Actividad Académicamente Dirigida (ADD)

SCRUM

Autores:

Fco. Javier Aquino Piosa

Raúl Castilla Bravo

Daniel Pérez Rodríguez

Asignatura: Principios y Fundamentos de la Ingeniería de Software

Curso: 2º Ingeniería Informática

Año: 2017/18

Profesora: María Pilar Polo Almohano

ÍNDICE

1. INTRODUCCIÓN

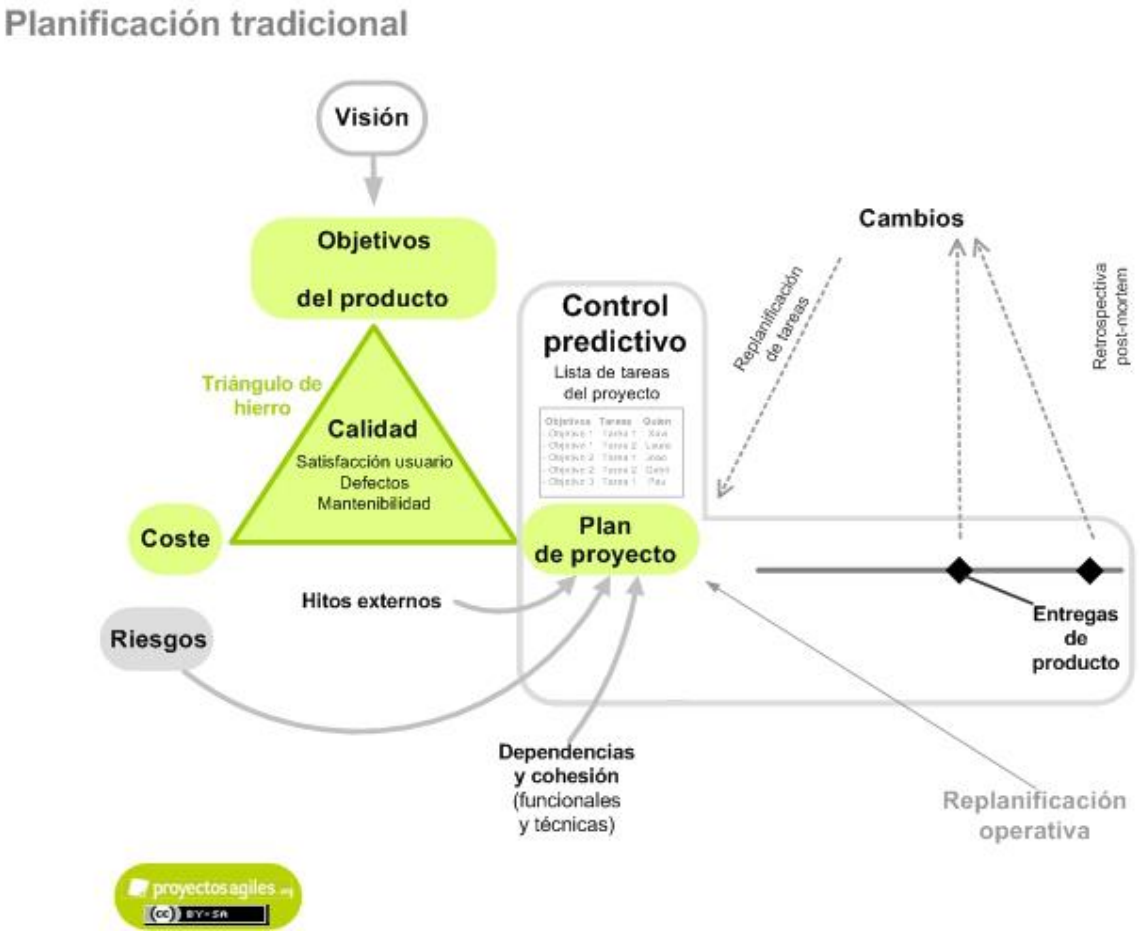
Con el nacimiento de la informática y la aparición de los proyectos software comienza una nueva etapa en las ramas tecnológicas y científicas. Sin embargo, el cambio vino acompañado de una gran cantidad de dificultades debido principalmente a los errores en el desarrollo de programas.

Para solucionar esta problemática, se diseñaron metodologías de trabajo adaptadas a los diferentes proyectos. Se puede hacer una distinción de ellas en dos corrientes: metodologías tradicionales y metodologías ágiles.

1.1 METODOLOGÍAS TRADICIONALES

Son metodologías que suelen basarse en el **encorsetado** de las **fases**, es decir, no se pasará a una nueva fase hasta que no se haya terminado la anterior. Dado al **estricto orden** de estos métodos y a la dificultad de volver a fases anteriores, las metodologías ágiles suelen ser las más admiradas por los desarrolladores.

Ejemplos de metodologías tradicionales: RUP, MSF (Microsoft Solution Framework), ICONIX.



