Gestión Ágil de Proyectos

Jose-Luis Poza-Luján

Roadmap

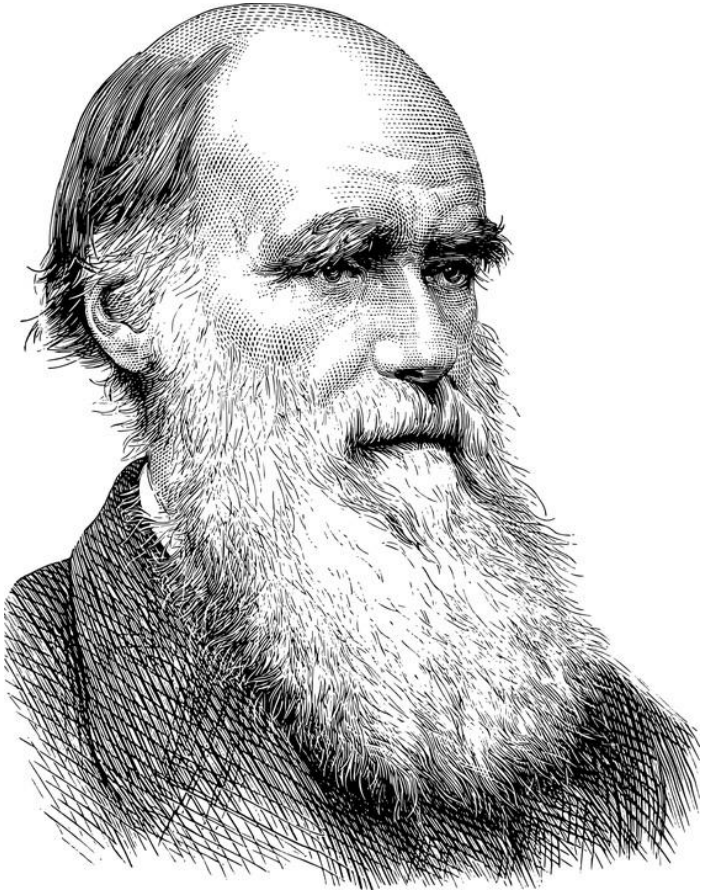
La necesidad de la agilidad → Objetivo: entender cómo se ha llegado a las metodologías ágiles.

El Manifiesto ágil → Objetivo: conocer el manifiesto ágil como fundamento de aplicación de las metodologías ágiles.

Metodologías ágiles → Objetivo: Conocer algunas de las metodologías ágiles originales y actuales.

La necesidad de la agilidad

La adaptación al cambio como base de la evolución



Los individuos menos **adaptados** al medio ambiente tienen menos probabilidades de sobrevivir y menos probabilidades de reproducirse; los individuos más aptos tienen más probabilidades de sobrevivir y más posibilidades de reproducirse y de dejar sus rasgos hereditarios a las generaciones futuras, lo que produce el proceso de selección natural.

Este proceso lento da como resultado **cambios** en las poblaciones para adaptarse a sus entornos, y en última instancia, estas variaciones se acumulan con el tiempo para formar nuevas especies.

Mayr, 1982 (resumen de “El origen de las especies” de Charles Darwin, 1859)

Mi bola de cristal tiene nubes

- ¿Qué ha cambiado en los últimos tiempos? a partir de la crisis del 2008, del 2020, del...
  - Antes de la crisis: certezas
    - La metodología predictiva funcionaba bien
  - Después de la crisis: incertidumbre
    - La metodología tradicional no cubre la incertidumbre
- Conclusión:
  - El futuro no se puede predecir
  - Las metodologías clásicas se basan en las certezas
  - En un entorno de incertidumbre (como la investigación) o cambiante a gran velocidad (como el tecnológico y el digital) las metodologías clásicas pueden no ser las más adecuadas.
  - Son necesarias metodologías con gran capacidad de adaptación

¿Dos mundos enfrentados?

Entornos no predecibles

- Se puede tener cierta idea del producto (alcance) que se busca, pero no se conoce el resultado final
- No necesariamente se conocen qué tareas se van a tener que realizar, consecuentemente no se sabe nada de las fechas de inicio ni fin.
- La gestión del proyecto consiste en decidir qué tareas a realizar, aportar valor y aprovechar incidencias.

Entornos predecibles

- Desde antes de empezar se conoce perfectamente el producto (alcance) a realizar, por lo que se conocen los resultados finales.
- Se pueden conocer todas las tareas necesarias para los resultados con fechas concretas de inicio y final.
- La gestión del proyecto consiste en supervisar las tareas, evitar riesgos y solucionar las incidencias.

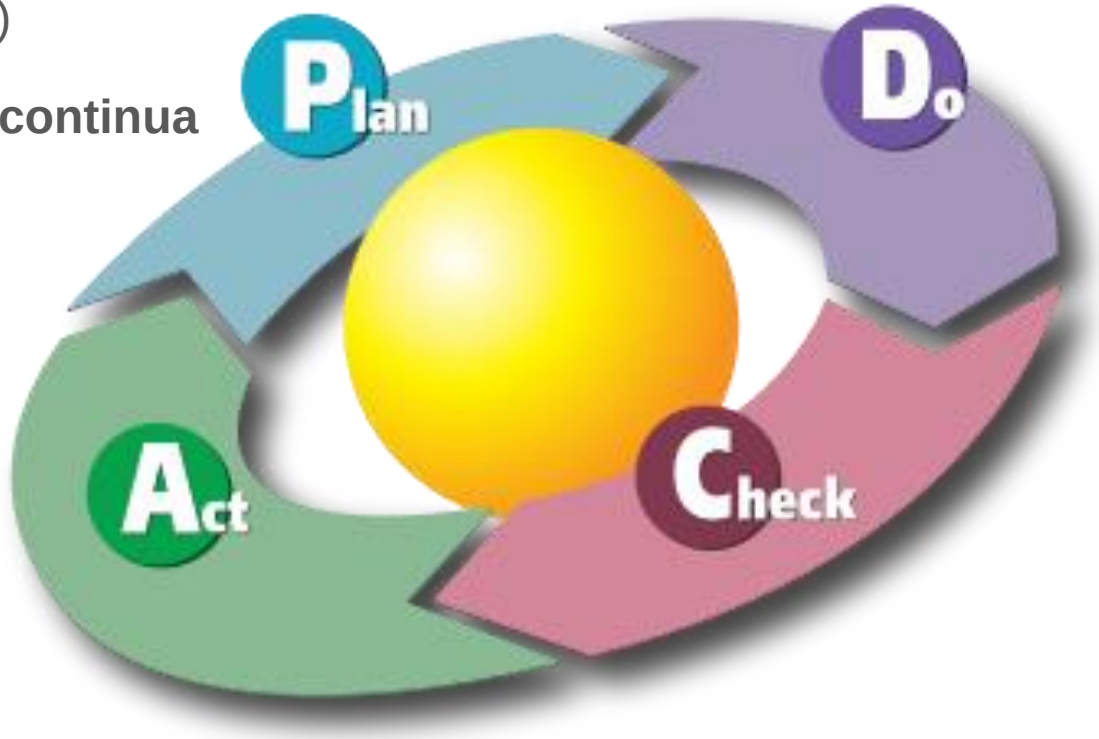
De acuerdo, cambiamos ¿Pero cómo?

Concepto: **Kaizen**: kai: cambio + zen: bueno

Objetivo: **Lean** (esbelto, esbelta)

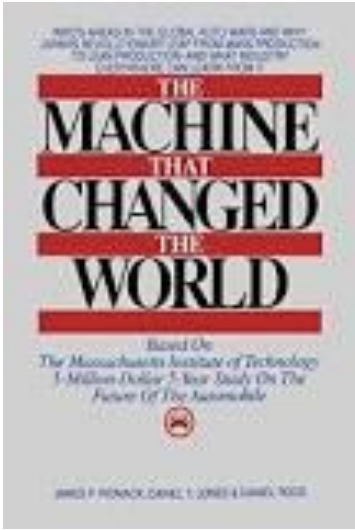
Método: Mecanismo de **mejora continua**

Ejemplo: Ciclo de Deming



El modelo Toyota

- Toyota Production System TOYOTA en los años 40 (siglo XX)
  - Son un conjunto de principios y valores: pensamiento
  - Busca el modelo Lean y, consecuentemente, trata de evitar los problemas de los procesos tradicionales:
    - Poca flexibilidad
    - Problemas repetidos
    - Jerarquías establecidas e incuestionables
    - Liderazgo entendido como supervisión y órdenes
    - Falta de motivación
    - Sin proceso de mejora
- La estableció como filosofía/método: James P. Womack (MIT) a principio de los años 90 (siglo XX)

