Contenido:

- Introducción. Manifiesto ágil
- Ideas sobre XP
- SCRUM
- Fuentes Bibliográficas

A modo de introducción

El esquema tradicional del desarrollo del software ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general se precisa bastante gestión del proceso.

Pero este enfoque no resulta ser el más adecuado para muchos proyectos actuales en los que el entorno del sistema es muy cambiante y en donde se exige reducir drásticamente los tiempos de desarrollo manteniendo una calidad elevada.

Ante las dificultades para utilizar metodologías tradicionales con estas restricciones de tiempo y flexibilidad, muchos equipos de desarrollo optan por prescindir del “buen hacer” de la ingeniería del software, asumiendo el riesgo que ello conlleva. En este escenario, las metodologías ágiles emergen como una posible respuesta para llenar ese vacío metodológico.

Por estar especialmente orientadas para proyectos pequeños, las metodologías ágiles constituyen una solución a medida para ese entorno, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto.

Fuente: Curso de desarrollo ágil. INTECO, 2009.

Manifiesto Ágil

"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."

Fuente: http://es.wikipedia.org/wiki/Manifiesto_%C3%A1gil

El problema de los riesgos

Ejemplos habituales de riesgos:

- Retrasos de planificación
- Proyecto cancelado
- El sistema se deteriora
- Tasa elevada de defectos
- Requisitos mal comprendidos
- Cambios en el negocio
- Falsa riqueza de las características (del software)
- Cambios de personal

Corresponden a muchos de la lista de los 10 riesgos más importantes de Boehm (1991).

Fuente: Kent Beck. Una explicación de la programación extrema. Addison-Wesley 2002.

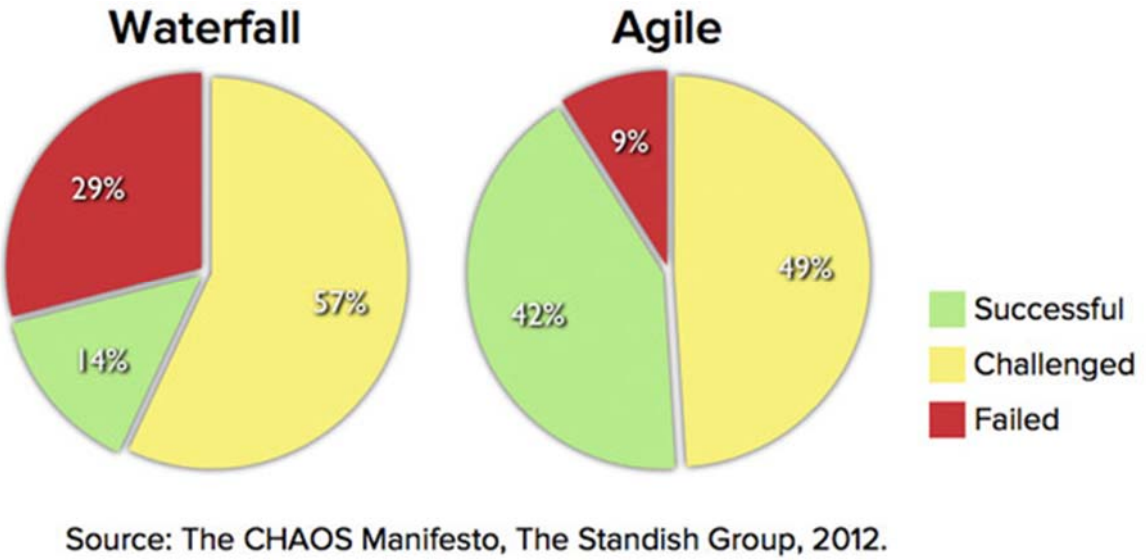
Formas de abordar la economía de un proyecto software

- Opción de abandonar
- Opción de cambiar
- Opción de aplazar
- Opción de crecer

Factores implicados en la toma de decisión:

- Cantidad de inversión que se precisa para conseguir la opción.
- El precio al que puedes comprar otra cosa si procedes con la opción
- El valor actual de la otra cosa
- La cantidad de tiempo en el que puedes ejercer las opciones
- La incertidumbre en el valor final del precio.

Fuente: Kent Beck. Una explicación de la programación extrema. Addison-Wesley 2002.



Source: The CHAOS Manifesto, The Standish Group, 2012.

Principios de Manifiesto Ágil (1/2)

- Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.

- Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.

- Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los periodos breves.

- Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.

-Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.

- La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.

Principios de Manifiesto Ágil (2/2)

- El software que funciona es la principal medida del progreso.
- Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica enaltece la Agilidad.
- La simplicidad como arte de maximizar la cantidad de trabajo que no se hace, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto-organizan.
- En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

Valores buscados por los programadores de XP

- **Comunicación**
  - Con el equipo (hay que recordad que el cliente es parte del equipo)
- **Simplicidad**
  - Mantener el proceso tan simple como sea posible
- **Realimentación**
  - Buscar la realimentación del cliente en cada iteracción
- **Coraje**

Cuándo usar XP

- Requisitos desconocidos o inciertos
- Riesgo elevado
- Grupos pequeños (2 - 12)
- El cliente puede ser implicado en el desarrollo
- Capacidad de prueba

Reglas de XP

- Se escriben las historias del usuario
- La planificación de la *release* crea la planificación temporal
- Hacer frecuentes *releases* pequeños
- Se mide la velocidad del proyecto
- El proyecto se divide en iteraciones
- Cada iteración comienza con el plan de dicha iteración
- Mover a la gente dentro del proyecto (cambio de roles)
- Cada día comienza con una reunión (*meeting*) de pie
- Corrige el XP cuando falla

Codificación en XP

- El usuario está siempre disponible
- El código se debe escribir de acuerdo a estándares
- Hay que codificar primero la prueba de unidad
- Toda la producción del código se hace por parejas
- Sólo un par (de personas) integra el código en un tiempo
- Se integra a menudo
- Utilizar la posesión colectiva del código
- Dejar la optimización hasta el final
- No hacer horas extra