1.2 METODOLOGÍAS ÁGILES

Metodologías basadas en el **''Divide y Vencerás''**. Se usan en ambientes cambiantes, proyectos con alto grado de dificultad y con un corto periodo de realización. Cuentan con las siguientes características:

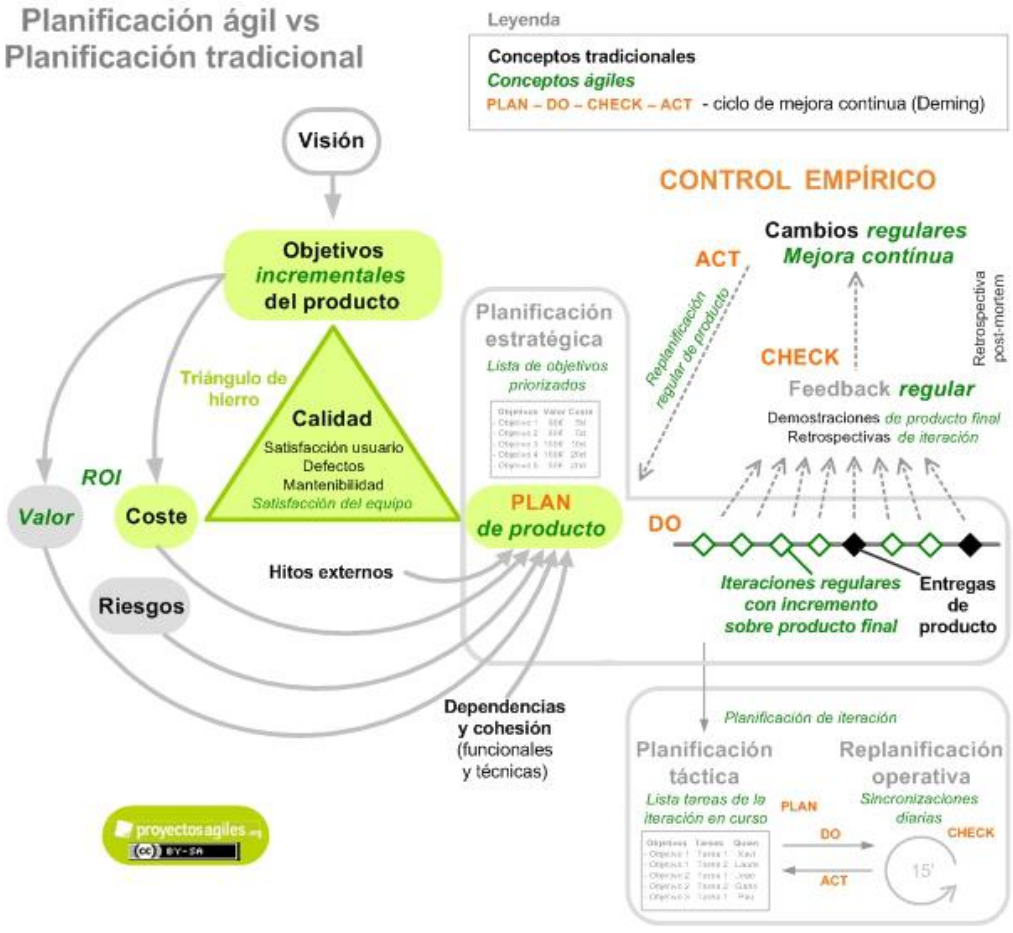
- Importancia en los individuos, ya que todos (desarrolladores y clientes) tienen un papel fundamental en el trabajo.
- Continua colaboración del cliente.
- Permiten responder a los cambios de manera flexible.

El sistema de trabajo destaca por:

- La continua comprobación del proyecto, mejora notablemente el resultado pedido por el cliente.
- El cliente se encuentra totalmente informado de lo ocurre.
- La subdivisión del proyecto produce una mejora en la localización de errores y como consecuencia una reducción de costos.

Ejemplos de metodologías ágiles: SCRUM, Programming XP, Kanban, Agile Inception.

