Universidad Nacional

De Educación a Distancia

Máster Universitario de Investigación en Ingeniería de Software y Sistemas Informáticos

Código (31105151) Trabajo Fin de Máster

Autor

Youssef KESBAOUI FELJY

Directora del Trabajo

Dra. MARIA MAGDALENA ARCILLA COBIAN

Curso 2018-2019 – septiembre 2019

Máster Universitario de Investigación en Ingeniería de Software y Sistemas Informáticos

Código (31105151) Trabajo Fin de Máster

Trabajo de investigación sobre las metodologías ágiles y su impacto en la mejora de los procesos software.

Autor

Youssef KESBAOUI FELJY

Directora del Trabajo

Dra. MARIA MAGDALENA ARCILLA COBIAN

Las metodologías ágiles y su impacto en la mejora de los procesos software

DECLARACIÓN JURADA DE AUTORÍA DEL TRABAJO CIENTÍFICO, PARA LA DEFENSA DEL TRABAJO FIN DE MÁSTER

Fecha: 18/07/2019

Quién suscribe:

Autor(a): Youssef Kesbaoui Feljy D.N.I/N.I.E/Pasaporte.: 03239462

Hace constar que es el autor(a) del trabajo:

En tal sentido, manifiesto la originalidad de la conceptualización del trabajo, interpretación de datos y la elaboración de las conclusiones, dejando establecido que aquellos aportes intelectuales de otros autores se han referenciado debidamente en el texto de dicho trabajo.

DECLARACIÓN:

- ✓ Garantizo que el trabajo que remito es un documento original y no ha sido publicado, total ni parcialmente por otros autores, en soporte papel ni en formato digital.
- ✓ Certifico que he contribuido directamente al contenido intelectual de este manuscrito, a la génesis y análisis de sus datos, por lo cual estoy en condiciones de hacerme públicamente responsable de él.
- ✓ No he incurrido en fraude científico, plagio o vicios de autoría; en caso contrario, aceptaré las medidas disciplinarias sancionadoras que correspondan.

Fdo.

IMPRESO TFDM05_AUTORPBL
AUTORIZACIÓN DE PUBLICACIÓN
CON FINES ACADÉMICOS

UNED ETS de Ingeniería Informática

Impreso TFdM05_AutorPbl. Autorización de publicación y difusión del TFM para fines académicos

Autorización

Autorizo/amos a la Universidad Nacional de Educación a Distancia a difundir y utilizar, con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado

Firma del/los Autor/es

Youssef Kesbaoui Feljy

Las metodologías ágiles y su impacto en la mejora de los procesos software

RESUMEN

La gestión de los procesos dentro de los proyectos de la ingeniería del software es una parte importante en una cadena que se desarrolla correctamente hace que un proyecto sea un éxito.

Dentro de la gestión de procesos, un aspecto muy importante es la calidad de los desarrollos software, en los últimos años, este aspecto se ha convertido en un punto crítico y se han ido investigando y aplicando Metodologías para llevar a cabo desarrollos certificados con una calidad según los distintos estándares.

Las metodologías llamadas ágiles son una máxima actual en el desarrollo software de mayor extensión dentro del mercado, hay una variedad de estas y se aplican de distintas maneras, en todas las fases de desarrollo desde el diseño hasta la entrega final.

Su objetivo es implicar a todos los componentes humanos en la concepción de un proyecto y llevarlo a cabo con un acompañamiento en todas sus fases. Para ello se usan herramientas y programas: de desarrollo propio o de terceros para ayudar a controlar esa línea de seguimiento y los parámetros y métricas del proceso.

Este trabajo tiene como objetivo exponer esas metodologías haciendo una comparativa entre ellas y con un caso práctico de un pequeño desarrollo

sin metodología aplicada realiza un estudio sobre el efecto de aplicarle una metodología ágil u otra y valorar todas las mejoras que se harían en consecuencia.

LISTA DE PALABRAS CLAVE

Metodologías, Ágil, Scrum, Scrum Master, Scrum Team, Software, Extreme Programming, Kanban, Jira, Sprint, Producto, Entrega, Equipo/Equipos, Mejora, Calidad, Product Owner, Stackholder, Sprint, User story, poller, Trello, MaestroToSvn, SVN, Backlog, Incremento, Agile Coach.

INDICE

LISTA DE FIGURAS

1 INTRODUCCION

Las metodologías ágiles surgieron como estudios en la década de los 70 del siglo pasado en una reacción contra de las metodologías tradicionales del desarrollo de proyectos, en los años 1970, Dr. William Royce abre la discusión sobre la posibilidad de que un proyecto software puede ser desarrollado como un producto en una cadena de montaje. Todo se resume en que una etapa no se empieza a desarrollar mientras la anterior no se haya acabado primero, empezando por la concepción hasta la última etapa.

Una formación sobre todo el proceso de los desarrolladores es primordial para que se lleven bien a cabo, todas estas etapas. [1]

En la última década, las metodologías ágiles se han abierto un camino en la gestión de los procesos dentro de la industria de la ingeniería de software,

Convirtiéndose en una gran medida en un actor muy importante para el desarrollo de cualquier actividad que forme parte del ciclo de vida del software.

Estas metodologías cubren todos los aspectos de un proceso software, desde definición de las necesidades hasta la entrega al cliente pasando por la fase de testing, gestionando, por lo tanto, los inputs y outputs de cada fase de un proyecto.

¿Pero porque aplicar las metodologías ágiles dentro de una organización?

En una empresa de consultoría o de desarrollo informático, los procesos se valoran por su eficiencia y su agilidad, eso se traduce en mejores costes y plazos, por consiguiente, clientes satisfechos con las entregas realizadas y fidelizados para que adjudiquen nuevos y futuros trabajos.

Por consecuente, una empresa que busque aplicar las metodologías ágiles en sus procesos de desarrollo, debe tener claro cuál de ella es más adecuada a su cultura empresarial y a su manera de hacer las cosas, también ver si el coste de aplicarlas puede ser una inversión fiable y que tendría beneficios a medio plazo.

Entonces, para conseguir esos beneficios hay que buscar en una metodología ágil los siguientes motivos:

- **Son de menor coste**

Una consecuencia de su uso dentro de una organización es que en un proyecto se emplean menos personas dentro del mismo ya que los tiempos de entregas se reducen de manera significativa y por lo tanto no se dilapidan recursos.

- **Mejor gestión de la configuración o cambio**

Aportan flexibilidad a los procesos, el trabajo se divide en iteraciones, Sprint, stories...etc... La supervisión de las tareas es más centrada en cada una de ellas, se adaptan a los cambios en las soluciones sin cambios en el rumbo del procedimiento.

- **Incrementa la motivación de los equipos**

Implementan un modelo de autogestión en los equipos, sobrepasan los modelos clásicos de jefe-subordinados, los equipos se van gestionando según las necesidades de la iteración o el proyecto. Esto hace que su auto estima dentro de su trabajo se hace más patente y con una incidencia positiva en el proyecto.

- **Compromiso**

La motivación provocada por la integración dentro de un equipo autogestionado hace que un miembro se sienta como una parte esencial en el desarrollo del proyecto, lo que implica un gran compromiso por su parte para llevarlo a buen puerto.

- **Acercamiento de los resultados a las necesidades del cliente**

La implicación continua del cliente dentro del equipo ágil como una referencia funcional o de producto, sus comentarios, indicaciones y observaciones hace que errores de desarrollo se rectifiquen en etapas iniciales y enderezar el desarrollo del producto para que se acerque le máximo posible de los resultados deseados.

> *"Las metodologías ágiles son más adaptativas que predictivas, Los métodos de ingeniería tienden a intentar planificar una gran parte del proceso software con mucho detalle para un gran periodo de tiempo, esto funciona muy bien hasta que las cosas cambian. Entonces por naturaleza resisten ese cambio. Las metodologías ágiles, de cualquier modo, acepta el cambio, Intentan ser procesos que se adaptan al cambio, hasta tal punto de cambiarles a ellos también.*
> *Los métodos ágiles son orientadas a las personas más que a los procesos. El objetivo de los métodos de ingeniería es definir un proceso que funcionará bien independientemente de quien lo esté usando. Los métodos ágiles afirman que ningún proceso compensará las habilidades del equipo de desarrollo, sino su rol es apoyar el equipo de desarrollo en su trabajo."* ***Martin Fowler - The New Methodology***

Más que la adaptabilidad al cambio, la incorporación de las metodologías orienta una organización hacia la aceptación de ese cambio, incluyendo procedimientos concretos, llegando incluso a veces, a reformular procesos ya implantados desde siempre y que se hayan quedado obsoleto.

Un aspecto importante es el cambio no solo es un paso de un procedimiento a otro mejorado, sino es una reformulación drástica: Se introducen conceptos nuevos y perfiles profesionales nuevos, lo equipos son pequeños y autogestionados, herramientas que se utilizan dependiendo de cada fase; seguimiento, desarrollo, integración, entrega y despliegue aplicativo…etc.
En cuanto a las herramientas utilizadas para llevar a cabo estas aplicaciones, existe un enorme abanico de aplicaciones gratuitas o de pago que apoyan esa transformación para que sea lo más transparente y menos cara para las empresas que deciden utilizarlas.
Hay herramientas de gestión del tiempo, otras de seguimiento de tareas y también de control económico de cada iteración, también existen herramientas de integración continua que controlan la parte de los tests de cada código subido al final de cada iteración, verificando por lo tanto el nivel de calidad es adecuado a lo exigido por parte del cliente.

La transformación al ágil en la que se han embarcado muchas empresas del mercado tecnológico en los últimos años ha tenido varias consecuencias:

- Hacer constar la obsolescencia de las antiguas metodologías clásicas.
- La creación de nuevos procedimientos de trabajo implicando la creación de herramientas y estilos de desarrollo nuevos.
- Amplias comunidades de divulgación del conocimiento sobre agilidad.
- Creación de nuevas profesiones dentro de la informática: Scrum Máster, Product Owner, Ágil Coach…etc.
- Certificaciones oficiales expedidas por organismos emanantes de un conjunto de gurús de renombre mundial de la agilidad.

Algunos ejemplos de comunidades ágiles:

- Scrum: www.scrumalliance.org
- Kanban: www.agilealliance.org
- eXtreme Programming: www.agilealliance.org/the-extreme-programming-tribe/

En este punto, hay que plantear la usabilidad de las metodologías ágiles y su posibilidad de aplicación dentro de una organización, lo que aportarían como

valor añadido y lo que no, realmente las metodologías ágiles no tienen lados negativos sino la posibilidad de ser aplicables o no.

Una vez que se estudia la naturaleza de los procesos o procedimientos dentro de la organización, se decide qué metodología aplicar y cómo hacerlo. Esto es, juntar todos los elementos o personas implicadas (o por su responsabilidad o puestos) para discutir la mejor metodología a elegir y la manera de aplicarla.

Existen casos en los que no son aplicables las metodologías ágiles, como un caso de tentativa de implantación dentro de un banco nacional, donde el Scrum Master decidió que no se haga ninguna documentación, y que el conocimiento fuese transmitido entre los empleados de manera experimental y oral, es decir: el conocimiento se va adquiriendo de los compañeros de trabajo y de la propia experiencia del miembro del equipo. Resultó un fracaso ya que, en un entorno bancario, hay una serie de leyes, normativas y circulares que hay que documentar para su divulgación, por lo que la agilidad no se podía aplicar.

No obstante, viéndolo desde otro ángulo, en este caso no tiene sentido aplicar una metodología ágil para resolver los problemas de los procesos, ya que en un ámbito normativo y sujeto a leyes, no resultarían de gran ayuda, y puede que lo contrario, ralenticen más los procedimientos.

En otro entorno, dónde las normativas son más superficiales o incluso inexistentes, las metodologías ágiles, pueden y deben ser aplicadas en mayor o menor medida, dependiendo de los casos y de los deseos de los clientes. La problemática que se plantea es sí se resolverían los problemas de ineficiencia y desorden en los desarrollos en un tiempo que permita amortizar lo que se ha invertido en implantar dicha metodología en un desarrollo, o simplemente alargaría más los problemas que acompañan el software desarrollado en todos sus ciclos.

En este trabajo se va a entrar dentro de las metodologías ágiles para resolver un problema de un desarrollo sobre el cual no se han aplicado, este proyecto se enmarca en la mejor de los procesos software para adaptar los procedimientos de las organizaciones a los avances actuales de la informática en cuanto a metodologías y procesos se refiere.

1.1 OBJETIVOS DEL TFM

En este Trabajo fin de Máster (TFM) se orientará más a los proyectos software, aunque la metodología ágil puede extrapolarse a todos los sectores industriales y tecnológicos que precisen de un ciclo de vida resumido en: definición - planificación - ejecución - testeo – entrega.

El objetivo principal del TFM es mostrar con una problemática de un desarrollo que se lleva a cabo sin aplicar ninguna metodología ágil, el contraste en términos de coste de proceso con la aplicación de alguna de las metodologías existentes y reducir esos costes.

También se estudiarán las metodologías más importantes existentes y utilizadas en el mercado:

- Su definición y detalle de sus estructuras.
- Sus cualidades y sus defectos.
- Diferencias entre unas y otras

Se presentarán los elementos importantes de los que se compone, de forma genérica, una metodología ágil, así como sus marcos de desarrollo.

El planteamiento será él de una problemática de un desarrollo software realizado en una primera fase sin seguimiento de ninguna metodología ágil, se le aplicará en una segunda fase una metodología elegida entre las que se van a detallar en los capítulos preliminares comparando las métricas entre las dos fases llegando a ver la eficiencia de la metodología aplicada.

Entre las razones para investigar las metodologías ágiles, tanto en el entorno empresarial, como en este caso en el entorno académico:

- Su capacidad de adaptabilidad a los cambios
- Las herramientas que aportan para llevar a cabo todas las fases del desarrollo software.
- La capacidad de respuesta de una empresa u organización frente a potenciales solicitudes de clientes.
- El ahorro económico tanto en tiempo y en dinero además en acortar los procesos clásicos que el empleo de las metodologías supone.

Ofrecen apoyo a las organizaciones dentro de un sector que está en constante cambio tanto en tecnologías que en procedimientos dejando atrás todas las metodologías clásicas.

Los beneficios que se consiguen son a medio-largo plazo, debido a que las metodologías ágiles adecuan los procesos a una serie de fases desde la planificación hasta la entrega final al cliente de cada incremento al finalizar una iteración.

Estas fases son:

- Concretar las funcionalidades y requisitos
- Planificación
- Desarrollo
- Revisión de código
- Tests
- Entrega (empaquetado y release o versión)
- Monitorización

En el caso particular de este TFM, la problemática que se plantea en el capítulo 2, el objetivo a alcanzar es mejorar los plazos de todo el procedimiento, la calidad de lo entregado además de asegurar un menor número de incidencias en producción cuando se ha pasado por una fase de tests después de la finalización de la codificación y de la revisión.

1.2 ESTRUCTURA DEL TFM

Este TFM se estructurará en cinco capítulos:

Capitulo primero: Introducción

Introducción a lo que son las metodologías ágiles y su incursión en el mercado de las consultorías informáticas, dando una visión global de lo que se a tratar en este TFM.

Capitulo segundo: Planteamiento del problema

Descripción de un caso de un desarrollo pequeño donde no se haya aplicado ninguna metodología y describir los problemas que surgieron en las fases previas y posteriores a la fase de implementación.

Capitulo tercero: Estado de la cuestión

Estudio de las metodologías existentes y más importantes del mercado del software, se detallarán sus puntos fuertes y sus puntos mejorables.

Se evaluarán las aportaciones que hacen a un proceso software.

Capitulo cuarto: Resolución

En este capítulo, se aplicará una metodología ágil, exponiendo los resultados obtenidos para hacer constar que se han mejorado respecto a lo planteado en el capítulo de la problemática.

Capitulo Quinto: Conclusiones y líneas futuras

Conclusiones y opinión sobre el tema del trabajo y sobre la preferencia de la utilización de una metodología sobre otra.

Líneas futuras sobre el tema del trabajo para desarrollarlo con más profundidad y poder ser capaces de especificar una metodología que se inspire las demás y fuera influenciada por los avances en el desarrollo de procesos del momento.

2 PLANTEAMIENTO DEL PROBLEMA

2.1 INTRODUCCIÓN

Presentación de un caso de un desarrollo que se llevó a cabo siguiendo una metodología clásica de cascada de agua, donde una fase no empieza hasta que termine la anterior.

El desarrollo en cuestión trata de crear una aplicación web tipo Servlet que se encarga de manejar ficheros creados, modificados o borrados en un directorio local frente a un repositorio fuente de SVN (https://subversion.apache.org/), donde se almacenan e historian todas las modificaciones sufridas por un fichero de código fuente.

Se usará una implementación de la librería SVN Kit(https://svnkit.com/) y con Java 6: https://svnkit.com/

El funcionamiento es el siguiente:

1- En un directorio se almacenan los ficheros fuentes según su nomenclatura.
2- El desarrollador puede Modificar un fichero.
3- El desarrollador puede crear un nuevo fichero dentro del directorio principal.
4- El desarrollador puede suprimir un fichero del directorio principal.
5- Desarrollador crea/modifica/borra algún fichero en el directorio principal, el cambio se replica en el directorio remoto (repositorio SVN)

6- Cualquier modificación sobre cualquier fichero mantiene su histórico de cambios con las credenciales del usuario quien la realizó.

