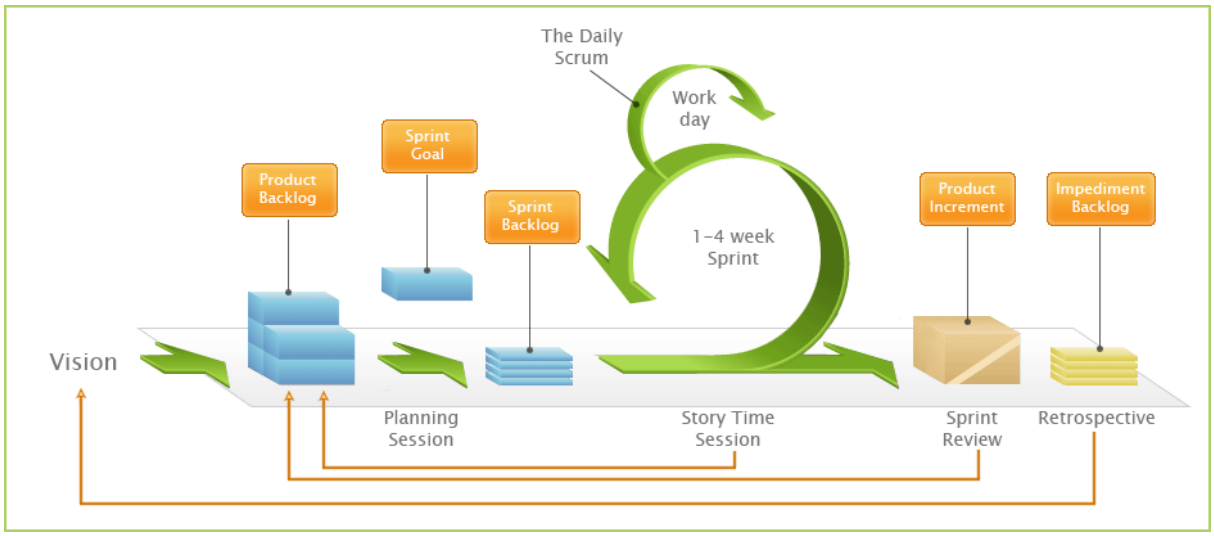
TFC

GESTION DE PROYECTOS INFORMÁTICOS

Metodología Scrum.

- ***Desarrollo detallado de la fase de aprobación de un proyecto informático mediante el uso de metodologías ágiles.***



Autor: Manuel Trigas Gallego.

Consultora: Ana Cristina Domingo Troncho.

Tabla de contenido

1.- ¿QUÉ ES UN PROYECTO?

Según la definición que nos proporciona PMI en su guía **PMBOOK**, un proyecto se podría definir como *"un servicio temporal que se lleva a cabo para crear un producto, servicio o resultado único"*.

Podemos decir entonces que un proyecto tiene un inicio y un fin, este fin se tiene que alcanzar dentro de un tiempo fijado.

En un proyecto la consecución de los objetivos al final del mismo es la máxima deseada, pero la mayor parte de las veces, bien por una mala planificación, o bien por una mala gestión de los recursos, es imposible finalizar el proyecto con éxito. Aun así se da por finalizado (Figura.- 1).

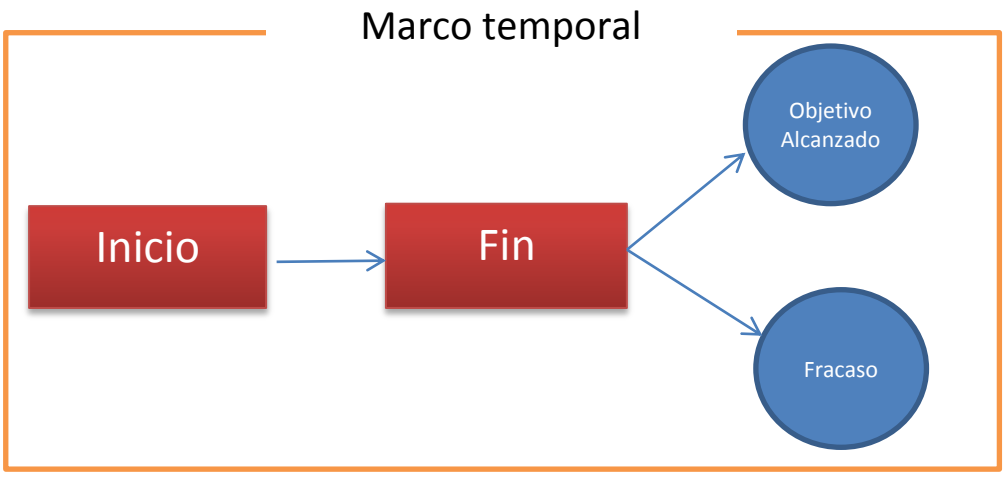


Figura.- 1: Gráfica definición de proyecto.

Que el proyecto llegue o no a alcanzar los objetivos depende en gran medida de varios factores. El proyecto ideal sería aquel que cumple las siguientes condiciones:

- **Entorno**: No sufre modificaciones de forma rápida, sino que se alargan en el tiempo.
- **Cliente**: Tiene muy claro que es lo que se necesita, sabe transmitirlo y nosotros entendemos perfectamente sus necesidades.
- **Equipo**: Disponemos del equipo de profesionales necesario para poder atender a esa necesidad y además sabemos cómo resolverla.
- **Fases**: Las fases se harían de una forma lineal, organizada y no surgiría ningún problema durante su realización.

Durante estas fases se realizan las siguientes actividades:

1. Se hace una toma de requerimientos al principio de nuestro proyecto y no sería necesaria ninguna reunión más hasta la entrega final.
2. Se crea la documentación necesaria en cada fase, y se hace entrega de ella a las fases siguientes.

3. El cliente ya tiene su producto y este se ajusta a lo indicado, por lo tanto no hay modificaciones que realizar sobre él.

Figura.- 2: Esquema proyecto ideal.

Pero desgraciadamente el desarrollo de un servicio o producto software no es tan fácil. Esto es debido a que el entorno en el que se desarrolla (tecnología, nuevas funcionalidades, la competencia) sufre modificaciones de forma constante, de tal manera que la idea de proyecto inicial difiere mucho de lo que realmente se desea finalmente.

En el planteamiento para crear cualquier producto sería necesario definir los elementos que van a participar:

Figura.- 3: Elementos principales a tener en cuenta en la creación de un producto.

1. **El cliente**: Será la persona que nos está pidiendo una solución para un problema específico.
2. **El usuario**: La persona que utilizará esta nueva solución.
3. **El tiempo**: El proyecto se atendrá a unas fechas de comienzo y unas fechas de fin.
4. **Jefe de Proyecto**: Figura necesaria para la gestión de los recursos, así como la planificación del proyecto.

Para poder realizar una buena interrelación de todos estos elementos se debería tener en cuenta una serie de puntos:

- Deberíamos definir qué estructura de organización se va a usar. Es importante en este punto, la aplicación de la experiencia en proyectos anteriores, puesto que, una buena comunicación entre los diferentes departamentos que la definen es vital en la gestión del proyecto.
- Cuando se defina la organización a usar, se tendrá especial cuidado en que pueda cambiar a lo largo del tiempo, es necesario realizar revisiones periódicas.
- Se deberían de definir qué fases van a componer el proyecto.
- Creación de roles y las responsabilidades que implica cada uno de ellos.
- Las tareas que definen el proyecto deberían de estar definidas de forma correcta. Al mismo tiempo, las tareas tienen que ser validadas por el personal responsable.
- Gestionar todos estos elementos necesitan de una figura - sería aquí donde aparece el **Jefe de Proyectos** - y también una entidad dentro de la organización que sería la **Oficina de Dirección de Proyectos.**

Todos estos puntos llevan a la aparición de una nueva disciplina, la ***Gestión de Proyectos***.

Realmente la **Gestión de Proyectos** surgió ya como necesidad en los años 50 en el ámbito militar. Los proyectos que se desarrollaban eran de gran magnitud y necesitaban de personal cualificado en diferentes disciplinas, por lo tanto, la coordinación de estos grupos era un paso natural.

A partir de este momento surgen nuevos conceptos a la par que herramientas que van a facilitar la gestión de un proyecto, como son el ciclo de vida, descomposición en tareas, realización de gráficos, etc…

La ***Gestión de Proyectos*** se podría definir como "*la disciplina de planear, organizar, asegurar y coordinar recursos y personas para cumplir con los Objetivos, Entregables y Criterios de Éxito de los proyectos" (fuente Wikipedia).*

Una breve descripción de la evolución histórica de la gestión de proyectos podríamos mostrarla en la siguiente gráfica.

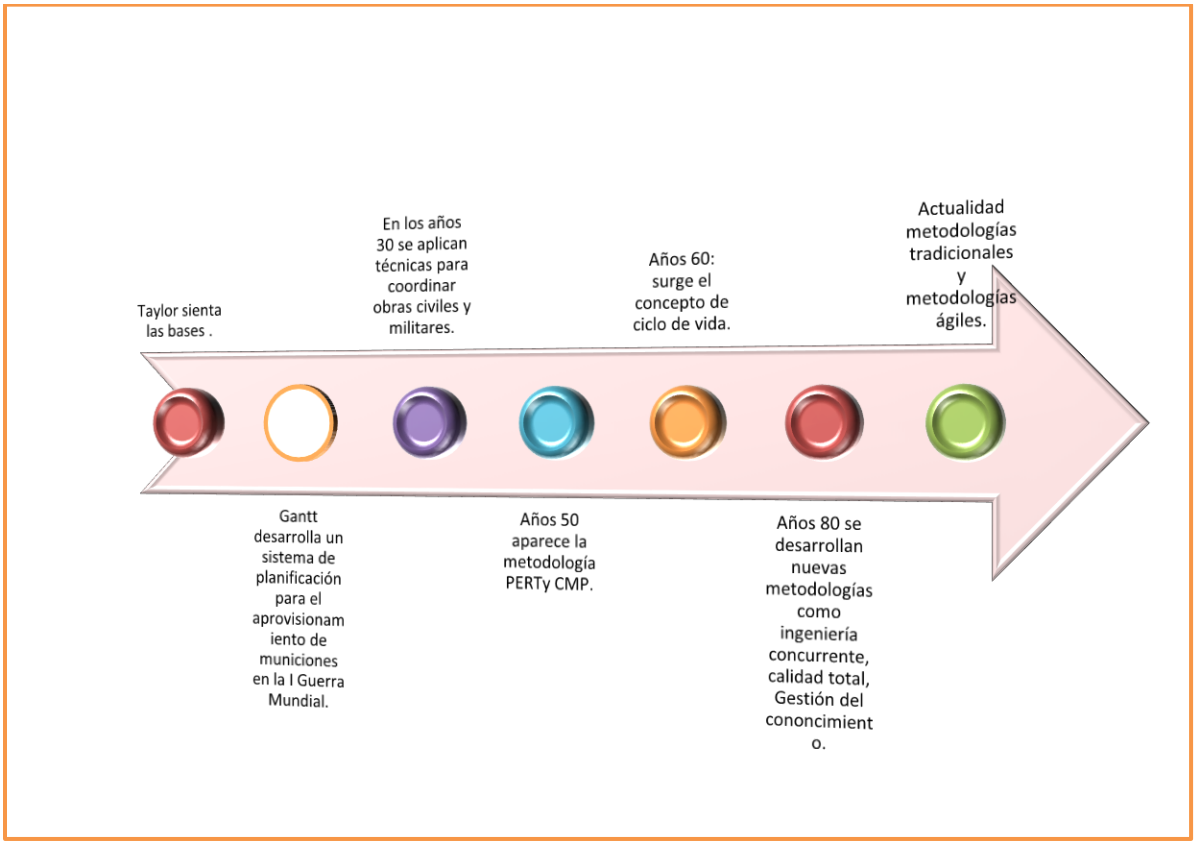


Figura.- 4: Evolución histórica de la gestión de proyectos.

1.1.- Fases y ciclos de vida de un proyecto

El comienzo de un proyecto siempre viene acompañado de un grado de incertidumbre. Como opción para paliarlo se optará por anidar las tareas a realizar en diferentes **fases**.

Estas fases servirán para tener un control más específico de cada una de las tareas, y formarán el denominado ***Ciclo de Vida de un Proyecto.***

Los ciclos de vida de un proyecto servirán para definir el comienzo y el final del mismo. Generalmente cuando una empresa emprende un proyecto lo primero que solicitará será un estudio de viabilidad o factibilidad. Este procedimiento puede ser la base para que se marque un inicio del proyecto.

De la misma manera, el ciclo de vida se usa como vínculo con las distintas tareas que se realizan dentro de la empresa u organización.

Las fases definidas por la mayoría de los ciclos de vida de proyectos en general incluyen alguna forma de transferencia de tecnología, tales como requisitos a diseño, construcción a operaciones o de diseño a implementación. Al finalizar una fase se entregará una documentación, proceso, etc…que comúnmente se denomina **entregable,** que será revisada y aceptada antes de continuar con la siguiente fase; además servirá para corregir y detectar los posibles errores que se vayan

produciendo, así como las desviaciones en los costes. Las evaluaciones de cada final de fase suelen tener diferentes nombres como *salidas de fase, puntos de impacto o puertas de etapas.*

Sin embargo, en algunas ocasiones se comienza la fase siguiente antes de la aprobación de los productos a entregar de la fase anterior, cuando los riesgos percibidos se pueden asumir. A la práctica de superponer fases suele denominarse *vía rápida*.

La mayor parte de los ciclos de vida comparten las siguientes características:

- El coste del personal es bajo al comienzo, más alto hacia la mitad, y menor al llegar al final del proyecto.
- El riesgo y la incertidumbre de finalizar el proyecto con éxito es más alta al principio del proyecto y la probabilidad de fracaso disminuye a medida que se va avanzando en el proyecto.
- La influencia por parte de los participantes en el producto es más alta al comienzo del mismo y decae progresivamente con la continuación del proyecto. Hay que tener en cuenta que el coste de los cambios y los errores tiende a aumentar a medida que avanza el proyecto.

Los ciclos de vida del proyecto responden a:

- ¿Qué trabajo se va a realizar en cada fase?
- ¿Quién participará en cada fase?
- Las fases se definirán de forma secuencial, es decir, que una fase no comienza hasta que termina la otra. Suelen contener una serie hitos o tareas que marcan los momentos más importantes en el desarrollo del proyecto.

Así pues, la base de cada fase se puede resumir en:

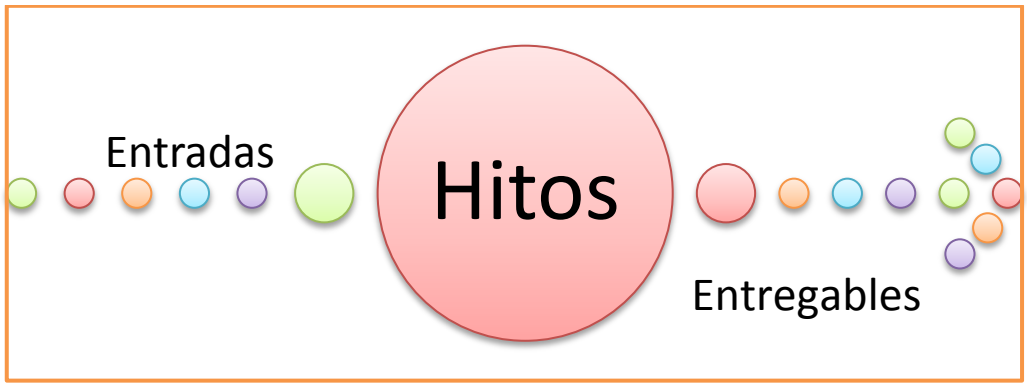


Figura.- 5: Elementos básicos que formarán una *fase.*

El enfoque de ciclo de vida más usado en las ciencias de la información es el que usa las siguientes seis fases definidas en función de los recursos necesarios:

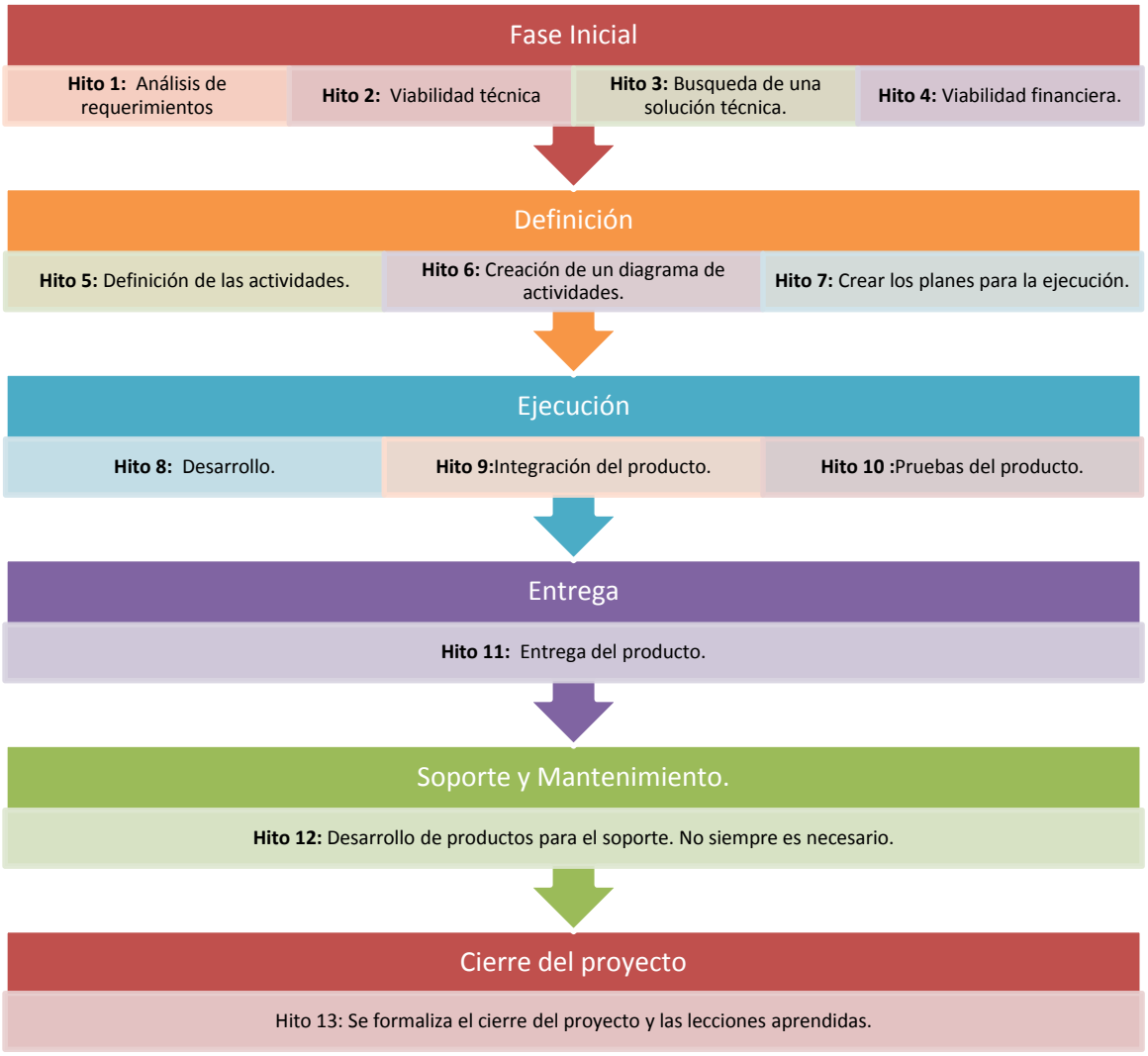


Figura.- 6: Ciclo de vida básico de un proyecto.