Cuando se piensa que puede funcionar XP

- Pequeños proyectos autocontenidos
- Cuando el cliente está disponible siempre (mejor todavía si tú eres tu propio cliente)
- Cuando un proyecto no implica interfaces con otros sistemas que también se están desarrollando
- Cuando el equipo de programación está muy preparado, particularmente en diseño.
- Cuando el equipo de programación es muy disciplinado – XP no es fácil de seguir

Dónde pueden estar los problemas

- Diseño – nadie prueba realmente la calidad del diseño
- Arquitectura – Nunca hay realmente una arquitectura – justamente emerge en el tiempo
- Reutilización – No está claro cómo se puede reutilizar cualquier software desarrollado bajo XP: La filosofía de XP tiende a mitigar la reutilización
- Programación en parejas – Parece claro que se necesitan programadores muy experimentados y disciplinados para conseguir que XP funcione – "*I don't believe that these types of programmers always work most productively in pairs (I believe that less experienced programmers may be more productive in pairs)*"
- El usuario está en el camino crítico para la vida completa del programa XP – esto da un poco de miedo

Opiniones personales que aparecen en: Apuntes de la asignatura Software Engineering. Process and Practice (CS 433-341) de la universidad de Melbourne.

TABLE 1

Process comparison of Toyota and Microsoft.

Toyota-style "lean" production (manual demand-pull with Kanban cards)	1990s Microsoft-style "agile" development (daily builds with evolving features)
JIT "small lot" production	Development by small-scale features
Minimal in-process inventories	Short cycles and milestone intervals
Geographic concentration—production	Geographic concentration—development
Production leveling	Scheduling by features and milestones
Rapid setup	Automated build tools and quick tests
Machine and line rationalization	Focus on small, multifunctional teams
Work standardization	Design, coding, and testing standards
Foolproof automation devices	Builds and continuous integration testing
Multiskilled workers	Overlapping responsibilities
Selective use of automation	Computer-aided tools, but no code generators
Continuous improvement	Postmortems, process evolution

Source: M. Cusumano, Staying Power, *Oxford, 2010, p. 197.*

SCRUM

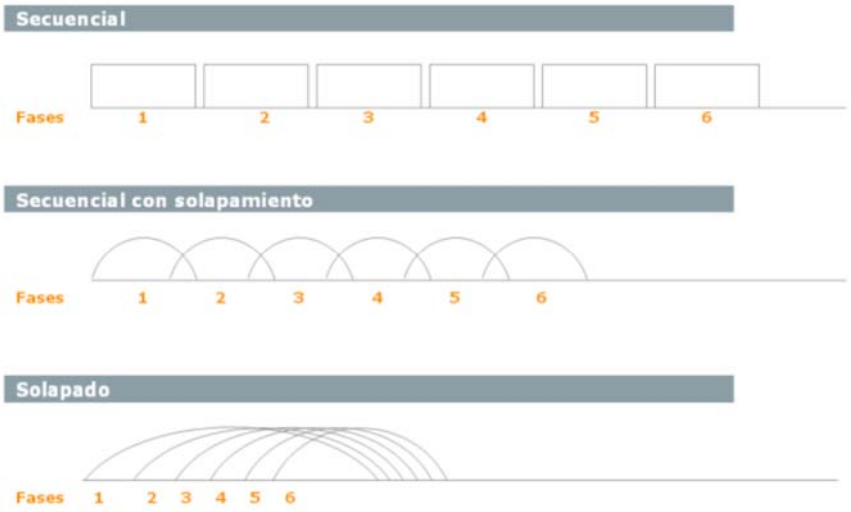
- 1986 (Hirotaka Takeuchi e Ikujiro Nonaka)
- Modelo para el desarrollo de todo tipo de productos tecnológicos
- Filosofía ágil:

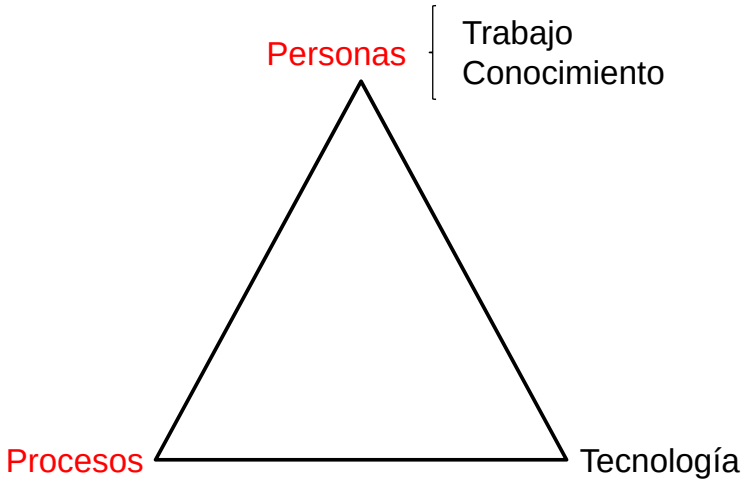
  Flexibilizar el proceso

  Evitar secuencialización

  Estrecha colaboración inter-equipo y con los clientes

La figura representa el ciclo de vida al construir un producto con un patrón de gestión secuencial, y cuál es la diferencia con la nueva forma, observada por Nonaka y Takeuchi en empresas que ignoraban los principios de la gestión clásica de proyectos.