La mecánica para la especificación fue como sigue:

- La reunión inicial fue para explicar a grandes rasgos lo que se espera del sistema, se especificaron muy pocos aspectos detallados de la aplicación.
  Al final de la reunión se realizó un esquema en una pizarra.
- Al terminar de recoger los requerimientos el cliente decide que el desarrollador haga las especificaciones funcionales a partir de dichos esquemas.

Esta fase se llevó a cabo con un desplazamiento a casa del cliente por **3 días.**

Especificación funcional y técnica:

- En esta fase, se realiza la especificación funcional, se le envía al cliente y se espera su validación para seguir con el diseño técnico. **4 días**
- Después de la validación de la especificación funcional, se realiza la especificación técnica, esto es como se va a realizar la solución: elementos técnicos, lenguaje de programación...etc. Se le envía al cliente nuevamente para su validación: **5 días**.

Aprobación de los diseños y desarrollo de la solución:

Una vez aprobados los diseños se procede a la planificación de la estimación establecida en **14 días** para el desarrollo y los tests unitarios

Se inician los desarrollos:

- A los 4 días, se le pregunta al cliente una aclaración de una duda sobre una especificación, el cliente decide que lo que se estaba haciendo no se ha realizado como él quería.
- Después de discusiones sobre este punto de discordia, se hace referencia al documento de especificaciones funcionales aprobadas por el cliente, se añaden al presupuesto inicial **3 días más.**
- Unos días más tarde, se repite la misma anécdota y se añaden **4 días** más para implementar una parte de un desarrollo que a priori no se ha especificado.

Llegados a este punto, se han añadido **7 días** al presupuesto inicial además de lo que se ha tardado en las fases previas de especificación funcional y técnica.
Las idas y venidas de preguntas y respuestas, además del tiempo perdido en esperar las validaciones y respuestas del cliente, hacen que este desarrollo se fuera de presupuesto ya que ha pasado de los 23 días iniciales a 30 días.
adicionalmente al problema presupuestario, hay problemas de entendimiento de las necesidades del cliente ya que partimos de una especificación funcional pobre y con pocos detalles, estos problemas añaden lentitud sobre todo el proceso y comprometen los plazos de entrega y el buen desarrollo de otros proyectos los cuales dependen de la finalización de este.
A parte de mala planificación y especificación, el desarrollo no cumplía con los estándares básicos de calidad de la ingeniería del software, carecía de tests unitarios, de verificación de calidad del código, de procedimiento de integración continua con la parte de verificación de calidad del desarrollo (cobertura de código con los tests y las incoherencias de programación)
En el capítulo de la solución a la problemática, aplicaremos una de las metodologías ágiles para mejorar el proceso y los tiempos así que la calidad del producto entregado.

Se añadir una fase de tests unitarios que se ejecutarán dentro de la fase de integración continua para asegurar la coherencia y la validez del código.

De esta manera se evitarán retornos por parte del cliente y menos incidencias a la hora de la puesta en producción.

La aplicación de todos estos aspectos ofrecidos dentro del marco las metodologías ágiles, resultarán en un entregable funcional y con un desarrollo robusto y de calidad en cada iteración.

3 ESTADO DE LA CUESTIÓN

3.1 DEFINICIÓN

En los últimos años las metodologías ágiles se han convertido en un elemento esencial en el desarrollo de proyectos y procesos informáticos, muy solicitados en el mercado de trabajo para el sector del desarrollo de aplicaciones informáticas.

Las metodologías ágiles y su impacto en la mejora de los procesos software

Las metodologías de desarrollo ágil han ganado muchos adeptos entre las empresas consultoras informáticas, han aportado mucha flexibilidad y foco en los procesos informáticos, ya que las metodologías clásicas de desarrollo se ven obsoletas debido a la dinámica del sector.
La informática, en los últimos diez años, dio un salto enorme con el desarrollo de muchos Frameworks y cambios sustanciales en los existentes.
Además, había que empezar a cambiar el enfoque sobre las metodologías de trabajo que se habían vuelto muy costosas en tiempo y esfuerzo y empezar a emplear las metodologías ágiles.

3.2 METODOLOGÍAS ÁGILES

Las metodologías ágiles son una serie de técnicas o modos de trabajo que permiten adaptar la manera de trabajo a las circunstancias de un proyecto, esto permite que los equipos tengan una cierta flexibilidad y prontitud en las respuestas para ajustar el proyecto a las circunstancias impuestas por el entorno.
En esencia, son un grupo de tareas y procedimientos orientados a la gestión y pilotaje de los proyectos informáticos, estas tareas son capaces de evolucionar con el paso del tiempo según las necesidades establecidas por los equipos multidisciplinarios que trabajan juntos para alcanzar un objetivo qué es tener un producto completamente funcional y mantenible.
Estos equipos se distinguen por tener las siguientes características:

- Comunicación
- Planificación
- Autonomía de los equipos
- Desarrollo flexible y evolutivo.

En este contexto, aparece el “Manifiesto por el desarrollo Ágil de Software”, este manifiesto es un documento realizado por varios autores quienes establecieron los 12 principios del software ágil. [2]

Figura 1: los 12 principios del manifiesto ágil[1]

Estos 12 principios son [3] y [4]:

1- Satisfacer al cliente a través de entregas tempranas y continuas de software con valor añadido.
2- Aceptar que los requisitos cambien:
    - El cliente aprovecha el cambio para adquirir ventaja competitiva.
3- Entrega frecuente de software funcional:
    - La frecuencia de las entregas suele ser de entre dos semanas y dos meses, con la preferencia a los periodos más cortos, siempre que sea posible.
4- Los gerentes y los desarrolladores trabajan en equipo:
    - Esta colaboración se hace de forma cotidiana durante todo el ciclo de vida del proyecto.
5- Motivación de los individuos:
    - Hay que proporcionar un entorno adecuado a los componentes de los equipos del proyecto para que desarrollen sus tareas.
    - Darles la confianza necesaria para la ejecución de sus tareas.

[1] Autentia.com - entrega de fichas ágiles: https://www.autentia.com/recurso/cuarta-entrega-fichas-ágiles-coaching/

6- Comunicación directa:
    - La manera más eficiente para el paso de la información al equipo de desarrollo es la conversación cara a cara.
7- Un software tiene que ser funcional:
    - El progreso se mide con un software funcional y operativo.
8- La sostenibilidad del desarrollo es promovida por los procesos ágiles:
    - Todos los actores en un proyecto, desde el cliente hasta los desarrolladores, deben tener un ritmo constante durante toda la vida del proyecto.
9- Prestar atención a la calidad técnica:
    - La obligación de mantener un nivel alto en la calidad de los desarrollos y el diseño del software, mejoran la agilidad de los procesos.
10- La simplicidad:
    - Saber maximizar la cantidad de trabajo no realizado.
11- Equipos Auto organizados:
    - De estos equipos salen las mejores arquitecturas, diseños y requisitos.
12-Retrospectiva y autocritica:

    A intervalos regulares, los equipos reflexionan sobre su efectividad y como mejorarla para optimizar su comportamiento a continuación.

A partir de estos principios, se genera una estrategia para la aplicación de las metodologías ágiles en los proyectos de las organizaciones o empresas informáticas.

Para embarcarse en esta aventura, hay que evaluar las ventajas y limitaciones de dichas metodologías, Suponen varios beneficios para las empresas:

- Reactividad frente a los cambios:

Siendo el proceso ágil un proceso evolutivo, los equipos tienen la capacidad y la facilidad de implementar soluciones durante el desarrollo del proyecto sin tener que esperar hasta el final del ciclo.

- El rol del cliente en el proceso:

El cliente puede intervenir activamente en cada una de las etapas del proceso, aportando ideas, modificando prioridades, modificando las especificaciones del

producto y dando un feedback sobre lo que se está entregando en cada hito del proceso, mostrando su acuerdo o en su defecto relevando los defectos encontrados.

- **Entregas por "Fascículos"**

Este tipo de entregas por partes, mejora y optimiza el uso de los recursos y focalizan los trabajos de seguimiento y pilotaje.

El producto entregado al final, es en realidad una suma de todos los bloques o productos parciales que han sido controlados y supervisados cada vez.

- **Supresión de las tareas innecesarias**

La priorización de las tareas es el proceso más importante para iniciar el desarrollo de un producto, esto sirve para saber cuáles de las tareas son necesarias para empezar y cuales son menos prioritarias incluso innecesarias para centrar más à los equipos en las labores que hay que realizar y tener un modo más rápido de actuación frente a las circunstancias que puedan surgir durante la construcción de un o varios bloques de producto.

Sin embargo, aunque las ventajas suponen un aspecto muy atractivo de las metodologías ágiles, tiene algunos inconvenientes ya que muchas empresas que basan su actividad en entregas de productos por ciclo de vida encontraron serias dificultades en aplicar estas metodologías dentro de sus procesos:

- **Dependencia de los jefes**

Los equipos dependen, en este caso de las decisiones de sus jefes. Los continuos puntos de sincronización entre estos y el cliente y los cambios frecuentes, hacen que sean como un cuello de botella para todas las decisiones.

- **La falta de documentación**

Documentar proyectos es una parte esencial para su futura mantenibilidad o transfer hacia otros equipos para su evolución o mantenimiento, las metodologías ágiles solo indican como se tiene que realizar cada acción y no resuelven la disyuntiva de la recogida y almacenamiento de datos e información.

- **Soluciones equivocadas en iteraciones extendidas**

Cuando las soluciones propuestas se extienden en etapas muy largas, tienden a ser plagadas de errores y acaban siendo distintas a lo que se ha pedido por parte del cliente en el inicio, lo que implica un tiempo perdido y trabajo por duplicado para poder enderezar la iteración.

- **Uso de las metodologías ágiles en las empresas**

El siguiente grafico muestra el porcentaje de uso de cada metodología en el mercado mundial en el año 2016:

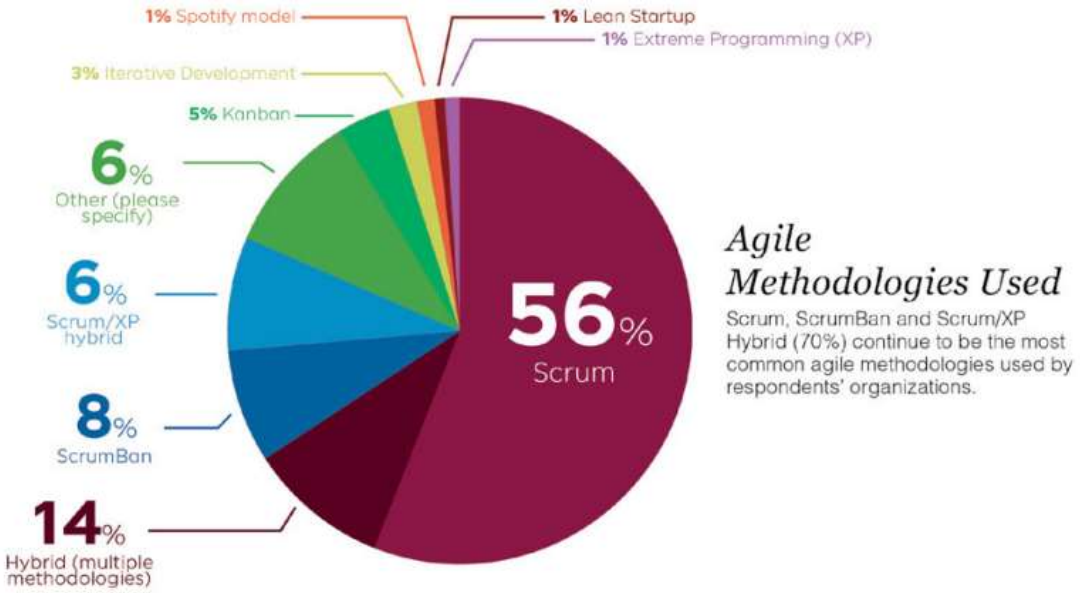

Figura 2- Distribución de uso de las distintas metodologías ágiles - Informe 2016[2]

El 56% de las empresas optan por el uso de la metodología Scrum, el otro 44% se distribuye entre las otras metodologías sean integrales o híbridas (combinación entre dos o más metodologías).

En el siguiente grafico se muestran como es el impacto en termino de existo y fracaso de las metodologías ágiles frente a la clásica waterfall o cascada:

[2] Jose Antonio Eusamio Mazagatos - https://www.astic.es/sites/default/files/articulosboletic/boletic82-tecnologia3-jaeusamio.pdf

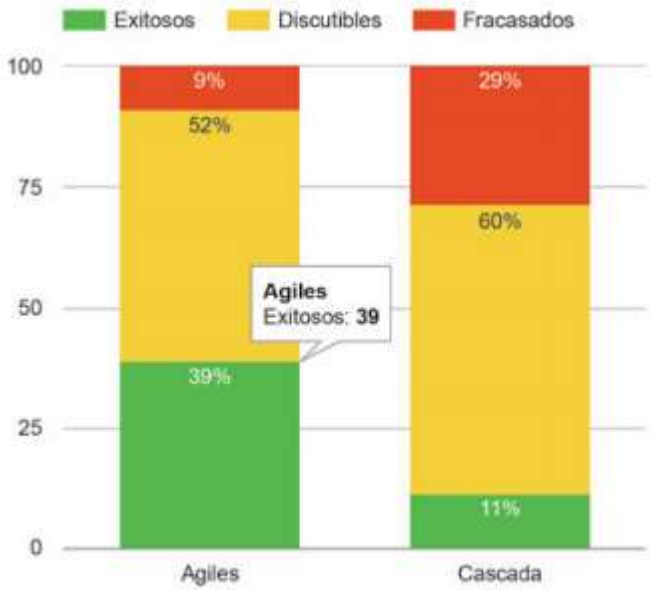

Figura 3: Existo y fracaso según el modelo ágil frente al tradicional[3]

En el siguiente capítulo se adentrará en el detalle de cada una de las metodologías más usadas en el mercado, enfocándose en Scrum, XP (Extreme programming) y Kanban.

En este capítulo se presentarán las metodologías más más conocidas y usadas en el mercado de las consultorías informáticas.

3.2.1 Scrum

Es un Framework con el cual las personas abordan problemas complejos y entregan producto de una manera eficiente con un valor óptimo.

A partir de esta definición se puede empezar a sacar propiedades particulares de Scrum: es ligero, fácil de asimilar y tiene una dificultad para dominarlo.

Scrum no es un método sino un marco de trabajo, donde se emplean un conjunto de procesos y técnicas para llevar a cabo un proyecto hasta su final. La eficacia de dichas técnicas en términos de gestión de producto y de trabajo hace que continuamente se pueda mejorar el producto final, los equipos y el entorno del trabajo.

"Scrum es un marco de trabajo iterativo e incremental para el desarrollo de proyectos, productos y aplicaciones. Estructura el desarrollo en ciclos de trabajos llamados Sprints. Son iteraciones

[3] Cristina Bombín del Palacio, PMP – TIC, Metodologías ágiles y contratación publica https://pmi-mad.org/index.php/quienes-somos-2/proyectos-publicos/343-articulos/1881-tic-metodologias-ágiles-y-contratacion-publica

de 1 a 4 semanas, y se van sucediendo una detrás de otra. Los Sprints son de duración fija – terminan en una fecha específica, aunque no se haya terminado el trabajo, y nunca se alargan. Se limitan en tiempo. Al comienzo de cada Sprint, un equipo multifuncional selecciona los elementos (requisitos del cliente) de una lista priorizada. Se comprometen a terminar los elementos al final del Sprint. Durante el Sprint no se pueden cambiar los elementos elegidos. [...]" (The Scrum Primer, 2009, pág. 5).

Todo marco de trabajo Scrum se compone de las siguientes partes:

- Equipos Scrum y sus roles
- Eventos
- Artefactos y sus reglas asociadas

Cada uno de estos componentes tiene un propósito concreto y es esencial su presencia para el éxito del proyecto dentro de este marco de trabajo Scrum.
Dentro de este contexto, existen las reglas de Scrum, Las reglas se detallarán cuando se hablará de los distintos roles que intervienen en el marco de Scrum:
*"[…]Las Reglas de Scrum relacionan los roles, eventos y artefactos, gobernando las relaciones e interacciones entre ellos […]" (*La guía Scrum 2017*)* [4].

Los roles Scrum se distribuyen de la siguiente manera:

- **Scrum Máster**: es el responsable de gestionar y organizar los procesos.
- **Producto Owner** o el **Dueño del producto** quien juega el papel de voz del cliente y es el responsable de que el producto final tenga el valor buscado.
- El **Equipo de desarrollo,** es el equipo a quien se encarga la creación del proyecto y las entregas.

La figura 4 muestra un resumen de las funciones y de la relación entre los distintos roles.

