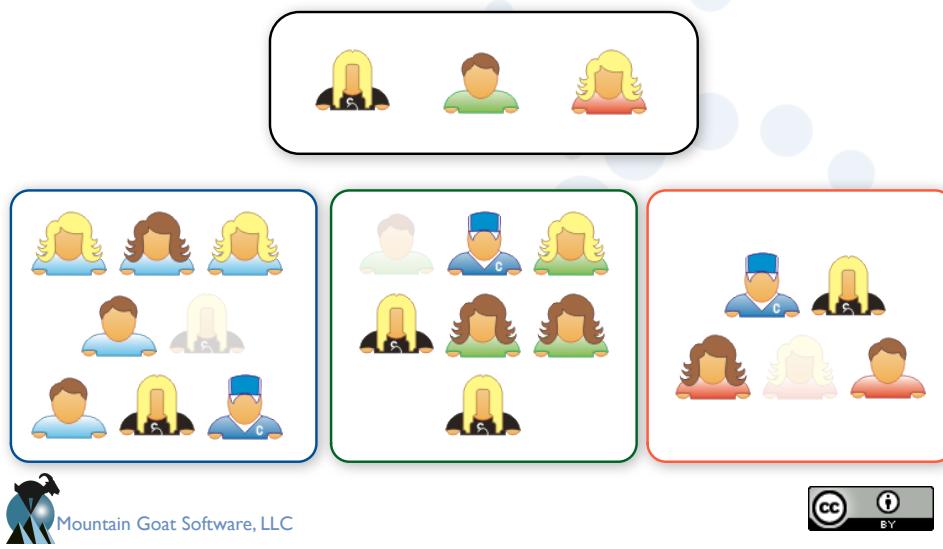
Escalabilidad

- Normalmente los equipos son de 7 ± 2 personas
 - La escalabilidad proviene de equipos de equipos
- Factores a tener cuenta
 - Tipo de aplicación
 - Tamaño del equipo
 - Dispersión del equipo
 - Duración del proyecto
- Scrum se ha utilizado en múltiples proyectos de más de 500 personas