Triángulo clásico de los factores de producción

Scrum considera solo dos, Personas y Procesos

Beta

Lanzamiento constante de novedades

Reducción del tiempo de desarrollo

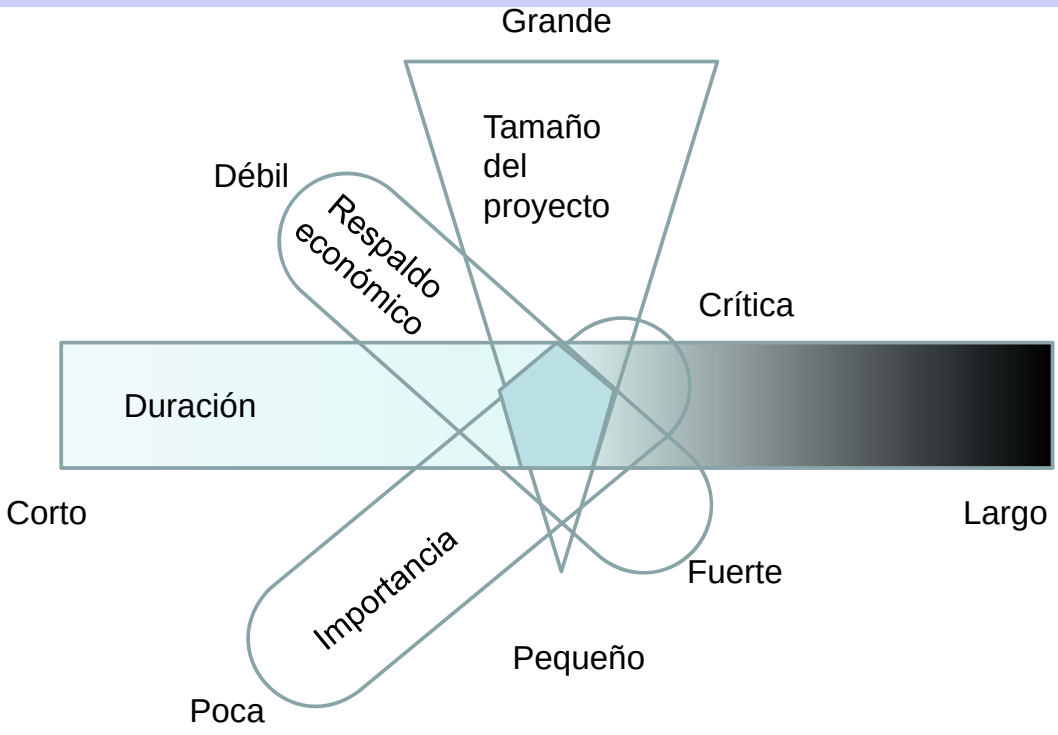
Cambios y mejoras rápidas

Innovación

Características del nuevo escenario

En los 40, 50 y 60 los productos tardaban años en quedar obsoletos, y las empresas los producían con variaciones mínimas a lo largo del tiempo. Apple ha desarrollado 6 versiones de su popular iPod, en sólo 6 años. Hoy determinados productos permanecen en un continuo estado “beta”.

Los cuatro atributos de un proyecto ideal



Fuente: Mike Cohn. *Succeding with agile. Software development using SCRUM*. Addison Wesley 2011

SCRUM. Principios

- Colaboración cliente
- Respuesta al cambio
- Desarrollo incremental

  Entregas funcionales frecuentes
- Auto-gestión, auto-organización y compromiso
- Simplicidad
- Supresión de artefactos innecesarios en la gestión del proyecto

Preferencias de la Gestión Ágil

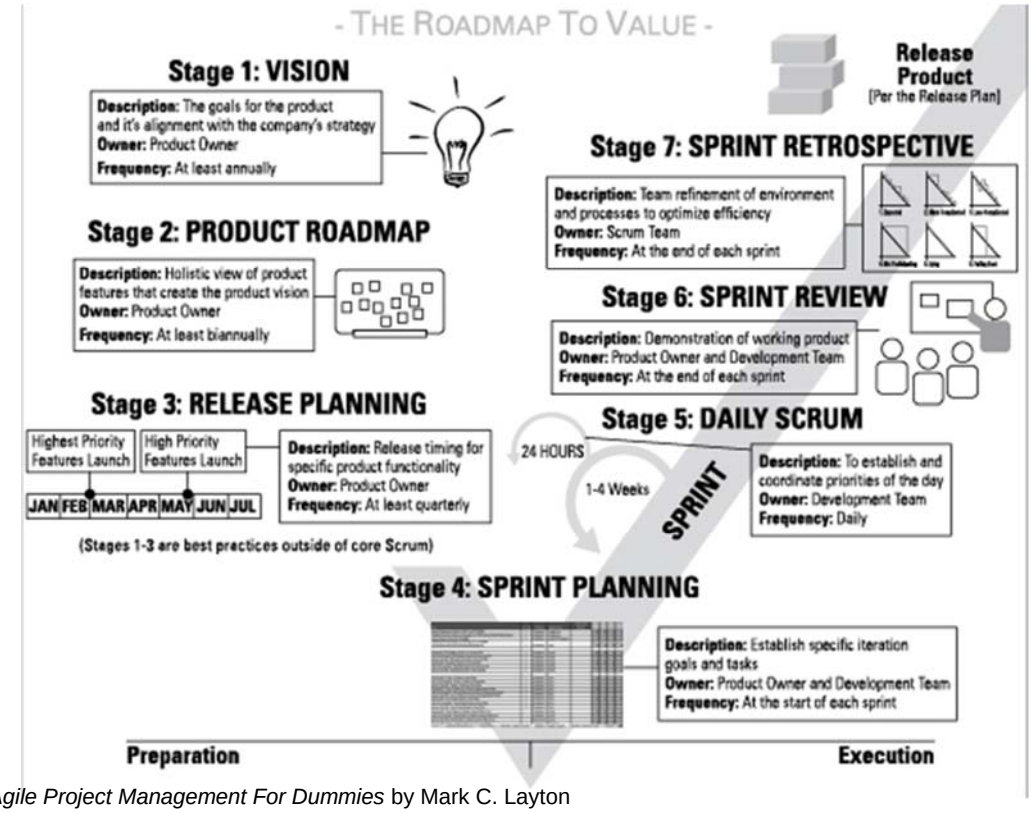
DESARROLLO TRADICIONAL	DESARROLLO ÁGIL
Especialización	Equipo multidisciplinar
Fases	Solapamiento
Requisitos detallados	Visión del producto
Seguimiento del plan	Adaptación a los cambios

Diferencias entre el desarrollo tradicional y el desarrollo ágil.

Las características “ambientales” en las empresas que desarrollan los nuevos productos con modelos de gestión ágil son:

- Incertidumbre consustancial al entorno y la cultura de la organización.
- Equipos auto-organizados.
- Control sutil
- Difusión y transferencia del conocimiento

El resultado es la gestión ágil de proyectos, que no se formula sobre el concepto de anticipación (requisitos, diseño, planificación y seguimiento) sino sobre el de adaptación (visión, exploración y adaptación)



From *Agile Project Management For Dummies* by Mark C. Layton

Roles de la Gestión Ágil (adaptados para SCRUM)

Equipo de desarrollo: El grupo de personas que trabajan en la creación de un producto. Diseñadores, programadores, probadores, etc.