[4] Scrum Guide - versión en español:
http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%20ES.pdf

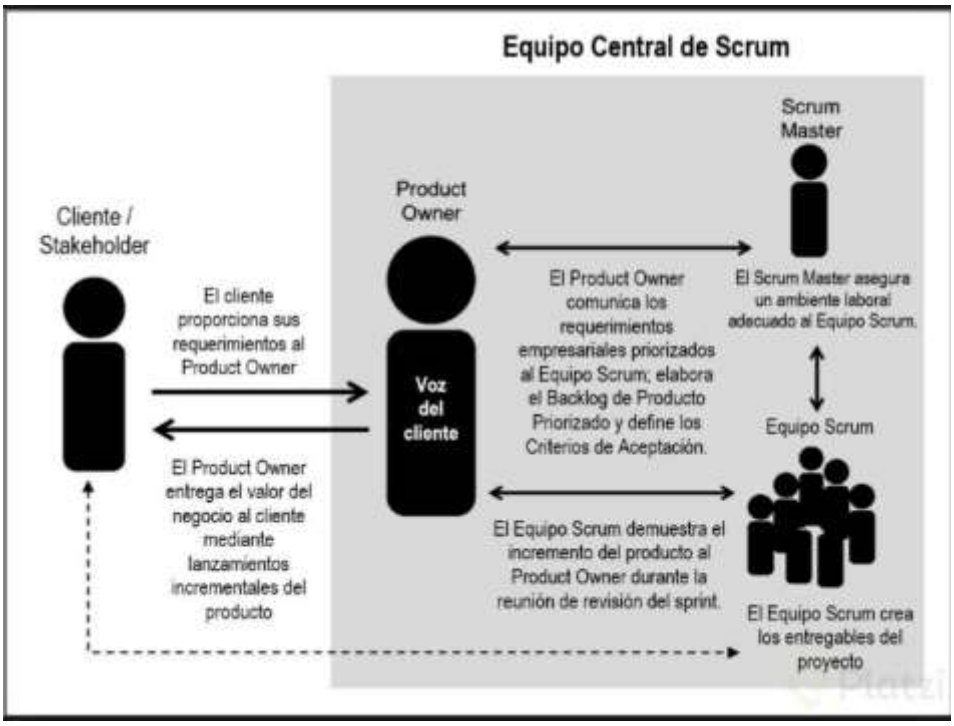

Figura 4: Roles de Scrum y sus principales funciones[5]

Scrum Máster

Tiene un rol de servidor para los otros roles Scrum, no es un líder propiamente dicho, sino ayuda a los equipos y organizaciones a asimilar los conceptos y principios ágiles de Scrum.

"El Scrum Master es el responsable en promocionar y apoyar Scrum como se define en la Guía de Scrum. Los Scrum Masters hacen esto ayudando a todas las teorías de Scrum, prácticas, reglas y valores." (Scrum Guide, 2017, Pág. 8)

Responsabilidades del Scrum Master

El Scrum Master tiene varias funciones las que destacaremos:

- Hacer de árbitro para resolver los conflictos que puedan surgir dentro de un equipo o entre los otros roles, establece el ambiente adecuado para llegar a una situación de entendimiento entre ellos.
- Fomentar y motivar al Equipo Scrum, creando un clima de trabajo de conjunto, hacer hincapié en la autogestión y auto planificación de los equipos de desarrollo, bloquear cualquier incursión de terceros en esta misma gestión.

[5] Yesica Lizeth – Qué es Scrum y los roles en Scrum - https://platzi.com/blog/que-es-scrum-y-los-roles-en-scrum/

- Asegurar la asimilación de los principios de scrum por todas las esferas de la organización y que cada parte (producto Owner y Equipo de desarrollo sepan cuáles son sus tareas y evitar que cada uno intente hacer el trabajo del otro.

La figura 5 indica las funciones desempeñadas por el Scrum Master:

Figura 5 Roles y responsabilidades del Scrum Master.[6]

Aptitudes del Scrum máster:

En cierta manera, el Scrum Master es el dueño del proceso y decide cómo se va a llevar, pero ciñéndose a las características de las empresas.
Y por ello debería de tener ciertas aptitudes personales y profesionales para desempeñar esta labor:

- Tener unos amplios conocimientos del marco de trabajo Scrum, ser conocedor de los valores y los principios de los procesos ágiles, tener visión clara de las reglas organizativas y prácticas de Scrum.
- Tener un rol servil frente a los otros miembros sin pretensiones de liderazgo ya que eso no es un objetivo de un Scrum Master, sino de acompañar, enseñar y educar al Scrum team en la agilidad.
- Tener facilidad y capacidad de resolución de problemas todos los bloqueos e impedimentos que puedan surgir durante el desarrollo de una iteración o de todo un producto, esta aptitud es esencial para que

[6] Hari Krishan Verma - http://knowledgeblob.com/agile/agile-scrum-master-role-and-responsibilities/

el proceso llegue a buen puerto y resulte en un producto de alta calidad.

- Tener la capacidad de ver los detalles y vigilante ante posibles situaciones que puedan aparecer y a las cuales el equipo de Scrum tenga que enfrentarse.
- Tener metodología y el saber hacer llegar las ideas al Scrum team y también a toda la organización.
- Imponer su posición como mediador de una manera carismática de tal manera evitará las que las tensiones provocadas por los cambios hacia un mundo ágil en la organización tengan protagonismo en el Scrum team y/o la organización.

"El Scrum Master es el responsable en promocionar y apoyar Scrum como se define en la Guía de Scrum. Los Scrum Masters hacen esto ayudando a todas las teorías de Scrum, prácticas, reglas y valores." (Scrum Guide, 2017, Pág. 8)

Errores que tiene que evitar un Scrum Master

Figura 6: Errores comunes del Scrum Master[7]

Un Scrum máster NO tiene que:

- No prestar servicios a los demás miembros del Scrum Team (producto Owner y el equipo de desarrollo.

[7] beagilemyfriend.com. (s.f.). *https://www.beagilemyfriend.com/errores-scrum-masters*

- Dirigir los eventos de la iteración o Sprint, realmente los tiene que apoyar y ver si siguen las normas Scrum.
- Solo tiene que trabajar con un solo tipo de sprint o iteración, de esta manera el procedimiento ágil está más enfocado y centrado en el objetivo de dicha iteración.
- Organizar el trabajo del equipo de desarrollo, ya que según las normas Scrum, los equipos tienen que ser auto gestionado y multi disciplinares.
- Evangelizar o centralizar el conocimiento Scrum en sí mismo, tiene que educar a los demás en el equipo Scrum y en la organización en los valores de Scrum.
- Separar los equipos: los equipos trabajan en conjunto dentro de una iteración y su trabajo se complementa, no se tiene que separar y evitar que se comuniquen, eso hará que los avances de cada equipo sean desconocidos para el otro, esto afectaría al avance de la iteración y por lo tanto la entrega del producto se vería mermada.

Product Owner o Propietario del producto

"El Propietario del Producto (Product Owner) es el responsable de maximizar el valor del producto del trabajo del Equipo de Desarrollo (Development Team). Cómo se lleva a cabo puede variar ampliamente entre distintas organizaciones, equipos Scrum e individuos.
El Propietario del Producto (Product Owner) es la única persona responsable de gestionar la Pila del Producto (Product Backlog)." (Guía Scrum 2017-pg 6)

El propietario del producto juega el rol de representante del cliente dentro del equipo Scrum, definiendo las tareas que se tengan que hacer en cada iteración y estableciendo la prioridad que tiene cada una, gestiona lo que se llama la pila de productos o Product Backlog que es un artefacto Scrum, que se definirá cuando se hable de los artefactos Scrum.
La figura 7 resume los atributos que debe tener un Product Owner:

Figura 7: Atributos de un producto Owner[8]

- **Visión**

Debe tener la capacidad de ver hacia dónde va el producto que pide al equipo de desarrollo, si va a buen puerto y que el equipo vaya hacía él.

- **Mentalidad "Most Valuable Product"**

Lo que viene a ser el producto o funcionalidad con más valor, el Product Owner, tiene que centrar al equipo en hacer entregas de las funcionalidades más críticas o valiosas, con la certeza de que son esas funcionalidades las que van a ser las más usadas.

- **Comprensión de la competencia**

Tiene que estar atento a lo que pasa alrededor, el sector informático está en constante evolución, se sacan productos nuevos productos de manera constante, estar pendiente de las oportunidades que puedan surgir en cualquier momento para ser ágil en la toma de decisiones que podrían afectar positivamente al proceso para mejorar su producto.

- **Comunicador y colaborador**

Tener habilidades comunicadoras es un plus, también saber manejar los hilos con distintos interlocutores, es una manera de colaborar en los desarrollos de las iteraciones hasta conseguir terminar el producto final.

- **Negociador**

En momentos de criticidad, donde los proyectos pueden estar estancados debidos a factores humanos o técnicos, el Product Owner debe saber cómo tratar

[8] https://www.beagilemyfriend.com/atributos-product-owner/

con las distintas partes para poder llegar a soluciones que hagan que esos problemas se resuelvan y el trabajo siga adelante.

- **Empoderamiento**

Como su propio nombre lo indica, el Product Owner representa la máxima autoridad dentro de Scrum Team, sobre los aspectos esenciales del producto, organización de las prioridades y que decisiones se deben tomar al respecto.

- **Siempre disponible**

Debido a que las muchas de las tareas que tiene el Product Owner no tienen relación con Scrum, simplemente le es difícil estar disponible para el equipo de desarrollo cuando este lo solicita.

Muchos de los proyectos Scrum son multi-equipos y multidisciplinares, es más puede que estén dispersados en varios sitios en el mundo con horarios distintos. En este caso, es el producto Owner quien debe encontrar la manera de hacer que pueda estar disponible en algún momento concreto del día para resolver las posibles peticiones que les puedan llegar.

Equipo de desarrollo o Development team

Figura 8 Development team dentro de un proyecto Scrum[9]

"El Equipo de Desarrollo (Development Team) se compone de profesionales que realizan el trabajo de entregar un Incremento de producto "Terminado" ("Done") que potencialmente se pueda poner en producción al final de cada Sprint. Un Incremento de producto "Terminado" es obligatorio en la Revisión del Sprint (Sprint Review). Solo los miembros del Equipo de Desarrollo (Development Team) participan en la creación del Incremento." (Guía Scrum 2017-pg 7).

[9] https://www.devteam.space/blog/how-to-build-a-scrum-development-team/

El Development team, es compuesto por desarrolladores multidisciplinarios, van desde programadores hasta testers, se auto gestionan para desarrollar el producto que se entregará al final al cliente.

Tareas del equipo de desarrollo

- Transformar la pila de producto (Backlog) en producto entregable para el final de cada iteración (sprint)
- Controlar su disciplina, adquiriendo más experiencia constantemente.
- Tener la actitud necesaria y saber cómo trabajar en equipo.
- Ser autosuficientes y saber cómo gestionarse dentro del equipo cada miembro.

Los miembros del equipo de desarrollo deben demostrar tener unas ciertas actitudes personales que hacen que el equipo sea cohesionado y sepa desarrollar el trabajo en un ambiente colaborativo y no competitivo.

Esto se traduce en una serie de “virtudes” que debería tener cada miembro de los equipos de desarrollo Scrum:

- Tiene que ser una persona solidaria y colaborativa, se tiene que preocupar del bien estar profesional de los compañeros, dar la ayuda necesaria si alguno lo necesita, aportar o donar conocimiento cuando algún compañero lo necesite y no lo tenga.

  Siempre que pueda, un miembro del equipo debe colaborar en el avance de una tarea de un compañero que este atascado en la misma.
- Tiene que ser una persona motivadora y no tratar de destacar.

  Un buen miembro del equipo Scrum, evita buscar sobresalir a costa de otros compañeros mostrando su grado de conocimiento técnico frente a los de sus compañeros, sino debe motivarlos a que mejoren los suyos y acudan a él en caso de necesidad para adquirirlo.
- Tiene que evitar la competencia, en vez de invertir tiempo en ser o parecer mejor que los compañeros en el mismo Dev team, es mejor invertirlo en aprender de sus experiencias y empaparse de su maestría de las técnicas y tecnologías para mejorar continuamente, esto tendría un impacto positivo para que el equipo llegue a tiempo y con los mejores resultados al final de cada iteración.

Artefactos y herramientas Scrum

"Los artefactos de Scrum representan trabajo o valor en diversas formas que son útiles para proporcionar transparencia y oportunidades para la inspección y adaptación. Los artefactos definidos por Scrum están diseñados específicamente para maximizar la transparencia de la información clave, necesaria para asegurar que todos tengan el mismo entendimiento del artefacto." (Guía Scrum 2017-pg 15)

Los artefactos Scrum son los elementos que permiten que el trabajo de todos los miembros del equipo Scrum (Scrum Master, Product Owner y el Equipo de desarrollo) sepan que es lo que hay que hacer y cómo hay que hacerlo en cada iteración para llegar bien al final con un producto plenamente funcional.
Estos artefactos son:

- **Product Backlog o lista de productos:**

Es una pila priorizada de funcionalidades, de la misma manera todos los requisitos del producto que se está desarrollando por el equipo de desarrollo, es responsabilidad del dueño del producto, es quien decide su contenido, su validez y las priorizaciones.
Todo lo que se quiere saber del producto está en la back log, es una serie de características, mejoras y correcciones del producto final o en su caso de las que en un futuro se van a hacer.
Un ejemplo de los ítems u objetos que contiene la back log:

- Casos de uso del producto a desarrollar
- Mejoras funcionales
- Mejoras técnicas
- Especificaciones sobre elementos de integración y/o instalación.

La figura 9, da un ejemplo de lo que es una Product Backlog:

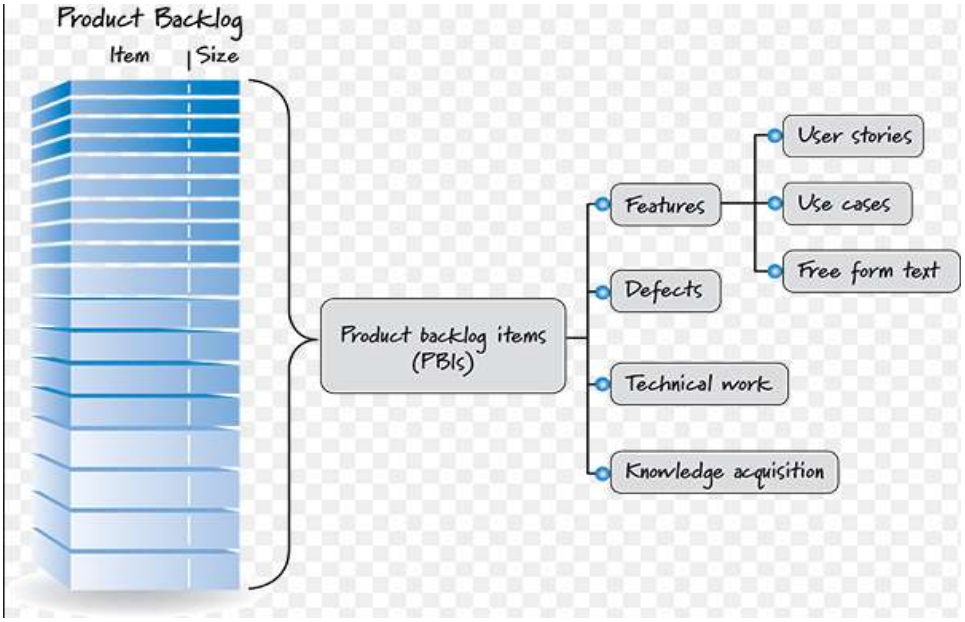

Figura 9: Ejemplo de Product Backlog[10]

Las particularidades de la Product Backlog son entre otras:

- La priorización de los desarrollos:

El responsable de esta tarea es el Producto Owner, es quien establece que ítem u objeto es más prioritario que otro, dependiendo de varios factores:

  - Su utilidad o ganancia para el producto final y su nivel de criticidad
  - Los riesgos que supone desarrollar dicho ítem en primer lugar frente a otros. Qué valor o valores añadidos respecto a funcionalidades homologas presentes en el mercado.
  - Sí esta acción es coherente con la política de intereses generados por el negocio.

- La estimación:

El esfuerzo que llevaría hacer el desarrollo de los ítems más prioritarios es importante y los menos prioritarios o los que están en duda (pendientes de validación por parte del cliente o el stackholder o por falta de recursos.), se establece por parte del dueño del producto que debe hacerlo, aunque es una tarea del equipo de desarrollo, el Scrum Master puede enseñarle cómo hacerlo.

- La granularidad de sus objetos (un objeto es un componente de la Product Backlog)

[10] https://innolution.com/essential-scrum/table-of-contents/chapter-6-product-backlog

El grado de granularidad de los ítems de una Backlog de producto define su tipo, si la granularidad des alta se agrupan en “Epics” y en el caso contrario se agrupan en historias de usuario o “User stories”

Una User Story es simplemente definir una necesidad dentro del contexto de un usuario, generalmente se asocia a un caso de uso cuando se define un software, pero para simplificar: Es definir el tipo de usuario y lo que quiere hacer o el problema a resolver para alcanzar un objetivo, aunque se da el caso de que el objetivo se releva a un segundo plano o se obvia.

