Trabajo Fin de Máster
Máster Universitario en Ingeniería Industrial

Aplicación de Metodologías Ágiles a Desarrollo de Proyectos

Autor: Jesús Santiago Rial Huerta
Tutor: Guillermo Montero Fernández-Vivancos

Dpto. Organización Industrial y Gestión de Empresas II
Escuela Técnica Superior de Ingeniería

Sevilla, 2019

Proyecto Fin de Máster
Máster en Ingeniería Industrial

Aplicación de Metodologías Ágiles a Desarrollo de Proyectos

Autor:
Jesús Rial Huerta

Tutor:
Guillermo Montero Fernández-Vivancos
Profesor titular

Dpto. Organización Industrial y Gestión de Empresas II
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Proyecto Fin de Carrera: Aplicación de Metodologías Ágiles a Desarrollo de Proyectos

Autor: Jesús Rial Huerta

Tutor: Guillermo Montero Fernández-Vivancos

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

Llegar aquí fue un sueño,
hasta hoy...

Agradecimientos

Es difícil, acordarse de tanta gente cuando uno emprende un camino muy largo. Un camino, que comencé en septiembre de 2006, cuando decidí cruzar "el charco" (como solía llamar a la distancia que separa Ceuta de la península) para venir a Sevilla. Una Sevilla, de la que me tuve que despedir: aceptar que no estaba preparado para afrontar todo lo que se requería para realizar esta carrera fue lo más difícil, y durante varios años (siete) me costó entenderlo, los mismos siete que pasé en Córdoba aceptando lo que para mí era una derrota, pero no la partida.Creo que cada una de las personas que han vivido, que han convivido conmigo, o con las que he coincidido en todo este tiempo, han dejado una huella en mí, y me es inevitable no recordar a todas aquellas personas que se han ido yendo durante todos estos años. A todos ellos: ***gracias****.*

Quiero darle las ***gracias*** *también a mi tutor Guillermo Montero, por su tiempo, dedicación y buenos consejos para la realización de este Trabajo Fín de Máster.*

También quiero darle las gracias a mi familia, ese pilar fundamental sin el cual nada de esto tendría sentido, y que siempre han guiado mi camino: Miguel, Sonsoles, Diego, ***gracias*** *por estar siempre ahí.*

También quiero agradecer a mi segunda familia, ese otro apoyo que ha hecho de este viaje un camino más llevadero: Leonor, Alfonso, Alfonsito, ***gracias*** *por todo lo que habéis hecho en todo este tiempo. Alfonso, va por ti allá donde estés.*

Por último, quiero acordarme de la hija de esta segunda familia, que sacó de mí el hombre que soy hoy, la que me motivó a terminar aquelló que empecé y por la que hoy estoy aquí defendiendo este trabajo: Cristina, te quiero y ***gracias*** *por estar siempre ahí.*

Jesús Rial Huerta

Sevilla, 2019

Resumen

Es un hecho que los proyectos, entendiendo por proyecto a todo aquello que se hace de forma planificada, atendiendo a unos criterios de orden, planificación, y ejecución, lleva con nosotros desde el origen de nuestros tiempos (desde el levantamiento de las pirámides de Egipto, la construcción de la Gran Muralla China, o el Arca de Noé).

Sin embargo, la metodología de proyectos, tal como la conocemos hoy en día, surge hacia mediados del siglo XX de la mano de Henry Gantt (1861-1919), Frederick Winslow Taylor (1856-1915), Henri Fayol (1841-1925), y Adam Smith (1723-1790), como precursores de una serie de modelos de gestión y planificación, que fueron la base de una revolución industrial que estaba por llegar, y que tantos beneficios ha dado.

Hoy, inmersos de lleno en el siglo XXI, podemos decir que, debido a la aparición de las nuevas tecnologías ya a finales del siglo XX, además de ayudar a mejorar, perfeccionar, y redefinir los modelos de gestión de proyectos que se conocen, ha surgido la necesidad de generar nuevos modelos adaptados a nuevas necesidades.

En este proyecto, se aborda la metodología ágil para la gestión de proyectos, aplicada al desarrollo de sistemas informáticos, porque el origen de dicha metodología nace de la necesidad de crear nuevos desarrollos de software en unas condiciones de adaptación, que no eran aceptables con las metodologías tradicionales.

Sin embargo, esta nueva metodología, promulgada desde el año 2001 a través del *Manifiesto Ágil*, se ha extendido, aplicándose a otros campos, como el desarrollo de proyectos convencionales, redefinición de procesos elementales, o la aplicación de dicha metodología a nivel personal.

Abstract

It is a fact that projects, understanding by project everything that is done in a planned way, according to criteria of order, planning, and execution, takes with us from the origin of our times (from the construction of the pyramids in Egypt, to the construction of the Great Wall of China, through Noah's Ark).

However, the methodology of projects, as we know it today, arose towards the middle of the twentieth century from the hands of Henry Gantt (1861-1919), Frederick Winslow Taylor (1856-1915), Henri Fayol (1841-1925), and Adam Smith (1723-1790), as forerunners of a series of management and planning models, which were the basis of an industrial revolution that was to come, and that has given so many benefits.

Today, fully immersed in the 21st century, we can say that, due to the emergence of new technologies already at the end of the 20th century, in addition to helping to improve, perfect, and redefine the project management models that are known, the need has arisen to generate new models adapted to new needs.

This project deals with the Agile methodology for project management, applied to the development of computer systems, because the origin of this methodology stems from the need to create new software developments in conditions of adaptation, which were not acceptable with traditional methodologies.

However, this new methodology, promulgated since 2001 through the Agile Manifesto, has been extended and applied to other fields, such as the development of conventional projects, redefinition of elementary processes, or the application of this methodology at a personal level.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Glosario

Agile Coach	Persona formada en metología Agile, cuyo objetivo es velar porque se desarrolle la metología.
Burndow Chart	Gráfico de quemado, en el sentido de cuánto se lleva gastado o realizado en el proyecto, comparado con el ritmo teórico.
Pain-Points	Puntos del proceso que identifican aspectos de mejora.
Planning Poker	Proceso de identificación de la duración de las tareas mediante cartas.
Product Owner	Representa la voz del cliente: conoce las necesidades como negocio, tiene la visión global del *product*, y es capaz de tomar decisiones.
Product Backlog	Lista de funcionalidades requerida por negocio. Se encuentra priorizada.
Retrospective Meeting	Reunión tras la realización de un sprint, para encontrar fortalezas y mejoras.
Scrum	*Marco de trabajo* para gestionar proyectos dentro de la metodología Agile.
Scrum Master	Rol con capacidad de gestión. En Scrum, hace referencia a la persona que se encarga de que apliquen las metodologías seleccionadas, y resuelve cualquier problema acerca de las mismas durante todas las etapas del sprint.
Sprint	Ciclo de trabajo. Proceso organizado en un período de tiempo (usualmente de una a cuatro semanas) que se repite. Es una forma de organización utilizada en Scrum.
Sprint Backlog	Conjunto listado de tareas que conforman un sprint.
Sprint Daily	Reunión para revisar la planificación del día anterior, y el día en curso.
Sprint Review	Reunión de revisión de entregables con el Product Owner.
Sprint Planning	Reunión de planificación de las tareas del Product Backlog.
Time-Box	Duración de un evento (reunión, sprint, etc.).

Prólogo

Durante estos 4 años en los que he realizado el Máster de Ingeniería Industrial, mi andadura profesional y académica han convivido juntas: antes de empezar el máster, trabajé en una de las cinco únicas fábricas del mundo donde se fabrican los dos principales tipos de transformadores (Columnas y Acorazados) en **A.B.B.**

Posteriormente a esta experiencia, mi vida siguió ligada a ese elemento común como es la electricidad, pero vista desde otra perspectiva: la oportunidad venía de la mano de **Endesa Generación** y la Unidad de Producción Hidráulica de la Zona Sur de España.

Por último, han sido estos últimos 4 años, de la mano de **Endesa Medios y Sistemas**, los que me han permitido, además de conocer en profundidad el sector eléctrico, colaborar y tratar de mejorar el proceso y los sistemas que soportan la Facturación y el Cobro de los servicios que presta **Endesa Distribución**.

Además de realizar proyectos regulados por la legislación española mediante B.O.E., órdenes ministeriales, o bien acordados en la sede de la Comisión Nacional de los Mercados y la Competencia, la empresa ha apostado por el desarrollo de proyectos, utilizando las metodogías ágiles para la redefinición de los procesos que afectan a todas las áreas de la empresa.

A través del presente documento pretendo ofrecer una visión del enfoque y del desarrollo del proyecto, mediante un caso de estudio concreto.

Por motivos de confidencialidad con la empresa, el caso de estudio no será un caso real, sino uno ficticio. Dicho caso recoge y abarca, no obstante, datos muy similares al proyecto real.

1 INTRODUCCIÓN

De un tiempo a esta parte, estamos viendo y viviendo, una serie de continuos cambios que afectan a todos lo sectores, tanto laborales, como no laborales. Todos estos cambios, comparten una misma palabra común: La Globalización.

Pero ¿Qué es la globalización? Para definir el término "globalización" pueden utilizarse varias acepciones(Hirsch 2004):

i. De forma **técnica**, globalización y desarrollo de nuevas tecnologías están vinculados.

ii. Desde un punto de vista **político,** la globalización se entiende como un marco de consenso donde la igualdad y la justicia social forman parte de la convivencia mundial.

iii. Desde un punto de vista **cultural**, la globalización se entiende como la aplicación de los derechos humanos de una forma generalizada en todas las comunidades del mundo.

iv. Por último, desde un punto de vista **capitalista**, la globalización representa el libre comercio: la internacionalización y el acceso completo a cualquier recurso en cualquier parte del mundo.

De este conjunto de significados, podemos decir que, para la industria (además de para cualquier otro sector), que participa en un mercado globalizado, cada vez es mayor la necesidad de adecuarse ágilmente y de forma estratégica, a las necesidades del mercado, para poder ser competitivo.

Estas necesidades, han llevado a que los valores sobre los que se mantienen las industrias, y el negocio que acude a las mismas, se transforme: valores como la sapiencia, la comunicación, y la resiliencia a la alta variabilidad del mercado, han dejado en un segundo plano, otros que eran determinantes hasta ahora, como la preparación de personal y/o la gestión de los recursos.

La entrada en el mercado de nuevos productos, con ciclos de vida extremadamente cortos en algunas circunstancias, obliga a incrementar la productividad para ofrecer una mejora contínua de los procesos, tratando de prever las demandas, y siendo capaces de readaptarse en un tiempo récord, para poder seguir compitiendo en un mercado global.

Este hecho, afecta de forma directa, al desarrollo de sistemas informáticos, que aporta valor, cada vez en mayor medida, al resto de procesos industriales en los que la informática forma parte como eje necesario sobre el que desarrollar los productos.

Ya en otros libros, como por ejemplo, en "Ingeniería de Software" (Addison-Wesley 2006) se defiende el desarrollo de la economía mundial como motivo principal para el desarrollo informático. También, entre otros factores, se defienden como fuentes de desarrollo, la descentralización empresarial, el reparto de tareas de forma automática, la volatibilidad de los requerimientos, la reducción del ***time-to-market***[1], las ***U.X.***[2], el avance tecnológico y la aparición de problemas de forma asíncrona como puntos clave en el desarrollo software en la actualidad.

Ante esta situación, cabe preguntarse sobre el nivel de efectividad de las metodologías utilizadas en el desarrollo de sistemas software hasta ahora, también llamadas *metodologías convencionales*, frente a nuevas formas de desarrollo. ¿Se adaptan los métodos actuales de forma rápida a los cambios, y a los cambios de necesidades? ¿Cabría la opción de desarrollar un nuevo método adaptado a la situación actual? Según Kent Beck (miembro de la composición del manifiesto ágil), *"cada cosa en software cambia: los requisitos cambian, el diseño cambia, el negocio cambia, la tecnología cambia, el equipo cambia y los miembros del equipo cambian. El problema no es el cambio, porque el cambio, inevitablemente, va a ocurrir. El problema, realmente, es nuestra incapacidad para hacer frente a los cambios"*.

[1] Tiempo desde que un producto es concebido, hasta que llega al mercado.
[2] Experiencias de usuario (User Experience)

Por todo ello, podemos concluir que una parte del mercado del software está sufriendo evoluciones contínuas, y que no existía una metodología que cubriera dicha casuística, ni en tiempo ni en forma.

Ante estas situaciones, las metodologías ágiles se presentan como una alternativa asumible, debido a que otros métodos de desarrollo, basados en aspectos tradicionales como el control sobre la estricta calidad de la documentación, el control total de jerarquías, o el cumplimiento de hitos y fases de una manera secuencial, no son los más recomendados para proyectos donde las condiciones de desarrollo son muy variables, y donde priman los tiempos, y la calidad del software, frente a aspectos más procedimentales.

Además, una gran parte del desarrollo de sistemas que se realiza hoy día, está más centrado en el desarrollo de productos ya creados, añadiendo nuevas funcionalidades, o modificando otras existentes, con el objetivo de rentabilizar la inversión inicial (un caso que da fe de esta situación, podría plasmarse en el actual sistema informático comercial de la compañía sobre la que se desarrolla este trabajo, cuya versión inicial de lanzamiento, data del año 2000, y sobre el que se han ido añadiendo nuevas características, manteniendo el mismo ***core***).

De hecho, algunos estudios (Addison-Wesley 2006) ponen de manifiesto que el presupuesto dedicado al mantenimiento del software y pequeñas mejoras pueden llegar a suponer un 80% del presupuesto, dedicando el 20% restante a nuevas funcionalidades.

1.1 Objetivos

El objetivo de este proyecto es presentar el uso de las metodologías de desarrollo ágil en proyectos de software, para posteriormente, dar fe de su aplicación en un desarrollo realizado con la metodología SCRUM en la creación de un proyecto concreto: SMART, un sistema software capaz de monitorizar de facturación y cobros de una empresa, y notificando al área correspondiente los errores detectados, incluyendo el flujo de resolución de estos.

Además, se explicarán de forma detallada todas las etapas, herramientas y terminologías que se usarán para el desarrollo de proyectos, mediante esta metodología, encontrándose al principio del documento, un glosario de toda la terminología a emplear, para que resulte más cómoda su lectura, debido a la cantidad de términos que se emplean para describirlas.

2 METODOLOGÍAS

En esta sección haremos una introducción sobre las metodologías de proyectos, enfocadas al desarrollo del software: trataremos desde una visión histórica cómo surgen, en qué consisten, y cuáles son son los principios en los que se basan, así como las ventajas y desventajas respecto a otras metodologías de desarrollo de proyectos. Posteriormente, trataremos dentro del marco de las metodologías ágiles, la metodología **SCRUM** que será la que se usará en el modelo con el que se desarrollará este trabajo.

2.1 Qué se entiende por Metodología

Según la *R.A.E.*[3], el término "*Metodología*" proviene del griego "*μέθοδος métodos"* y significa "*Ciencia del método"*, o en su segunda acepción, "*Conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal"*.

Pero quizá, nos sea de mayor ayuda, la definición realizada por Avison y Fitzgerald en el libro "*Desarrollo de Sistemas Informacionales. Metodologías, Técnicas y herramientas*":

"Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología esta formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas mas apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo." (Avison y Fitzgerald 1998)

Esta acepción de los autores nos da una definición de metodología mucho más clara, que hace referencia a varios componentes de esta: las fases que componen las metodologías, las herramientas que se utilizan en el uso de estas, y las técnicas de las que se disponen para aplicar las metodologías.

Sin embargo, nos podemos seguir planteando que una metodología, debe ir más allá de ser una simple recolección de una serie de pasos a realizar o seguir, para conseguir algo.

Mediante esta definición de los autores, podemos ver que las diferencias presentes en las distintas metodologías están basadas en las distintas técnicas, y por tanto herramientas, aplicadas en las distintas fases, sobre el mismo contenido de estas.

2.1.1 Términos relacionados con las metodologías

Usando el mismo procedimiento de análisis del punto anterior, una **técnica,** para la R.A.E., se entiende como *"un conjunto de procedimientos y recursos de los cuales se sirve una ciencia o un arte"*. Para el caso en estudio, aplicado a la metodología, los procedimientos consistirán en unas normas específicas con las que realizar una serie acciones que compondrán el proceso de desarrollo (generalmente en más de una fase), y los recursos serán habitualmente las herramientas para llevar a cabo dichos procedimientos. Podemos encontrarnos, que el término ***técnica***, se utiliza para hacer referencia a una metodología, pero esto sería un error, puesto que una metodología puede utilizar diferentes técnicas y no es en sí misma una sola técnica.

Una **herramienta**, siguiendo con la ayuda de la *R.A.E.,* se define como *"un conjunto de instrumentos que se utilizan para desempeñar un oficio o un trabajo determinado*". En nuestro caso, las herramientas constituirán los medios a través de los cuales podremos llevar a cabo las técnicas descritas por las distintas metodologías.

[3] R.A.E.: Real Academia Española

Este instrumento, aplicado a las condiciones de contorno de este proyecto, consistiría en un producto informático. Un ejemplo de herramienta para desarrollar aplicaciones informáticas podría ser una herramienta de modelado, repositorios de software, compiladores, depuradores, conversores de códigos, etc.

Un **método**, según la *R.A.E.*, es *"el modo de obrar o proceder, hábito o costumbre que cada uno tiene y observa"*. Este método, nos dirá dónde están los límites: nos guiará a lo largo de una tarea. A cada técnica, comentada anteriormente, se le aplica un método, que nos ayuda a usar las técnicas mediante las herramientas concretas.

Un **proceso de desarrollo**, en este caso, **de software**, se podría definir como el grupo de tareas que, son necesarias realizar, para crear un producto, en este caso informático. En este proceso, que se puede hacer de muchas maneras y con resultados, totalmente diferentes, es donde las metologías tratan de dar unas pautas comunes, para obtener un buen software. Dicho proceso, está compuesto en sí mismo por más procesos, pero a su vez, también está conformado por subprocesos, y es en el control de dichos procesos, donde aparece de forma implícita la calidad.
Gran parte de las metodologías, introducen como una técnica más dentro del proceso, el control de los procesos con el fin de garantizar una solución software de calidad. Habitualmente, este control es llevado a cabo como si se tratate de un producto más, y se certifican para dar fe de que se está siguiendo el proceso conforme a la norma.

Un modelo de procesos es una representación esquemática de las distintas etapas y actividades, que se deben realizar para obtener un proceso. Los modelos de procesos que hay a día de hoy son (Derniame, Kaba y Wastell 1999):

- **Modelo secuencial:** Cuyo máximo exponente conocido es la metodología *Waterfall*.
  Se parte de una recepción y análisis, de los requerimientos de los clientes. Tras dicho análisis, se establece un conjunto de requisitos funcionales y no funcionales necesarios, que serán documentados para que, en la fase de diseño, los ingenieros trabajen entre sí de cara a establecer una arquitectura sobre la que se desarrollará el software. En la siguiente etapa, los desarrolladores generan dichas funcionalidades, y por último, el sistema es probado y validado por los usuarios y/o clientes, y entregado, o puesto en producción.
  El modelo *waterfall* no resuelve las modificaciones de los requisitos, ya que no permite cambios una vez avanzado en las fases del proyecto, porque supondría empezar de nuevo.

