UIMP Universidad Internacional Menéndez Pelayo

GOBIERNO DE ESPAÑA MINISTERIO DE POLÍTICA TERRITORIAL Y FUNCIÓN PÚBLICA INAP INSTITUTO NACIONAL DE ADMINISTRACIÓN PÚBLICA

Máster Universitario en Liderazgo y Dirección Pública

Curso académico: 2018 - 2019

Título del TFM: Plan de aplicación de metodologías ágiles de desarrollo de *software* en las administraciones públicas

Trabajo realizado por: Víctor Balbás Valderrábano

Dirigido por: Pablo de Amil

INDICE DE CONTENIDOS

INDICE DE TABLAS

INDICE DE ILUSTRACIONES

1 RESUMEN EJECUTIVO

En el presente trabajo de fin de máster, se analizarán los problemas asociados al desarrollo de *software* en las Administraciones Públicas, que son similares a los problemas que se ha encontrado la industria a lo largo de los años y que están principalmente relacionados con la metodología de desarrollo en cascada.

Como solución a esos problemas, se plantea la aplicación de un marco ágil de trabajo, como *Scrum*, en el desarrollo de *software*. Para llevar a cabo esta implantación se propone seguir los principios ágiles de inspección y adaptación al entorno. Por ello, en vez de dar un plan determinado y claro, se listarán un conjunto de medidas en el marco de la contratación, la metodología de desarrollo, la gestión de recursos humanos, la tecnología y en el marco organizativo. Estas medidas permitirán implantar ese marco de trabajo en las Administraciones Públicas. Para concretar un poco más, se hará una propuesta de cronograma para la aplicación, el primer año, de esas medidas; que deberá ser adaptado a la situación concreta de la unidad en la que se pretenda aplicar.

Además, se describirá como se ha aplicado *Scrum* en la elaboración del trabajo de fin de máster y se explicarán las condiciones que se deben cumplir para que se pueda aplicar un marco de trabajo ágil como *Scrum*, en cualquier contexto diferente del desarrollo de *software*.

In this master's dissertation, problems associated with software development in public administrations, will be analized. These problems are similar to those that development industry has suffered over the years and which are related to waterfall methodology which has been spreadly used.

As a solution to these problems, application of an agile framework, such as Scrum, is proposed. To carry out this implementation, the agile principles of inspection and adaptation must be followed. For this reason, a set of measures will be listed in the context of procurement, development methodology, human resources management, technology and organizational measures; instead of giving a specific and clear plan.

However, in order to deepen a little more, a timeline will be proposed for the first year, of application of these measures. This timeline must be adapted to the specific organization.

In addition, it will be described how Scrum has been applied in the elaboration of this master's dissertation and the conditions that must be fulfilled in order to apply an agile framework in any context different from software development.

2 PALABRAS CLAVE

software, metodologías ágiles, *Scrum*, tecnologías de la información y las comunicaciones (TIC), contratación.

3 INTRODUCCIÓN

El objetivo del presente Trabajo de Fin de Máster titulado "PLAN DE APLICACIÓN DE METODOLOGÍAS ÁGILES DE DESARROLLO DE *SOFTWARE* EN LAS ADMINISTRACIONES PÚBLICAS" es realizar un plan para la aplicación de los principios y bases de las metodologías ágiles de desarrollo de *software* en la SGTIC del MINCOTUR[1]. Debido a las características comunes de la Administración General del Estado, es posible que este plan pueda extenderse, con las adaptaciones precisas, a cualquier organismo público con características presupuestarias y de recursos humanos similares.

La Administración, como la sociedad, cada vez es más digital y por tanto dependiente de las nuevas tecnologías. En este contexto, el desarrollo de *software* a medida, que proporcione el necesario soporte a las competencias de las administraciones públicas, se ha convertido en un elemento esencial para que las Administraciones Públicas puedan cumplir con su cometido.

[1] Durante la elaboración del TFM el Ministerio ha cambiado de nombre, inicialmente se llamaba MINETAD (Ministerio de Energía, Turismo y Agenda Digital), pero en la reestructuración de mediados de 2018 cambió su nombre por MINCOTUR (Ministerio de Industria, Comercio y Turismo).

Por ello, el éxito de los proyectos informáticos se ha convertido en sinónimo de éxito de las Administraciones Públicas, por lo que resulta imprescindible abordar los problemas a los que se enfrentan tradicionalmente estos proyectos de desarrollo de *software*.

Una solución que ha aplicado la industria ha sido la implantación de un marco ágil de trabajo en el desarrollo de *software* puesto que se ha demostrado como mucho más eficiente y eficaz en la consecución de los objetivos de un proyecto de desarrollo de estas características.

4 ¿QUÉ SON LAS METODOLOGÍAS ÁGILES?

Las metodologías ágiles de desarrollo de *software*, se pueden definir como marcos[2] o buenas prácticas de trabajo y de toma de decisiones en relación al desarrollo del *software*. (Beck, y otros, 2001).

Dependiendo de las fuentes consultadas, hay quien no las considera metodologías, o las considera metodologías ligeras o "blandas".[3] A lo largo del trabajo se las denominará indistintamente metodologías ágiles o marcos de trabajo ágil.

Estas metodologías surgen en contraposición a los métodos que tienen un fuerte peso de la planificación y la documentación y que se han demostrado poco efectivos en los proyectos de desarrollo de *software*.

Kent Beck, uno de los fundadores del desarrollo ágil de *software*, define estas metodologías ágiles como un conjunto de métodos de desarrollo de *software* basados en un ciclo de vida iterativo e incremental, donde los requisitos y las soluciones evolucionan por medio de la colaboración de equipos multifuncionales y auto-organizados. Es decir, es un enfoque del desarrollo de *software* que permite la flexibilidad y la rápida respuesta a los cambios (Islam, 2013).

[2] Traducción del inglés *framework*

[3] Del inglés *mushy*

Aunque algunos métodos iterativos e incrementales surgieron ya por los años 60 y continuaron desarrollándose en los años 90 con el desarrollo de métodos como *RAD*[4], *Scrum*, *Crystal Clear* o *eXtreme Programming*, el lanzamiento y difusión definitiva de las mismas, se situa en el año 2001, a raíz de la firma y publicación del Manifiesto ágil, por parte de 17 desarrolladores de *software*, Kent Beck, James Grenning, Robert C. Martin, Mike Beedle, Jim Highsmith, Steve Mellor, Arie van Bennekum, Andrew Hunt, Ken Schwaber, Alistair Cockburn, Ron Jeffries, Jeff Sutherland, Ward Cunningham, Jon Kern, Dave Thomas, Martin Fowler y Brian Marick. (Beck, y otros, 2001)

4.1 Tipos de metodologías ágiles

A lo largo de la historia se han desarrollado diferentes tipos de metodologías ágiles, con sus características y particularidades. En la Tabla 1. Tipos de metodologías ágiles, se describen someramente las que se consideran más relevantes, junto con los puntos a los que da más importancia y su fundador (Ashmore & Runyan, 2014).

Metodología Ágil	Puntos importantes	Fundador/es
eXtreme Programming (XP)	Eficiencia, orientación al cliente, *feedback* y calidad.	Kent Beck
Scrum	Equipos, organización del trabajo	Jeff Sutherland y Ken Schwaber
Feature-Driven Development	Desarrollo iterativo de componentes orientados al usuario.	Jeff de Luca
***Dynamic Systems Development Method* (DSDM)**	Enfoque estructurado al desarrollo rápido. Colección de buenas prácticas.	DSDM Consortium

[4] *Rapid Application Development*

Lean Software Development	Eliminar desperdicio	Mary y Tom Poppendieck
Kanban Method	Visualizar y gestionar el flujo de trabajo. Desarrollo a demanda[5].	David J. Anderson
Crystal Family	Personas, comunicación y dinámicas organizativas.	Alistair Cockburn

Tabla 1. Tipos de metodologías ágiles

4.2 Características comunes, el Manifiesto ágil

En febrero de 2001, un conjunto de 17 profesionales relacionados con el desarrollo de *software* que habían tenido experiencias en diferentes metodologías ágiles, se reunieron en Snowbird (Utah), convocados por Kent Beck para debatir sobre diferentes modos de llevar a cabo desarrollo de *software*.

El resultado es el famoso Manifiesto ágil (Beck, y otros, 2001), cuyo texto principal, que ha sido traducido a más de 60 idiomas, dice lo siguiente:

"Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:

- *Individuos e interacciones sobre procesos y herramientas*
- *Software funcionando sobre documentación extensiva*
- *Colaboración con el cliente sobre negociación contractual*
- *Respuesta ante el cambio sobre seguir un plan*

Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda."

[5]Del inglés *Just-in-time*

Este texto, que remarca la filosofía de las metodologías ágiles, se complementa con 12 principios, que subyacen a este texto:

- Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de *software* que aporte valor.
- Damos la bienvenida a los requisitos cambiantes, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entregamos *software* que funciona frecuentemente, entre dos semanas y dos meses, con preferencia al período de tiempo más corto posible.
- Los responsables del negocio y los desarrolladores debemos trabajar juntos de forma cotidiana durante todo el proyecto.
- Los proyectos se construyen en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan y confiarles la ejecución del trabajo.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El *software* que funciona es la medida principal de avance.
- Los procesos ágiles promueven el desarrollo sostenido. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante desarrollo de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
- La simplicidad, es decir, el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños surgen de equipos auto-organizados.
- De forma periódica, el equipo reflexiona sobre cómo ser más efectivo para afinar y ajustar su comportamiento en consecuencia.

4.3 ¿Por qué *Scrum*?

Aunque todas las metodologías ágiles descritas llevan asociada una filosofía similar basada en los principios y valores del Manifiesto ágil, para concretar el plan y las

medidas a implantar se ha escogido *Scrum*, como marco de trabajo principal en el presente plan de mejora.

Las principales razones para tomar esta decisión son las siguientes[6]:

- Según la última encuesta mundial de uso de metodologías ágiles[7], *Scrum* es, con diferencia, la metodología ágil más usada (56% frente al 14% de la segunda más utilizada).
- La comunidad de desarrolladores y personal con conocimientos de *Scrum* es amplia, lo cual favorece la concurrencia en la contratación de los equipos de desarrollo.
- La disponibilidad de formación en *Scrum* es amplia, por lo que el personal de la Administración se puede formar fácilmente en este tipo de metodologías, favoreciendo por tanto el cambio cultural.
- *Scrum* tiene unos roles, eventos y artefactos claros y definidos (detallados en el ANEXO II – *Scrum*) que ayudan a establecer una metodología común. Este hecho, dada la idiosincrasia de la Administración puede favorecer su aceptación.
- *Scrum* se ha establecido como la metodología más apropiada para una organización como una SGTIC de un Ministerio, según una guía elaborada por (Pérez Pérez, 2012).

Por otro lado, es habitual combinar diferentes metodologías ágiles, por ejemplo, *Scrum* y *eXtreme Programming* (Kniberg, Scrum and XP from the Trenches, 2015), por lo que a lo largo del presente trabajo se harán menciones también a algunas referencias de prácticas de *eXtreme Programming*.

Asimismo, para proyectos complejos, con unas necesidades de coordinación entre equipos, es muy habitual mezclar los principios de *Scrum* con otras metodologías como

[6] No se ha realizado un análisis exhaustivo, puesto que lo más importante del trabajo es la aplicación de las metodologías y no tanto la elección de una de ellas.

[7] *12th Annual State of Agile Report* - https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report

Lean y *Kanban*, que favorecen el escalado y la coordinación necesaria en proyectos de gran tamaño y con varios equipos (Kniberg, Lean from the Trenches, 2011).

4.3.1 Escalado de *Scrum*

En ocasiones, debido al tamaño del producto, es necesario tener equipos de desarrollo de decenas de personas

En estos casos, se hace necesario segmentar los equipos *Scrum* de forma que los equipos de desarrollo cumplan con el tamaño adecuado de entre 3 y 9 personas. Lo cual plantea importantes dudas ¿Cuántos equipos crear? ¿cómo decidir quién estará en cada equipo?

Algunas herramientas que han sido probadas para lidiar con esta complejidad son la figura del jefe de equipo, los eventos de *Scrum* de *Scrums* o las retrospectivas multi-equipo (Kniberg, Scrum and XP from the Trenches, 2015). Existen muchas referencias a otro tipo de técnicas de gestión asociadas al manejo de estas situaciones, pero no son propias de *Scrum*, por lo que no se tratan más en detalle en este trabajo.

Además, la aplicación conjunta de los principios y valores de la filosofía Lean y la aproximación a esa metodología de *Kanban*, puede ayudar enormemente en la coordinación de varios equipos de desarrollo (Kniberg, Lean from the Trenches, 2011).

5 DIAGNÓSTICO

5.1 Problemas

Tradicionalmente, el desarrollo de *software* ha utilizado una metodología denominada en cascada, con unas fases diferenciadas y que se desarrollan secuencialmente, es decir, no empieza una fase hasta que no termina la anterior y los productos de una fase previa sirven de entrada a la fase siguiente. Es un modelo predictivo, en el que se realiza una gran labor de planificación y se sigue el plan establecido, con la confianza de que las circunstancias no cambien.

Las fases habituales del desarrollo en cascada, y que son básicamente las descritas en la metodología de desarrollo Métrica v3, son las siguientes (Consejo Superior de Informatica, 2001):

- Análisis del sistema de información.
- Diseño del sistema de información.
- Construcción del sistema de información.
- Implantación y Aceptación del sistema.