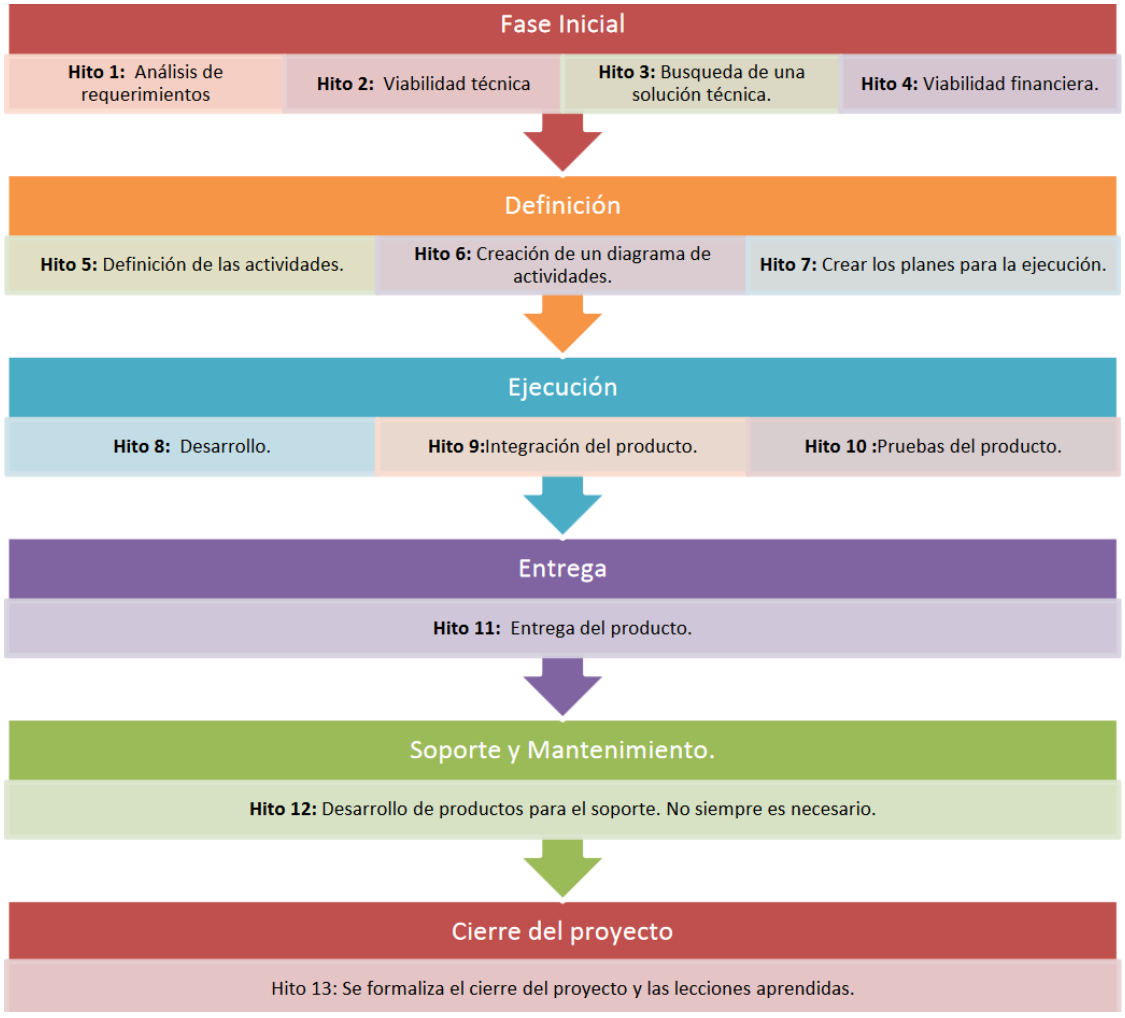
> ***"El problema no es el cambio en sí, porque el cambio va a ocurrir; el problema es la incapacidad para adaptarse al cambio cuando llega."***
>
> ***eXtreme Programming***



2. CICLO DE VIDA

De entre todas las metodologías ágiles, la más usada es el SCRUM (en torno a un 58% de los desarrollos software). El proyecto se divide en fases, las tareas se definen a nivel conceptual y se asegurarán las entregas en los plazos establecidos. Al conjunto de fases se conoce como Ciclo de Vida.

El modelo SCRUM no se realiza de forma lineal, es decir, el proyecto puede empezar con una determinada tarea y pasar a otra en cualquier momento, maximizando así la flexibilidad y productividad del equipo de desarrollo. Cada una de las fases del ciclo de vida con sus principales características se puede ver en la imagen adjunta:



Manuel Trigas Gallego – "Metodología SCRUM" (Pág. 9)

Todas las fases del ciclo de vida, van acompañadas por un cumplimiento de normas para asegurar la calidad final del producto. En España, existe la norma UNE157001 de la cual destacamos el siguiente extracto relacionado con el objetivo de la normativa:

"E*stablecer las consideraciones generales que permitan precisar las características que deben satisfacer los proyectos […] software (soporte lógico), para que sean conformes al fin a que están destinados".*

'Criterios Generales para la elaboración de proyectos'-Universidad de Oviedo.

3. ACTIVIDADES

Todas las actividades del método SCRUM se realizan en periodos de tiempo fijo llamados Sprint. Cuando se acaba un Sprint, se procede a comenzar el siguiente y así sucesivamente hasta la entrega del proyecto.

Las actividades del SCRUM son:

- First Sprint.
- Daily meeting SCRUM.
- Sprint Review.
- Release.
- Restrospective Sprint.

3.1 FIRST SPRINT

Reunión en la que el cliente presenta unos requisitos del proyecto a cumplir. Tras examinarlos, el equipo de trabajo negocia con el cliente los objetivos de mayor prioridad, asegurando la entrega de los objetivos al final del Sprint. En esta actividad:

- Se elige el entorno de desarrollo.
- Se desarrolla la lista de requisito.
- Se reparten las historias de usuario.
- Se estudia la arquitectura del programa.

3.2 DAILY MEETING SCRUM

Reunión diaria de unos 10-15 minutos donde cada miembro del equipo debe responder las siguientes preguntas:

- ¿Qué he hecho?
- ¿Qué voy a hacer?
- ¿Qué problemática tengo o voy a tener?