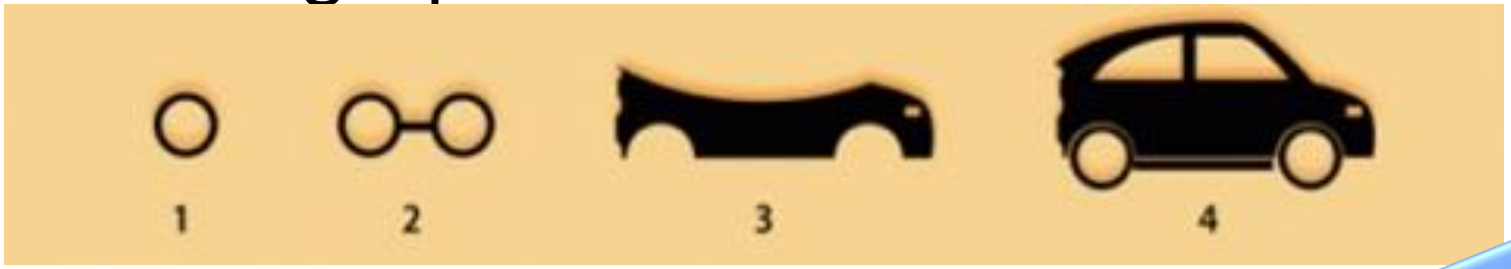


Los orígenes de la agilidad: Lean

- Fundamentos Lean
  - Pasión por hacer las cosas simples… mejor sencillas
  - Motivación para apartar lo superfluo (no aporta valor)
  - Búsqueda de la reducción de plazos pero por medio del uso eficiente del tiempo
  - Interés por cómo el cliente usa el producto desarrollado
  - Talento: en lugar de buscar clones en los equipos se favorece la aportación diferenciada de cada componente.
  - Sinergia: como las conexiones entre distintos talentos generan más talento
- LEAN: Las tres M
  - MUDA: desperdicios (algo se nos escapa por algún sitio)
  - MURI: sobrecargas (consumo excesivo de recursos)
  - MURA: desajustes (algo está desequilibrado)

Producto ágil

- Entorno no ágil: producto final conocido

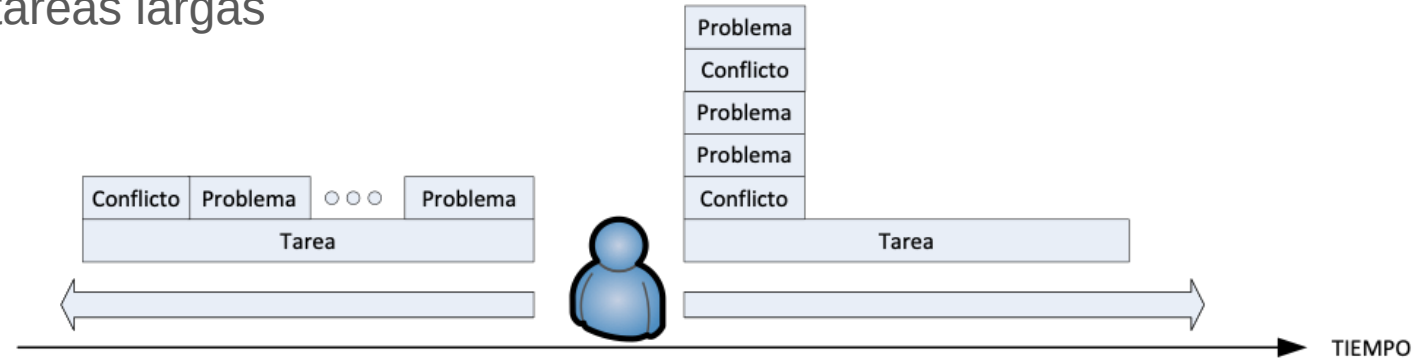


- Entorno ágil: mínimo producto viable

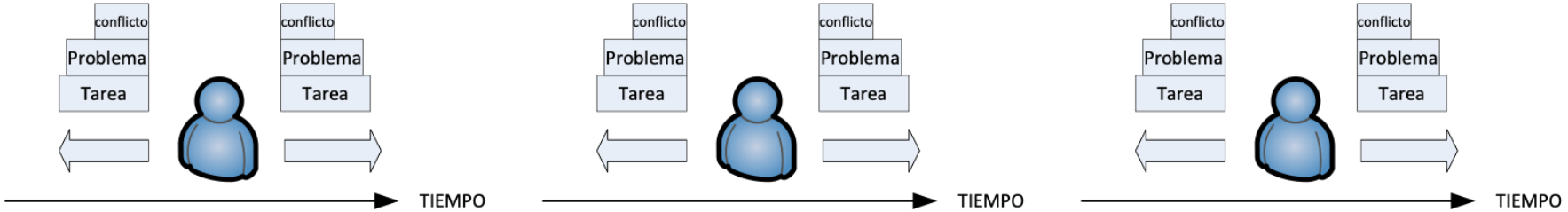
Design Thinking (problema cliente)