El resultado de las dos primeras fases son documentos de análisis y diseño, que sirven de base para la construcción y posterior implantación y aceptación, del sistema de información.

El ciclo de vida del *software* resultante de esta metodología también se denomina en cascada y puede verse en la siguiente ilustración:

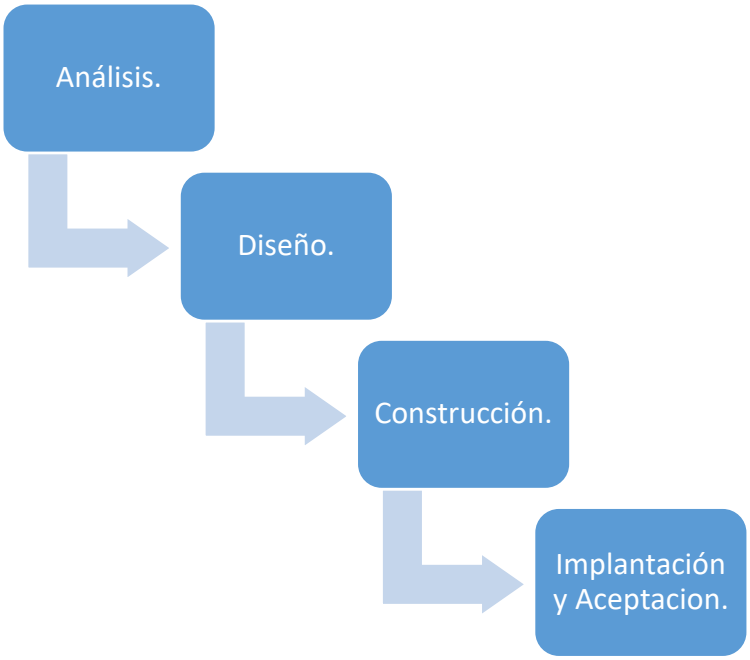

Ilustración 1. Ciclo de vida en cascada

Los problemas derivados de esta forma de trabajar son los siguientes:

- El periodo de tiempo que transcurre desde que surge la necesidad hasta que empieza a operar el *software* es excesivamente largo. De este modo cualquier tipo de desarrollo, incluso el más sencillo, necesita de un plazo mínimo de 3 meses para ser implantado. En la Administración, con los procesos de contratación necesarios este plazo se convierte en un mínimo de 6 meses.

- Los errores en el análisis y el diseño, que se detectan en la fase de aceptación, conduce necesariamente a un rediseño y nueva construcción, lo cual incrementa el coste de desarrollo y los plazos. Esto es así, ya que en la corrección de un error, cuanto más tarde se produce, mayor es el coste asociado a dicha corrección (Sutherland, Waste is a Crime, 2015).
- Los requisitos determinados en la fase de análisis deben ser estables a lo largo del tiempo para que el proyecto sea exitoso. En el dinámico mundo actual, y en especial en el mundo político-administrativo, las prioridades cambian con frecuencia, por lo que este modelo genera frustración en los proveedores e insatisfacción en los usuarios de las aplicaciones.

Aunque no es habitual en la Administración utilizar una metodología en cascada pura como la que se ha descrito con anterioridad, la legislación en materia de contratación TIC y la falta de personal técnico en las administraciones, obligan a la aplicación de un modelo de similares características y, por tanto, con similares problemas.

Detallando más el caso concreto de la unidad administrativa objeto de este plan de mejora, todos los desarrollos de *software* se llevan a cabo mediante contratación centralizada por Acuerdo Marco 26/2015.

Los desarrollos deben ser contratados a proveedores de servicios ya que la unidad no dispone de personal técnico especializado que pueda ejecutar estos desarrollos[8]. En concreto en la unidad objeto de estudio, desarrollan su trabajo en torno a 250 personas, de las cuales un 75% es personal contratado a proveedores de servicios.

Por otro lado, la contratación de servicios de desarrollo de *software* relacionado con la Administración Electrónica está declarada como servicio de contratación centralizada, en base a lo establecido en el artículo 206 del texto refundido de la Ley de Contratos del Sector Público (TRLCSP). En ese sentido, es importante reseñar, que el mencionado artículo establece que *"podrá declarar de contratación centralizada los suministros,*

[8] Las razones por las que no se dispone de personal técnico especializado o la decisión de si la unidad debe disponer o no de este personal es una cuestión que se escapa del marco de este plan de mejora, por lo que no se desarrolla en mayor medida.

obras y servicios que se contraten de forma general y con características esencialmente homogéneas por los diferentes órganos y organismos". En ese sentido, al declarar estos servicios de contratación centralizada, se está asumiendo una homogeneidad en los servicios de desarrollo de software que dista mucho de la realidad.

Esta centralización ha tomado forma en el artículo 2 de la Orden EHA/1049/2008, de 10 de abril, de declaración de bienes y servicios de contratación centralizada. Y debe aplicarse siempre y cuando el presupuesto de licitación no supere 862.000 euros, I.V.A. excluido.

La aplicación del Acuerdo Marco 26/2015 determina que el documento de licitación asociado debe tener las siguientes características (Ministerio de Hacienda, 2019):

- Detalle de las especificaciones del servicio. Es decir, se debe detallar en el momento de la contratación el servicio de desarrollo a realizar, con sus requisitos, necesidades, entregables, etc.
- En ningún caso puede tener como objeto la contratación de horas de disponibilidad de profesionales.
- El criterio de precio tendrá necesariamente un peso mínimo del 51 por ciento del total de la puntación, lo cual implica una preponderancia de ofertas de precio bajo en detrimento de la calidad del servicio o de la experiencia del personal que ejecuta ese servicio.

Además, los plazos asociados a la adjudicación de contratos por Acuerdo Marco 26/2015 se estiman entre 2 y 4 meses, dependiendo del importe y los tiempos de tramitación de los diferentes documentos. Aunque este plazo es reducido en comparación con otras formas de contratación, es demasiado elevado para adaptarse a la realidad social actual, con cambios frecuentes y, por tanto, cambios en las necesidades de productos *software* también frecuentes.

Por otro lado, el mundo del desarrollo de *software* debe evolucionar para adaptarse a las nuevas realidades sociales:

- Se producen cambios muy drásticos en las organizaciones. En especial en la Administración General del Estado, muy dependiente de los ciclos políticos que

cada vez son más cortos y que generan no solo cambios en las prioridades sino también en la propia estructura de los Ministerios.

- En muchas ocasiones el usuario no tiene muy claro cuál es la solución a su problema, por lo que se hace necesario acompañarle en esa labor de prospección de la solución más adecuada. Esto requiere una labor iterativa de solución-evaluación hasta lograr un resultado satisfactorio, lo cual es completamente antagónico a la idea de las metodologías en cascada, donde los requisitos están claros, se definen al principio del proyecto y deben estar cerrados antes de la fase de diseño. Este problema se ve agravado por la modalidad de contratación.
- La confianza, y por tanto la transparencia, es fundamental para el éxito de un proyecto. Por ello, se deben llevar a cabo acciones para incrementarlas como, por ejemplo, visualizar el trabajo en curso o mostrar cada cierto tiempo los avances alcanzados en el proyecto. Ya no es suficiente con "desaparecer" y volver después de un año con un *software* desarrollado en base a lo que se especificó; es necesario mostrar el trabajo periódicamente y evaluar si el producto que se está desarrollando cumple con las expectativas.
- El conocimiento está distribuido, por lo que la calidad y utilidad de los productos *software* se incrementa si hay una participación intensiva de todos los actores involucrados.
- La escasez de recursos y la competencia hacen necesario incrementar la productividad de los recursos disponibles. En el caso de las Administraciones Públicas, aunque no sufren la competencia, sí tienen cada vez más responsabilidades y menos recursos disponibles, en especial en materia de tecnologías de la información.

5.2 Soluciones

Para abordar los problemas descritos, se propone cambiar el marco de trabajo en el que se producen los desarrollos de *software* en las Administraciones Públicas.

La solución propuesta es aplicar un marco ágil de desarrollo de *software*, como *Scrum*, que es adaptativo, es decir, se va desarrollando, revisando y adaptando a las nuevas circunstancias o a cambios en los criterios.

Scrum tiene las siguientes características que ayudan a resolver los problemas descritos:

- Al basarse en un ciclo de vida iterativo e incremental, con ciclos de entrega lo más corto posibles, se despliega un producto *software*, que aporta valor al usuario/cliente en periodos que van de 1 semana a 1 mes.
- Este desarrollo continuo de productos mínimos, reduce el impacto de los errores cometidos en el análisis y el diseño, ya que se detectan y se pueden corregir antes.
- El cambio es bienvenido, por lo que, si se producen cambios en los requisitos, se adaptan los trabajos para que el producto *software* resultante los tenga en cuenta. Esto también es aplicable a los cambios en las organizaciones o en las prioridades político-administrativas.
- El ciclo iterativo, facilita que el usuario/cliente vaya construyendo la visión de la solución que necesita al mismo tiempo que se va construyendo, y que vaya adquiriendo conocimiento sobre el problema que pretende resolver. De este modo la solución resultante es mucho más efectiva que la que habría resultado de un análisis sin conocimiento del proceso.
- Se fomenta la confianza, con las entregas de valor en periodos cortos y con la visualización del trabajo por parte de todos los participantes e implicados en el proyecto.
- Las demostraciones y revisiones frecuentes, abiertas a todo aquel que quiera participar, fomentan la colaboración y que el producto *software* se alimente de todo el conocimiento disponible en la organización.
- Aunque no es el objetivo de las metodologías ágiles, sino más bien una consecuencia, trabajar con tiempos fijos para entregar productos de valor genera una sensación de avance y reduce los tiempos muertos. Fomentar la colaboración y la comunicación entre las personas y reducir la burocracia

innecesaria optimiza el valor aportado. Todo ello incrementa la productividad de los equipos que trabajan con estas metodologías (Lynn Cooke, 2014).

Además de resolver los problemas vistos anteriormente, la aplicación de metodologías ágiles presenta las siguientes ventajas:

- Se tiene un mayor conocimiento del producto que se está desarrollando. La transparencia inherente al marco ágil, hace que las organizaciones tengan un mayor conocimiento de los detalles de los productos que se desarrollan y del negocio asociado a los mismos. En una organización como las Administraciones Públicas, donde el desarrollo de *software* está mayoritariamente externalizado, este punto es especialmente relevante para que el conocimiento continúe en las organizaciones una vez el contrato haya finalizado.
- Mayor motivación de los empleados. Las metodologías ágiles se centran en las personas, en la comunicación y en las interacciones entre las mismas. Esto favorece la motivación de los empleados lo cual, a la larga, incrementa la productividad (Concas, Damiani, Scotto, & Succi, 2007).
- Mejora continua de los procesos y productos. Un principio básico de las metodologías ágiles, es la adaptación. Esto supone una revisión constante de los procesos de desarrollo de *software* para detectar problemas y corregirlos o áreas de mejora. Por medio del evento de retrospectiva, en *Scrum* se dedica un tiempo específico a la mejora continua del proceso de desarrollo.

6 PROPUESTA DE MEJORA

6.1 Diseño

6.1.1 Objetivos

Mediante la aplicación del plan de mejora se pretende lograr los siguientes objetivos:

- Incrementar la frecuencia con la que se entrega un producto *software* que aporte valor a los interesados. Se propone como objetivo llegar a las 3 semanas de duración de los *Sprint* (la definición de lo que es un *Sprint* puede verse en el

ANEXO II – *Scrum*). Es decir, que cada 3 semanas se entregue y se enseñe un incremento del *software* a desarrollar del cual se pueda recibir *feedback*.

- Mejorar la satisfacción de los usuarios con el *software* desarrollado por la SGTIC. Para la medición de este parámetro se pueden utilizar, por ejemplo, encuestas de opinión. De este modo se podrá comprobar si se están obteniendo las mejoras deseadas en cuanto a valor aportado a los usuarios.
- Incrementar la productividad de los equipos de desarrollo de *software*. Se pueden utilizar indicadores como la velocidad, que permiten medir la productividad de los equipos. De este modo se podrá comprobar si se están obteniendo los resultados deseados o si por el contrario hay que tomar medidas correctoras.

Además de los objetivos mencionados, que se pueden medir y cuantificar, se pretende lograr otros objetivos adicionales, que, aunque de difícil cuantificación, se asocian a la aplicación correcta de un marco de trabajo ágil, y son beneficiosas para la organización. Estos objetivos cualitativos son los siguientes:

- Mayor motivación de los empleados públicos y el personal asociado a los contratos de proveedores de servicios.
- Mayor transparencia de las acciones y resultados de la unidad, con respecto a los interesados a los que presta servicio.
- Mayor satisfacción de los ciudadanos con la Administración Electrónica.

6.1.2 Retos

A la hora de plantear el plan de aplicación, se ha recabado la experiencia de diferentes unidades administrativas que están avanzando en la aplicación de metodologías ágiles en relación al desarrollo de *software*.

Fruto de estas entrevistas, se han recopilado un conjunto de retos que a los que se enfrenta la aplicación de metodologías ágiles en el desarrollo de *software* en la Administración.

