# Glosario - Open Knowledge Scrum

Volver a: Sitio

Versión para impresión

Glosario

Buscar

¿Buscar en conceptos y definiciones?

Navegue por el glosario usando este índice.

Especial | A | B | C | D | E | F | G | H | I | J | K | L | M | N | Ñ | O | P | Q | R | S | T | U | V | W | X | Y | Z | TODAS



# A

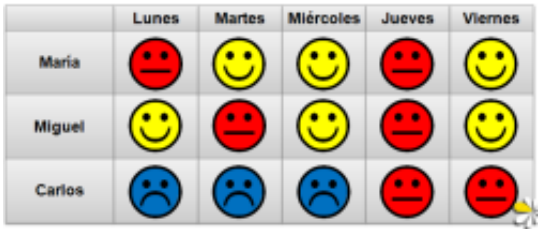
## Agile Unified Process

Es una versión simplificada y ágil de RUP, desarrollada por Scott W. Ambler: aplica las siete disciplinas de RUP (modelado, implementación, prueba, desarrollo, gestión de la configuración, gestión de proyecto y entorno), pero dentro de ejecuciones iterativas, y sobre una base de cultura de desarrollo ágil.

Página del modelo

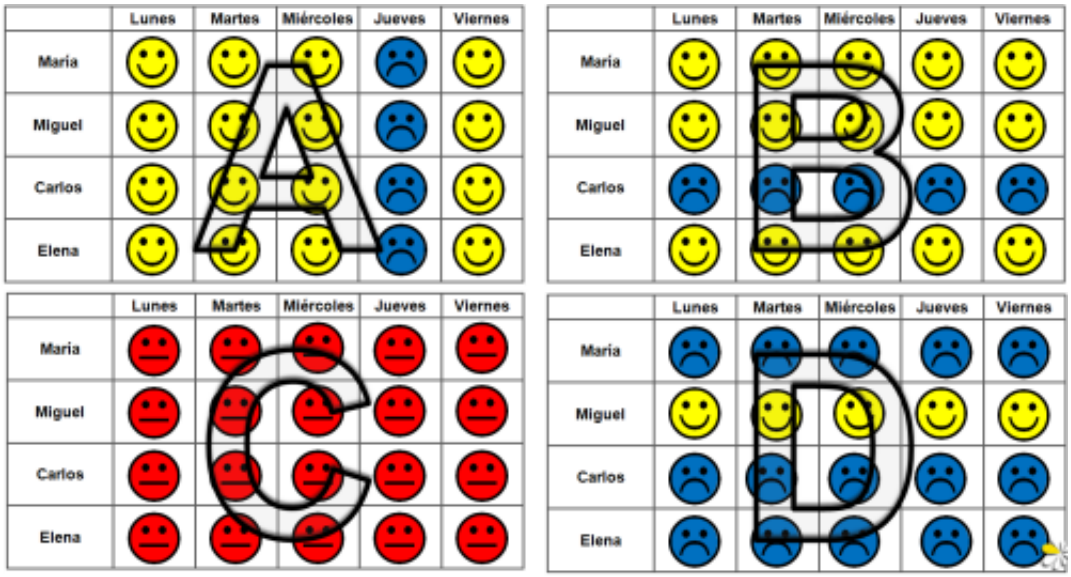
# C

## Calendario Niko




Propuesto por Akinori Sakata, es un indicador del nivel de motivación del equipo.
Se sitúa en un lugar visible de la oficina, preferiblemente junto al resto de indicadores (gráfico de avance, pizarra kanban...) y cada día los miembros del equipo ponen un adhesivo o dibujan uno de los tres "smiles" posibles, reflejando su estado de ánimo con respecto a las tareas del proyecto.

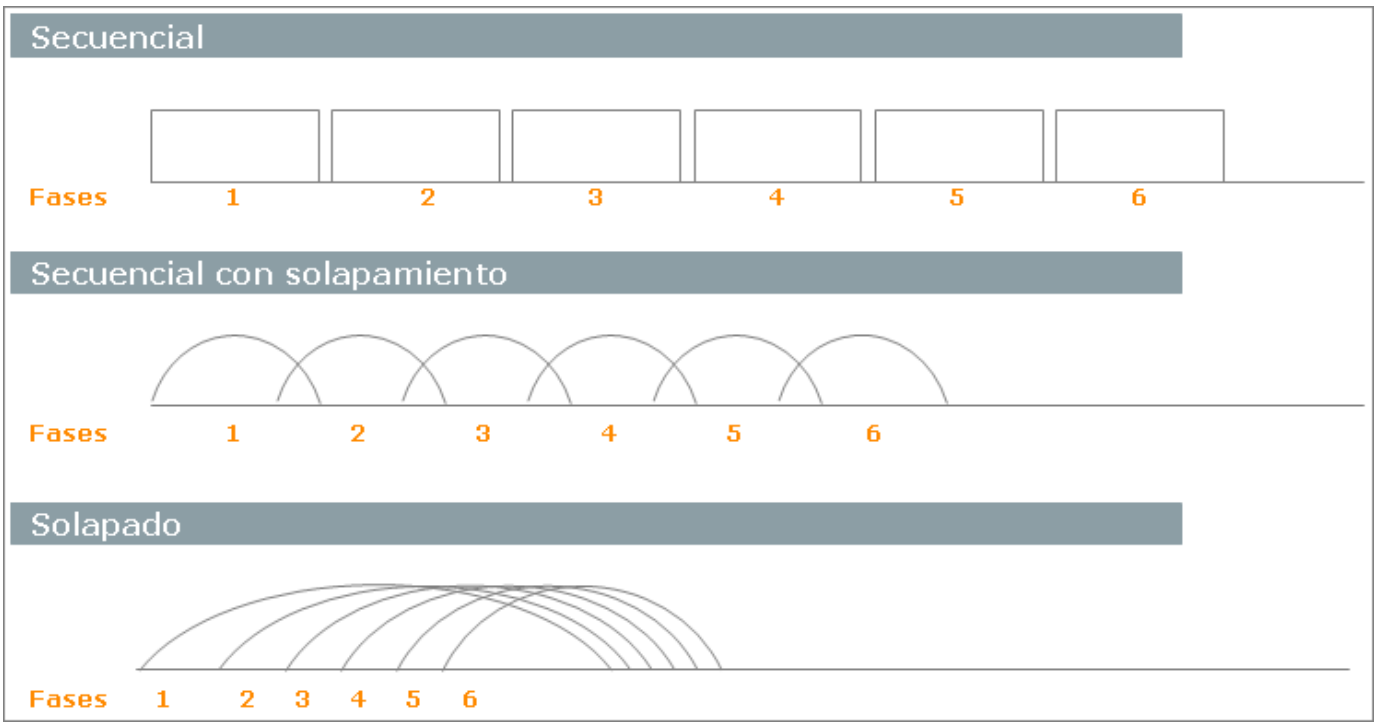
Permite detectar configuraciones que revelan situaciones tipo:




## Campo de Scrum

Término acuñado por Ikujiro Nonaka e Hirotaka Takeuchi para describir la forma de trabajar que analizan en su artículo "The New New Product Development Game", y que descubren al analizar, en 1986, los modos empleados por empresas que con mejores resultados que su competencia: Fuji-Xerox, Canon, Honda, Nec, Epson, Brother, 3M y Hewlet Packard.

La principal diferencia que caracteriza a los equipos que "*avanzan en un campo de scrum*" es el solapamiento de las fases de desarrollo.




Características de un Campo de Scrum:

- **Incertidumbre**, como elemento consustancial y asumido en la cultura de la organización.

Se trabaja con una visión genérica del objetivo que se debe lograr y la dirección estratégica que hay que seguir. No con un plan detallado del producto y su ejecución. El equipo dispone de un margen amplio de libertad.

- **Equipo de trabajo auto-organizado.**

Los equipos de los campos de scrum son equipos auto-organizados, sin roles de gestión para marcar pautas o asignar tareas. Son similares a una pequeña empresa, en la que todos los integrantes trabajan de forma conjunta y auto-organizada, sin patrones organizativos impuestos desde una estructura empresarial ajena al equipo.