Todas estas fases, sobre todo en la fase de ejecución del proyecto suelen ir acompañadas de unas etapas complementarias como son, el control, la verificación y el cumplimiento de las normas. En España existe la norma **UNE157001** que está ligada a las TI.

Es importante diferenciar entre ***ciclo de vida de un proyecto,*** el cual puede comenzar o terminar de forma independiente al inicio de la gestión de proyecto, y el ***ciclo de vida de la gestión de proyectos*** el cual puede finalizar antes de que el producto esté finalizado. A pesar de que son ciclos diferentes, no son independientes.

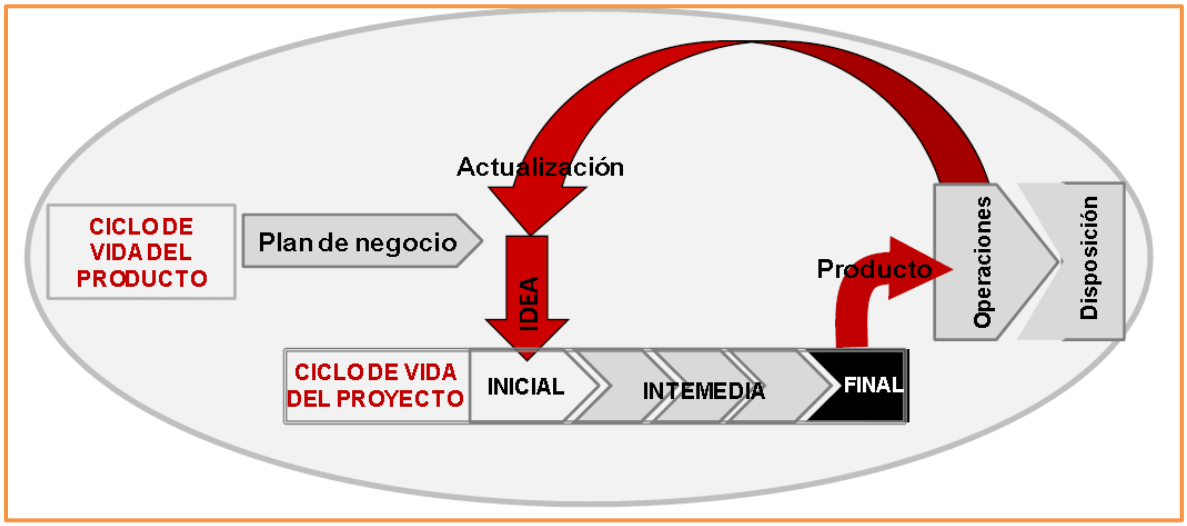


Figura.- 7: Relación entre el ciclo de vida del producto y el ciclo de vida de un proyecto (Fuente INTECO*: "Guía práctica de gestión de Proyectos"*).

Tanto para el *ciclo de vida de un proyecto* como para el *ciclo de vida de gestión de proyectos* se puede seguir aplicando el modelo tradicional de planificación **PDCA (P**lan, **D**o, **C**heck, **A**ct), es decir, Planificar, Hacer, Verificar, Actuar.

1.2.- El modelo tradicional de planificación PDCA

El modelo PDCA fue desarrollado por **Walter Shewhart** en los laboratorios Bell en 1930, pero fue popularizado por **W. Edwards Deming** y se suele uno referir a este modelo como *"circulo de Deming"* o también "*espiral de mejora continua"*.

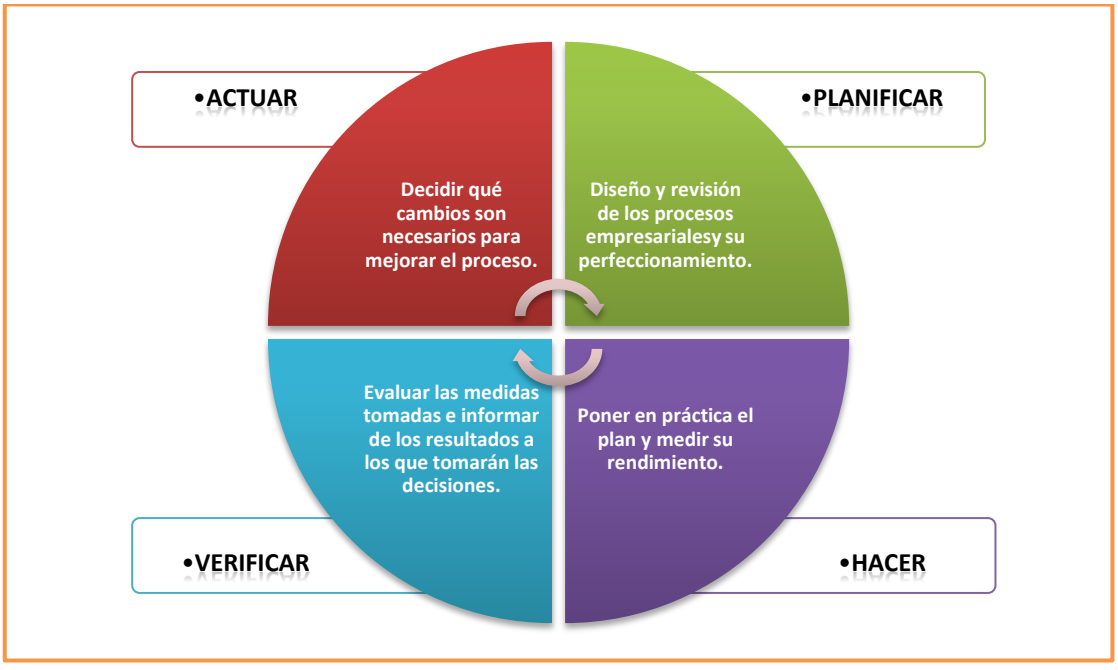


Figura.- 8: Gráfico PDCA.

1.2.1.- Primer paso: PLANIFICAR.

- **Reconocer el problema y establecer prioridades**. El problema puede ser descrito en términos muy generales basados en la información de varias fuentes.
- **Formar el equipo de la resolución de problemas**. Equipos interdisciplinarios de las personas cercanas al problema son los mejores.
- **Definir el problema y su ámbito de aplicación claramente**. Quién, Qué, Cuándo y Dónde. Análisis de Pareto puede ser útil en la definición del problema.
- **Analizar el problema / proceso**. Diagramas de flujo del proceso pueden ser herramientas útiles.
- **Determinar las posibles causas**. Los diagramas de Causa y efecto servirán para identificar las causas de la raíz de un problema. Los datos de los diagramas se pueden organizar mediante el uso de hojas de verificación, diagramas de dispersión, histogramas, y gráficas de ejecución.
- **Identificar las posibles soluciones**. Lluvia de ideas para encontrar soluciones.
- **Evitar la tentación de proponer soluciones rápidas e inmediatas**.
- **Los objetivos deben ser específicos, medibles, alcanzables y realistas.**
- **Evaluar las posibles soluciones.** Centrarse en las soluciones que aborden las causas fundamentales y la prevención de la ocurrencia problema.
- **Las soluciones deben ser rentables**, lograr el consenso del grupo es importante.

1.2.2.- Segundo paso: HACER.

- **Implementar la solución de cambio o proceso.**
- **Seguimiento de los resultados y recopilar datos.**

1.2.3.- Tercer paso: VERIFICAR.

- **Revisar y evaluar el resultado del cambio.**
- **Medir el progreso en contra de los hitos.**
- **Verificar que no haya consecuencias imprevistas.**

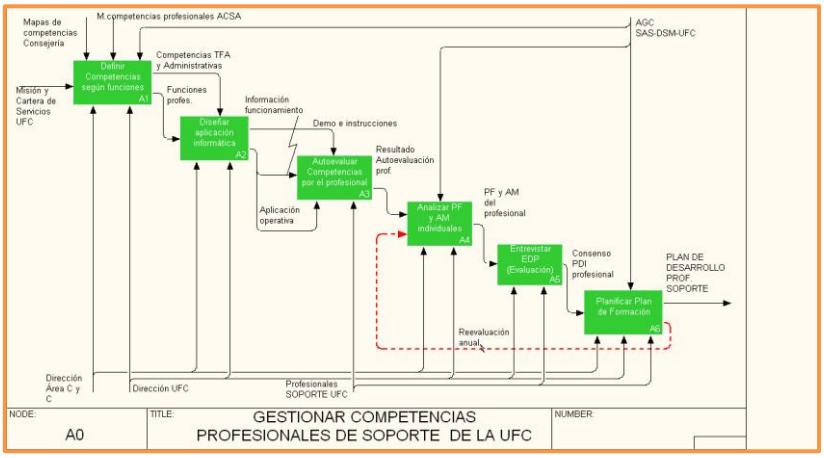


Figura.- 9: Ejemplo desarrollo básico de un ciclo de mejora con PDCA para la gestión de competencias (fuente: www.pdca.es).

A la hora de describir el ciclo de vida de un proyecto se podría hacer de forma general o más detalladamente.

Estos enfoques detallados en los que se incluyen, formularios, gráficos, etc… dan lugar al siguiente punto: ***metodologías para la gestión de proyectos***.

2.- EL CONCEPTO DE METODOLOGÍA.

Definiremos la metodología como aquella disciplina que indicará que métodos y técnicas hay que usar en cada fase del ciclo de vida de desarrollo del proyecto.

Los elementos que componen a una metodología son:

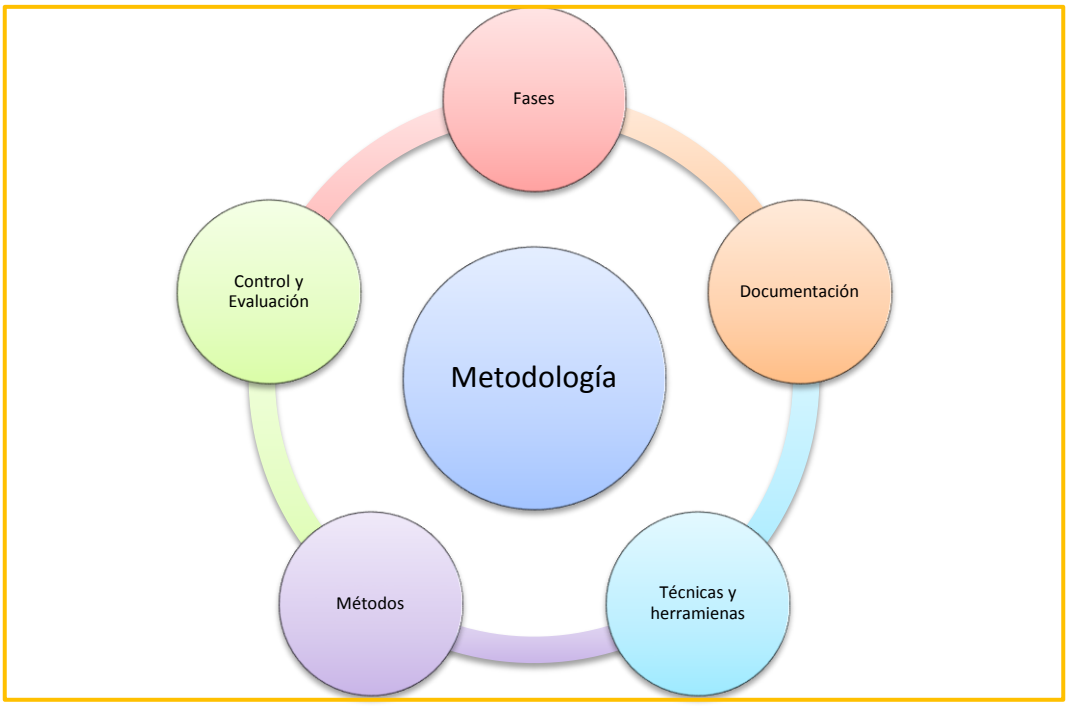


Figura.- 10: Elementos básicos de una metodología.

1. **LAS FASES:** En este punto se marcarán las diferentes actividades que hay que realizar por cada fase.

2. **LOS METODOS:** Se tendrá que identificar el modo en el que se realizará el proceso de desarrollo del producto software. Generalmente se suele descomponer los procesos en tareas más pequeñas, en estas tareas se definen los valores que recibirá cada fase así como los que generará y la técnica que se tendrá que usar.

3. **TECNICAS Y HERRAMIENTAS:** Indicarán cómo se debería de resolver cada tarea y qué herramientas podríamos usar. Existe diferentes tipos de técnicas, algunas de ellas son:

    a. ***De recopilación de datos***: Uso de entrevistas, formularios, etc…
    b. ***Técnicas gráficas***: Diagramas, organigrama, diagramas de matrices, etc…
    c. ***Técnicas de modelado***: Desarrollos estructurados y orientados a objetos:

4. **DOCUMENTACIÓN:** Es necesario indicar qué documentación se va a entregar durante todas las fases, esa documentación se debería de realizar de una manera exhaustiva y completa usando todos los valores de entrada y salida que se van generando, esto servirá para recoger los resultados y tomar decisiones de las diferentes situaciones planteadas.

5. **CONTROL Y EVALUACIÓN:** El control y la evaluación también se debe de realizar a lo largo de todo el ciclo de vida. Consistirá en comprobar y aceptar/denegar todos los resultados que se vayan obteniendo y poder replantear, si es necesario, una nueva planificación de las tareas asignadas, la meta será lograr el objetivo.
   Suelen usarse técnicas, como PERT o los diagramas de Gannt.

Aunque no todas las metodologías tienen las mismas características, sí deberían de compartir los siguientes puntos para poderse catalogar como una buena metodología:

1º.	**Ha de ser una metodología impersonal.**
2º.	**El tiempo de aprendizaje tendría que ser reducido.**
3º.	**Apoyar de forma gráfica.**
4º.	**Uso de un estándar para la documentación.**
5º.	**Que cubra todo o la mayor parte del ciclo de vida del proyecto.**
6º.	**Que simplifique el trabajo y consecuentemente que aumente la productividad.**
7º.	**Herramientas que ayuden a aumentar también la productividad.**
8º.	**Que sea de fácil mantenimiento.**
9º.	**Que la metodología haya evolucionado y madurado.**
10º.	**La formación.**
11º.	**El coste.**

Las ventajas que aporta el uso de una metodología para crear un producto se podría resumir en los siguientes puntos:

- **Facilitan la planificación.**
- **Facilitan el control y el seguimiento adecuado del proyecto.**
- **Mejoran el uso de los recursos.**
- **Permiten evaluar de forma más fácil los resultados obtenidos y valorar los objetivos conseguidos.**
- **Mejoran la comunicación entre el cliente y las personas que van a llevar a cabo el proyecto.**
- **Garantizan que el producto final tendrá la calidad esperada.**
- **Se tendrán presentes unos plazos para el desarrollo del producto.**
- **Permitirá definir el ciclo de vida adecuado al proyecto.**

Una vez conocidas las ventajas llegará el momento de escoger la metodología. Esta selección dependerá de factores como *el tamaño de la organización, experiencia profesional y la capacidad de innovar, las herramientas técnicas de las que se dispone y el tipo de proyecto a desarrollar*.

Estos factores se tienen que tener en cuenta, puesto que una metodología adecuada para el desarrollo de un servicio o de un producto, no tiene por qué serlo para otro, por muy similar que sea.