6.1.2.1 Estabilidad del personal

La aplicación del marco de trabajo ágil requiere un elevado nivel de capacitación de las personas involucradas. Esto es logra mediante una adecuada formación y mediante la estabilidad en el puesto de trabajo. Dado que los equipos de desarrollo suelen pertenecer a proveedores de servicios la rotación suele ser muy elevada, en especial en aquellos perfiles con menor remuneración. Este reto se ve agravado por el peso del 51% mínimo que se da al criterio de precio en la valoración de las ofertas en los contratos basados en AM 26/2015 y que provoca ofertas con precios muy ajustados y por tanto con personal peor remunerado.

6.1.2.2 Dedicación necesaria para ser Product Owner

Las tareas asociadas al rol de *Product Owner*, que debería ser ejercicio por personal propio de la Administración, son muy exigentes, en especial en negocios o en productos complejos.

El *Product Owner* debe tener un amplio conocimiento del negocio, lo cual solo se logra mediante un contacto frecuente y directo con los *Stakeholders* y mediante una formación continua.

Además, el *Product Owner* debe trabajar en colaboración con el equipo de desarrollo, transmitiendo la visión que tiene del producto, resolviendo dudas, aclarando conceptos y priorizando los elementos que se van a desarrollar.

Estas tareas tienen una elevada carga de trabajo lo que supone que una misma persona no pueda gestionar más de 1 o 2 productos de elevada complejidad.

En la Administración, esta no es la tónica habitual; sino que un mismo empleado público suele ser responsable de múltiples aplicaciones complejas.

6.1.2.3 Naturaleza de la Administración como organización

La Administración Pública es una organización muy compleja, con una marcada estructura jerárquica y política de persona difícil de cambiar. Además, por su propia naturaleza, está orientada a procedimientos, y no a servicios o productos.

Esta cultura de jerarquía y prevalencia de los procedimientos está fuertemente arraigada, tanto en los directivos y trabajadores de la propia organización como en la ciudadanía. Además, es contraria a varios de los principios de las metodologías ágiles que se han visto, por lo que el cambio de valores y principios que requiere es especialmente complejo y difícil.

6.1.2.4 Contratación

La contratación de productos y servicios en la Administración está sujeta a multitud de controles y a una normativa muy estricta.

En especial, en materia de desarrollo de *software*, todos los servicios orientados al desarrollo de una aplicación a medida, deben contratarse mediante contratación centralizada marcada en el Acuerdo Marco 26/2015. Esto incrementa la rigidez de la contratación, en una materia, que necesita aún mayor flexibilidad y dinamismo, por la propia naturaleza del servicio a desarrollar.

Este punto se ha visto más en detalle en la sección de Problemas.

Por otro lado, debido a los mecanismos de licitación existentes, se puede dar el caso que la oferta técnica sea de elevada calidad, pero que no haya sido preparada por la misma persona que luego ejecuta el contrato. De hecho, es algo habitual que las empresas tengan personal especializado en la preparación de ofertas diferenciado del personal que ejecuta los contratos.

6.1.2.5 Cesión ilegal de trabajadores.

La gran mayoría de los desarrollos de aplicaciones en las Administraciones Públicas los lleva a cabo personal asociados a contratos de servicios de desarrollo de *software*. Por ello, se hace necesario establecer unas pautas de trabajo para la ejecución correcta de estos contratos y que no se produzca la cesión ilegal de trabajadores.

Estas pautas se tradujeron en 2012 en un dictamen denominado “Instrucciones sobre buenas prácticas para la gestión de las contrataciones de servicios y encomiendas de gestión a fin de evitar incurrir en supuestos de cesión ilegal de trabajadores”, elaborado

de forma conjunta por la Secretaria de Estado de Administraciones Públicas y la de Presupuestos y Gastos. (Ministerio de Hacienda y Administraciones Publicas, 2012)

Estas instrucciones incluyen una sección de buenas prácticas en la fase de ejecución, que afectan a la manera en la que el personal asociado a contratos de servicios se relaciona con el personal propio de las Administraciones. Aunque estas buenas prácticas no son puramente contrarias a la filosofía ágil, si establecen unos límites y un marco de trabajo que pudieran dificultar la implantación completa de un marco de trabajo ágil en el desarrollo de *software* en las Administraciones Públicas.

Adicionalmente a esas instrucciones, el artículo 308.2 de la Ley de Contratos del Sector Público(LCSP), también hace referencia a que se tomen las precauciones para evitar caer en los supuestos de cesión ilegal de trabajadores. Concretamente dice lo siguiente:

"A la extinción de los contratos de servicios, no podrá producirse en ningún caso la consolidación de las personas que hayan realizado los trabajos objeto del contrato como personal de la entidad contratante. A tal fin, los empleados o responsables de la Administración deben abstenerse de realizar actos que impliquen el ejercicio de facultades que, como parte de la relación jurídico laboral, le corresponden a la empresa contratista."

6.2 Medidas a implantar

En esta sección se van a desarrollar un conjunto de medidas que se podrían aplicar para el establecimiento de un marco de trabajo ágil en relación al desarrollo de *software* en las Administraciones Públicas.

Se han agrupado por área temática, sin establecer un orden concreto de implantación. En la sección de Planificación temporal, se establecerán referencias temporales y una propuesta de planificación para la aplicación de algunas de estas medidas en la unidad en concreto que es objeto de este plan de mejora.

Se ha decidido desarrollar la propuesta de este modo ya que la aplicación de metodologías ágiles supone un cambio cultural muy importante, una transformación. Además, siguiendo los valores del Manifiesto ágil, lo más importante es la inspección y

la adaptación del proceso de implantación, por lo que se enumeran un conjunto de medidas que se podrían aplicar y que es más importante que el orden concreto en el que se apliquen.

De hecho, la propia planificación de la implantación, es contraria a los principios del marco de trabajo ágil, que se basa en una implantación iterativa y una revisión y adaptación constante del proceso de implantación.

6.2.1 Medidas relativas a la contratación

6.2.1.1 Información a incorporar a los pliegos

Como ya se ha comentado, habitualmente el equipo de desarrollo de *software* es contratado mediante un contrato de servicios mediante contratación centralizada.

Para implantar un marco de trabajo ágil en el desarrollo de *software*, es necesario incorporar a los pliegos cierta información, en especial en relación a la metodología de desarrollo a utilizar y los perfiles a contratar.

Respecto a la metodología de desarrollo se debe mencionar que se utilizará *Scrum* como marco de trabajo. Al ser *Scrum* muy conocido y de amplio uso en el mercado esta mención indicará claramente a los licitadores la forma de trabajo que se seguirá en el desarrollo del *software*.

Respecto a los conocimientos y experiencia requeridos al personal, además de los concretos necesarios para el pliego del que se trate (tanto a nivel de negocio como a nivel técnico), se debe incluir experiencia en *Scrum* y otras metodologías ágiles. Asimismo, se recomienda incorporar perfiles con amplia experiencia, pues un marco de trabajo ágil como *Scrum* requiere equipos maduros y, por tanto, personal muy capacitado profesionalmente.

Además, se debe evitar la distinción de perfiles y tareas entre analista y programador, con el fin de evitar los silos de conocimiento. Siguiendo la filosofía de *Scrum*, todos los miembros del equipo deben realizar tareas de análisis, diseño y construcción, lo que significa que todos los perfiles deben realizar todas las tareas (o la gran mayoría de ellas) en relación con la puesta en servicio de un producto *software*.

Por otro lado, se podrían redactar los requisitos del pliego en forma de historias de usuario, que es una técnica habitual que se emplea en la especificación de requisitos al trabajar en un marco de trabajo ágil. (Cohn, 2004)

6.2.1.2 Flexibilidad de requisitos en el marco actual de contratación.

Como se ha comentado, es frecuente que los desarrollos de *software* que se realizan en las Administraciones Públicas, se realicen en el marco de un contrato de servicios.

Esta circunstancia resta flexibilidad a las posibilidades de actuación en esta materia respecto a que el desarrollo lo llevase a cabo personal propio de la Administración.

En este apartado se analizan algunas medidas que se pueden adoptar en el marco normativo actual, para dotar de una mayor flexibilidad a la contratación y acercarla a un marco de trabajo más ágil.

6.2.1.2.1 Modificaciones de contratos basados en AM 26/2015

Para los desarrollos de *software* que se han contratado mediante un contrato basado en AM 26/2015, una posibilidad de poder tener cierto margen de flexibilidad en cuanto a la priorización y modificación de requisitos, es la modificación del contrato.

La modificación de los contratos basados en AM 26/2015 está recogida en el punto 9 del Pliego de Cláusulas Administrativas Particulares de dicho Acuerdo Marco.

En ese punto se dice que los contratos basados en el Acuerdo Marco podrán modificarse, siendo obligatorio para el contratista conforme al artículo 219 del TRLCSP[9], en el caso de que, manteniendo la finalidad del mismo y de acuerdo con el artículo 106 del TRLCSP, se produzcan las circunstancias siguientes y éstas estén definidas en el documento de licitación:

[9] En el momento de la redacción del trabajo, el Acuerdo Marco 26/2015 está aún vigente y se rige por la ley que estaba vigente cuanto se licitó y es a la que se referencia en los PCAP del citado AM. Esta ley es el RD-Legislativo 3/2011, de 14 de noviembre, por el que se aprueba el texto refundido de la Ley de Contratos del Sector Público.

- Cuando el organismo peticionario y el contratista manifiesten su mutuo acuerdo para la no realización de una parte de la prestación, con la consecuente reducción del precio del contrato basado.
- Cuando el organismo peticionario manifieste la necesidad de realizar una modificación en las tareas objeto del contrato basado, siempre que el importe de la modificación planteada no supere el 20% del precio del contrato basado. En todo caso, las nuevas tareas propuestas deberán estar concretadas en el documento de licitación y ajustarse al objeto definido en el Acuerdo Marco.

El procedimiento a seguir en el caso de acordar la modificación de alguno de los contratos será el establecido en el artículo 211 del TRLCSP, en relación con el artículo 108 también del TRLCSP.

En la práctica, esta modificación exige establecer un conjunto de características que deben tener las nuevas tareas a realizar, ya que las tareas propuestas deben estar previstas en el documento de licitación.

Por otro lado, la modificación está limitada a un 20% del importe del contrato. En este porcentaje se deben incluir tanto los requisitos que se han suprimido como los que se han añadido. De este modo, si se decide cambiar unos requisitos del contrato por otros que no estaban contemplados, de forma que no se modifique el importe, el peso de esos requisitos solo debe ser del 10% del contrato.

A esta poca flexibilidad debe añadirse que el procedimiento de modificación de un contrato basado en AM 26 tiene unos plazos de tramitación que pueden llegar a alargarse 1 mes.

6.2.1.2.2 Contratación por procedimiento abierto

Otra posibilidad para aportar mayor flexibilidad en los contratos asociados a los servicios de desarrollo de *software*, es aplicar un procedimiento abierto de adjudicación, en vez de contratación centralizada.

Para los servicios de desarrollo de *software*, tal y como establece el artículo 2 de la Orden EHA/1049 /2008, se declaran de contratación centralizada *"los servicios dirigidos*

al desarrollo de la Administración Electrónica cuyo presupuesto de licitación no supere 862.000 euros, I.V.A. excluido".

Por tanto, para poder aplicar un procedimiento abierto en la adjudicación del contrato, el presupuesto de licitación debe ser superior a dicho importe.

Contratar mediante un procedimiento abierto permite especificar el objeto del contrato en términos de unidades de trabajo, como componentes de la prestación del servicio; en el caso que no se pueda determinar con exactitud dichas prestaciones en el momento de la preparación del contrato (lo cual es habitual en la contratación de servicios de desarrollo de *software*). Esto es posible ya que el artículo 308.3 de la Ley de Contratos del Sector Público (LCSP) establece lo siguiente:

"En los contratos de servicios que impliquen el desarrollo o mantenimiento de aplicaciones informáticas el objeto del contrato podrá definirse por referencia a componentes de prestación del servicio. A estos efectos, en el pliego de cláusulas administrativas particulares se establecerá el precio referido a cada componente de la prestación en términos de unidades de actividad, definidas en términos de categorías profesionales o coste, homogéneas para cualquier desarrollo, de unidades de tiempo o en una combinación de ambas modalidades.

Esta definición deberá completarse con referencia a las funcionalidades a desarrollar, cuyo marco deberá quedar determinado inicialmente, sin perjuicio de que puedan concretarse dichas funcionalidades por la Administración atendiendo a consideraciones técnicas, económicas o necesidades del usuario durante el período de ejecución, en los términos en que se prevean en el pliego de cláusulas administrativas particulares."

Además, el precio del contrato también se puede determinar en base a precios unitarios en base a los artículos 102.4 y 309 de la LCSP.

A efectos de la contratación, se consideran unidades de trabajo al trabajo técnico que requiera una cierta cantidad de horas de cada uno de los perfiles determinados adecuadamente ponderados por la tarifa media de ese perfil.

Además, este tipo de contratos permiten una modificación del contrato, y una prórroga equivalente a la duración; lo cual aporta mayor flexibilidad a la hora de poder aplicar un marco ágil de trabajo.

Para que la modificación del contrato se pueda realizar se deben dar una de las dos condiciones establecidas en los artículos 204 y 205 de la LCSP:

- Artículo 204. Que las modificaciones se hayan previsto en los pliegos de cláusulas administrativas particulares.
- Artículo 205. Que sean prestaciones adicionales necesarias, que se den circunstancias imprevisibles o que sean modificaciones no sustanciales.