Product owner: La persona implicada en establecer un puente entre el usuario final, los responsables del negocio y el equipo de desarrollo. Trabaja día a día para clarificar los requisitos. A veces se le conoce como *customer representative.*

Scrum master: La persona responsable de apoyar al equipo de desarrollo, eliminar las barreras organizativas y mantener la consistencia del proyecto ágil. A veces se le denomina *project facilitator.*

Stakeholders: Cualquiera con interés en el proyecto. No son los responsables del producto pero pueden proporcionar entradas y estar afectados por las salidas del proyecto. Puede ser gente muy diversa, de diferentes departamentos y, a veces, de diferentes compañías.

Agile mentor: Alguien que tiene experiencia en implementar proyectos ágiles y puede compartir su experiencia con el equipo del proyecto.

From *Agile Project Management For Dummies* by Mark C. Layton

Artefactos de la Gestión Ágil (adaptados para SCRUM)

Igual que en otros modelos el progreso de los proyectos necesita ser medible. Los proyectos ágiles utilizan habitualmente seis artefactos (o entregables):

Establecimiento de la visión del producto: Un resumen rápido para comunicar cómo está enmarcado el producto en la empresa o en la organización. El documento de visión debe articular los objetivos del producto.

***Product backlog*:** La lista total de los requisitos que están en el alcance del proyecto ordenado por prioridad. En el momento en que se tenga el primer requisito, se debe tener un *backlog* del producto.

Product roadmap: Es una vista de alto nivel de los requisitos del producto con un marco temporal poco preciso de cuando se desarrollarán los requisitos.

Plan de Release: Un calendario de nivel alto para el lanzamiento de software funcionando.

Sprint backlog: El objetivo, las historias del usuario y las tareas asociadas con el sprint actual.

Incremento: La funcionalidad del producto en funcionamiento al final de cada sprint.

From *Agile Project Management For Dummies* by Mark C. Layton

Eventos de la Gestión Ágil (adaptados para SCRUM)

En los proyectos ágiles existen siete eventos para el desarrollo de productos:

Planificación del proyecto: La planificación inicial. Incluye una visión del producto y un "*product roadmap*". Se puede realizar en menos de un día.

Planificación de release: La planificación del conjunto siguiente de características que va a tener la nueva release del producto. Solo se planifica una release cada vez.

Sprint: Un ciclo de desarrollo corto para crear funcionalidad de producto vendible. A veces se llaman iteraciones, normalmente entre una y cuatro semanas. La duración debe ser la misma durante todo el proyecto.

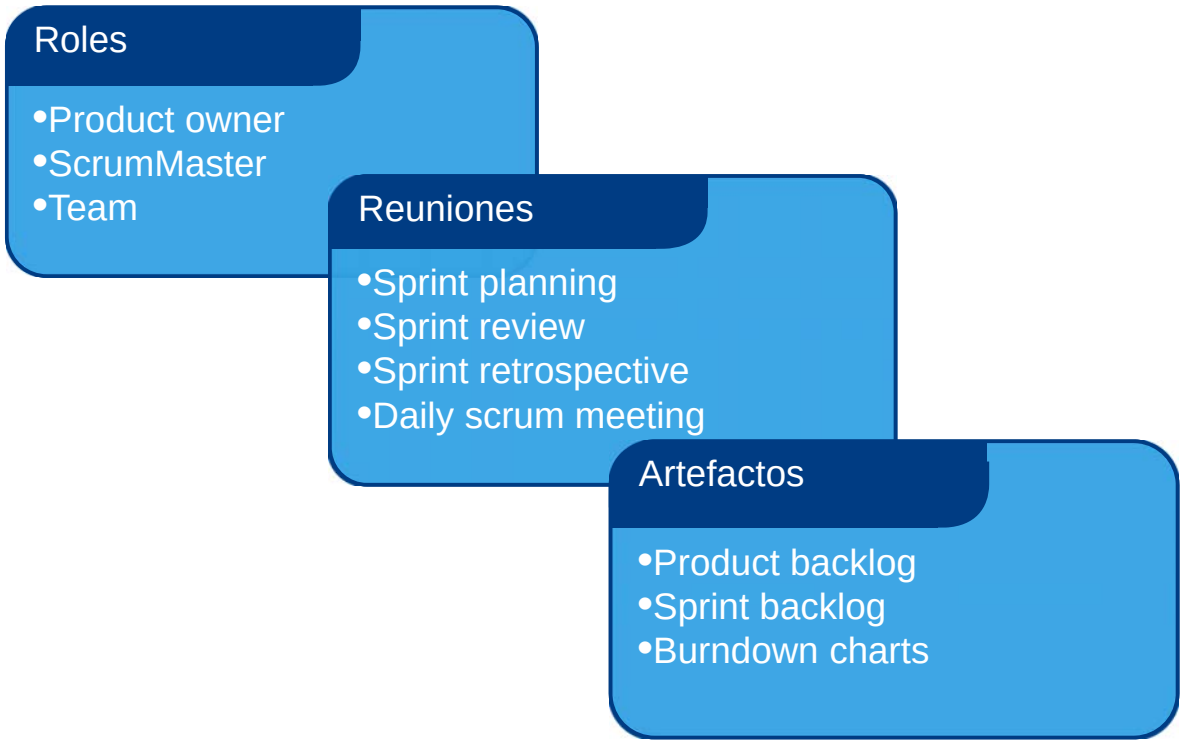
Planificación de Sprint: Una reunión al comienzo de cada sprint donde se fijan los objetivos del mismo. Se identifican los requisitos y las tareas a realizar para completar cada requisito.