Podríamos realizar una clasificación de las metodologías existentes en base a los siguientes enfoques:

SEGÚN LA NATURALEZA	**Metodologías orientadas al flujo de información:**	El sistema se concibe como un conjunto de unidades que entran-se procesan-salen. Aplican los conceptos de la programación estructurada y fueron las primeras en aparecer. ▪ **Diseño estructurado de "Yourdon**
	Metodologías orientadas a objetos:	Basado en la orientación de objetos. Se desarrollan alrededor del concepto de clase. ▪ **Rational Unified Process.**
	Metodologías híbridas:	Son metodologías que abarcan más de una de las familias anteriores. No se centran en la naturaleza tecnológica del proyecto, sino en normalizar todos los desarrollos de software de una organización. ▪ **Métrica versión 3** ▪ **SSADM.**

SEGÚN EL GRADO DE FORMALISMO	**Metodologías Pesadas:**	Son las metodologías clásicas, los métodos de trabajo son muy formales. Conlleva realizar una gran carga de trabajo de gestión y generar una gran cantidad de documentación. ▪ **Cascada, RUP.**
	Metodologías ágiles:	Son las últimas en aparecer y se basan en dar respuestas a los problemas con los que se encuentran las metodologías tradicionales. Usan el concepto de adaptación a los requisitos que no se conocen en lugar de la predicción. ▪ **Extrem Programming.** ▪ **Scrum.**

Este último enfoque da lugar a que aparezcan detractores de las metodologías ágiles. El motivo es que estas metodologías al simplificar el proceso ya no son tan rigurosas y afectarán a la calidad del software, sin embargo será la nueva tendencia a seguir.

2.1.- La metodología tradicional.

Las metodologías tradicionales son el primer tipo de metodología que surge como guía para garantizar la creación de un producto con un nivel de calidad.

Esta metodología es una disciplina que tiene como base una gestión predictiva, es decir, que parte de unos requisitos iniciales. Con estos requisitos se configurará un plan adecuado usando los recursos y el tiempo necesarios, durante la fase de creación se comprueba si hay desviaciones, si las hay se definen las medidas a tomar y valorar cuáles son las modificaciones que puede experimentar la planificación original.

Por lo tanto, esta metodología define un conjunto de fases secuenciales en las que se indican las operaciones que se van a realizar, el tiempo que van a llevar y cual va a ser su coste.

Las características más relevantes de esta metodología son:

- Los requisitos son definidos durante todo el proyecto.
- Se basa en los procesos.
- Se supone que el proyecto no va a surgir ningún tipo de cambio por lo tanto no está sujeto a variables.
- Los proyectos suelen estar bien documentados.
- Gestión predictiva.
- El desarrollo se define en fases cuyo conjunto se denomina “ciclo de vida”.
- Documentación exhaustiva de todo el proyecto.
- Se enfocan en obtener el producto en tiempo estimado y con el coste establecido.

PMBOOK define los siguientes procesos para una gestión de proyectos con base de una metodología tradicional.

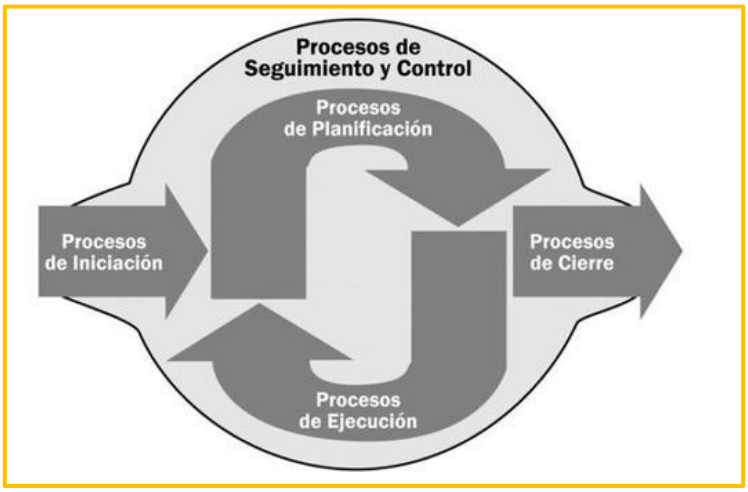


Figura.- 11: Imagen extraída del PMBOOK 2004 que representa los procesos de una gestión de proyectos clásica o tradicional.

2.1.1.- La metodología en cascada.

La primera metodología que se empieza a usar es la que denominamos *metodología en cascada.* Dará lugar al ciclo de vida usado en su implantación denominado ***ciclo de vida en cascada.***

El ciclo de vida en cascada o waterfall se caracteriza porque todas las fases se realizan de forma secuencial, es decir, que las etapas se llevan a cabo una después de la otra, pero eso sí, cada etapa tiene que estar finalizada antes de comenzar la siguiente. Cada una de estas etapas o fases son realizadas por personas o equipos de trabajo especializados.

Las fases que se corresponden con:

1. **Requerimientos del sistema.**
2. **Requerimientos del software.**
3. **Análisis.**
4. **Diseño del programa.**
5. **Codificación.**
6. **Pruebas.**
7. **Implantación.**

Las tareas que se realizan durante las etapas son:

1. **Análisis de requerimientos:** Se analizará el problema, se definirán los requisitos y qué objetivos se tienen que conseguir.
2. **Diseño:** Toda la información recogida se intenta plasmar mediante una estructura de datos adecuada, una arquitectura algoritmos a usar, integrales.
3. **Codificación:** El diseño se implementará usando la tecnología escogida como solución.
4. **Prueba:** En esta fase se intenta encontrar los errores para corregirlos además de comprobar si el software cumple con el objetivo inicial.

5. **Implantación y mantenimiento**: Esta fase servirá para corregir errores que no se detectaron antes, adaptarse al entorno de trabajo y mejorar la aplicación.

Los roles que se puedan identificar en el modelo de cascada son:

- **Gestor de proyectos**: Gestionará y controlará el proyecto en todas las etapas del mismo. Deberá de controlar si el proyecto sigue la planificación y si se están cumpliendo los plazos temporales.
- **Arquitecto del software**: Son los encargados de buscar una solución y están en las fases de especificación y obtención de los requisitos además de en la fase de diseño.
- **Desarrolladores**: Implementarán las instrucciones indicadas por el arquitecto software. Participa en las fases de desarrollo, pruebas y mantenimiento.
- **Probadores**: Realizaran las pruebas necesarias para comprobar el correcto funcionamiento del software creado. Se tendrá que comprobar si realmente la aplicación satisface las necesidades marcadas por los clientes.
- **Consejeros**: Es una figura necesaria porque a pesar de que no se trata de una persona técnica, es experta en ayudar a entender el problema y qué solución se podrá tomar.

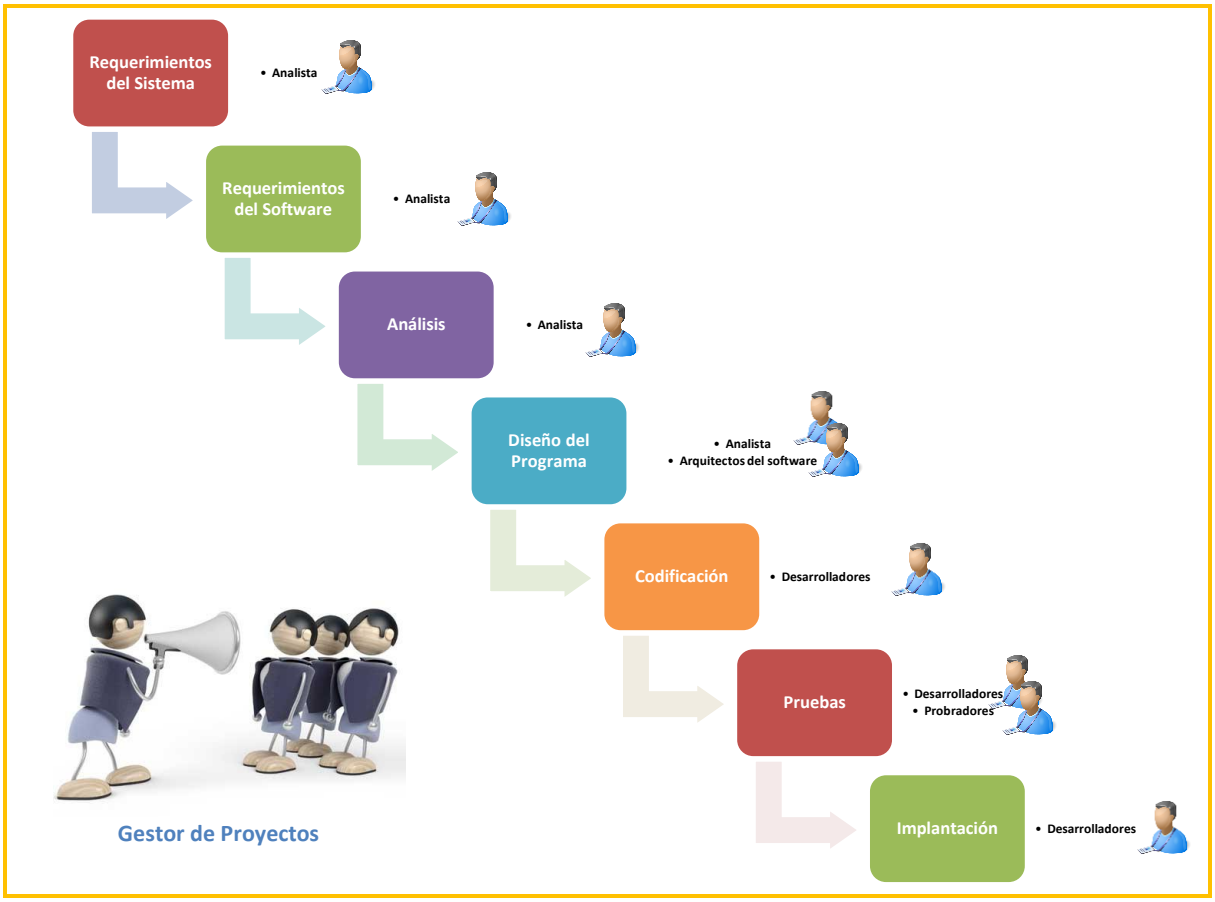


Figura.- 12: Relación de Roles con cada una de las fases.

La metodología en cascada presenta la ventaja de ser una planificación sencilla, la calidad obtenida del producto es alta y no se necesita de un personal altamente cualificado.

Sin embargo tiene el inconveniente de que para el desarrollo es necesario tener todos los requisitos al principio, el problema es que la mayor parte de las veces el cliente al comienzo del proyecto no tiene definidas todas las especificaciones y puede ser que surjan situaciones o casos imprevistos. Además añade el inconveniente de que al ser secuencial la corrección de errores se vuelve más difícil, estos errores no se descubren hasta el final que es cuando el cliente va a ver el resultado, por lo tanto habrá que gastar recursos otra vez.

Todos estos inconvenientes hacen que la metodología finalmente sea más costosa y lenta.

Quizás sí es una metodología adecuada cuando se tienen todas las especificaciones de forma inicial, el tipo de producto ya es conocido o es un proyecto que se entiende su naturaleza perfectamente.

2.1.2.- Metodología RUP.

Otra metodología característica de la metodología tradicional es **RUP (Rational Unified Proces).**

RUP fue desarrollado por *Rational Software* y ahora perteneciente a **IBM**. Se basa en un marco de procesos de trabajo que pueden ser adaptados por las organizaciones que hagan el desarrollo y por los desarrolladores, seleccionando los elementos más apropiados del proceso.

El proceso **Unificado Rational** resulta de una combinación de varias metodologías y se vio influenciado por otros métodos como el espiral. Es una metodología que está basada en Objectory, metodología que fue creada por *Ivan Jacobson,* y el proceso fue desarrollado con las mismas técnicas que el equipo de creadores y desarrollo usaba para el diseño del software. Se usaría **UML** (Unified Modeling Language).

RUP se basa en tres módulos principales que contestan a las preguntas de: quién hace el proceso, qué productos de trabajo se van a realizar, qué documentos y modelos se van a producir y cómo se van a realizar las tareas.

Las fases que forman el ciclo de vida de RUP se dividen en cuatro:

1. **Inicio**: Se establece el objetivo del sistema y se recogen los requisitos del usuario.
2. **Elaboración**: Se busca reducir riesgos y cumplir con la planificación y coste indicado. Se genera una estructura arquitectónica que se puede ejecutar y que servirá de punto de partida para después permitir desarrollar la disciplina de diseñar, implementar y probar.
3. **Fase de construcción**: Partiendo de la arquitectura elaborada en la fase anterior se realizará casi toda la implementación, creando versiones totalmente funcionales para comprobar que satisface las necesidades del usuario.
4. **Transición**: Se comprueba que el software cumple con todas las necesidades y se realizan feedback con el cliente para ajustar el software, dado una de estas fases contiene interacciones necesarias para alcanzar los objetivos del producto y cada fase tiene un objetivo y un hito que indicará que el objetivo se ha alcanzado.

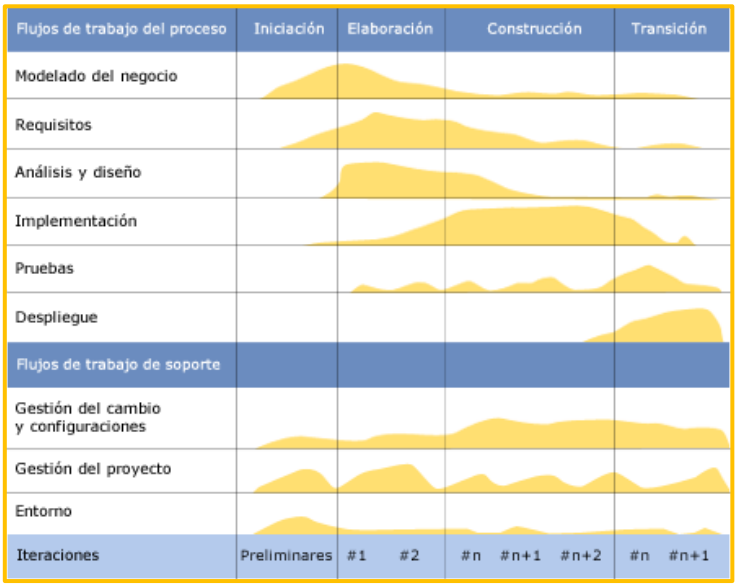


Figura.- 13: Ciclo de vida RUP (Fuente: http://es.wikipedia.org)

Cada una de estas fases se desarrollará mediante un ciclo de interacciones, éstas consisten en hacer un ciclo de vida en cascada reducido, en la que el flujo de trabajo irá variando según la fase en la que se encuentre.

Estas interacciones son llevadas a cabo bajo las disciplinas de:

1) **Disciplina de desarrollo**

- **Requerimientos**: Se trasladan las necesidades del negocio a un sistema automatizado.
- **Análisis y diseño**: Los requerimientos se trasladan a una arquitectura software.
- **Implementación**: Se crea el software adaptándolo a las necesidades.
- **Pruebas**: Se comprueba que el software actúa de forma adecuada.

2) **Disciplina de soporte:**

- **Configuración y administración de los cambios.**
- **Administración de los horarios y recursos.**
- **Ambiente de desarrollo y su administración.**

Los roles que se definen en RUP indican las tareas que tiene que hacer cada uno de los miembros del proyecto y el objetivo que se debe de conseguir.

Anthony Crain nos da una visión de los roles según su grado de detalle, donde en primera instancia se tiene una visión global de la solución y el rol asociado, y en una nueva iteración se obtiene el rol específico:

Tabla 1: Roles de RUP (Fuente: Blog de Anthony Crain).

DISCIPLINA	ROLES GENERALES	ROLES ESPECÍFICOS
Modelado de negocio	**Analista de procesos de negocio.** Descubrir todos los casos de uso de negocio.	**Diseñador de negocio**. Detallar un conjunto de los casos de uso de negocio.
Requisitos	**Analista de sistemas**. Descubrir todos los casos de uso.	**Especificador de casos de uso**. Detallar un conjunto de los casos de uso.
Análisis y diseño	**Arquitectos de Software**. Toma decisiones tecnológicas de la solución a nivel global.	**Diseñadores**. Detallan el análisis y diseño para un conjunto de casos de uso.
Implementación	**Integrador**. Es el propietario del plan de construcción que muestra como se integrarán cada una de las clases, las unas con las otras.	**Desarrollador o programador**. Implementa un conjunto de clases o un conjunto de operaciones de una clase.
Pruebas	**Gestor de las pruebas**. Asegura que las pruebas han sido realizadas correctamente. **Analista de pruebas**. Selecciona qué se va a probar según lo estimado. **Diseñador de pruebas.** Decide qué pruebas deberían ser automáticas o manuales, y crea las automáticas.	**Diseñador de pruebas**. Implementa las pruebas automáticas de la iteración. **Probador**. Ejecuta un test específico.
Despliegue o Implantación	**Gestor de la implantación**. Supervisa la implantación de todas las unidades.	**Artista gráfico, escritor tecnológico y desarrollador de material.** Crean el material necesario para asegurar la correcta implantación.
Gestión del proyecto	**Gestor del proyecto**. Crea los casos de negocio y un plan general, y toma decisiones críticas al respecto de que cosas hacer y cuales no hacer.	**Gestor de proyectos**. Planifica, monitoriza y gestiona los riesgos para una sola iteración.