- **Solapamiento de las fases de desarrollo.**

Las fases pasan a ser actividades. El concepto de fase implica sucesión secuencial. En un campo de scrum los trabajos pierden el carácter de fase; son actividades que se realizan en cualquier momento, de forma simultánea o a demanda, según las necesidades en cada momento. En el desarrollo secuencial las transiciones entre fases acaban siendo fronteras. Cada fase la realiza un equipo que siente como responsabilidad su parte, pero no el proyecto en su conjunto. Los documentos de diseño, requistios, prototipos... pueden acabar siendo barricadas, que lejos de favorecer la comunicación directa, favorecen la separación. El retraso en una fase es un cuello de botella para el proyecto. El solapamiento diluye el ruido y los problemas entre fases.

Control sutil

El equipo trabaja con autonomía en un entorno de ambigüedad, inestabililidad y tensión. La gestión establece puntos de control suficientes para evitar que el ambiente de ambigüedad, inestabilidad y tensión derive en descontrol, pero no ejerce un control rígido que impediría la creatividad y la espontaneidad. El término "control sutil" se refiere a generar el ecosistema adecuado para un "auto-control entre iguales", consecuencia de la responsabilidad y del gusto por el trabajo que se realiza. Las acciones para generar el ecosistema son:

- Selección de las personas adecuadas.
- Espacio de trabajo abierto.
- Animar a los ingenieros a mezclarse con el mundo real de las necesidades de los clientes.
- Reconocimiento basado en el rendimiento del equipo.
- Gestión de las diferencias de ritmo con el proceso de desarrollo.
- Tolerancia y previsión con los errores: son un medio de aprendizaje, y el miedo al error merma la creatividad y la espontaneidad

  Implicación de los proveedores en el proyecto.
- Difusión y transferencia del conocimiento

Tanto a nivel de proyecto como de organización.

Se trata de organizaciones basadas en equipos multi-disciplinares, en los que todos los miembros aportan y aprenden tanto del resto del equipo como de las investigaciones, innovaciones del producto y de la experiencia acumulada.

Las personas que participan en un proyecto, con el tiempo van cambiando de equipo en la organización, a otros proyectos; de esta forma se van compartiendo y comunicando las experiencias en la organización.

Los equipos y las empresas mantienen libre acceso a la información, herramientas y políticas de gestión del conocimiento.

## Cascada

Ver "Ingeniería secuencial"

## Cerdo

En el contexto de gestión ágil de proyectos, el término "cerdo" hace referencia a un rol "implicado" con el resultado.

**Una gallina y un cerdo paseaban por la carretera. La gallina dijo al cerdo: "Quieres abrir un restaurante conmigo". El cerdo consideró la propuesta y respondió: "Sí, me gustaría. ¿Y cómo lo llamaríamos?". La gallina respondió: "Huevos con beicon".**

**El cerdo se detuvo, hizo una pausa y contestó: "Pensándolo mejor, creo que no voy a abrir un restaurante contigo. Yo estaría realmente comprometido, mientras que tu estarías sólo implicada".**

## CMMI

Modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software.

Su desarrollo se inicia en los años 80 partiendo del principio de calidad de Jurán de solvencia contrastada en la producción industrial: la calidad del resultado depende de la calidad de los procesos empleados en su construcción.

# D

## Desarrollo ágil

El desarrollo ágil se fundamenta en ritmos de planificación y ejecución inter-dependientes. Es un esquema conceptualmente simple, pero que requiere un trabajo relativamente complejo al implementarlo en la realidad de cada organización, hasta institucionalizar la serie de reuniones, prácticas y momentos de entrega que permite en cada caso encontrar su propio ritmo ágil.

Los ciclos del desarrollo ágil:

- **Estratégico**: El proyecto comienza con una visión asociada a una necesidad o línea de negocio. Esta visión, junto con la estrategia y las metas que de ella se derivan, establece el área de referencia de todas las sesiones y decisiones del proyecto.

- **Versión**: Cada ciclo de versión comprende un punto de avance significativo en el valor del producto o servicio desarrollado. Según los proyectos y las organizaciones pueden ser ciclos de 2, 4, 6, 8 (?) meses. Cada versión se realiza en un ciclo de versión y comprende uno o varios hitos de producto ("epics" o "grandes historias de usuario").

- **Iteración**: También denominado "sprint". Ciclo de construcción en el que se completa un "incremento" del producto o servicio. Suelen ser ciclos de 1 a 6 semanas.

- **Diario**: El seguimiento del avance del trabajo en las iteraciones lo revisa el equipo de forma auto-gestionada a diario, compartiendo, planificando el trabajo que se va a realizar en ese día, el trabajo que que queda para terminar la Iteración, y si es necesario resolver algún impedimento.

## Deuda técnica

Término acuñado por Martin Fowler (Technical Debt) para describir que la modificación de un programa, normalmente se puede hacer de dos maneras: una rápida pero "sucia" y otra más lenta y limpia.
La primera reduce la calidad del código y supone adquirir una deuda (deuda técnica) cuyos intereses pagaremos irremediablemente.
La refactorización es un método válido para evitar la deuda técnica.

## Disponibilidad de valor

Objetivo de la gestión ágil, consistente en entregar el mayor valor operativo para el cliente, en el menor tiempo posible, y a partir de ese momento, incrementar de forma continua el valor del producto.

## DSDM

DSDM es el acrónimo que da nombre a un modelo de procesos para el desarrollo de sistemas de software, desarrollado y concebido por el denominado DSDM Consortium, que se fundó en Inglaterra en 1994, y que actualmente tiene presencia en Inglaterra, EE.UU. Benelux, Dinamarca, Francia y Suiza; y con interés y contactos para futuras representaciones en Australia, India y China [...]

Es un modelo que estuvo representado en la firma del Manifiesto Ágil. Arie van Bennekum, firmante del manifiesto, era miembro del consorcio en Benelux, consultor y formador de DSDM.
En 2001, año del Manifiesto Ágil, DSDM publicó la versión 4.1 de su modelo, y se consideró una metodología ágil; y aunque mantuvo las siglas, cambió la denominación original (Dynamis Systems Development Method) por Framework for Business Centred Development.

El modelo es propiedad del DSDM Consortium y solo sus miembros pueden emplearlo con fines comerciales.

# Origen

Tomando como fuente las bases teóricas y las experiencias de RAD (Rapad Application Development) el consorcio se fundó en enero de 1994 con la finalidad de desarrollar un modelo de desarrollo independiente de herramientas y que fuera de dominio público.
Ed Holt, presidente del consorcio en sus dos primeros años, afirmó que el uso de herramientas RAD estaba necesitando un marco de procesos.

La primera versión del modelo se publicó a principios de 1995, junto con un programa de adopción temprana para obtener retro-información de las primeras organizaciones que lo adoptaran.
Con la información y experiencia que el consorcio iba obteniendo se publicó la versión 2 en noviembre de 1995, y la 3 en agosto de 1997.

# Principios

En su versión actual (4.2) el marco de procesos DSDM se basa en 9 principios.

- La implicación activa de los usuarios es imprescindible.
- Los miembros de los equipos de desarrollo DSDM deben tener la autonomía y potestad necesarias para tomar decisiones.
- Entrega frecuente de incrementos operativos del producto.
- El principal criterio de prioridad, desarrollo y validación de las entregas incrementales es el objetivos y la salud del negocio.
- El desarrollo iterativo o incremental hace posible obtener la solución más adecuada a las necesidades del negocio.
- Todos los cambios realizados en el desarrollo son reversibles.
- Los requisitos se establecen a un nivel general
- Las pruebas forman parte del ciclo de desarrollo
- Es imprescindible trabajar con espíritu de colaboración con todos los agentes implicados en el sistema que se desarrolla.

# Procesos del ciclo de desarrollo DSDM

El ciclo de desarrollo de DSDM está compuesto de 5 fases, precedidas de un pre-proyecto y un post-proyecto.