Si un integrante del equipo no consigue realizar sus tareas, solicitará ayuda para poder solucionarlo lo antes posible. Si no se consigue realizar en varios días, se revisará la viabilidad de la tarea.

Otra de las temáticas es decidir quién va a trabajar con quien. Una forma de trabajo común es el Pair Programming. Se hacen grupos de dos que van alternando la codificación y la revisión del código.

3.3 SPRINT REVIEW

Reunión semanal en la que el Scrum Manager verifica el trabajo llevado a cabo por el Scrum Team y, si es posible, muestra al Product Owner lo realizado hasta el momento para que éste dé a conocer su grado de satisfacción y manifieste posibles cambios.

3.4 RELEASE

Se trata de una versión del software que va a pasar a la producción. Este término está relacionado con el ''Release Candidate'', una versión previa al software oficial (no todos los proyectos cuentan con un Release Candidate).

3.5 RETROSPECTIVE SPRINT

Crítica constructiva en la que los componentes del equipo de desarrollo evalúan el Sprint Review realizado instantes antes. Los miembros del equipo escriben en post-its una crítica y se vota para elegir las más relevantes, de este modo, todo el mundo tiene la oportunidad de hablar y expresar sus ideas.

Una vez hecho esto, se sigue un formato de cinco fases:

1. Preparar el ambiente. Se recuerda que, en el Sprint anterior todos han hecho el mejor trabajo posible en función de las habilidades, recursos y situación hasta el momento. Es, en otras palabras, una forma de ''romper el hielo''.
2. Recolectar información acerca del último Sprint.
3. Generar ideas. Dichas ideas han de mejorar el próximo Sprint.
4. Decidir que se hace, para poder implementar las ideas en el siguiente Sprint.
5. Cierre. Se hace una pequeña evaluación de la propia retrospectiva.

"Un proyecto SCRUM es un Sprint de largo. Un lanzamiento de software es la suma de incrementos múltiples (y el software desarrollado previamente, si corresponde). Un proyecto de Scrum no puede fallar, solo genera un retorno de la inversión inaceptable".

Ken Schwaber

4. ARTEFACTOS

Los artefactos son herramientas para la gestión de la información acerca de los requisitos del proyecto. Los principales son:

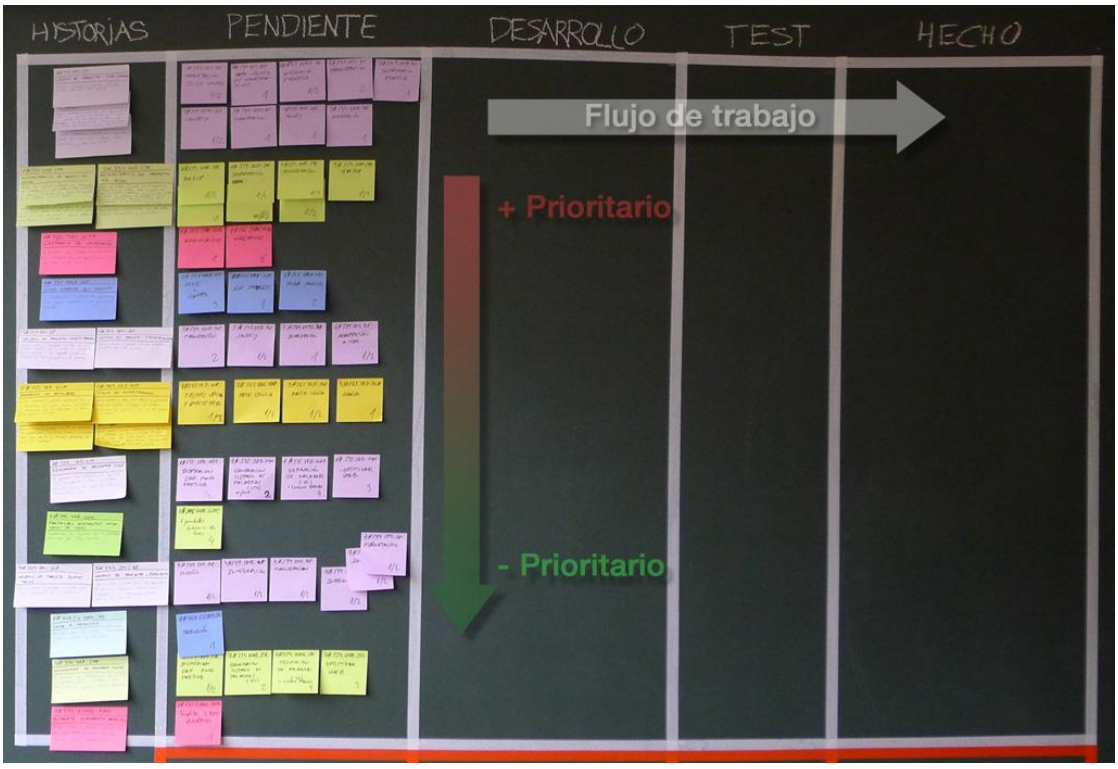
- **Product Backlog**: recoge todos los requisitos del Product Owner para el producto. Durante la fase de desarrollo esta lista de requerimientos se irá modificando, agregando o eliminando elementos acorde a las peticiones del cliente. Así, se asegura que el producto se adapta lo mejor posible al usuario.