Tareas ágiles

De las tareas largas

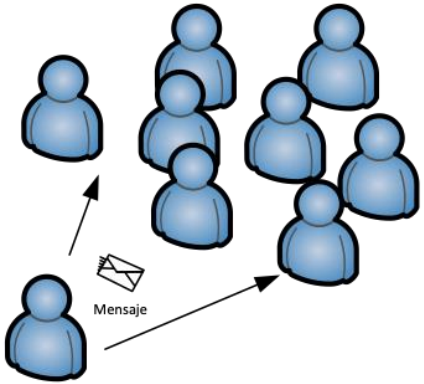


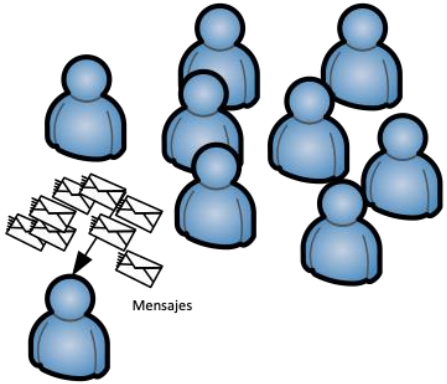
Pasamos a las tareas cortas



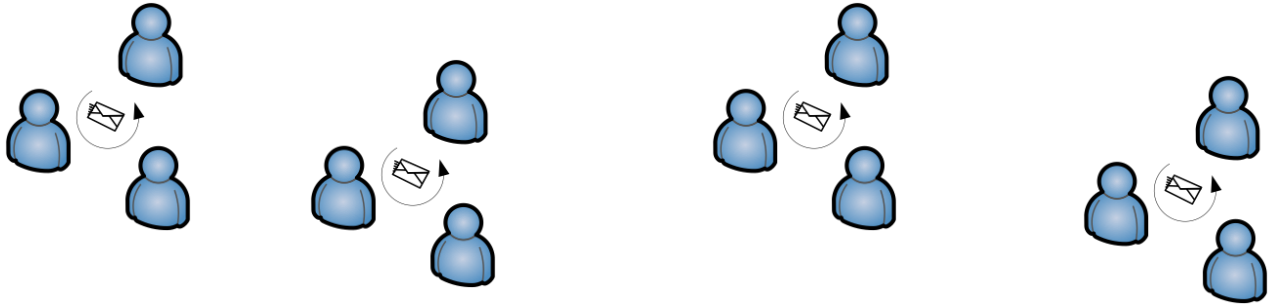
Equipos ágiles

De los equipos grandes





Pasamos a los equipos pequeños



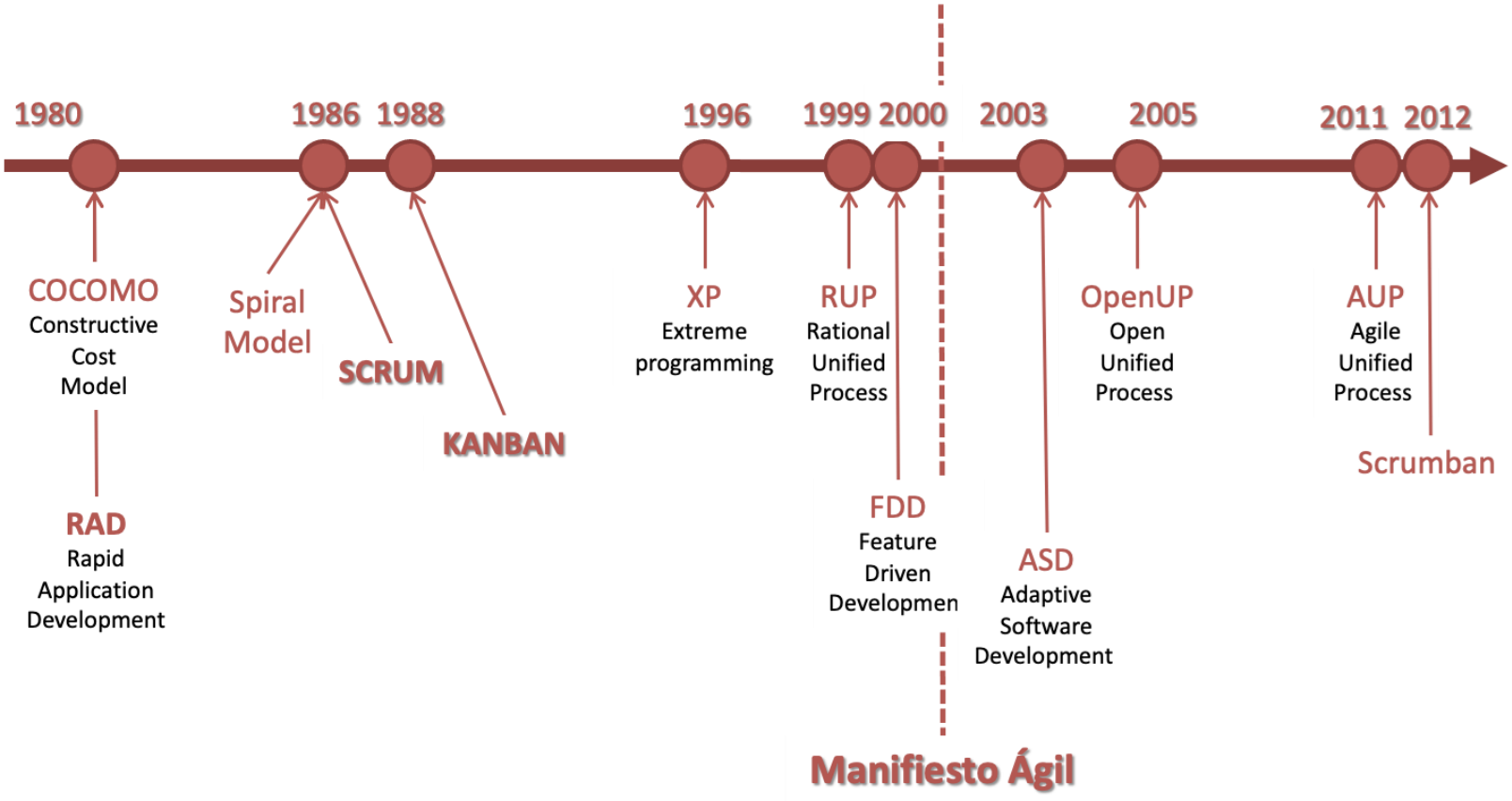
El concepto de compromiso

Personas comprometidas :sacrifican recursos no sustituibles

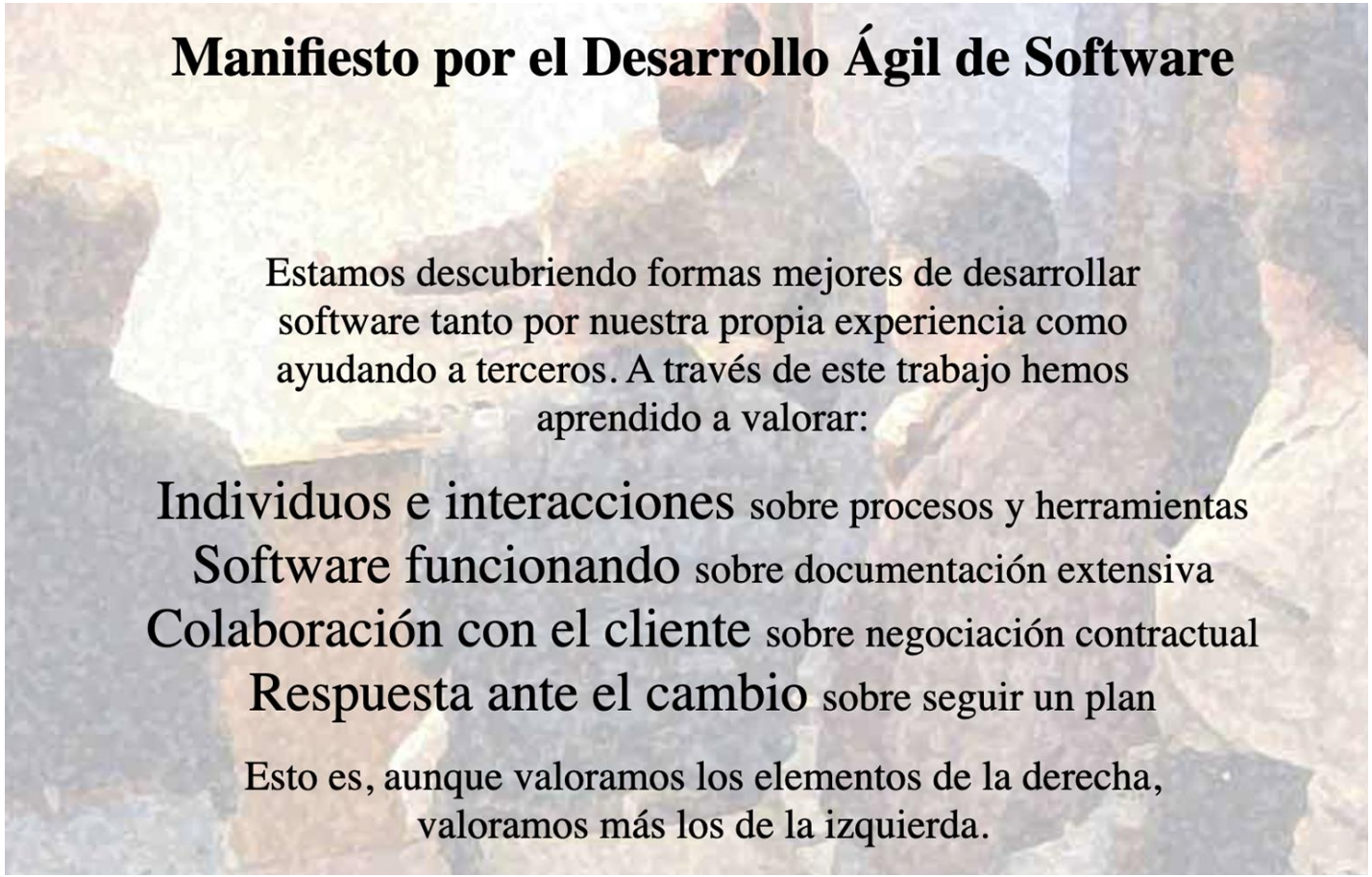
Personas involucradas: contribuyen con recursos sustituibles

El Manifiesto ágil

Historia


1980
1986
1988
1996
1999
2000
2003
2005
2011
2012
COCOMO
Constructive Cost Model
RAD
Rapid Application Development
Spiral Model
SCRUM
KANBAN
XP
Extreme programming
RUP
Rational Unified Process
FDD
Feature Driven Developmen
ASD
Adaptive Software Development
OpenUP
Open Unified Process
AUP
Agile Unified Process
Scrumban
Manifiesto Ágil

El Manifiesto (https://agilemanifesto.org/)



Los principios del desarrollo ágil

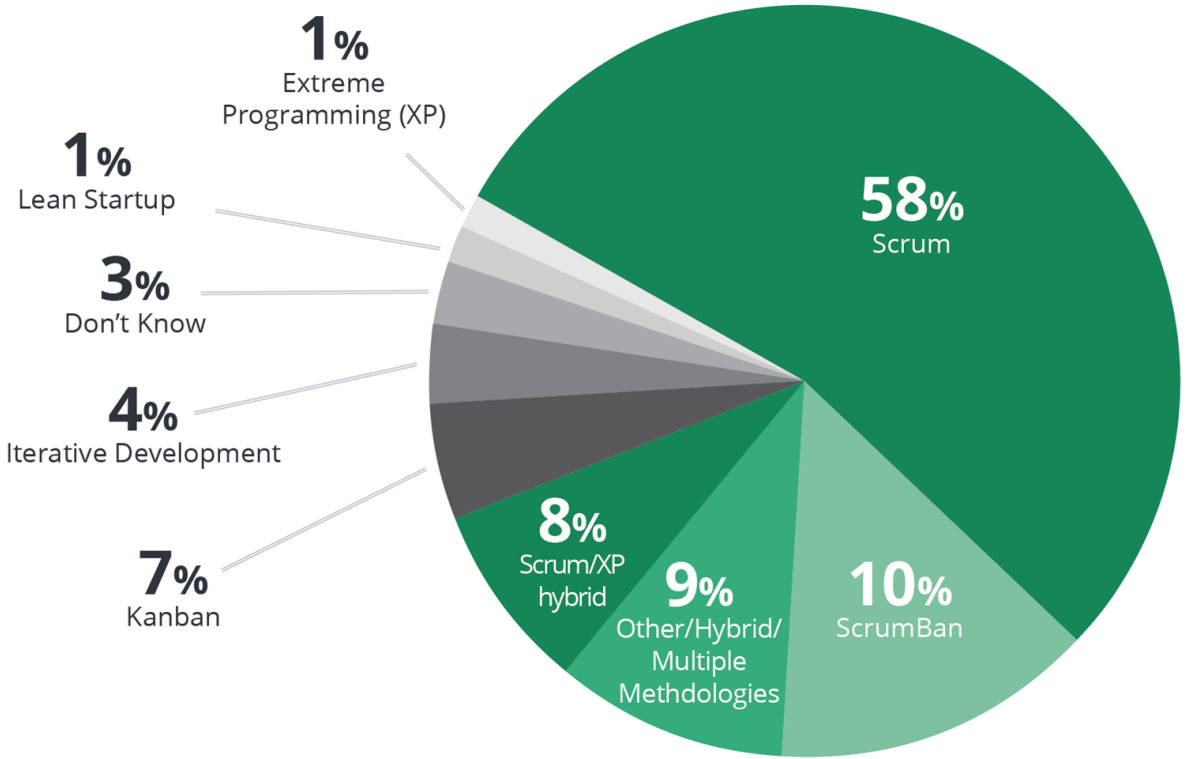
1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Del Software a los Proyectos