La figura 10 muestra un ejemplo de user stories:

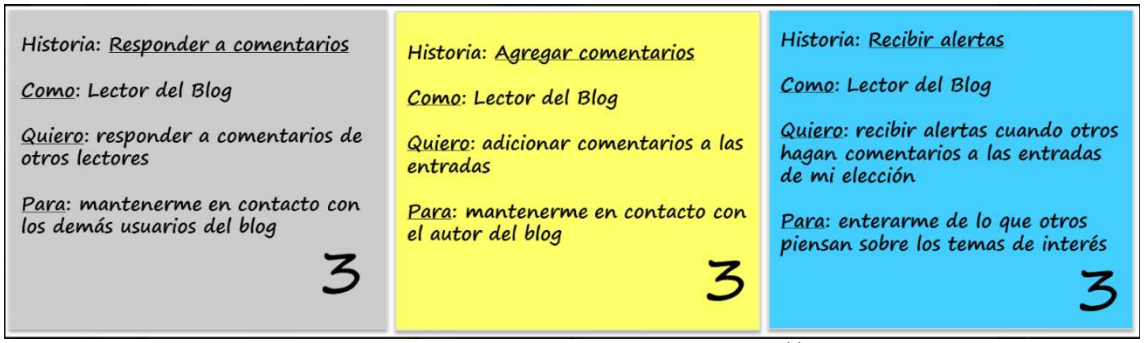

Figura 10: ejemplo de user stories[11]

- **Criterio de aceptación**

Para que un ítem desarrollado por una user **story** o **Epics** sea aceptado por el Product Owner para que se incorpore a los otros ítem que constituyen el producto final, tiene que pasar unas pruebas de aceptación o lo que viene a ser los tests de validación, esto es establecer unas pautas que el ítem deba cumplir antes de ser finalmente validado.

Por ejemplo: como viajero quiero sacar un billete de avión con mi foto para que el control de mi identidad sea más fácil.

Los criterios de aceptación pueden ser:

1- Validar que la foto tiene un formato admitido (jpeg, bmp …).
2- La foto tiene unas dimensiones concretas para poder tenerla en el billete.
3- La foto debe imprimirse en la esquina superior derecha.
4- En la foto tiene que ver una cara y nítida.

[11] http://www.gazafatonarioit.com/2013/08/escribiendo-historias-de-usuario.html

- **Backlog de Sprint**

"La Lista de Pendientes del Sprint es el conjunto de elementos de la Lista de Producto seleccionados para el Sprint, más un plan para entregar el Incremento de producto y conseguir el Objetivo del Sprint. La Lista de Pendientes del Sprint es una predicción hecha por el Equipo de Desarrollo acerca de qué funcionalidad formará parte del próximo Incremento y del trabajo necesario para entregar esa funcionalidad en un Incremento Terminado." (*Guía Scrum 2017-pg 16-17*).

La figura 11 muestra un ejemplo de lo que es una Sprint Backlog:

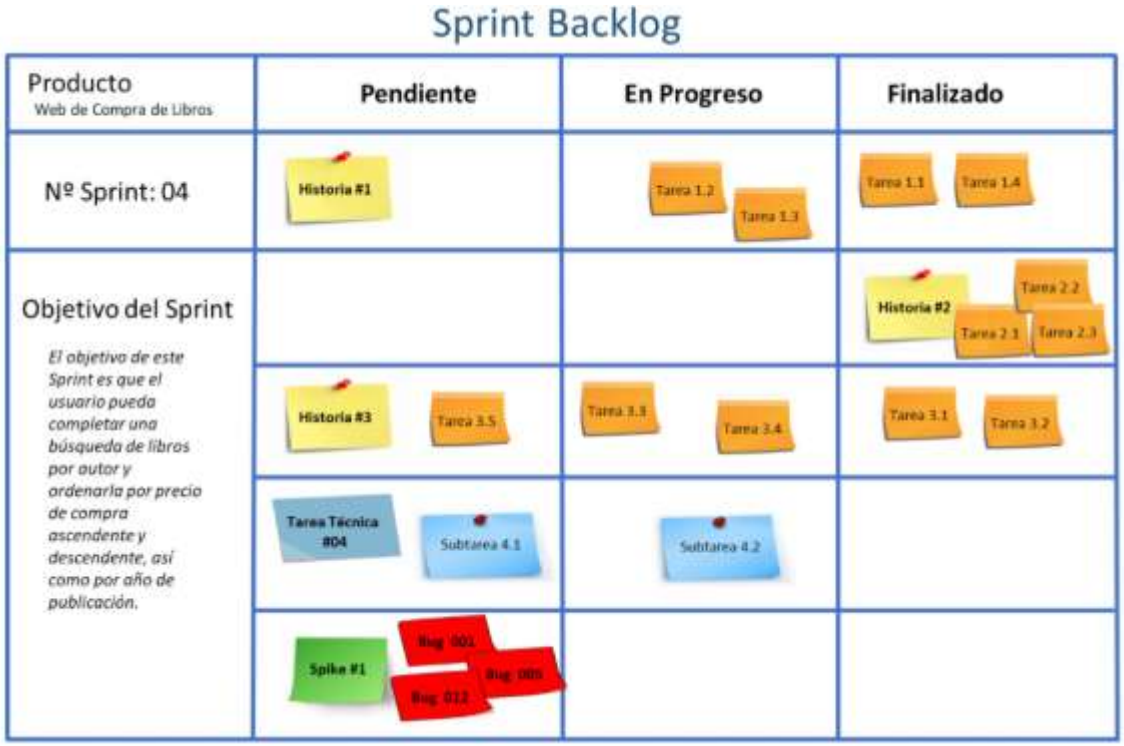

Figura 11: Sprint Backlog[12]

La Sprint Backlog se compone de los ítems del Product Backlog, que fueron elegidos según la prioridad establecida y que van a ser desarrollados durante el sprint (iteración). En cada ítem seleccionado se establece el detalle de las tareas que ira desarrollando el equipo de desarrollo para alcanzar una entrega parcial y funcional al final de la iteración, esta parte del producto se le llama "Incremento".

Es una lista que se crea a cada inicio de iteración, estableciendo la estimación de cada tarea que suele ser de un día de trabajo, el equipo de desarrollo es el responsable de poner al día la lista manera cotidiana, la información implicada es:

[12] https://muyagile.com/product-backlog-y-sprint-backlog/

- Las tareas pendientes de empezar, las que están en desarrollo y las que ya están cerradas o terminadas.
- Lo que queda en la estimación de esfuerzo de cada tarea hasta su finalización.
- El nombre del responsable del desarrollo de dicha tarea.

La herramienta usada para tal propósito suele ser física (Tablero con etiquetas llevando la información mencionada arriba) o virtual (dentro de alguna herramienta de gestión de proyectos) como ***Jira o Trello***[13] donde también se crean las etiquetas en columnas especificando el estado de cada tarea y por su tipo (incidencia, mejora o test).

Llegado a este estado, ¿cómo se dividen las historias de usuario o ítem de Product Backlog en tareas?

La respuesta es desgranando los ítems desde el aspecto general pasando al aspecto particular lo que desemboca en el detalle de cada tarea.

- Aspecto general:
  Es cuando definimos qué es lo que queremos hacer y de que se trata.
- Aspecto particular:
  Es definir la manera de cómo vamos a realizar el aspecto general, puede tratarse de detalles de software, arquitectónicos, tecnología…etc.
- Aspecto detallado:
  Es cuando se define cuáles son las tareas necesarias para llegar a crear el incremento del producto del objetivo del sprint.

Eventos de Scrum

"En Scrum existen eventos predefinidos con el fin de crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum. Todos los eventos son bloques de tiempo (time-boxes), de tal modo que todos tienen una duración máxima. Una vez que comienza un Sprint, su duración es fija y no puede acortarse o alargarse. Los demás eventos pueden terminar siempre que se alcance el objetivo del evento, asegurando que se emplee una cantidad apropiada de tiempo sin permitir desperdicio en el proceso." (Scrum Guide 2017. Pag.9).

[13] www.trello.com & www.altassian.com/es/software/jira

El principio de “timeboxing” definido en por Scrum es, junto con un ritmo de desarrollo sostenible en el tiempo, entregas de software de valor y funcional de manera regular y una alta capacidad de adaptación de los equipos, la base fuerte de una adopción de los principios de Scrum:

- Aprendizaje.
- Adaptación.
- Inspección.

A partir de ahí entraremos en los distintos eventos definidos por Scrum.

Definición de Sprint (Iteración)

El sprint es la pieza más importante sobre la que gira Scrum como marco ágil, se trata de una etapa donde el trabajo se hace de forma incremental, entregando a cada final de sprint una parte del producto que, funcionalmente, aporta un valor a las partes ya entregadas.

Un sprint tiene una duración de entre 2 y 4 semanas, pudiendo ser más corto o largo dependiendo del tipo de negocio, si es un entorno muy cambiante o no. Un sprint se cierra al principio, es decir: se establecen todos los temas que se van a tratar en él y si hay cambios por parte del Product Owner, sean de priorización o de especificación, ese cambio se encaja sacan lo menos prioritario al sprint siguiente.

Sí un sprint tiene previsión de terminar antes de lo previsto o en el caso contrario, se prevé que terminará más tarde de lo planificado, hay que ajustar el sprint Backlog para poder terminar en ese plazo preestablecido.

La figura 12 establece una definición resumida del proceso de un Sprint (como ejemplo).

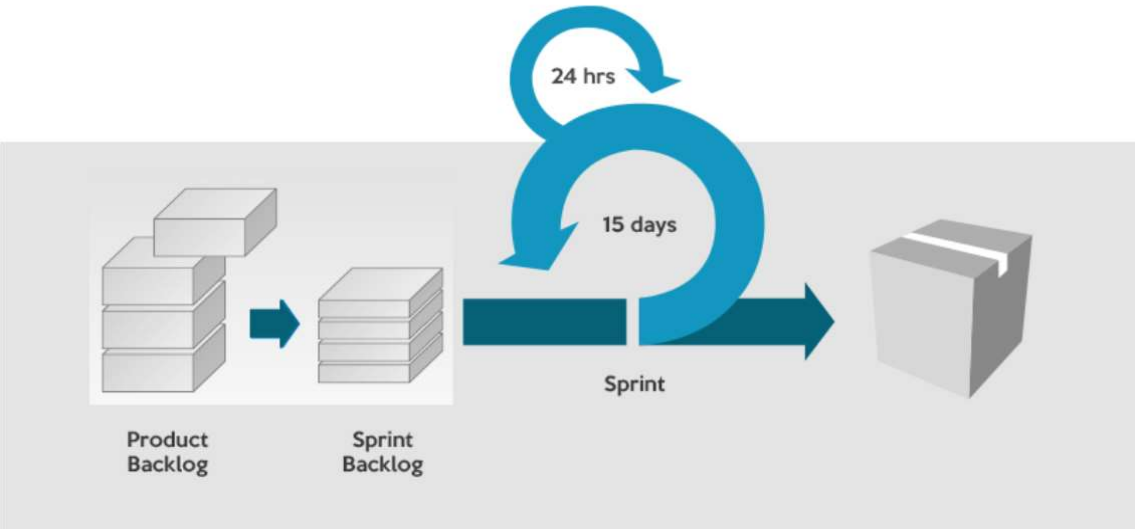

Figura 12: vista resumen de un Scrum sprint[14]

Planificación del sprint (Sprint Planning Meeting)

Es la reunión que se realiza en el inicio de cada Sprint, es facilitada por el Scrum Master para que asisten, además de él, el Product Owner y el Equipo de desarrollo.

En esta reunión, el equipo de desarrollo y el Product Owner, se ponen de acuerdo sobre qué de la Product Backlog se va a incluir dentro de ese sprint, el Product Owner indicara desde su Backlog cada ítem a incluir y a realizar para finales de sprint, el Equipo de desarrollo se encargará de estimar el esfuerzo a emplear en cada uno, teniendo en cuenta los posibles cambios que puedan surgir y las tareas parecidas que se había realizado en el pasado.

Al mismo tiempo el equipo tiene que hacer las preguntas pertinentes y/o aclaraciones necesarias al producto Owner para entender mejor el ítem a realizar, evitando malentendidos y futuros retrasos e incidencias que ello pueda suponer.

En un segundo tiempo de la reunión, el equipo de desarrollo y el Scrum Master, con la información entre manos sobre los ítems a incluir en el siguiente incremento, debate qué:

- Tareas se van a generar y realizar.
- Como se van a hacer y desarrollar.
- A quien se afectara cada ítem/tarea.

[14] Scrum process overview https://luis-goncalves.com/pt-pt/o-que-e-scrum/

Con todos estos elementos definidos de forma precisa, se empiezan a identificar cada tarea con su estimación, sus plazos y a quien le está afectada, y se coloca dicha información en la pizarra o tablero Scrum para su seguimiento.
El Product Owner puede ser contactado en cuanto una duda o ambigüedad surja mientras se esté especificando o desarrollando el trabajo que se va a hacer.

Scrum Diario (Daily Stand-Up)

Es una reunión diaria, de no más de 15 minutos, donde se juntan el equipo de desarrollo y el Scrum Master, Se puede requerir la presencia del Product Owner o el cliente (stackholder) si es necesaria su presencia.
En esta reunión, que se hacer de pie delante de la pizarra dónde se encuentran definidas las tareas.
Hay un maestro de la ceremonia del daily, puede ser el Scrum Master o cualquier miembro del equipo de desarrollo, se pasando la palabra miembro por miembro hasta el último, cada uno comenta en qué ha dedicado la jornada anterior y a qué va a dedicar la jornada en curso y/o qué tareas terminará o empezará.
También se discuten los bloqueos, sean técnicos (problemas de entornos, herramientas o comunicación) o funcionales que requerirán la intervención del Product Owner.
Esta reunión diaria a lo largo del sprint fomenta la comunicación entre el equipo, también la mejora de los conocimientos que puedan aportar cada uno en situaciones que lo requieran (alerta, bloqueos, incidencias criticas…etc.), la sensación de compromiso de los miembros del equipo con el trabajo y también la visibilidad de lo que está haciendo cada uno todos los días de la duración del Sprint.

Revisión de Sprint (Sprint Review)

Esta es una ceremonia que se celebra el último día del Sprint su duración varia de la hora a una jornada laboral entera, donde se presentan los resultados del sprint al Product Owner, los usuarios, los clientes y también los managers. Generalmente se le llama demo o demostración, en ella se les cuenta a los asistentes por parte del equipo de desarrollo, como ha ido el sprint, las dificultades encontradas las mejoras aportadas y las tareas desarrolladas.

El Product Owner, en la demo, averigua si la funcionalidad entregada es acorde con el objetivo del sprint para validar el incremento y estar seguro de que esta entrega es una parte valida y aporta valor al producto.
Los usuarios dan el visto bueno cuando verifican que los criterios de validación establecidos en la ceremonia de la planificación inicial del sprint.
El producto Owner y los otros asistentes pueden pedir aclaraciones sobre cualquier parte del sprint, también se podrá actualizar la Product Backlog.
Al final de la revisión, el Scrum Máster establece las fechas para el próximo encuentro al final del siguiente del sprint.

Retrospective de Sprint (Sprint retrospective)

Es una es una especie de autocrítica constructiva, es cuando se reúnen el equipo de desarrollo con el Scrum Master, donde se debaten los puntos fuertes del equipo en este sprint y las oportunidades de mejora, con vistas a los Sprint siguientes, se hace un aporte de ideas por parte de todo el equipo para que se apliquen en cuanto a mejoras con respecto a lo que se va a desarrollar para alcanzar el siguiente incremento.
La figura 13 hace una ilustración de las dos ceremonias

Figura 13 : Sprint Review y sprint Retrospective.[15]

Valores de Scrum

"Cuando el Equipo Scrum incorpora y vivencia los valores de compromiso, coraje, foco, apertura y respeto, los pilares Scrum de transparencia, inspección y adaptación se materializan y fomentan

[15] https://programacionymas.com/blog/sprint-review-meeting-y-sprint-retrospective

la confianza en todo el mundo. Los miembros del Equipo Scrum aprenden y exploran estos valores a medida que trabajan en los eventos, roles y artefactos de Scrum.
El uso exitoso de Scrum depende de que las personas lleguen a ser más virtuosas en la convivencia con estos cinco valores. Las personas se comprometen de manera individual a alcanzar las metas del Equipo Scrum. Los miembros del Equipo Scrum tienen coraje para hacer bien las cosas y para trabajar en los problemas difíciles. Todos se enfocan en el trabajo del Sprint y en las metas del Equipo Scrum. El Equipo Scrum y sus interesados acuerdan estar abiertos a todo el trabajo y a los desafíos que se les presenten al realizar su trabajo. Los miembros del Equipo Scrum se respetan entre sí para ser personas capaces e independientes." (Scrum Guide 2017. Pag.5).

Para el marco de trabajo Scrum, hay una serie de valores que se verifican y se refuerzan durante toda la vida del proceso Scrum en un proyecto concreto.
Estos valores se resumen en:

1- Valor de los equipos frente a los proceso y herramientas
   El equipo humano de un proyecto es el punto más fuerte en cualquier organización, en él se debe basar su construcción y desarrollo, un equipo fuertemente formado en la parte técnica y que haya construido su propio entrono de trabajo, puede ser un factor decisivo de éxito en el desarrollo del producto final, Scrum pone mucho énfasis en este punto.
2- Valor del producto funcional frente a la documentación
   Como la funcionalidad de un software es el punto métrico esencial en la medición del progreso del proyecto, generar documentación debe ser un aspecto segundario enfocado a lo más importante, no hay que perderse en los detalles que no aportan nada a la funcionalidad que se esté desarrollando en una o más iteraciones.
3- Valor de la relación cliente frente a las condiciones de contrato
   Una buena relación con el cliente también es un punto fuerte, genera confianza, esa relación tienen que entrar dentro del marco de la colaboración, el cliente puede aportar soluciones a problemas que surjan, pero también el equipo de desarrollo puede anticiparse y aportar soluciones en el otro sentido. Esta colaboración no solo puede garantizar un gran éxito en el producto final sino generar posibles acuerdos futuros de trabajo.
4- Valor de la flexibilidad con los cambios frente al ceñirse a lo establecido

Un equipo que sabe adaptarse a los cambios de manera rápida y eficiente es un equipo que se mueves con mucha facilidad dentro de los entornos parecidos.