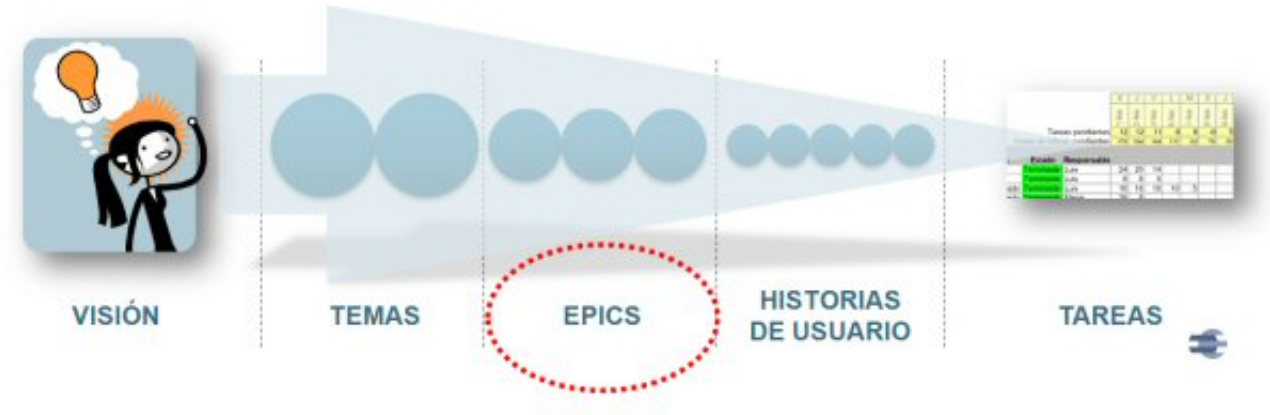
1. Pre-proyecto
2. Estudio de viabilidad
3. Estudio de negocio
4. Iteración de modelado funcional
5. Iteración de diseño y desarrollo
6. Implementación
7. Post-desarrollo

Los cinco procesos centrales se suelen representar con el siguiente gráfico, familiarmente denominado "de las 3 prizzas y el queso".


Es frecuente que DSDM se implante en combinación con Exteme Programming o de Prince2

# E

## Epic




Especificación de requisitos ágil que comprende varias historias de usuario, y describen funcionalidades genéricas del sistema. Es habitual referirse a los como "grandes historias de usuario"

## Extreme programming

eXtreme Programming (XP) o programación extrema es un conjunto de prácticas ágiles formuadas por Kent Beck en el libro que dio origen a su difusión como metodología ágil: "Extreme Programming Explained: Embrace Change" (1999).

Extreme Programming considera que las modificaciones de requisitos constantes son un aspecto natural, inevitable e incluso deseable en los desarrollos de software, configurándose de esta forma (como todos los modelos ágiles) en un modelo enfocado en la adaptabilidad antes que en la previsibilidad.

Los principios originales de la programación extrema son: simplicidad, comunicación, retroalimentación y coraje. En la segunda edición de Extreme Programming Explained, se añadió un quinto principio: respeto.

# F

## Flexibilidad

Principio de identidad de Scrum Manager: Las prácticas deben adaptarse a las características de la organización y de los proyectos en los que trabaja, y no al revés.

# G

## Gallina

En el contexto de gestión ágil de proyectos, el término "gallina" hace referencia a un rol "comprometido" con el resultado.

**Una gallina y un cerdo paseaban por la carretera. La gallina dijo al cerdo: "Quieres abrir un restaurante conmigo". El cerdo consideró la propuesta y respondió: "Sí, me gustaría. ¿Y cómo lo llamaríamos?". La gallina respondió: "Huevos con beicon".**

**El cerdo se detuvo, hizo una pausa y contestó: "Pensándolo mejor, creo que no voy a abrir un restaurante contigo. Yo estaría realmente comprometido, mientras que tu estarías sólo implicada".**

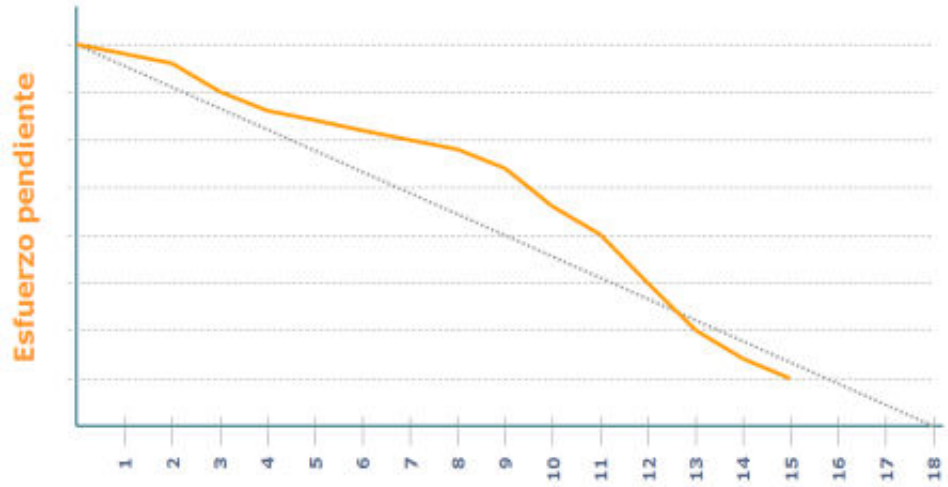
## Gráfico de avance

En inglés burndown chart.

Es un gráfico cartesiano que representa el esfuerzo pendiente para terminar una lista de trabajos.

La abcisa representa el tiempo: normalmente iteraciones, cuando se emplea para representar el avance del producto; o días cuando se emplea para seguir el avance de una iteración.

La ordenada representa el esfuerzo en puntos, tiempo ideal o en la unidad que emplee el equipo.




Con este gráfico da forma a dos claves del seguimiento ágil de proyectos:

- Medir el esfuerzo pendiente, no el realizado.
- Seguimiento muy cercano (diario a ser posible)

## Gráfico de producto

En inglés burnup chart

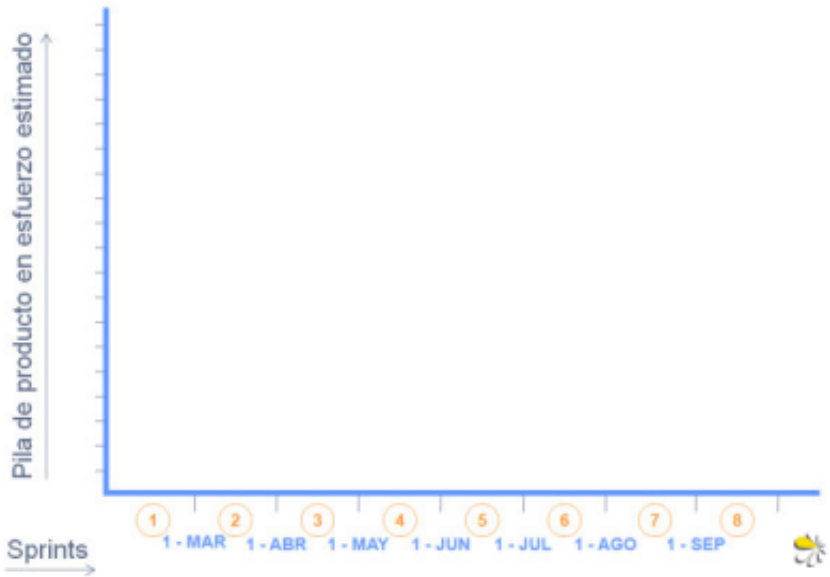
Herramienta de gestión y seguimiento propia del propietario del producto. Presenta en un vistazo las funcionalidades comprendidas en cada versión de producto, la velocidad de avance estimada, la fecha probable de cada versión, el margen de error previsto y el avance real.

Se confecciona a partir de:

La estimación de esfuerzo prevista en la pila de producto.

La velocidad del equipo.

El eje Y representa el esfuerzo, y sobre él se marcan los hitos de versiones previstas en el backlog.




El eje X representa el tiempo de desarrollo con las fechas de los sprints previstos. En el área del gráfico se proyecta la línea que representa la velocidad de desarrollo del equipo. Este dato se obtiene sobre el histórico de velocidad desarrollada por el mismo equipo en proyectos o sprints anteriores. Si no se tiene información histórica, un buen dato para comenzar es utilizar “tiempo real” como unidad para el esfuerzo y la velocidad (horas o días reales) y suponer como velocidad del equipo un tercio del tiempo disponible de trabajo.

# Ejemplo

Representación del plan del producto, a partir de los temas previstos en el product backlog.