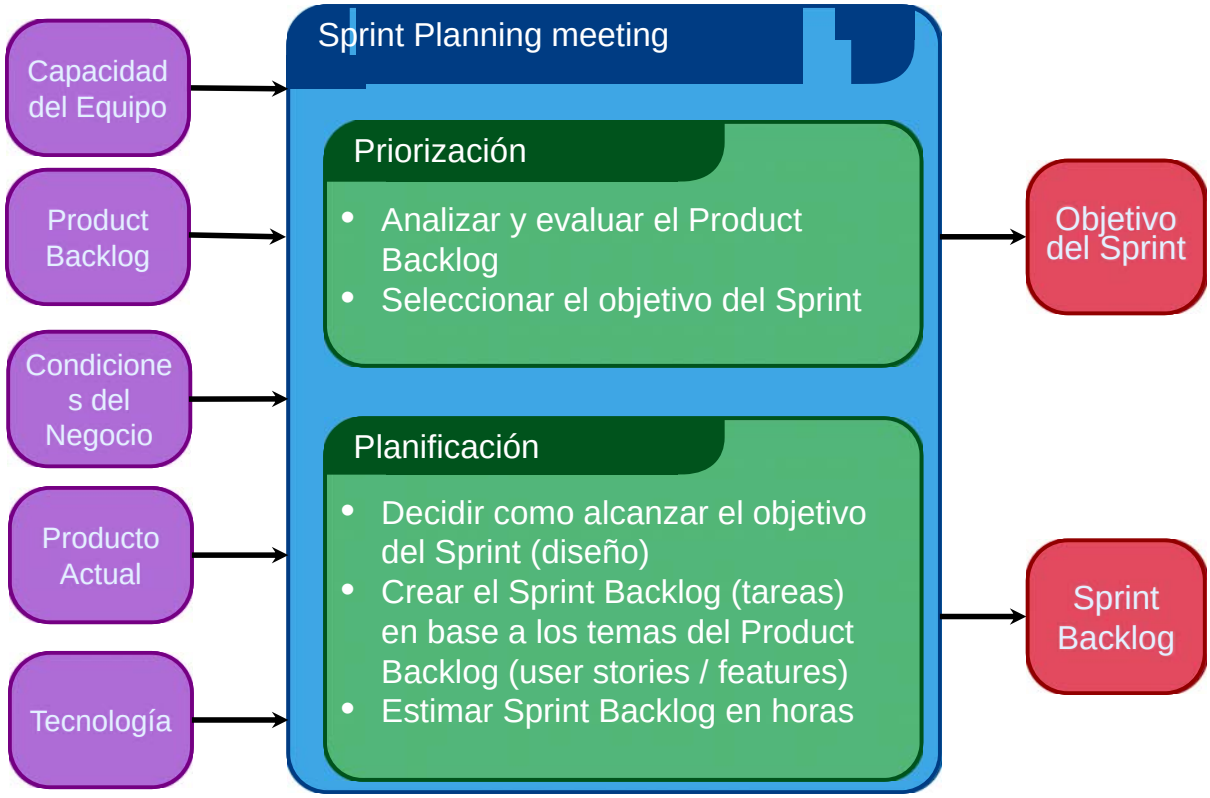
Daily scrum: Una reunión de unos 15 minutos a tener cada día de un sprint donde se indica lo realizado el día anterior y lo que se va a realizar en el actual.

Revisión de Sprint: Una reunión al final de cada sprint, iniciada por el "*product owner*", donde se analiza la funcionalidad del producto conseguida durante el sprint.

Sprint retrospective: Una reunión al final de cada sprint, donde el equipo del sprint analiza qué fue bien, qué debe cambiar y cómo realizar los cambios.

Marco de trabajo SCRUM





Planificación del Sprint

- El equipo selecciona los temas a partir 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

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

→

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)

Scrum, reunión diaria

- 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

Scrum, reunión diaria

Todos deben responder a tres preguntas

1. ¿Qué hiciste ayer?
2. ¿Qué vas a hacer hoy?
3. ¿Hay obstáculos en tu camino?

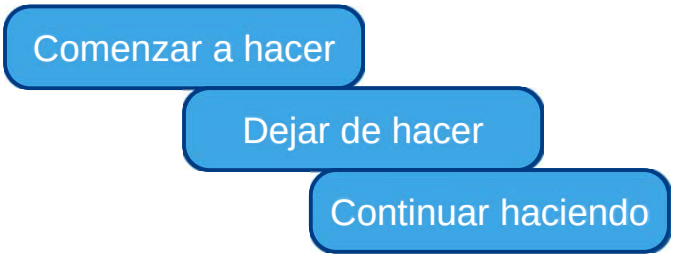
- No es dar un status report al Scrum Master
- Se trata de compromisos delante de pares

Sprint review

- El equipo presenta lo realizado durante el sprint
- Normalmente adopta la forma de una demo de las nuevas características o la arquitectura subyacente
- Informal
  - Regla de 2 horas preparación
  - No usar diapositivas
- Todo el equipo participa
- Se invita a todo el mundo

Sprint retrospective

- Periódicamente, se echa un vistazo a lo que funciona y lo que no
- Normalmente 15 a 30 minutos
- Se realiza después de cada sprint
- Todo el equipo participa
  - ScrumMaster
  - Product owner
  - Equipo
  - Posiblemente clientes y otros
- Todo el equipo discute lo que le gustaría



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
- La lista es priorizada por el Product Owner
- Repriorizada al comienzo de cada Sprint