Entorno	**Ingeniero de procesos**. Es el responsable de los procesos del proyecto.	**Especialista en herramientas**. Crea manuales de uso de herramientas específicas.
Configuración y Mantenimiento	**Gestor de la configuración**. Establece las políticas y planes. **Gestor de control de cambios**. Establece un proceso de control de los cambios.	**Gestor de la configuración**. Crea una unidad de despliegue o implantación, reportes del estado de la configuración, auditorias, ... **Gestor de control de cambios**. Revisa y gestiona las peticiones de cambios.

RUP se define por una serie de características denominadas prácticas y que vienen definidas en "*RUP, Best Software practices development*".

RUP, BEST SOFTWARE PRACTICES DEVELOPMENT	
Gestión de los riesgos	Decidir qué riesgos tener en cuenta en cada iteración. Actualizar la lista de riesgos y hacerla visible. Utilizar herramientas de gestión de riesgos y trazabilidad.
Desarrollos iterativos	Entregas iterativas. **Replanificación** basada en el **feedback**. Planificar las iteraciones en función de los riesgos. Usar mini y super iteraciones
Usar componentes basados en la arquitectura	Aplicar componentes y principio de diseño de servicios. Modelar componentes, servicios e interfaces. Crear especificaciones detalladas de los componentes y los servicios.
Gestión de los requisitos	Identificar escenarios. Capturar casos de uso, y relacionarlos con sus respectivos escenarios. Trocear casos de uso en requisitos gestionados independientemente. Utiliza **UML** para la realización de un modelo visual del software.
Modelado del software visual	La calidad debe ser revisada respecto a los requisitos basándose en la fiabilidad, funcionalidad, rendimiento de la aplicación y del sistema. Planificación, diseño, implementación, ejecución y evaluación de las pruebas.
Verificar la calidad del software	Controlar y hacer el seguimiento de los cambios. Utilización de entornos de trabajos independientes para aislar los diferentes cambios.
Control de los cambios del software	La metodología RUP quizás sea la más adecuada para proyectos grandes ya que necesita disponer de un equipo de trabajo que sea capaz de manejar procesos en varias etapas.

2.2.- Metodologías Ágiles

Las metodologías ágiles surgen como una alternativa a las metodología tradicionales las cuales, tal y como acabamos de ver en los apartados anteriores son demasiado burocráticas y por tanto rígidas para las actuales características del mercado.

Años atrás la evolución de los productos era lenta y se producía siempre en un entorno estable en el que apenas había variaciones.

Hoy en día sin embargo el entorno en el que se mueve el software es demasiado inestable y cambiante por lo que estas metodologías no se adaptan, ya que hay que reducir el tiempo de creación pero sin dejar de todo la calidad del software.

Las metodologías convencionales presentan para este tipo de proyectos los siguientes inconvenientes:

1. Es necesario conocer desde el inicio qué desea el cliente.
2. No se deben de cambiar los requisitos iniciales puesto que a medida que se sigue construyendo el proyecto las modificaciones y la corrección de los errores es más costosa. Además todos los cambios que se produzcan los sufrirá económicamente el cliente.

Figura.- 14

3. Se establecen mecanismos de control para el proyecto que a veces dan sensación de inflexibilidad a posibles cambios y que de hacerlo incrementaría el coste.
4. Excesiva documentación que a veces incluso no es útil.
5. Quizás este sería el punto más importante, “la lentitud” del desarrollo. Hoy en día para ser competitivos es necesario la agilidad y flexibilidad a la hora de la creación de los productos.
6. Las metodologías tradicionales se encuentran con las dificultades al final del proyecto, lo que termina retrasando las entregas.

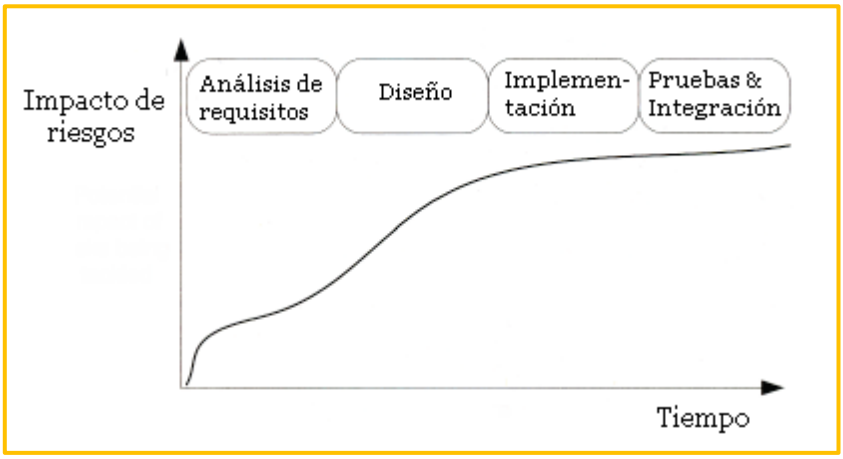


Figura.- 15

Todos estos inconvenientes han hecho que las metodologías clásicas no hayan sido capaces de eliminar los fallos y que haya una explosión de creación de software orientado a la Web, en la que se requieren cambios constantes, y que los tiempos de desarrollo sean más cortos hacen que aparezca el concepto de "metodología ágil" como una alternativa atractiva.

El desarrollo ágil está centrado en la iteración, comunicación y en reducir elementos intermedios.

El desarrollo con interacciones se realiza normalmente en porciones de tiempo pequeñas denominadas "***timeboxes***" y se ocupará de desarrollarlas un equipo multidisciplinar autoorganizado, ellos mismo decidirán como realizar las tareas de la iteración.

Además el método de desarrollo ágil fomenta la comunicación entre los miembros del equipo lo que previene problema que en otra metodología quedan escondidos.

Esta comunicación no solo se establece de forma cerrada entre los miembros del equipo sino que también se realiza con la figura que representa al cliente.

El representante del cliente es necesario como elemento de apoyo para los desarrolladores puesto que será la persona a la que se podrán hacer las preguntas necesarias y que junto con el resto de personas involucradas en el negocio comprobarán si se cumplen los objetivos.

Por lo tanto trabajar con una buena comunicación permite que se puedan tomar las decisiones de forma más rápida y aplicarlas.

La característica realmente nueva que aportan estas metodologías es reconocer a las personas como el principal valor para que un proyecto consiga terminarse de forma correcta. "*El factor más importante en el desarrollo de software no son las técnicas y las herramientas que emplean los programadores, sino la calidad de los programadores*." (**Robert. L. Class**).

Podemos deducir que las metodologías ágiles a diferencia de las metodologías tradicionales o clásicas son más adecuadas cuando el entorno presenta una cierta incertidumbre o es cambiante.

En este contexto se podrían definir las siguientes ventajas:

1. Gran capacidad de respuesta ante los cambios, los cuales no se entienden como un problema sino como algo necesario para que el producto sea mejor y satisfaga al cliente. Los cambios formarán parte del proceso de desarrollo.
2. Las entregas no se hacen al final sino que se hacen pequeñas entregas. Estas entregas permiten al cliente valorar el producto además de ir trabajando con algunas funcionalidades.
3. Los ciclos cortos de entrega ayudarán a disminuir los riesgos sobre todo al principio del proyecto.
4. Se trabaja en equipo entre el cliente y los desarrolladores mediante una comunicación casi diaria para evitar errores y documentación innecesaria.
5. Eliminar el trabajo que no es necesario y que realmente no aporta un valor al negocio.
6. Buscar la mejor técnica y el mejor diseño para conseguir productos de calidad.
7. Mejorar los procesos y al equipo que realiza el desarrollo.

2.3.- Comparativa Entre Desarrollo Ágil Y Desarrollo Tradicional

En la planificación de cualquier proyecto hay una serie de conceptos que son comunes:

1. **El triángulo de hierro**, en el que se tienen en cuenta las tres variables principales del proyecto
    - El **alcance**, definirá las tareas necesarias para alcanzar las características que deseamos obtener de nuestro producto.
    - El **tiempo**, previsión de la duración que llevará el proyecto.
    - El **presupuesto** o **coste**, dinero y recursos que se van a destinar al proyecto.

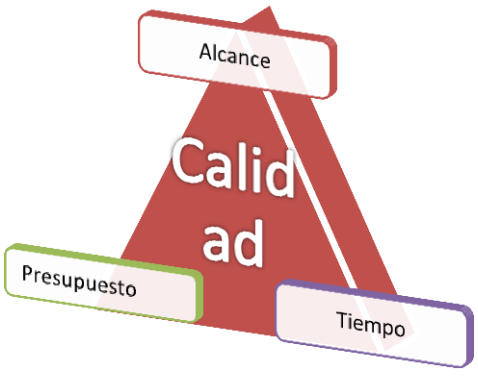


Figura.- 16: Triángulo de hierro.

La modificación de uno de los vértices de este triángulo influenciará sobre los otros 2, atacando al punto más importante que es la calidad del software o del producto.

2. Qué riesgos pude haber para poder reducirlos.
3. Definición de hitos para realizar entregas.
4. Las dependencias funcionales y la cohesión de los trabajos para ahorrar esfuerzos y realizar planificaciones coherentes.

De forma esquemática podemos presentarlas siguientes diferencias entre la metodología tradicional y la metodología ágil.

DESARROLLO TRADICIONAL

1. Se identifican las tareas al inicio del proyecto.
2. Control predictivo en que predice las variables de tiempo, alcance y presupuesto. El desarrollo se realiza en cascada, se hace una entrega final del proyecto.
3. Debido a que las entregas se hacen al final, puede que el producto no cumpla con los requisitos establecidos y sea necesario hacer cambios que comprometan a dos elementos o vértices del triángulo: coste y tiempo.
4. Proceso controlado con muchas más normas.
5. Uso de reuniones entre equipo y cliente.
6. Resistencia a los cambios.
7. No se suelen hacer análisis del trabajo realizado, denominado retrospección, en el que se intenta ver las cosas que se han realizado, de forma correcta, qué hay que mejorar y qué problemas podrían aparecer.
8. Mayor número de roles.

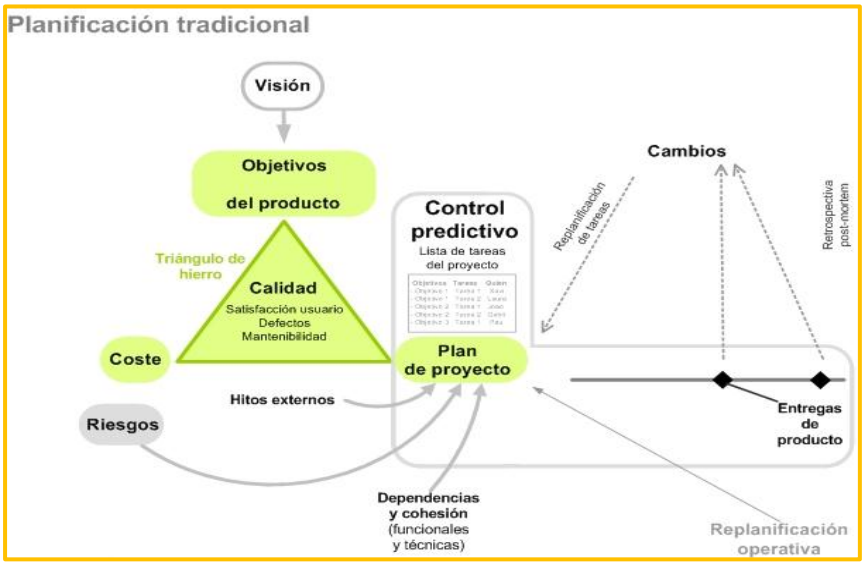


Figura.- 17: Esquema planificación tradicional (Fuente www.proyectosagiles.org)

DESARROLLO ÁGIL

1. Se basa en el control empírico, en que se asume que va a haber cambios en el contexto del proyecto, por lo tanto, el control del proyecto se basará en controlar los resultados obtenidos y en función de éstos, hacer las adaptaciones adecuadas (ciclo PDCA).
2. Las fases se plantean en función de los objetivos del producto, que suelen ser en cortos períodos de tiempo y en los que se hacen demostraciones del producto a los clientes; de esta forma es más fácil realizar los cambios.
3. El proceso no necesita de tanto control.
4. El cliente es parte del proyecto.
5. Todo el equipo participa en todas las fases del proyecto.
6. Hay menos roles.
7. Se realiza retrospectiva durante todo el proyecto.

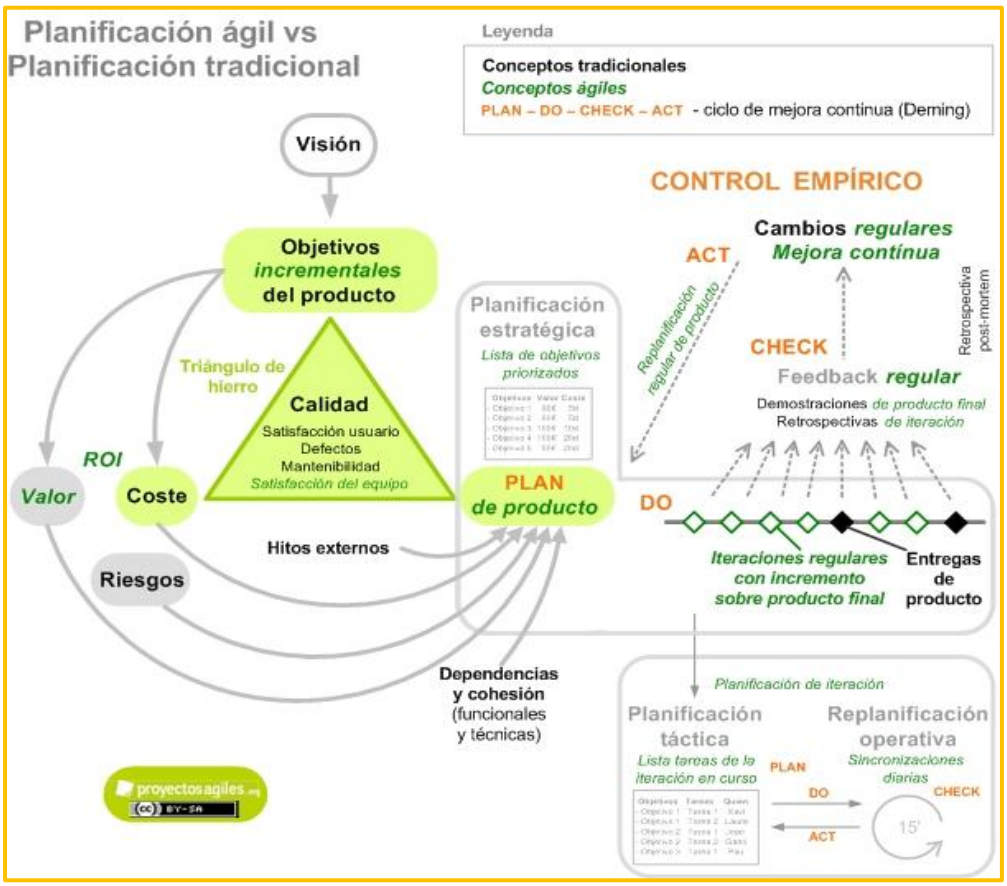


Figura.- 18: Esquema Planificación ágil (Fuente www.proyectosagiles.org)

2.4.- Manifiesto Ágil

Podríamos utilizar como definición de agilidad la ofrecida por Quomer y Henderson Selles

"La agilidad es un comportamiento persistente o habilidad, de entidad sensible, que presenta flexibilidad para adaptarse a los cambios esperados o inesperados, rápidamente; persigue la duración más corta en tiempo, usa instrumentos económicos; y utiliza los conocimientos y experiencias previos para aprender tanto del entorno interno como del externo."

Basándose en esta metodología, y entendiendo que las necesidades del entorno han cambiado el tipo del cliente y las necesidades de su empresa, en el año 2001 se produce una reunión convocada por Kent Beck – autor años antes del libro "*Extrem Programming Explained*" - en el que se dio origen al término **Métodos Ágiles**.

A causa de estos métodos ágiles, que estaban surgiendo, se definieron los 4 postulados que dan origen a:

EL MANIFIESTO

"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 iteración	Por encima de	los procesos y las herramientas.
El software que funciona	Por encima de	documentación exhaustiva.
La colaboración con el cliente	Por encima de	de la negociación contractual.
Las respuestas al cambio	Por encima de	seguimiento de un plan.

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

Jeff Sutherland inventor de Scrum en 1993, y que participó en la creación del manifiesto ágil, nos describe de la siguiente manera los 4 postulados del manifiesto en su artículo "***Principios y valores de Agile, por Jeff Sutherland ".***

El siguiente artículo fue extraído, por su interés, íntegramente para este trabajo de investigación del blog de ***Jeff Sutherland****:*

"Estos valores no son sólo algo que los creadores del Manifiesto Ágil (Agile Manifesto) pensaban encomiar y, a continuación, olvidarse. Son valores que funcionan. Cada metodología ágil individual enfoca estos valores de una manera ligeramente diferente, pero todas estas metodologías tienen procesos y ejercicios concretos que fomentan uno o más de estos valores"

Individuos e interacciones

Los individuos e interacciones son esenciales para los equipos de alto rendimiento. Los estudios de "saturación de la comunicación" durante un proyecto mostraron que, cuando no existe ningún problema de comunicación, los equipos pueden rendir 50 veces más que el promedio del sector. Para facilitar la comunicación, los métodos ágiles confían en los ciclos de inspección y adaptación frecuentes.

Estos ciclos pueden variar de cada pocos minutos con programación entre iguales, cada pocas horas con integración continua o todos los días con una reunión de puesta en marcha, hasta cada iteración con una revisión y retrospectiva.