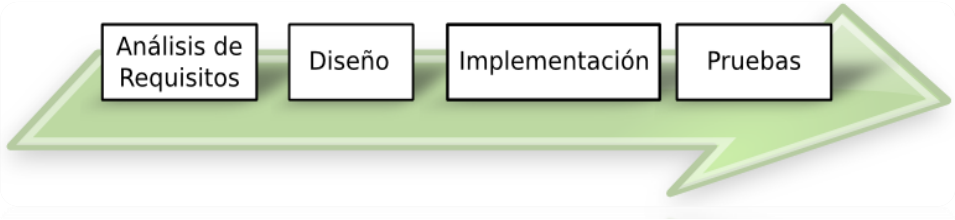


Figura 1. Fases del modelo de desarrollo con la metodología *Waterfall*. Fuente Propia

- **Modelos iterativos e incrementales:** rompen el modelo anterior, y repiten waterfall en cada fase.
  - **Desarrollo incremental:** disminuyen el tiempo total, partiendo el proyecto en tramos concatenados. De la misma manera que en *waterfall*, los requerimientos se analizan antes de forma previa. La diferencia está, en que cada requerimiento se desarrolla de forma independiente. De esta forma, el desarrollo de cada parte se puede solapar, ahorrando tiempo gracias al reparto de las tareas a distintos equipos.
  - **Desarrollo iterativo**: se centra en entender bien los requerimientos que puedan variar y la gestión de riesgos. En este modelo, el proyecto se divide en varias iteraciones, de forma que en la última se obtenga un producto final. La primera iteración viene a ser como un boceto, al que se le van añadiendo funcionalidades en cada repetición de ciclo. En cada una de ellas, se utiliza el modelo secuencial. Funciona en entornos variables, debido a que los requerimientos no cambian en cada ciclo

- **Modelo en Espiral**: mezcla lo mejor del modelo tradicional, introduciendo el prototipado (mediante repeticiones). Además, en él se analizan otras opciones, y se realizan tareas para identificar y tratar de minimizar los riesgos.

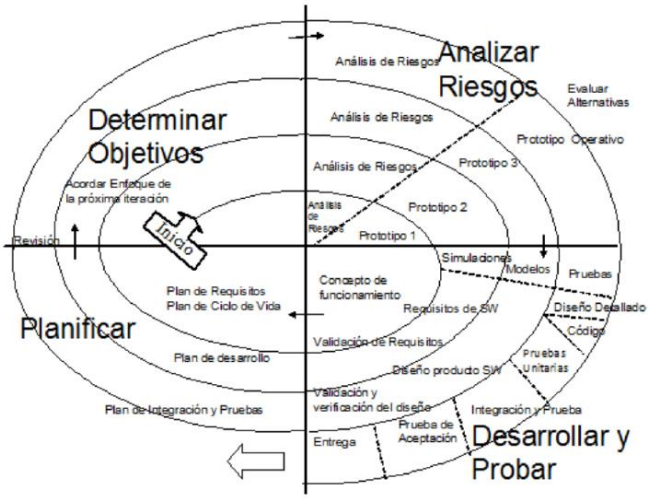


Figura 2. Modelo en Espiral (Rojas Contreras et al. 2011)

Tando este modelo, como el iterativo, permiten una adaptabilidad mucho mayor que el modelo secuencial, pero no son pocas las personas, que creen que no son suficientes para la variabilidad del mercado: extensas planificaciones y periodos de análisis, juntamente con la excesiva documentación, llevan a que este tipo de modelos, se parezcan más a las metodologías tradicionales.
Otros modelos, que no se tratarán aquí, porque no tienen tanta relevancia para el desarrollo de las metodologías ágiles, pero se mencionan aquí por si el lector quiere profundizar sobre más modelos, son: el V-Model, W-Model, X-Model, RAD, o el Orientado a Objetos.

2.2 Contexto histórico.

Para explicar la aparición y evolución de las metodologías, - en este caso que nos ocupa, sobre el desarrollo de software- debemos hacer referencia al contexto histórico relacionado con el desarrollo de la tecnología informática en últimos 60 años.
A finales de los años 60, los programadores encargados de codificar los ordenadores, y/o computadoras de la época, no prestaban gran atención al estilo de programación que utilizaban, debido a los limitados recursos de espacio y velocidad que prestaban dichas máquinas: su principal preocupación consistia en escribir un código de programación que ocupase el menor espacio posible y fuese eficiente. A todo esto, hay que sumar que los compiladores de la época eran bastante limitados en perjuicio de la calidad en la decodificación.
Esta situación, dio un vuelco cuando empezaron a aparecer los primeros equipos "pequeños" y "baratos", que venían como resultado de la *Ley de Moore*[4]. Este hecho, cambió los criterios para medir la eficiencia de cualquier programa, ya que los equipos al ser más rápidos, y sin limitaciones de espacio, no requerían un código fuente tan eficiente y liviano como antes.
Con la aparación del *System/360* de IBM en 1964, -primeros pc´s para ejecutar aplicaciones, indistintamente de su tamaño o ambiente (científico o comercial)- y la estandarización de los lenguajes de alto nivel, se aseguró portabilidad y perdurabilidad sobre los programas, consiguiendo que los costes del software superasen a los costes del hardware, siendo el arranque de la ***crisis del software*** que hoy día perdura.

[4] La ley de Moore, (de Gordon. E. Moore), predice que el número de transistores de un microprocesador se duplicará cada 2 años aproximadamente.

Figura 3. Sistema 360 de IBM. Fuente: ibm.com

En base a estos hechos, se establecieron los siguientes criterios para marcar el éxito del desarrollo software:

1. El coste inicial debe ser relativamente bajo.
2. Debe ser fácil de mantener.
3. Debe de ser llevable a nuevo hardware.
4. Debe hacer lo que el cliente quiere.

A pesar de estas pautas, los programadores siguieron creando software sin hacer mucho caso. *Edsger Dijkstra* (1930-2002), - padre de la programación distribuida- propuso la computación como una parte de las matemáticas aplicadas y sentó las bases de la **programación estructurada**, origen de las metodologías para el desarrollo informático.

Dicha programación se basa en:

- Desarrollar utilizando la filosofía **top-down** (del proceso global, a los procesos concretos), en contraposición a **bottom-up** (funcionalidades detalladas se van componiendo y entrelazando).
- Uso de un conjunto de premisas para programar (utilizando lógicas iterativas tipo *Do* o *While* en lugar del *"Go to"*).
- Seguimiento de una serie de principios, o criterios, para descomponer problemas de mayor tamaño.

A raíz de este planteamiento de Dijkstra (a través de su libro en 1976, *A Discipline of Programming*), emergieron lenguajes de programación con claras referencias a sus bases, como el lenguaje ***Pascal,*** que trata de crear un código con una buena estructura y entendible. Pero Dijkstra no solo planteó la cuestión sobre la lógica basada en el comando "*Go to*", sino que, mediante sus principios, implantó la idea de diseñar de una forma estructurada, también llamada, Ingeniería del Software.

Durante la conferencia de la **OTAN**[5] de 1968, los países que se reunieron para discutir los problemas de fracasos derivados del desarrollo informático usaron denominaciones como, "*ingeniería del software*", planteando soluciones, como la de formular unas pautas sustentadas en dos principios básicos en toda ingeniería:

[5] OTAN: Organización del Tratado del Atlántico Norte.

1. **Gestión predictiva:** antes de comenzar un proyecto hay que clarificar qué se va a realizar: toma de requirimientos, planificación, y cálculo de costes y riesgos. Tras ello, realizar un control de riesgos para que estos se minimicen y el plan se lleve a cabo.

2. **Principio de calidad:** La calidad del producto, es el resultado de la suma de las calidades parciales de los pasos que llevan a la generación de dicho producto.

Todo esto, juntamente con una bajada del coste del hardware, hizo que se pusisera el foco en el coste del software: Los clientes comenzaron a molestarse por la cantidad de dinero invertido y los bajos resultados, que pedían inyectar más y más dinero. Fue el origen, de las metodologías de desarrollo del software.

2.3 Cronología de las metodologías.

El desarrollo ágil, y el conjunto de metodologías que lo componen, surge como un método necesario para afrontar la gestión de los proyectos, frente a los modelos clásicos de desarrollo en cascada, o *waterfall.*

La historia del nacimiento de las distintas metodologías, se puede dividir según Avison y Fitzgerald (Avison y Fitzgerald 1998) en:

1. Pre-metodologías: entre los años 1950-1960.
2. Primeras metodologías: entre los años 1970-1980.
3. La era de las metodologías: entre los años 80 y principios de los 90.
4. Era post-metodológica: a partir de los 90.

La era ***pre-metodológica*** está marcada por ser anterior a la aparición de las primeras metodologías, y en la que destaca la existencia de grandes computadoras, que se encargaban de ejecutar aplicaciones científicas y militares. De forma gradual, se fueron empleando en empresas, realizando pequeños informes económicos como resultado de una serie de cálculos, o para el almacenamiento de información de usuarios y/o documentos en ficheros.

Los sistemas que se empleaban fueron desarrollados por programadores, como resultado de ejercicios, para su ejecución en plazos cortos de tiempo. Los usuarios normalmente quedaban insatisfechos con las soluciones planteadas, al ser la documentación escasa o en ocasiones no existir. Además, cualquier modificación necesitaba tiempo haciendo más compleja la aplicación, transformando a los programadores en un bien cotizado, al ser los únicos que sabían el código.

Poco a poco, fue siendo necesario más tiempo para analizar y diseñar las aplicaciones, apareciendo las figuras de los analistas de sistemas, que hacían el papel de intermediario entre el programador y el sistema; y, los operadores de sistemas, dedicados a controlar su funcionamiento.

Las ***primeras metodologías*** comienzan en 1971 con la aparición del modelo en cascada (Royce 1970).Esta metodología, se basa en la realización de una serie de etapas, de forma que para considerar un producto como entregable, todas las etapas anteriores asociadas debían estar finalizadas. Royce presentó un modelo de siete etapas:

1. Requerimientos del sistema.
2. Requerimientos de software.
3. Análisis.
4. Diseño del programa.
5. Codificación.
6. Pruebas.
7. Operaciones (Implantación y mantenimiento).

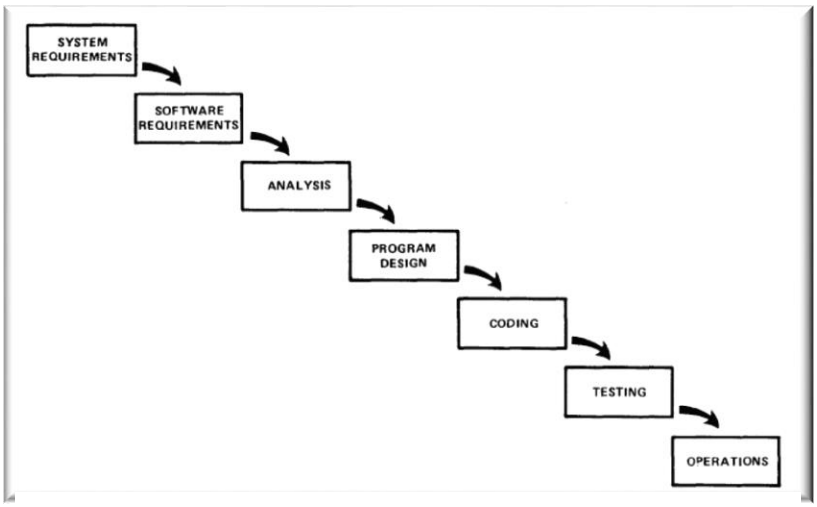


Figura 4. Esquema de etapas presentado por *W. Royce* en *Managing the development of large software systems (1970)*

Las siete etapas pueden resumirse en las siguientes cinco:

- **Análisis y especificación de los requirimientos.**

Estas tres primeras, se centran en analizar el problema, estableciendo las necesidades de la aplicación, así como los requerimientos. Requerimiento que son inamovibles durante el ciclo de vida del desarrollo.

- **Diseño.**

Los requerimientos de las anteriores etapas se mapean con un esquema del software que será necesario. En esa etapa, el ingeniero debe centrar su atención en:

  - Estructura de los datos.
  - Arquitectura del software.
  - Detalles algorítmicos.
  - Representación de la interfaz.

- **Codificación.**

El diseño de la anterior etapa debe traducirse en una aplicación desarrollada en algún lenguaje informático, en algún entorno específico, utilizando una tecnología existente.

- **Pruebas.**

En esta etapa se trata de validar que el software cumple con los requisitos y se detectan posibles errores y se tratan de corregir, antes de implantarse.

- **Implantación y mantenimiento.**

En esta última fase, el software se prepara para:

  - Adaptarse a los nuevos requisitos de los clientes.
  - Adaptarse a los diferentes entornos de trabajo.
  - Corregir errores no detectados en la fase de pruebas.
  - Mejorar la eficiencia de la aplicación.

Las principales desventajas que plantean este modelo son el grado de satisfacción de los clientes con la entrega final (no suele ser acorde), la falta de flexibilidad a los cambios, el exceso de documentos, y la facilidad para no cumplir los tiempos estipulados.

En la **era de las metodologías** nacieron nuevos enfoques en respuesta a las limitaciones enumeradas anteriormente, que se pueden agrupar en dos vertientes: las que mejoraron el modelo *waterfall* y las que buscaron una nueva forma de gestionar proyectos. Además, a finales de los años 70 surgieron nuevas herramientas y técnicas sobre los que se basaron estas nuevas metodologías, como pudieron ser los diagramas de flujos, el modelo entidad-relación, la normalización, los diagramas de actividad, etc., y herramientas, como las de gestión de proyectos, bases de datos de código, repositorios de modelado, etc.

Entre las metodologías que mejoraron el modelo waterfall, surgieron MERISE, Yourdon System Method, o Structured Systems Analysis and Design Method (SSADM), que integraban las novedades enumeradas en el párrafo anterior.

Entre las metodologías alternativas de los años 80, Avison realizó una clasifición de las mismas en varios grupos de una forma sencilla de entender (Avison y Fitzgerald 1998):

- **Sistemáticas:** emplean técnicas cualitativas a situaciones no sistemáticas. Un modelo que aplica esta medología podría ser Soft Systems Methodologies (SSM[6]).
- **Estratégicas:** enfocadas en la planificación previa al desarrollo de software, para cumplir los requisitos de negocio. Un modelo que aplica esta medología podría ser Business Systems Planning.
- **Participativas:** basadas en la involucración de usuarios en el análisis, diseño e implementación. Como podría ser, ETHICS.
- **Prototipadas:** utilizan prototipos para visualizar y validar el producto. Un modelo que aplica esta medología podría ser, Rapid Application Development (RAD[7]).
- **Estructuradas:** extienden la programación estructurada al análisis y diseño, con técnicas que permiten un análisis descendente y la representación compleja de procesos.
- **Análisis de datos:** tratan de documentar y comprender los datos. Un modelo que aplica esta medología podría ser, Information Engineering.

Posteriormente, en los primeros años de los 90 nacieron nuevas metodologías (Carvajal Riola 2008) como:

- **Orientadas a objetos**. Enfocadas en objetos que representan entidades del mundo real.
- **Desarrollo incremental o evolutivo**. Basadas en prototipos primitivos sobre los que desarrollar por etapas. Un modelo que aplica esta medología podría ser, Dynamic System Development Methods (DSDM).
- **Específicas**. Llevadas a cabo para cierto tipo de proyectos, buscan un objetivo en concreto:
    - *Welti*, para desarrollar aplicaciones tipo ERP[8].
    - *CommonKADS*, para aplicaciones de gestión de conocimientos.
    - *Process Innovation*, para aplicaciones de reingeniería de procesos.
    - *Renaissance*, para aplicar ingeniería inversa en sistemas heredados.
    - *Web IS Development Methodology (WISDM)*, para aplicaciones Web.

Sin embargo, gran parte de los usuarios de estas metodologías, no acabaron de estar satisfechos con las metodologías que emplearon, bien fuera *waterfall* o alguna de sus variantes.
La mayoría, estaban destinadas para aplicarse en situaciones ideales, y en las situaciones reales, aparecían problemas, debido también a la singularidad de cada proyecto.

Uno de los motivos principales de este descontento, fue que los usuarios de dichas metodologías esperaban encontrarse un guión paso a paso, y con un enfoque descendente (desde arriba hacia abajo) a lo largo del desarrollo (rara vez se da), pues lo normal, suele consistir en adaptar el guión, omitiendo fases y modificando el órden de ejecución de los pasos de éste.

[6] Metodología Suave de sistemas.
[7] Desarrollo rápido de aplicaciones.
[8] Planificación de Recursos.

El resultado fue la aplicación de metodologías más permisivas en las que el desarrollador elige la estructura, las herramientas, las técnicas, y las fases, en función de la necesidad del momento. (Un modelo que aplica esta medología podría ser Multiview).

El éxito o fracaso en el desarrollo no dependió únicamente a la utilización de una metodología en concreto, pero a pesar de ello, nació una corriente, que replanteó los beneficios de las metodogías, que, juntamente con otros conceptos no metodológicos, darían paso a la era **post-metodológica**.

Existen diversos motivos que explicarían este rechazo al uso de las metodologías, pero una de las principales fue la decepción por el no cumplir las expectativas iniciales: las metodologías no pudieron solucionar, todos aquellos problemas para los que supuestamente la metodología pondría fin.

Algunas de las razones que Fitzgerald y Avison (Avison y Fitzgerald 1998) presentan como las causantes del debate metodológico fueron:

- **Productividad:** no aumentaron la productividad prometida.
- **Complejidad:** en ocasiones eran demasiado complejas.
- **Habilidades:** eran necesarias habilidades específicas.
- **Herramientas:** era difícil utilizarlas, caras y no eran rentables.
- **No contingente:** no se adaptaban a las necesidades.
- **Enfoque unidimensional:** basadas en una sola idea sobre cómo desarrollar.
- **Inflexible:** imposibilidad de adaptarse a los cambios.
- **Supuestos no válidos:** no se permite la modificación de requerimientos en ningún momento.
- **Desplazamiento del objetivo:** el uso literal de las herramientas, dejando de lado la creatividad.
- **Poco orientadas:** a personas y al entorno.
- **Dificultades:** para ponerlas en práctica.

Como hemos visto hasta ahora, cuando una metodología, o varias, no cubrían las necesidades, surgían nuevas variantes que trataban de dar solución.