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

Gestión del Sprint Backlog

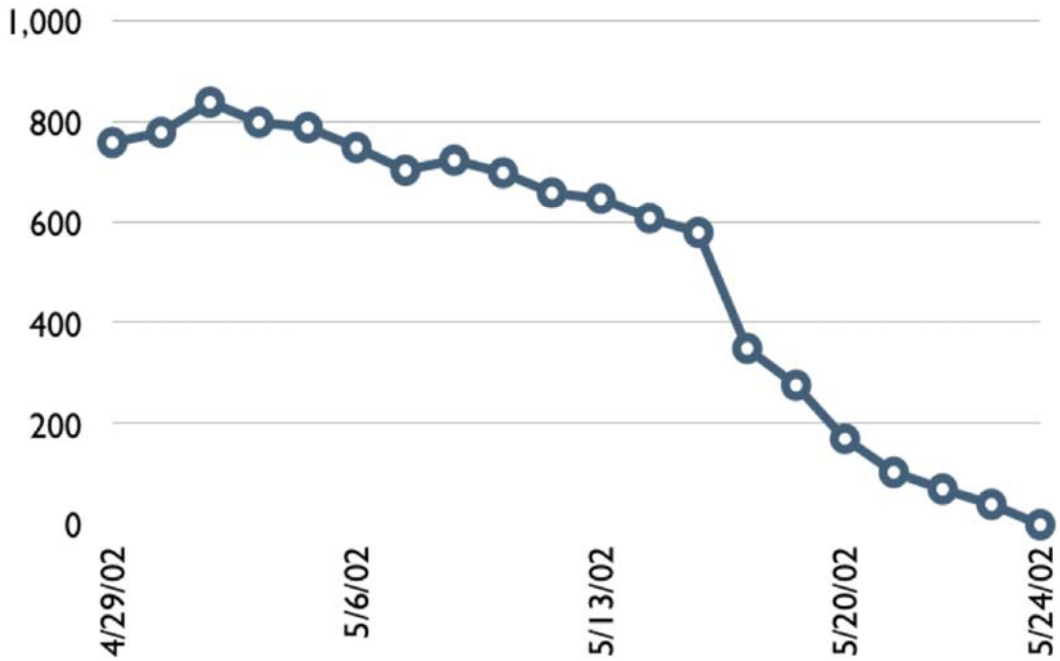
En el Sprint se hace una declaración breve de cuál será el foco del trabajo durante el mismo.

- 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

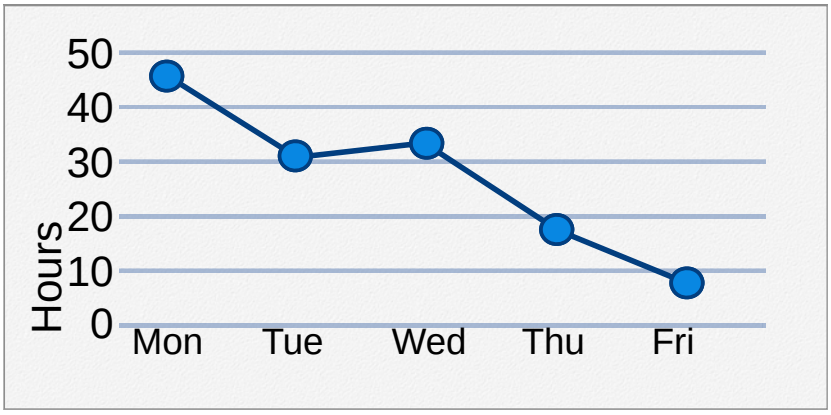
Ejemplo de Sprint Backlog

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

Sprint Burndown Chart



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				



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

Objetivo de la Gestión Ágil

El objetivo es dar garantías a las demandas principales de la industria actual:

- **Valo**r. Para dar el mayor valor posible a un producto cuando dicho producto se basa en:

  Innovación

  Flexibilidad

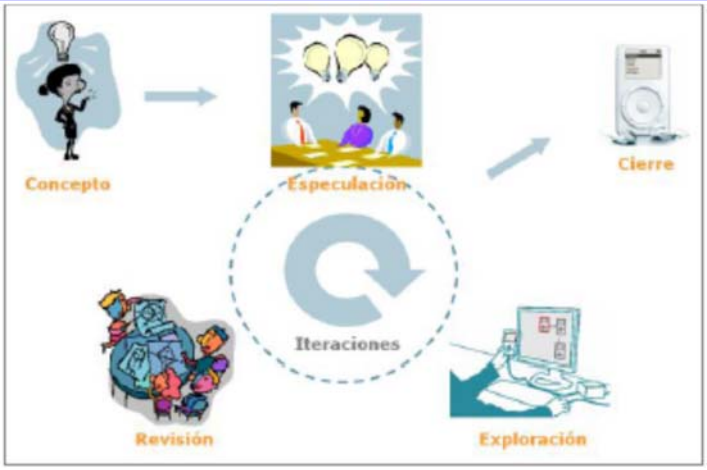
- **Reducción del tiempo de salida al mercado**.
- **Agilidad**. Capacidad para producir partes completas del producto en periodos breves de tiempo.
- **Flexibilidad**. Capacidad de adaptar la forma y el desarrollo a las características del proyecto
- **Resultados fiables.**

Para lograr la auto-organización los equipos deben reunir tres características:

Autonomía. Son libres para elegir la estrategia de la solución. En este sentido la dirección de la empresa actúa como un capitalista de capital-riesgo.

Auto-superación. El equipo va desarrollando soluciones, que evalúa, analiza y mejora.

Auto-enriquecimiento. La multi-disciplinaridad del equipo favorece el enriquecimiento mutuo y la aportación de soluciones valiosas complementarias.



Los ciclos breves de desarrollo, se denominan iteraciones y se realizan hasta que se decide no evolucionar más el producto.
Este esquema está formado por cinco fases:

1.- Concepto
2.- Especulación
3.- Exploración
4.- Revisión
5.- Cierre

Especulación

Una vez que se sabe qué hay que construir, el equipo especula y formula hipótesis basadas en la información de la visión. En esta fase se determinan las limitaciones impuestas por el entorno de negocio: costes y agendas principalmente, y se cierra la primera aproximación de lo que se puede producir.