- **Sprint Backlog**: conjunto de peticiones del Product Backlog negociadas con el Product Owner para el sprint actual.

- **Scrum Board**: pizarra o mural en formato físico o digital donde se refleja el desarrollo global del proyecto. En él se incluyen: el número de equipos y sus integrantes, los objetivos del sprint, las fechas de entrega, los requisitos completados, los requisitos en progreso… En definitiva cualquier dato relevante para el desarrollo del software.

- **Incremento**: mejora añadida al software después de un Sprint. Se suele hacer un análisis del trabajo para comprobar qué es lo que se puede mejorar para el próximo Sprint.

- **Terminado**: versión del producto final. El Product Owner y el equipo de desarrollo deberán discutir cómo será esta versión. El concepto de terminado podrá ir variando durante el desarrollo, siempre y cuando ambas partes estén de acuerdo.



5. ROLES EN SCRUM

En SCRUM tenemos dos categorías de roles:

- **Roles centrales**: imprescindibles para el proyecto y son los responsables del éxito de cada sprint. Son: Product Owner, Scrum Master y Scrum Team.

- **Roles no centrales**: no son necesariamente imprescindibles. Pero en ocasiones pueden ser decisivos (por ejemplo un sponsor). Son Stakeholder, vendedores, Scrum Guiadance Body.

5.1 PRODUCT OWNER

El Product Owner es el **dueño del producto** y como tal, es el encargado de **definir los requerimientos** del software (backlog). Este rol no pertenece a un comité sino a una única persona que, además, participa de forma activa en el desarrollo del proyecto. El Product Owner tiene una visión general del producto que va desde la conceptualización hasta la puesta en marcha del mismo.

En cada sprint se ocupa de decidir qué se quiere conseguir, así que, desde el punto de vista de los desarrolladores, el Product Owner es la **principal fuente de información.** Ahora bien, las decisiones del Product Owner están apoyadas y sustentadas en la opinión y el feedback de los usuarios.

Por ser el sustentador de la información y además la voz del cliente, es necesario que el dueño del producto sea capaz de expresar sus ideas con claridad.

Asimismo, debe estar completamente disponible para el equipo por si surgiesen dudas en alguna fase concreta. Sin embargo, debe ser consecuente con las tareas que se les encargan a los desarrolladores.

Recordemos que todo tiene un fin económico y quizás un complemento añadido, puede disparar el precio del software o puede complicar enormemente su desarrollo. Por ello, es indispensable que posea un excelente conocimiento del negocio al que irá destinado el programa.

Puntos clave del Product Owner

- Dueño del producto.
- Representante del cliente e interesados en el producto.
- Encargado de establecer requisitos.

5.2 SCRUM MASTER

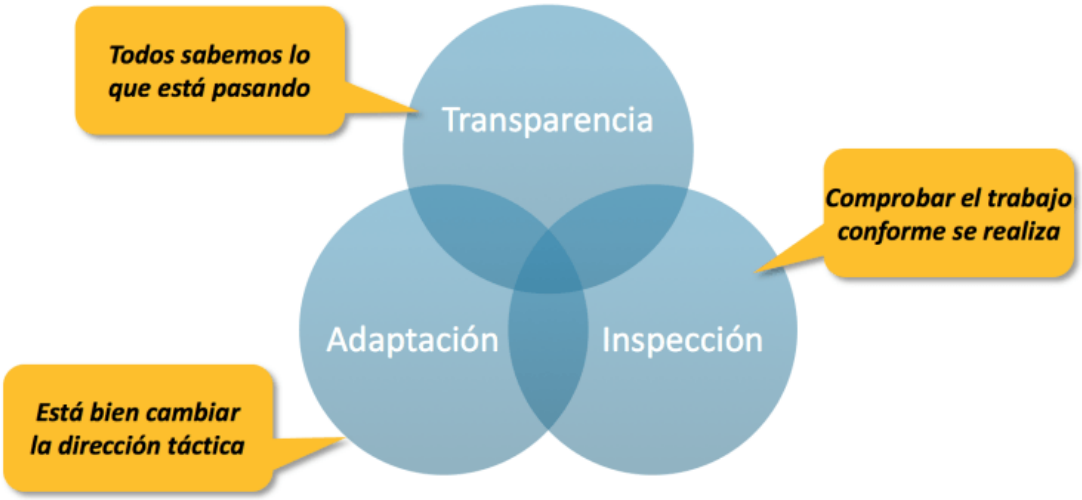
El Scrum Master es el encargado de supervisar que **se aplica el Scrum correctamente**, cada persona juega un papel en el desarrollo y tiene unas competencias definidas que deben quedar claras. Por ejemplo, el Product Owner no puede actuar como representante del equipo de trabajo ni el equipo de trabajo puede añadir complementos al software que no haya especificado el Product Owner.