Mostramos ahora, los planteamientos que se llevaron a cabo en las empresas, y en los usuarios respecto a esta situación:

- ***Desarrollo externo:*** las empresas optaron por no realizar de forma interna proyectos informáticos y externalizarlos.
- ***Mejora continua*****:** se apuesta por la mejora continua.
- ***Desarrollo ad hoc[9] y contingencia***: el desarrollo *ad hoc* supone la vuelta a la era pre-metodológica.
- ***Desarrollo contingente o flexible*****:** permitiendo, manteniendo una cierta estructura, posibilidades de elección de componentes a los desarrolladores.
- ***Desarrollo ágil*****.** Centrada en la participación de los usuarios y clientes más que en procesos y herramientas, trabajando más en el software y menos en la documentación, colaborando más con clientes en vez de negociar, y responder a los cambios por encima de la planificación de los tiempos del trabajo.

[9] Ad hoc: Específico

2.4 Requesitos para el Nacimiento de una Metodología.

La enciclopedia de la ingeniería de software (Scacchi 2002) propone los siguientes aspectos a cumplir por toda metodología:

1. **Visión del producto:** todas las partes deben de conocer todos los detalles sobre el producto: plazos, tareas, cambios, etc.
2. **Vinculación con el cliente:** la metodología trata de ser la base para indicar las relaciones entre los distintos actores dentro de un proyecto: usuarios, programadores, soporte, etc.
3. **Establecer un modelo de ciclo de vida:** elegir un modelo: iterativo, waterfall, etc. Así se podrán conocer las fases, y gestionar los recursos.
4. **Gestión de los requisitos:** mínima calidad exigible a los requerimientos del producto.
5. **Plan de desarrollo:** plan lo suficientemente claro, y conciso, para que los programadores puedan llevar a cabo los desarrollos de una forma clara y precisa, sin equivocaciones, y sin dudas. También debe ser posible añadir modificaciones con posibilidad de recuperar el plan original.
6. **Integración del proyecto:** La metodología debe clarificar como el producto se relacionará con el resto de los productos actuales, en el presente, y a futuro.
7. **Medidas de progreso del proyecto:** deben ser accesibles, y lo suficientemente claras, para que todas las partes del proyecto sean capaces de ver, de una forma fácil, y rápido el estado actual de un proyecto, y del tiempo que resta para su terminación.
8. **Métricas para evaluar la calidad:** el equipo encargado de gestionar los despliegues del producto debe controlar y preservar la calidad del software que se está desarrollando, como un símbolo de estabilidad del producto.
9. **Maneras de medir el riesgo**: se deben tener controlados los riesgos: no solo en la gestión de estos, en las fases donde aparezcan, sino también en la medida en que puedan ser cuantificados y cualificados para dar una mejor respuesta a los pasos necesarios a dar, para afrontarlos.
10. **Cómo gestionar los cambios:** debe ser posible incluir la posibilidad de que aparezcan cambios, o bien en las necesidades del negocio, o bien en el propio desarrollo del software.
11. **Establecer una línea de meta:** se debe tener presente con qué funcionalidades probadas, se dará por bueno un producto. Es necesario que todas las partes que conforman el proyecto sepan de forma clara, cuando algo está “finalizado”.

Viendo todos estos aspectos enumerados por la enciclopedia del software, parece que no hay ninguno de ellos que se aleje del uso del sentido común, y con los que todos podríamos estar de acuerdo en que son necesarios cumplir en cualquier metodología.

2.5 Metodologías Ágiles.

2.5.1 Necesidad de las metodologías Ágiles.

En la industria del software, el desarrollo de procesos, y por tanto de productos, ha venido utilizando metodologías que lo encasillaban como algo rígido e inflexible, aspecto que se desmiente al analizar la variedad y la dinámica del mercado de software. Como indica Boehm (Boehm 1979) en su libro *"Software Engineering as It Is"* la tendencia del sector se dirige hacia desarrollo en menos tiempo, y a una vida más corta de los productos finales. Es en estas condiciones, donde la entrega de valor de producto en plazos de tiempos muy cortos para llegar al mercado, marcan la diferencia.

Sin embargo, las metodologías tradicionales, presentan ciertos problemas a la hora de gestionar proyectos cambiantes, o con cierto dinamismo. Algunos de estos problemas son:

- La toma de requerimientos como primera etapa ante de comenzar el desarrollo, debe cubrir todas las funcionalides y todos los aspectos de aquello que se desea implantar. Con esto, se busca evitar cambios a mitad de proyectos, que resulten inviables por costes y tiempo (también se conocen como metodologías predictivas). La realidad de este hecho es que difícilmente una toma de requerimientos completa justifica el tiempo y coste invertido en realizarla.
- El cliente no tiene los conocimientos suficientes para definir con el nivel de detalle necesario, lo que necesita, cambiando sus necesidades a lo largo del desarrollo. Llevar un control de cambios, y distintas herramientas para controlar este tipo de situaciones, permiten proteger al proyecto en su desarrollo, pero desde el punto de vista del cliente, se entiende la gestión como algo que no se adapta a sus necesidades cambiantes, y que, en caso de necesidad de cambios, tengan costes adicionales.
- El desarrollo del proyecto contiene documentación que no siempre se utiliza, ni tampoco se revisa. La entrega por hitos suele implicar la dedicación de recursos a elaborar mucha documentación con gran nivel de detalle que, en ocasiones, ni se utiliza, no la revisa el usuario, y por tanto no aporta valor sobre el producto final, teniendo que hacerse un esfuerzo exponencial en caso de que surjan cambios sobre el proyecto.
- El ritmo para desarrollar un software es lento. A los programadores, les resulta difícil entender sistemas tan complejos e interconectados, lo que desencadena en ritmos más lentos. Por ende, los propios desarrollos se abordan por etapas en pequeños trozos para reducir la posibilidad de errores y de costes a la hora de desarrollar. Además, las funcionalidades se van agregando, de forma que se puedan ir liberando parcialmente en los distintos entornos para que puedan ser probadas. Sin embargo, el ritmo que marca competencia exige velocidad, calidad y ahorro de costes, desplegando de manera contínua funcionalidades, debido a que, si no, los productos en su totalidad puedan quedar obsoletos.

Todos estos factores, hacen que sea difícil cumplir el ***Time to Market*** consiguiendo un malestar por parte de los usuarios:

- Al no haber sabido responder a tiempo, a lo que demandaba el mercado.
- A enfrentarse con los gestores de proyecto por no haberse involucrado lo suficiente para entregar el software a tiempo.
- A tener una sensación de que los desarrollos siempre son muy lentos, y muy costosos, pensando que faltan más programadores para adelantar dichos procesos de desarrollo.

A todo esto, se suma que en las metodologías tradicionales los riesgos se acumulan, siendo críticos si surgen en la etapa final, repercutiendo en retrasos, o incumpliendo la ejecución de las últimas fases del ciclo del proyecto.

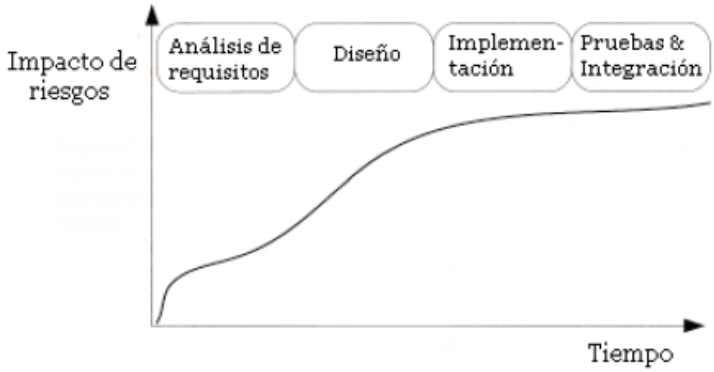


Figura 5. Impacto de riesgos vs Evolución de un proyecto Waterfall

A diferencia de las metodologías clásicas, las metodologías ágiles surgen como una variante para dar respuesta a estos problemas.Como bien indica el artículo "*Hallazgos empíricos en Métodos Ágiles*" (Lindvall et al. 2002) las metodologías ágiles son adecuadas cuando los requisitos son emergentes, y cambian de forma rápida.

De este modo, presentan varias ventajas:

1. **Respuesta a los cambios durante el desarrollo** puesto que durante toda la vida del desarrollo existe la posibilidad de mejorar el producto, y por tanto la percepción del cliente hacia el mismo, ya que es partícipe, y además, se integra la gestión de cambios como una característica del proceso.
2. **Entrega continúa en plazos breves** permitiendo al cliente saber en todo momento el estado de avance del proyecto, pudiendo probar las funcionalidades del producto incrementalmente y comprobar si cumplen sus necesidades, mejorando su satisfacción. Además, el trabajo y evaluación en periodos cortos de tiempo, disminuye el tamaño de los riesgos, las dificultades a dichos ciclos, y permite ir aprendiendo de los mismos.

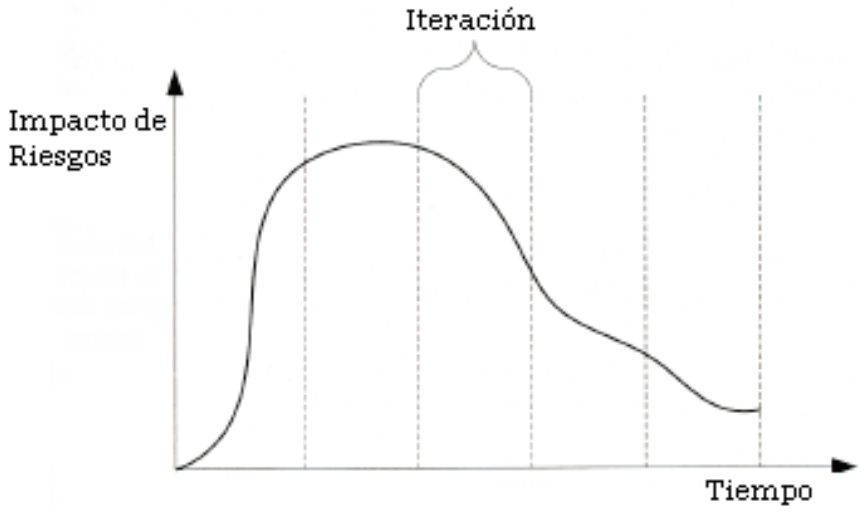


Figura 6. Impacto de riesgos vs Evolución de un proyecto Ágil

3. **Trabajo codo con codo entre cliente y equipo de desarrollo** con una comunicación directa y fluida eliminando malentendidos (principal origen de errores), y documentación que no aporta valor.
4. **Importancia de la simplicidad** eliminando trabajos innecesarios.
5. **Gestión de cambios integrada** en el propio proceso.
6. **Mejora continua de los procesos** y del equipo de desarrollo, dependientes del éxito técnico, del éxito personal y del éxito organizacional.

En la siguiente tabla, podemos hacer una comparación entre las metodologías tradicionales y las metodologías ágiles:

Tabla 1: Comparación entre Metodologías Tradicionales y Ágiles

Metodologías Tradicionales	Metodologías Ágiles
Basadas en estándares.	Basadas en buenas prácticas aprendidas y las experiencias.
Inflexibles.	Preparadas para cambios en el proyecto.
Normas impuestas externamente.	Normas impuestas internamente.
Jerárquicas, donde se aplican políticas/normas.	Proceso con menos principios.
Hay un contrato previo.	No hay contrato, o si hay, es flexible
Diálogo cliente-desarrollador inexistente.	Cliente integrado en el equipo.
Grupos grandes y en distintas localizaciones.	Pequeños grupos trabajando en el mismo sitio.
Muchos roles en la organización	Pocos roles.
La arquitectura es esencial.	Menos énfasis en la arquitectura.

2.5.2 Manifiesto Ágil: Origen, y principios.

El origen de la metodología ágil se remonta a febrero de 2001, donde durante una reunión celebrada en Utah (Estados Unidos), nació el uso de la palabra "ágil". En dicho evento, participaron 17 expertos de la industria del software, incluyendo a fundadores de algunas metodologías. El propósito de la reunión fue determinar los pasos a seguir por los grupos de desarrolladores para dar respuesta a la necesidad de crear un producto en el menor tiempo posible, admitiendo modificaciones durante su desarrollo. De esta forma, se buscaba una nueva forma de desarrollar software frente a los modelos tradicionales, más rígidos e inflexibles.

Como resultado, se fundó *"**La Alianza Ágil**"*, una organización dedicada a promulgar estos principios, ayudando a las empresas a aplicar estos conceptos. La base sobre la que se construyó dicha alianza fue el Manifiesto Ágil, un documento que resume la filosofía "ágil" en 12 principios: los dos primeros sintetizan la idea de agilidad, y el resto, desarrollan las pautas de desarrollo para el equipo de programadores, su organización y cómo conseguir los objetivos (José, H., Canós, P. L., & Carmen 2003):

De forma análoga, a lo ya comentado anteriormente en los requisitos para el nacimiento de una nueva metodología, en estos principios lo que se pretende es aplicar de nuevo un poco de sentido común, planteando cuestiones elementales, basadas en experiencias previas de fracasos que generalmente coinciden siempre en los mismos aspectos.

1. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporten valor.
2. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
3. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
4. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
5. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan, y confiar en ellos para conseguir finalizar el trabajo.
6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
7. El software que funciona es la medida principal de progreso.
8. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
9. La atención continua a la calidad técnica y al buen diseño, mejora la agilidad.
10. La simplicidad es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
12. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

Estos 12 principios, se resumen en 4 aspectos que valoran las metodologías ágiles:

I. Se valora al **individuo y las interacciones del equipo** de desarrollo **sobre el proceso y las herramientas**. El éxito depende de las personas: es mejor tener un gran equipo, que un buen entorno de trabajo. En ocasiones se comienza preparando un entorno de desarrollo sobre el que se selecciona al equipo, esperando que éstos se adecúen a él, en lugar de dejar que sea el propio equipo el que defina sus condiciones de desarrollo.

II. **Desarrollar software** en lugar de tener una detallada documentación. Se aboga por "no producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante". Los documentos han de ser de poca extensión y muy concretos.

III. **La colaboración con el cliente más que la negociación de un contrato**. Se plantea la relación continua entre usuario/negocio y equipo de programación. Esto hará que el proyecto fluya hacia un entendimiento total, y por tanto al éxito.

IV. **Responder a los cambios más que seguir estrictamente un plan**. Es necesario poder integrar las modificaciones que surjan a lo largo del desarrollo (ya sea al principio, o durante el proyecto) para poder llevar a cabo con éxito el proyecto, siendo necesaria una gran flexibilidad en la planificación.

2.5.3 Tipos de Metodologías Ágiles.

En este apartado, haremos un recorrido introductorio por las distintas metodologías Ágiles, para posteriormente, analizando los marcos de trabajo donde se mueven estas metodologías, realizar un análisis profundo de **Scrum**, que será la que se desarrolle con este proyecto, pero siendo necesarias citar al resto, para poder crear un contexto.

2.5.3.1 Agile Project Management (APM)

Fue desarrollada por Jim Highsmith en 1999 a través de su libro *"Adaptive Software Development: A Collaborative Approach to Managing Complex System"*. En el libro, Jim defiende que el proceso de desarrollo debe estar en consonancia con los objetivos, de forma que, si los objetivos se repiten y se pueden definir de forma previa en su totalidad, un proceso prescriptivo sería idóneo, pero si los objetivos de negocio son cambiantes, y con cierto carácter innovador, entonces el marco de trabajo debiera ser ágil, con una gran flexibilidad y adaptabilidad.

El modelo APM se basa en 5 fases: la previsión, la especulación, la exploración, la adaptación y el cierre.

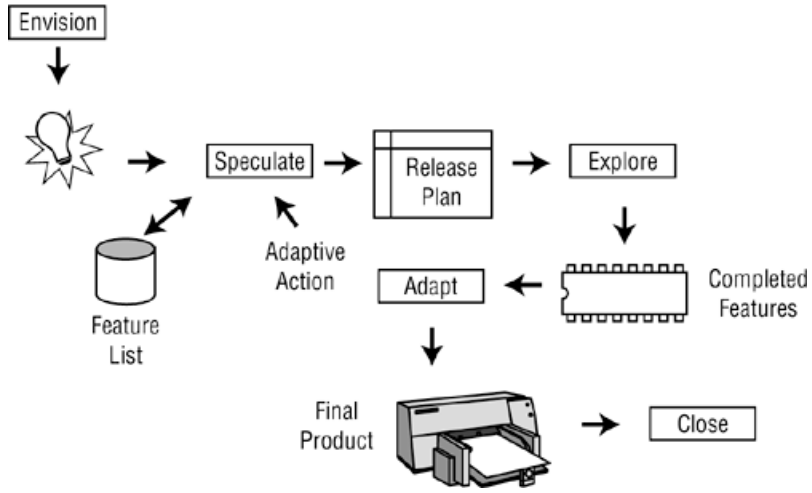


Figura 7. Fases del modelo APM. (Carvajal Riola 2008)

Previsión.

En esta fase, se definen los objetivos del proyecto, componentes del equipo y las formas de trabajar de forma conjunta.

Especulación.

En esta fase, se hace una planificación de una serie de entregables con el conjunto de funcionalidades del producto (esta fase reemplaza a la planificación tradicional, porque planificar implica predecir en cierta forma el futuro, mientras que la especulación permite designar el futuro como algo incierto).

Exploración.

En esta fase, se obtienen las funcionalidades requeridas, para ser probadas y aceptadas por el usuario final. En esa fase se da mayor importancia a la integración del equipo para el desarrollo de las funcionalidades.

Adaptación.

En esta fase, se revisan las funcionalidades desplegadas, el estado y cómo ha trabajo el equipo, haciendo algunas modificaciones si fuera necesario.

Cierre.

Terminado el proyecto, se celebra aprendiendo de las experiencias vividas.