En todo caso, la modificación del contrato solo será obligatoria para el contratista cuando la alteración en su cuantía no exceda del 20% del precio inicial del contrato, IVA excluido; tal y como señala el artículo 206 de la LCSP.

La modificación del contrato se puede realizar por un importe de hasta el 50% del precio inicial del contrato tal y como se determina en el artículo 205 de la LCSP.

El problema de realizar modificaciones de los contratos es que requiere largos procedimientos administrativos, por lo que es una herramienta con una flexibilidad limitada.

6.2.1.3 Propuestas de modificación de la normativa de contratación para agilizar los cambios de requisitos.

Uno de las grandes barreras a la hora de aplicar un marco de trabajo ágil en el desarrollo de *software* en las Administraciones Públicas es la normativa de contratación. Esta normativa impide una flexibilidad que es necesaria en la transformación digital en las que las Administraciones Públicas se encuentran inmersas.

Como ya se ha comentado en la sección Problemas, la contratación de servicios de desarrollo de *software* debe llevarse a cabo mediante contratos basados en el Acuerdo Marco 26/2015, de forma general, con todo los problemas que se han detallado.

Para tratar de paliar la falta de flexibilidad y al mismo tiempo respetar los principios de libertad de acceso a las licitaciones, publicidad y transparencia de los procedimientos,

así como la no discriminación e igualdad de trato entre los licitadores, a continuación, se proponen una serie de medidas relativas a la modificación de la normativa de contratación:

- Incorporar la figura del Acuerdo Marco flexible en relación a los servicios de desarrollo de *software*. En estos Acuerdos Marco se podrían establecer diferentes alcances del servicio, con un presupuesto tasado para cada uno de los alcances. Y que la Administración pueda decidir, en el momento de la ejecución, el alcance del proyecto y por tanto el presupuesto. De este modo se podrían tratar de prever circunstancias como un cambio de Gobierno o similar.
- Posibilidad de contratar perfiles por horas mediante Acuerdo Marco. Estableciendo simplemente el tipo de servicio a desarrollar y el contexto del mismo, pero sin determinar los requisitos concretos a desarrollar. Esto daría una gran flexibilidad a la hora de poder hacer frente a cambios repentinos no previstos. Por supuesto, se deberían establecer controles respecto a los trabajos realizados para evitar un uso ineficiente del dinero público.
- Incorporar un nuevo tipo de contrato de desarrollo de *software*, cuya tramitación y posibilidades de modificación sean acordes a un marco de trabajo ágil.

6.2.2 Medidas relativas a la metodología de desarrollo

Como ya se ha explicado en la sección de Diagnóstico, el ciclo habitual que se sigue en el desarrollo de *software* es un modelo en cascada, es decir, seguir una serie pasos de manera secuencial y empezar una fase cuando se ha terminado la anterior. Para aplicar *Scrum* se debe evolucionar a un modelo iterativo e incremental, en el cual se produzcan entregas frecuentes y a periodos fijos y lo más cortos posibles.

El ciclo de vida iterativo e incremental que se sigue en *Scrum* puede verse en la siguiente ilustración, disponible en la web *Scrum*.org:

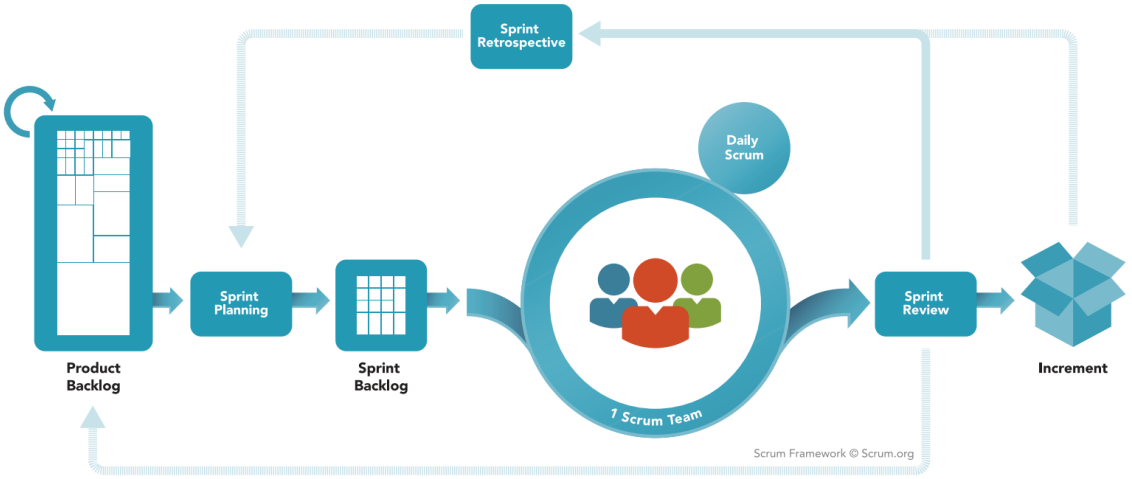

Ilustración 2. Ciclo de vida iterativo e incremental de Scrum

La diferencia fundamental con un ciclo de vida en cascada es que, el producto final, se produce a incrementos, que se van analizando al final de cada *Sprint*; en vez de producir un único producto al final del desarrollo.

Además, debe existir una constante colaboración y comunicación entre todos los miembros del equipo *Scrum*, en especial entre el *Product Owner* y el equipo de desarrollo. Por ello, la cercanía física del puesto de trabajo de los implicados es importante, y cuando no sea posible esta cercanía, se deben establecer herramientas que faciliten lo máximo posible una comunicación bidireccional en tiempo real.

Esto supone un cambio importante en la forma de trabajo habitual que requiere adaptación de todos los participantes del proceso de desarrollo, tanto del equipo *Scrum* como del resto de interesados (*stakeholders*).

6.2.3 Medidas relativas a la gestión de Recursos Humanos

6.2.3.1 Plan de formación

Un aspecto fundamental para implantar el marco de trabajo ágil es la mentalidad, creencias y forma de trabajar de las personas.

Esto es especialmente importante en la Administración donde principios como el de jerarquía y eficacia están muy arraigados, dado que se debe cambiar esa mentalidad hacia los principios de colaboración cara a cara, simplicidad y auto-organización de equipos y las personas.

Por ello, es necesario incorporar en el plan de formación de la unidad diferentes cursos de métodos ágiles de trabajo, o incluso crear un plan de formación específico. Estos cursos deben ser de diferentes niveles, desde cursos básicos de *Scrum* y otras metodologías ágiles, para el personal que no tenga conocimiento sobre la materia; hasta cursos avanzados para aquellos con los conocimientos básicos afianzados y que desean seguir avanzando en el conocimiento del marco de trabajo ágil.

No se recomienda forzar la asistencia a estos cursos. La esencia del marco de trabajo ágil se basa en los individuos motivados, lo cual difícilmente se puede lograr mediante la imposición.

Para fomentar el interés del personal se pueden incorporar charlas de corta duración que ayuden a difundir la existencia de las metodologías ágiles y sus resultados positivos en otras organizaciones, tanto a nivel de productividad como a nivel de motivación.

Se recomienda que la formación sea presencial y que se produzca simultáneamente para aquellas personas que trabajan en el mismo servicio/producto, de modo que se pueda crear mayor sensación de equipo y que las dinámicas que se apliquen tengan un mayor impacto en la mentalidad de los participantes.

6.2.3.2 Plan de seguimiento/acompañamiento.

La formación, antes mencionada, suele centrarse en aspectos teóricos, aunque se utilicen dinámicas y actividades orientadas a la práctica. Sin embargo, cuando se intenta llevar a la práctica es posible que surjan dudas, que a veces no se sepa cómo actuar o que no se apliquen los principios y valores correctamente.

Por ello, adicionalmente a esta formación, se recomienda que personal experto en el marco de trabajo ágil acompañe durante la implantación a los equipos, al menos inicialmente.

Este acompañamiento lo puede llevar a cabo el *Scrum Máster* o alguien externo al equipo *Scrum* con conocimientos avanzados tanto teóricos como prácticos en marcos de trabajo ágiles. Y podría ser alguien perteneciente a la Oficina de agilidad, si ésta está implantada, o alguien independiente de ella.

Esta persona participaría en los diferentes eventos, ayudaría en el ejercicio de los diferentes roles, así como en la elaboración de los diferentes artefactos *Scrum*.

El objetivo es aplicar los conocimientos teóricos adquiridos durante la formación al trabajo diario, de forma que el conocimiento se consolide en base a la práctica.

6.2.3.3 Transformación de personal administrativo a Product Owner.

Uno de los retos de aplicar un marco de trabajo ágil en la Administración es la dedicación que requiere el rol de *Product Owner*. Las tareas asociadas a este rol suponen que una persona solo puede ejecutar este rol para 1 o máximo 2 productos complejos.

Tradicionalmente, son los Jefes de Servicio (nivel 26) los que han asumido unas competencias similares a *Product Owner*, dejando los niveles inferiores para otras tareas de carácter más administrativo.

Para paliar esta carencia de personal se puede reconvertir el personal administrativo (niveles 22) a *Product Owner*.

Este enfoque tiene la ventaja adicional de permitir que el valioso conocimiento necesario para ejercer el rol de *Product Owner*, esté en manos del personal funcionario y por tanto sea más estable en la organización.

6.2.4 Medidas relativas a la tecnología

Dado que la aplicación de una metodología ágil de trabajo como *Scrum*, requiere despliegues de nuevas versiones de *software* con frecuencia (y cuanto mayor sea la frecuencia mejor), es muy importante reducir el esfuerzo de llevar a cabo estos despliegues. Para ello es importante aprovechar la tecnología existente para automatizar, en la medida de lo posible, las acciones que se requieren para el desarrollo y despliegue de un nuevo *software*.

Adicionalmente, se deben tomar medidas para que la modificación de las aplicaciones también suponga el menor esfuerzo posible, ya que un ciclo de vida iterativo, implica modificaciones frecuentes del código desarrollado con el objetivo de encontrar la solución óptima.

Se describen en este apartado medidas, eminentemente tecnológicas, que permiten reducir tanto el esfuerzo asociado al desarrollo y la modificación de *software* existente, como la puesta en producción de una nueva versión del mismo.

Relacionado con esta cuestión, está la cultura *DevOps*, que pretende eliminar los silos existentes tradicionalmente entre el departamento de desarrollo (*Dev*) y el de operaciones (*Ops*) para lograr reducir los tiempos y costes asociados a la puesta en producción de las aplicaciones. Esta cultura está siendo cada vez más adoptada por la industria, como muestran los datos de un 74% de empresas en proceso de adopción de *DevOps* en 2016 (Vadapalli, 2018).

6.2.4.1 Sistema de control de código fuente

Un sistema de control de código fuente, también denominado de gestión de código fuente (*SCM – Source Code Management*) permite llevar un registro de los cambios que se han realizado sobre un producto o sobre su configuración.

Este mecanismo es esencial para poder detectar errores y corregirlos de manera temprana.

Además, estos sistemas permiten trabajar con ramas de código, de forma que los miembros del equipo, o diferentes equipos del mismo producto, puedan trabajar en paralelo, logrando una gran eficiencia y productividad.

Un ejemplo de forma de trabajo es tener una rama principal, y que cada equipo que trabaja en el proyecto tenga su rama de equipo. Cada día, cada equipo y cada programador actualizan su rama a partir de la rama principal. De este modo siempre tienen el código actualizado y realizan las modificaciones pertinentes en su código para que todo compile correctamente. Cuando un desarrollador termina un trabajo, consolida ese código a la rama de su equipo y cuando el equipo termina una característica del *software*, consolida ese código en la rama principal. De esta manera todos los problemas y conflictos que surjan al consolidarlo, se resuelven inmediatamente y la rama principal siempre tiene la versión más reciente del *software* (Kniberg, Lean from the Trenches, 2011).

Algunas herramientas que se pueden utilizar como sistema de control de versiones son las siguientes:

- *Subversion.*
- *Git.*
- *Codeville.*
- *GNU Bazaar.*
- *Team Foundation Server*.

6.2.4.2 Mantenibilidad del código fuente

Otro elemento fundamental para reducir el coste de las frecuentes modificaciones asociadas al ciclo de vida iterativo e incremental, es implantar buenas prácticas que permitan que el código fuente sea mantenible.

En ese sentido, existen numerosas métricas que permiten evaluar la mantenibilidad de un proyecto de *software* (Irrazábal & Garzás, 2010).

Una de ellas es la cobertura de pruebas unitarias, que mide el porcentaje de código fuente que está cubierto por pruebas automatizadas.

En ese sentido, filosofías de desarrollo como *Test-Driven Development* (TDD) o *Behavior-Driven Development* (BDD), pretenden que el desarrollo esté dirigido por las pruebas que hay que superar para que ese desarrollo se dé por satisfecho (Beck, Test-driven Development: By Example, 2003).

Además, para facilitar la tarea de monitorización y seguimiento de las métricas de mantenibilidad, existen diferentes herramientas como *SonarQube, Checkstyle, FindBugs, PMD* o *Kiuwan*. Adicionalmente, se podrían incluir en los contratos de servicios de desarrollo que se liciten, la medición de algunas de las métricas existentes.

Relacionado con la mantenibilidad del código fuente tenemos las pruebas de usuario automatizadas, que son diferentes de las pruebas unitarias que rigen TDD. Este tipo de pruebas pretenden cubrir aquellos casos repetitivos, monótonos y tediosos, complementando las pruebas que realizará un humano (Vadapalli, 2018).