Una planificación con previsión de posibles cambios en las especificaciones, entornos técnicos y/o recursos humanos, es una planificación que puede garantizar un éxito en el desarrollo de la actividad, en el caso contrario puedo ser un factor determinante en el retraso o el fracaso de un producto funcional o final.

En la siguiente tabla, se enumeran las ventajas y limitaciones de Scrum

Scrum	
Ventajas	Limitaciones
Gestión de las expectativas de los clientes: Se permite la participación del cliente en todo el ciclo de vida de las iteraciones	Funciona sobre todo con equipos reducidos: En cuanto el equipo exceda el numero de 9 persones, se empiezan a segmentar en pequeños equipos.
Resultados anticipados: Los resultados de cada iteración son visibles al final de cada una, no hay que esperar hasta el final para verlos.	Requiere una exhaustiva definición de las tareas y sus plazos: hay que estar pendiente de los detalles para no dejar nada sin atar, de otra manera Scrum no tiene sentido.
Flexibilidad y adaptación a los contextos: Amplio abanico de sectores económicos donde se pueda aplicar	Exige una alta cualificación o formación: para entender la esencia de scrum, se necesitan de profesionales experimentados.
Gestión sistemática de riesgos: Los riesgos son tratados en cuanto surjan, los equipos se involucran en sur mitigación de inmediato	

Tabla 1 Ventajas y limitaciones de Scrum.

3.2.2 Programación Extrema

De los autores del manifiesto ágil, ***Kent Beck, Ward Cunningham y Ron Jeffries,*** es el origen de la identificación de la programación extrema a la que se llamaría “XP”.

Es una metodología del desarrollo del software que se basa en el grado de flexibilidad de un proyecto frente a la previsión de los cambios que puedan surgir a lo largo de la vida de un proyecto.

La metodología XP, pretende establecer el hecho de que el poder de un proyecto para adaptarse a los cambios es más fructífero y beneficioso para alcanzar sus objetivos, que intentar definir todos los escenarios o especificaciones en la fase inicial.

Es una metodología que, en comparación a Scrum, solo propone un conjunto de herramientas técnicas que cuando si se aplican al mismo tiempo, persiguen el objetivo de alcanzar unos efectos beneficiosos en el software final.

La figura 14 resume la definición del funcionamiento de la metodología XP

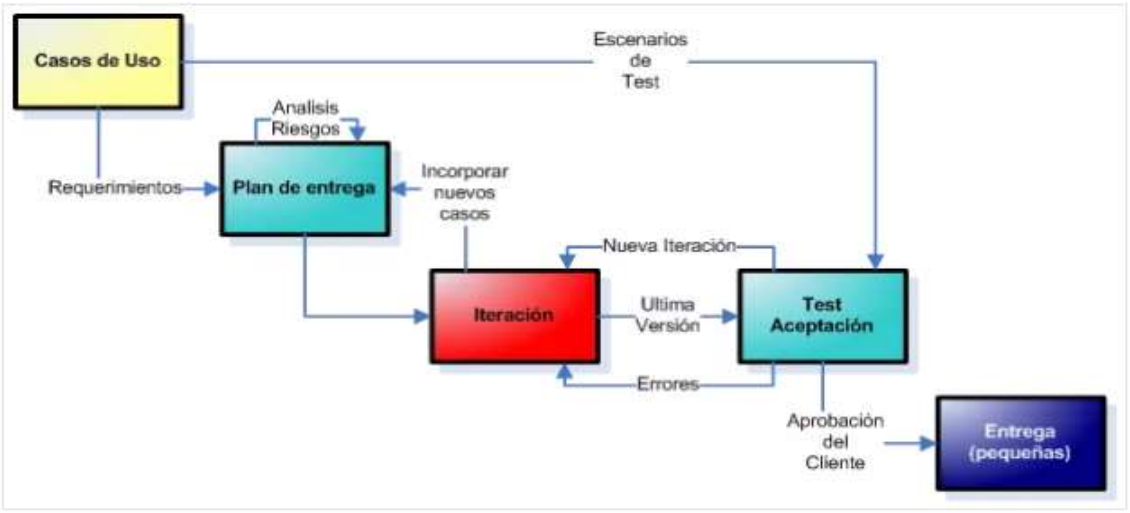

Figura 14: Programación extrema[16]

Bases de la programación extrema

La programación extrema se construye sobre cinco valores, estos últimos inciden en la colaboración entre los miembros de un equipo dentro de un proyecto informático.

[16] Programación extrema-Proyecto Guerrilla- sobre sistemas, operaciones y la cadena de suministro...: http://proyectosguerrilla.com/blog/2013/06/programacion-extrema-xp/

Figura 15: Valores de la Programación extrema[17]

1- Comunicación

"*Everyone is part of the team and we communicate face to face daily. We will work together on everything from requirements to code. We will create the best solution to our problem that we can together.*"

(http://www.extremeprogramming.org/values.html)

En las metodologías tradicionales, las comunicación de las necesidades de los clientes o de los usuarios son mediante envío de documentación donde figuran los requisitos o incluso los diseño funcionales, en XP esta manera de hacer las cosas cambia, se inclina más por la comunicación directa con el cliente o los usuarios, de manera a que la transferencia del conocimiento necesario para la realización del proyecto sea mediante reuniones, comunicación verbal o mediante correos electrónicos o puntos telefónicos.

2- Simplicidad

*"We will do what is needed and asked for, but no more. This will maximize the value created for the investment made to date. We will take small simple steps to our goal and mitigate failures as they happen. We will create something we are proud of and maintain it long term for reasonable costs." (*http://www.extremeprogramming.org/values.html*)*

En las metodologías tradicionales, se enfoca más en el resultado a largo plazo, en XP se focaliza en solucionar los requisitos iniciales y las funcionalidades o mejoras se van añadiendo más tarde mientras el equipo recibe más detalles sobre los requerimientos venideros.

Esta manera de trabajo hacer que se centre en lo que se esté desarrollando en este momento y dejando lo que se prevé en el futuro que pueda cambiar hasta que se asegure su especificación final.

[17]Los 5 valores de la programación extrema (XP) - La oficina de proyectos de informática http://www.pmoinformatica.com/2012/11/los-5-valores-de-la-programacion.html

Por lo tanto, una simplicidad en el desarrollo software haría que la comunicación sea mejor y más fluida para que se entiendan el equipo y el cliente esto implica que el producto final será de mejor calidad.

3- Retroalimentación (feedback)

"We will take every iteration commitment seriously by delivering working software. We demonstrate our software early and often then listen carefully and make any changes needed. We will talk about the project and adapt our process to it, not the other way around." *(*http://www.extremeprogramming.org/values.html*).*

La retroalimentación es el hecho de dar una opinión o una contestación a un hecho establecido: en este aspecto intervienen tres componentes: El cliente, el equipo y el sistema.

El cliente da un reporte al equipo de desarrollo después de ejecutar sus pruebas sobre el software entregado, estas pruebas certifican si el funcionamiento es el solicitado y que no haya ningún error, en el caso contrario se detallan los errores detectados contrastándolas con el funcionamiento deseado.

El equipo de desarrollo es cuando realiza una estimación del trabajo a efectuar después de recibir unas especificaciones por parte del cliente, una vez hecho, se le remite a este último para que lo valide.

El sistema, con la realización de pruebas, sean unitarias o de integración, los resultados obtenidos dan una imagen sobre cómo se comporta el sistema y los fallos detectados.

4- Coraje

*"We will tell the truth about progress and estimates. We don't document excuses for failure because we plan to succeed. We don't fear anything because no one ever works alone. We will adapt to changes whenever they happen"(*http://www.extremeprogramming.org/values.html*)*

De las prácticas de coraje resumiremos las más importantes:

- Hay que desarrollar (diseño e implementación) a diario, las dos tareas van de la mano no hay que priorizar el diseño sobre la implementación en código.
- La refactorización es una práctica muy aconsejada, siempre que sea necesario, no se deberían tener reservas al respecto.

- Hay que hacer revisiones e inspecciones periódicas del código para detectar partes donde se puede hacer una refactorización, haciendo que se pueda mantener y mejorar el código fácilmente.
- Siempre que es necesario, hay que decomisar partes obsoletas o complejas del código que se puedan rehacer de manera más clara y más sencillas sin preocuparse de los costes que pueda o haya acarreado el desarrollo de dichas partes.
- Tener una persistencia a la hora de enfrentarse a la resolución de problemas encontrados cuando se esté probando el desarrollo y no dejarse vencer por los mismos.

5- Respeto

*"Everyone gives and feels the respect they deserve as a valued team member. Everyone contributes value even if it's simply enthusiasm. Developers respect the expertise of the customers and vice versa. Management respects our right to accept responsibility and receive authority over our own work." (*http://www.extremeprogramming.org/values.html*).*

Todos los miembros del equipo implicado en el proyecto deben de tener un respeto mutuo por sí mismos y por su trabajo y él de todos.
Un miembro del equipo no debe sentirse menos valorado o ignorado, la opinión de todos y cada uno es respetada y tenida en cuenta ya que esto promueve un ambiente de alta lealtad y compromiso con el proyecto.
Siempre se tiene que perseguir la mejor calidad, en el código, en el diseño y en las pruebas. No se debe de subir un código que haga fallar las pruebas unitarias de lo que ya han entregado otros miembros del equipo.

Practicas técnicas de XP

Como detallamos en la metodología Scrum los 12 principios de Scrum, la metodología XP, tiene establecidas 12 practicas técnicas que son fáciles de asimilar y vierten en el sentido de un proyecto bien ejecutado.
A continuación, se describen las 12 practicas:

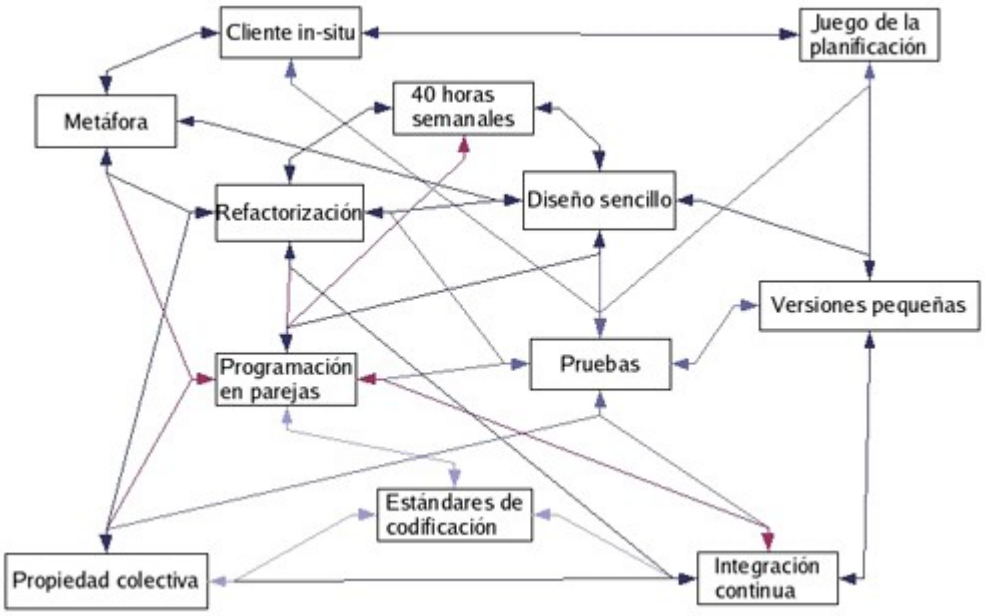

Figura 16: Interacción entre las prácticas de XP[18]

1- Client in-situ (on-site customer)

"Siempre es recomendable contar con más de una persona (por parte del cliente), asignada a trabajar de forma permanente con el equipo."

Una comunicación directa entre el equipo y el cliente tiene que ser necesariamente fluida para que se consigan unos avances rápidos en el proyecto:

- En cuanto surjan dudas sobre las especificaciones, hay que compartirlas y discutirlas con el cliente en el momento, así será más fácil resolverlas.
- Según marcha el proyecto, se establecerán las prioridades a resolver, la resolución de los conflictos que han surgido en la fase de análisis.
- El equipo debe tener expertos en el campo de lo que se quiere producir, estos expertos pueden ser los mismos usuarios del futuro software para asegurar que se esté siguiendo el desarrollo de las funcionalidades pedidas.
- El equipo debe tener personas con los detalles necesarios sobre el software y el negocio al fin de ser la autoridad competente a la hora de

[18] Metodologías ágiles para el desarrollo de software: extreme Programming (XP) http://www.cyta.com.ar/ta0502/v5n2a1.htm

actuar y tomar decisiones que impliquen una mejora del recorrido del desarrollo.

2- SEMANA DE 40 HORAS (40 HOUR WEEK)

- Un equipo no debe asumir responsabilidades en las que no tenga muchas experiencias para no tener que llevar un sobreesfuerzo y esto impactara negativamente en el curso del proyecto.
- Este punto muestra un contraste con las metodologías tradicionales que, con unas malas especificaciones, los equipos tienen que llevar un sobre esfuerzo para enderezar los desarrollos.

3- METAFORA (METAPHOR)

- Es una práctica muy aconsejable cuando surgen situaciones de no entendimiento entre los técnicos y los usuarios (que no necesariamente entienden los aspectos técnicos del proyecto).
  Se usa para presentar un símil que juega el papel de una referencia para comparar una situación de la vida real a un problema técnico en el proyecto.
- Una metáfora debe cumplir el propósito de identificar un paralelismo entre la vida real y el problema técnico planteado, en el caso de que esto no se cumpla, habrá que buscar otra metáfora que lo haga.
- En muchas ocasiones, es necesario emplear más de una metáfora para detallar un solo problema técnico, es más, se da el caso de que no se llegue a poder explicar, con ninguna metáfora, el problema y es cuando hay que echar mano de la creatividad de los desarrolladores para llegar a una explicación clara.

4- DISENO SIMPLE (SIMPLE DESIGN)

- Es una práctica derivada de un principio técnico del desarrollo software con las siglas **KISS**: “Keep it simple, stupid!” del castellano: “¡Mantenlo simple, estúpido!”
- Este principio camina en el sentido de que hay que hacer un diseño simple, de mantenimiento, entendimiento y refactorización sencillos – es la ley del esfuerzo mínimo: hacer lo importante tan entendible como sea posible.

5- REFACTORIZACION (REFACTORING)

- La refactorización del código escrito cada vez que se quiere añadir una nueva funcionalidad es una actividad bastante buscada y aconsejada, se alcanza con ello una mayor cohesión y un bajo acoplamiento.
- Un desarrollador, debe hacer regularmente una refactorización para limpiar el código, simplificar funcionalidades y obtener una mantenibilidad de muy alto grado.

6- PROGRAMACION DE A PARES (PAIR POGRAMMING)

- Esta práctica personifica la expresión coloquial de *"Cuatro ojos ven mejor que dos"*, esto viene a explicar la práctica de programar en par, es decir: dos programadores, uno implementa el código y el otro lo va revisando, de esta manera un programador puede tener varias funciones en el proyecto, puede desarrollar, programar y revisar código según las necesidades que se planteen.

7- ENTREGAS CORTAS (SHORT RELEASES)

- Al igual que en Scrum, se busca realizar una serie de pequeñas entregas funcionales del software, en cada sprint o iteración, estas entregas se irán incrementando con pequeñas funcionalidades.
- De esta forma, se evita volver a testear unitariamente lo que ya se habías testeado en las entregas previas y por lo tanto proporciona un control sobre el comportamiento del software.

8- TESTING

- Extreme programming identifica tres tipos de tests:
  - Tests unitarios

    Son tests que se incluyen dentro de la práctica de TDD: Test Driven Developpement o desarrollo guiado por tests. Es el hecho de probar cada método por separado para controlar el buen funcionamiento.

  - Tests de aceptación

    Estos tests, son más orientados a verificar que la funcionalidad perseguida es la obtenida en cada entrega.

  - Tests de integración

Son tests que engloban todos los tests o todas las funcionalidades de inicio a fin, lo que se denomina como tests End-to-End.

9- CODIGO ESTANDAR (CODING STANDARDS)

- Como ejemplo del lenguaje de programación Java, hay unas normas de escritura de código y de desarrollo según qué patrones de diseño.
- De esta manera, el código se hace más legible y mantenible, también fácil de transmitir a otros programadores para hacerlo evolucionar.
- Hay una serie de herramientas que ayudan a seguir estas normas sin tener que aprendérselas de memoria, **Sonar**, **Checkstyle**, **Cobertura**…etc. Estas herramientas se basan en unos ficheros de reglas automáticas que se van verificando mientras se va escribiendo el código, señalando aquellas que se han infringido y como hay que corregirlas.