En otras palabras, el Scrum Master actúa como un coordinador. Como consecuencia, debe ser una persona con un amplio conocimiento en esta metodología de trabajo, capaz de explicarla, enseñarla y aplicarla. Cabe aclarar que no es un jefe sino un instructor y como tal, **motiva el trabajo en equipo** y trata de **evitar o resolver los conflictos** que puedan aparecer entre trabajadores.

En general, es un **líder al servicio de todos**. En primer lugar, ayuda al Product Owner a entender la planificación del proyecto. Por otra parte, elimina impedimentos para el progreso y desarrollo de las tareas y crea un **ambiente laboral adecuado**. Por último, sirve a la Organización buscando posibles cambios para **mejorar la productividad** del Scrum Team.

Puntos clave del Scrum Master

- Coordinador del método Scrum.
- Ejerce de líder e instructor.
- Motiva el trabajo del Scrum Team.
- Busca mejoras en la productividad del equipo



5.3 SCRUM TEAM

El objetivo del Scrum Team o Equipo Técnico es la elaboración del proyecto. Son los encargados de, una vez recibido los requisitos, explicar **cómo cumplirlos y hacerlos realidad**. De acuerdo a las metodologías ágiles, es necesario que los desarrolladores realicen su tarea en el **menor tiempo posible y con la máxima calidad**, asegurando un incremento del producto en cada sprint.

Se trata de un grupo de trabajo **auto-gestionado y multifuncional** donde **no existen jerarquías** internas. Cada miembro forma parte de un todo. En este rol se encuentran expertos y profesionales capaces de desempeñar las tareas pertinentes.

Es importante que cada uno de los integrantes no solo tenga una buena aptitud para el trabajo en equipo sino además una verdadera **vocación por todo el proyecto**, ya que los errores y las dificultades se presentarán de forma inevitable y deben ser capaces de superar los baches y contratiempos.

Puntos clave del Scrum Team

- Encargados de hacer realidad las peticiones del proyecto.
- Trabajo rápido y de calidad.
- Auto-gestionado y multifuncional.
- Sin jerarquías.
- Con vocación.

5.4 STAKEHOLDER

Los Stakeholder son todos aquellos con intereses en el proyecto. Participan aportando ideas y opiniones pero sin ocupar el papel del Product Owner. En cada sprint, los incrementos del software pueden afectar a muchos departamentos de una empresa, así que todos pueden aportar información. Dentro de este grupo, encontramos también a los clientes y los patrocinadores (sponsor).

5.5 CLIENTE

Son los usuarios finales del software, quienes van a pagar por él, y proporcionan un feedback muy relevante a la hora de decidir cuáles van a ser los nuevos incrementos del programa. Los desarrolladores deben conocer a quiénes está destinado el producto y adaptarse a sus necesidades. Recordemos que un producto de mala calidad supone un soporte al cliente mayor y puede llegar a ser demasiado costoso a la larga.

5.6 PATROCINADOR (SPONSOR)

Es una figura que aparece ocasionalmente y que aporta ayuda económica a cambio de incluir publicidad en el proyecto. Los patrocinadores se interesan principalmente en empresas de prestigio con mucho alcance para que sus anuncios sean notorios. Como consecuencia, si la empresa entrega software de mala calidad, perderá fama y el sponsor no querrá continuar con su inversión. Por ello, es también uno de los principales interesados en el éxito del producto.

5.7 SCRUM GUIDANCE BODY

El Scrum Guidance Body es un conjunto de documentos o expertos relacionados con los objetivos de calidad, las regulaciones gubernamentales y otros aspectos de nivel organizativo. No toman decisiones relacionadas con el proyecto, son un órgano consultivo que guía al personal para conseguir una planificación adecuada.

6. USER HISTORY

Durante el desarrollo del software, es indispensable que el equipo de trabajo entienda qué es lo que hay que hacer en cada momento y para ello se usan las historias de usuario (User History).

Esta práctica es una forma sencilla de explicar a los programadores las funciones que necesitan implementar en el producto. Se utilizan tarjetas para resaltar los puntos más importantes que se quieren tratar. No se usará un lenguaje técnico, deben ser fáciles de comprender y deben quedar claro los resultados esperados.

Las historias presentan la siguiente estructura:

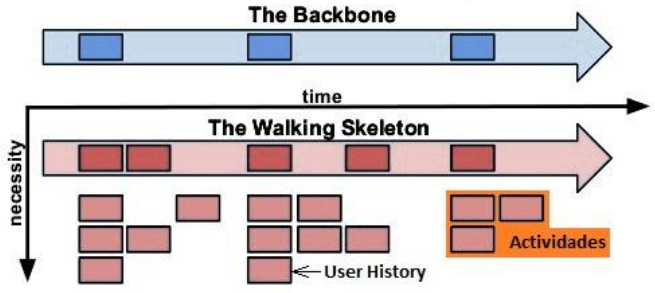
Como <quién> Quiero <qué> Para <objetivo>

El redactor de la historia debe especificar quién será el usuario que utilizará la funcionalidad a implementar, para seleccionar la mejor forma de contentar al target (grupo de personas al que va destinada la función). Se deberá concretar en pocas líneas qué se va a añadir y el propósito de la funcionalidad. Si aun así, las ideas no quedan claras, se redacta un ejemplo práctico para poner en situación al Scrum Team sobre la función.