Algunas herramientas para la automatización de pruebas son las siguientes:

- *Visual Studio Test Professional.*
- *SoapUI.*
- *Selenium.*
- *QTP.*
- *TestDrive.*

6.2.4.3 Integración continua

La integración continua (en inglés *continuous integration*) consiste en la combinación de los cambios de código fuente de todos los desarrolladores en un repositorio central de forma periódica (mínimo una vez al día). Además, suele implicar la ejecución de tareas automáticas como la ejecución de pruebas automatizadas, la compilación del código y la generación de un ejecutable, el despliegue en un entorno de desarrollo u otras que se configuren en el servidor.

Una de las ventajas de la integración continua es la detección temprana de errores en las fases de integración de los desarrollos de las diferentes personas que trabajan en el equipo.

Algunas herramientas que se pueden utilizar para implantar una solución de integración continua son las siguientes:

- *Jenkins*.
- *TeamCity.*
- *Travis CI.*
- *Go CD.*
- *Bamboo.*
- *GitLab CI.*
- *CircleCI.*
- *Codeship.*

Un paso previo habitual a la integración continua, es la gestión de compilaciones, o la compilación automática (denominado en inglés *Build Managment* o *Build automation*).

Lo que se persigue mediante esta técnica es automatizar la compilación y construcción del incremento para que pueda ser posteriormente será desplegado en el entorno que corresponda.

Algunas herramientas que se puede utilizar para la compilación automática son las siguientes (Vadapalli, 2018):

- *Ant.*
- *Buildr.*
- *Maven.*
- *Gradle.*
- *MSBuild.*

6.2.4.4 Entrega continua

El objetivo de todas las medidas desarrolladas y explicadas anteriormente, es lograr que el despliegue de una nueva versión de *software* se produzca de forma casi automática, lo que se denomina entrega continua (en inglés *continuous delivery)*. Automatizar las tareas tiene la ventaja de que se reducen los errores asociados a intervenciones humanas, así como el esfuerzo necesario para la puesta en producción.

La siguiente figura identifica como sería el ciclo de entrega continua con sus fases diferenciadas.

Ilustración 3. Ciclo de entrega continua

El objetivo final de la entrega continua es lograr un despliegue del *software* en los diferentes entornos (desarrollo, preproducción y producción) de una forma automática o lo más automatizada posible.

La mayoría de las herramientas que se han visto para el proceso de integración continua se pueden utilizar para automatizar las tareas asociadas al despliegue del *software*.

Las tareas concretas que se deben llevar a cabo, y por tanto que se deben automatizar, dependen del entorno tecnológico del que disponga la unidad.

6.2.5 Medidas organizativas

Se incorporan en este apartado un conjunto de propuestas relativas a la estructura de la organización y al funcionamiento general de la misma.

6.2.5.1 Plan de comunicación

Avanzar hacia la aplicación de metodologías ágiles, como *Scrum*, tiene importantes implicaciones culturales, dado que algunos de los principios en los que se basa el marco ágil pueden friccionar con los principios y valores presentes en la organización.

Por ello, una vez se tenga cierta experiencia en la aplicación de metodologías ágiles, se haya comprobado su efectividad en la organización y se desee implantar de forma más

general, es necesario establecer un plan de comunicación, o establecer estos nuevos valores en el plan de comunicación existente.

Esta comunicación debe ir dirigida a transmitir a todos los miembros de la organización y a los interesados a los que pueda afectar, los nuevos valores y principios, así como los pasos que se están dando hacia un trabajo más centrado en principios ágiles.

Los principales pasos para implantar o modificar un Plan de comunicación son los siguientes[10] (Aljure, 2016):

1. Análisis y Diagnóstico. ¿Cómo se percibe actualmente a la organización?
2. Definición de objetivos del plan.
3. Determinación del público objetivo.
4. Diseño del mensaje. Un mensaje principal y 3 o 4 mensajes secundarios.
5. Elección de medios o canales de comunicación.
6. Recursos disponibles
7. Plan de acción. Acciones para lograr los objetivos del plan.
8. Calendario o cronograma con las próximas acciones.
9. Seguimiento y monitorización.
10. Evaluación y revisión del plan.

6.2.5.2 Oficina técnica de apoyo

Aunque en un marco de trabajo ágil es más importante el *software* que funciona sobre la documentación, la normativa actual en materia de contratación hace que la Administración deba ser muy estricta y escrupulosa en asegurarse que todos los entregables asociados a un contrato se cumplen.

Se debe comprobar además que se cumplen siguiendo los parámetros de calidad determinados en los pliegos de licitación y en la oferta ganadora.

[10] No es el objeto de este trabajo de fin de máster desarrollar un plan de comunicación relacionado con la implantación de metodologías ágiles en las Administraciones Públicas, por lo que solo se dan unas pinceladas sobre la materia.

Estas tareas de seguimiento y control de los contratos las hace tradicionalmente el personal propio de la Administración. Sin embargo, una posibilidad en este marco de trabajo ágil, es externalizar el control y seguimiento diario. Por supuesto, sin menoscabo de que la certificación final de que un contrato se ha cumplido, deba ser verificado por personal propio de la Administración.

De este modo, el personal funcionario se puede dedicar casi en exclusiva a ejercer el rol de *Product Owner*, ejecutando las tareas de mayor valor para la organización.

Esta medida pretende mitigar uno de los problemas habituales en la ejecución del rol de *Product Owner*, que es precisamente la falta de tiempo para poder realizar todas las tareas que supone la asunción completa de ese rol.

Esta oficina técnica puede contratarse mediante un contrato de servicios. La adjudicación de este servicio debería realizarse a una empresa independiente de aquellas que están ejecutando los servicios de desarrollo de *software*. Es decir, la Oficina Técnica debe ser completamente independiente de las empresas que debe controlar.

6.2.5.3 Oficina de agilidad

Una vez que haya varios equipos trabajando con metodologías ágiles, puede ser necesario instaurar una oficina de agilidad.

Esta oficina será la encargada de establecer pautas comunes en la aplicación de metodologías ágiles, aprovechar los aprendizajes de cada uno de los equipos, así como favorecer la coordinación de los diferentes equipos que participen en un mismo proyecto.

Dentro de la metodología *Scrum*, el rol de *Scrum Master* estaría ejercido por una persona de esta oficina; de forma que se independice el marco de trabajo ágil, de la empresa proveedora del servicio de desarrollo de *software*. Con esto se logra una continuidad en el modo de aplicar un marco de trabajo ágil en toda la unidad, con independencia del equipo de desarrollo.

En productos complejos, donde más de un equipo esté involucrado (ver apartado de Escalado de *Scrum*), esta oficina será la encargada de facilitar la coordinación de todos los equipos.

Como el caso más habitual es la carencia de personal funcionario que pueda ejercer la labor se podría contratar como un contrato de servicios este tipo de personal. Incluso se podría incorporar en el mismo contrato que el personal tratado en el apartado Oficina técnica de apoyo

Se ha separado esta oficina de agilidad de la oficina técnica de apoyo, por ser funciones claramente diferenciadas y que deben ser desarrolladas por personal con perfiles diferentes.

6.2.5.4 Reorganización de los espacios de trabajo

La filosofía de trabajo ágil y los principios de transparencia, revisión y adaptación requiere un espacio de trabajo que lo facilite.

Por un lado, se necesitan zonas de pared amplias y sin elementos para poder colocar paneles físicos en los que se pueda tener una visión conjunta del trabajo en curso y de los productos que se están desarrollando. Esta práctica favorece en gran medida la transparencia y la visión holística del trabajo.

Adicionalmente, se recomienda que, junto a los paneles, haya espacio suficiente para poder realizar el *Scrum* Diario; evento fundamental para la inspección y adaptación diaria del trabajo, hacia el objetivo del *Sprint*.

También se recomienda espacios donde todos los miembros del equipo *Scrum* puedan comunicarse lo más fácilmente posible y a ser posible cara a cara. Un despacho amplio tipo “pradera” con todos los miembros del equipo es lo más recomendable y separados del resto de equipos para minimizar el ruido.

Se da por supuesto, que se debe disponer de salas de reuniones con suficiente capacidad para poder realizar los eventos de *Sprint Planning*, *Sprint Review* y *Sprint Retrospective*.

6.3 Planificación temporal

En este apartado se describirá una propuesta de planificación temporal de las medidas a implantar para iniciar un proceso de cambio hacia la aplicación de un marco ágil de trabajo en las Administraciones Públicas.

Como ya se ha comentado, la adaptación de este plan a la realidad de cada organización es vital para lograr el éxito y es además coherente con el cambio cultural que se pretende.

En el siguiente diagrama se puede ver una planificación temporal de las tareas iniciales, todos los plazos son estimativos y podrían estar sujetos a variaciones.

Tarea \ Mes	1	2	3	4	5	6	7	8	9	10	11	12
Medidas relativas a la tecnología												
Establecimiento de indicadores de seguimiento												
Formación general												
Análisis de los equipos y proyectos.												
Formación específica												
Contratación del servicio desarrollo AM 26												
Ejecución - *Sprint* 1												
Ejecución - *Sprint* 2												
Ejecución - *Sprint* 3												
Ejecución - *Sprint* 4												
Ejecución - *Sprint* 5												
Ejecución - *Sprint* 6												

Tabla 2. Planificación temporal de tareas iniciales

6.3.1 Medidas relativas a la tecnología

Como parte fundamental de la implantación de un marco de trabajo ágil en el desarrollo del *software*, se debe reducir el coste asociado al desarrollo y modificación del *software*, así como el asociado al despliegue y puesta en producción de nuevas versiones. Estas medidas ya se han mencionado en el apartado Medidas relativas a la tecnología, la aplicación concreta de estas medidas dependerá del grado de madurez de la unidad.

Aunque en la Tabla 2. Planificación temporal de tareas iniciales, se ha indicado que son tareas que se prolongan durante 1 año, las medidas tecnológicas encaminadas a reducir los costes de desarrollo deben darse de forma constante. Se debe incorporar un ciclo de

mejora continua al entorno de desarrollo y puesta en producción, en consonancia con los principios ágiles que se quieren implantar.

6.3.2 Establecimiento de indicadores de seguimiento

Como parte del plan de mejora, se deben implantar métricas o indicadores para asegurar que el plan está cumpliendo con los objetivos del mismo.

Aunque solo se ha establecido una planificación de tareas para el primer año y un proyecto, es de suponer que la aplicación de un marco ágil de trabajo en el desarrollo de *software* se extienda a toda la unidad y durante todos los años sucesivos, hasta una completa aceptación de los principios ágiles de desarrollo de *software*.

Algunos indicadores que se deben implantar serían:

- Frecuencia de puesta en producción de nueva versión de la aplicación.
- Satisfacción de los usuarios de las aplicaciones.
- Velocidad del equipo de desarrollo de *software*. Medido como el número de funcionalidades que entrega al final de cada *Sprint*.

6.3.3 Formación general

El siguiente paso será la formación, a todo el personal que vaya a estar involucrado en el proceso de desarrollo de *software*, en los principios, prácticas y herramientas asociadas a un marco ágil de trabajo y a *Scrum*.

Es importante recalcar que el proceso de desarrollo de *software* no solo involucra al personal de las empresas proveedoras de servicios y al de la SGTIC, sino que el personal de las unidades que harán uso de las aplicaciones también deben participar en esta formación, pues juegan un papel muy relevante.

Esta formación hará que todo el personal tome conciencia de la nueva forma de trabajo y permitirá detectar a aquellos equipos y personas más interesados para su posterior selección.

Dadas las características de la Administración, el equipo de desarrollo no participará en esta formación, dado que es personal externo, pero sí deben participar todos los

funcionarios que actuarán como *Product Owner*, así como el resto de personal que pueda ser un *Stakeholder* de un proyecto de desarrollo de *software*.

6.3.4 Análisis de los equipos y proyectos.

El siguiente paso, una vez se han llevado a cabo las jornadas de formación a todos los trabajadores de la unidad y de las unidades gestoras, es analizar los proyectos y equipos existentes para identificar cuáles son aquellos idóneos para que comiencen a trabajar con una metodología ágil como *Scrum*.

Se recomienda que el proyecto/producto elegido tenga las siguientes características:

- Sea de cierto impacto, pero no sea crítico.
- Requiera de un equipo de desarrollo de entre 5 y 7 personas.
- Que el equipo de desarrollo se ubique en las oficinas del cliente.

Fruto de este análisis, se seleccionará a un conjunto de personas que ejercerán los roles específicos determinados por *Scrum*, que no son el equipo de desarrollo, es decir *Product Owner* y *Scrum Máster*; así como los trabajadores de las unidades gestoras que serán los *Stakeholder*s del proyecto.

6.3.5 Formación específica y acompañamiento

Una vez identificado el proyecto en el que se empezará a trabajar con la metodología *Scrum* e identificados los actores; se iniciará una formación específica y más detallada y personalizada al equipo en cuestión.

Se realizarán jornadas prácticas para entender en detalle la filosofía de la metodología *Scrum* y de los principios de funcionamiento del marco de trabajo agile.

Además, se aprovechará esta formación para iniciar el trabajo de elaboración de los pliegos asociados al *software* a desarrollar. Para ello se contará con la asistencia de personal externo experto en metodologías ágiles que ayudará en la redacción del mismo siguiendo un formato de historias de usuario.

Además, el desarrollo del pliego servirá para la formación inicial del *Product Backlog*, uno de los elementos clave de la metodología *Scrum*.