10-PROPIEDAD COLECTIVA (COLLECTIVE OWNERSHIP)

- El código tiene que ser compartido entre todos y no tiene que ser una propiedad del programador quien lo haya hecho, si surge un problema, no habrá impedimento en corregirlo, ni nadie podrá entenderlo.

11-INTEGRACION CONTINUA (CONTINUOUS INTEGRACION).

- Actualmente en el Desarrollo software se aplican procesos de generación de ejecutables que se basan en la integración continua, se utilizan almacenamiento de código (svn, git, mercurial), motores de compilación (Jenkins, Hudson) y de validación de calidad de código (Sonar, SonarQube).
- La integración continua, asegura que todos los tests se ejecutan correctamente y no fallan, el código tiene una calidad según los estándares y la generación del ejecutable o entregable se hace de manera correcta.

12-JUEGO DE PLANIFICACION (PLANNING GAME)

- De un gran parecido con scrum en lo que se trata de la planificación, en extreme programming se panifica al principio de un sprint que suele durar de 1 a 4 semanas.

- El cliente anuncia las funcionalidades que quiere desarrollar como historias de usuario (user strory) y se las transmite al equipo de desarrollo
- El equipo de desarrollo estima cada historia y establece los recursos disponibles para ejecutarla.
- El cliente establece el orden de prioridad dentro de las historias des usuario para empezar los trabajos.

En resumen, la metodología Extreme programming, es una pequeña implementación del marco de trabajo Scrum, hay algunas similitudes, pero diferencias o deficiencias.
Las herramientas utilizadas por scrum son ausentes en XP, pero eso no la hace menos efectiva, sino que se emplearía en el contexto que le sea necesaria, sin necesidades de pizarras o reuniones diarias para el seguimiento los cuales son elementos esenciales en Scrum.

La tabla 2 es un resumen de las ventajas y limitaciones de la programación eXtrema:

Programación Extrema (XP)	
Ventajas	Limitaciones
El proceso donde es aplicada resulta muy organizado	Se recomienda su uso en los proyectos pequeños.
Fomenta la comunicación entre los clientes y los desarrolladores.	Cuando hay fallos, las correcciones pueden ser muy costosas
Permite un gran ahorro en recursos económicos.	Los principios XP se tienen que seguir por el proceso de manera muy exhaustiva

El cliente es quien establece las prioridades	No siempre es más fácil que un desarrollo clásico.
Las pruebas son parte continua del proceso donde se realizan en cada momento	

Tabla 2: Ventajas y desventajas de eXtreme programming

3.2.3 Kanban

"*Kanban ha ido ganando popularidad durante las últimas décadas. Nació para aplicarse a los procesos de fabricación y con el tiempo se convirtió en un territorio reclamado por los desarrolladores de software. Últimamente, ha empezado a ser reconocido por las entidades empresariales de diferentes ámbitos*." (Kanban para principiantes: https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kan).

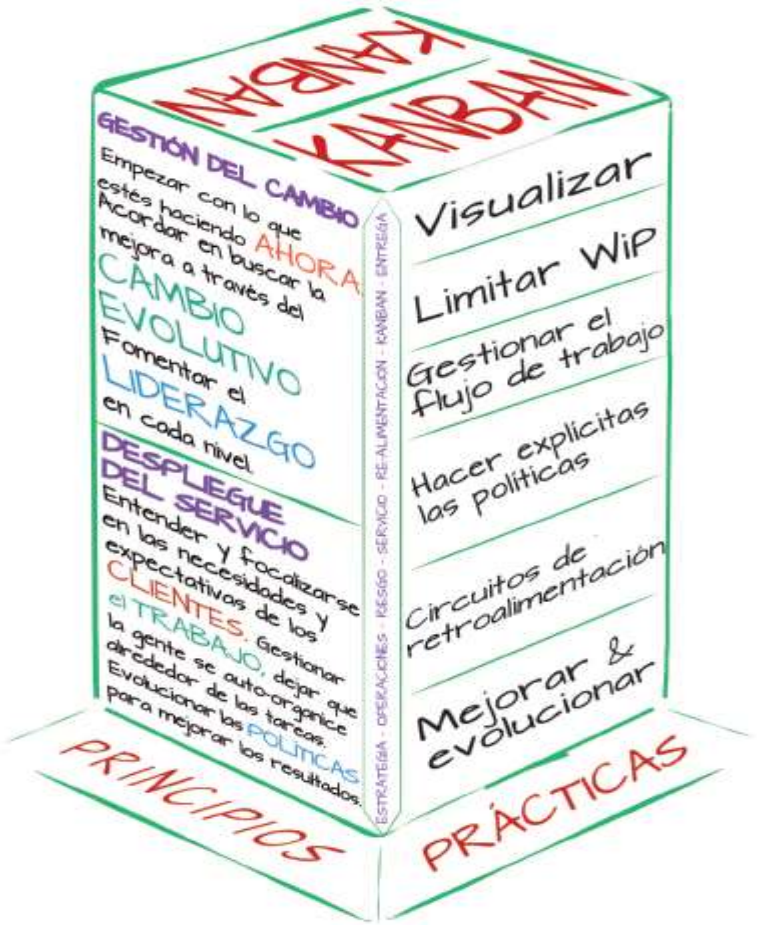

Figura 17: Elementos de Kanban[19]

La metodología Kanban es para para establecer, administrar y mejorar herramientas de desarrollo y entrega de trabajos intelectuales o de la

[19] Essential Kanban: David J. Anderson and Andy Carmichael PhD, FBCS

información, como las profesiones donde hay una creación de un producto final físico o intelectual fruto de genio o creatividad.

La palabra Kanban procede del japonés y se puede interpretar como “tarjeta visual” dónde aparecen las tareas definidas en un proceso concreto con tres posibles estados: “To Do”,”Doing”,”Done” lo que viene a ser “Por hacer”, “En curso” y “hecho”.

Figura 18: Kanban en japonés[20]

La metodología Kanban, de las metodologías ágiles es la más nueva, descubierta por el **Ingeniero Taiichi Ohno** de la empresa de fabricación de automóviles Toyota ™.

La metodología aterrizó en el mundo de la fabricación de software en el año 2004 de la mano de Microsoft ™, con esta introducción de esta metodología en la empresa generó resultados muy buenos y prometedores.

La metodología Kanban se basa en el concepto de la producción necesaria con respecto a la demanda de un sector o cliente.

La manera de trabajo de Kanban es establecer recursos justos para generar la misma cantidad de producto, el procedimiento es que no tiene que haber tarjetas en exceso, sino que un número limitado para no saturar el sistema de producción.

Reglas de Kanban

Kanban se basa en tres reglas para la construcción de un mundo practico para la implementación de la metodología en un desarrollo de software:

- Mostrar el proceso.

[20] Símbolo japonés de Kanban: https://soka.gitlab.io/TallerTrello/doc/chapter-kanban/01-intro/intro-kanban.html

- Limitar el trabajo que está en curso (WIP).
- Optimizar del flujo de trabajo.

1- Mostrar el proceso

Para hacer constante el proceso de trabajo en vigor en la organización, hay que hacer visibles los ítems de trabajo que lo hagan visible. Esta muestra, se organiza dentro de tableros físicos, con diferencia de columnas respecto a las que aparecen en Scrum.

Para Kanban la existencia de un tablero físico llevando las distintas tarjetas que representan las tareas que componen el proceso, es fundamental para visualizar la capacidad productiva del equipo de desarrollo.

Para ello, se tienen que definir:

- **Punto de partida y llegada de la visibilidad**

Establecer menos visibilidad en las tareas donde el equipo de desarrollo tiene influencia para realizar cambios. ¡Con esta limitación se implanta la manera de interactuar los procesos entre sí, como empieza uno cuando el otro termina!

- **Tipos de elementos de trabajo**

Cuál va a ser la naturaleza de cada tarea identificada en una tarjeta, esto es si va a ser un evolutivo, bug o incidencia…etc.

- **Diseño de las tarjetas representativas de las tareas**

Qué información debe contener cada tarjeta, generalmente tiene que ser la información sobre quien pertenece la tarea, los plazos de realización, el coste en días y también su naturaleza así que su título con una breve descripción comentando el motivo de su creación.

En este capítulo, hay que hablar de los tableros Kanban ya que son realmente el motor de esta metodología y su punto central de toda la actividad.

Como Scrum, Kanban tiene tablero físico o alguno que este mediante una herramienta digital de seguimiento de desarrollos informáticos.

La diferencia es que, en Kanban, se pone una columna por cada fase del desarrollo, partiendo del Análisis hasta la entrega final, pasando por la codificación, testing y correcciones de errores. Cada columna a su vez puede dividirse en dos: En curso y terminado, en algunos casos aparece una columna intermedia con el estado de Pendiente, esto viene a ser que hay alguna tarea que está pendiente de alguna acción de terceros.

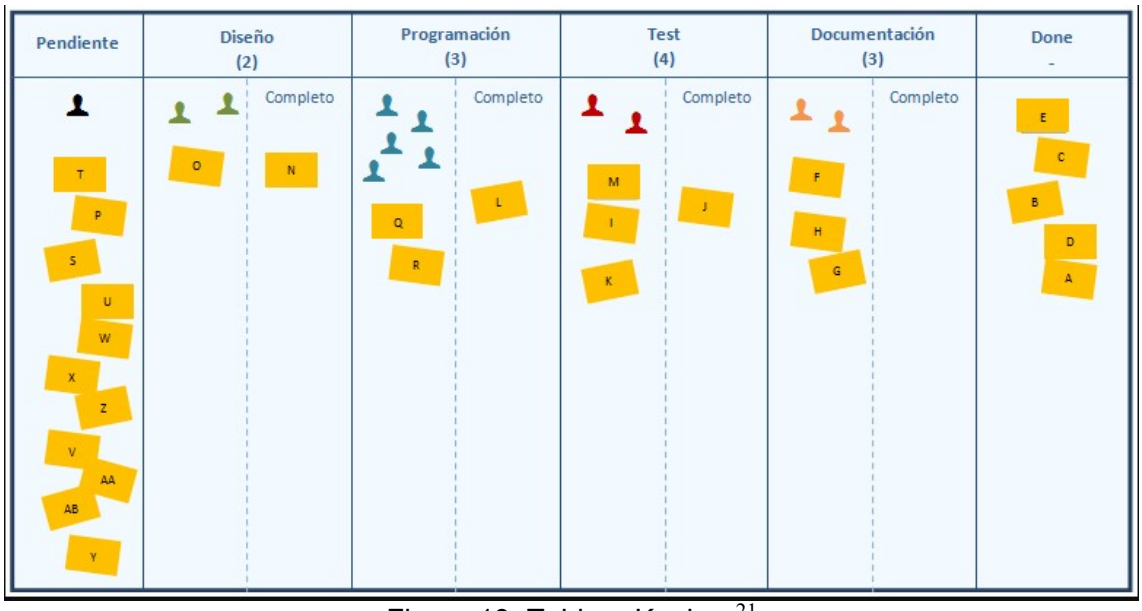

Figura 19: Tablero Kanban[21]

2- Limitar el trabajo que está en curso (WIP).

En Kanban, el límite de trabajo que se pueda llevar en un momento dado está dado por el WIP: Work in progress. Consiste en establecer cuantos elementos se podrían ejecutar o desarrollar en un mismo proceso.

El límite WIP permite de cierta manera ver cuando un proceso incidirá con otro provocando una especie de atasco, esto es debido a que uno de los procesos puede estar bloqueado en una o más tareas, de este modo se podría estudiar la manera de remediarlo para seguir avanzando en el proceso.

El hecho de poner un cupo para las tareas que se puedan lanzar en el mismo momento da un espacio de tiempo que se resuelvan las posibles complicaciones o errores que puedan surgir a lo largo del proceso.

3- Optimizar el flujo de trabajo

En Kanban, el flujo de trabajo tiene que ser lo más optimo posible con el objetivo de llegar a un proceso estable y previsible, para las necesidades del proyecto, cuando una tarea entra en una fase de bloqueo, todo el equipo tiene que hacerla avanzar para que los otros procesos en espera o en curos, no entren a su vez en un bloqueo.

No hay reglas establecidas para el control del flujo de trabajo, pero se pueden estudiar distintas métricas para contener estos elementos, control sobre la cola

[21] managementplaza.es: tablero Kanban.

de entrada de tareas, tener en cuenta mejoras que puedan aportar modificaciones beneficiosas sobre el proceso de plusvalía.

En la tabla 3, se resumen las ventajas y limitaciones de la metodología Kanban:

Kanban	
Ventajas	Limitaciones
Medir el rendimiento a través de un análisis exhaustivo de las estimaciones de las tareas, esto permite una gran flexibilidad frente a los cambios	Ninguna previsión sobre cómo resolver o absorber un gran volumen repentino de trabajo.
control del flujo de trabajo mediante el tablero, puede ser físico o digital.	Como su origen es japones, no tuvo una implantación optima en los países occidentales debido a las diferencias culturales
Distribución visual de las tareas con tarjetas, cada miembro del equipo sabe qué tarea tiene y por donde va o tiene que ir	Solo funciona en un contexto de Just intime o JIT.
Es una metodología fácil de usar y asimilar.	
Es una metodología muy visual, tener una idea genera de cómo va el proyecto es muy fácil solo con observar el tablero.	

Tabla 3: Ventajas y desventajas de la metodología Kanban

3.2.4 Comparativa entre metodologías

En este capítulo, será una manera de establecer una comparativa entre las metodologías estudiadas en este TFM, vamos a comparar algunos aspectos principales como la configuración de los equipos, la metodología de trabajo, seguimiento de las tareas y la priorización:

	Scrum	Kanban	XP
Equipos	En el equipo Scrum cada uno tiene un rol, generalmente se establecen cada vez que se forme un scrum team	No se definen roles de los equipos, cada uno tiene responsabilidades de sus tareas, pero no un rol concreto se tiene a que los equipos se "autogestionen".	No hay definición expresa de roles dentro de los equipos ya que se centra más en cómo hacer las cosas y no en qué cosas se van a hacer.
Seguimiento	El seguimiento se hace en el tablero igual que en Kanban, pero el nombre y número de columnas puede variar	El tablero tiene un seguimiento de las tareas según los tres conceptos de: En espera, en progreso y finalizado.	El seguimiento y las asignaciones se basan en la comunicación oral o por correo, no hay una herramienta parecida al tablero en Kanban y Scrum
Tareas	La prioridad se establece según como este definida el backlog del Product Owner, los cambios se pasan al siguiente sprint nunca en el que está en curso.	La prioridad de las tareas se hace según las necesidades del negocio, nunca se empieza una tarea antes de terminar la anterior.	Se aceptan cambios en la prioridad de las tareas dentro de la misma iteración,
Proceso de trabajo	Trabaja en iteraciones cuya duración depende del tamaño del equipo y es una duración cerrada, hay reuniones diarias y se utiliza el mismo tablero de Kanban	Las entregas se hacen en iteraciones, pero sin tener fechas cerradas y hay flexibilidad ante los cambios.	Las tareas se van asignando y se tienen que terminar en el tiempo acordado con el cliente, hay una flexibilidad frente a los cambios y estos se puede aplicar en cualquier momento.

Tabla 4: Resumen comparativo entre Scrum-Kanban-eXtreme Programming

4 RESOLUCIÓN DE LA PROBLEMÁTICA

En este capítulo se procederá a definir la metodología que se va a seguir para corregir el procedimiento que se siguió con la metodología clásica, en este caso se ha optado por Scrum, por motivos de experiencia profesional en esta metodología, por su adecuación con las necesidades del cliente, por la segmentación de las tareas que las vuelve más fáciles de manejar, por el tamaño reducido del equipo (**3 personas**) y por la corta de duración que se va a planificar para el desarrollo (**15 días**) para cuadrarlo con 3 sprints.

El proyecto es llamado **MaestroToSvn**: para pasar de gestionar fuentes en un gestor llamado Maestro a uno más actual y gratuito como SVN.

Se ha decidido realizar el proyecto en 2 sprints de 1 semana el primero y de 2 el segundo, con una demostración de cada incremento el último día del sprint y una reunión de retrospectiva además de la planificación del siguiente sprint.

En un primer lugar habría que definir la herramienta de seguimiento que se va a usar en este caso, se ha optado por usar Trello (https://trello.com/)

Figura 20: Pagina inicio de Trello.com[22]

Esta herramienta es gratuita para su uso en la planificación de proyecto y permite crear tableros que contienen las tareas que se van a ejecutar en cada iteración (Sprint) además de la planificación y lo que se tiene que hacer en cada una.

El tablero es accesible vía este enlace: https://trello.com/invite/b/vKN0L2EE/eddedbdd54f64a0999de8177fedfb2e3/maestrotosvn

Se puede registrar con la cuenta Google o por registro ordinario de usuario contraseña para poder acceder.