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de **producto** con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos **producto** funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El **producto** funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Estado actual

https://stateofagile.com/



Total exceeds 100% due to rounding.

Las metodologías ágiles

De dónde partimos

- Años 80 del siglo XX: COnstructive COst Model (como ejemplo de la visión predominante)
- Modelo para estimar el costo de un proyecto software
  - Base empírica, pero estimación subjetiva de los parámetros.
- El producto se clasifica según el tipo
  - Organic (orgánico)
    - Desarrollado por el personal experimentado
    - Entorno de trabajo “familiar”
  - Semi-detached (semi-desacoplado)
    - Tipo intermedio entre el orgánico y el embebido.
    - Desarrollado por una mezcla de personal experimentado y no experimentado.
  - Embedded (embebido)
    - Tipo de proyecto con restricciones y de alta criticidad.
    - Proyectos innovadores, no es necesaria la experiencia por lo que no hay condicionantes con respecto al personal.

El modelo COCOMO

- COnstructive COst Model es un ejemplo de intentar medir con muchas fórmulas un proyecto
  - Generalmente software

- **RELY**: garantía de funcionamiento requerida al software. Indica las humanas (*extremadamente alto*, software de alta criticidad).
- **DATA**: tamaño de la base de datos en relación con el tamaño del pr código.
- **CPLX**: representa la complejidad del producto.
- De hardware
  - **TIME**: limitaciones en el porcentaje del uso de la CPU
  - **STOR**: limitaciones en el porcentaje del uso de la me
  - **VIRT**: volatilidad de la máquina virtual.
  - **TURN**: tiempo de respuesta requerido.
- De personal
  - **ACAP**: calificación de los analistas.
  - **AEXP**: experiencia del personal en aplicaciones similares.
  - **PCAP**: calificación de los programadores.
  - **VEXP**: experiencia del personal en la máquina virtual.
  - **LEXP**: experiencia en el lenguaje de programación a usar.
- De proyecto
  - **MODP**: uso de prácticas modernas de programación.
  - **TOOL**: uso de herramientas de desarrollo de software.
  - **SCED**: limitaciones en el cumplimiento de la planificación.

$$E = a(Kl)^b * m(X), \text{ en persona-mes}$$
$$Tdev = c(E)^d, \text{ en meses}$$
$$P = E/Tdev, \text{ en personas}$$

Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

- Útil para hacer estimaciones que...

  …es de lo que huyen las metodologías ágiles (las de medio y largo plazo
- Pero es lo que quiere el cliente

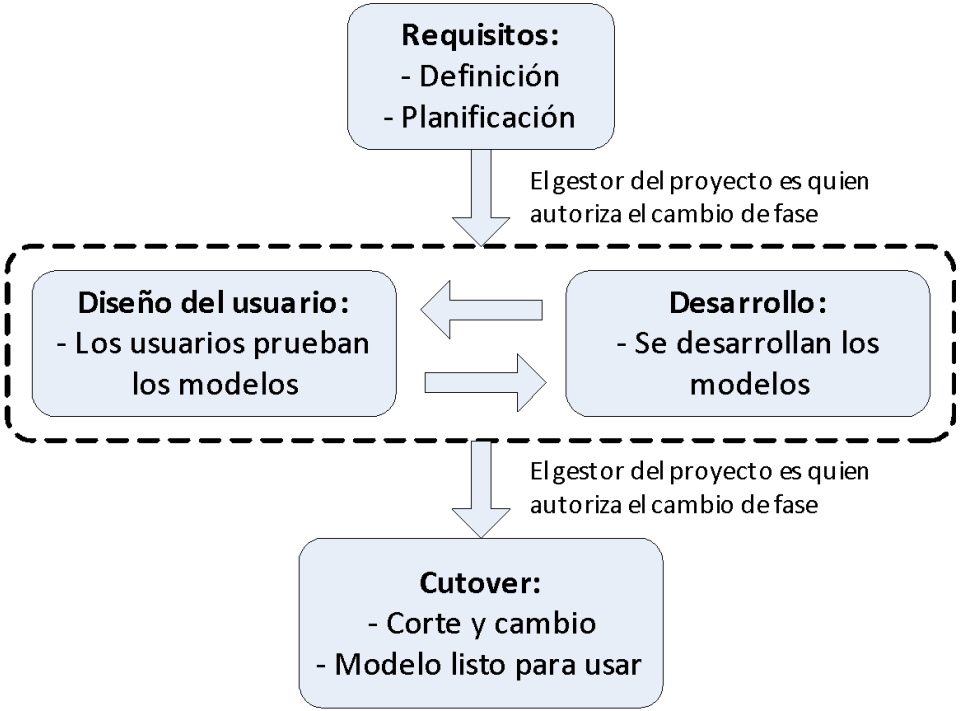
El problema de encontrar una métrica adecuada

- Paradoja de la carrera de relevos

- ¿Cuántas personas corren en un equipo?
- ¿Cuál es el objetivo?
- ¿Qué porcentaje de tiempo está corriendo cada deportista?
- ¿Cómo debemos medir la “productividad” de cada deportista?

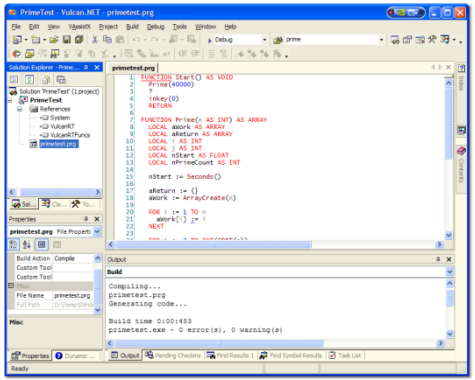
Rapid Application Development

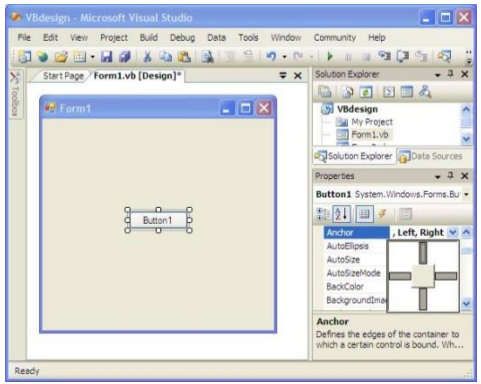
- RAD: desarrollo rápido de aplicaciones
  - Primera aproximación en la que se hace participar al cliente
  - Diferentes roles
  - Cliente, gestor y desarrolladores
  - Diferentes fases… y ciclos