No obstante, simplemente aumentar la frecuencia de los comentarios y la comunicación no es suficiente para eliminar los problemas de comunicación. Estos ciclos de inspección y adaptación solo funcionan bien cuando los miembros del equipo presentan varios comportamientos clave:

- **respeto por el valor de cada persona**
- **veracidad en cada comunicación**
- **transparencia de todos los datos, acciones y decisiones**
- **confianza en que cada persona respaldará al equipo**
- **compromiso con el equipo y los objetivos del equipo**

Para fomentar estos tipos de comportamiento, la administración ágil debe proporcionar un entorno de apoyo, los entrenamientos del equipo deben facilitar su inclusión y los miembros del equipo deben mostrarlos. Sólo entonces podrán lograr los equipos todo su potencial.

Acercarse a estos tipos de comportamiento es más difícil de lo que puede parecer. La mayoría de los equipos evitan la verdad, la transparencia y la confianza debido a las normas culturales o las experiencias negativas pasadas de conflictos generados por comunicadores sinceros.

Para combatir estas tendencias, la dirección y los miembros del equipo deben facilitar el conflicto positivo. Hacerlo no solo ayuda a crear un comportamiento productivo, también tiene varias ventajas más:

- **La mejora del proceso depende de que el equipo genere una lista de impedimentos o problemas en la organización, se enfrente a ellos directamente y, a continuación, los elimine sistemáticamente en orden de prioridad.**
- **La innovación solo se produce con el intercambio libre de ideas conflictivas, un fenómeno estudiado y documentado por Takeuchi y Nonaka, los padrinos de Scrum.**
- **Alinear al equipo hacia un objetivo común requiere que el equipo manifieste y resuelva las agendas conflictivas.**

- **El compromiso de trabajar juntos sólo se consigue cuando las personas acuerdan los objetivos comunes y, a continuación, se esfuerzan por mejorar personalmente y como equipo.**

Este último punto, sobre el compromiso, es especialmente importante. Sólo cuando los individuos y los equipos se comprometen, se sienten responsables para entregar un alto valor, que es la línea de base para los equipos de desarrollo de software. Las metodologías ágiles facilitan el compromiso animando a los equipos a extraer una lista de trabajo clasificada por orden de prioridad, administrar su propio trabajo y centrarse en mejorar sus prácticas de trabajo. Esta práctica es la base de la organización del equipo por sí mismo, que es la fuerza motriz para lograr resultados en un equipo ágil.

Para crear equipos de alto rendimiento, las metodologías ágiles valoran a los individuos y las interacciones sobre los procesos y herramientas. Hablando prácticamente, todas las metodologías ágiles buscan aumentar la comunicación y colaboración a través de los ciclos de inspección y adaptación frecuentes. Sin embargo, estos ciclos solo funcionan cuando los coordinadores ágiles fomentan el conflicto positivo que se necesita para generar una base sólida de verdad, transparencia, confianza, respeto y compromiso en sus equipos ágiles.

Software que funciona sobre la documentación completa

El software que funciona es una de las grandes diferencias que aporta el desarrollo ágil. Todas las metodologías ágiles que se representan en el Manifiesto Ágil (Agile Manifesto) subrayan la entrega al cliente de partes pequeñas de software que funciona a intervalos fijos.

Todos los equipos ágiles deben establecer lo que significa que digan "software que funciona", conocido con frecuencia como la definición de terminado. En un nivel alto, una parte de la funcionalidad se ha completado solo cuando sus características pasan todas las pruebas y pueden ser utilizadas por un usuario final. Como mínimo, los equipos deben ir más allá del nivel de la prueba unitaria y probar en el nivel del sistema. Los mejores equipos también incluyen pruebas de integración, rendimiento y aceptación del cliente en su definición de lo que significa haber terminado una parte de la funcionalidad.

Una compañía de CMMI nivel 5 ha mostrado, a través de datos extensos en muchos proyectos, que definir las pruebas de aceptación junto con la característica, implementar las características consecutivamente y en orden de prioridad, realizar las pruebas de aceptación en cada característica inmediatamente y corregir cualquier error identificado como prioridad máxima duplicará la velocidad de producción sistemáticamente y reducirá los defectos en un 40 por ciento. Esto procede de una compañía que ya tiene una de las tasas de defectos más bajas del mundo.

El Manifiesto Ágil (Agile Manifesto) recomienda que los equipos entreguen el software que funciona a intervalos establecidos. Acordar una definición de "terminado" es una de las maneras prácticas en que los equipos ágiles dan lugar al alto rendimiento y alta calidad que se necesitan para lograr este objetivo.

Colaboración del cliente sobre la negociación del contrato

Durante las dos últimas décadas, las tasas de éxito de los proyectos han aumentado a más del doble en todo el mundo. Esto se atribuye a los proyectos menores y las entregas frecuentes, que permiten al cliente proporcionar comentarios sobre el software que funciona a intervalos regulares. Los redactores del manifiesto sabían de qué estaban hablando cuando enfatizaron que conseguir la implicación del cliente en el proceso de desarrollo de software es esencial para el éxito.

Las metodologías ágiles fomentan este valor ya que un representante del cliente trabaja mano a mano con el equipo de desarrollo. Por ejemplo, el primer equipo de Scrum tenía miles de clientes.

Dado que no era factible implicarlos a todos en el desarrollo del producto, seleccionaron a un representante del cliente, llamado propietario del producto, para representar no solo a todos los clientes en el campo respectivo, también la a administración, ventas, soporte técnico, servicios al cliente y otras partes interesadas. El propietario del producto era responsable de actualizar la lista de requisitos cada cuatro semanas a medida que el equipo liberaba el software que funciona, teniendo en cuenta la realidad actual y los comentarios reales de los clientes y partes interesadas. Esto permitía al cliente ayudar a dar forma al software que estaban creando.

El primer equipo de XP comenzó con un proyecto de TI interno. Pudieron tener a un usuario final de la compañía en su equipo para que trabajara con ellos a diario. Las consultorías y los equipos internos pueden encontrar a un usuario final que trabaje con el equipo todos los días aproximadamente el 10 por ciento del tiempo. Para el otro 90 por ciento del tiempo, deben designar a un representante. Este representante del cliente, a quien los equipos de XP llaman "el cliente", trabaja con los usuarios finales para proporcionar una lista clara, clasificada por orden de prioridad de las características que los desarrolladores de software deben implementar.

Colaborar con el cliente (o el representante del cliente) todos los días es uno de los motivos clave por los que los datos del sector muestran que los proyectos ágiles tienen más del doble de éxito que los proyectos tradicionales por término medio en todo el mundo. Las metodologías ágiles reconocen esto y, por tanto, han creado en sus equipos de desarrollo un lugar que es específicamente para el representante del cliente.

Responder al cambio sobre seguir un plan

Responder al cambio es esencial para crear un producto que agrade el cliente y proporcione valor comercial. Los datos del sector muestran que más del 60 por ciento de los requisitos de producto o de proyecto cambian durante el desarrollo de software. Incluso cuando los proyectos tradicionales finalizan a tiempo, según el presupuesto y con todas las características del plan, los clientes se sienten a menudo insatisfechos porque lo que encuentran no es exactamente lo que deseaban. "La "Ley de Humphrey" dice que los clientes nunca saben lo que desean hasta que ven el software que funciona. Si los clientes no ven el software que funciona hasta el fin de un proyecto, es demasiado tarde para incorporar sus comentarios. Las metodologías ágiles buscan los comentarios del cliente a lo largo del proyecto para que puedan incorporar comentarios y nueva información a medida que se desarrolla el producto.

Todas las metodologías ágiles tienen procesos integrados para cambiar sus planes a intervalos regulares en función de los comentarios del cliente o su representante. Sus planes están diseñados para entregar siempre primero el mayor valor comercial. Dado que el 80 por ciento del

valor representa el 20 por ciento de las características, los proyectos ágiles bien realizados tienden a finalizar temprano, mientras que la mayoría de los proyectos tradicionales finalizan tarde. Como resultado, los clientes están más satisfechos y los desarrolladores de software disfrutan más de su trabajo. Las metodologías ágiles están basadas en el conocimiento de que para tener éxito, se debe planear para cambiar .Por eso han establecido procesos, como revisiones y retrospectivas, diseñados específicamente para desplazar las prioridades con regularidad en función de los comentarios del cliente y el valor comercial.

2.5.- Los 12 principios del manifiesto ágil.

Figura.- 19: Los 12 principios del Manifiesto Ágil.

Algunos de las metodologías ágiles más utilizadas hoy en día, por mayor orden de presencia según la mayoría de los estudios realizados son:

- **Extreme Programming.**
- **Test Drive Development.**
- **Agile Project Management.**
- **Scrum.** Será en este último será en el que centraremos el resto del trabajo de investigación.

3.- SCRUM.

3.1.- Introducción:

En el año 1986 **Takeuchi y Nonaka** publicaron el artículo "The New Product Developroent Game" el cual dará a conocer una nueva forma de gestionar proyectos en la que la agilidad, flexibilidad, y la incertidumbre son los elementos principales.

Nonaka y Takeuchi se fijaron en empresas tecnológicas que, estando en el mismo entorno en el que se encontraban otras empresas, realizaban productos en menos tiempo, de buena calidad y menos costes.

Observando a empresas como Honda, HP, Canon…etc., se dieron cuenta de que el producto no seguía unas fases en las que había un equipo especializado en cada una de ellas, si no que se partía de unos requisitos muy generales y el producto lo realizaba un equipo multidisciplinar que trabajaba desde el comienzo del proyecto hasta el final.

Se comparó esta forma de trabajo en equipo, con la colaboración que hacen los jugadores de Rugby y la utilización de una formación denominada **SCRUM**.

Scrum aparece como una práctica destinada a los productos tecnológicos y será en 1993 cuando realmente Jeff Sutherland aplique un modelo de desarrollo de Software en Ease/Corporation.

En 1996, **Jeff Sutherland** y **Ken Schwaber** presentaron las prácticas que se usaban como proceso formal para el desarrollo de software y que pasarían a incluirse en la lista de Agile Alliance.

Scrum es adecuado para aquellas empresas en las que el desarrollo de los productos se realiza en entornos que se caracterizan por tener:

1. **Incertidumbre**: Sobre esta variable se plantea el objetivo que se quiere alcanzar sin proporcionar un plan detallado del producto.
   Esto genera un reto y da una autonomía que sirve para generar una "**tensión**" adecuada para la motivación de los equipos.

2. **Auto-organización**: Los equipos son capaces de organizarse por sí solos, no necesitan roles para la gestión pero tienen que reunir las siguientes características:

- **Autonomía**: Son los encargados de encontrar la solución usando la estrategia que encuentren adecuada.
- **Autosuperación**: Las soluciones iniciales sufrirán mejoras.
- **Auto-enriquecimiento**: Al ser equipos multidisciplinares se ven enriquecidos de forma mutua, aportando soluciones que puedan complementarse.

3. **Control moderado**: Se establecerá un control suficiente para evitar descontroles. Se basa en crear un escenario de "autocontrol entre iguales" para no impedir la creatividad y espontaneidad de los miembros del equipo.

4. **Transmisión del conocimiento**: Todo el mundo aprende de todo el mundo. Las personas pasan de unos proyectos a otros y así comparten sus conocimientos a lo largo de la organización.

Scrum al ser una metodología de desarrollo ágil tiene como base la idea de creación de ciclos breves para el desarrollo, que comúnmente se llaman ***iteraciones*** y que en Scrum se llamarán "***Sprints***".

Para entender el ciclo de desarrollo de Scrum es necesario conocer las **5 fases** que definen el ciclo de desarrollo ágil:

1. **Concepto**: Se define de forma general las características del producto y se asigna el equipo que se encargará de su desarrollo.

2. **Especulación**: en esta fase se hacen disposiciones con la información obtenida y se establecen los límites que marcarán el desarrollo del producto, tales como costes y agendas.
   Se construirá el producto a partir de las ideas principales y se comprueban las partes realizadas y su impacto en el entorno.
   Esta fase se repite en cada iteración y consiste, en rasgos generales, en:

   - Desarrollar y revisar los requisitos generales.
   - Mantener la lista de las funcionalidades que se esperan.
   - Plan de entrega. Se establecen las fechas de las versiones, hitos e iteraciones. Medirá el esfuerzo realizado en el proyecto.

3. **Exploración**: Se incrementa el producto en el que se añaden las funcionalidades de la fase de especulación.

4. **Revisión**: El equipo revisa todo lo que se ha construido y se contrasta con el objetivo deseado.

5. **Cierre**: Se entregará en la fecha acordada una versión del producto deseado. Al tratarse de una versión, el cierre no indica que se ha finalizado el proyecto, sino que seguirá habiendo cambios, denominados "mantenimiento", que hará que el producto final se acerque al producto final deseado.

Figura.- 20: Ciclo de desarrollo ágil.

Scrum gestiona estas iteraciones a través de reuniones diarias, uno de los elementos fundamentales de esta metodología.

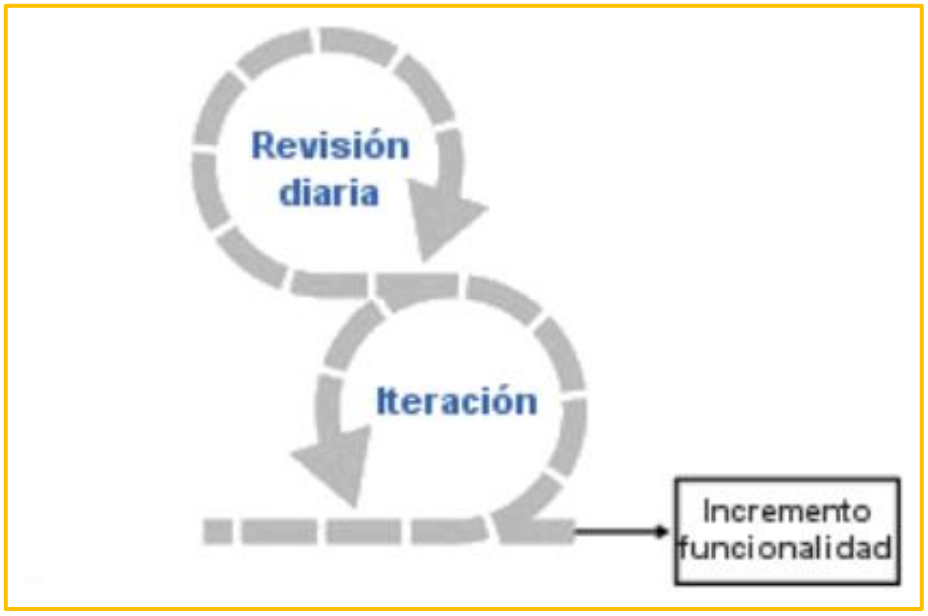


Figura.- 21: Ciclo principal de Scrum

3.2.- Componentes de Scrum.

Para entender todo el proceso de desarrollo del Scrum, se describirá de forma general las fases y los roles. Estas fases y roles se detallarán de forma más concisa más adelante.

Scrum se puede dividir de forma general en 3 fases, que podemos entender como ***reuniones***. Las reuniones forman parte de los artefactos de esta metodología junto con los roles y los elementos que lo forman.

3.2.1.- Las Reuniones.

1. **Planificación del Backlog**

Se definirá un documento en el que se reflejarán los requisitos del sistema por prioridades.

En esta fase se definirá también la planificación del **Sprint 0,** en la que se decidirá cuáles van a ser los objetivos y el trabajo que hay que realizar para esa iteración.

Se obtendrá además en esta reunión un **Sprint Backlog,** que es la lista de tareas y que es el objetivo más importante del Sprint.

2. **Seguimiento del Sprint**

En esta fase se hacen reuniones diarias en las que las 3 preguntas principales para evaluar el avance de las tareas serán:

- **¿Qué trabajo se realizó desde la reunión anterior?**
- **¿Qué trabajo se hará hasta una nueva reunión?**
- **Inconvenientes que han surgido y qué hay que solucionar para poder continuar**.

3. **Revisión del Sprint**

Cuando se finaliza el Sprint se realizará una revisión del incremento que se ha generado. Se presentarán los resultados finales y una demo o versión, esto ayudará a mejorar el **feedback** con el cliente.

3.2.2.- Los Roles.

Los roles se dividen en 2 grupos: cerdos y gallinas, esto surge en el chiste sobre un cerdo y una gallina y su intención de poner un restaurante.

1. **LOS CERDOS**

Son las personas que están comprometidas con el proyecto y el proceso de Scrum.

- **Product Owner**: Es la persona que toma las decisiones, y es la que realmente conoce el negocio del cliente y su visión del producto. Se encarga de escribir las ideas del cliente, las ordena por prioridad y las coloca en el Product Backlog.
- **ScrumMaster**: Es el encargado de comprobar que el modelo y la metodología funciona. Eliminará todos los inconvenientes que hagan que el proceso no fluya e interactuará con el cliente y con los gestores.
- **Equipo De Desarrollo**: suele ser un equipo pequeño de unas 5-9 personas y tienen autoridad para organizar y tomar decisiones para conseguir su objetivo. Está involucrado en la estimación del esfuerzo de las tareas del Backlog.

2. **LAS GALLINAS**

Aunque no son parte del proceso de Scrum, es necesario que parte de la retroalimentación dé la salida del proceso y así poder revisar y planear cada sprint.

- **Usuarios:** Es el destinatario final del producto.
- **Stakeholders:** Las personas a las que el proyecto les producirá un beneficio. Participan durante las revisiones del Sprint.
- **Managers**: Toma las decisiones finales participando en la selección de los objetivos y de los requisitos.

3.3.- Elementos de Scrum.