2.5.3.2 Extreme Programming (XP).

La programación extrema fue desarrollada por Kent Beck en el año 2000, y propuesta en el libro *"Extreme Programming Explained: Embrace Change"* (Beck 2000).
Esta metodología, se centra en desarrollar las relaciones entre personas como forma de llegar al éxito en cualquier desarrollo. Por ello esta metolodogía tiene un diseño muy acorde a lo que se demanda dentro del desarrollo ágil.
En el Extreme Programming se busca la cooperación entre los distintos miembros del equipo, rodeados de un gran ambiente de trabajo.
XP se basa en el feedback continuo entre usuarios y programadores, dando paso a una comunicación fluida entre los miembros, busca la simplicidad de las soluciones y la determinación ante los cambios.
Su nombre tiene relación, a los principios que defiende, aplicando el sentido común, pero de una forma extrema.
Esta metodología es muy útil en proyectos donde los requisitos puedan ser imprecisos, variables, y donde los riesgos sean altos (Ortiz 2010)

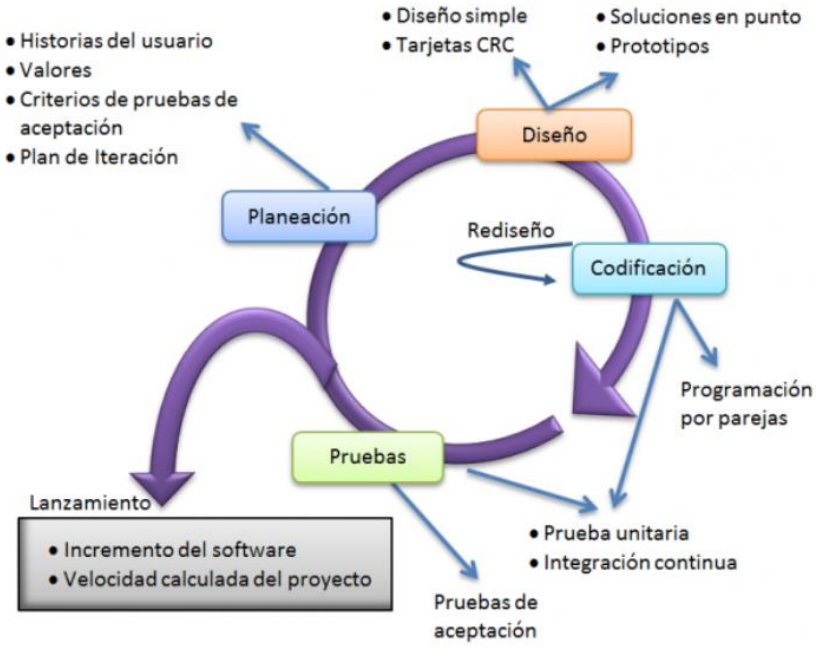


Figura 8. Marco de trabajo de la Metodología XP (Lopez.R.E. 2015)

2.5.3.3 Test Driven Development (TDD)

También conocida como "desarrollo orientado a pruebas", sería como la forma de aplicar el Extrem Programming. Esta técnica, por decirlo de alguna forma, se da en el ciclo de vida de la metodología XP, durante las fases de iteración, producción y mantenimiento. Siendo una técnica orientada al desarrollo de proyectos a través de las pruebas, y debido a su importancia, ya se considera una metodología fuera del Extreme Programming. Como comenta Carvajal (Carvajal Riola 2008), se puede considerar esta forma de desarrollar proyectos una metodología, al presentar normas y formas de desarrollo, que condicionan el desarrollo del proyecto, aumentando su calidad. Es aplicable de forma independiente, o juntamente con otras metodologías como SCRUM, o FDD, o waterfall.

El proceso consiste en iterar 3 procesos que se repiten dentro de esta metodología:

1. Detallar las pruebas para la funcionalidad que se quiere implementar.
2. Codificar dicha funcionalidad hasta que pase la prueba.
3. Recodificar ambos códigos, para estructurarlo.

De esta forma, se itera en bucle, construyendo las funcionalidades del sistema a desarrollar.

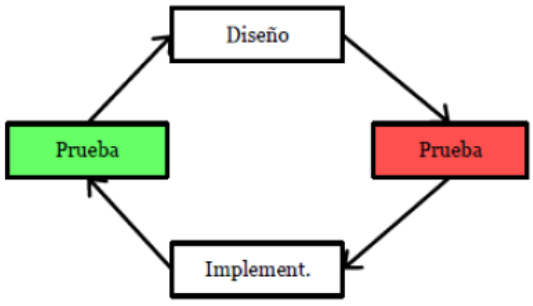


Figura 9. Esquema del proceso TDD. (Gimson 2012)

2.5.3.4 Crystal Methodologies

Metodología basada en la clasificación de la complejidad de un proyecto, en función de una serie de colores, de forma que cuanto más oscuro más complejo será el proyecto. Este método, además propone utilizan un color por proyecto dependiendo del tamaño de este y de su criticidad. Debido a esto, no hay una norma estándar de Crystal, si no que hay una aplicación específica de la metodología para cada proyecto sobre el que se haya aplicado cada tipo de proyecto.

Figura 10. Diagrama de criticidad Crystal (Gimson 2012)

Cada letra de la figura del ejemplo anterior hace referencia a los distintos tipos de riesgos:

- **C**: Pérdida de confort por un fallo en el sistema.
- **D**: Pérdida de dinero (dinero propio).
- **E**: Pérdida de dinero (dinero por préstamos).
- **L**: Pérdida de vidas humanas por un fallo en el sistema.

Los números hacen referencia a las personas que forman el proyecto de la siguiente forma (Gimson 2012):

- *Clear*, para menos de 8 personas.
- *Amarillo*, entre 10 y 20 personas.
- *Naranja* entre 20 y 50 personas.
- *Rojo* entre 50 y 100 personas, siguiendo este patrón con otros colores.

2.5.3.5 Dynamic Systems Development Method (DSDM)

Desarrollo de sistemas centrada exclusivamente en que se satisfagan las necesidades del usuario. Este sistema fue publicado en febrero de 1995, como un método genérico para personas, procesos y herramientas. ***DSDM Consortium*** es la propietaria y administradora del método DSDM y solo sus miembros pueden utilizarlo.

Para el DSDM, el desarrollo se ve como algo iterativo e incremental, en el que el usuario debe estar siempre presente, y en el que hay estar preparado para los cambios, siendo la única metodología que cubre el ciclo de vida del proyecto en su totalidad, al incorporar técnicas de gestión de proyectos, y otras técnicas para garantizar la rentabilidad del proyecto, de acuerdo a la solución prevista.

Las principales características del DSDM son:

1. Alta velocidad de desarrollo, adaptabilidad, y entrega de producto sin que merme la tecnología aplicada.
2. Está considerada como una RAD (Rapid Aplication Development) siendo idónea para proyectos cuyo plazo de tiempo, márgenes de costes son mínimos.
3. Al tratarse de un desarrollo por trozos, y en bucle, tanto usuarios como desarrolladores trabajan codo con codo, dejando de lado la programación de reuniones, pues se encuentran en una "reunión indefinida" al trabajar juntos.

Este método se divide en 5 fases, siendo las 3 últimas iterativas: estudio de viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación.

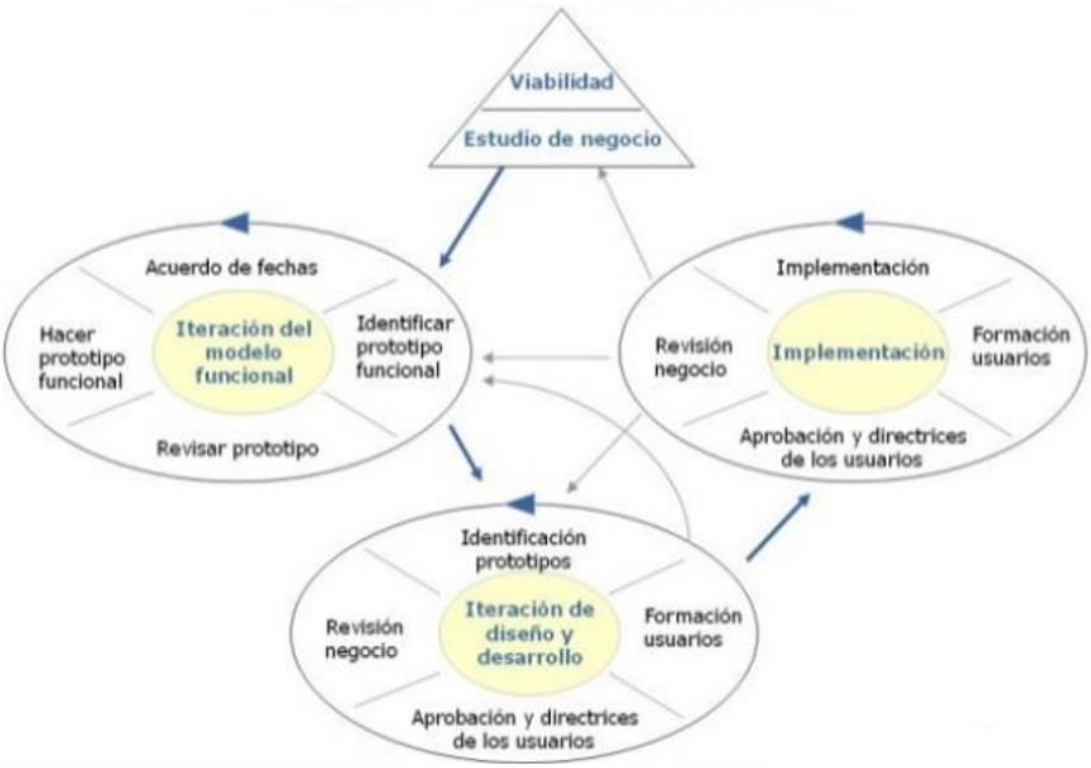


Figura 11. Diagrama del proceso DSDM(Gimson 2012)

2.5.3.6 Adaptive Software Development (ASD)

Fue creada por Jim Highsmith. Sus principales características son:

1. Se trata de un proceso iterativo.
2. También está considerada una RAD (Rapid Aplication Development) siendo idónea para proyectos cuyo plazo de tiempo, márgenes de costes son mínimos.
3. Se centra en los componentes software en lugar de las tareas a desarrollar.
4. Permite introducir modificaciones.

Esta medología propone la ejecución de tres fases: especulación, colaboración y aprendizaje.

Figura 12. Fases Proceso ASD

- **Especulación:** se da comienzo al proyecto planificando las características de lo que se necesita.
- **Colaboración:** se realiza el desarrollo de software del punto anterior.
- **Aprendizaje:** se revisa el software a presentar en términos de calidad, funcionalidad, etc, y se realiza la entrega, o puesta en producción.

Tras estas etapas, la reflexión por parte de todos los miembros permite obtener nuevos aprendizajes, y volver a iniciar el proceso.

2.5.3.7 Feature-Driven Development (FDD)

Se basa en un proceso, creado por Jeff De Luca y Peter Coad en 1997, con iteraciones relativamente breves, que crean un software funcional que los usuarios pueden ver y probar. Cada iteración no debería de extenderse más allá de las 2 semanas, estando centradas las tareas en diseñar y programar el sistema requerido a raíz de un listado de funcionalidades o características que debieran llevar dicho software y que se puedan construir en dicho periodo de tiempo.

Al contrario que el resto de las metodologías, esta metodología no cubre todo el ciclo de vida, centrándose en diseñar y construir el software.

La FDD divide el proceso en 5 fases en los que se va diseñando el software que se necesita:

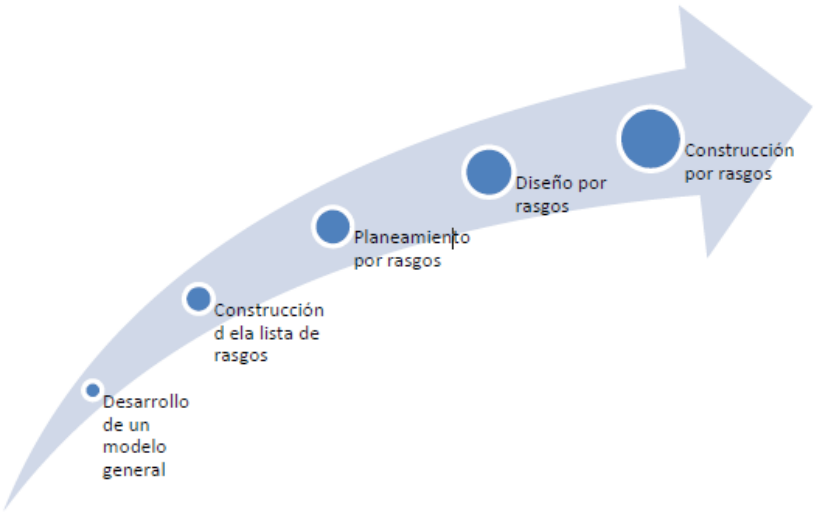


Figura 13. Fases de la Metodología FDD.

2.5.3.8 Lean Development (LD)

Fue creada por Bob Charette en 2003, a raíz de los conocimientos adquiridos al trabajar en el sector automovilístico japonés durante la década de los 80, pero se dio a conocer en 2003 mediante el libro "*Lean Software development: An Agile Toolkit*" de los aurores *Mary y Tom Poppendieck*, donde se establecieron las bases de la aplicación de esta metodología.
El LD, se podría definir como el "*Lean Manufacturing*" que se aplicaba durante la época en Toyota, pero aplicado al desarrollo informático. Se basa en los principios Lean considerando los cambios como riesgos, que si se manejan adecuadamente pueden convertirse en oportunidades para mejorar. Su principal característica es el método de implementación de los 7 principios de la filosofía Lean:

1. **Eliminar los residuos.**
2. **Ampliar el aprendizaje.**
3. **Decidir lo más tarde posible.**
4. **Entregar lo antes posible.**
5. **Potenciar el equipo.**
6. **Crear integridad.**
7. **Visión global.**

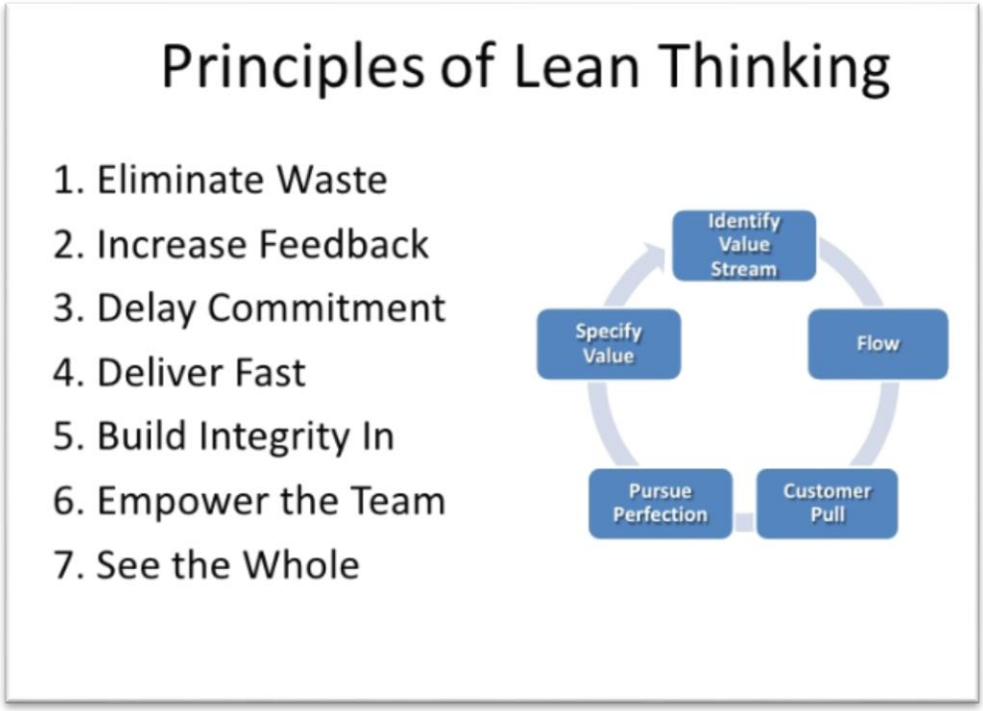


Figura 14. Principios de Lean aplicados al desarrollo Sotware. Disponible en: *www.softwareprojectmanagement.org/lean-software-development*

3 MARCO DE LAS METODOLOGÍAS ÁGILES

En esta sección haremos una introducción sobre el sistema Kanban, y su uso en el desarrollo de proyectos de software, para continuar abordando la metología Scrum: su origen, elementos principales demetodología y herramientas relacionadas con el uso de dicha metodología.

3.1 Kanban

El Sistema Kanban es considerado como el uso de *Lean* al desarrollo de software, siendo este último, quien sirvió de inspiración al *Kanban aplicado a la ingeniería de software*, gracias entre otras personas a David Anderson, Arlo Belshee y Kenji Hiranabe, y a los ya citados autores *Mary y Tom Poppendieck* que con su libro "*Lean Software development: An Agile Toolkit*" sentaron las bases de esta herramienta. Kanban postula que el trabajo en curso (WIP[10]) debería restringirse, de forma que única y exclusivamente comience una tarea, o trabajo cuando la tarea anterior haya sido realizada, finalizada o haya avanzado a un estado posterior. El Kanban (también conocido por utilizarse "tarjetas señalizadoras") indica mediante, en el caso del software mediante una tarjeta, que existen nuevas tareas que pueden ser llevadas a cabo porque las actuales ya han sido acabadas o no pueden seguir avanzando. Según David J. Anderson (Anderson 2010), aunque el uso de Kanban pueda parecer sutil, es capaz de modificar todos los aspectos que rodean a una empresa.

3.1.1 Objetivos.

Los objetivos de Kanban son:

- Equilibrar demanda de trabajo con la capacidad de trabajo.
- Limitar el "work in progress", mejorando el "flujo" de trabajo, descubriendo los problemas de forma anticipada, logrando un ritmo constante.
- Controlar el trabajo (en lugar de las personas) coordinándolo, de forma que se puedan encontrar los encolamientos se puedan tomar decisiones que aporten valor.
- Equipos autogestionados.
- Implementar una cultura de mejora continua.

3.1.2 Beneficios.

Los beneficios que aporta Kanban al desarrollo de proyectos son:

- Ajuste de cada proceso y flujo de valor, a medida.
- Simplificación de reglas para optimizar el trabajo en función del valor que genera.
- Mejor gestión de riesgos.
- Tolerancia a la experimentación,
- Mayor colaboración dentro y fuera del equipo.
- Mejor calidad de trabajo.
- Ritmo de trabajo constante.

[10] WIP: Work In Progress

3.1.3 Sistema.

Los sistemas Kanban constituyen una herramienta para la planificación de los trabajos. En contra de las metodologías de proyectos ágiles estándar, que utilizan periodos de tiempo prefijados (conocidas como *TimeBox*) acordadas al inicio de los mismos, en los que al final de cada iteración se entrega una o varias historias de usuarios[11], los sistemas Kanban se centran en un flujo continuo de trabajo, de forma que los equipos de desarrollo van trabajando de forma contínua, afrontando cada historia de usuario, desarrollándola y entregándola en cuanto esté finalizada, siempre de forma individual.
El proceso del estado del trabajo se entiende a través de un mapa de flujo de valor donde se acuerdan los límites del *"work in progress"* y a través del cual, se comienza a trabajar empujando las distintas tareas a través de las distintas señales, o tarjetas Kanban.

3.1.3.1 Flujo de Valor

El flujo de valor de un proyecto se compone de todos aquellos procesos y subprocesos que, realizados, o en fase de realización, van desde la planificación hasta la entrega y agregan valor al producto.