6.3.6 Contratación del servicio desarrollo AM 26

Durante la fase de contratación se seguirán los trámites administrativos necesarios para la licitación de un servicio de desarrollo de *software* por Acuerdo Marco 26/2015.

En la redacción del pliego se tendrán en cuenta la información que se ha descrito en el apartado Información a incorporar a los pliegos y el *Product Backlog* generado durante la formación específica.

6.3.7 Ejecución del desarrollo

El proyecto se llevará a cabo siguiendo los principios, pasos y prácticas recogidos en el ANEXO II – *Scrum*.

La duración del *Sprint* se podría establecer en 4 semanas, aunque esa duración dependerá en gran medida de las tareas a llevar a cabo en el proyecto en cuestión. La duración del *Sprint* debe ser suficiente para que durante el mismo se pueda desarrollar un incremento del producto que aporte valor al negocio.

6.4 Otros ámbitos de aplicación de metodologías ágiles

6.4.1 Aplicación de *Scrum* al desarrollo del TFM

Las metodologías ágiles se pueden aplicar a multitud de entornos y ámbitos de negocio, con las adaptaciones necesarias para que todo tenga sentido.

Para la elaboración del TFM, el tutor y el alumno han acordado seguir una adaptación de la metodología *Scrum*, para ir produciendo incrementos sobre el producto que, en este caso, ha sido el propio TFM.

En ese sentido, se estableció la duración del *Sprint* en 4 semanas, que en ocasiones no se pudo cumplir por problemas de agenda, con entregas de una versión del TFM a la finalización de cada *Sprint*. Y una reunión presencial al finalizar ese *Sprint* que servía al mismo tiempo de *Sprint Review*, *Sprint Retrospective* y *Sprint Planning*.

De este modo, se ha ido trabajando ya sobre el documento final de TFM desde la primera reunión de lanzamiento. Se ha tenido *feedback* frecuente sobre el trabajo realizado. Y

se ha ido adaptando el producto y lo que se iba a realizar en base a lo que se ha considerado más apropiado o digno de estudio y análisis.

A continuación, se detallan los *Sprint* que se han ejecutado, con la duración, que ha ido variando por disponibilidades de agenda, y los elementos que se desarrollaron:

- De 5 de diciembre a 4 de febrero de 2019.
    - ¿Qué son las metodologías ágiles?
    - ANEXO II - *Scrum*
- De 5 febrero a 4 de marzo de 2019.
    - Diagnóstico
- De 5 de marzo a 1 de abril de 2019
    - Propuesta de mejora – Diseño
    - Propuesta de mejora – Medidas a implantar – Medidas relativas a la contratación.
    - ANEXO I – Normativa de interés
- De 2 a 29 de abril de 2019.
    - Propuesta de mejora – Medidas a implantar
        - Medidas relativas a la metodología de desarrollo
        - Medidas relativas a la gestión de Recursos Humanos
        - Medidas organizativas
- De 30 de abril a 28 de mayo de 2019
    - Propuesta de mejora
        - Medidas a implantar – Medidas relativas a la tecnología
        - Planificación temporal
        - Otros ámbitos de aplicación de metodologías ágiles
- 29 mayo. Revisión final del TFM.

Todo *Sprint* finalizaba y empezaba con una reunión presencial, en la que se revisaba el trabajo, se decidían modificaciones sobre lo realizado y se proponían nuevos elementos para el siguiente *Sprint*.

6.4.2 Extendiendo las metodologías ágiles a otros contextos.

Aunque las metodologías ágiles surgieron en el marco del desarrollo de *software*, su aplicación se puede extender a otros contextos con beneficios similares.

Para poder aplicar este marco de trabajo ágil se deben dar, en mayor o menor grado, ciertas condiciones, que se producen habitualmente en el desarrollo de *software*:

- Coste de la reversibilidad o de modificar el trabajo ya realizado es bajo.
- El coste de fallar no es elevado. Una de las premisas de las metodologías ágiles es fallar rápido para fallar barato y aprovechar el conocimiento adquirido para adaptarse mejor. Sin embargo, si de entrada el coste del fallo es elevado, esto puede suponer un problema importante.
- No se tiene claro el problema ni la solución. Esto no es una condición imprescindible, pero los beneficios de aplicar una metodología ágil serán mayores cuanto mayor sea la incertidumbre y la necesaria adaptación.
- La solución es fácilmente divisible y se puede entregar en incrementos. Es decir que ciertos "trozos" de la solución pueden aportar valor y pueden servir para aprender. Si la solución se debe entregar completa no se puede aplicar un enfoque incremental que es esencial en las metodologías ágiles. En este punto conviene apuntar que muy rara es la solución que no aporta nada al entregar una parte. Esta condición suele cumplirse siempre, aunque por una cuestión de mentalidad y creencias, a veces, se tiene la percepción de que no se cumple.
- Relacionado con el anterior, las historias de usuario deben poder ser divisibles en elementos que puedan ser entregados en un *Sprint*. Esta condición no es estricta, es posible aplicar metodologías ágiles y que alguna historia de usuario no pueda entregarse en un único *Sprint*, o que el producto mínimo viable requiera más de un *Sprint*, pero no debe ser lo habitual, sino la excepción.
- El equipo necesario debe tener perfiles versátiles. En el desarrollo de *software* se consigue con personal altamente cualificado y con experiencia en diferentes ámbitos; sin embargo, en otros contextos, puede ser más complicado de conseguir debido a la especialización estricta del personal que participa en el

equipo. Esta condición también se cumple fácilmente, aunque en ocasiones tengamos una creencia contraria.

7 BIBLIOGRAFÍA

Aljure, A. (2016). *El plan estratégico de comunicación: Método y recomendaciones prácticas para su elaboración.* UOC.

Ashmore, S., & Runyan, K. (2014). *Introduction to Agile Methods.* Addison-Wesley Professional.

Beck, K. (2003). *Test-driven Development: By Example.* Addison-Wesley Professional.

Beck, K., Grenning, J., Martin, R. C., Beedle, M., Highsmith, J., Mellor, S., . . . Marick, B. (2001). *Manifesto for Agile Software Development*. Recuperado el diciembre de 2018, de https://agilemanifesto.org/

Cohn, M. (2004). *User Stories Applied: For Agile Software Development.* Addison-Wesley Professiona.

Concas, G., Damiani, E., Scotto, M., & Succi, G. (2007). *Agile Processes in Software Engineering and Extreme Programming: 8th International Conference,.*

Consejo Superior de Informatica. (2001). *Métrica V.3.* Obtenido de https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html

Irrazábal, E., & Garzás, J. (2010). Análisis de métricas básicas y herramientas de código libre para medir la mantenibilidad. *Revista Española de Innovación, Calidad e Ingeniería del Software*.

Islam, K. A. (2013). *Agile Methodology for Developing & Measuring Learning: Training Development for Today'S Worl.* AuthorHouse.

Kniberg, H. (2011). *Lean from the Trenches.* The Pragmatic Programmers.

Kniberg, H. (2015). *Scrum and XP from the Trenches.* C4Media.

Lynn Cooke, J. (2014). *Agile Productivity Unleashed: Proven approaches for achieving real productivity gains in any organization.* IT Governance Publishing.

Ministerio de Hacienda. (2019). *Contratacion Centralizada*. Obtenido de https://contratacioncentralizada.gob.es/ficha-am/-/journal_content/XXA1X8YVROqE?_56_INSTANCE_XXA1X8YVROqE_articleId=15309&_56_INSTANCE_XXA1X8YVROqE_groupId=11614

Ministerio de Hacienda y Administraciones Publicas. (2012). *Instrucciones sobre buenas prácticas para la gestión de las contrataciones de servicios y encomiendas de gestión a fin de evitar incurrir en supuestos de cesion ilegal de trabajadores.*

Pérez Pérez, M. J. (2012). *Guía comparativa de metodologías ágiles.* Segovia: Universidad de Valladolid. Escuela Universitaria de Informática. Obtenido de http://uvadoc.uva.es/handle/10324/1495

Schwaber, K., & Sutherland, J. (Noviembre de 2017). *Scrum Guides.* Obtenido de https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

Sutherland, J. (2015). Plan Reality, Not Fantasy. En *The Art of Doing Twice the Work in Half the Time* (págs. 111-144).

Sutherland, J. (2015). Priorities. En *The Art of Doing Twice the Work in Half the Time* (págs. 171-202).

Sutherland, J. (2015). Teams. En *The Art of Doing Twice the Work inm Half the Time* (págs. 41-70).

Sutherland, J. (2015). Waste is a Crime. En *The Art of Doing Twice the Work in Half the TIme* (págs. 85-110).

Vadapalli, S. (2018). *DevOps: Continuous Delivery, Integration, and Deployment with DevOps: Dive into the core DevOps strategies.* Packt Publishing Ltd.

8 ANEXOS

ANEXO I – Normativa de interés

A continuación, se menciona la diversa normativa que resulta de interés y ha sido estudiada en mayor o menor detalle para la elaboración de este Plan de Mejora:

- Normativa en materia de contratación:
    - Ley 9/2017, de 8 de noviembre, de Contratos del Sector Público (LCSP).
    - Real Decreto 817/2009, de 8 de mayo, por el que se desarrolla parcialmente la Ley 30/2007, de 30 de octubre, de Contratos del Sector Público.
    - Real Decreto 1098/2001, de 12 de octubre, por el que se aprueba el Reglamento general de la Ley de Contratos de las Administraciones Públicas.
    - Orden EHA/1049/2008, de 10 de abril, de declaración de bienes y servicios de contratación centralizada.
- Acuerdo Marco 26/2015 sobre Servicios de desarrollo de sistemas de Administración electrónica, y sus documentos asociados:
    - Pliego de cláusulas administrativas particulares que rige la celebración del acuerdo marco para los servicios de desarrollo de sistemas de Administración electrónica (AM 26/2015).
    - Pliego de prescripciones técnicas que rige la celebración del acuerdo marco para la contratación de servicios de desarrollo de sistemas de Administración electrónica.
    - Instrucciones para la tramitación de las segundas licitaciones en el acuerdo marco 26/2015.
- Instrucciones sobre buenas prácticas para la gestión de las contrataciones de servicios y encomiendas de gestión a fin de evitar incurrir en supuestos de cesión ilegal de trabajadores.

ANEXO II – *Scrum*

Introducción

Scrum es un marco de trabajo mediante el cual, un equipo puede abordar problemas complejos y cambiantes, a la vez que generan productos del mayor valor posible, de un modo productivo y creativo. Este marco es ligero, fácil de entender, pero muy difícil de dominar. (Schwaber & Sutherland, 2017)

Este marco consta de equipos *Scrum* y sus roles, eventos, artefactos y reglas asociadas. Cada componente sirve un propósito específico y es esencial para el éxito. Las reglas de *Scrum* vinculan a estos elementos y gobiernan las relaciones entre ellos. (Schwaber & Sutherland, 2017).

A lo largo del presente apartado se indicarán características deseadas para el proceso, roles, eventos y artefactos de *Scrum*. Estas características son las óptimas para que el proceso obtenga los mejores resultados, pero debido a las características de las organizaciones donde se implanta puede que no todas ellas se puedan cumplir al 100%.

Pilares

El funcionamiento y el valor de *Scrum* se fundamenta en tres pilares, sin los cuales no se obtienen los resultados deseados:

- Transparencia. Los aspectos más significativos del proceso de desarrollo de *software* deben permanecer visibles para todos los responsables del resultado. (Schwaber & Sutherland, 2017). Esto requiere que se establezcan unos criterios comunes para que la información pueda ser correctamente interpretada por todas las partes.
- Inspección/Revisión. El equipo debe revisar con cierta frecuencia los diferentes artefactos y el progreso realizado hacia el Objetivo del *Sprint*. Esta revisión no debe suponer un impedimento a la realización del trabajo real, por lo que su frecuencia debe ser controlada.
- Adaptación. Fruto de la revisión se pueden detectar desviaciones no aceptables. Debe procederse a adaptar el proceso y el producto para lograr el resultado deseado. La adaptación debe producirse lo antes posible para minimizar el coste de la corrección necesaria (Sutherland, Waste is a Crime, 2015). *Scrum* establece

4 eventos formales para la revisión y la adaptación como son el *Sprint Planning*, *Daily Scrum*, *Sprint Review* y *Sprint Retrospective.*

El equipo *Scrum* – roles

El equipo *Scrum* consta de un *Product Owner*, un equipo de desarrollo y un *Scrum Master*.

Los equipos *Scrum* están auto-organizados y son multi-disciplinares[11]. Un equipo auto-organizado escoge el mejor modo de realizar su trabajo sin ser dirigido por otras personas fuera del equipo. Un equipo multi-disciplinar tiene todas las competencias necesarias para lograr los resultados deseados sin depender de otras personas que no son parte del equipo. (Schwaber & Sutherland, 2017) (Sutherland, Teams, 2015)

Aunque no son dirigidos por nadie externo al equipo, los equipos *Scrum*, deben tener un propósito o una visión de su trabajo que les inspira y motiva (Sutherland, Teams, 2015), y que, la mayoría de las veces, es definida por la organización o por personas externas al equipo.

Este tipo de equipos auto-organizados y multi-disciplinares, son ideales que muchas veces no se pueden producir en la práctica debido a las características intrínsecas de las organizaciones. Pero son conceptos que permiten identificar las características que deben tener los equipos en *Scrum*, para lograr un rendimiento óptimo; y marcarán la tendencia a la hora de proponer medidas y actuaciones para aplicar *Scrum*.