Rapid Application Development

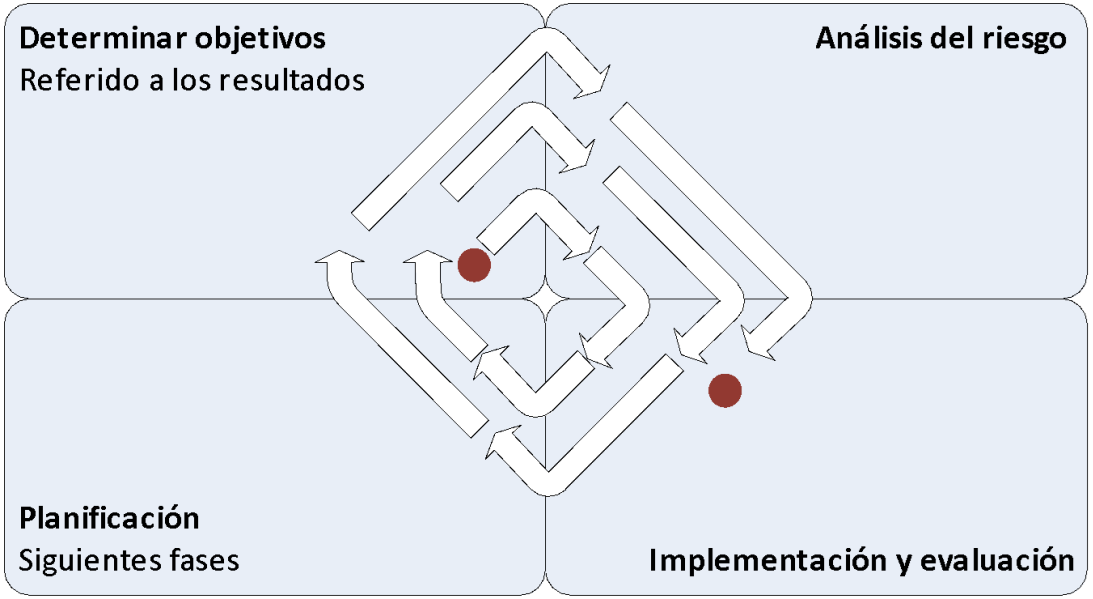
- RAD: desarrollo rápido de aplicaciones
  - Cliente involucrado
  - Aparecen los ciclos de desarrollo
- En la gestión clásica los ciclos
  - No son predecibles o son difíciles de estimar
  - Son difíciles de modelar
  - Se puede estimar el resultado final (si se sabe a dónde se quiere llegar)
  - Pero no se puede estimar qué pasa con cada ciclo (no sabemos las dificultades con las que nos encontraremos)
- Herramientas CASE (Ingeniería Asistida por Ordenador)





Desarrollo en espiral

- El bucle es completo e incluye todas las fases
- Se basa en el modelo de prototipado
  - Cada iteración enriquece el prototipo
  - Las fases de cada iteración son las mismas (hay varias versiones)



eXtreme Programming

- eXtreme Programming (XP)
  - Adaptabilidad: los requisitos deben cambiarse a medida que el proyecto avanza

- Valores
  - Sencillez: en todos los aspectos (documentación, diseño…)
  - Comunicación: abierta
  - Aprendizaje: retroalimentación
  - Valentía: frente a los cambios
  - Respeto: confianza (hoy no me enfado)

eXtreme Programming

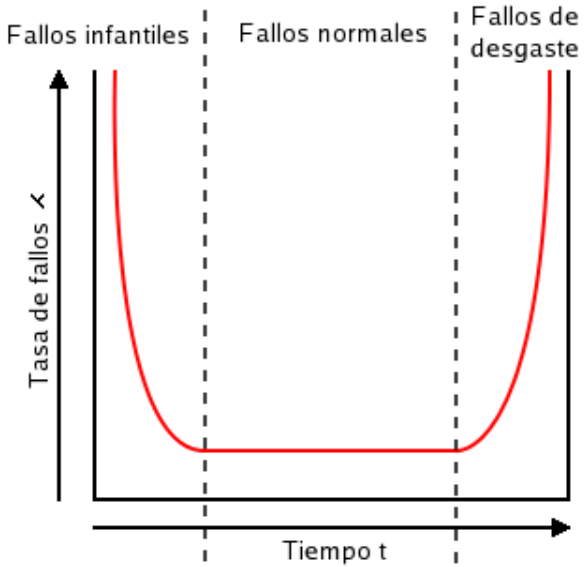
- Desarrollo iterativo
  - Desde segundos a meses
  - Cada iteración, una mejora.
- Adaptable a metodologías predictivas

eXtreme Programming

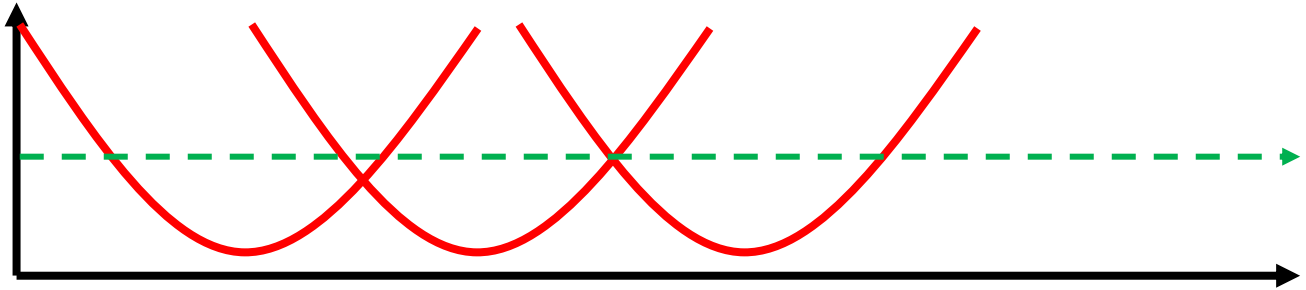
- Desarrollo en parejas
  - En algunos casos se permite grupos o individual.
  - Simplicidad en los productos.
- Regresión
  - Primero se desarrolla el sistema de evaluación.
  - Luego se desarrolla el producto.
- Pruebas continuas
  - No pueden haber defectos antes de pasar a la siguiente iteración.
- Roles
  - Desarrolladores, testadores, entrenadores, clientes, etc.

Rational/Open Unified Process

Concepto de “death valley”