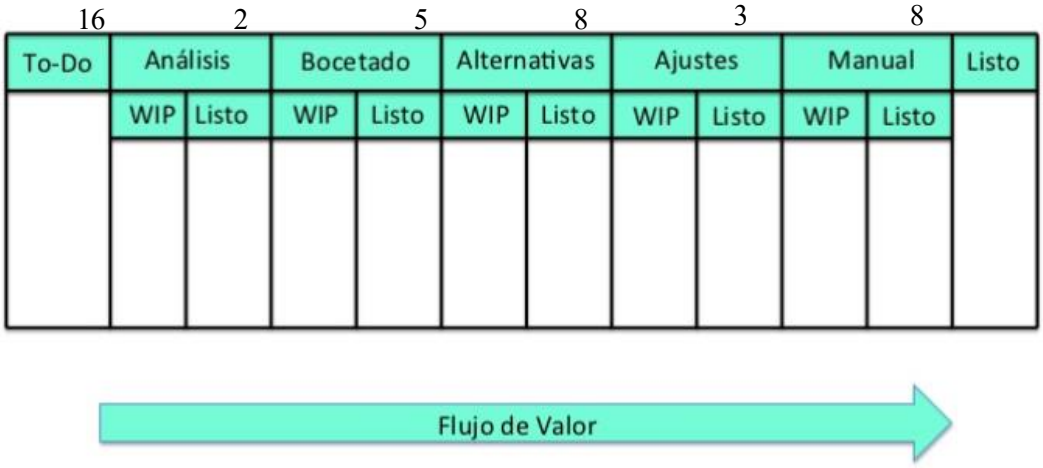


Figura 15. Evolución del flujo de valor en el desarrollo de un proyecto.

Cada proceso (o columna), está compuesta de dos columnas más: una para las tareas en curso (WIP), y otra para las tareas terminadas. Los números encima de cada columna representan la máxima cantidad de tareas que puede estar realizándose de manera simultánea en cada proceso.

3.1.3.2 Límites en Kanban

En el sistema Kanban, es importante poner límites en cada proceso y subproceso. Esto es debido a que si se limita una carga de trabajo a una capacidad asignada, dicha calidad aumenta reduciendo los costes, aumentando también la velocidad y disminuyendo los retrabajos.
Esto se debe a 3 motivos (Gimson 2012):

- La velocidad de las tareas aumenta cuanto menor es la cantidad de tareas por hacer (para una misma etapa.
- El hecho de trabajar en varias tareas a la vez, provoca que una pérdida de tiempo y de concentración entre las tareas en las que se está trabajando.
- Cuanto antes se finalice una tarea, antes se aprenderá de los errores de la misma, y se descubrirán los problemas con antelación.

[11] Historia de Usuario: funcionalidad requerida por un cliente, vista como parte de un sistema.

Otro detalle importante, es el hecho de que la cantidad de trabajo es limitada. Esto se debe a que, debido a la variabilidad de las necesidades del cliente, cuanta mayor cantidad de trabajo se asocie, mayor será la dificultad de cambiar las necesidades, si el cliente cambia también de prioridad u objetivo. En este sentido, en el ámbito de desarrollo de proyectos de software, es más cómodo trabajar no con la cantidad de tareas, si no como la suma de *story points*[12] u equivalente (talla de camisetas[13], etc.)

3.1.3.3 Tablero Kanban

La aplicación de Kanban al desarrollo de proyectos, puede verse de una forma rápida gracias a la agrupación de las tareas en un tablero, en el que los post-its que representan dichas tareas, de forma que sea visible y de un vistazo sea posible saber en qué está trabajando cada miembro del grupo, y la carga de trabajo a la que se está viendo sometido cada uno, de forma que el ritmo de trabajo sea unísono.

Este tablero, se suele colocar en una zona visible dentro del espacio de trabajo, de forma que todo el mundo pueda verlo sin problemas, y debe ser continuo, de forma que las tareas no vayan acumulando post-its hasta que la tarea haya sido terminada, sino que conforme se vaya ejecutando una tarea, las nuevas tareas relacionadas con esta última (fallos, mejoras, o nuevas tareas), se dispongan al comienzo del tablero, de forma que en la revisiones con los usuarios, se prioricen las tareas que le sean prioritarias.

Un tablero Kanban, ha de tener toda la información necesaria, para que el grupo de trabajo, sea capaz de decidir sin tener que consultar o sin ser supervisados.

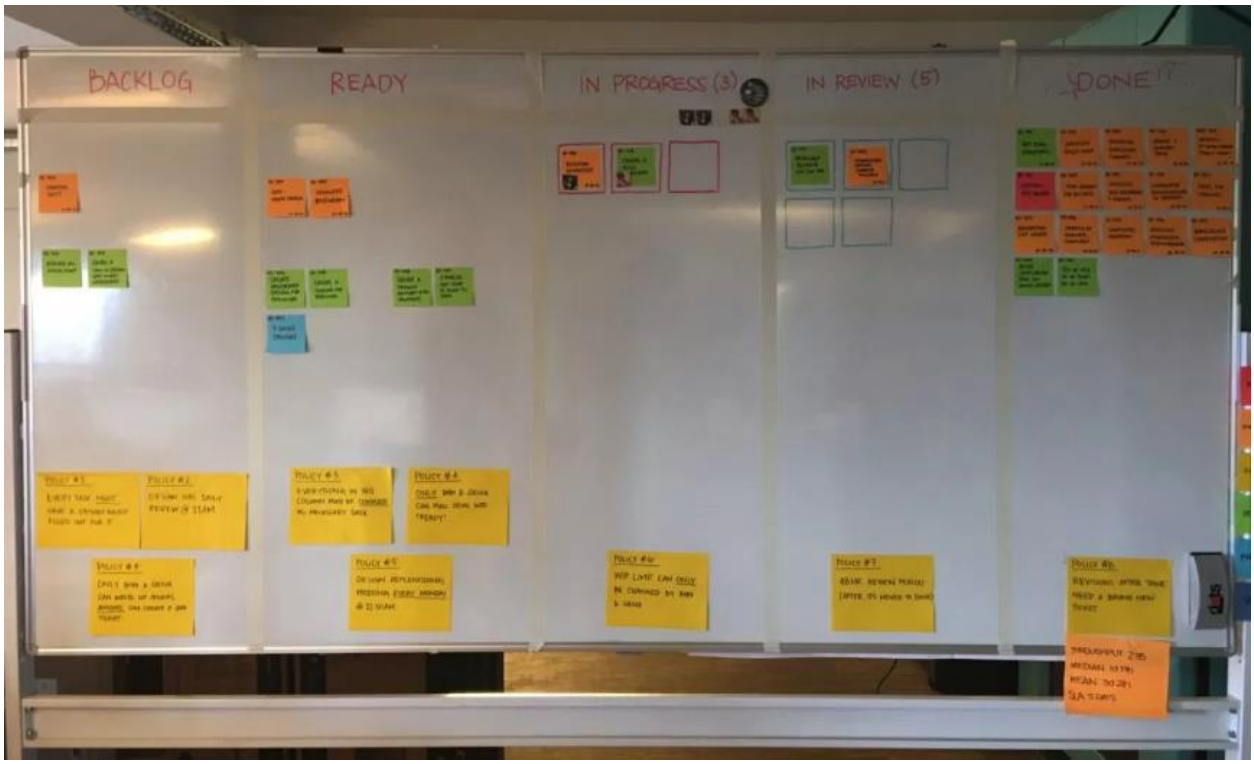


Figura 16. Tablero Kanban

[12] Story Points: Puntuación que se asiga a las historias de usuario por parte del equipo que desarrollará las funcionalidades, en función de las distintas tareas a realizar. Primero se llega a un consenso para establecer una medida de referencia, y posteriormente se puntua siguiendo la serie de Fibonacci, de forma que, si hubiera 2 valores extremos, tendrían que justificar al resto, el motivo de haber elegido la carta, y volver a votar.

[13] Camisetas: En ocasiones, cuando ya se tiene cierta experiencia en la medida de los puntos que conlleva una serie tareas y/o historias de usuario, se emplea el tallaje de las camisetas para estimar la complejidad de la historia de usuario (xs,s,m,l,xl,xxl,xxxl,etc.)

En cada equipo o grupo de trabajo, las normas se definen dentro del mismo, pudiendo establecerse cuantos criterios necesite el equipo para poder trabajar, como por ej:

- Cada columna representa un proceso, con la marca del límite de tareas máximas simultáneas.
- Las tarjetas blancas representan productos en proceso para realizarse.
- Las tarjetas amarillas representan tareas a realizar para terminar un producto o subproducto.
- Los avatares (emoticos) representan a los responsables de cada tarea. No se suele nombrar al responsable de cada tarjeta porque puede ser tratada por diferentes personas en diferentes momentos.
- Las tarjetas verdes representan tareas completadas.
- Las tarjetas rojas representan tareas bloqueadas por otras, o limitaciones encontradas durante el proceso.
- Existen también reglas de decisión que indican el procedimiento a la hora de modificar las tareas en el tablero, pero también pueden contener información relevante para el proceso, notas, feedback, que se suelen colocar en la parte inferior de cada proceso del tablero.

Cada tarjeta puede tener los siguientes datos:

1. Fecha de la tarea y de entrada y salida de cada proceso.
2. Fecha comprometida (si aplica).
3. Descripción.
4. Prioridad.
5. Estimación de la duración.

El sistema Kanban puede entremezclar varios proyectos y sus tareas asociadas. El propósito es mantener un continuo ritmo de trabajo constante, con las tareas encoladas para ser realizadas de forma que no se pierda el ritmo y se tenga un seguimiento continuo.

Como ventaja, Kanban permite no solo llevar la gestión de uno o varios proyectos, sino también tener presente las incidencias que van sucediendo y tenerlas en cuenta en próximos pasos.

3.1.3.4 Indicadores.

Kanban dispone de varios indicadores como pudiera ser el límite de trabajo, que permite decidir si se debe comenzar un nuevo trabajo. De esta forma, y teniendo presente cuándo se plantearon estos límites se pueden regular los problemas que surgen en los procesos, así como liberar parte de recursos asignados a otras tareas, si fuera necesario.

Además, las colas de tareas por hacer, y en los propios procesos, no dan pie a malas interpretaciones, porque todo el mundo puede ver en qué tarea está trabajando cada miembro, y con qué prioridad.

3.1.3.5 Funcionamiento y problemas en Kanban

Una vez que comienza el equipo a trabajar, se suele lograr un ritmo constante, equilibrado y sostenido en el tiempo conforme las tareas se van realizando, y cada miembro se habitua a esta nueva forma de trabajar.

Una de las situaciones que se deben evitar es la generación de cuellos de botella, ya que además de parar el ritmo de trabajo, son origen de errores, fallos, y olvidos que se pueden, y deben evitar. Esto puede ser debido a una mala definición de límites, problema de limitación de recursos, o una oportunidad de mejora en otros procesos. En caso de producirse, lo mejor es usar recursos libres para desatascar las colas, y examinar el proceso a posteriori para buscar mejoras.

Otra situación que puede darse es la falta de asignación de tareas, porque una persona haya completado todas sus tareas, pero no pueda proseguir porque la tarea con la que comenzar, todavía no la ha terminado la persona de la que depende en ese momento dicha tarea. De nuevo, puede deberser a un problema de definición de límites, recursos, y una oportunidad de mejora.

Una situación adicional que se puede presentar se da cuando un recurso asignado a una tarea se libera y otro miembro aún no ha finalizado su tarea. En estos casos, la persona que no tengan tareas asignadas en ese momento podría asignarse otra tarea de la cola de tareas o ayudar al compañero a terminar la suya. La regla no escrita, dice que un recurso libre debería ayudar al compañero a terminar el trabajo siempre que sea posible. De esta forma cuando los dos finalizan dicha tarea, pueden asignarse dos tareas de la cola de tareas.

3.2 Scrum

3.2.1 Breve Historia sobre Scrum

Scrum forma parte de una evolución del modelo *Shashimi*, que fue concebido en Japón como una variante a la metodología waterfall, que buscaba una mayor velocidad, flexibilidad, y un mayor feedback al final de cada etapa del modelo en cascada.

La primera vez que apareció el término *Scrum* fue en el año 1986, de la mano de Hirotaka Takeuchi e Ikujiro Nonaka en *The New New Product Development Game* (Takeuchi y Nonaka 1986) donde se hizo una recopilación de nuevos métodos de desarrollo exitosos,utilizados en empresas de Japón y en Estados Unidos (como por ej. las cámaras fotográficas Canon, fotocopiadoras de Xerox, vehículos Honda, los ordenadores de Hp,etc…). Los equipos detrás de estos productos fueron capaces de lanzar al mercado, partiendo de requisitos muy generalistas, de funcionalidades, y, por tanto, productos muy novedosos, en mucho menos tiempo que la competencia, compartiendo ciertos patrones en el desarrollo de proyectos. En dicho estudio presentado por los autores anteriormente citados, se realizó una analogía entre la manera de trabajar entre los equipos de desarrolladores, y la colaboración entre los jugadores de un equipo de rugby. La palabra Scrum tiene el mismo origen que en el deporte del rugby, donde se utiliza en la jugada en la que se debe devolver el balón que ha salido del campo de forma colectiva (Scrum significa *melé* en castellano).

Scrum no fue puesto en práctica hasta 1993, cuando Jeff Sutherland aplicó el modelo al desarrollo informático para la compañia *Easel Corporation*. En 1996, Jeff Sutherland, y Ken Schwaber presentaron las las normas que usaron para los desarrollos informáticos en OOPSLA 95 en Austin, Texas (Conferencia Internacional sobre programación, lenguajes y aplicaciones que es realizada cada año por la **A.C.M.** (Association for Computing Machinery))

Posteriormente, Schwaber y Mike Beedle presentaron SCRUM en 2001 mediante el libro *Agile Software Development with Scrum*.

Actualmente se encuentra publicada la Guía de Scrum (*The Scrum Guide*), como la referencia oficial de normas de Scrum, escrita por Ken Schwaber y Jeff Sutherland.

La última versión de esta guía fue publicada en noviembre de 2017.

3.2.2 Descripción general de Scrum.

Según la Guía del Conocimiento de Scrum (Satpathy 2017) un proyecto Scrum consiste en "*un esfuerzo de colaboración para crear un nuevo producto, servicio u otro resultado tal como se define en la declaración de la visión del proyecto" (Project Vision Statement)*.

Los proyectos suelen estar perjudicados por motivos de tiempo, coste, alcance, calidad, recursos, y demás restricciones que dificultan la planificación, y ejecución de estos. No obstante, la finalización de un proyecto en tiempo y forma con éxito repercute de forma económica a la empresa. Por ello, es necesario que las empresas sean capaces de implantar una metodología capaz de gestionar todos estos aspectos de forma adecuada.

Scrum es un *framework*[14] adaptable, rápido, iterativo, eficaz, y flexible concebido para dar valor a un proyecto de una manera ágil y rápida. Scrum permite mejorar la comunicación y crear un entorno de responsabilidad común entre los participantes del proyecto, y una tasa de avance de forma constante.

Scrum, por cómo está definido, es compatible con todo tipo de productos y todo tipo de proyectos, independientemente de la complejidad.

El desarrollo de Scrum, se fragmenta en pequeños periodos de tiempo de trabajo, en los que se trabaja de una forma concentrada, llamados ***Sprints***[15]. Estos sprints, permiten un alto ritmo de trabajo, en parte debido a que los equipo son altamente cualificados, y con las autorizaciones suficientes como para tomar decisiones relativamente importante sobre la marcha, eliminando tiempos de espera que no aportan nada.

Scrum se basa en tres pilares fundamentales:

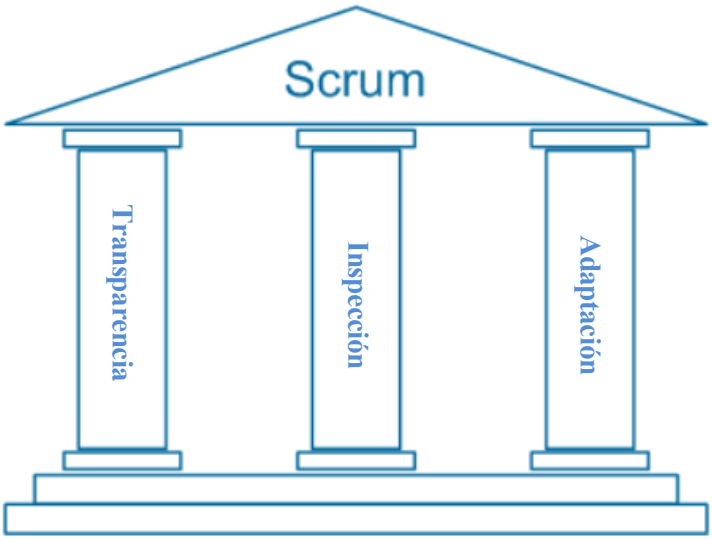


Figura 17. Pilares de la Metodología Scrum

3.2.3 Roles de Scrum.

En Scrum, existen una serie de roles que se aglutinan en 2 conjuntos de personas: de una parte, estarían las personas que deben estar involucradas tanto en el proyecto como en Scrum, y de otra parte, están las personas que no siendo imprescidibles para el desarrollo del proyecto, o la aplicación de Scrum, su presencia en determinadas situaciones es necesaria.

Estos 2 grupos, también son conocidos como el grupo de los *cerdos* y el de las *gallinas*, por una viñeta muy conocida en relación con un cerdo y una gallina, y la intención de montar un restaurante entre ambos:

Figura 18. Viñeta humorística relacionada con los roles en Scrum

[14] Framework: Marco de Trabajo
[15] Sprint: Con sprint, se hace referencia al conjunto de tareas y acciones que se hacen dentro del Sprint. Un proyecto se divide en *n* sprints.

Roles Comprometidos

- **Product Owner**: Representa la voz del cliente, toma las decisiones, y es la que conoce realmente las necesidades del producto como usuario. Escribe las ***historias de usuario***, y las prioriza en el ***Product Backlog***.
- **Scrum Master**: Es la persona encargada de garantizar que se aplica la metodología. Se encarga también de que el equipo no se desvíe de sus tareas, y trata de resolver aquellos problemas que impidan la consecución del ***sprint***. Realizando tareas de coordinación, dirige los ***scrums diarios***, y sigue el avance del proyecto.
- **Equipo**: Está compuesto por menos de 8 personas (en caso de haber más, se dividiría en más equipos que trabajarían sobre el producto backlog del mismo). Son responsables de desarrollar y presentar el producto, tomando decisiones, pues tienen asumidas responsabilidades de decisión y de gestión. Son capaces de estimar el esfuerzo de las tareas del **product backlog**, y entregar un grupo de tareas del producto Backlog al final del Sprint.

Roles Involucrados

- **Usuarios**: personas para las que se diseña el producto.
- **Stakeholders o Interesados (Clientes, Proveedores)**: personas que, de forma indirecta, están relacionados con el proyecto porque de alguna forma les afecta (ya sea en positivo o en negativo). Participan durante la revisión de cada **Sprint.**
- **Managers**: Son aquellas personas encargadas de preparar el marco de trabajo en el que se desarollará el producto. Además, participan en la toma de decisiones en situaciones bloqueantes o de atasco.