Convenciones empleadas por el equipo de este ejemplo:

- Unidad para medición de trabajo: tiempo ideal
- Tiene previsto realizar sprints de duración fija mensual
- El equipo está formado por 4 personas, y viene desarrollando una velocidad de 300 puntos por sprint (300 horas ideales de trabajo)

| Id | Historias | Trabajo | Criterio de validación |
|---|---|---|---|
| 1 | Historia A 1.0 | 150 | Lorem ipsum dolor sit amet |
| 2 | Historia B 1.0 | 250 | consectetuer adipiscing elit |
| 3 | Historia C 1.0 | 250 | Aliquam vehicula accumsan tortor |
| 4 | Historia D 1.0 | 300 | Pellentesque turpis |
| 5 | Historia A 1.1 | 250 | Phasellus purus orci |
| 6 | Historia D 1.1 | 350 | penatibus et magnis dis partur |
| 7 | Historia E 1.0 | 150 | Quisque volutpat ante sit amet velit |
| 8 | Historia B 1.1 | 500 | Cras iaculis pede eu tellus |
| 9 | Historia C 1.1 | 150 | Vestibulum vel diam sed pede |
| 10 | Historia E 1.1 | 200 | Suspendisse aliquam felis et turpis |
| 11 | Historia F 1.0 | TBD | Nullam imperdiet lorem vitae justo |
| 12 | Historia A 1.2 | TBD | Suspendisse potenti. In nec nunc |
| 13 | Historia B 1.2 | TBD | Nam eros tellus, facilisis sed, pretium |
| 14 | Historia F 1.1 | TBD | Morbi arcu tellus, condimentum |



La figura de la derecha representa la situación actual de la pila de producto: el propietario del producto tiene previsto cerrar la versión 1.0 al disponer de los cuatro primeros temas, y su estimación inicial de trabajo para llevarlos a cabo es de 950 puntos.

La versión 1.2 incluirá 3 temas más que, según la estimación inicial, supondrán unos 750 puntos de trabajo.

Y están también trazados los temas con los que piensa cerrar la versión 1.3 que se prevén con 850 puntos más de trabajo.

Para representar el plan del producto con un gráfico de producto, se representan, con los fondos de escala apropiados:

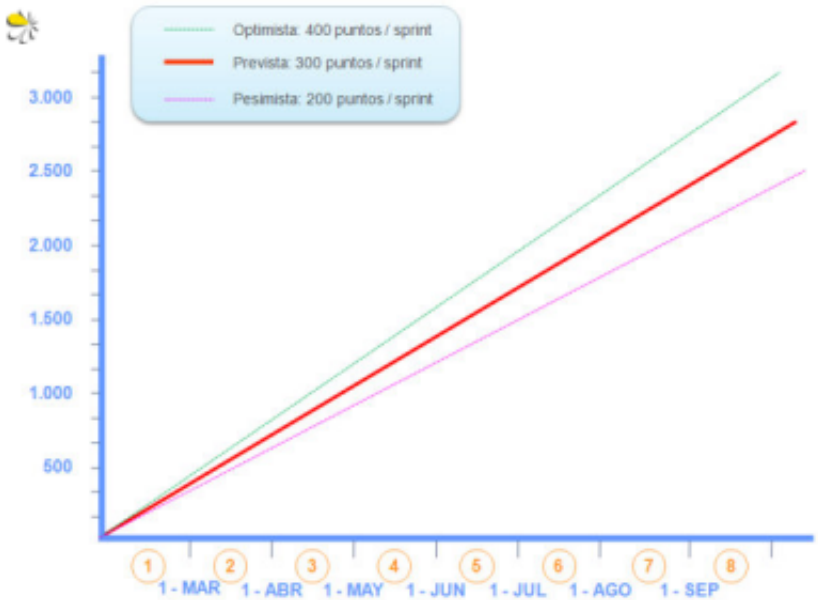
Eje X = Fechas de los sprints previstos Eje Y = Puntos de trabajo

Proyección de las velocidades previstas sobre el gráfico de producto

A continuación se traza en el gráfico la línea de velocidad prevista.

Siguiendo con el ejemplo, la línea roja de la imagen representa la velocidad de 300 puntos por sprint.

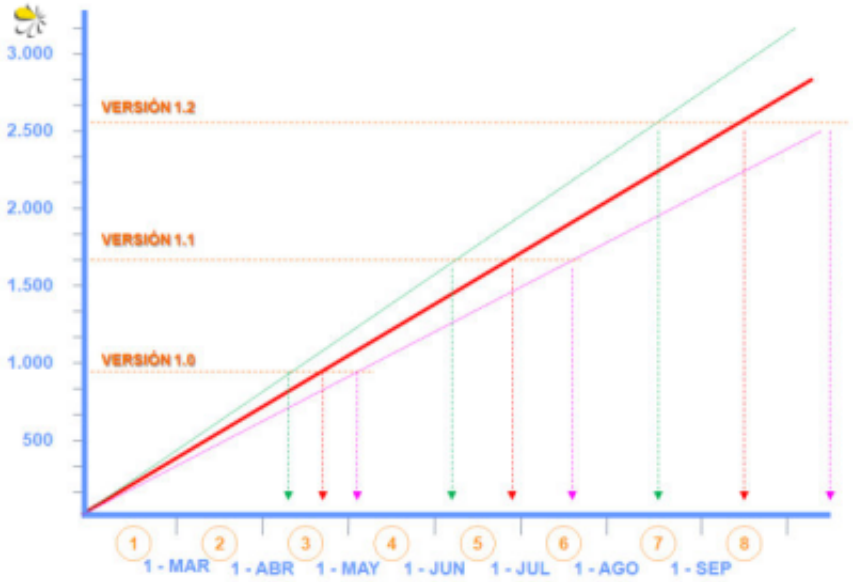
También puede resultar útil esbozar una estimación optimista y otra pesimista para tener la visión de una holgura de fechas aceptable.





Previsión de cierre de versiones sobre el gráfico de producto

La proyección sobre X del punto de la línea de velocidad correspondiente al esfuerzo previsto para una versión del proyecto, marca la fecha o sprint en el que previsiblemente estará realizada.




# H

## Historia de usuario

Es la descripción de un requisito del sistema, escrito en pocas líneas, con lenguaje habitual en el entorno del cliente, y escrito por éste, o si no, validado por él.

Las características de implementación habituales son:

- Longitud limitada: se puede escribir en una tarjeta o nota adhesiva.
- Ampliación. Para las historias que necesitan información adicional como textos literales que se deben integrar en el software, reglas de negocio detalladadas, etc.
- Criterio o pruebas de validación: Inclusión en cada historia del criterio de validación que empleará el cliente para considerarla terminada.

Principales beneficios:

- Fácil mantenimiento.
- Relación cercana con el cliente.
- Facilita la división del proyecto en entregas.
- Facilita la estimación del esfuerzo.

# I

## Incremento

Parte de producto construida en una iteración, completamente operativa, probada y documentada.

## Ingeniería concurrente

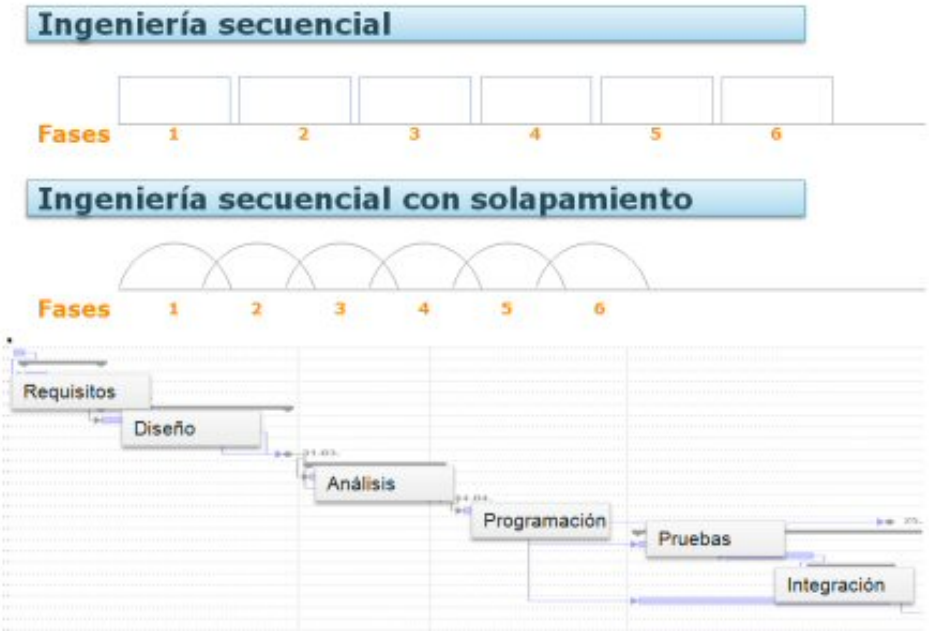
También llamada ingeniería paralela o simultánea.