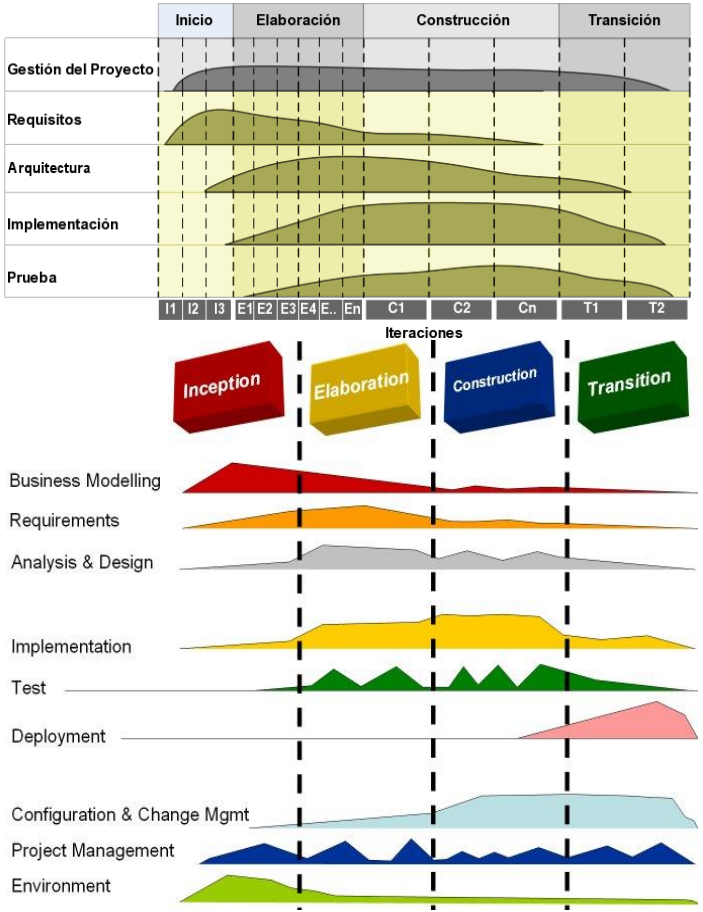


Optimización de las etapas

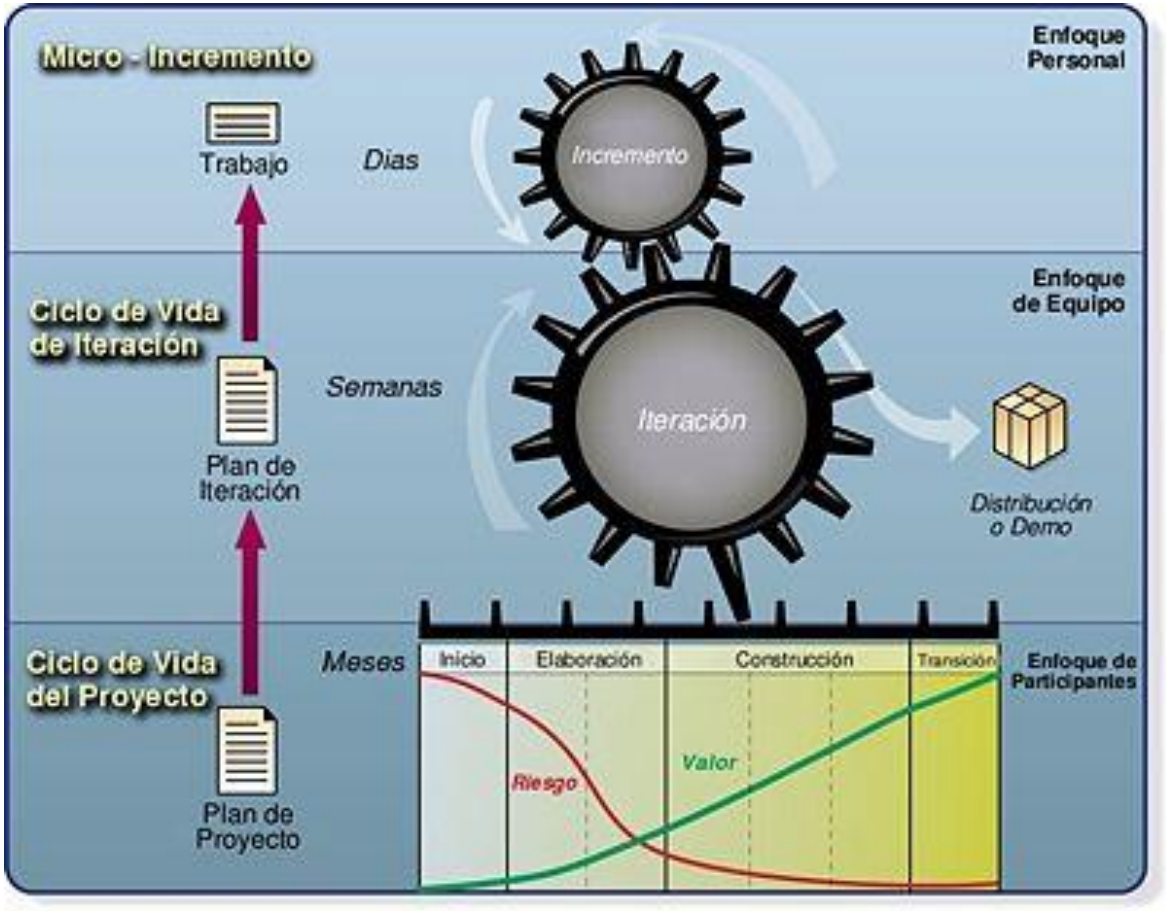


Rational/Open Unified Process

- Fases
  - Inicio
  - Elaboración
  - Construcción
  - Transición
- Diagrama de fase/esfuerzo
- Hay muchas variaciones

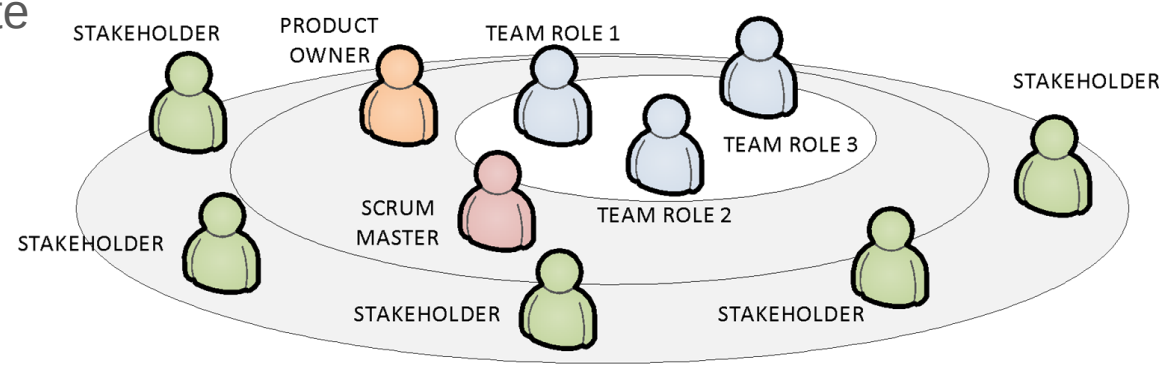


Rational/Open Unified Process



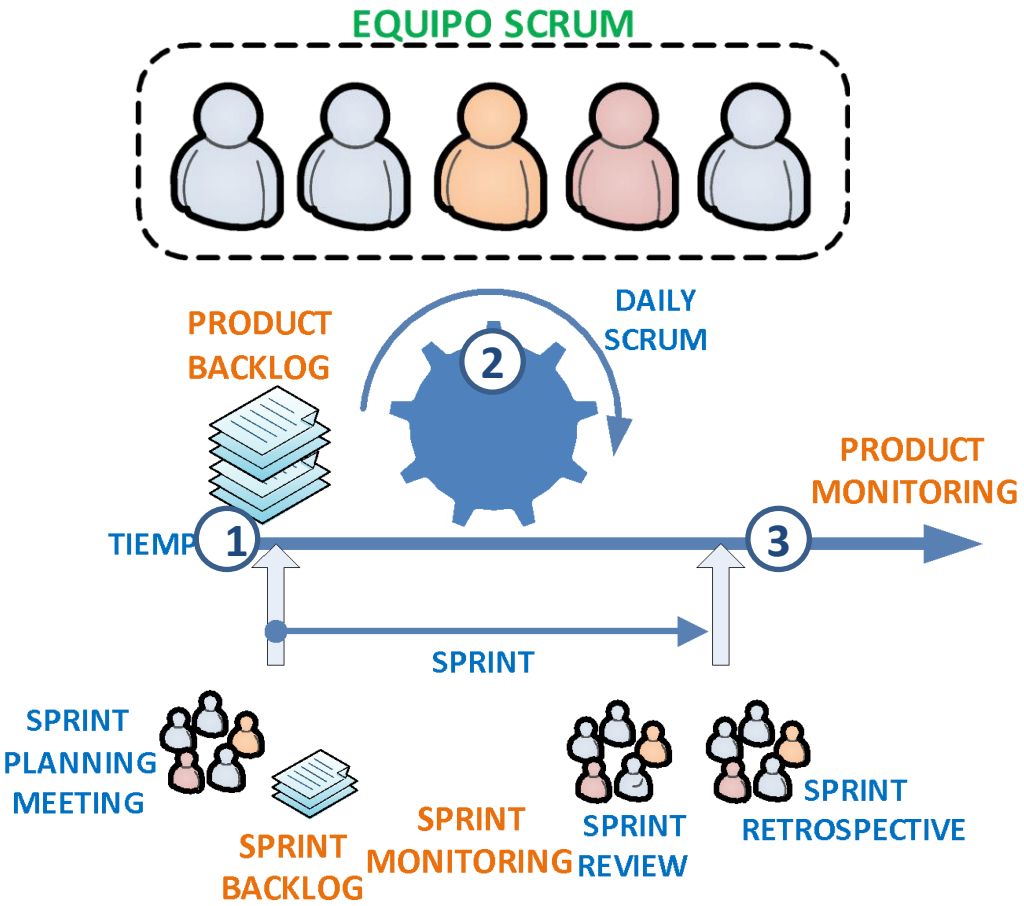
SCRUM

- Scrum Master (facilitador)
  - Genera el entorno de trabajo
  - Elimina obstáculos
  - NO es el líder
- Componente del equipo
  - Colabora con el resto de componentes
  - El trato siempre es de igual a igual
- Product Owner (interfaz con el mundo)
  - Da la visión del cliente
  - NO es el cliente
- Stakeholders
  - Clientes
  - Proveedores
  - Etc..