Para exponer y organizar la información se utilizan herramientas como el Visual Story Mapping, creada por Jeff Patton. Se elabora sobre el Scrum Board, donde las historias se visualizarán durante todo el desarrollo del proyecto y se ordenarán de forma bidimensional, es decir, el eje vertical está destinado a representar la importancia de la historia y el horizontal, a mostrar las historias de usuario de forma secuencial. Sin embargo, las historias son solo una parte del Story Mapping.

El Visual Story Mapping se descompone en 4 elementos:

- **Backcbone:** En él, se expanden y se muestran los conceptos e ideas sobre cómo se estructura el proyecto
- **The Walking Skeleton:** Se le llama a la forma de estructurar la información de forma que se pueda contemplar la relevancia de una función a implementar, así como su desarrollo en el tiempo.
- **User Storys:** Son tarjetas donde se recoge información de los usuarios a los que va dirigido el producto.
- **Actividades:** Se refiere a la contemplación de todas las posibles tareas de un usuario cuando se enfrenta a la utilización de un software en cuestión.



Esta forma de planificación es útil para separar el trabajo realizado de los distintos Sprint en secciones (Release), y poder así contemplar que funciones son más necesarias de cara a próximas versiones.

7. PRUEBAS DE ACEPTACIÓN

Todo el trabajo que se desarrolla debe ser testeado y verificado mediante pruebas de aceptación. Estas pruebas les darán a los usuarios una visión clara del trabajo realizado hasta el momento y pueden manifestar de distinta forma:

- **Pruebas de aceptación de usuarios:** Verifican el correcto funcionamiento del sofware desde el punto de vista del usuario final.
- **Pruebas operativas:** Se comprueba las funciones administrativas de la aplicación. Tales como: Gestión de usuarios, backups, mantenimiento, etc.
- **Pruebas de aceptación contractual y normativa:** Se basan en el testeo las funciones acordadas entre el Product Backlog y el Scrum Team al principio del desarrollo del proyecto.
- **Pruebas Alfa y Beta:** Para captar la atención de usuarios potenciales, se elaboran una serie de versiones del producto para su testeo. Cuando hablamos de pruebas alfas, su desarrollo transcurre en el zona de trabajo. Por otro lado, las pruebas betas las realizan los clientes en sus propios lugares de trabajo.

Gracias a las pruebas beta se puede obtener mucha más información y opiniones, permitiendo mostrar las flaquezas y virtudes del software en cuestión.

8. ESTIMACIÓN

Cuando hablamos de estimaciones hemos de ser conscientes de que no vamos a obtener un resultado preciso del que podamos tener completa confianza. Las estimaciones están condicionadas por una gran cantidad de variables que en muchas ocasiones son aproximaciones de los valores reales. Si bien, a pesar de esta incertidumbre, los errores que se comenten pueden ser asumibles siempre que los estimadores que se utilicen sean apropiados.

A la hora de hacer un estudio para un proyecto, se tienen en cuenta estimaciones de recursos y costes que podrán ser más o menos precisas en función del tamaño del proyecto, la experiencia previa en otros proyectos similares, la estabilidad de los requisitos…

8.1 ESTIMACIÓN DE RECURSOS

La estimación de recursos recoge información sobre el personal de trabajo, los recursos software, las herramientas hardware disponible y el capital a invertir entre otros. Además de la fecha en la que se requiere cada uno de ellos y el tiempo que se mantendrá ocupado dicho recurso.

8.2 ESTIMACIÓN DE COSTES

La estimación de costes se centra especialmente en la relación entre la cantidad de personal y los meses necesarios para llevar a cabo el proyecto. El principal motivo de error en estas estimaciones reside en los trabajadores, debido a que influyen aspectos subjetivos como la motivación y la experiencia.

8.3 PUNTOS DE HISTORIA

En las metodologías ágiles se suele utilizar una unidad de medida diferente al tiempo a la hora de expresar la duración de una tarea, los puntos de historia. A resumidas cuentas, una tarea llevará más tiempo realizarla cuantos más puntos de historia tenga asignados. La pregunta recurrente es: ¿bajo qué criterio se asignan los puntos de historia?

Para responder a esto hay que basarse en: complejidad, esfuerzo e incertidumbre (que no esté del todo definido lo que hay que hacer). Hay que tener en cuenta que los puntos de historia están sujetos a la subjetividad, así que para ser más precisos, se suelen tomar como referencia tareas ya completadas y se realiza una estimación relativa.

8.4 VELOCIDAD

La velocidad es el número de puntos de historia que han podido completarse al finalizar un sprint. Es una medida útil a la hora de decidir qué tareas se van a llevar a cabo en el siguiente sprint. Si de puntos de las tareas elegidas es muy superior a la velocidad, estaremos sobrecargando al equipo de trabajo. Por el contrario, si la suma de puntos es menor, el equipo no estará siendo todo lo productivo que podría. Es necesario encontrar un término medio para asegurar la rapidez y la calidad.