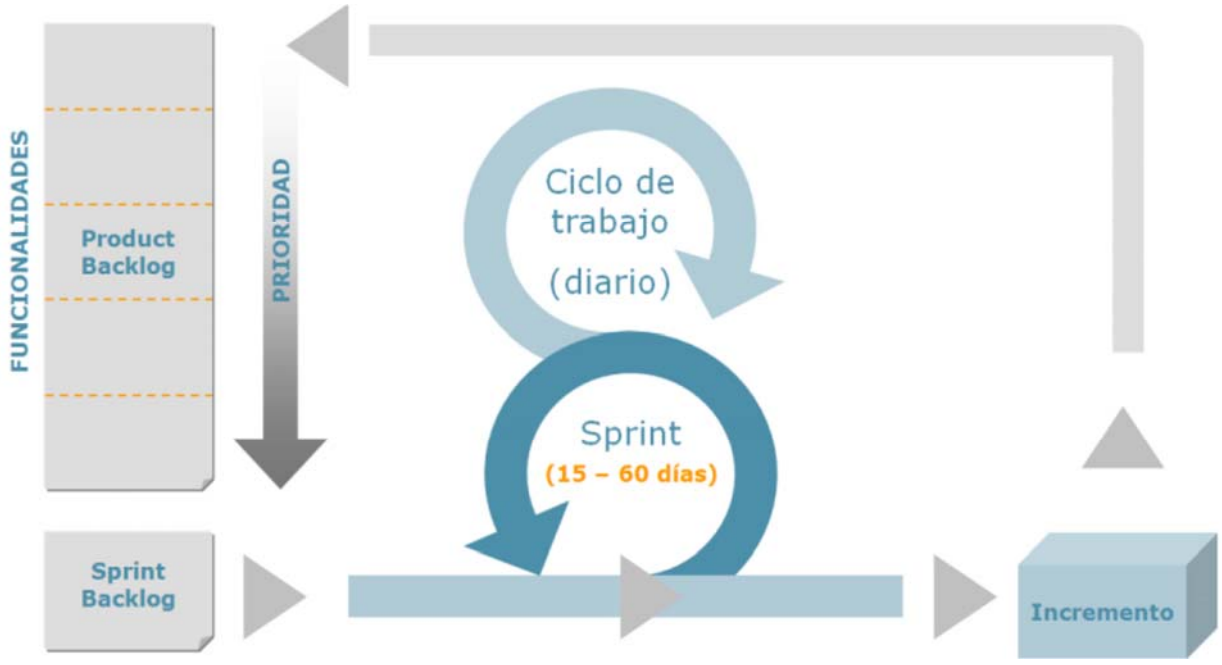
La gestión ágil investiga y construye a partir de la visión del producto. Durante el desarrollo confronta las partes terminadas: su valor, posibilidades, y la situación del entorno en cada momento.

La fase de especulación se repite en cada iteración, y teniendo como referencia la visión y el alcance del proyecto consiste en:

Desarrollo y revisión de los requisitos generales.
Mantenimiento de una lista con las funcionalidades esperadas.
Mantenimiento de un plan de entrega: fechas en las que se necesitan las versiones, hitos e iteraciones del desarrollo. Este plan refleja ya el esfuerzo que consumirá el proyecto durante el tiempo.
En función de las características del modelo de gestión y del proyecto puede incluir también una estrategia o planes para la gestión de riesgos.

Si las exigencias formales de la organización lo requieren, también se produce información administrativa y financiera.

SCRUM. Ciclo de vida: sprints

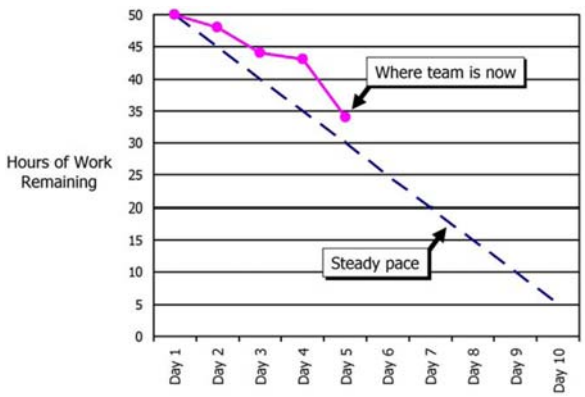


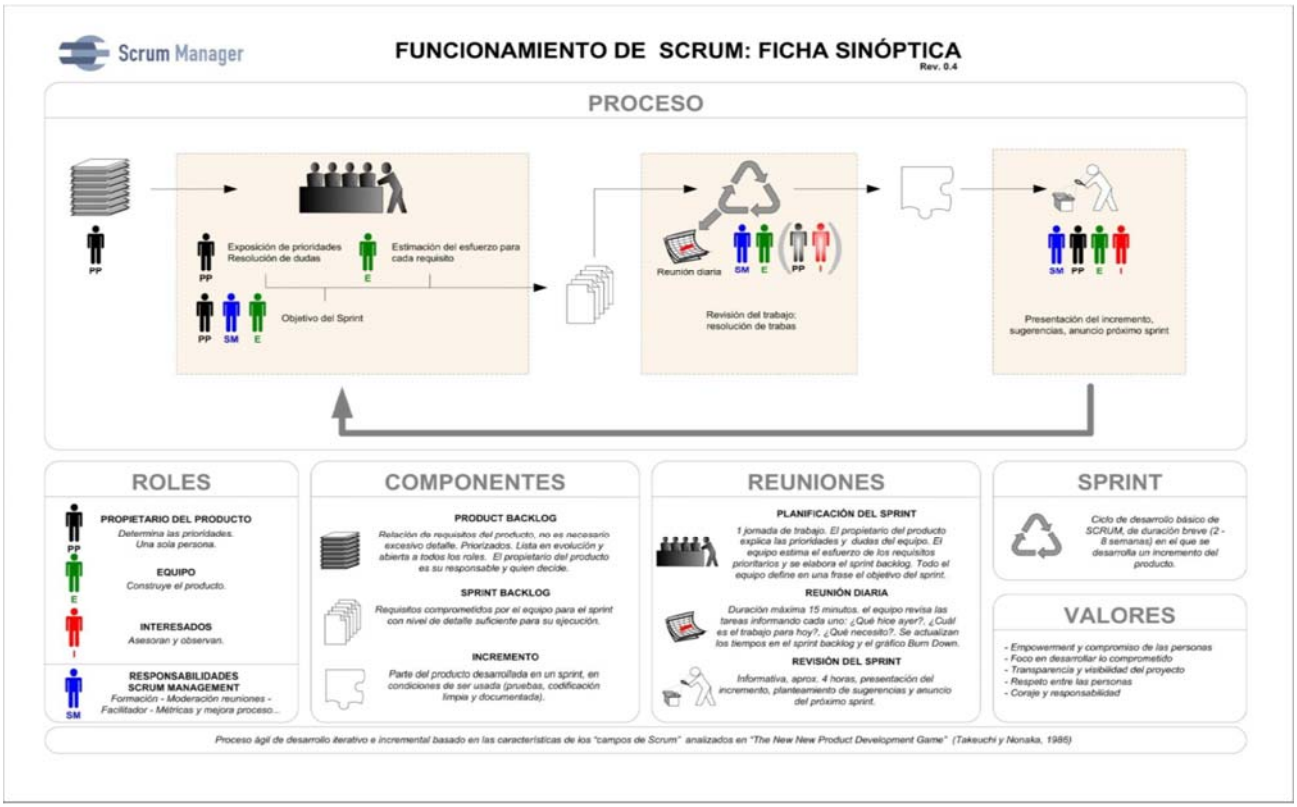
Backlog Item	Task	Owner	Initial Time Estimate
Enable all users to place book in shopping cart	Configure database and space IDs for Trac	Sanjay	4 hours
	Use test data to tune the learning and action model	Jing	2 hours
	Setup a cart server code to run as apache server	Philip	3 hours
	Implement pre-Login Handler	Tracy	3 hours
Upgrade transaction processing module (must be able to support 500 transactions /sec)	Merge DCP code and complete layer-level tests	Jing	5 hours
	Complete machine order for pRank	Jing	4 hours
	Change DCP and reader to use pRank http API	Tracy	3 hours