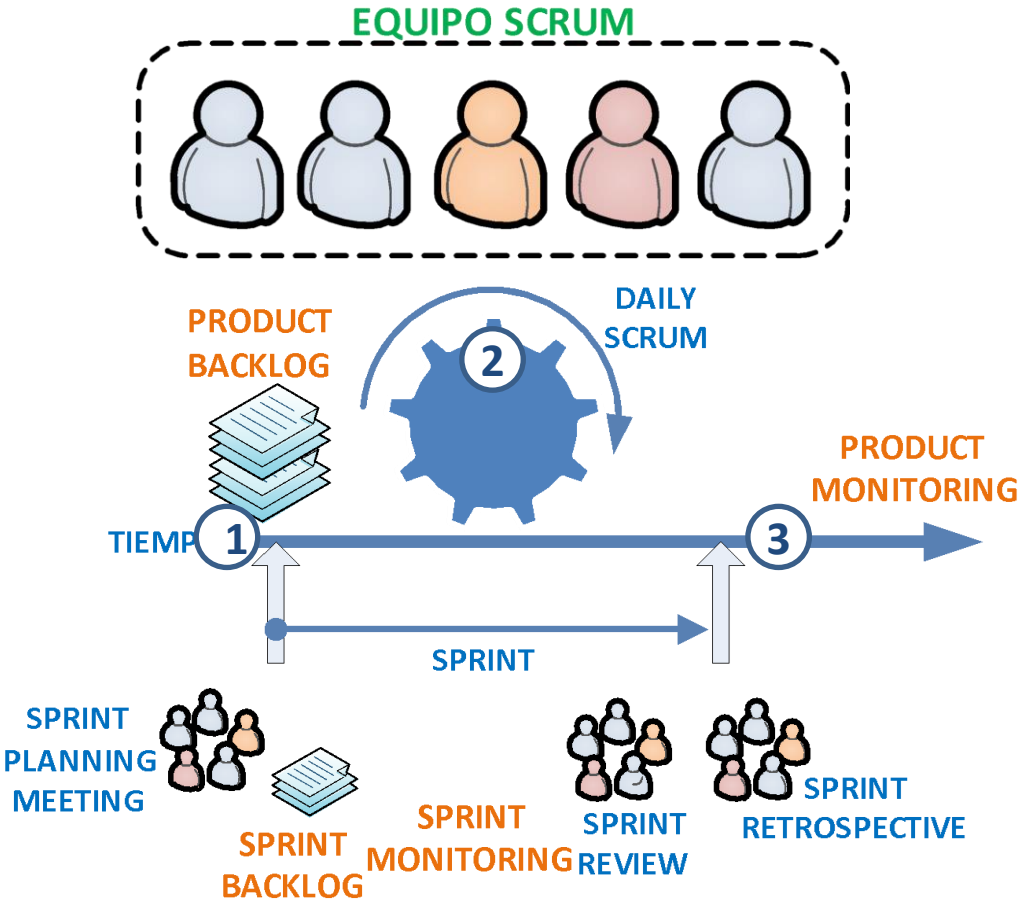
SCRUM

- Equipo
  - Estructurado: roles
  - Con plena capacidad de decisión
  - Igualitario
- Eventos
  - Sprint: contenedor
  - Duración definida
  - Reuniones: mínimas
- Artefactos
  - Representan trabajo o valor
  - Transparentes
  - Documentación: mínima



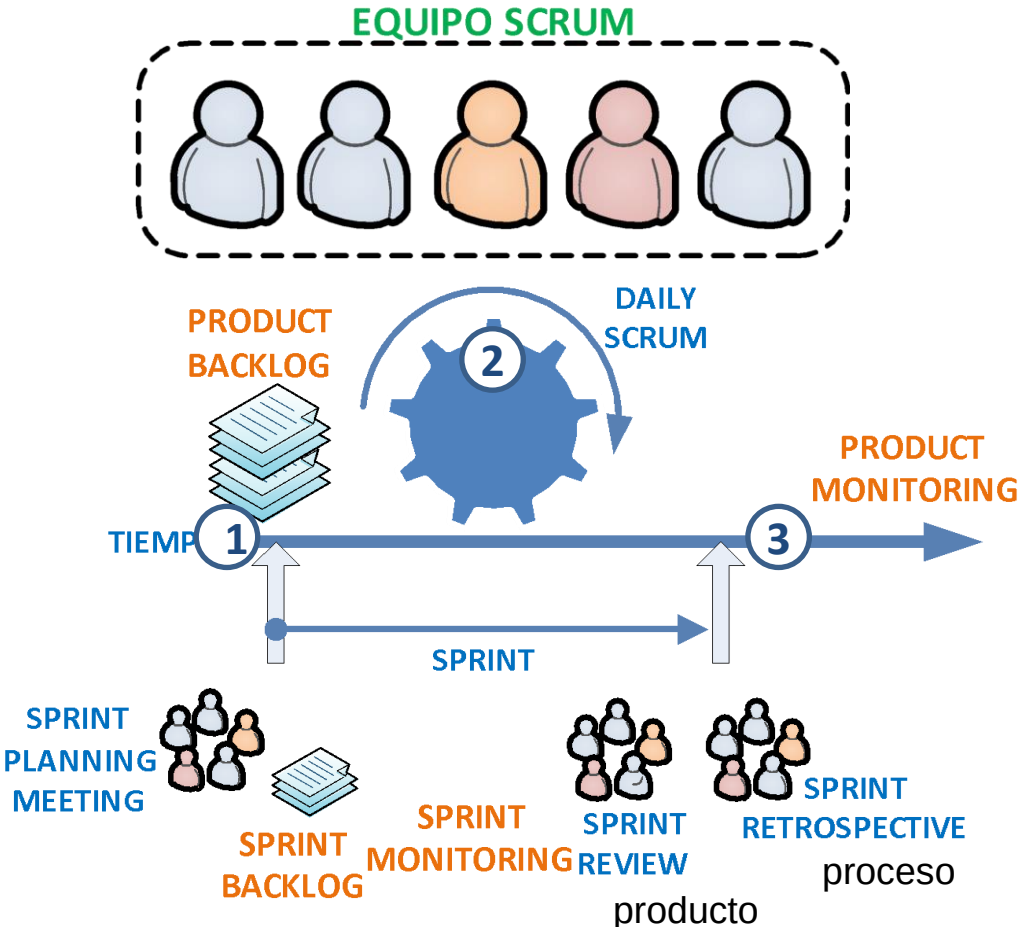
SCRUM

- SPRINT: Periodo en el que se realiza el trabajo
  - Reuniones: frecuentes, periódicas y cortas
- Daily-Scrum (standup meeting)
  - Reunión de mayor frecuencia
  - Reunión muy corta
- El esquema está prefijado
  - ¿Qué se hizo ayer?
  - ¿Qué problemas surgieron?
  - ¿Qué se va a hacer hoy?
- ¿Documentar la reunión?



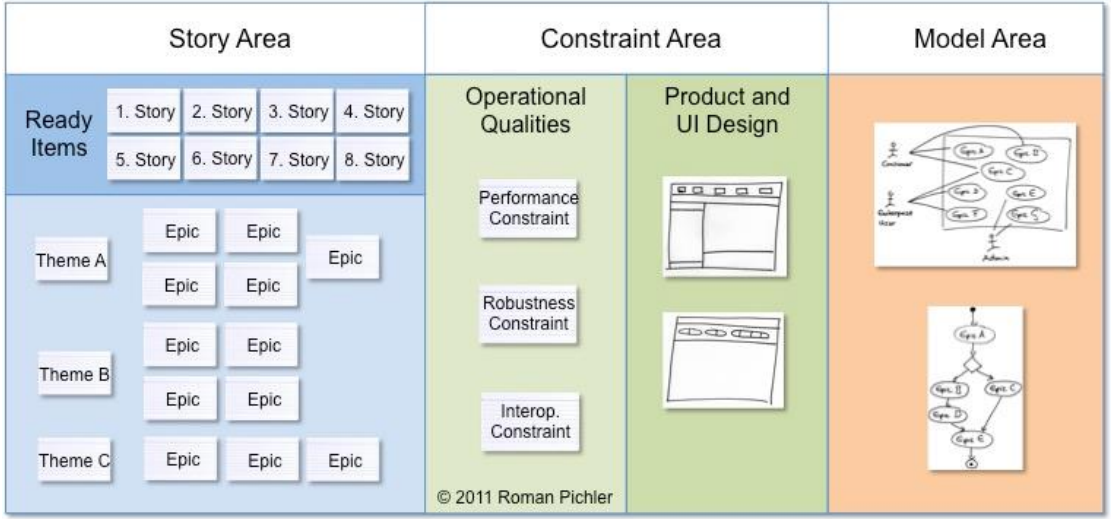
SCRUM

- Inicio
  - Planificación del Sprint (Sprint Planning Meeting)
  - Trabajo a realizar.
  - Preparar los requisitos a lograr en el sprint (Sprint Backlog)
  - Controlar el tiempo que dichos requisitos van a precisar
  - Identificar y comunicar cuánto del trabajo final
- Final:
  - Revisión del Sprint (Sprint Review Meeting)
  - Trabajo completado y no completado
  - Presentación del trabajo a los interesados: demostraciones
- Retrospectiva del Sprint (Sprint Retrospective)
  - Impresiones sobre el sprint finalizado
  - Mecanismo de mejora continua
  - ¿Documentar la reunión?