El *Product Owner* (Propietario del producto)

El *Product Owner* es el responsable de maximizar el valor del producto resultante del trabajo del equipo de desarrollo. Es una única persona, aunque puede representar a un comité o a un conjunto de interesados. Solo puede haber un *Product Owner*, cuyas decisiones deben ser respetadas en toda la organización.

[11] Del inglés *cross-functional*. Se ha escogido el término multi-disciplinar por ser el más usado en la bibliografía en castellano.

El *Product Owner* es el único responsable de gestionar el *Product Backlog*, el cual debe permanecer visible para el equipo de desarrollo y el resto de la organización, siguiendo el principio de transparencia de *Scrum*. (Schwaber & Sutherland, 2017).

La gestión del *Product Backlog* incluye:

- Expresar con claridad los elementos del *Product Backlog*.
- Ordenar los elementos para lograr los objetivos deseados del mejor modo posible.
- Asegurarse que el *Product Backlog* es visible y claro para todos. Y que muestra lo que el equipo de desarrollo está haciendo y lo que hará.
- Asegurarse que el equipo de desarrollo entiende perfectamente los elementos indicados en el *Product Backlog*.

Para cumplir con su labor adecuadamente, el *Product Owner* debe cumplir una serie de características esenciales (Sutherland, Priorities, 2015):

- Debe conocer el dominio o negocio asociado al producto. Esto significa que debe entender el proceso de trabajo del equipo de desarrollo para saber qué se puede hacer y qué no; Y saber qué elementos del producto aportarán más valor a los diferentes interesados. Esto implica un dialogo constante con los interesados y los usuarios finales del producto.
- Debe tener capacidad y autoridad para tomar decisiones, sin interferencias externas, dado que será responsable del impacto de los resultados del equipo. Nadie debe poder forzar al equipo de desarrollo a trabajar en algo diferente de lo indicado por el *Product Owner*.
- Debe estar disponible para el equipo de desarrollo, De esta se puede asegurar que el equipo tiene claro lo que hay que hacer y por qué. Y resolver las dudas que surjan con agilidad y diligencia.
- Es necesario que sea responsable del valor aportado por el equipo, en los términos del dominio/negocio que maneje.

El development team (equipo de desarrollo)

El equipo de desarrollo consta de un conjunto de profesionales que trabajan para entregar una versión incremental del producto al final de cada *Sprint* (Schwaber & Sutherland, 2017). En la *Sprint Review*, es necesario tener una versión del producto entregable que aporte valor y permita obtener *feedback* sobre el producto.

Las características del equipo de desarrollo son las siguientes:

- Son auto-organizados, es decir, son autónomos. Nadie (ni el *Scrum Máster*) puede decir al equipo de desarrollo como convertir el *Product Backlog* en un incremento de producto que puede entregue valor.
- Son multi-disciplinares. Con todas las habilidades necesarias como equipo para producir un incremento del producto.
- A pesar de que cada miembro del equipo de desarrollo pueda tener unas habilidades y conocimientos la responsabilidad por el resultado recae sobre el equipo como un todo.
- Tienen un propósito. Los equipos de desarrollo conocen la razón y el impacto de su trabajo.

El tamaño del equipo de desarrollo debe ser suficientemente pequeño para ser ágil, y suficientemente grande para entregar valor en un *Sprint*; esto se traduce en un tamaño de entre 3 y 9 miembros, aunque lo ideal es que sean entre 5 y 7 (Sutherland, Teams, 2015).

Equipos de menos de 3 personas pueden tener problemas para tener todas las habilidades necesarias y para aprovechar las sinergias derivadas del trabajo en equipo. Equipos de más de 9 miembros generan un trabajo excesivo de coordinación lo cual disminuye la productividad del equipo. En este recuento de miembros no se tiene en cuenta ni al *Product Owner* ni al *Scrum Master*, a no ser que también lleven a cabo trabajo del *Product Backlog*.

El *Scrum* Máster

El *Scrum Máster* es el encargado de apoyar y promocionar la implantación de *Scrum*. Logra esto por medio la enseñanza de la teoría, práctica, reglas y valores de *Scrum* a todos los integrantes de la organización (Schwaber & Sutherland, 2017).

Además, actúa como líder-sirviente del equipo de desarrollo. Ayuda a todo el personal ajeno al equipo de desarrollo a entender que interacciones con el equipo *Scrum* son productivas y cuáles no y promueve la modificación de esas interacciones para maximizar el valor creado por el equipo *Scrum* (Schwaber & Sutherland, 2017).

Relacionado con la responsabilidad citada anteriormente, entre las tareas y responsabilidades del *Scrum Máster* se encuentran las siguientes:

- Asegurar que todos los componentes del equipo *Scrum* entiende a la perfección el objetivo, el alcance y el dominio del producto.
- Encontrar técnicas para una gestión efectiva del *Product Backlog.*
- Ayudar al equipo a entender la necesidad de crear elementos del *Product Backlog* lo más claros y concisos posibles.
- Asegurarse que el *Product Owner* sabe cómo ordenar el *Product Backlog* para maximizar el valor entregado.
- Entender y practicar los principios y valores ágiles.
- Facilitar los eventos *Scrum* siempre que necesario.
- Ayudar al equipo de desarrollo en mejorar la auto-organización y la multi-disciplina.
- Ayudar al equipo de desarrollo a crear productos de gran valor.
- Quitar los obstáculos o impedimentos que dificulten el progreso del equipo de desarrollo.
- Apoyar al equipo de desarrollo en entornos organizativos donde *Scrum* no está completamente implantado.
- Ayudar al equipo de desarrollo a eliminar el desperdicio (Sutherland, Waste is a Crime, 2015).

Eventos *Scrum*

El *Sprint*

El *Sprint* es el corazón de la metodología *Scrum* y es el marco temporal en el que se desarrolla la metodología (Schwaber & Sutherland, 2017). Se trata de un periodo de

tiempo, de como máximo un mes, en el cual el equipo de desarrollo trabaja para producir un incremento del producto que aporte valor.

La duración de los *Sprint* debe ser fija y definida previamente por parte del equipo *Scrum*, en base a diferentes factores como la complejidad del producto, las condiciones de trabajo, el tamaño del equipo, la experiencia del equipo etc. Esta duración debe establecerse de forma que sea lo más pequeña posible, pero sea suficiente para generar un incremento que aporte valor. La duración no debe ser inferior a 2 semanas ni superior a 4 semanas.

Cuando se termina un *Sprint*, se revisa el resultado y el proceso, se comienza otro inmediatamente, aunque algunos autores recomiendan dar un cierto margen para aplicar las mejoras correspondientes al proceso de desarrollo (Kniberg, Scrum and XP from the Trenches, 2015).

Las historias de usuario que se van a implementar y el objetivo del *Sprint* se seleccionan en el *Sprint Planning* y no se pueden realizar modificaciones en el mismo que puedan poner en peligro la consecución de la meta.

En el caso extremo que, por cambios radicales e inesperados, el objetivo del *Sprint* pase a carecer de sentido, se puede producir la cancelación del mismo.

La duración limitada de máximo 1 mes, favorece la adaptación a los cambios y minimiza el riesgo, ya que se limita a 1 mes o menos el potencial desperdicio.

Sprint Planning (Planificación del *Sprint*)

Este evento es el que da comienzo al *Sprint*. En el *Sprint Planning*, partiendo del *Product Backlog*, se especifica lo que se hará en el *Sprint* que va a dar comienzo.

En esta reunión intervienen el equipo *Scrum* al completo (equipo de desarrollo, *Product Owner* y *Scrum Máster*), que de forma colaborativa planificarán el *Sprint*.

La duración de este evento es de aproximadamente 2 horas por cada semana de duración del *Sprint* (Schwaber & Sutherland, 2017), con sus correspondientes descansos cada hora.

Para que este evento no se alargue demasiado, se recomienda realizar un adecuado refinamiento de los elementos más prioritarios del *Product Backlog* de forma previa al *Sprint Planning* (Kniberg, Scrum and XP from the Trenches, 2015).

El *Sprint Planning* se estructura en 2 partes, para responder a 2 preguntas, ¿qué se va a entregar al final del *Sprint*? y ¿cómo se va a trabajar?

¿Qué se va a entregar al final del Sprint?

En esta primera parte, el *Product Owner* determina el objetivo del *Sprint* desde un punto de vista de negocio. Es decir, qué se pretende lograr durante el *Sprint* desde un punto de vista de negocio. Este objetivo permite dotar de propósito al trabajo del equipo de desarrollo y de cierta flexibilidad para cumplir ese objetivo. Se recomienda definir un objetivo aunque no sea muy inspirador, ya que es mejor tener un objetivo mediocre que no tenerlo en absoluto (Kniberg, Scrum and XP from the Trenches, 2015).

Adicionalmente, el *Product Owner* establece las historias de usuario/elementos del *Product Backlog*, que permitirán cumplir con ese objetivo y las ordena de mayor a menor prioridad.

El equipo de desarrollo, estima los elementos seleccionados por el *Product Owner* y determina lo que se podrá llevar a cabo en el *Sprint*. La cantidad de elementos que se incluirán se basará en la velocidad de los *Sprint* anteriores o en la intuición del equipo de desarrollo para los primeros *Sprint*.

El equipo de desarrollo es el encargado de estimar los elementos y por tanto de decir qué puede entrar en el *Sprint*; el *Product Owner* puede modificar el alcance de algunos elementos para reducir su estimación (en base a reducir la complejidad) y de este modo incluir elementos en el *Sprint* que le interesen.

Algunos autores recomiendan que se negocie siempre con el alcance de un elemento y que la calidad del trabajo (como índice de mantenibilidad del producto final) sea innegociable (Kniberg, Scrum and XP from the Trenches, 2015).

¿Cómo se va a trabajar?

El equipo de desarrollo desglosa los elementos del *Sprint Backlog* seleccionados en la parte anterior en tareas.

Se asignan las tareas para los primeros días de trabajo y se establecen las condiciones de trabajo del equipo de desarrollo (lugar de celebración y hora de la *Daily*, canales de comunicación, etc.) que le permiten trabajar como un equipo auto-organizado.

Estimación de los elementos – Planning Póker

Una parte muy importante del *Sprint Planning* es la estimación de los elementos que se van a incluir en el *Sprint*. Normalmente se emplea mucho tiempo para estimar los elementos con el mayor detalle posible. Esto, además del tiempo necesario, da una falsa sensación de certidumbre y seguridad en las estimaciones. Por lo que se recomienda utilizar métodos de estimación por aproximación y comparación, que aportan gran cantidad de información sobre el tamaño del trabajo a realizar y necesitan poco tiempo para su realización (Sutherland, Plan Reality, Not Fantasy, 2015).

Uno de los métodos más conocidos de estimación aplicados en las metodologías ágiles es el llamado *Planning* Póker (Kniberg, Scrum and XP from the Trenches, 2015). Aunque existen diferentes variantes que cambian en algunos detalles, en este método, cada miembro del equipo de desarrollo estima de forma independiente y en secreto cada elemento y en caso de discrepancias elevadas se inicia un debate entre los miembros del equipo sobre el elemento en cuestión.

Para hacerlo, cada miembro del equipo dispone de un mazo de cartas con números siguiendo una secuencia de *Fibonacci* (1, 2, 3, 5, 8, 13, 20, 40, 100). Para cada elemento del *Product Backlog* a estimar, cada miembro del equipo de desarrollo saca una única carta, sin ver la que han sacado los demás. Si hay una diferencia de más de 2 cartas entre miembros del equipo (por ejemplo, uno saca 3 y otro 13), se inicia un debate, ya que es posible que cada uno lo está interpretando de un modo diferente. El proceso se repite hasta que no hay diferencias tan grandes y simplemente se calcula la media.

La unidad de medida habitual en las metodologías ágiles es puntos de historia, pero se puede utilizar cualquier unidad que permita comparar tamaños de diferentes elementos

y, al finalizar el *Sprint*, calcular la velocidad del equipo, que no es más que la cantidad de trabajo que se ha realizado en un *Sprint*, medido en la unidad que el equipo haya determinado (pueden ser puntos historia u otra medida). Hay autores que recomiendan incluso no hacer estimaciones en absoluto, sino reducir los elementos del *Product Backlog* lo máximo posible y solo medir los elementos producidos.

Daily *Scrum* (*Scrum* Diario)

El *Daily Scrum* es una reunión de un tiempo fijo y máximo de 15 minutos de duración, que tiene lugar todos los días a la misma hora y lugar (Schwaber & Sutherland, 2017).

Durante esta reunión, en la que participa el *Scrum Team* al completo, se planifica el trabajo para las próximas 24 horas y se inspecciona el progreso del equipo hacia el objetivo del *Sprint*, definido en el *Sprint Planning*.

Es una reunión de trabajo, donde el objetivo fundamental es favorecer la comunicación y la difusión del conocimiento dentro del equipo de desarrollo.

Es el equipo de desarrollo el que define la estructura de la reunión, aunque lo más habitual es que cada miembro del equipo de respuesta a las 3 siguientes preguntas:

- ¿Qué hice ayer para cumplir el objetivo del *Sprint*?
- ¿Qué voy a hacer hoy para cumplir el objetivo del *Sprint*?
- ¿Detecto algún impedimento que nos pueda evitar cumplir el objetivo del *Sprint*?