3.2.4 Elementos en Scrum.

Los principales elementos de Scrum que son necesario conocer para entender el proceso, si bien se utilizarán otros que se irán describiendo más adelante, son (José, H., Canós, P. L., & Carmen 2003):

3.2.4.1 Product Backlog.

El Product Backlog consiste en una lista de funcionalidades necesarias para negocio priorizada en función de las necesidades. Estas funcionalidades están basadas en descripciones a muy alto nivel de lo que se espera que realice el producto, en el que se detallan de una forma resumida, las características, funciones, mejoras que deberían tener el producto. Esta lista, supone el comienzo de la recopilación de los requisitos del producto, y de los nuevos que irá adquiriendo el producto en su evolución.

El Product Backlog debe ser gestionado por el negocio, que, con la ayuda del Scrum Master, le dará una idea del coste estimado, y contendrá todo lo que aporte un valor final al producto.

Las principales características son:

- Debe incluir todas las necesidades del producto, expresados mediante historias de usuario.
- Para cada necesidad, se incluirá el valor que aporta al negocio y un coste estimado, de forma que puedan ser priorizadas basándose en el ROI[16].
- Se tendrán que indicar el número de sprints o iteraciones, y los despliegues acordados.
- Se han de incluir los riesgos que puedan contemplarse, y cómo se van a mitigar.

Antes de comenzar a desarrollar deben definirse los objetivos, y las necesidades del producto. Sin tener que entrar en mucho detalle, basta con que tenga el contenido suficiente para poder empezar a trabajar.

[16] ROI: Retorno de Inversión

La aplicación de este conjunto de características hará que:

- El proyecto no se detendrá ante necesidades poco claras, permitiendo obtener los resultados sobre los que se está trabajando desde el principio, antes.
- El resto de las necesidades irán surgiendo a lo largo del avance del desarrollo, no perdiendo tiempo en analizar dichas necesidades al comienzo.
- Estas necesidades secundarias, es posible que no sean necesarias al estar ya cubiertas, porque hayan surgido nuevas más interesantes, o que se descarten por su coste. Una vez definidos los requisitos se tendrá que acordar cuándo se tiene que entender un objetivo como terminado o completado.

Un producto o entrega, se dará por finalizado si:

- Se puede generar un documento que contenga una demostración con todas las necesidades que se pedían al comienzo, cubiertas y operativas.
- Contiene todo aquello que se está realizando sobre el producto y que así ha sido solicitado por el cliente.

Adicionalmente a la determinación de una funcionalidad completada, se deberían establecer una serie de criterios o condiciones que definan cuándo, o en qué momento se dará por buena la prueba, o la aceptación de las funcionalidades que se solicitaban al comienzo de la iteración.

Todos estos aspectos, harán que el Product Backlog vaya evolucionando con el proyecto, dando por completadas funcionalidades, surgiendo nuevas, descartando otras existentes, o re-priorizando las necesidades que queden por desarrollar.

A pesar de no haber una norma específica a la hora de hacer un Product Backlog, se recomienda que contengan por lo menos, información relacionada con:

- Identificador de la funcionalidad.
- Descripción de la funcionalidad.
- Orden de prioridad de la funcionalidad.
- Estimación de coste/esfuerzo de llevar a cabo la funcionalidad.

ID	Prioridad	Historia de Usuario	Estimación
1	Muy alta	Motor de Cálculo	250
2	Alta	Criterios de Intervención de Cálculo	210
4	Baja	Registro de errores	130
5	Muy baja	Consulta de facturas	90

Tabla 2. Ejemplo de Product Backlog.

3.2.4.2 Historias de Usuario.

Las historias de usuario son el conjunto de funcionalidades necesarias para el negocio, que necesita que sean implementadas en el desarrollo software. Lo lógico es que sean diseñadas de forman común entre usuarios y resto del equipo, de forma que irán variando a lo largo del desarrollo.

Cada historia de usuario cumple una serie de características, también conocidas como "*Las 3 C*":

- **Card:** breve descripción a modo de recordatorio de la historia.
- **Conversation:** Pequeño resumen que garantiza que todos han entendido lo que se necesita y lo que quiere llevar a cabo.
- **Confirmation:** casos de prueba que describirán si la funcionalidad cubre lo que se necesita

En cuanto al modelo, existen varios tipos, pero podría ser uno parecido al siguiente:

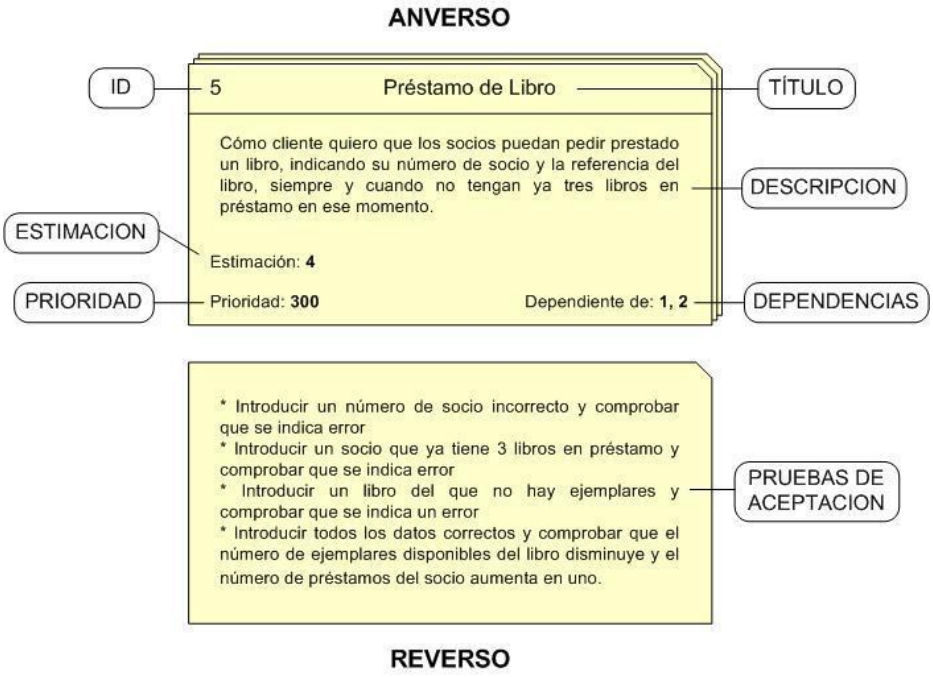


Figura 19. Ejemplo de Historia de usuario. Fuente: *devnettips.blogspot.com.*

- **ID**: Identificador de la historia de usuario.
- **Título**: breve descripción de la historia.
- **Estimación**: Medida del esfuerzo (criterio establecido al comenzar el proyecto) para implantar la necesidad.
- **Prioridad**: Rango de prioridad respecto al resto de historias, siendo más prioritario cuanto mayor sea dicho valor. Otra aproximación a la priorización se realiza mediante algún método de priorización, como el método **Moscow:**
    - **Must**: Es necesario terminar el requisito para finalizar el proyecto.
    - **Should**: Es necesario completar la funcionalidad como sea, pero el éxito del proyecto no depende de ella.
    - **Could**: Se podría completar esta funcionalidad, si la implementación de esta no afecta a la consecución de los objetivos principales.
    - **Would**: Se podría llevar a cabo la funcionalidad si sobrase tiempo.
- **Dependencias:** Una historia de usuario debería ser independiente, pero no siempre es así.

3.2.4.3 Sprint Backlog.

Está compuesto por el conjunto de tareas que componen un sprint. Mediante este, se identifican y definen todas las tareas necesarias para completar el sprint.

Mientras que en el Sprint Backlog aparecen las distintas funcionalidades que se necesitan, por otro lado, también aparecen todas las tareas necesarias para completar dichas funcionalidades, de forma que el Sprint Backlog es una parte del Product Backlog (como un subconjunto).

También al estar las historias de usuario ya priorizadas, las tareas asociadas a cada historia, ya vienen priorizadas por la propia historia de usuario que fue ordenada durante el product backlog.

Sin embargo, el hecho de que puedan existir tareas que no se hayan terminado en un anterior sprint, harán que, en el comienzo del siguiente, se acumulen dichas tareas.

Todo sprint backlog debe ser aceptado por el equipo del proyecto (sobretodo por los programadores), quedando a criterio del equipo la posibilidad de modificarlo.

En caso de necesitar añadir nuevas funcionalidades de forma urgente, deberán ser incluidas en el producto backlog, y tenidas en cuenta para el siguiente sprint.

Un ejemplo de Sprint Backlog, podría ser el siguiente:

Entrada	Historia de Usuario	Estimación	Prioridad	Aceptada (Sí/No)	Observaciones
1	Poder visualizar tendencia de consumo	250	1	Sí	
2	Poder consultar histórico facturas	210	2	Sí	
3	Poder consultar histórico de contratos	150	3	Sí	
4	Poder descargar facturas	130	4	Sí	
5	Poder reclamar una factura	90	5	No	
6	Disponer de un buzón de correo	50	6	No	
7	Hacer Login	40	7	No	
8	Pagar una factura	30	8	Sí	
9	Mandar un mensaje	10	9	Sí	
10	Mostar notificación al usuario	5	10	Sí	

Tabla 3. Ejemplo de Sprint Backlog.

Las características son:

- Es un listado de necesidades para el usuario ordenado por prioridad.
- Podrían existir dependencias entre ellas, y en ese caso habría que reflejarlo.
- Las tareas deben tener un esfuerzo estimado entre 4 a 16 horas.

El formato posible para hacer estas listas es variado:

- Hojas de Cálculo.
- Pizarras.
- Herramientas Colaborativas (Trello, Microsoft Teams, …)

Estas tareas se gestionan mediante un *Scrum Taskboard*, donde cada objetivo tiene asociadas las tareas necesarias para completarlo mediante post-its que van cambiando de columna en función del su estado de avance.

Dicho tablero cumple las siguientes características:

- Contiene las tareas a llevar cabo.
- Incluye a la persona que lleva la tarea, estado de esta, y plazo para finalizarla.
- Permite la consulta diaria por sus miembros.
- Permite ver cada día lo que le resta a cada tarea.

3.2.4.4 Incremento.

Un incremento se define como la suma de todos los ítems completados en el Sprint backlog. Si hubiera ítems incompletos, deben ser incluidos en el Product backlog para que sean realizados en el próximo sprint. Un ítem se dará por finalizado si este es funcional. El sumatorio de todos ellos debería de dar como resultado el producto final, preparado para ser puesto en producción o desplegado, tras la ***Revisión del Sprint*** con algunas tareas, aunque a veces se necesiten acciones adicionales (pruebas o documentos)

3.2.5 El proceso de Scrum

Scrum establece un marco para la gestión y desarrollo (entendiendo desarrollo como toma de requisitos, la fase de diseño funcional, y codificación posterior) de proyectos, en el que se definen una serie de iteraciones de una duración máxima de 4 semanas, conocidas como ***sprint***.

Cada ***sprint***, estaría compuesto por 3 fases:

1. **Pre-sprint**: planificación de lo que se va a llevar a cabo en el sprint.
2. **Sprint**: periodo de trabajo.
3. **Post-sprint**: revisión y retrospectiva del sprint.

El ciclo de Scrum se podría resumir en:

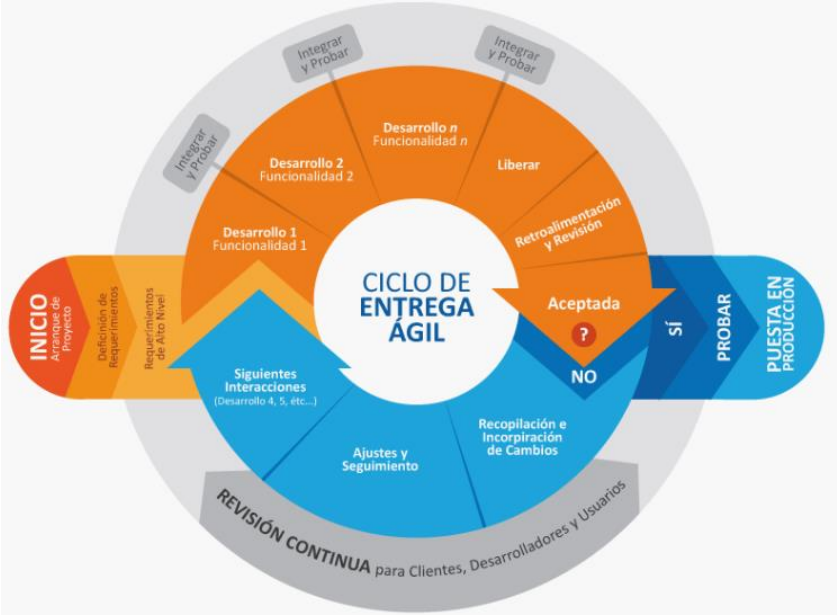


Figura 20. Ciclo de vida de un proyecto Scrum. Fuente: *northware.mx*

3.2.5.1 Planificación Pre-Sprint.

También llamado **sprint 0**, es la primera fase en la que se escuchan las necesidades del usuario, con el objetivo de que las posteriores decisiones que se realicen en las siguientes etapas, agreguen siempre valor.
Para este sprint, el *product owner* debería tener ya elaborado y priorizado un **Product Backlog**. En dicha reunión, en la que se encuentran el product owner, el equipo de programadores, y el scrum master, el dueño del producto define las funcionalidades que se deberían incluir en el sprint y se definen las tareas necesarias para llevar a cabo dichas funcionalidades. Generalmente, en esta etapa se producen errores en la estimación, ya que se hacen por encima, por lo que es aconsejable no parar mucho en esto, e invertir dicho tiempo en otros aspectos. Además, el equipo determina la lista de tareas y estima cuánto puede cumplirse para ese sprint.
Las tareas que se realizan en este sprint 0 son:

- **Definir el objetivo**: se indican los objetivos del proyecto a alto nivel de forma que los componentes del squad, puedan comprender las necesidades. El objetivo del sprint debe ser posible aunque haya tareas incompletas o abandonadas: "*La razón del objetivo de un sprint es dar al equipo cierta flexibilidad respecto a la funcionalidad*" tal como Ken Schwaber y Mike Beedle lo enuncian (Schwaber y Beedle 2001). El objetivo sirve para no perder el foco en las tareas que deben ser llevadas a cabo como principales. Además, el objetivo sirve para que el equipo se se encuentre animado y sean capaces de integrar cualquier circunstancia que pueda darse mientras dura el sprint.

- **Definir "terminado":** esto permitirá saber cuando se ha finalizado una tarea.
- **Definición del Backlog inicial**: Esto permitirá que el sprint que se comience contenga una cantidad suficiente de tareas para llevar a cabo. Dicha lista, como ya comentamos anteriormente, debe ser priorizada por el product owner, de forma que conforme se vayan realizando sprints, se vayan añadiendo nuevas funcionalidades al producto de forma que se vayan cumpliendo las expectativas del usuario.
- **Definición de los entregables**: Aunque anteriormente se haya comentado que la entrega de documentación pasa a un segundo plano con Scrum, sí que es cierto, que sería bueno para el proyecto, establecer cuándo se van a realizar estos documentos para añadir valor al producto, y poder tener un mayor feedback.

Este plan de entregas puede variar por diversos motivos como:

- Cambios en las necesidades, o nuevas necesidades para los usuarios.
- Aparición de nuevas funcionalidades que añaden valor.
- Cambios en el entregable.

Este plan lo debe definir el usuario, que debe determinar las funcionalides y el coste asociado que asumirá. También, dicho entregable estará condicionado por el equipo de trabajo debido a:

- **El tiempo necesario para desarrollarlo.**
- **La estimación de tiempo y coste.**
- **Selección de tareas del Backlog del producto a asumir.**

Una vez planificado el sprint inicial, se realizará una reunión con todas las personas del equipo para:

- Dimensionar el proyecto.
- Revisar el product backlog.
- Establecer el horario de trabajo, y las próximas reuniones.

Tras esto, el equipo en su conjunto comenzará a trabajar hasta ajustar los recursos con la cantidad de trabajo planteada para el primer sprint, teniendo definido el **Sprint Backlog**.

3.2.5.2 Estimaciones del Sprint Backlog

Para poder comenzar a planificar en la primera reunión de planificación, el equipo de programadores debe saber cuáles son sus capacidades de trabajo, y a qué ritmo puede ir realizando las distintas tareas.

Para ello, primeramente, se han de decidir qué historias se incluirán en la pila de historias de usuario del primer sprint.

La forma para decidir qué historias incluir, puede realizarse de dos formas:

1. Mediante tanteo.
2. Mediante estimaciones de velocidad de trabajo, basados en datos reales.

- **Mediante tanteo:**

En esta situación, los miembros analizan la cantidad historias de usuarios hasta alcanzar un consenso. Este método es útil en sprints cortos.

- **2.Mediante estimaciones de velocidad de trabajo:**

Este método se realiza en 2 fases:

A. Elegir una velocidad estimada.

B. Determinar la cantidad de historias realizables sin alcanzar la velocidad estimada.

La forma óptima de estimar sería revisando el historial del equipo basándonos en sprints anteriores que pudieran dar una idea de la velocidad del equipo.

Una de las formas para realizar esta técnica, sería utilizando un equipo que tenga un mismo historial, de forma que los sprints que se ejecuten sean similares a los ya realizados anteriormente por los miembros en equipos anteriores, y que en este compartan dimensionamiento y condiciones.

Otra forma de realizar la técnica sería mediante el cálculo de recursos:

1º. Cálculo del número de días - hombre disponibles (aunque el dato sea poco significativo, porque influyen factores externos que pudieran alterar la dedicación).

Nombre	Días
Usuario 1	15
Usuario 2	8
Usuario 3	9
Total	32 días-hombre

2º. La velocidad estimada se calcula como:

$$\boldsymbol{Velocidad\ Estimada} = (Dias - hombre\ disponibles) * (factor\ dedicación)$$

3º. El factor de dedicación se estima en función del estado del equipo, siendo razonable, estimarlo en función del estado de los últimos sprints.

$$\boldsymbol{Factor\ dedicación} = \frac{(Velocidad\ Real)}{(Dias - hombre\ disponibles)}$$

La velocidad real sería el sumatorio de las estimaciones que se llevaron a cabo para el último sprint.

Ejemplo:

Supongamos que un equipo realizó 30 historias de usuario, con 4 personas, sumando 40 dias-hombre, para calcular la velocidad para el próximo sprint, con un total de 50 días-hombre, el resultado sería:

$$\boldsymbol{Factor\ dedicación} = \frac{30\ historias}{40\ \ días - hombre} * 100 = 75\ \%$$

$$\boldsymbol{Velocidad\ estimada} = 50\ días - hombre * 0{,}75 = 37{,}5\ \ puntos$$

Para equipos nuevos, puede tratar de comporarse con equipos similares, o usar un valor inicial para el factor de dedicación. Se suele recomendar el 70%.