Ciclo de desarrollo basado en principios de ingeniería que integra de forma solapada y concurrente los diferentes procesos de producción, realizándolos en paralelo siempre que resulta posible, en lugar de ejecutarlos de forma secuencial (ver ingeniería secuencial)


La ingeniería concurrente se diferencia de la agilidad en que basa la calidad del resultado en la calidad de los procesos empleados, más que en el conocimiento tácito de las personas.

## Ingeniería secuencial

Frecuentemente llamada "cascada" por la representación gráfica con la que se suele representar el ciclo de vida de un sistema de software desarrollado de forma secuencial:




En la ingeniería secuencial cada fase del proceso se desarrolla de forma consecutiva, de modo que cada etapa de la secuencia no se inicia hasta que no ha concluido la anterior. Si durante el proceso se detecta algún error se retrocede hasta la etapa correspondiente para subsanarlo.

Es un proceso relativamente lento, pero requiere poco esfuerzo de gestión y genera poca cooperación interdepartamental: "Tú fabricas lo que yo diseño y él vende lo que tú fabricas".

## Institucionalización

Se aplica a procedimientos de trabajo

Un procedimiento **institucionalizado** es un procedimiento documentado, conocido y aplicado de forma sistemática en la organización.

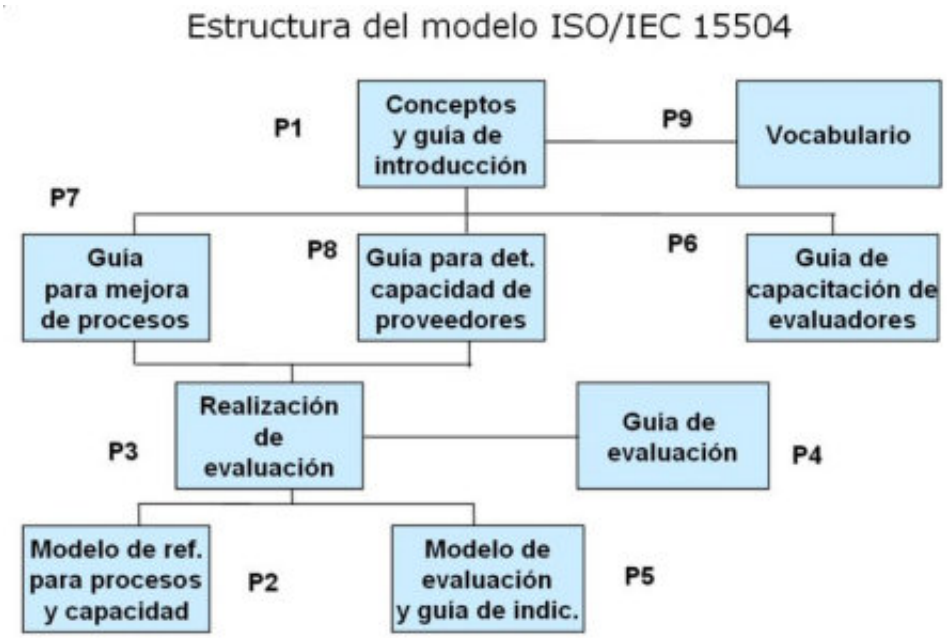
**Institucionalizar** un procedimiento es incorporarlo a la cultura de la organización.

## ISO 15504

Modelo para la mejora y evaluación de los procesos de desarrollo y mantenimiento de sistemas y productos de software.

En enero de 1993 la comisión ISO/IEC JTC1 aprobó un programa de trabajo para el desarrollo de un modelo que fuera la base de un futuro estándar internacional para la evaluación de los procesos del ciclo de vida del software. Este trabajo recibió el nombre de proyecto SPICE (Software Process Improvement and Capability dEtermination), y en junio de 1995, con la publicación de su primer borrador, desde ISO fueron invitadas diferentes organizaciones para aplicarlo y valorar sus resultados.

En 1998, pasada la fase de proyecto, y tras las primeras evaluaciones, el trabajo pasó a la fase de informe técnico con la denominación ISO/IEC TR 15504. La instrucción técnica consta de 9 apartados, recogidos en volúmenes independientes que se han ido publicando como redacción definitiva del estándar internacional ISO/IEC 15504 durante el periodo 2003 - 2005.




# Estructura de la norma

- Establece un marco para métodos de evaluación, no es un método o modelo en sí.
- Comprende: evaluación de procesos, mejora de procesos, determinación de capacidad.
- Está alineado con el estándar ISO/IEC 12207 que define los procesos del ciclo de vida del desarrollo, mantenimiento y operación de los sistemas de software.
- Equivalencia y compatibilidad con CMMI. ISO forma parte del panel elaborador del modelo CMMI y SEI mantiene la compatibilidad y equivalencia de ésta última con 15504. Sin embargo CMMi aún no es conforme a SPICE - ISO 15504.

# Dimensiones

Tiene una arquitectura basada en dos dimensiones: de proceso y de capacidad de proceso.

Desde la dimensión de proceso agrupa a los procesos en tres grupos que contienen cinco categorías de acuerdo al tipo de actividad:

## Procesos primarios

- CUS: Cliente - Proveedor
- ENG: Ingeniería

## Procesos de soporte

- SUP: Soporte

Procesos organizacionales

- MAN: Gestión

- ORG: Organización

Para todos los procesos se definen los componentes: Identificador, Nombre, Tipo, Propósito, Salidas y Notas.

**Desde la dimensión de capacidad** el modelo define una escala de 6 niveles para determinar la capacidad de cualquier proceso:

- Nivel 0: Incompleto
- Nivel 1: Realizado
- Nivel 2: Gestionado
- Nivel 3: Establecido
- Nivel 4: Predecible
- Nivel 5: En optimización

## Iteración

Cada uno de los ciclos en los que se descompone la construcción ágil de un sistema. En cada iteración produce un incremento o parte del sistema.

# L

## Ley de Parkinson

Principio que afirma que la duración de una tarea se alarga hasta completar todo el tiempo que se tiene disponible para ella.

Cuanto más tiempo se tiene, más se divaga y más alternativas, mejoras o problemas se plantean.

Fué enunciado por Cyril Northocote Parkinson en 1957 en el libro del mismo nombre, como resultado de su experiencia burócrata en el Servicio Civil Británico.

# M

## Manifiesto ágil

En marzo de 2001, 17 críticos de los modelos de mejora basados en procesos, convocados por Kent Beck, que había publicado un par de años antes el libro "Extreme Programming Explained" en el que exponía una nueva metodología denominada Extreme Programming, se reunieron en Salt Lake City para discutir sobre el desarrollo de software.

En la reunión se acuñó el término “Métodos Ágiles” para definir a los que estaban surgiendo como alternativa a las metodologías formales, (CMM-SW, PMI, SPICE) a las que consideraban excesivamente “pesadas” y rígidas por su carácter normativo y fuerte dependencia de planificaciones detalladas, previas al desarrollo.

Los integrantes de la reunión resumieron en cuatro postulados lo que ha quedado denominado como “Manifiesto Ágil”, que son los principios sobre los que se basan estos métodos.
Hasta 2005, entre los defensores de los modelos de procesos y los de modelos ágiles han sido frecuentes las posturas radicales, quizá más ocupadas en descalificar al otro que en estudiar sus métodos y conocerlos para mejorar los propios.

# Manifiesto Ágil

(http://www.agilemanifesto.org)

Estamos poniendo al descubierto mejores métodos para desarrollar software, haciéndolo y ayudando a otros a que lo hagan. Con este trabajo hemos llegado a valorar:

- A los individuos y su interacción, por encima de los procesos y las herramientas.
- El software que funciona, por encima de la documentación exhaustiva.
- La colaboración con el cliente, por encima de la negociación contractual.
- La respuesta al cambio, por encima del seguimietno de un plan.