Cualquier persona, con interés en el proyecto, puede asistir a esta reunión, dado que la transparencia es un pilar fundamental de la metodología *Scrum*, pero el *Scrum Máster* debe asegurarse que no interrumpen al equipo, de hecho algunos autores afirman que no deben si quiera intervenir (Kniberg, Scrum and XP from the Trenches, 2015).

Asimismo, el *Scrum Máster* debe asegurarse que esta reunión se produce y que se desarrolla dentro de los límites temporales establecidos.

Para favorecer que la *Daily Scrum* no se alargue más de los 15 minutos fijados, es una buena práctica realizaras de pie.

En esta reunión se inspeccionan y se actualizan los artefactos asociados al *Sprint* (*Sprint Backlog* y *Burndown Charts*).

Sprint Review (Revisión del *Sprint*)

La *Sprint Review* se produce el final de cada uno de los *Sprint*, y tiene como objetivos fundamentales revisar el incremento de producto entregado, obtener *feedback* sobre el mismo y adaptar el *Product Backlog* en base al *feedback* recibido de esta reunión (Schwaber & Sutherland, 2017).

En esta reunión participa el *Scrum team* al completo, así como todos aquellos s*takeholder*s del proyecto que deseen asistir.

No es una reunión de control, sino de presentación del trabajo realizado durante el *Sprint* con el objetivo de obtener información valiosa que pueda ser utilizado en los siguientes *Sprint*.

La duración máxima de esta reunión, para un *Sprint* de 1 mes, se establece en 4 horas, y el *Scrum Máster* debe asegurarse que este evento se produce y que dura el tiempo establecido (Schwaber & Sutherland, 2017). Una buena práctica es establecer una duración máxima de 1 hora por semana de duración del *Sprint***.**

El guion de la reunión puede ser el siguiente:

- Si asiste alguien no familiarizado con el producto, el *Product Owner* hacer una breve introducción del mismo.
- El *Product Owner* explica los elementos del *Product Backlog* que se han hecho y los que no.
- El equipo de desarrollo hace una demostración práctica sobre el incremento entregado y responde preguntas sobre el mismo. Para ello se llevan a cabo los criterios de Aceptación definidos para cada historia de usuario.
- Se debate sobre los elementos existentes en el *Product Backlog*, para obtener información valiosa para el siguiente *Sprint Planning*.

El resultado del *Sprint Review* es un *Product Backlog* revisado y actualizado que define los elementos del *Product Backlog* más útiles para el siguiente *Sprint*.

En algunas organizaciones, esta reunión está abierta a cualquier persona de la organización, pero solo pueden intervenir los miembros del *Scrum* Team y los *Stakeholder*s (Kniberg, Scrum and XP from the Trenches, 2015).

Una práctica habitual es prohibir las presentaciones gráficas tipo *PowerPoint*, ya que toda presentación debe basarse en el incremento real entregado. De este modo, el foco se hace en el producto, que se puede ver y resolver dudas directamente sobre el mismo.

Sprint Retrospective (Retrospectiva)

La *Sprint Retrospective* se produce al finalizar el *Sprint* después de haber realizado el *Sprint Review* y antes del siguiente *Sprint Planning* (Schwaber & Sutherland, 2017).

En ella participa el *Scrum* Team al completo, *Product Owner*, *Development Team* y *Scrum Máster*.

La duración máxima es de 3 horas para un *Sprint* de 1 mes, por lo que puede establecerse la regla de una duración de 45 minutos por semana de duración del *Sprint*.

El *Scrum* Máster debe garantizar que el evento tiene lugar, que los asistentes entienden el propósito y la importancia de la *Sprint Retrospective*, y que el evento es productivo y positivo (Schwaber & Sutherland, 2017).

Además, el *Scrum Máster*, o alguien en quien delegue, actuará como secretario de la reunión, para dejar constancia de lo acordado (Kniberg, Scrum and XP from the Trenches, 2015).

El objetivo de la *Sprint Retrospective* es revisar cómo fue el *Sprint*, no desde un punto de vista del resultado que eso se analiza en la *Sprint Review*, sino desde un punto de vista de las personas, las relaciones entre ellas, las herramientas y los procesos. Y, en base a esa revisión, identificar potenciales mejoras a aplicar y crear un plan para implementar esas mejoras en la forma de trabajo del *Scrum Team*.

Aunque las mejoras se pueden aplicar en cualquier momento y a iniciativa de cualquier miembro del *Scrum* Team, la retrospectiva da una oportunidad formal para centrarse en la revisión, adaptación y mejora. Además, favorece que los cambios sean aceptados por

el equipo, al ser propuestos, comentados, desarrollados y acordados por el equipo al completo de forma conjunta.

Un ejemplo de agenda de una reunión de retrospectiva puede tener los siguientes puntos (Kniberg, Scrum and XP from the Trenches, 2015):

- El *Scrum* Máster hace un resumen del *Sprint*, con ayuda de todos, indicando los eventos, decisiones, etc. que han tenido especial relevancia.
- Se hacen rondas en las que, cada miembro del equipo tiene la oportunidad de dar su opinión sobre lo que se hizo bien, lo que podría haberse hecho mejor y lo que le gustaría hacer diferente en el siguiente *Sprint*.
- Se revisa la velocidad estimada y la velocidad real, en caso de haber una gran diferencia se buscan las causas.
- Cuando quedan pocos minutos para finalizar el tiempo asignado, el *Scrum* Máster resume las sugerencias concretas que se han realizado.
- El equipo decide por votación las mejoras o sugerencias que se aplicarán.

Artefactos *Scrum*

Historias de usuario

Se denominan historias de usuario a una descripción breve de una funcionalidad *software* que aporta valor (Cohn, 2004). Las historias de usuario constan básicamente de 3 elementos:

- Descripción de la historia.
- Conversaciones sobre la historia que han servido para clarificarla.
- Criterios de aceptación. Pruebas y acciones que se llevarán a cabo para determinar cuándo una historia se ha completado.

Es una forma de describir los requisitos del *software* desde un punto de vista del usuario, se usan tradicionalmente en las metodologías ágiles, ya que son una herramienta muy potente de comunicación.

Existen diferentes técnicas para desarrollar la descripción de las mismas, una de las más famosas utiliza el siguiente formato:

Como <rol> quiero <funcionalidad> para <objetivo>

Definir de este modo las historias de usuario, y por tanto los requisitos, permite, por un lado, poner al usuario de la aplicación en primer lugar y, por otro, conocer el objetivo que se busca con el desarrollo de la funcionalidad.

Product Backlog (Pila del producto)

El *Product Backlog* es una lista ordenada de las características, funcionalidades, requisitos, mejoras y correcciones que se van a hacer al producto. Es la única fuente de requisitos y es responsabilidad única del *Product Owner* (Schwaber & Sutherland, 2017).

El *Product Owner* es responsable de generar el *Product Backlog*, en un momento previo a la primera reunión de *Sprint Planning*.

El *Product Backlog* es un elemento dinámico y está en cambio constante. Evoluciona a la vez que lo hace el producto desarrollado y el entorno.

Los elementos del *Product Backlog* tienen típicamente los siguientes atributos:

- Descripción.
- Orden. Se prefiere el orden a la prioridad, ya que muchos elementos pueden tener la misma prioridad, pero el orden debe ser único para cada elemento (Kniberg, Scrum and XP from the Trenches, 2015).
- Estimación. Esfuerzo necesario para desarrollar el elemento.
- Valor. Valor que aporta el desarrollo del elemento desde un punto de vista de negocio.
- Definition of Done. Descripción de cuándo se da por terminado el elemento. O también como se va a probar en el *Sprint Review*, el elemento (Kniberg, Scrum and XP from the Trenches, 2015).
- Equipo de trabajo. Cuando varios *Scrum* Team trabajan sobre el mismo *Product Backlog*, puede ser necesario un atributo que permita agrupar los elementos que va a realizar cada uno de los equipos (Schwaber & Sutherland, 2017).

A lo largo del proceso de desarrollo, se produce lo que se denomina el refinamiento del *Product Backlog*. Este proceso consiste en añadir o cambiar información, detalles, estimaciones y el orden de los diferentes elementos. Es un proceso continuo durante el cual los elementos se revisan y completan. El *Scrum* Team debe decidir cuándo se realiza esta tarea, que no debe llevar más del 10% de la capacidad del equipo, aunque el *Product Owner*, como responsable del *Product Backlog*, puede actualizar elementos del mismo en cualquier momento.

Fruto de este refinamiento, los primeros elementos (en base a su atributo orden), estarán más detallados y claros. Los elementos del *Backlog* que vayan a desarrollarse en el siguiente *Sprint* deben estar suficientemente claros, de forma que se puedan realizar en el *Sprint*.

A estos elementos del *Backlog* que están lo suficientemente claros para ser ejecutados en un *Sprint* se dice que están en estado "*Ready*". Este estado se alcanza por medio de refinamientos sucesivos.

El equipo de desarrollo es el responsable de todas las estimaciones. El *Product Owner* puede influir ayudando a entender el elemento o simplificándolo, pero son las personas que van a hacer el trabajo los que realizan la estimación final.

Sprint Backlog (Pila del *Sprint*)

El *Sprint Backlog* consta de los elementos del *Product Backlog* que han sido seleccionados para ser desarrollados en el *Sprint* (Schwaber & Sutherland, 2017). Esta tarea se realiza en el *Sprint Planning*.

Los elementos seleccionados, deben permitir cumplir con el objetivo del *Sprint* y lograr un incremento del producto.

El equipo de desarrollo es el encargado de prever la funcionalidad que se incluirá en el siguiente incremento, que permita conseguir el objetivo del *Sprint*.

Para garantizar la mejora continua en el proceso de desarrollo, debe incluir, al menos, un elemento de mejora que se haya detectado en la retrospectiva.

A medida que va realizando el trabajo y aprendiendo, el equipo de desarrollo puede modificar el *Sprint Backlog* para asegurarse que consigue el objetivo del *Sprint*. Solo el equipo de desarrollo puede modificar el *Sprint Backlog* durante un *Sprint*.

El *Sprint Backlog* es una fotografía en tiempo real del trabajo que el equipo de desarrollo ha planeado que va a ejecutar durante el *Sprint*.

Los elementos del *Sprint Backlog* deben estar suficientemente definidos y ser de un tamaño reducido, para poder monitorizar el avance diariamente durante los *Daily Scrum*.

Burndown Charts (Gráficos de evolución)

Los gráficos de evolución o *Burndown Charts* son elementos visuales para poder monitorizar el progreso del trabajo a lo largo de un *Sprint*.

Se actualizan diariamente, durante el *Daily Scrum*, y en ellos se establece en el eje de ordenadas el trabajo que queda pendiente (en las unidades que el *Scrum* Team haya determinado que se realizan las estimaciones) y en el eje de abscisas los días que han transcurrido del *Sprint*.

Idealmente, la curva del gráfico debe ser tal que el esfuerzo pendiente al finalizar el *Sprint* sea 0, pero errores en la estimación o imprevistos pueden hacer que no sea así, o que incluso se acabe el trabajo a realizar incluso antes del *Sprint*.

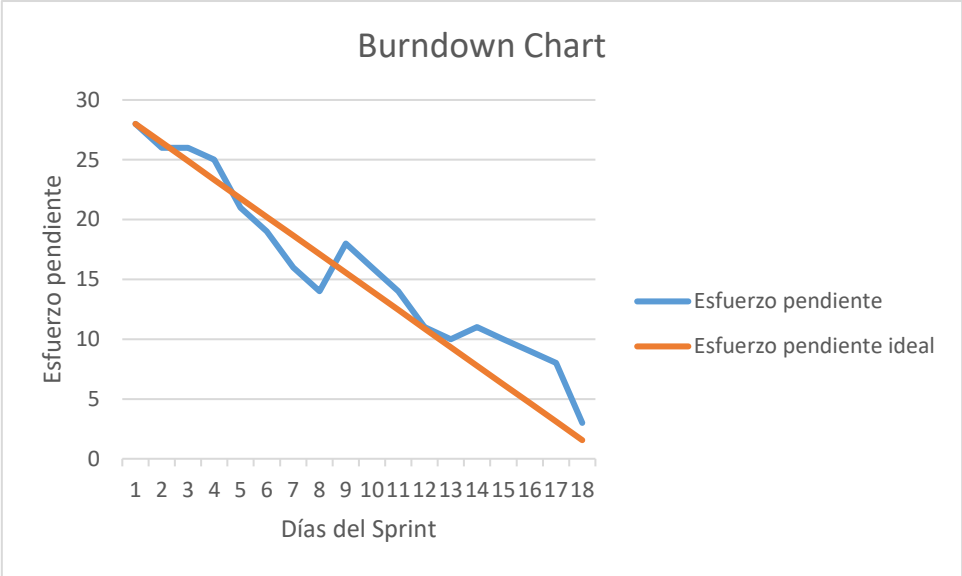

Ilustración 4. Ejemplo de Burndown Chart

Incremento

Se denomina incremento a la suma de todos los elementos del *Product Backlog* completados durante un *Sprint*. (Schwaber & Sutherland, 2017).

Al final de cada *Sprint*, el nuevo incremento debe estar en condiciones de ser usado y cumplir la definición de hecho del *Scrum Team*.

El incremento supone un paso más hacia la visión o meta y debe estar en condiciones de ser usado con independencia de que el *Product Owner* decida desplegarlo en producción o no.