3.2.5.3 Planificación del Sprint.

Conocido como **"Sprint Planning Meeting"**, se trata de una reunión entre product owner, scrum master, y equipo, para elegir que funcionalidades del product backlog se realizarán en el sprint.

Para esta reunión, como comentamos anteriormente, el product owner debe de tener ya realizada su lista de funcionalidades del product priorizada, de forma que en la reunión se trate de ver, cuáles se pueden aglutinar en un sprint.

El *time-box*[17] de esta reunión es de 8 horas, divididas en 2 partes de 4 horas:

Primera parte de la reunión:

- Se eligen las historias que pasarán a ser entregables.
- El equipo puede opinar sobre las historias, y de su capacidad para llevarlas a cabo en el sprint, pero será el product owner el que tendrá la última palabra.
- El equipo decidirá qué podrá implantar, de todo lo seleccionado por el Product Owner para el sprint.

Segunda parte de la reunión:

- El equipo de desarrollo consultará todas las dudas que pueda tener sobre el Product Backlog al Product Owner.
- El equipo transformará las necesidades o historias en un entregable.

Como fruto de la reunión, se obtendrá, el "**Sprint Backlog**" con todo lo necesario (tareas, especificaciones, estimaciones, y reparto de tareas), para comenzar a desarrollar.

Las características del Sprint Backlog son:

- Las estimaciones de tiempo tendrán una duración de entre 4 y 16 horas. En caso de valores superiores, se dividirán.
- Las tareas deben ser estar enfocadas a la realización de una de las necesidades del Product Backlog.
- El avance del progreso y la velocidad se controlarán mediante gráficos ***Burndown Chart***[18] (explicado más adelante).

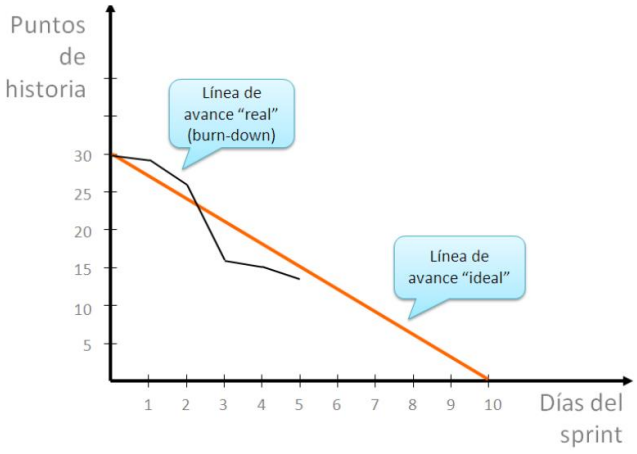


Figura 21. Gráfico de Burndown de seguimiento.

[17] Time-box: Hace referencia a la duración de la reunión. Terminología utilizada también para referir la duración de un sprint.
[18] Burndown Chart: También conocido como gráfico de quemado, en el sentido de cuánto se lleva gastado o realizado en el proyecto, comparado con el ritmo teórico: tiene cierta equivalencia al Valor Ganado o EVM.

El sprint backlog puede presentarse y, por tanto, ser controlado y actualizado en distintos soportes:

- Hojas de cálculo (tipo Excel)
- Pizarra.
- Soportes digitales (tipo Jyra por ej.).

La herramienta más usada es el ***Scrum Taskboard*** consistente en una pizarra, en la que se reflejan las siguientes entidades:

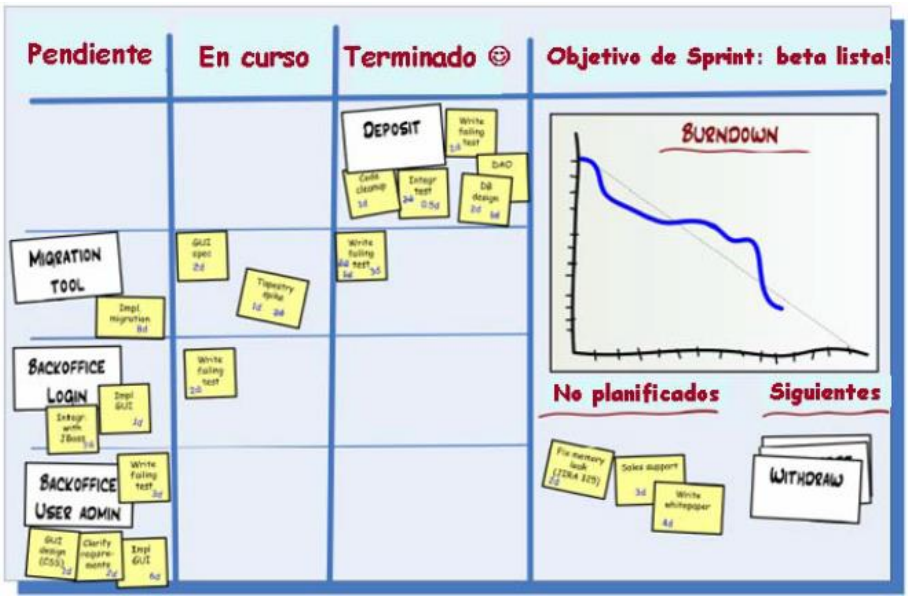


Figura 22. Tablero Scrum Taskboard. (Kniberg et al. 2007)

- **Tareas Pendientes**: Tareas planificadas que aún no se han comenzado.
- **Tareas en curso**: Tareas planificadas que están siendo llevadas a cabo, por una o varias personas del equipo.
- **Tareas Terminadas**: Tareas planificadas finalizadas.
- **Tareas no planificadas**: Tareas no planificadas pero que puede ser interesante tenerlas como referencia.
- **Tareas Siguientes**: Tareas no planificadas, que pueden ser llevadas a cabo si se acaban todas las tareas del Sprint antes de que este finalice.
- **BurnDown**: Gráfico de avance, que irá reflejado el avance diario.
- **Impedimentos**: También suele reflejarse una zona donde se colocan aquellas tareas bloqueantes.
- **Retrospectiva**: Otra zona donde se reflejan las enseñanzas que merezca tener presentes, tanto positivas como negativas.

3.2.5.4 Estimación del Sprint: *Planning Póker*

La estimación de la complejidad de las distintas tareas que pueden componer un sprint es una tarea de análisis que debe ser consensuada con todo el equipo que conforma el squad.

Para poder llevarla a cabo, de una forma desenfadada, y que las distintas opiniones no condiciones al resto de participantes, se recurren a técnicas de opinión indirectas, como la del **Planning Póker.**

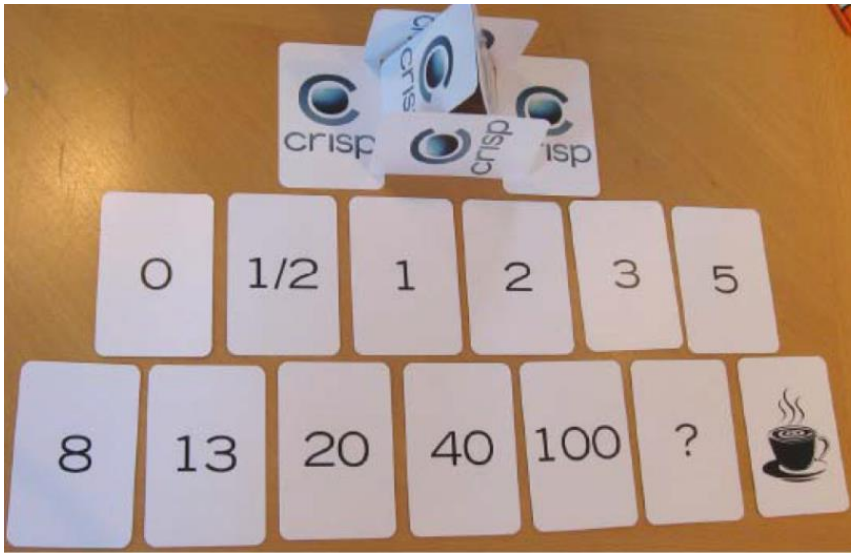


Figura 23. Baraja de cartas de Planning Póker.

El juego de valoración se lleva de la siguiente manera:

1º. Se reparte a cada miembro una baraja de 13 cartas.

2º. Se muestra una historia de usuario y cada miembro ha de elegir una carta en función de su estimación de dificultad/esfuerzo, y ponerla boca abajo.

3º. Cuando todos han escogido carta, se les da la vuelta al mismo tiempo.

4º. Se revisan las cartas, y si existiesen grandes diferencias, se debate (siendo los valores extremos quienes tengan que justificar su valoración) y se tratan las razones para haber valorado de dicha forma la historia. Se repite posteriormente la votación, hasta que las estimaciones sean parejas o uniformes.

5º. El valor estimado es el valor de puntos de trabajo necesarios para el desarrollo de dicha historia, de ahí que la secuencia de números no sea lineal y, por ejemplo, haya un salto entre 40 y 100. De esta forma se evitan sensaciones falsas, para estimaciones grandes.

- La carta 0 se utiliza cuando una historia ya está hecha o requiere muy poco tiempo de trabajo.
- La carta ¿? Se emplea cuando se puede llegar a tener una idea de la complejidad de la tarea.
- La carta con la taza de café se emplea para pedir un descanso después de la estimación.

3.2.5.5 Actualización del diagrama de Burndown

Es importante que el diagrama de Burndown esté actualizado para poder tener feedback del avance de los sprint, y poder tomar decisiones de forma rápica.

Para ello planteamos varias opciones que podrían darse en el desarrollo de un sprint:

En este ejemplo 1, se estima un ritmo de trabajo constante con pendiente negativa, y en la realidad, a día 16, se observa que,s egún el ritmo que se desarrolla, se completaría el Sprint antes del día 19.

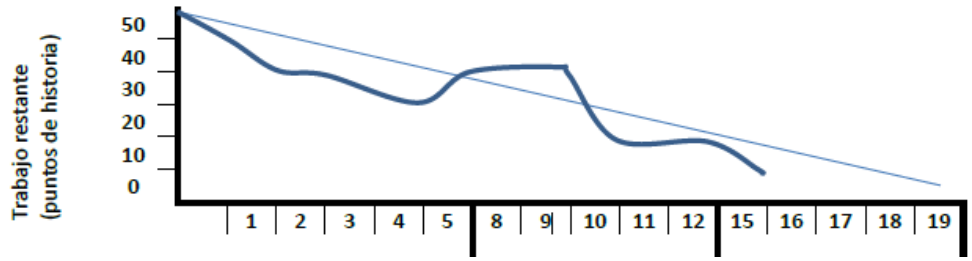


Figura 24. Ejemplo 1 de diagrama de burndown

No se tienen en cuenta los días no laborables para no alterar los diagramas.

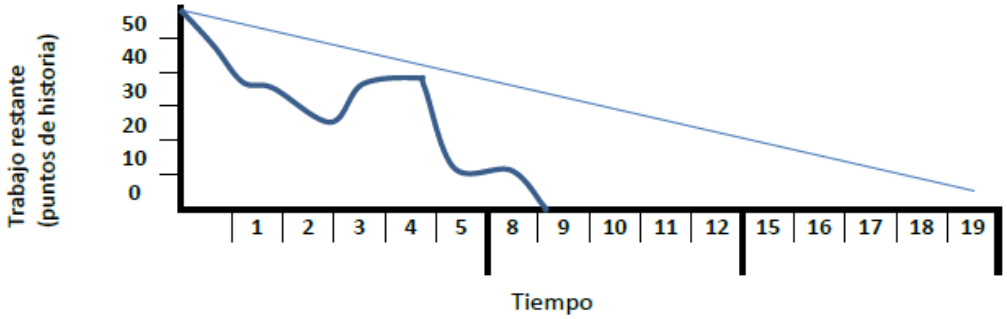


Figura 25. Ejemplo 2 de diagrama de burndown

En este 2º ejemplo, los puntos de historia se han realizado antes de lo previsto, pudiendo realizar más puntos de historia antes de que acabe el sprint, y así adelantar tareas.

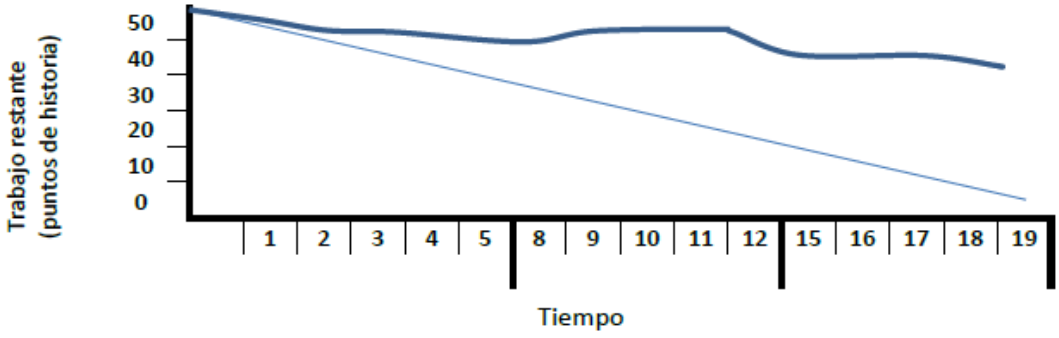


Figura 26. Ejemplo 3 de diagrama de burndown

En este último caso, el diagrama refleja una carga excesiva de tareas a realizar, siendo necesario analizar qué tareas habría que sacar de dicho sprint, de cara a planificarlos para el siguiente.

El intervalo adecuado para los sprints está entre las 2 y las 4 semanas.

3.2.5.6 Reuniones durante el sprint, y post-sprint

Durante el desarrollo de los sprints, y al finalizar el sprint, hay que destacar 2 tipos de reuniones necesarias y que aportan un gran valor al proceso:

3.2.5.6.1 Reunión Diaria (Sprint Daily Meeting)

En esta reunión diaria se busca obtener un estado de situación de cada miembro de cara a mejorar el rendimiento de cada uno.

Teniendo como referencia el Sprint Backlog, y el gráfico de Burndown con la información del día anterior, y con las tareas que tiene asignado cada uno, en esta reunión, que debe durar menos de 15 minutos y que se hace de pie, se contestan a 3 preguntas básicas:

- **¿Qué hiciste ayer?**
- **¿Qué vas a hacer hoy?**
- **¿Qué problemas has tenido para realizar tus tareas?**

Con esta reunión, se busca el compromiso de todo el equipo, evitando ausencias o bajo ritmo de trabajo.

3.2.5.6.2 Reunión de Revisión del Sprint (Sprint Review Meeting)

En esta reunión, el Scrum Master, junto con el equipo, presentan los productos entregables que se hayan realizado de forma que tanto usuarios, como Product Owner, u otros actores involucrados, puedan analizar el producto entregado y los problemas que hayan podido encontrar durante el sprint.

Las características de esta reunión son:

- Duración máxima de 4 horas.
- Se presenta el producto "entregado", de acuerdo con las definiciones que se hicieron como "entregables".
- Si la funcionalidad no está completa no se presenta.
- La funcionalidad se presenta en un entorno lo más parecido al de producción.

En esta reunión se comprueba si se ha entendido lo que necesitaba el usuario, y si se cubren todos los requisitos.

3.2.5.6.3 Reunión de Retrospectiva (Sprint Retrospective Meeting)

En esta reunión, el equipo analizará el sprint que ha terminado y sacará recomendaciones a aplicar en próximos sprints.

Esta reunión está convocada por el Scrum Master, y suele durar unas 3 horas.

Las características de la reunión son:

- Asisten Scrum Master, equipo y el Product Owner.
- Los temas de la reunión serán: ¿Qué ha ido bien en el último Sprint?, ¿Qué podemos mejorar para el siguiente Sprint?
- El Scrum Master tomará notas de las respuestas.
- El equipo comentará las posibles mejoras que se puedan realizar, indicando cuáles van a tener preferencia.
- El Scrum Master tratará de ayudar a que estas mejoras se implementen para el siguiente sprint.

3.2.6 Resumen

La metodología Scrum, se centra en la gestion del proyecto incorporando la gestion de cambios como una tarea más en el día a día y puede aplicarse a todo tipo de proyectos. Busca el trabajo cooperativo de equipos multidisciplinares y de alto rendimiento. Se definen sprints relativamente cortos (2-3 semanas) de duración fija y se determinan una serie de reuniones a mantener a lo largo de todo el proyecto, ya sean antes, durante, o después de los sprints.

Figura 27. Reunión de Scrum en oficinas de Endesa.

Como resultado de cada sprint, se obtienen incrementos y, debido a la realización de un control de proceso contínuo, es adaptable a nuevas necesidades o cambios del negocio, de una forma ágil y rápida, siempre respetando las normas mínimas acordadas antes de comenzar el sprint, como son el compromiso de no interrumpir el proceso de elaboración de cada sprint, o cambiar las historias de usuario dentro del sprint

Vemos por tanto la necesidad de un compromiso necesario entre el product owner, dueño del producto, que toma decisiones y prioriza las historias de usuario, y el scrum master, que es la cabeza visible del equipo de trabajo, desarrollador de cada una de las tareas de usuario que compondrán cada historia, y cada épica.

Scrum no plantea, por tanto, un trabajo que no sea en equipo, que no sea consensuado, y en el que todas las partes del proyecto apunten hacia el mismo lugar en todo momento: la participación del usuario en el desarrollo de los productos en cada sprint, hace que éste se sienta un miembro más del equipo, pasando de frases del estilo: “habéis construido…”, “habéis hecho”, al “hemos construido” o “hemos hecho”.

Ese cambio de percepción tan sutil introduce varios aspectos que, hasta ahora en la gestión de proyectos, no se tenían en cuenta, como son conciencia y colaboración. Mediante el desarrollo ágil, el usuario es consciente de que las cosas que pide, las funcionalidades que reclama tienen un esfuerzo en tiempo y dinero, y que los errores existen.

Además, el hecho de que se pida su participación introduce el factor de la colaboración, entendiendo los éxitos como un logro de todos, y los errores o fallos, como una carga repartida entre todo el equipo.

4 APLICACIÓN AL CASO PRÁCTICO

En esta sección veremos la aplicación de la metodología Agile, con el marco de SCRUM como referencia, aplicado a dos proyectos en la empresa.