Los elementos que forman a Scrum son:

- **Product Backlog:** lista de necesidades del cliente.
- **Sprint Backlog:** lista de tareas que se realizan en un Sprint.
- **Incremento:** parte añadida o desarrollada en un Sprint, es un parte terminada y totalmente operativa.

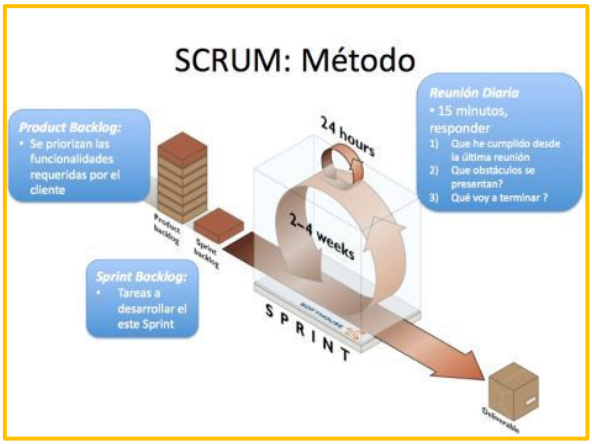


Figura.- 22: Ciclo de desarrollo Scrum.

3.3.1.- Product Backlog.

Es el inventario en el que se almacenan todas las funcionalidades o requisitos en forma de lista priorizada. Estos requisitos serán los que tendrá el producto o los que irá adquiriendo en sucesivas iteraciones.

La lista será gestionada y creada por el cliente con la ayuda del Scrum Master, quien indicará el coste estimado para completar un requisito, y además contendrá todo lo que aporte un valor final al producto.

Las tres características principales de esta lista de objetivos serán:

- **Contendrá los objetivos del producto, se suele usar para expresarlos las historias de usuario.**
- **En cada objetivo, se indicará el valor que le da el cliente y el coste estimado; de esta manera, se realiza la lista, priorizando por valor y coste, se basará en el ROI.**
- **En la lista se tendrán que indicar las posibles iteraciones y los releases que se han indicado al cliente.**
- **La lista ha de incluir los posibles riesgos e incluir las tareas necesarias para solventarlos.**

Es necesario que antes de empezar el primer Sprint se definan cuáles van a ser los objetivos del producto y tener la lista de los requisitos ya definida. No es necesario que sea muy detallada, simplemente deberá contener los requisitos principales para que el equipo pueda trabajar. Realizar este orden de tareas tiene como beneficios:

- **El proyecto no se paraliza simplemente por no tener claro los requisitos menos relevantes, y el cliente podrá ver resultados de forma más rápida.**
- **Los requisitos secundarios aparecerán a medida que se va desarrollando el proyecto, por lo tanto, no se pierde tanto tiempo en analizarlos al principio y el cliente será más consciente de sus necesidades.**
- **Los requisitos secundarios puede que no se lleguen a necesitar porque se han sustituido o porque no reportan un retorno ROI interesante.**

Una vez definidos los requisitos se tendrá que acordar cuándo se tiene que entender un objetivo como terminado o completado.

Se entiende que un producto está completado si:

- **Asegura que se puede realizar un entregable para realizar una demostración de los requisitos y ver qué se han cumplido.**
- **Incluirá todo lo necesario para indicar que se está realizando el producto que el cliente desea.**

Como complemento a la definición de completado, se debería de asociar una condición de aceptación o no aceptación a cada objetivo en el mismo momento en el que se crea la lista.

Finalmente el Product Backlog irá evolucionando mientras el producto exista en el mercado. Esta es la forma para evolucionar y tener un valor de producto para el cliente suficiente para ser competitivo.

3.3.1.1 Las historias de Usuario.

Son las descripciones de las funcionalidades que va a tener el software.

Estas historias de usuario, serán el resultado de la colaboración entre el cliente y el equipo, e irán evolucionando durante toda la vida del proyecto.

Las historias de usuario se componen de tres fases denominadas "Las 3 C":

- **Card:** Será una breve descripción escrita que servirá como recordatorio.
- **Conversation:** Es una conversación que servirá para asegurarse de que se ha entendido bien todo, y concretar el objetivo.
- **Confirmation:** Tests funcionales para fijar detalles que sean relevantes e indicar cuál va a ser el límite.

En cuanto al formato, un modelo podría ser como el que se muestra en la siguiente imagen:

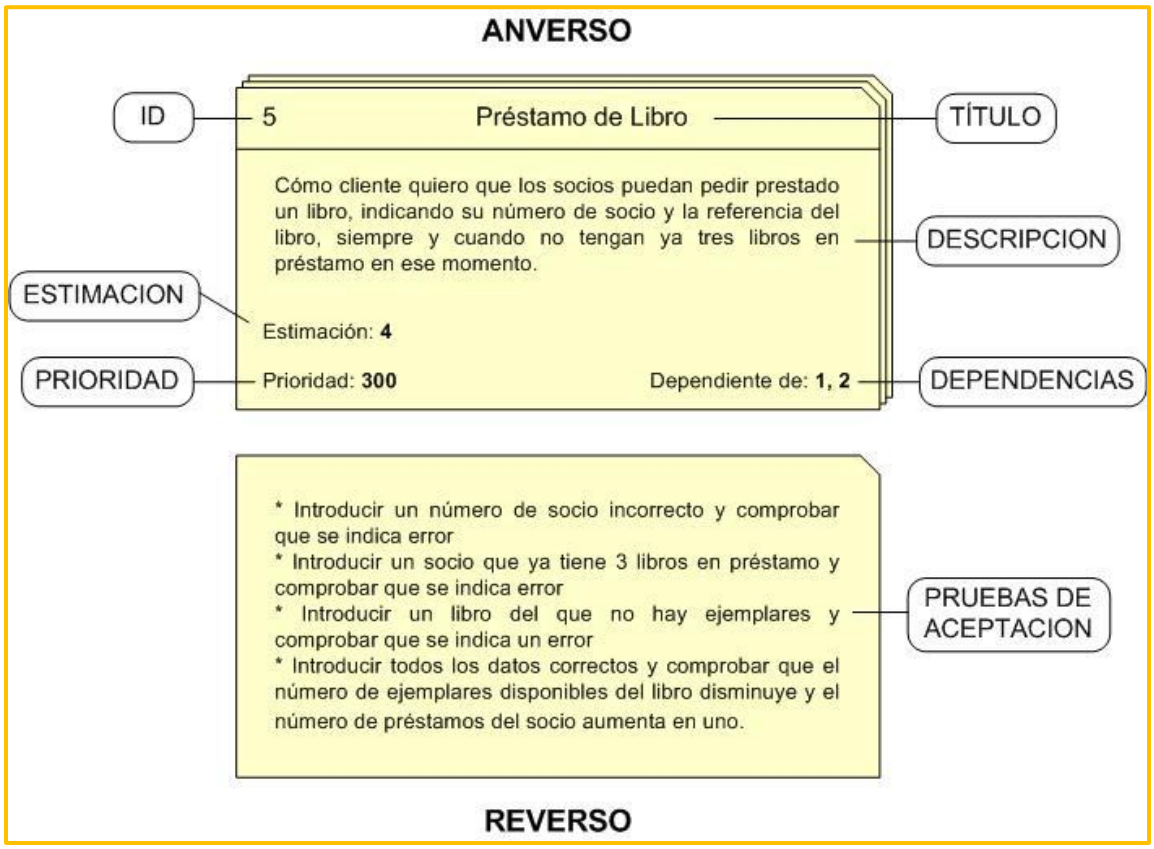


Figura.- 23: Ejemplo de Historia de usuario (fuente: http://devnettips.blogspot.com.es/)

- **ID:** Identificador de la historia de usuario.
- **TÍTULO:** Título descriptivo de la historia de usuario.

- **DESCRIPCIÓN:** Descripción sintetizada de la historia de usuario.
- **ESTIMACIÓN:** Evaluación del coste de implementación en unidades de desarrollo. Estas unidades representarán el tiempo teórico (desarrollo/hombre) que se haya estimado al comienzo del proyecto.
- **PRIORIDAD:** Prioridad en la implementación de la historia de usuario respecto al resto de las historias de usuario. A mayor número, mayor prioridad. Otra aproximación a la priorización de tareas se hace a través del método **MoSCoW**:

  M – Must, se debe completar este requerimiento para finalizar el proyecto.

  S – Should, se debe completar este proyecto por todos los medios, pero el éxito del proyecto no depende de él.

  C – Could, se debería completar este requerimiento si su implementación no afecta a la consecución de los objetivos principales del proyecto.

  W – Would, se puede completar este requerimiento si sobra tiempo de desarrollo (o en futuras versiones del mismo)
- **DEPENDENCIAS**: Una historia de usuario no debería ser dependiente de otra historia, pero a veces es inevitable. En este apartado se indicarían los IDs de las tareas de las que depende una tarea.

3.3.1.2 Formato de la Pila Del Producto (Product Backlog).

En Scrum, la preferencia por tener documentación en todo momento es menos estricta. Se encuentra más necesario el mantener una comunicación directa con el equipo, por eso se usa como herramienta el Backlog.

Aunque no hay ningún producto especial a la hora de confeccionar la lista, es conveniente que incluya información relativa a:

- **Identificador para la funcionalidad.**
- **Descripción de la funcionalidad.**
- **Sistema de priorización u orden.**
- **Estimación.**

Id	Prioridad	Descripción	Est.	Por
1	Muy alta	Plataforma tecnológica	30	AR
2	Muy alta	Interfaz usuario	40	LR
3	Muy alta	Un usuario se registra en el sistema	40	LR
4	Alta	El operador define el flujo y textos de un expediente	60	AR
5	Alta	Etc...	999	XX

Figura.- 24: Ejemplo de un Product Backlog(Fuente: Srum Manager. Gestión de Proyectos)

3.3.2.- Sprint Backlog.

Es la lista de tareas que elabora el equipo durante la planificación de un Sprint. Se asignan las tareas a cada persona y el tiempo que queda para terminarlas.

De esta manera el proyecto se descompone en unidades más pequeñas y se puede determinar o ver en qué tareas no se está avanzando e intentar eliminar el problema.

Requisito	Tarea	Quien	Estado (No iniciada / en progreso / completada)	Dia:	1	2	3	4	5	6	7	8	9	10
				Horas pendientes	1120	1088	1076	1048	1040	1032	1020	1008	992	972
Requisito A	Tarea 1	Joao	Completada		16	8								
Requisito A	Tarea 4	Laura	Completada		4									
Requisito A	Tarea 5	Laura	Completada		4									
Requisito A	Tarea 3	Gabri	Completada		8									
Requisito A	Tarea 2	Laura	Completada		16	8	4							
Requisito A	Tarea 6	Gabri	Completada		8	8	8							
Requisito A	Tarea 7	Joao	Completada		16	16	16	8						
Requisito A	Tarea 8	Laura	Completada		8	8	8							
Requisito A	Tarea 9	Laura	Completada		8	8	8	8	8					
Requisito A	Tarea 10	Laura	Completada		8	8	8	8	8	8	4			
Requisito A	Tarea 11	Joao	Completada		16	16	16	16	16	16	8			
Requisito B	Tarea 12	Gabri	Completada		16	16	16	16	16	16	16	16	8	
Requisito B	Tarea 13	Laura	Completada		16	16	16	16	16	16	16	16	8	
Requisito B	Tarea 14	Joao	En progreso		8	8	8	8	8	8	8	8	8	4
Requisito B	Tarea 15	Gabri	En progreso		8	8	8	8	8	8	8	8	8	8
Requisito B	Tarea 16	Laura	En progreso		8	8	8	8	8	8	8	8	8	8
Requisito C	Tarea 17	Joao	No iniciada		4	4	4	4	4	4	4	4	4	4
Requisito C	Tarea 18	Gabri	No iniciada		8	8	8	8	8	8	8	8	8	8
Requisito C	Tarea 19	Laura	No iniciada		16	16	16	16	16	16	16	16	16	16
Requisito C	Tarea 20	Joao	No iniciada		8	8	8	8	8	8	8	8	8	8

Figura.- 25: Ejemplo de Sprint Backlog.

Cómo funciona la lista:

- **Es una lista ordenada por prioridades para el cliente.**
- **Puede haber dependencias entre una tarea y otra, por lo tanto se tendrá que diferenciar de alguna manera.**
- **Todas las tareas tienen que tener un coste semejante que será entre 4-16 horas.**

Formato de la lista:

Hay 3 opciones:

- **Hojas de cálculo.**
- **Pizarras.**
- **Herramientas colaborativas.**

Generalmente, las tareas a completar se suelen gestionar mediante el Scrum Taskboard, a cada objetivo se le asignan las tareas necesarias para llevarlo a cabo, se usan post-its que se van moviendo de una columna a otra para cambiar el estado.

Se debe incluir:

- **Lista de tareas.**
- **Persona responsable de cada tarea, el estado en el que se encuentra y el tiempo que queda por terminarla.**

- **Permite la consulta diaria del equipo.**
- **Permite tener una referencia diaria del tiempo que le queda a cada tarea.**

3.3.3.- Incremento.

Representa los requisitos que se han completado en una iteración y que son perfectamente operativos.

Según los resultados que se obtengan, el cliente puede ir haciendo los cambios necesarios y replanteando el proyecto.

4.- DESARROLLO DE LAS FASES DE UN PROYECTO EN SCRUM.

4.1.- Preparación del proyecto.

Conocida como **Sprint 0,** es la fase inicial en la que se intenta comprender el caso de negocio con la finalidad de tomar decisiones que agreguen valor al producto.

Durante esta fase se producen gran número de inexactitudes con las estimaciones, pero es lógico, debido a que se hacen a alto nivel, por lo tanto es aconsejable no perder tiempo en buscar las estimaciones exactas, es mejor invertir ese tiempo en el desarrollo del producto. De esta manera el Backlog del producto usará como unidad de tiempo "días".

Las tareas a realizar en el sprint O son:

- **Definir el proyecto**: Se debería de indicar de forma clara el propósito del proyecto, no es necesario entrar en detalle pero sí que todo el equipo sea capaz de entender cuáles son las necesidades del producto y del cliente.
- **Definir "terminado"**: Marcará el punto en el que se va a considerar que la tarea está terminada.
- **Definición del Backlog inicial**: Se comienza la creación del Backlog del producto para que el Sprint siguiente contenga elementos de la lista suficientes para comenzar a trabajar. Esta lista de elementos será marcada por el Product Owner, que tendrá como responsabilidad priorizar las funcionalidades que, al desarrollarlas e implementarlas cumplan las especificaciones consiguiendo además que su beneficio supere a su coste.
- **Definición de los entregables**: Una vez que se tiene el Backlog con las funcionalidades, es necesario establecer criterios para hacer pequeñas entregas "entregables" del producto y así obtener su valor y un feedback temprano.

El plan de entregables puede sufrir modificaciones, muchas de ellas propiciadas por:

- **Cambia el entorno y aparecen nuevas oportunidades para el negocio.**
- **Se encuentran nuevas funcionalidades y estas proporcionan un valor si se pusiese en producción.**
- **Replanteamiento del entregable.**

El plan de entregas lo determinará el negocio, que será el encargado de marcar qué funciones, calidad y coste va a tener.

De la misma manera, estará sujeto a unas determinadas condiciones determinadas por el equipo de trabajo que serán:

- **Tiempo para un entregable.**
- **La estimación inicial.**
- **Selección del Backlog del producto.**

Constitución del equipo:

Se hace una reunión inicial con todos los roles del equipo para tratar:

- **Dimensión del proyecto.**
- **Revisiones del Backlog.**
- **Organización del equipo y horario para establecer reuniones de control.**

4.1.1.- Las Estimaciones del Backlog

Antes de la primera reunión de la planificación el Equipo tiene que conocer cuál va a ser su velocidad inicial y su factor de dedicación.

Para poder realizar las estimaciones, primeramente hay que decidir qué historias incluir en la pila del Sprint.

La forma en la que se podrá decidir qué historias poner o no se puede realizar mediante 2 técnicas:

- **De forma aproximada.**
- **Realizando cálculos de velocidad.**

1. **De forma aproximada:**

En la estimación aproximada, el equipo debate el número de historias a introducir hasta llegar a un acuerdo. Suele funcionar de forma correcta si los Sprints son cortos.

2. **Realizando los cálculos de velocidad:**

Esta técnica se realiza en dos pasos:

- **Seleccionar la velocidad estimada.**
- **Calcular el número de historias que se pueden añadir sin pasar la velocidad estimada.**

La manera más adecuada de estimar la velocidad, es revisar el historial del equipo, de esta manera, basándonos en Sprints anteriores se podrá deducir que será muy similar.

Para poder realizar esta técnica, es necesario que los equipos tengan un historial, que vayan hacer los próximos Sprints de la misma manera, mismo tamaño de equipo y las mismas condiciones de trabajo. La técnica se conoce como "el tiempo que hizo ayer".

Otra forma de hacerlo es mediante el cálculo de recursos.

1º. Se calcula el número de días – hombre disponible, este es un valor poco real porque influyen factores externos de ahí que tengamos que usar el "factor de dedicación".

Nombre	Días
USER 1	15
USER 2	13
USER 3	7
USER 4	13
TOTAL	48 Días-Hombre Disponibles

⇒ Planificación de un Sprint de 3 semanas con disponibilidades distintas

VELOCIDAD ESTIMADA

$$(Dias - hombre\ disponibles) * (factor\ de\ dedicación) = VELOCIDAD\ ESTIMADA$$

El factor de dedicación se basa en estimar el estado del equipo, si es bajo entonces se espera encontrar dificultades.