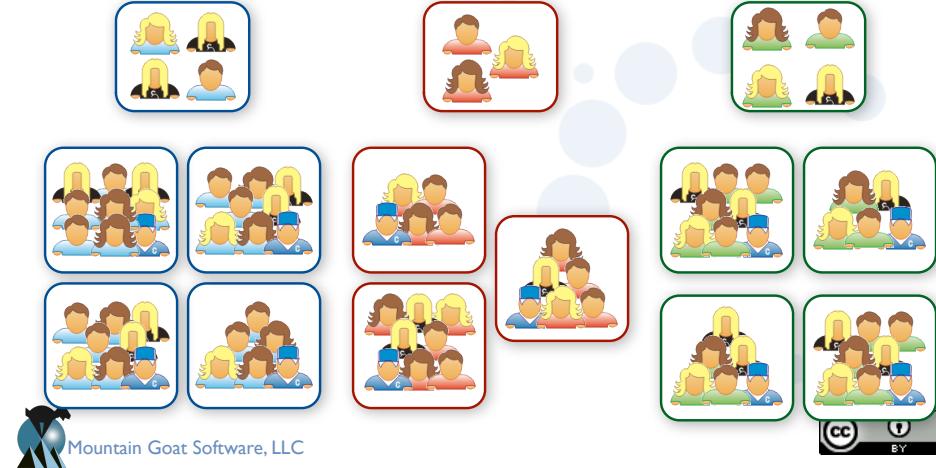
Expansión a través de Scrum de scrums



Scrum de scrums de scrums



Scrum de scrums de scrums



78

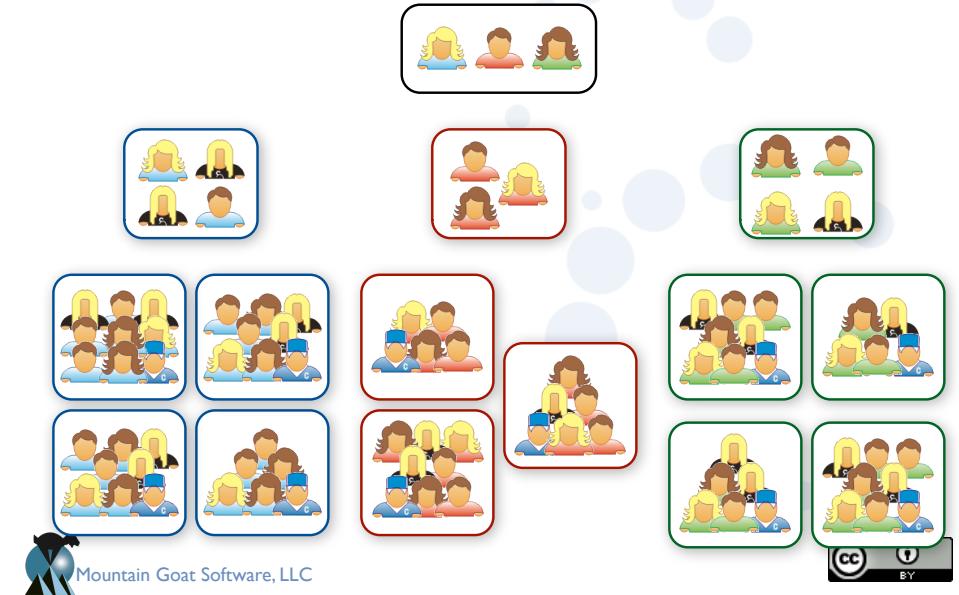
Beware



51

Henrik Kniberg

Scrum de scrums de scrums



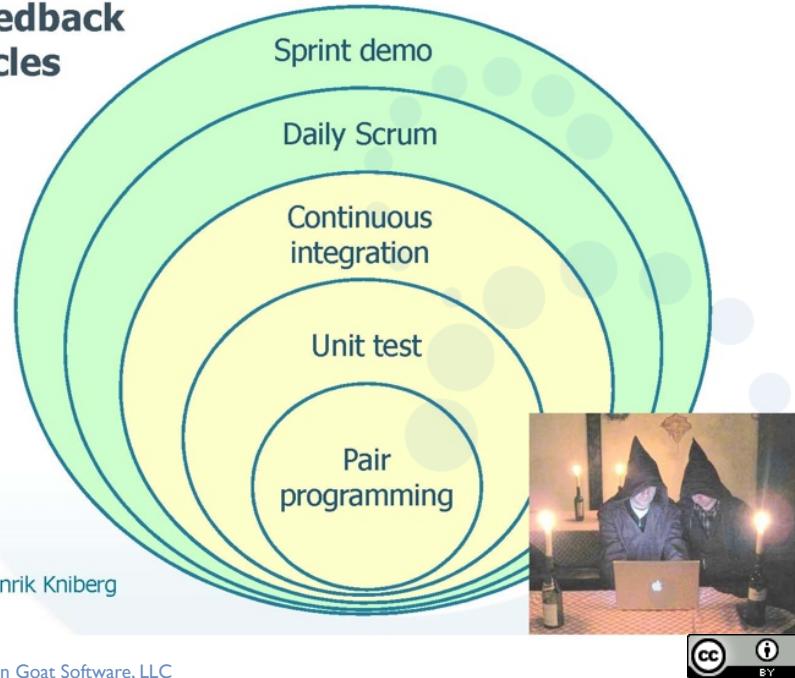
78

Scrum & XP



2007-09-28
Henrik Kniberg

Feedback cycles



Henrik Kniberg



Mountain Goat Software, LLC



81

Plan-driven vs. Agile

- Los partidarios de las metodologías ágiles sostienen que éstas pueden utilizarse con éxito en todo tipo de proyectos y organizaciones, como hemos visto en los ejemplos anteriores
- Los partidarios de las metodologías clásicas afirman que Agile es apropiado sólo para ciertos escenarios
 - Agile ha tenido éxito en desarrollos de productos
 - De tamaño medio o pequeño
 - Tb para productos de uso interno en la compañía desarrolladora, y que por tanto cuentan con la colaboración del cliente y no tienen que cumplir con regulaciones externas