- Definición del tablero: Sprint 1

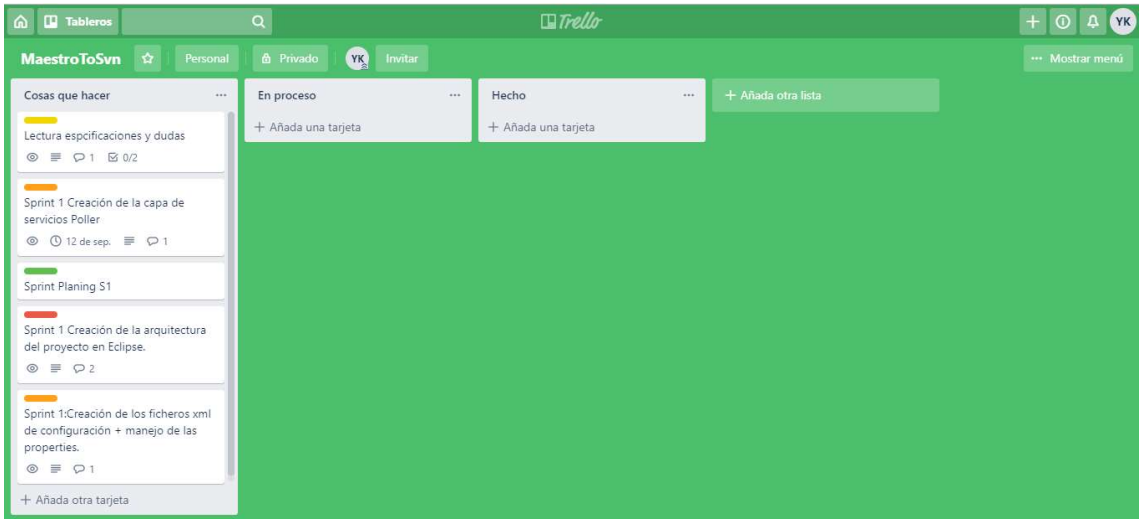

Figura 21: Tablero con las tareas del sprint 1

[22] www.trello.com

En este tablero se han definido las tareas del Sprint uno con sus estimaciones y que se tienen que realizar dentro de la semana dedicada a ese sprint.
El ultimo día es dedicado a la demostración (para las pruebas de aceptación del cliente), la reunión de retrospectiva del sprint y a la reunión de planificación del siguiente sprint.
Una vez definidas las tareas (en la columna “Cosas que hacer”) y su contenido desde la Product Backlog definida por el Product Owner, se afectan y se empiezan a desarrollar pasándolas a la columna “En proceso”.
Dado que el equipo es compuesto por 3 miembros, se le afecta una a cada uno, de manera que se podrán llevar las 3 tareas en paralelo, ya que las tareas de “Lectura de especificaciones…” y la de “Sprint plan” son en realidad tareas de todo el Scrum Team.
Ejemplo de ficha o tarjeta de una tarea:

Figura 22: Tarea de creación de un servicio dentro del Sprint 1

Una vez afectadas, se van moviendo las tareas a la otras columnas, cualquier impedimento o blocaje tiene que ser comunicado al Scrum Master, para que se tratar en prioridad: si es funcional, hablar con el Product Owner e insistirle que lo resuelva en cuanto antes, si es técnico: se busca en los otros miembros del equipo y si nadie puede aportar una solución se pide ayuda a la dirección de la organización para buscar experto en la materia y que pueda aportar su apoyo para resolver el problema.

Un aspecto muy importante en estos casos es trazar toda comunicación con los demás miembros del Scrum Team añadiéndolas como comentario en la misma tarjeta de la tarea (en caso de que sea por correo) o enviando un correo recapitulativo (en caso verbal o por vía telefónica).

Una vez los desarrollo y tests unitarios realizados se pasan las tareas a la columna “Hecho” y se tiene que hacer una demostración al cliente previa integración y entrega de la iteración.

En esta demostración se tiene que verificar que el incremento cumple con lo solicitado en este sprint, es decir: que el Poller detecte cualquier cambio en el directorio de lectura.

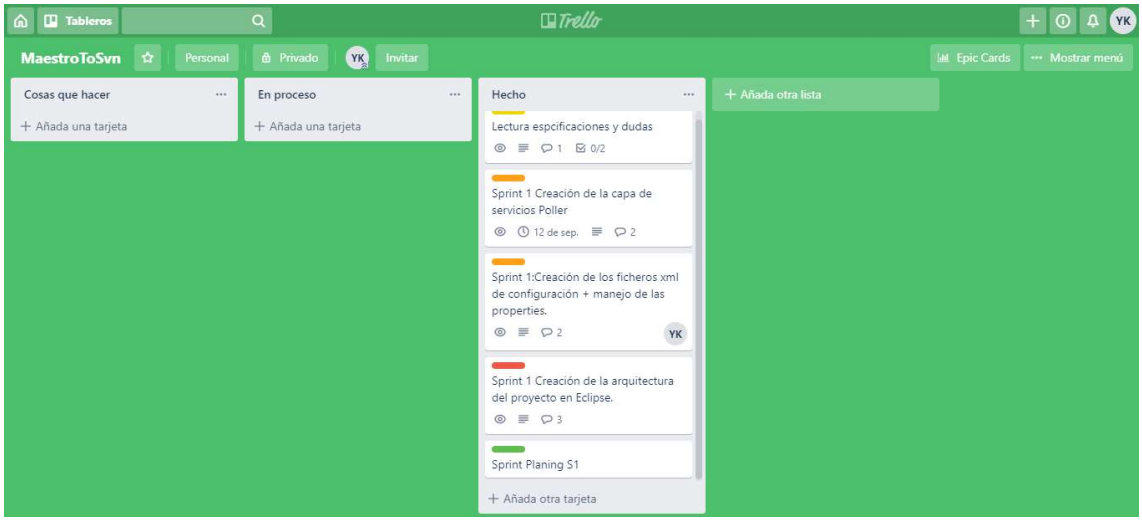

Figura 23: Finalización del sprint 1

Debido a que todo se dejó cerrado en la primera reunión de lectura de especificaciones y se aclararon todas las dudas, no hubo ningún cambio y la demostración del Sprint primero fue según lo planeado.

- Sprint 2

En este sprint se va a desarrollar la parte de servicios de intercambio con el gestor de configuración SVN de todos los ficheros presentes en el directorio y que sufran cambios.

La creación de las tareas se hace de forma análoga a lo que se ha hecho con las tareas del sprint 1.

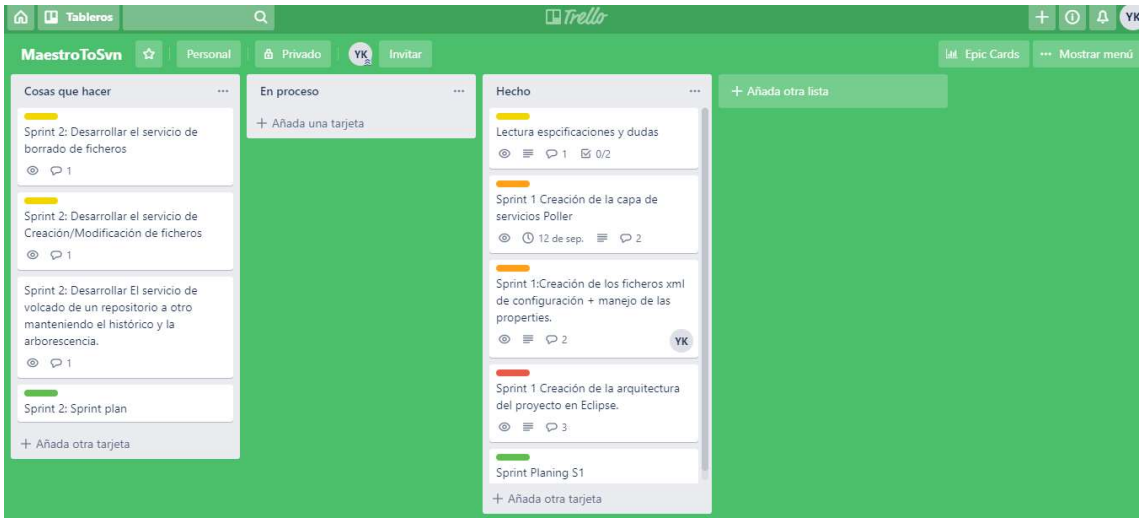

Figura 24: Creación del sprint 2

En este Sprint se desarrolla el grueso funcional de lo que se ha planificado desde la Product Backlog, se implementarán las tres operaciones de creación/modificación y borrado, también se desarrollará una operación que realiza el cambio de un repositorio a otro de las fuentes manteniendo el histórico de cada cambio.

Siguiendo la misma rutina, las tareas se afectan y se irán moviendo mientras se van realizando de la columna inicial a la de en proceso hasta llegar a la columna de “Hecho”, seguido por la validación dentro de la demostración y la entrega de la integración con el incremento del Sprint 1.

Después de finalizar cada uno de los Sprints, se realiza una reunión de retrospectiva para que cada miembro del Scrum Team, aporte lo que le pareció que se hizo de manera correcta y buena y lo que se podría mejorar dentro del proceso para tenerlo en cuenta en las siguientes iteraciones.

Planificación y seguimiento

El tablero está definido con tres columnas básicas: Cosas que hacer como el “To Do”, En proceso como “Doing” y Hecho como “Done”.

Los nombres se pueden personalizar según las necesidades del proyecto, se pueden añadir más etapas o columnas al tablero según como se quiera adaptar al proceso, pero Scrum desaconseja tener tantas columnas y generalmente es posible añadir columnas de etapas de test y de tareas pendientes, es decir cuando una parte del desarrollo representada por una tarjeta este en fase de test o cuando esté pendiente de algún otro procesos, tarea o retorno de información por parte del Product Owner o cliente.

Esto es posible con la opción de : “Añada otra lista” presente en el mismo tablero.

Cada tarjeta representa una tarea, en el momento de la creación, hay que informar una serie de campos pertenecientes a bloques.

En el tablero del proyecto, una columna se compone de un número concreto de etiquetas que representa una tarea cada una, a la hora de crear una tarjeta, la ventana de creación se compone de varios bloques:

- Identificación:

Figura 25: bloque identificación de tareas

Se compone de

- El título de la tarea.
- La columna está presente.
- El campo ETIQUETAS que sirve para poner un tag a la tarea sea para distinguirla de las demás o para asociarla a otra tarea e incluso definir su prioridad según el color: Rojo es crítico, naranja o amarillo es mediano y verde es bajo, por ejemplo.
  Las etiquetas se pueden personalizar por color y título.

- Descripción:

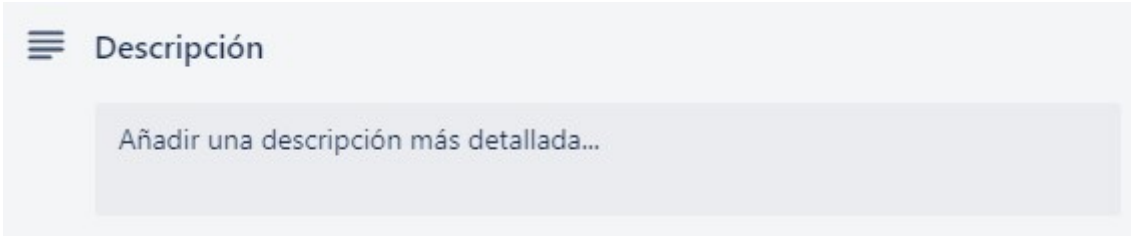

Figura 26: Descripción de la tarea de la tarjeta creada.

Este campo sirve para describir en qué consiste la tarea que vamos a crear para que cualquier miembro del equipo que vaya a visualizarla sepa de qué se trata.

- Actividad

Figura 27: Actividad de una tarjeta

La actividad de la tarjeta lleva el conteo o el detalle de todos los cambios que ha sufrido una tarjeta, cada cambio se tiene que indicar en el comentario y se registra con el nombre del usuario quien lo haya hecho y la hora en la que se registró ese cambio.

Un cambio puede ser en la estimación, la afectación de la tarea, especificación o en el cambio de estado de una columna a otra.

- Añadir elementos:

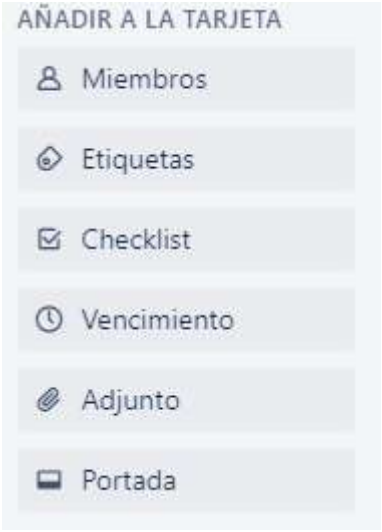

Figura 28: Elementos a añadir a una tarjeta

- **Miembros**: son los miembros del proyecto a los que se puede añadir a la tarjeta si intervienen en la tarea.
- **Etiquetas**: Añadir una etiqueta a la tarjeta.
- **Checklist**: sirve para añadir una lista de subtareas a la tarea, pueden estar afectadas a otros miembros del equipo, que se deberían de ejecutar antes de terminar la tarea principal.
- **Vencimiento**: definir los plazos para la finalización de la tarea.
- **Adjunto**: Con esta opción se pueden adjuntar documentos, sean documentos funcionales, documentos de preguntas y respuestas o comunicaciones con el cliente o con el Product Owner.
- **Portada**: Sirve para poner un logo (una imagen) a la tarea.

- Acciones:

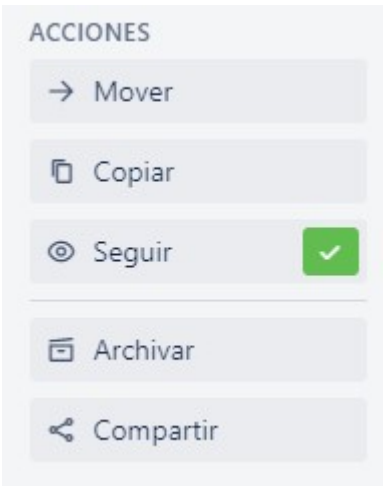

Figura 29: Acciones sobre las tarjetas

- **Mover**: Sirve para cambiar a la tarjeta a su siguiente estado.
- **Copiar**: Para crear una nueva tarjeta idéntica a la anterior.

- **Seguir**: Al hacer clic en esta opción hace que cualquier cambio sobre esta tarea se le avise al correo del usuario quien lo haga.
- **Archivar**: Generalmente cuando se termina la tarea y se cierra, se puede archivar.
- **Compartir**: Es cuando se comparte con otro usuario del proyecto, aunque no esté afectado por la tarea de la tarjeta.

Lo que conseguimos con esta herramienta al implantar su uso en el proyecto, es una correcta **planificación** y un **seguimiento** preciso de las tareas e incluso una **comunicación** trazada en cada momento entre los intervinientes del proyecto.

Esto mejora de manera notable como se habían hecho esas fases cuando no se había implantado Scrum, realmente esas fases no estaban separadas, sino que se solapaban de una manera que hacía difícil llevar correctamente los inicios del proyecto.

Integración Continua

Es un proceso que asegura la buenas compilación y construcción del aplicativo para su posterior despliegue, para ello se utiliza una herramienta bastante extendida en el mercado: Jenkins (https://jenkins.io/) :

Figura 30: Pantalla de inicio de Jenkins

Es una herramienta gratuita, dónde se integran todos los proyectos que tenga la organización, se usa como compilador y constructor de las fuentes para crear el ejecutable, se conecta a un gestor de configuración (Svn o Git) et se basa en un fichero descriptor del proyecto, ese fichero se llama **pom** es un fichero XML y es una definición de muchos parámetros de construcción, generación y librerías o dependencias que se vayan a utilizar dentro del proyecto.

El fichero pom.xml es proprio a un marco de trabajo llamada Maven **(**https://maven.apache.org/). Es un proyecto open source de la comunidad apache (http://apache.org/), creado para simplificar la inclusión de las librerías dentro de un proyecto de desarrollo Java de manera automática.