La forma más adecuada para determinar factores de dedicación razonable es el estado de los últimos Sprints.

FACTOR DE DEDICACIÓN

$$(Factor\ de\ dedicación) = \frac{Velocidad\ Real}{Días - hombre\ disponible}$$

- **Velocidad real** = suma de las estimaciones iniciales que se realizaron completamente en el último Sprint.

Ejemplo:

Si tenemos un equipo que completó 20 historias de usuario, con 3 personas sumando 35 días-hombre, para calcular el próximo Sprint en el que habrá un componente más, sumando 45 días-hombre y teniendo en cuenta que en el último Sprint se completaron 20 puntos, el resultado sería:

$$57\% = \frac{20\ ptos}{35\ dias-hombre} \qquad 45\ días - hombre * 57\% = 25\ ptos.$$

De esta manera se obtendrá la velocidad estimada por el siguiente Sprint.

Si el equipo fuese nuevo, se puede mirar el factor de dedicación de otros equipos con lo que fijarte y de no haberlo, se pondría un valor aproximado para empezar por defecto. El factor que se suele usar es de 70%.

4.2.- Planificar un Sprint.

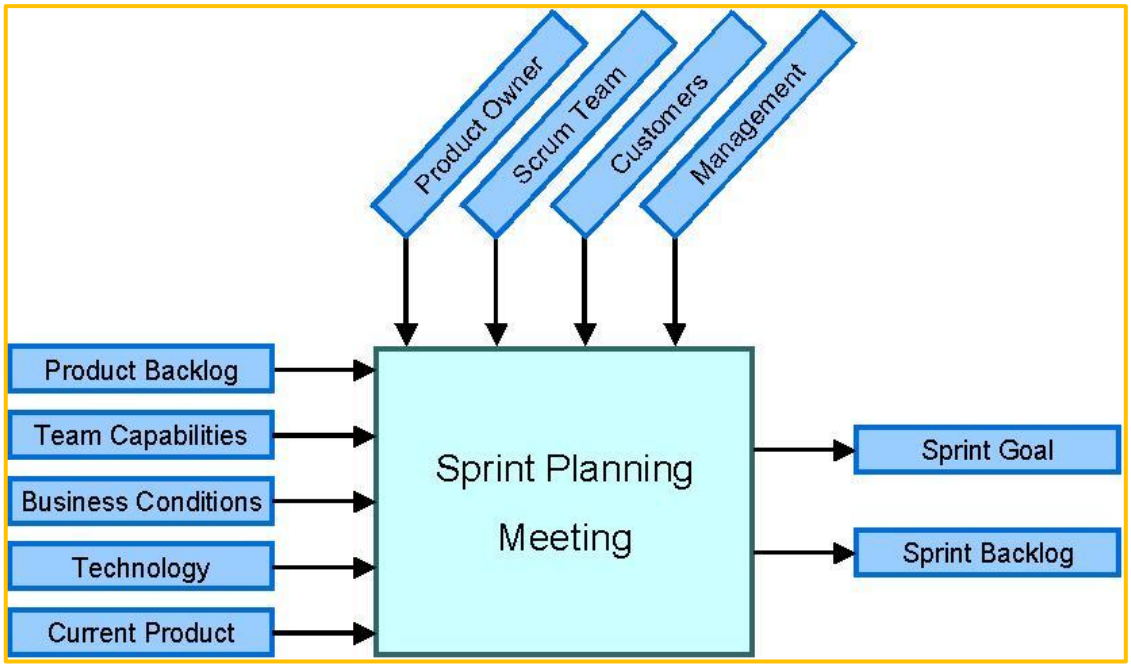


Figura.- 26: Entras/Salidas de un Planning Meeting.

Denominado también "**Sprint Planning Meeting**", tiene como finalidad realizar una reunión, en la que participarán el Product Owner, el Scrum Master y el equipo, con la intención de seleccionar de la lista Backlog del producto las funcionalidades sobre las que se va a trabajar, y que darán valor al producto.

Antes de comenzar la reunión el **Product Owner** tendrá que preparar el Backlog.

La reunión se realiza en con time-box de ocho horas que se divide en 2 partes de 4 horas.

Primera parte de la reunión:

- **El equipo selecciona los items para transformarlos en entregables.**
- **El equipo hace sugerencias, pero es el Product Owner el que decidirá si formarán parte del Sprint.**
- **El equipo seleccionará el elemento a implementar, de los seleccionados por el Product Owner para ese Sprint**.

Segunda parte de la reunión:

- **El equipo hará las preguntas necesarias que tengan sobre el Product Baklog al Product Owner.**
- **El equipo se encargará de encontrar la solución adecuada para transformar la parte seleccionada de una funcionalidad entregable.**

El resultado de la segunda parte de la reunión es una lista denominada "Sprint Backlog" con las tareas, estimaciones y las asignaciones de trabajo al equipo para poder empezar a desarrollar la funcionalidad.

Las características del Backlog del producto serán las siguientes:

- **La estimación será entre 4 y 16 horas, si son mayores a este valor, se descompondrán.**
- **Las tareas del Sprint deben de ser como consecuencia de la necesidad de un requerimiento del Backlog del producto.**
- **El progreso de las tareas en las que se mide la velocidad y el progreso del Sprint con respecto a las horas, se realizará mediante gráficos Burndown Chart.**

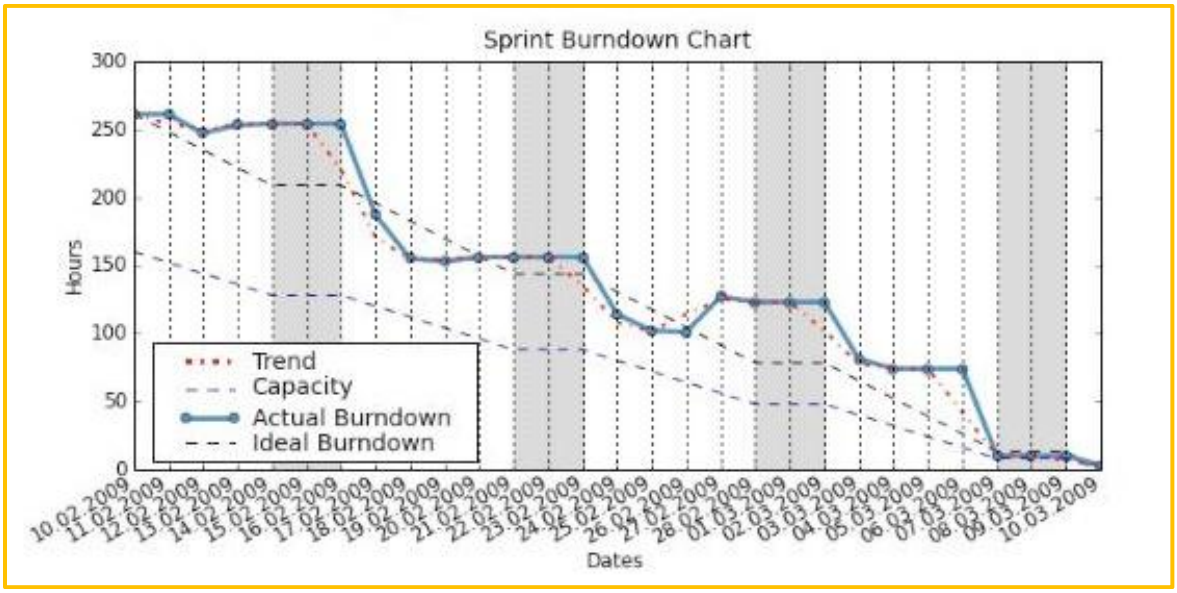


Figura.- 27: Gráfico de avance de un Sprint

El soporte en el que se presentan el Sprint Backlog puede ser:

- **Hoja de cálculo.**
- **Pizarra.**
- **Herramientas colaborativas de red.**

La herramienta como soporte quizás más utilizada es el (**Scrum Taskboard**) en la que en una pizarra se crea una tabla con la siguiente información:

Figura.- 28: Ejemplo de Scrum Task Board(Fuente: Proyectos ágiles.org)

- **1ª fila**: Tareas que no se han planificado pero que son necesarias de hacer por su urgencia (errores, cambios importantes decididos por el cliente, etc).
- **2ª fila**: Mejora continua. Se ponen las tareas de retrospectivas anteriores que se quieren analizar en ese Sprint.
- **Columna tareas no empezadas**: Se pondrán todas las tareas que se han especificado en la reunión del Sprint planning.
- **En curso**: Tareas que se están realizando y que deberían ser mínimas y resueltas de arriba abajo.
- **Hecho**: Tarea realizada completamente.
- **Impedimentos**: Lista de obstáculos que impedirán continuar de forma adecuada el proyecto. Hay que indicar quién será el responsable de solucionarlos.
- **Retrospectiva**: Sirve para anotar qué partes están bien durante la iteración y cuáles mal. Se reflejan con un "+" y un "-"

Otro ejemplo de Scrum TaskBoard sería:

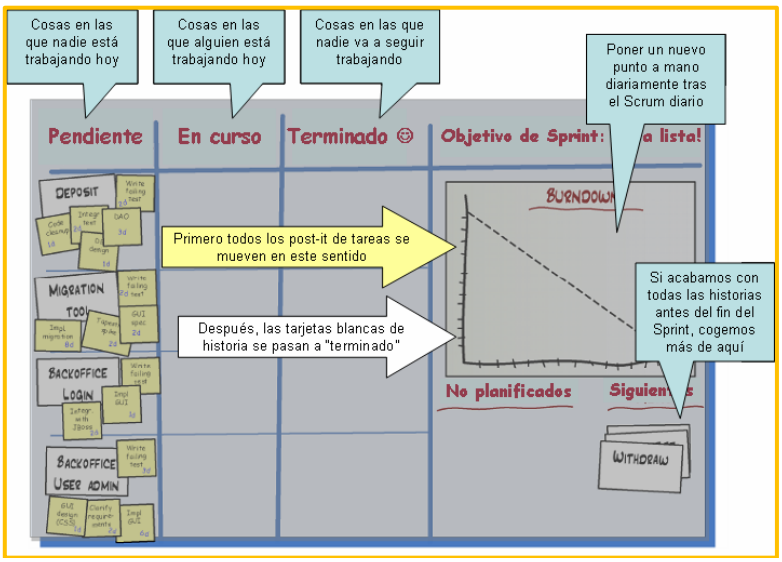


Figura.- 29: Scurm TaskBoard(Fuente: Scrum y Xp desde las trincheras)

4.2.1.- La Estimación del Sprint.

4.2.1.1 Planificación De Póker.

Realizar las estimaciones para los ítems seleccionados, es una tarea que deberá involucrar a todos los miembros del equipo. Para poder hacer una estimación y que los miembros del equipo no estén condicionados por la estimación de los compañeros se usará, la técnica de **Planning Poker**.

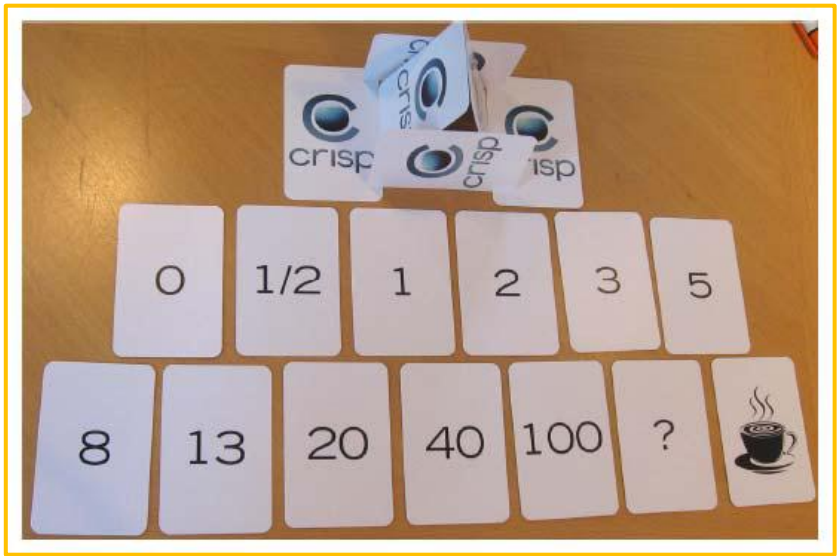


Figura.- 30

1º. Cada miembro del equipo tendrá una baraja de 13 cartas.

2º. Se propone una historia, el miembro del equipo selecciona una carta y la coloca boca abajo. Esta carta representará su estimación para la historia propuesta.

3º. Cuando todos los miembros han seleccionado su carta, se le da la vuelta al mismo tiempo.

4º. Se comprueban las estimaciones, y si hay muchas discrepancias se discute sobre esas diferencias y se ponen en común las ideas sobre la naturaleza del trabajo. Este proceso se repetirá hasta que las estimaciones sean parejas o aproximadas.

5º. Se comprueban las estimaciones y si hay muchas discrepancias se discute sobre esas diferencias y se ponen en común las ideas sobre la naturaleza del trabajo. Este proceso se repetirá hasta que las estimaciones sean parejas o aproximadas.

6º. El tiempo estimado es para el desarrollo de toda la historia, por ese motivo la secuencia de números no es lineal y, por ejemplo, hay un salto entre 40 y 100. De esta manera se evita sensaciones falsas para estimaciones grandes.

Si las estimaciones fuesen más detalladas las historias se podrían dividir en historias más pequeñas.

Hay 3 cartas especiales:

- **O** → "Esta historia ya está hecha" o requiere poco tiempo de trabajo.
- **¿** → No hay idea de cuánto puede ser la estimación.
- **Taza de café:** Realizar un descanso para retomar después la estimación.

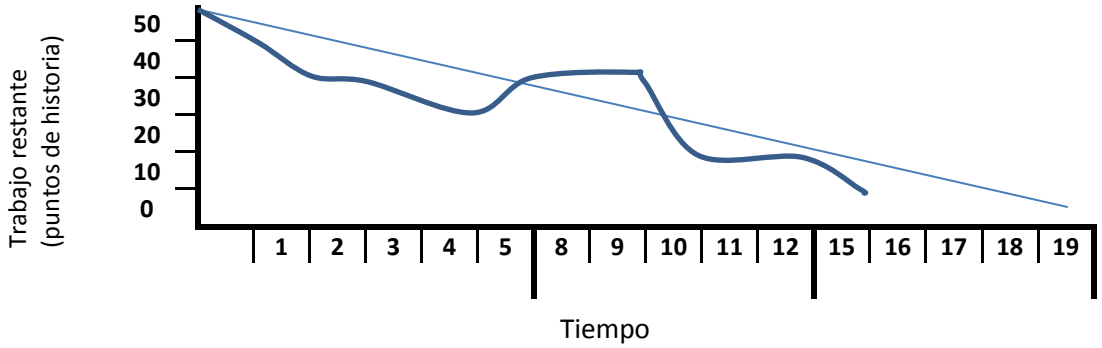
4.2.1.2 Mantener el Backlog del Sprint.

El equipo tiene que mantener actualizado el Backlog del Sprint para poder tener feedback y tomar cualquier decisión de manera rápida.

Es necesario, además, para que el gráfico de Burdown del Sprint también esté actualizado.

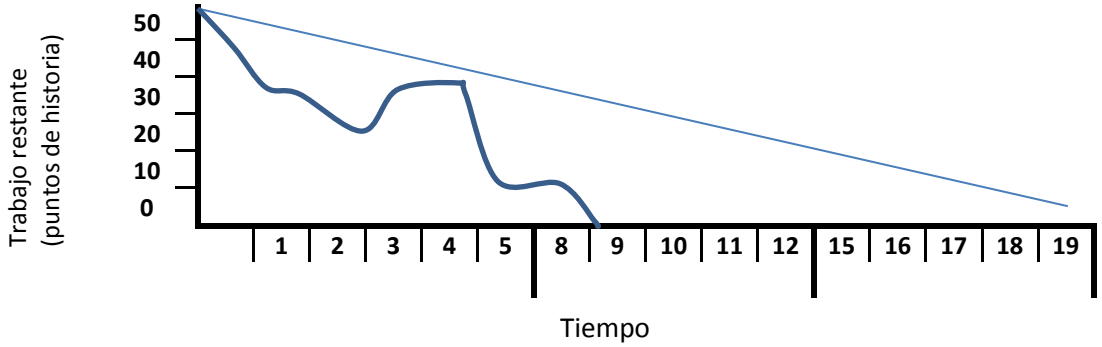
4.2.1.3 Interpretación del diagrama de Burndown.

Ejemplo 1:



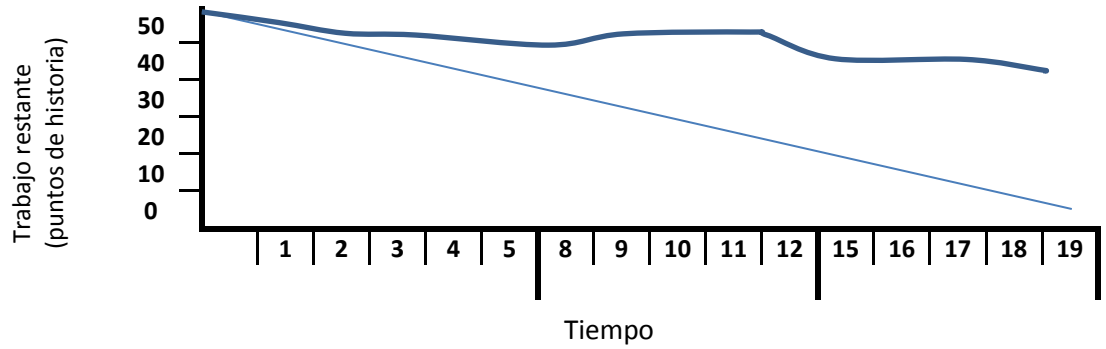
El día1 de Enero se estiman 7 puntos de historia, y a día 16 se observa que, incluso se está bien de tiempo y que según estos datos, completaríamos el Sprint. Hay que tener en cuenta que se saltan los fines de semana para no generar confusiones en la lectura del diagrama.