Aunque hay valor en los elementos de la derecha, valoramos más los de la izquierda

Firmado por:

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

## Mínima Funcionalidad Facturable

Se denomina MFF o Mínima Funcionalidad Facturable al conjunto de funcionalidades de un sistema, que pueden ponerse en el mercado como una parte del producto funcional y operativa, y que puede comenzar a proporcionar un retorno de la inversión. Es un criterio clave para la descomposición del sistema final en sub-sistemas, y determinar las prioridades en las que se deben desarrollar, empleado por los modelos de planificación de producto basados en el concepto de financiación incremental.

La primera y principal referencia como modelo de gestión de producto para su financiación incremental es el libro Software by Numbers.

## Moor (ley de)

La Ley de Moore expresa que cada 18 meses se duplica el número de transistores que puede contener un circuito integrado. Es una afirmación que formuló Gordeon E. Moore, co-fundador de Intel el 19 de abril de 1965, y cuyo cumplimiento se viene constatando hasta hoy. En 1965 Gordon Moore afirmó que la tecnología tenía futuro, que el número de transistores por pulgada en circuitos integrados se duplicaba cada año y que la tendencia continuaría durante las siguientes dos décadas.[2] Más tarde, en 1975, modificó su propia ley al afirmar que el ritmo bajaría, y que la capacidad de integración se duplicaría aproximadamente cada 24 meses.[5] Esta progresión de crecimiento exponencial, duplicar la capacidad de los circuitos integrados cada dos años, es lo que se considera la Ley de Moore. Sin embargo, el propio Moore ha puesto fecha de caducidad a su ley: "Mi ley dejará de cumplirse dentro de 10 o 15 años -desde 2007-".[6] Según aseguró durante la conferencia en la que hizo su predicción afirmó, no obstante, que una nueva tecnología vendrá a suplir a la actual.[7] La consecuencia directa de la Ley de Moore es que los precios bajan al mismo tiempo que las prestaciones suben: la computadora que hoy vale 3000 dólares costará la mitad al año siguiente y estará obsoleta en dos años. En 26 años el número de transistores en un chip se ha incrementado 3200 veces.

## MoScoW

Nombre de un criterio empleado para determinar la prioridad de los requisitos (funcionalidades, epics o historias de usuario).

El enfoque MoScoW es originario de la metodología DSDM y su nombre está formado por las iniciales de los cuatro criterio de prioridad que recomienda (en inglés):

1.- Must have (es necesario)
2.- Should have (es recomendable)
3.- Could have (podría implementares)
4.- Won't have (no lo queremos... quizá en un futuro)

# P

## Personal Software Process

Personal Software Process (PSP) o proceso personal de software, es un conjunto de prácticas para la medición y gestión del tiempo de trabajo de ingenieros de software, para mejorar su productividad.

Está alineado y diseñado para emplearse en organizaciones que trabajan con modelos de procesos CMMI o ISO 15504.

## Pila de producto

En inglés product backlog
Lista ágil de los requisitos del cliente o requisitos del sistema:

- Simples: Expresados de forma breve, con una sentencia para cada uno, habitualmente con formato de historia de usuario o similar.
- Estimados: Está estimado el esfuerzo de construcción de cada requisito
- Priorizados: Ordenados según la importancia para el cliente o responsable de la lista

# Información

Es responsabilidad del cliente conocer cuál es su problema; y cómo lo desea solucionar, por eso, al igual que los requisitos del sistema de la ingeniería clásica, la pila del producto es responsabilidad del cliente (cliente, responsablie de producto, propietario del producto...) que tiene como responsabilidad decidir qué elementos forman parte de la pila del, y su orden de prioridad.

La pila del producto es el inventario de funcionalidades, mejoras, tecnología y corrección de errores, que deben incorporarse al producto a través de las sucesivas iteraciones del ciclo de desarrollo. Representa todo aquello que esperan los clientes, usuarios, y en general todos los interesados en el producto.Todo lo que suponga un trabajo que debe realizar el equipo tiene que estar reflejado en la pila de producto.

A diferencia de un documento de requisitos en gestión de proyectos predictiva, la pila es un documento vivo, y nunca se da por completa; está en continuo crecimiento y evolución.

Habitualmente se comienza a elaborar con el resultado de una reunión de "fertilización cruzada" o brainstorming en la que colabora todo el equipo partiendo de la visión del cliente.

El formato de la visión no es relevante. Según los casos, puede ser una presentación informal del responsable del producto, un informe de requisitos del departamento de marketing, etc. Sí que es importante, sin embargo, disponer de una visión real, comprendida y compartida por todo el equipo.

La pila del producto evolucionará de forma continua mientras el producto esté en el mercado, para proporcionarle constantemente el mayor valor posible, y que resulte en todo momento útil y competitivo.

Lo necesario para comenzar a desarrollar el producto es disponer de una visión comprendida y conocida por todo el equipo, y elementos suficientes en la pila del producto para llevar a cabo el primer sprint.

Formato de la pila del producto

El desarrollo ágil prefiere la comunicación directa, a la comunicación con documentos.

La pila del producto no es un documento de requisitos, sino una herramienta de referencia para el equipo. Si se emplea formato de lista, es recomendable que al menos incluya la siguiente información en cada línea:

- Identificador único de la historia o funcionalidad.
- Descripción.
- Campo o sistema de priorización.
- Esfuerzo estimado para su ejecución.

Dependiendo del tipo de proyecto, funcionamiento del equipo y la organización, pueden resultar aconsejables otros campos:

- Observaciones.
- Criterio de validación.
- Persona(s) asignada(s).
- Nº de iteración en la que se realiza.
- Módulo del sistema al que pertenece.
- Etc.

Es preferible no adoptar ningún protocolo de trabajo de forma rígida. El formato de la pila del producto no es cerrado. Los resultados de Scrum Management no dependen de la rigidez en la aplicación del protocolo, sino de la institucionalización de sus principios y la implementación en un formato adecuado a las características de la empresa y del proyecto.

## Pila de sprint

# Definición

Lista de tareas que va a realizar el equipo en una iteración, para construir un incremento. Para cada una registra la información:

- Descripción breve.
- Persona que la tiene asignada.
- Esfuerzo pendiente para terminarla.

# Información

Es útil porque descompone el proyecto en unidades de tamaño adecuado para determinar el avance a diario, e identificar riesgos y problemas sin necesidad de procesos complejos de gestión. Es también una herramienta de soporte para la comunicación directa del equipo.

Condiciones:

- Realizada de forma conjunta por todos los miembros del equipo.
- Cubre todas las tareas que el equipo ha identificado para cubrir el objetivo de la iteración.
- El equipo es su responsable, y sólo él puede modificarla durante la iteración.
- El esfuerzo máximo para cada tarea no debe sobrepasar un par de jornadas de trabajo ideal (a mayor dimensión de la tarea, mayor porcentaje de error previsible en su estimación y mayor complejidad en la gestión del riesgo)
- Es visible para todo el equipo. Idealmente en una pizarra o pared, en el espacio físico de trabajo del equipo.

# Formato y soporte de la pila de iteración

Opciones:

- Hoja de cálculo.
- Pizarra física o pared.
- Herramienta colaborativa o de gestión de proyectos.

Y sobre la que mejor se adecúa a las carac-terísticas del proyecto, oficina y equipo, lo apropiado es diseñar el formato más cómodo para todos, teniendo en cuenta los siguientes criterios:

- Incluye la información: lista de tareas, persona responsable de cada una, estado en el que se encuentra y tiempo de trabajo que queda para completarla.
- Sólo incluye la información estrictamente necesaria.
- El medio y modelo elegido es la opción posible que más facilita la consulta y comunicación diaria y directa del equipo.
- Sirve de soporte para registrar en cada revisión diaria el esfuerzo que le queda a cada tarea.

## Plaza de aparcamiento

Plaza de aparcamiento ("parking lot") es una represetación gráfica en la que aparece registrado el tamaño de un tema o epic, y el porcentaje realizado.
Su propuesta se atribuye a Jeff DeLuca en 2002, y por su formato gráfico resulta especialmente apropiado para su representación sobre tarjetas kanban.

Cada "plaza de aparcamiento" representa un tema y su uso más normal es para reflejar:

Título del tema.

Nº de historias de usuario que lo componen.

Tamaño en la unidad de medición de las historias de usuario.

Porcentaje realizado. Refleja el tamaño de las historias de usuario terminadas.


La representación de aparcamiento (parking lot) muestra las métricas de un tema o epic, y el porcentaje realizado.