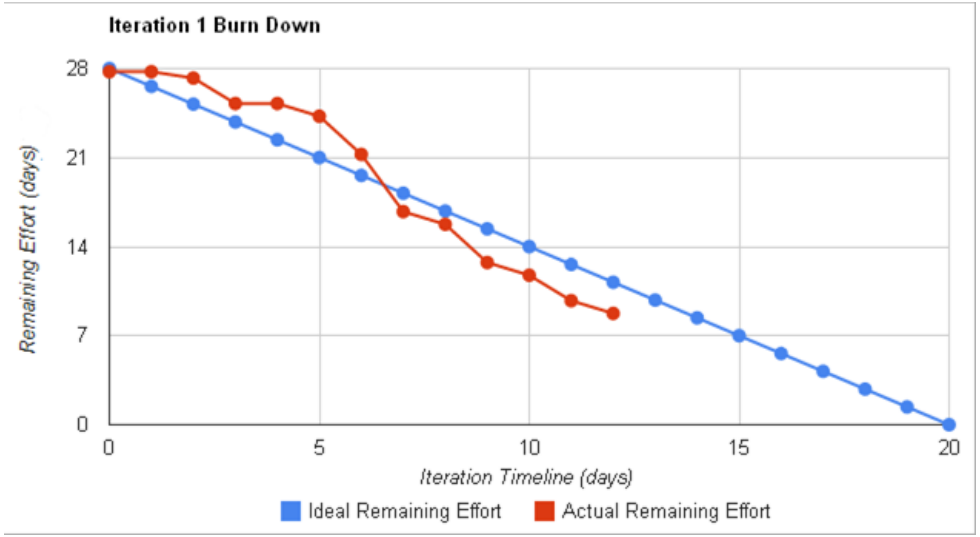
SCRUM

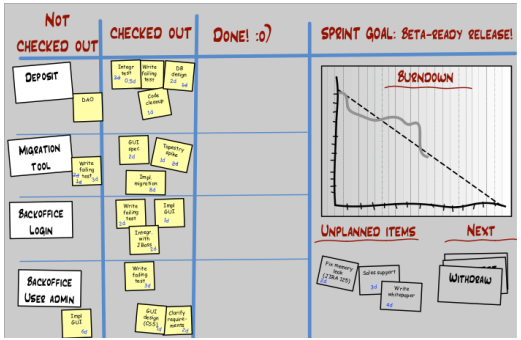
- Product backlog
  - Documento con todos los requisitos del proyecto
  - Muy genérico
  - Funcionalidades deseables
  - Priorizadas (ROI: return on investment)
  - A ser posible con diagramas, figuras… gráficos
  - Libre acceso
  - Sólo lo modifica el Product Owner.
  - Contiene la carga temporal
  - Prioriza las tareas



SCRUM

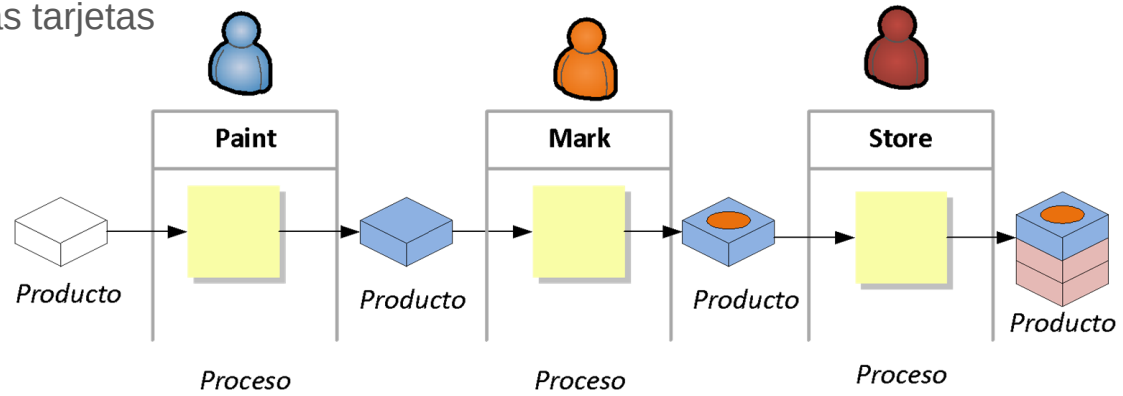
- Burn down chart
  - Gráfica pública que muestra el avance del proyecto
  - Cantidad de requisitos del Backlog pendientes al comienzo de cada sprint
  - Descendente: los requisitos se están completando
  - Ascendente: se están añadiendo nuevos requisitos





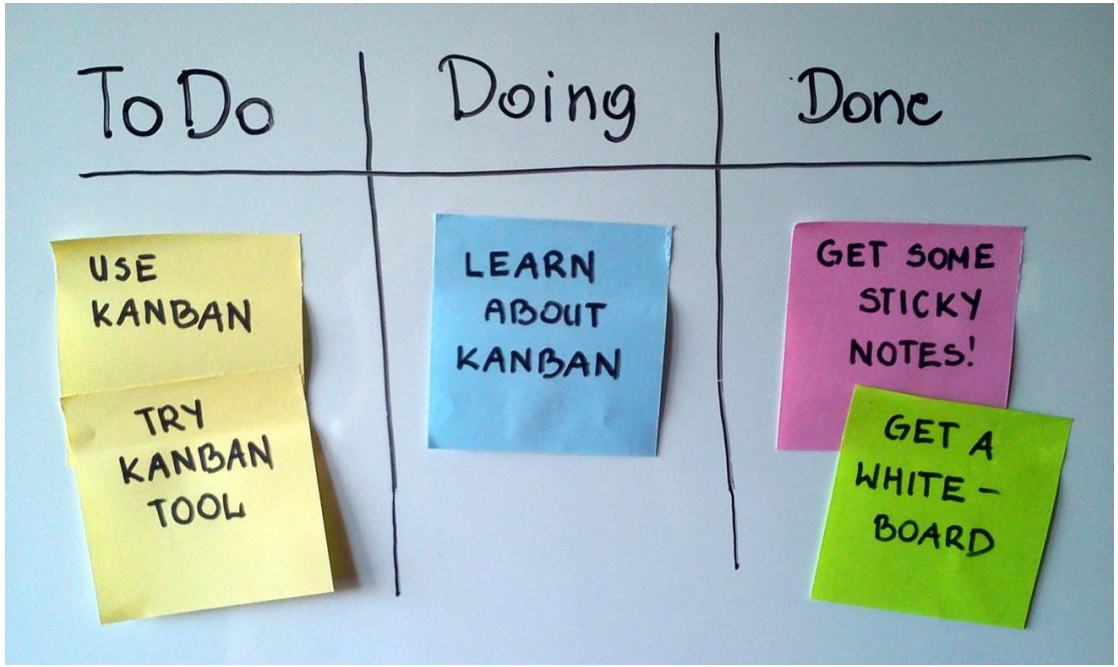
KANBAN

- Limitar las tareas que se hacen a la vez
  - Las tareas se deben empezar lo antes posible
  - Se debe buscar el equilibrio
- El equipo debe poder interactuar con todas las tareas
  - Mecanismo de mejora continua
- Definición de procesos del proyecto
  - Emplear la misma “representación” de todos los procesos del proyecto:
  - Método basado en la gestión por “tarjetas”
  - Información común en todas las tarjetas
    - Tareas
    - Responsables
    - Comentarios
- Lugar común de las tareas
  - Kanban Board



KANBAN

- Tablero Kanban
  - Representa las tareas que se están realizando
  - WIP: work In Progress
  - Flujo de valor
  - Medición del beneficio que el proyecto está aportando



Para profundizar más

Para ver

http://www.ted.com/talks/bruce_feiler_agile_programming_for_your_family

Para ver

The art of doing twice as much in half the time | Jeff Sutherland | TEDxAix

https://www.youtube.com/watch?v=s4thQcgLCqk

Para visitar

http://www.devagile.com/

Agile-Spain

http://agile-spain.org/

Scrum Alliance®

https://www.scrumalliance.org/

Agile Alliance

http://www.agilealliance.org/

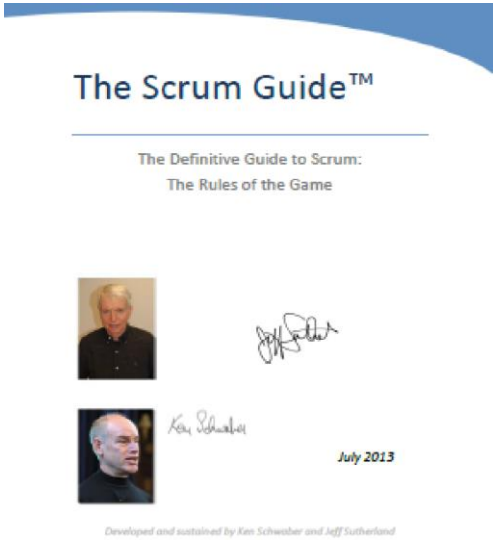
Extreme Programming

http://www.extremeprogramming.org/

Para leer

http://www.navegapolis.com

www.proyectalis.com



www.scrumguides.org

https://proyectosagiles.org/2009/08/15/scrum-proceso-trabajo-2-0/

Para escuchar

Gestión de proyectos ágiles con Alejandro Alonso, CIO de Kaleidos y cofundador de Taiga: http://www.ivoox.com/gestion-proyectos-agiles-alejandro-alonso-cio-audios-mp3_rf_3648023_1.html

Podcast de la plataforma de conocimiento abierto de Scrum Manager (http://www.scrummanager.net)

http://www.ivoox.com/podcast-podcasts_sq_f12397_1.html