Ejemplo 2:



Si el gráfico que se genera es similar al del ejemplo 2, puede ocurrir que los puntos de historia se hagan más rápido de lo previsto, con lo que se podrían añadir más puntos de historia.

Ejemplo 3:



En este caso, lo que nos muestra el gráfico es que se han escogido demasiados ítems para realizarlo y habría que analizar qué ítems del Backlog habría que eliminar en esta fase.

En los Sprints, el equipo trabaja para conseguir un incremento del producto, que será productivo para el Product Owner y los Stakeholders.

El tiempo más conveniente está entre 2 y 4 semanas, o 30 días consecutivos como máximo. Estos intervalos de tiempo, son los que se consideran más apropiados para que el Stakeholders no pierda el interés.

4.3.- El desarrollo del Sprint

En los Sprints, el equipo trabaja para conseguir un incremento del producto, que será productivo para el Producto Owner y los Stakeholders.

El tiempo más conveniente está entre 2 y 4 semanas, o 30 días consecutivos como máximo. Estos intervalos de tiempo, son los que se consideran más apropiados para que el Stakeholders no pierda el interés.

4.3.1.- Reuniones del Sprint

Durante la ejecución del Sprint se van a realizar 3 reuniones:

- **Reunión de Planificación** (Sprint Planning Meeting).
- **Reunión diaria** (Scrum Daily Meeting).
- **Reunión Revisión del Sprint** (Sprint Review Meeting).

4.3.1.1 Reunión de Planificación (Sprint Planning Meeting)

Definirán qué tareas se tienen que realizar y cuáles son los objetivos.

Una vez definidos, el equipo comienza su desarrollo, pero teniendo en cuenta una serie de normas:

- **El equipo puede realizar consultas de agenda fuera del Sprint.**
- **No se permite a nadie gobernar al equipo durante el Sprint. El equipo se autogestionará.**

- **Si durante el desarrollo del Sprint no se puede realizar, porque no es viable, se puede realizar una nueva planificación para realizar un nuevo Sprint.**
- **Si el equipo no puede comprometerse a cumplir todo el Backlog, realizará una consulta con el Producto Owner para decidir qué ítems eliminar.**
- **Si de la misma manera, el equipo se ve capaz de realizar más ítems del Backlog durante el Sprint, que el indicado inicialmente, consultará también con el Product Owner qué ítems se podrán añadir.**

Teniendo en cuenta estas características, a lo largo del desarrollo, se producirán además, reuniones diarias.

4.3.1.2 Reunión Diaria (Sprint Daily Meeting)

En esta reunión, los componentes del equipo comparten información relativa al desarrollo y colaborarán para hacer las adaptaciones necesarias, aumentando así su productividad.

En esta reunión se tendrá como referencia el Backlog del Sprint y el equipo gráfico burn-down con la información de la reunión anterior y, además, qué tareas hizo cada persona del equipo. La reunión no podrá consumir más de 15 minutos y contestará a tres preguntas básicas:

- **¿Qué se ha hecho de nuevo con respecto a la última reunión diaria?**
- **¿Qué será lo siguiente a realizar?**
- **¿Qué problemas hay para realizarlos?**

Se usará como herramienta de apoyo, con la lista de tareas del Sprint actualizada y con el esfuerzo pendiente de cada tare. También se tendrá un gráfico con las tareas pendientes en la iteración.

4.3.1.3 Reunión Revisión del Sprint (Sprint Review Meeting)

Figura.- 31

En esta reunión, los desarrolladores presentan el producto entregable que han implementado y, los gestores, clientes, usuarios y Product Owner analizan esa entrega y escuchan al equipo sobre los problemas que han tenido durante el proceso.

Esta reunión servirá para tomar decisiones que ayudan a escoger el camino más adecuado para alcanzar las metas.

Las características de esta reunión se limitan a:

- **Máximo 4 horas.**
- **Se presenta el producto "completado", entendiendo como tal la definición acordada con el Product Owner y los Stakeholders.**
- **Si la funcionalidad no está completa no se puede presentar.**
- **Artefactos que no son funcionalidades, no se presentan para no equivocar a los Stakeholders.**
- **Se presenta la funcionalidad en equipos que pertenezcan a los desarrolladores. Siempre se hará desde un servidor lo más parecido posible al de producción.**

El Sprint Review transcurre siguiendo el siguiente proceso:

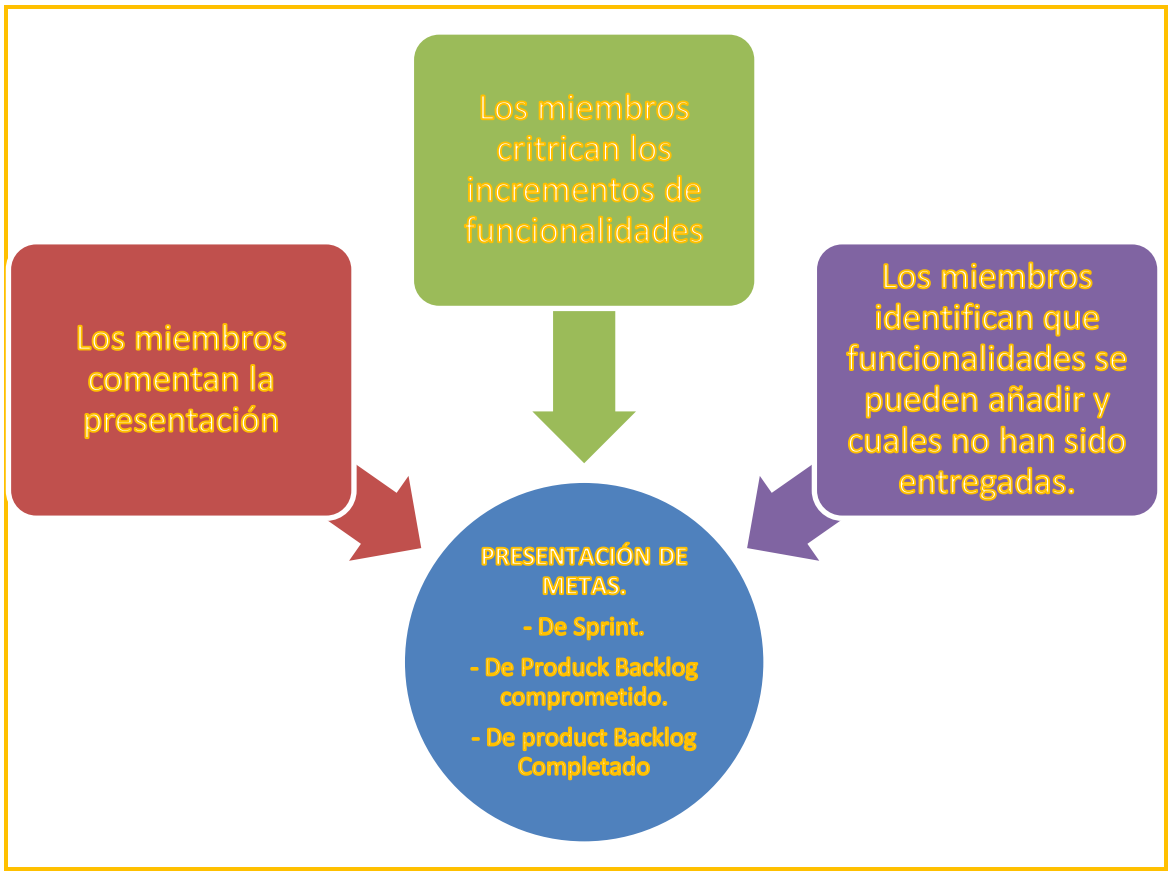


Figura.- 32

- **El ScrumMaster determina cuántas personas asistirán al Sprint Review Meeting.**
- **Al final de la reunión, el ScrumMaster anuncia al Product Owner y a los Stakeholders la próxima reunión.**

Estas reuniones son beneficiosas para todos, puesto que se comprueba:

- **Si el equipo ha cumplido las expectativas.**
- **Y si el equipo ha entendido al cliente**.

Se produce entonces una satisfacción personal al comprobar y ver funcionando la entrega.

4.3.1.4 Reunión de Retrospectiva (Sprint Retrospective Meeting)

En esta reunión, el equipo debatirá temas relacionados con el Sprint recientemente finalizado y los cambios que se podrían hacer para mejorar el próximo Sprint y que sea más productivo.

Generalmente, será el ScrumMaster quién realiza esta reunión y tendrá una duración máxima de 3 horas.

Las características generales de la reunión serán:

- **Asistirán el ScrumMaster, el Equipo y el Product Owner.**
- **La reunión girará entorno a dos preguntas básicas: ¿Qué ha ido bien durante el último Sprint?, ¿Qué será mejorado para el siguiente Sprint?.**
- **El ScrumMaster toma nota delas respuestas del Equipo en un formulario.**
- **El Equipo hablará de las posibles mejoras que se puedan realizar, indicando cuáles van a tener mayor preferencia.**
- **El ScrumMaster ayudará a entrar mejores vías de apoyo, para las mejoras indicadas por el Equipo.**
- **Si hay puntos que merezca atención se añadirán para el siguiente Sprint, considerándolo como un Backlog no funcional, pero de alta prioridad**

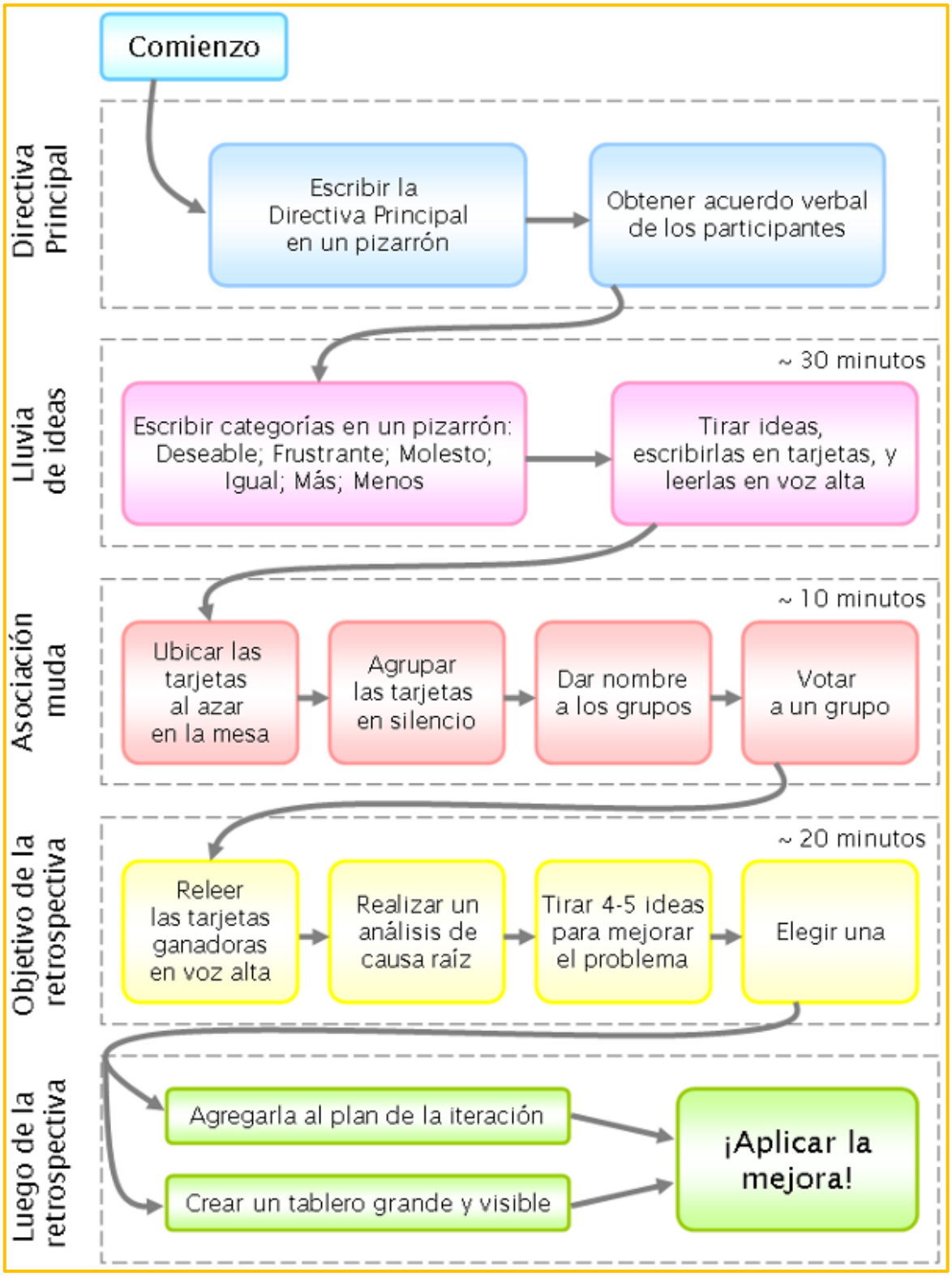
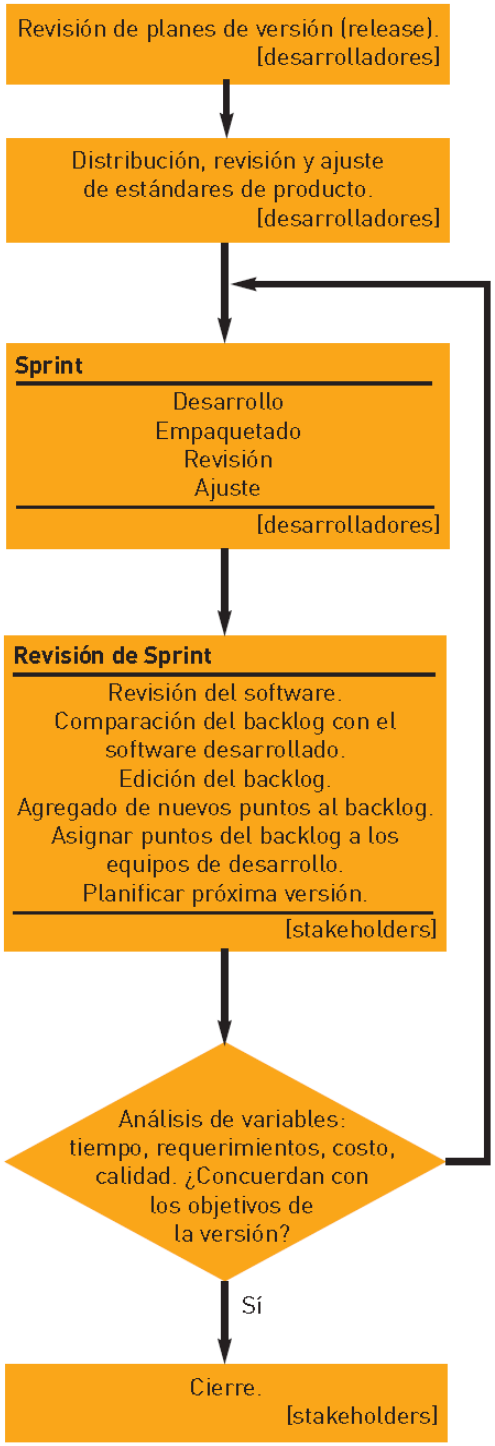


Figura.- 33: Ejemplo de Retrospectiva, de James Shore

4.4.- Diagrama detallado de las fases de Scrum.

Se ha realizado de esta manera una guía por todo el proceso de creación de un proyecto Scrum en el que se van realizando las diferentes 5 fases en forma de ciclos, hasta completar todas las tareas del Backlog.



REVISIÓN DE PLANES DE VERSIÓN:
Se revisa que hay que hacer y en que punto está la distribución actual.

SPRINT:
Es la fase de desarrollo iterativa.
Desarrollo: Análisis, implementación, testing.
Empaquetar: Generar paquetes ejecutables
Revisión: Resolución de problemas y se añaden nuevos ítems.
Ajustes: Uso de los ajustes para mejorar el producto.

SPRINT REVIEW:
Después del Sprint se hace una reunión con el ScrumMaster donde se revisa el producto del Sprint anterior y en el que se pueden añadir puntos nuevos al backlog.

Cierre:
En esta fase se encuentran las típicas actividades de fin de proyecto como, hacer una versión distribuible, testear, marketing etc….

Tabla 2: Esquema de las 5 fases de desarrollo en Scrum.

5.- BIBLIOGRAFÍA:

Libros.

Scrum Manager Gestión de Proyectos. Juan Palacio, Claudia Ruata.

Flexibilidad con Scrum. Juan Palacio.

Scrum y Xp desde las trincheras. Henrik Kniberg.

Agile Project Management with Scrum. Ren Schaner.

Páginas Web Consultadas.

http://www.scrumsense.com

http://www.proyectosagiles.org

http://www.acis.org.co

http://www.mastersoft.com.ar

http://assets.scrumfoundation.com

http://www.gestiondeproyectosit.es

http://agileee.org

http://www.pdca.es

http://msdn.microsoft.com/es-es/library/dd997578l