## Previsibilidad

Previsibilidad, en el contexto de "objetivo de un proyecto", ser refiere al grado de certeza del coste que tendrá la ejecución, la fecha en la que estará terminado, y la descripción detallada de cómo será el producto final.

Lograr la mayor previsibilidad, es el objetivo de la gestión de proyectos tradicional.

## Procedimiento

Los procedimientos son conjuntos de acciones definidas, que en combinación con la tecnología y la acción de las personas, producen resultados esperados, y que pueden ser:

# Procesos

Procedimientos que, junto con la tecnología, aportan el conocimiento clave para lograr el resultado. En los sistemas de producción que emplean procesos, las personas "ayudan" al proceso.

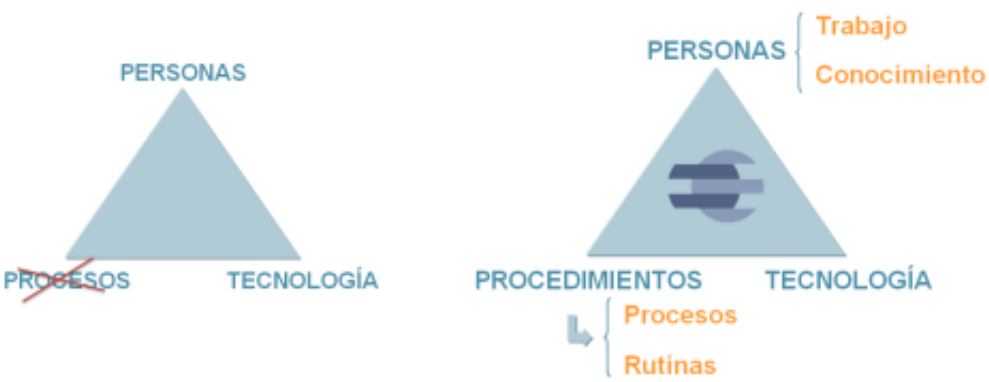
Las cadenas de producción industrial emplean procesos.

# Rutinas

Procedimientos empleados en los sistemas de producción en los que las personas son las que aportan el conocimiento clave para lograr el resultado. En los sistemas de de producción que emplean rutinas, éstas "ayudan" a las personas.

Los talleres artesanales emplean rutinas.

Scrum Manager evoluciona los tres elementos clásicos de la producción: personas, tecnología y procesos en: personas, tecnología y procedimientos




## Propietario de producto

Término habitual para designar el rol de "responsable de producto" o "product manager" en equipos ágiles.

Desempeñado por una persona (una sóla) conocedora del entorno del negocio del cliente y de la visión del producto, y que representa a todos los interesados en el producto final.

Es el responsable de la pila del producto, y su responsabilidad es determinar cuál es el resultado posible de mayor valor para los usuarios o clientes. También responde de la financiación necesaria para el proyecto, de decidir cómo debe ser el resultado final, del lanzamiento y del retorno de la inversión. En desarrollos internos puede ser el product manager o responsable de marketing quien asuma este rol. En desarrollos para clientes externos lo más aconsejable es que sea el responsable del proceso de adquisición del cliente.

## Punto de función

También llamado en gestión ágil: punto de scrum.

Unidad de medida relativa que determina la cantidad de trabajo necesaria para construir una funcionalidad o historia de usuario.

## Punto de historia

En inglés: Story Point.

Unidad de trabajo empleada habitualmente en Extreme Programming, definida por la cantidad de trabajo realizada en un día de trabajo ideal (tiempo ideal).

# R

## Refactorización

Re-estructuración del código fuente, alterando su estructura interna, sin modificar el comportamiento del programa, con la finalidad de "limpiar el código": mejorar la consistencia interna, claridad, comprensión en general su calidad.

Se recomienda realizar la refactorización, seguida de pruebas unitarias para comprobar que no ha cambiado el comportamiento del código.

Es una práctica especialmente recomendada en programación ágil, como garantía de calidad y para evitar que el cambio continuo genere "deuda técnica".

## Retrospectiva (reunión)

Al igual que los modelos de procesos incorporan prácticas de “ingeniería de procesos” para conseguir una mejora continua de su capacidad, en agilidad también van surgiendo prácticas para lo que sería el equivalente de mejora continua de la agilidad de la organización; y en esta línea, las reuniones retrospectivas son un una “meta-práctica” ágil.

El hecho de que se realicen normalmente al final de cada sprint lleva a veces a confusión y a tomarlas como reuniones de “revisión de sprint”, cuando suele ser aconsejable considerarlas como diferentes, porque sus objetivos son diferentes.

El objetivo de la revisión del sprint es analizar “QUÉ” se está construyendo, mientras que una reunión retrospectiva se centra en “CÓMO” lo estamos construyendo: “CÓMO” estamos trabajando, con el objetivo de analizar problemas y aspectos mejorables.

# S

## Scrum

Metodología ágil centrada en la gestión y seguimiento ágil de proyectos, a través de ciclos cortos de desarrollo. Es uma implementación de los campos de scrum para software realizada por Ken Schwaber, Mike Beedle y Jeff Shuterland.

Los principios de Scrum son: equipos auto-gestionados - Un avez dimensionadas las tareas no es posible agregar trabajo extra - reuniones de seguimiento diarias - ciclos de desarrollo de frecuencia inferior a un mes (propuesta original dos meses), que entregan una parte del producto completamente terminada.

Más información:

- Libro: Agile Project Management with Scrum (edición 2001)(edición 2004)
- Scrum alliance
- Flexibilidad con Scrum.

## Sprint

Cada una de las iteraciones de desarrollo que de forma sucesiva se realizan al emplear una metodología ágil para construir un sistema.

En función de las características del proyecto y de la metodología empleada, los sprints pueden tener duraciones entre una semana y dos meses.

Cada sprint produce un incremento terminado y operativo del producto.

## SWEBOK

Acrónimo de "Software Engineering Body of Knowledge", un documento promovido por la IEEE Computer Society y desarrollado por la Software Engineering Coordinating Comité para desarrollar el área de conocimiento propias de la Ingeniería del Software.

Su objetivo es alcanzar un consenso de las áreas de conocimiento que debe comprender esta ingeniería.

- Sitio oficial (swebok.org)

# T

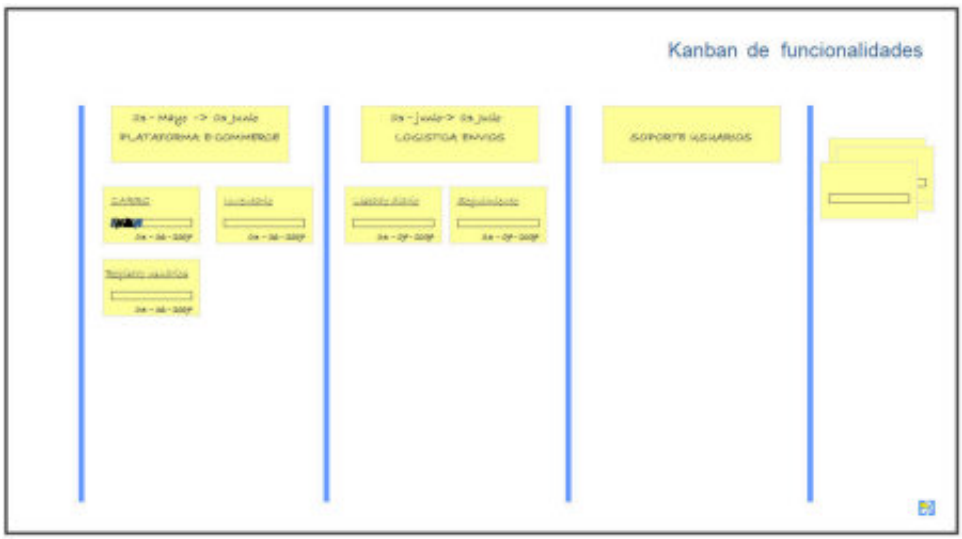
## Tablero Kanban

Tablero empleado para aplicar en el desarrollo de software el concepto original kanban, que es un medio utilizado para compartir la información en entornos de producción "JIT" (Just In Time).