Ejemplo de Sprint Backlog

SCRUM. Seguimiento temporal

		Hours of Work Remaining on Each Day of the Sprint									
Task	**Task Owner**	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**	**Day 6**	**Day 7**	**Day 8**	**Day 9**	**Day 10**
Configure database and space IDs for Trac	Sanjay	4	4	3	1	0					
Use test data to tune the learning and action model	Jing	2	2	2	2	1					
Setup a cart server code to run as apache server	Philip	3	3	5	2	0					
Implement pre-Login Handler	Tracy	3	3	3	3	3					
Merge DCP code and complete layer-level tests	Jing	5	5	2	2	2					
Complete machine order for pRank	Jing	4	4	3	3	3					
Change DCP and reader to use pRank http API	Tracy	3	3	0	0	0					
	Total	**50**	**48**	**44**	**43**	**34**					





SCRUM. Algunas prácticas (1/2)

- Los clientes se convierten en parte del equipo de desarrollo.

- Scrum tiene frecuentes entregables intermedios con funcionalidad operativa (el producto funciona), como otras formas de procesos de software ágiles. Esto permite al cliente conseguir trabajar con el software antes y permite al proyecto cambiar los requisitos de acuerdo con las necesidades.

- Se desarrollan planes de riesgos y mitigación frecuentes por parte del equipo de desarrollo. La mitigación de riesgos, la monitorización y la gestión de riesgos se lleva a cabo en todas las etapas y con compromiso.

Fuente: Curso de desarrollo ágil. INTECO, 2009.

SCRUM. Algunas prácticas (2/2)

• Transparencia en la planificación y desarrollo de módulos lo que permitirá saber a cada uno quién es responsable de qué y cuándo.

• Frecuentes reuniones de las personas involucradas en el negocio para monitorizar el progreso.

• Debería haber un mecanismo avanzado de advertencias.

• Los problemas no se han de barrer debajo de la alfombra. Nadie es penalizado por reconocer o describir un problema imprevisto.

Fuente: Curso de desarrollo ágil. INTECO, 2009.

Relación entre métodos Ágiles y Proceso Unificado (1/2)

- Participación activa de los "statkeholders"
- Aplicar estándares de modelado
- Aplicar patrones cuidadosamente
- Aplicar el artefacto correcto
- Propiedad compartida
- Considerar la realización de pruebas
- Crear varios modelos en paralelo
- Crear contenido sencillo
- Dibujar modelos de forma sencilla
- Desechar modelos temporales

Relación entre métodos Ágiles y Proceso Unificado (2/2)

- Exponer públicamente los modelos
- Formalizar los modelos de contrato
- Iterar a otro artefacto
- Modelar en pequeños incrementos
- Modelar para comunicar
- Modelar para comprender
- Probarlo con código
- Reutilizar recursos existentes
- Actualizar solo cuando sea preciso
- Utilizar herramientas sencillas

Condiciones de paso de grupo a empresa

La gestión por procesos encierra cuatro factores que hacen posible pasar de grupo de emprendedores a una empresa establecida:

Repetitividad de resultados. Al conseguir que la calidad del resultado sea consecuencia del proceso, desarrollar nuevas aplicaciones aplicando el mismo proceso garantiza la homogeneidad de los resultados.

Escalabilidad. Es una consecuencia de la repetitividad. No sólo un equipo consigue resultados homogéneos en todos los proyectos, sino que los obtienen todos los equipos, independientemente del tamaño y objetivos del mismo.

Mejora continua. Al aplicar meta-procesos que trabajan sobre los propios procesos de producción, midiendo y analizando los resultados se obtienen los criterios de gestión necesarios para aplicar medidas que mejoran de forma continua la eficiencia y calidad de los procesos base, y por tanto de los resultados.

Un **know-how propio**, consiguiendo finalmente una empresa que sabe hacer, porque su modelo de procesos termina conteniendo un activo valioso de la organización: la clave para hacer las cosas bien, con eficiencia y de forma homogénea.

Fuente: *Flexibilidad con Scrum* . Juan Palacio. 2008

Referencias bibliográficas

Scott W. Ambler. *Agile Modeling*. John Wiley and Sons 2002.

Kent Beck. *Una explicación de la programación extrema*. Addison-Wesley 2002.

Scrum Manager. Gestor de proyectos. Revisión 1.3. SafeCreative. 2010. Disponible en http://www.scrummanager.net/files/sm_proyecto.pdf

The Scrum Papers: Nuts, Bolts, and Origins of an Agile Process

Jeff Sutherland, Ph.D., Ken Schwaber, Co-Creators of Scrum

https://www.scrum.org/ Página de Scrum.org

Apuntes de la asignatura *Software Engineering. Process and Practice* (CS 433-341) de la universidad de Melbourne.