8.5 PUNTOS HISTORIA VS DÍAS IDEALES

Ocasionalmente en los proyectos se utiliza como unidad de medida los días ideales (jornadas laborales sin pausas y sin contratiempos). Sin embargo, las estimaciones suelen ser excesivamente optimistas, ya que no tiene en cuenta posibles bajas de personal, errores y otros factores poco previsibles.

Para evitar malas estimaciones, se usan los puntos de historia, que examinan el volumen de trabajo y no el tiempo ideal para finalizar un trabajo. A medida que un equipo completa tareas y proyectos, es más consciente de sus capacidades y las estimaciones con puntos de historia son más exactas (estimación relativa).

8.6 PLANNING POKER

El Planning Poker es una técnica para estimar el esfuerzo de cada una de las historias de usuario donde participan todos los miembros del equipo, desde los más experimentados hasta los más nóveles. Además, sirve para que los más novatos cojan experiencia en las estimaciones y las tareas queden claras para el equipo.

El procedimiento es el siguiente: a cada uno de los participantes se le entregan unas tarjetas con unas puntuaciones (normalmente se escogen valores numéricos pequeños porque se da menos error). A continuación, se van presentando y explicando las tareas que se van a realizar en el sprint y cada miembro coloca una de las tarjetas boca abajo.

Cuando todos hayan realizado la votación, se revela la puntuación que cada uno le ha dado a la tarea. Cuanto más alto sea el valor, mayor esfuerzo supone. Si se da el caso de que algunos en particular han dado una puntuación muy alta o muy baja en comparación a sus compañeros, explicaran al resto los motivos de su decisión.

Después del debate, la tarea recibe una estimación de esfuerzo (puntos de historia) equivalente a la media de todas las estimaciones que se han presentado.

8.7 AFFINITY ESTIMATION

Affinity Estimation sigue una filosofía de estimación grupal muy similar al Planning Poker pero con ciertas diferencias en el desarrollo de la actividad. Las tareas se escriben en tarjetas y se reúnen todas en una pila.

Cada participante, por turnos, saca una tarjeta de la pila y la coloca sobre la mesa de manera que las que requieren mayor esfuerzo se sitúan a la derecha. Pasados varios turnos, se podrán, no solo sacar nuevas tareas de la pila, sino también cambiar el orden de las tarjetas que ya hayan salido. Al terminar, se agrupan las tareas más livianas en historias de usuario.

BIBLIOGRAFÍA

1. Introducción. 1.1 Metodologías tradicionales. 1.2 Metodologías ágiles.	https://proyectosagiles.org/2010/12/15/planificacion-agil-vs-planificacion-tradicional https://www.iebschool.com/blog/que-es-agile-project-management-ventajas-de-ser-el-mas-rapido-y-agil-agile-scrum/
2. Ciclo de vida.	http://anaydisesistem.blogspot.com.es/2011/04/modelo-de-ciclo-de-vida-scrum.html
3. Actividades. 3.1 First Sprint. 3.2 Daily Meeting Scrum. 3.3 Sprint Review. 3.4 Release. 3.5 Retrospective Sprint.	https://jeronimopalacios.com/scrum https://proyectosagiles.org/como-funciona-scrum/ https://proyectosagiles.org/que-es-scrum
4. Artefactos.	https://cristinaramosvega.com/z-los-artefactos-scrum/ https://jeronimopalacios.com/2017/02/manteniendo-sprint-backlog-fisico-digital/ https://www.obs-edu.com/es/blog-project-management/scrum/principales-utilidades-y-beneficios-del-scrum-board
5. Roles en Scrum. 5.1 Product Owner. 5.2 Scrum Master. 5.3 Scrum Team. 5.4 Stakeholder. 5.5 Cliente 5.6 Patrocinador (Sponsor). 5.7 Scrum Guidance Body.	https://platzi.com/blog/que-es-scrum-y-los-roles-en-scrum/ https://cristinaramosvega.com/los-roles-scrum/ https://desarrolloweb.com/articulos/roles-scrum.html http://blog.scrumstudy.com/what-are-the-important-roles-in-scrum-project/
6. User History.	https://www.tenstep.ec/portal/articulos-boletin-tenstep/253-scrum-como-escribir-historias-de-usuarios-sin-morir-en-el-intento http://managementplaza.es/blog/historias-de-usuario-en-scrum-agile/ http://www.caminoagil.com/2013/02/visual-story-mapping-aplicado.html http://www.agileninja.org/visual-story-mapping/
7. Pruebas de aceptación.	http://scrum-qa.blogspot.com.es/2013/03/pruebas-de-aceptacion-k2-entender.html
8. Estimación. 8.1 Estimación de recursos. 8.2 Estimación de costes.	https://es.wikiversity.org/wiki/Estimaci%C3%B3n_de_proyectos_software
8.3 Puntos de historia. 8.4 Velocidad. 8.5 Puntos de historia vs días ideales.	https://samuelcasanova.com/2017/04/puntos-historia-en-agile/
8.6 Planning poker.	http://albertoromeu.com/scrum-planning-poker/
8.7 Affinity Estimation.	https://blog.versionone.com/why-affinity-estimation/