Mountain Goat Software, LLC



82

Plan-driven vs. Agile

- Las metodologías ágiles necesitan ser modificadas significativamente para ser utilizadas en proyectos grandes o para poder dar las garantías de seguridad necesarias para sistemas críticos
- Las prácticas de Agile a veces son difíciles de aplicar en la práctica
 - Participación del cliente en el desarrollo: es difícil encontrar una persona con la visión comercial para representar a los stakeholders y que a la vez tenga suficiente tiempo para colaborar con los desarrolladores
 - Los miembros del equipo pueden no tener personalidades adecuadas para el compromiso intenso que es típico de las metodologías ágiles



Mountain Goat Software, LLC



83



Mountain Goat Software, LLC



84

Plan-driven vs. Agile

- Se cuestiona la efectividad de las metodologías ágiles para el mantenimiento de SW, ya que es difícil convencer al cliente de que siga involucrado durante el mantenimiento, y la participación del cliente es una de las bases de Agile
- Es difícil mantener unido al equipo de desarrollo a lo largo del tiempo y en particular durante el mantenimiento. Como Agile se basa en el conocimiento y la comprensión del sistema por parte de los miembros del equipo, más que en la documentación exhaustiva, la descomposición del equipo de desarrollo puede tener un gran impacto en la capacidad de manejo del sistema



¿Plan-driven o Agile?

- Para decidir entre estos dos enfoques debemos plantearnos las siguientes preguntas (Sommerville)
 - ¿Es importante disponer de una especificación y diseño muy detallados antes de empezar la implementación? -> Plan-driven
 - ¿Es realista una estrategia de entrega incremental, para obtener feedback de los usuarios rápidamente? -> Agile
 - ¿Es el sistema a desarrollar muy grande?: Agile funciona mejor con equipos pequeños o medianos, plan-driven puede manejar equipos grandes



¿Plan-driven o Agile?

- Un proceso plan-driven puede soportar el desarrollo y la entrega incremental
 - Se pueden capturar los requisitos y entonces planear las fases de diseño e implementación como una serie de incrementos
- Un proceso ágil no tienen porqué estar centrado en el código de forma inevitable y también puede producir alguna documentación de diseño
 - A través de iteraciones “spike” usadas para generar documentación
- En la práctica muchos proyectos SW incluyen prácticas de ambos enfoques



¿Plan-driven o Agile?

- ¿Cuál es la vida esperada del sistema?: en sistemas con fase de mantenimiento largo se necesita una buena documentación que comunique las intenciones originales del sistema -> Plan-driven. Sin embargo los partidarios de Agile afirman que la documentación suele estar desincronizada con el sistema y acaba siendo poco útil para el mantenimiento de sistemas a largo plazo
- ¿Cómo está organizado el equipo de desarrollo? Si está disperso geográficamente o con outsourcing Agile funcionará peor pq la comunicación personal se dificulta
- ¿Problemas culturales? Las organizaciones de ingeniería tradicionales tienen muy interiorizado el modelo plan-driven en su cultura de empresa



¿Plan-driven o Agile?

- Habilidades del equipo de desarrollo: Agile se basa más en las habilidades individuales y necesita personal más cualificado
- ¿Está el sistema sujeto a alguna regulación externa? La documentación detallada generada por los procesos plan-driven suele ser útil en los procesos certificadores
- Sin embargo muchos profesionales que practican las metodologías ágiles afirman que estas limitaciones se pueden superar, y proponen formas para utilizar Agile en proyectos grandes y de larga duración, con contratos a precio cerrado, personal distribuido, y calidad crítica y certificada
- La cuestión Plan-driven vs. Agile es todavía un tema de debate abierto en la comunidad de desarrollo SW