Tres ejemplos de uso:

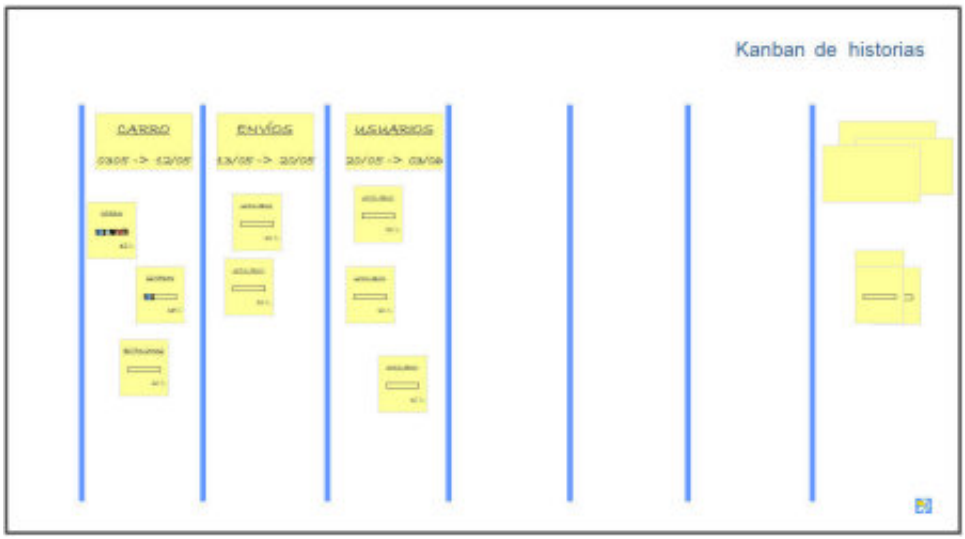
# De funcionalidades

Representan el trabajo, previsión y estado de las funcionalidades del sistema. Cada tarjeta Kanban, o elemento de información representa una funcionalidad general del sistema. La dimensión horizontal de la pizarra representa el tiempo, de forma que en las columnas temporales se van apilando las tarjetas con las funcionalidades que se irán cerrando en cada versión.




# De historias de usuario

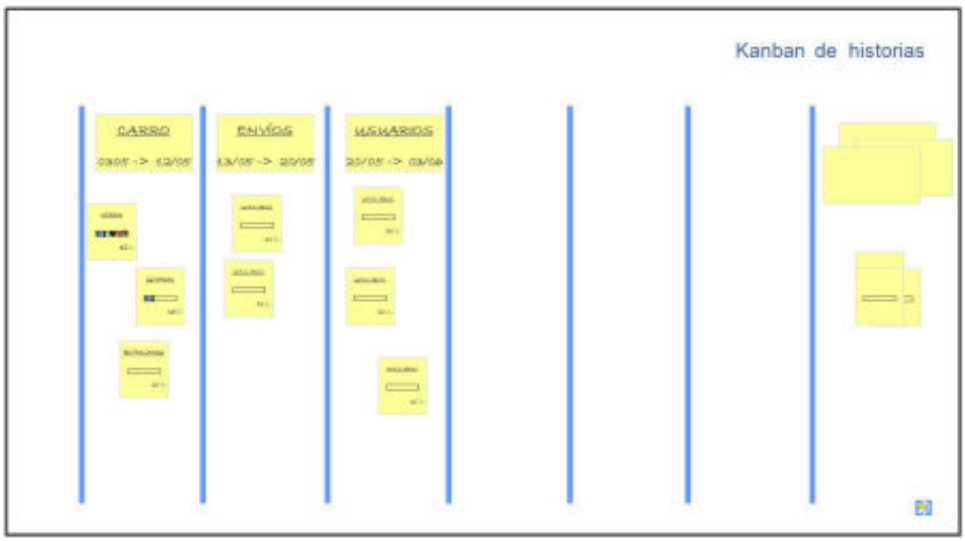
Representan el trabajo, previsión y estado de avance de la versión que se está programando. Una pizarra de historias descompone en historias de usuario las funcionalidades de una columna de la pizarra anterior. Representa por tanto un nivel de detalle mayor sobre los requisitos del sistema. La dimensión horizontal representa las diversas funcionalidades separadas por columnas y en cada una de ellas las historias de usuario en las que se descompone.




# De tareas

Representan el trabajo y el estado de avance de la iteración en desarrollo. Las tarjetas Kanban o elementos de información representan las tareas de programación en que se descomponen las diferentes historias de usuario.

Divide el espacio en tres columnas. En la primera de ellas están todas las tareas pendientes de realizar. La segunda contiene las tarjetas que actualmente se están programando, y en la tercera la de tareas completamente terminadas.




## Tiempo ideal

También llamado tiempo de tarea.

Tiempo que se estima necesario para realizar una tarea en condiciones ideales: sin ninguna interrupción, llamadas telefónicas, descansos, reuniones, etc.

Es el concepto similar al que PSP (Personal Software Process) denoina "Delta Time".

## Tiempo real

Tiempo real, o tiempo de trabajo.

Tiempo efectivo empleado en la ejecución de una tarea. Se suele medir en horas o días.

## Timeboxing

Técnica de administración de tiempo, comúnmente empleada en los modelos de gestión ágil, consistente en dividir el trabajo en tareas con una asignación de tiempo limitado y corto para su ejecución.

- Facilita la priorización de los objetivos.
- Combate la ley de Parkinson (tendencia a dilatar el tiempo de cada tarea) y la procrastinación.
- Incrementa los tiempos de transición entre tareas, útiles para recuperar el ritmo y tomar retro-información de la evolución del trabajo

Trabajar con una técnica de timeboxing requiere:

- Definición clara de los objetivos.
- Reducción de las interrupciones.

# U

## Unidad de trabajo

Las unidades para medir el trabajo pueden estar directamente relacionadas con el producto, como los tradicionales puntos de función de COCOMO, o a través del tiempo necesario para realizarlo.

La gestión ágil suele llamar a las unidades que emplea para medir el trabajo “puntos”, “puntos de funcionalidad” “puntos de historia”… pero se trata siempre de medición a través del tiempo, no del producto.

Así por ejemplo la unidad de medida “Story Point” de Extreme Programming define: la cantidad de trabajo que se realiza en un día de trabajo ideal.

Cada organización, según sus circunstancias y su criterio institucionaliza su métrica de trabajo definiendo el nombre y la definición de las unidades teniendo en cuenta que se basan en el tiempo necesario para ejecutarlo.

Pueden ser: puntos, puntos de función, puntos de historia, días, horas… y referirse a tiempo real o tiempo ideal.

Lo importante no es si emplea uno u otro nombre, si se refiere al trabajo realizado en cuatro o en ocho horas, o si éstas son reales o ideales. Lo importante es que la métrica empleada, su significado y la forma de aplicación sea consistente en todas las mediciones, en todos los proyectos de la organización y conocida por todas las personas:

Que se trate de un procedimiento de trabajo institucionalizado.

## Unified Process

Marco de desarrollo ágil, iterativo e incremental, que solapa las tres fases de desarrollo (elaboración, construcción y transición) en iteraciones breves de tiempo cerrado. En determinados proyectos también puede solaparse la fase previa de inicio.

Unified Software Development Process, comprende la definición general de este tipo de ciclo ágil: iterativo e incremental, pero no la concreción en procedimientos o prácticas determinadas, y son varios los autores y organizaciones que han desarrollado variaciones y concreciones sobre este patrón UP genérico:

- RUP - Rational Unified Process, por IBM / Rational.
- OpenUP Open Unified Process, por Eclipse como evolución del previo Basic.
- Unified Process (BUP).
- OUM Oracle Unified Method.
- AUP Agile Unified Process, por Scott W. Ambler.
- EssUP Essential Unified Process, por Ivar Jacobson.

# V

## Velocidad

Con carácter general, en el contexto de gestión ágil de proyectos, el término velocidad se define como la cantidad de trabajo realizada en una unidad de tiempo.

Cuando en Scrum no se indica más que velocidad, se suele referir a la cantidad de trabajo (puntos de función, puntos de historia...) realizada en un sprint.

## Velocidad absoluta

Cantidad de producto construido en un sprint. Se expresa en la misma unidad que la empleada para las estimaciones (puntos de función, horas o días, reales o teóricos).

## Velocidad relativa

Cantidad de producto construido en una unidad de tiempo de trabajo.
Ejemplos: puntos de función / semana de trabajo real; u horas teóricas / día de trabajo real