Un ejemplo de pom.xml:

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.redoute</groupId>
    <artifactId>mvsTransfertsvn</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <packaging>war</packaging>

    <name>mvsTransfertsvn</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <!-- <name>TransfertMvsToSvn</name> -->
    <build>
        <finalName>TransfertMvsToSvn</finalName>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>1.6</source>
                    <target>1.6</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

Figura 31: Ejemplo fichero descriptor pom.xml

A continuación una descripción breve de cómo funciona Jenkins:

Figura 32: pantalla creación de una tarea en Jenkins

La figura siguiente muestra el tipo de proyectos disponibles, se elegirá un proyecto simple:

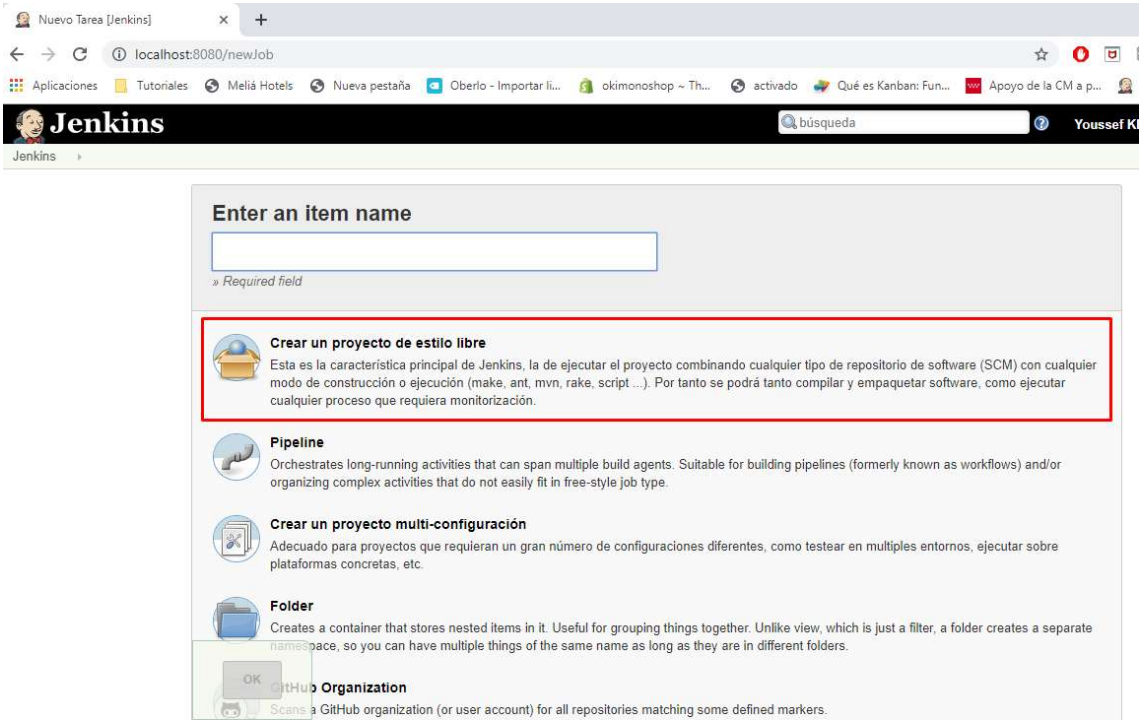

Figura 33: Pantalla de elección de tipo de proyecto

Con OK aparece la pantalla de configuración del proyecto:

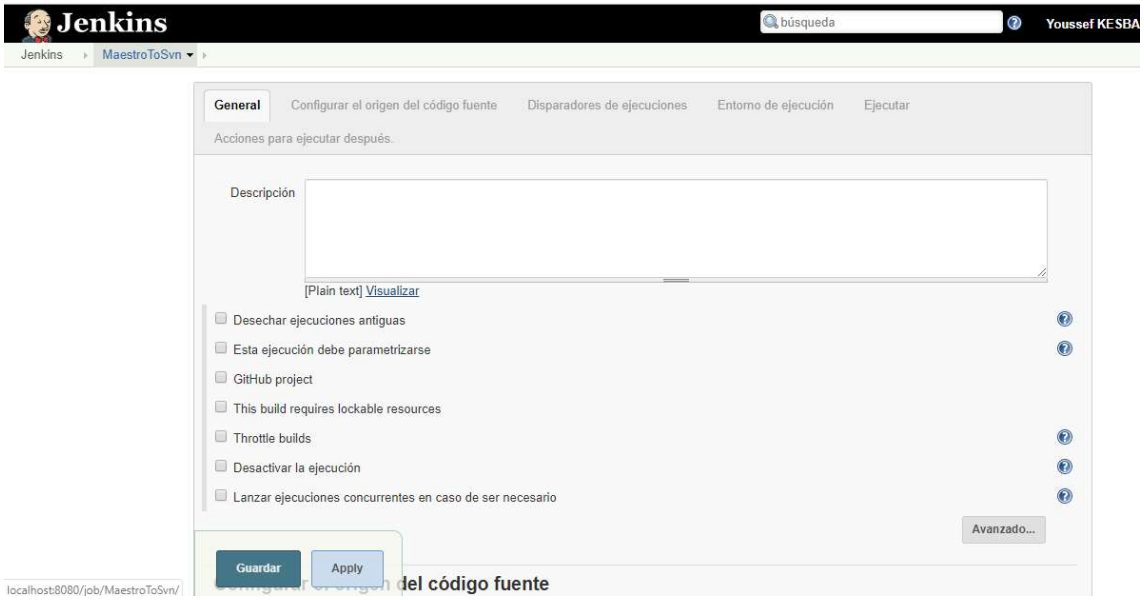

Figura 34: Opciones generales de creación de proyecto en Jenkins

La configuración tiene más opciones, donde viene el código fuente es decir si es Git o SVN, eligiendo una de las dos opciones hay que indicar la URL del repositorio con las credenciales para acceder a ellos automáticamente.

Figura 35: Opciones para la configuración de fuentes y ejecuciones

La opción de disparadores de ejecuciones, indica bajo qué condiciones se lanzaría la complicación de nuestro proyecto, o por algún script automático o condicionado por la compilación correcta de otro proyecto, o ejecutar de manera

periódica indicando el intervalo del tiempo, o después de cada vez que se suba un código en el repositorio.

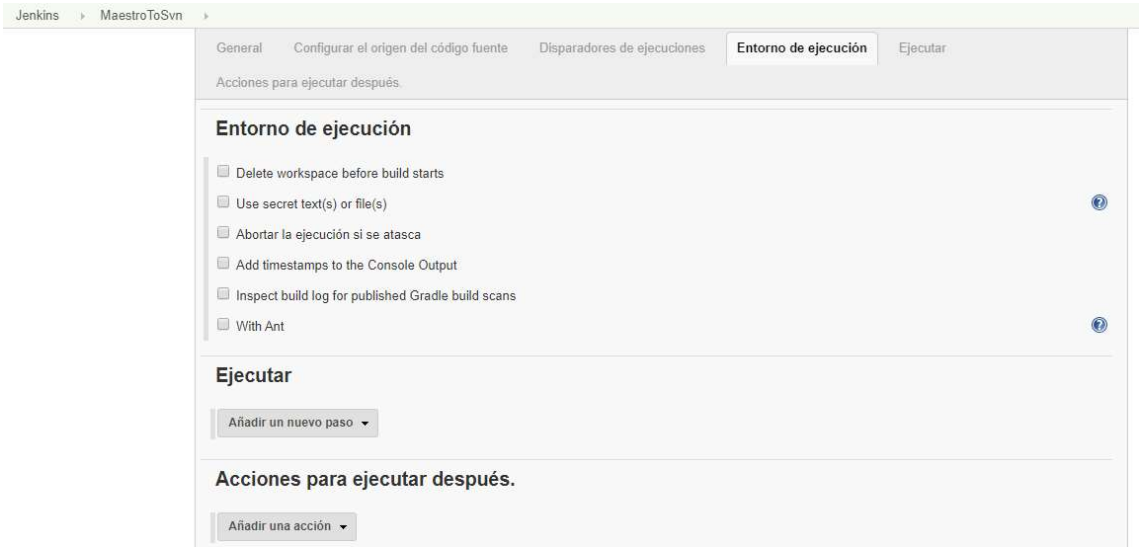

Figura 36: Más opciones de configuración

- **Entorno de ejecución**: Son las opciones para indicar qué se tiene que hacer en algunas circunstancias de ejecución.
- **Ejecutar**: Es cuando hay que lanzar una acción antes o mientras se ejecuta el proyecto.
- **Acciones para ejecutar después**: Son acciones ofrecidas por Jenkins o por scripts de usuario, para que se ejecuten cada vez que se termine la compilación y construcción correctamente.

Después de esto el proyecto ya está creado:

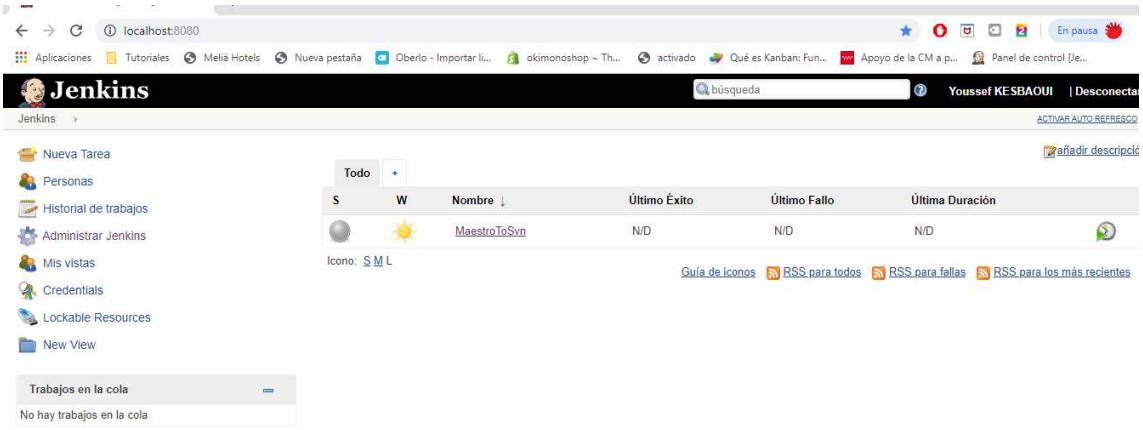

Figura 37: Proyecto creado en el panel principal de proyectos

Generalmente la complicación de los proyectos se lanza con comandos y opciones Maven empezando con **mvn** indicando si se lanzan los tests unitarios o no *"**-DskipTests = true/false**"*.

La mejora que se obtiene empleando esta herramienta, es la automatización de la inclusión de las librerías en el proyecto, la compilación automática y el aseguramiento de que las funcionalidades se testean correctamente y de manera automática con su integración en la tarea/proyecto de construcción.

Todo lo anterior no se hacía cuando no se aplicó la metodología ágil Scrum al proyecto sujeto de la problemática definida en el capítulo 2.

Después de rehacer el proyecto siguiendo una metodología ágil, se pueden comparar los resultados en la siguiente tabla:

	Desarrollo clásico	Desarrollo con Scrum
Reuniones e intercambios y validaciones	Más de 4 días	4 horas
Desarrollos	30 días	15 días
Resultados frente al cliente	Proyecto paralizado	Proyecto llevado a cabo

Tabla 5: Resultados de la resolución de la problemática.

Vemos que el tiempo de desarrollo y el tiempo total que en el que se ha llevado a cabo el desarrollo del aplicativo, se redujo a la mitad en comparación con la metodología clásica, es una proporción bastante atractiva cuando este tiempo se traduce en costes y en términos económicos.

Supone una reducción bastante significante para un cliente, de hecho, también para la consultora o la organización quien desarrolla el producto desde un punto de vista estratégico: un cliente que recibe su producto terminado y funcional en un tiempo menor y con mejor calidad, es un cliente que es fidelizado, esto aseguraría más clientes y más trabajos además de una buena reputación.

5 CONCLUSIONES Y LINEAS FUTURAS

5.1 CONCLUSIONES

A lo largo de este TFM, se ha introducido a los aspectos esenciales de algunas de las metodologías ágiles para ver sus beneficios y sus limitaciones más que desventajas.

El planteamiento de la problemática del capítulo 2 de un desarrollo que inicialmente no fue llevado a cabo con una metodología ágil y su posterior resolución mediante la elección de la metodología Scrum y aplicarla al mismo desarrollo desde cero.

Esta aplicación demostró que varios parámetros mejoraron, alcanzando, por lo tanto, los objetivos que se han establecido:

- Mejora del tiempo de desarrollo y entregas.
- Incluir la integración continua: gestión correcta de la configuración y del versionado aplicativo.
- Obligación de los Tests unitarios: blindar los desarrollos y entregar un código robusto que tendrá pocos incidentes en producción.
- Incluir la fase de revisión de código: código de calidad y según las normas, se traduce en facilidad de mantenimiento y evolución.

Se ve que después de la aplicación de Scrum para mejorar este desarrollo, implica muchas mejoras en todos los niveles: planificación, estimación, planificación, desarrollo, testeo, verificación y entrega. Esto supone una confirmación de que las metodologías ágiles aportan desde el punto de vista procesal y organizativo, herramientas y métodos muy eficientes y beneficiosos en un desarrollo, mejorando por lo tanto varios aspectos que son muy importantes para que un proceso software acabe con un producto fiable, de calidad y seguro.

Las metodologías ágiles con todo lo que las acompaña como herramientas y técnica, se han introducido con fuerza en los últimos años dentro del sector de la consultoría informática, aunque se pueden aplicar perfectamente a cualquier sector productivo y/o económico, muchas organizaciones las han adoptado como manera de trabajo y como herramienta de mejora y optimización de sus procesos, buscando con ello atraer a más clientes y a ganar una reputación muy positiva dentro del mercado.

Esto ha conllevado que se crearan nuevas categorías profesionales como Scrum Master, Product Owner, Agile Coach…etc. y orientar las profesiones informáticas, no solamente en la parte del desarrollo y diseño, sino también en

la metodología y el control y mejor de procesos, con ello aparecen muchas herramientas ampliando el concepto clásico de entorno de desarrollo de una herramienta de desarrollo a un conjunto de programas que combinados permiten que se realice un proceso de desarrollo informático desde la fase de planificación hasta la entrega en producción en un mismo hilo y sin necesidad de intervención de muchos actores.

Esta implicación de las empresas a introducir en sus procedimientos las metodologías ágiles, formando por ello a sus recursos humanos, lo que implica la aparición de profesionales expertos en la materia y que se organizan en comunidades estableciendo normas de cada metodología y especificando sus bases y como se aplican.

Por lo tanto, las empresas se han dado cuenta de que el beneficio en emplear dichas metodologías no es solamente un beneficio a largo plazo, sino que es un beneficio que hará que sean sólidas dentro del mercado, generando un valor añadido sobre su actividad y dándole una mayor capacidad de absorción de trabajos, también generando productos de calidad y que se harían reconocer por los propios clientes. Los mismos clientes empiezan a exigir a las consultoras que para optar a un concurso de oferta, que tengan implantada alguna metodología ágil de trabajo y que disponga de profesionales con la suficiente experiencia en el tema para sostener sus procesos.

Finalmente, las metodologías ágiles, tienen un impacto muy positivo en los procesos de cualquier empresa, puede que su aplicación sea tediosa al principio y que los recursos humanos se adapten a ellas lleven un poco de tiempo, pero es una inversión a corto plazo que merece la pena para ver que el beneficio que se va a ganar una vez la implantación es operativa y funcional, es muy cuantioso y muy interesante para la robustez de la organización frente a sus competidoras en un mercado cada vez más grande y agresivo.

5.2 LÍNEAS FUTURAS

Las líneas futuras que se plantean al final de este camino pueden ser varias, pero las más concretas son, no solo, líneas de investigación y mejora de las metodologías ya existentes aportando cambios sobre conceptos ya establecidos,

sino que intentar recopilar las ventajas de cada una de las metodologías o aspectos parecidos y hacer que se integren entre sí para generar una nueva metodología.

Esta metodología podría tener sus propias particularidades, sus propios roles y alcances.

Una metodología ágil de este tipo tendría un impacto muy positivo sobre los procesos dentro de la ingeniería del software optimizándolos aún más. Esto podría ser un detonante de la aparición de nuevas herramientas que mejoren las actuales tanto en la gestión de la configuración como en la integración continua.

Este campo del conocimiento es al final, un campo bastante amplio donde caben todo tipo de investigaciones, mejoras y aportes nuevos.

BIBLIOGRAFIA

[1]: Linchpinseo - https://linchpinseo.com/the-agile-method/

[2]: IEBS – Las metodologías ágiles más utilizadas y sus ventajas dentro de la empresa - https://www.iebschool.com/blog/que-son-metodologias-ágiles-agile-scrum/#metodologias_ágiles

[3]: Alvaro Ruiz de Mendarozqueta – Principios Ágiles: https://www.slideshare.net/AlvaroRuizdeMendaroz/principios-giles

[4]: Manifesto for Agile Software developpement: https://agilemanifesto.org/

[5]: OBI Business School - Descubre el Agile Project Management

[6]: Ken Schwaber y Jeff Sutherland - La Guía Definitiva de Scrum: Las Reglas del Juego.

[7]: Eugenia Bahit (2011-2012) – Scrum & eXtreme programming para programadores.

[8]: Agile Scrum Master Role and Responsibilities - http://knowledgeblob.com/agile/agile-scrum-master-role-and-responsibilities/

[9]: beagilemyfriend.com - atributos de Product Owner - https://www.beagilemyfriend.com/atributos-product-owner/

[10] Alaimo, Diego Martín - Proyectos ágiles con Scrum - flexibilidad, aprendizaje, innovación y colaboración en contextos complejos. - 1a ed. - Ciudad Autónoma de Buenos Aires: Kleer, 2013. EBook. ISBN 978-987-45158-1-0

[11] Advanced Development Methods Inc: Scrum Methodology – Incremental, Iterative Software Development from Agile Processes.

[12] Díaz Labrador, Marycarmen & Collazo Garcia, Antonio. (2013). La programación extrema. 10.13140/RG.2.2.29359.43687.

[13] PMOinformatica.com: Los 5 valores de la programación extrema (XP).

[14] Essential Kanban Condensed Copyright © 2016 David J. Anderson and Andy Carmichael PhD, FBCS ISBN 978-0-9845214-2-5 First digital version, 17 April 2016. This version 28 July 2016.

[15] Kanban: Explicación para principiantes: https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban/

[16] https://gbksoft.com/blog/scrum-vs-other-methodologies/#CdwPF Scrum vs Other Methodologies