4.1 Planificación de las etapas

Para entender la introducción del desarrollo agile, dentro del mundo empresarial, es necesario mostrar las distintas etapas que se van a llevar a cabo dentro de este proyecto denominado ***DIG&IN*** y debemos de conocer la planificación de estas, así como los objetivos comunes de todas las etapas:

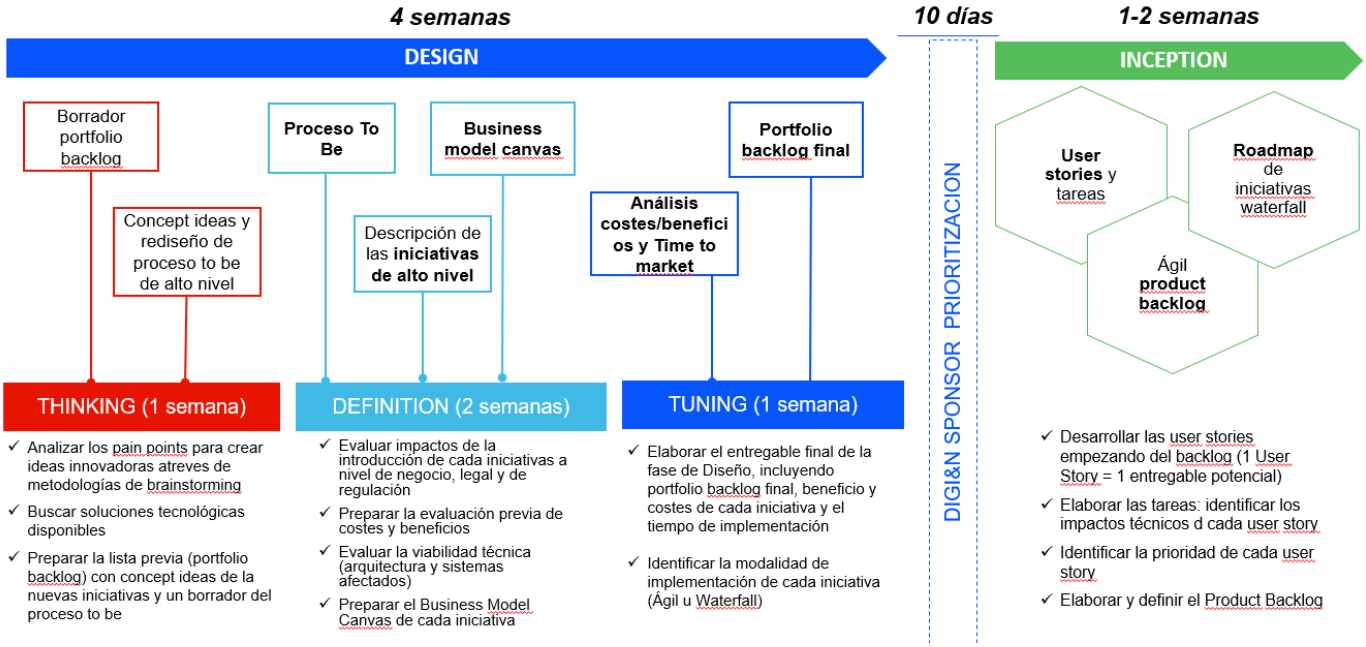


Figura 28. Planificación de las etapas de la aplicación de la Metodología Scrum

Esta planificación, divide el proceso en 4 fases: *Design*, *Sponsor & Prioritization*, *Inception y Execution*.

- **Design:** En esta fase, de una duración de unas 4 semanas, compuesta por los sprints 0, 1, 2 y 3, el objetivo es componer un equipo de trabajo (diferente al de desarrollo), revisar un proceso de negocio en concreto, identificar los *pain points*[19], estableces distintos kpis medibles, plantear posibles mejoras (sin ninguna limitación), evaluar los impactos y necesidades de las distintas iniciativas, analizar los costes y beneficios, preparar un Business Model Canvas sobre cada iniciativa, y definir la mejor metodología para el desarrollo de dichas iniciativas.
- **Sponsor & Priorization:** En esta segunda fase, de una duración de aproximadamente 2 semanas, el objetivo, es presentar las principales iniciativas detectadas como potenciales a un Sponsor[20] que hará las labores de aprobación presupuestaria de la/s iniciativa/s aprobada/s así como su priorización en función de la importancia y la urgencia.
- **Inception:** En esta tercera fase, se definen las historias de usuario, las tareas, se comienza a elaborar y definir el producto Backlog, y se priorizan dichas tareas.

[19] Pain Points: Puntos de dolor o ineficiencias detectadas dentro de un proceso.
[20] Sponsor: Dentro de la empresa, esta figura está represantada por la dirección ejecutiva global.

- **Execution:** En esta cuarta fase, se ponen en marcha los desarrollos, trabajando por sprints, y realizando entregables al final de cada sprint.

Para entender el desarrollo de las etapas, sus actividades, los recursos implicados, y los *outputs* que se esperan como salida de cada etapa se plantean los siguientes esquemas de planificación por etapas:

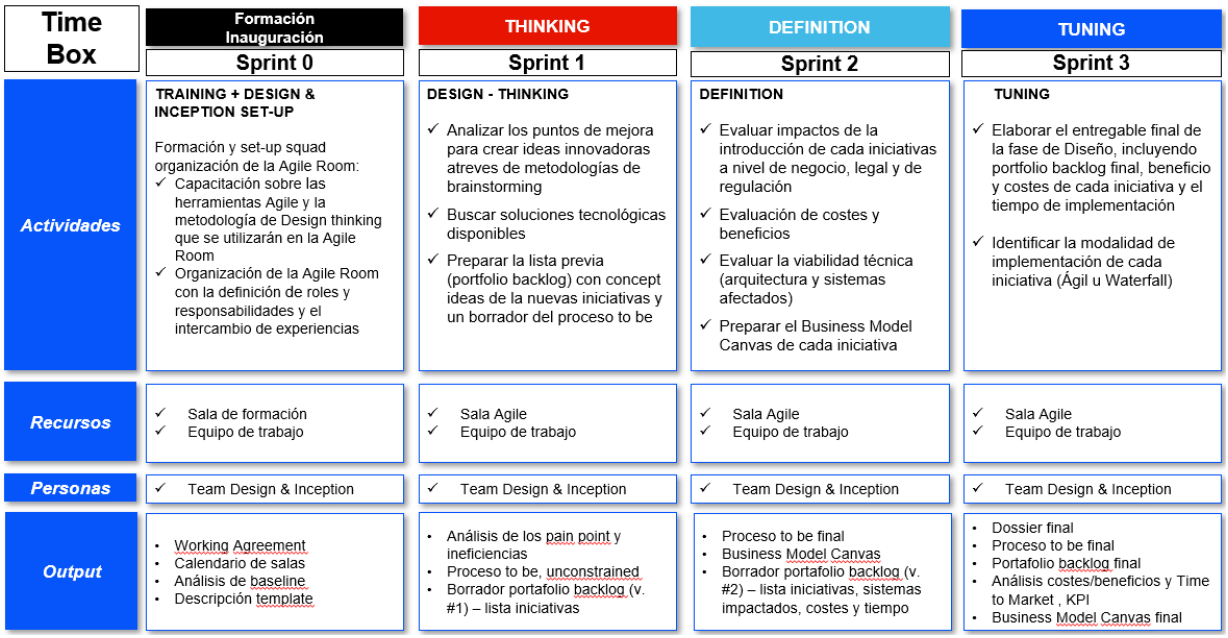


Figura 29. Planificación de los distintos Sprints.

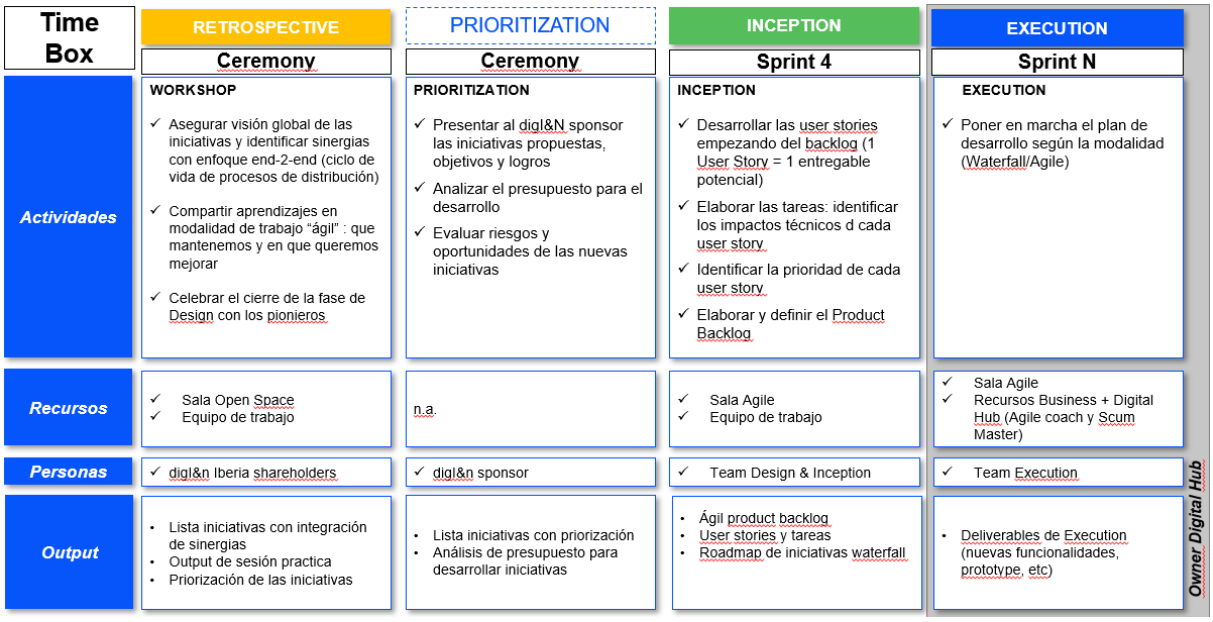


Figura 30. Planificación de los distintos Sprints y ceremonias.

4.2 Dedicación y roles de las personas designadas

Para el desarrollo de las distintas sesiones de esta fase se planteó una organización de personas compuesta por:

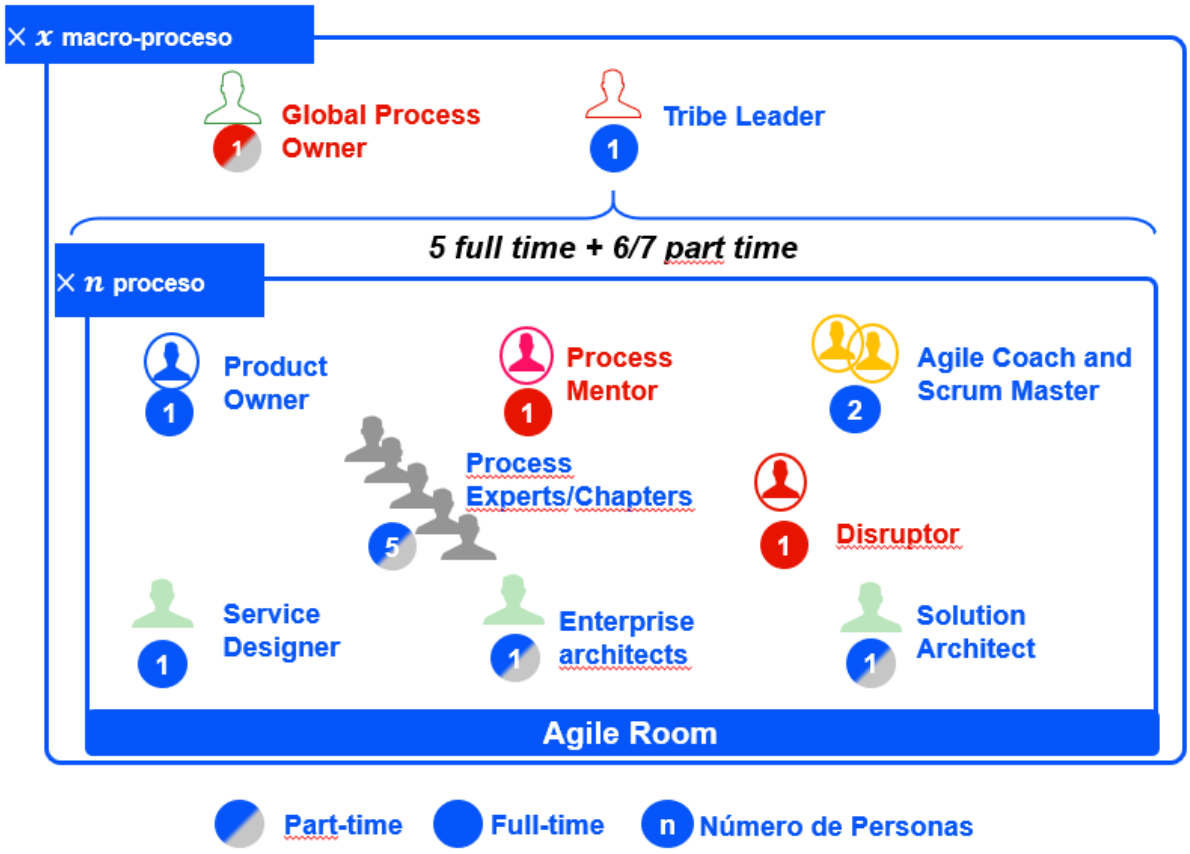


Figura 31. Participantes de las agiles room.

Como se puede ver en el cuadro anterior, existen nuevos roles, además de los ya comentados en la explicación de SCRUM, que se pasan a detallar a continuación:

- **Global Process Owner:** Esta persona representa a la dirección global de los procesos, y cómo se relacionan con otros.
- **Tribe Leader:** Representa la figura de negocio coordinadora de todas las rooms que se estén llevando a cabo. Coordina a los coordinadores de negocio.
- **Product Owner:** Figura que representa la propiedad del proceso en cuestión, siendo una persona de máxima responsabilidad, y con capacidad de decisión.
- **Process Mentor:** Esta figura representa a una persona dueña del proceso que se va a analizar, pero en otro país, donde se ya se ha realizado un análisis similar del proceso.
- **Process Experts:** Equipo de 1 a n personas que forman parte activa del proceso, juntamente con el Product Owner, siendo este último el que las coordina.
- **Disruptor:** Figura totalmente ajena al proceso, que proviene de otra área distinta, con el objetivo de dar un enfoque y una vision ajena al proceso.
- **Agile Coach and Scrum Master:** Figuras que posibilitan el desarrollo de los sprints, explicando como se van a desarrollar las actividades, y resolviendo dudas que puedan surgir sobre la aplicación de las metodologías.

- **Service Designer:** Figura que representa el soporte de los sistemas (TIC[21]) a nivel funcional.
- **Enterprise Architect:** Figura que representa el soporte técnico de los sistemas.
- **Solution Architect:** Figura encargada de analizar las viabilidades aportadas como soluciones a los procesos.

Si bien es cierto que aquí se han definido todos estos roles, no será necesaria la presencia de todos, durante todo el proceso, variando su presencia, en función de la etapa en la que se esté, o solicitada su presencia, en el momento en que se estime oportuno.

De esta forma se planteó el siguiente cuadro de asistencia:

Team Design & Inception	Roles \ Time Box	Formación Inauguración	Baseline	THINKING	DEFINITION		TUNING
		Sprint 0		Sprint 1	Sprint 2		Sprint 3
	Product Owner	Full time	Full time	Full time	Full time	Full time	Full time
	Process Expert	Full time	Full time	Full/Part time	Full/Part time	Full/Part time	Full/Part time
	Service Designer	Full time	Full time	Full time	Full time	Full time	Full time
	Solution Architect	Full time	Full time	Full time	Full time	Full time	Full time
	Agile Coach	Full time	no	Full time	Full time	Full time	Full time
	Disruptor	Full time	no	Full time	Full time	no	no
	Global Process Owner	no	no	Full time	Full time	Full time	no
	Process Mentor	no	no	Full time	Full time	Full time	no
	Scrum Master	Full time	no	Full time	Full time	Full time	no
	Methodology & Planning	Full time	Full time	Full time	Full time	Full time	Full time
	Planning & Control	no	Full time	no	Full time	Full time	Full time
	HR	Full time	Full time	no	Full time	Full time	Full time
	Network Technology	no	no	no	Full time	Full time	Full time
	Regulation/Legal/Safety/Procurement	no	no	no	On demand	On demand	no

Figura 32. Dedicación para cada etapa en función de los roles.

Viendo el cuadro anterior se observa que aparecen otros roles, a las que también se suman su presencia en la sala, como son las siguientes:

- **Methodology & Planning**: Área de Metodología y Planificación.
- **Planning & Control:** Área encargada de la planificación económica, y control económico.
- **Network Technology:** Área encargada de las tecnologías de redes e infraestructuras.
- **HR:** Área de Recursos Humanos
- **Regulation:** Área de Regulación.
- **Legal:** Área de Legal (cualquier aspecto legal).
- **Safety:** Área de Seguridad y prevención.
- **Procurement:** Área de Compras.

Todas estas áreas, además de integrarse en su fase correspondiente (si aplica), están a disposición del product owner en todo momento, en caso de necesitar ayuda para resolver alguna cuestión, o necesidad de cualquier información.

[21] TIC: Tecnologías de la Información y de la Comunicación

4.3 Desarrollo de las distintas etapas.

4.3.1 Training, Design & Inception Set-up.

En esta primera etapa, de una duración aproximada de 3 días, se formaron los grupos de trabajo, se habilitaron las salas preparadas para un entorno más dinámico, y se dio una formación inicial a todos los asistentes para conocer las distintas actividades que se fueran a desarrollar, mediante presentaciones y dinámicas, para reforzar los objetivos que buscan el Desarrollo ágil.

Figura 33. Diapositiva de Formación del proceso Agile, a través de la consultora Siag.

Además, se estableció un consenso entre todos los participantes de la sala en cuanto a la organización de las reuniones (martes, miércoles y jueves en horario de 9:00 a 15:00).

Por útimo, en esta primera etapa se presentó el análisis de la *baseline*[22] sobre el proceso, y los puntos de dolor identificados de forma inicial, como una tarea previa de los product owners con sus respectivos responsables.

4.3.1.1 Sala de Liquidaciones.

El proceso de Liquidaciones está conformado por un equipo de unas 4 personas de negocio, un par de personas que dan soporte desde sistemas, y 4 personas más de Software Factory. La principal actividad de este área consiste en la elaboración de la declaración mensual y/o anual a la Comisión Nacional de Mercados y Competencia (CNMC) de la actividad regulada (consumos de energía y facturación liquidables), asegurando forma y plazo, incluyéndose en dicho proceso, las actividades de autodeclaración de Cuotas/Tasas y Liquidación.

[22] Baseline: Línea base o directrices sobre las que enmarcaran los procesos

Entre las tareas de dicha área, destacan:

- Realización de las declaraciones mensuales según modalidad establecida por regulación (Informes SINCRO) a la Comisión Nacional de Mercados y Competencia (CNMC).
- Declaración y liquidación de las cuotas y tasas a la CNMC, y tasas a la Agencia Estatal Administración Tributaria (AEAT).
- Liquidación de costes de transporte, distribución y comercialización a tarifa, los costes permanentes del sistema, los costes de diversificación y seguridad de abastecimiento.
- Gestión y atención de inspecciones de la CNMC.
- Actualización de los sistemas y tarifas de acuerdo con la reglamentación vigentes.

4.3.1.1.1 Baseline de Liquidaciones.

El modelo presentado como Baseline del proceso es el siguiente:

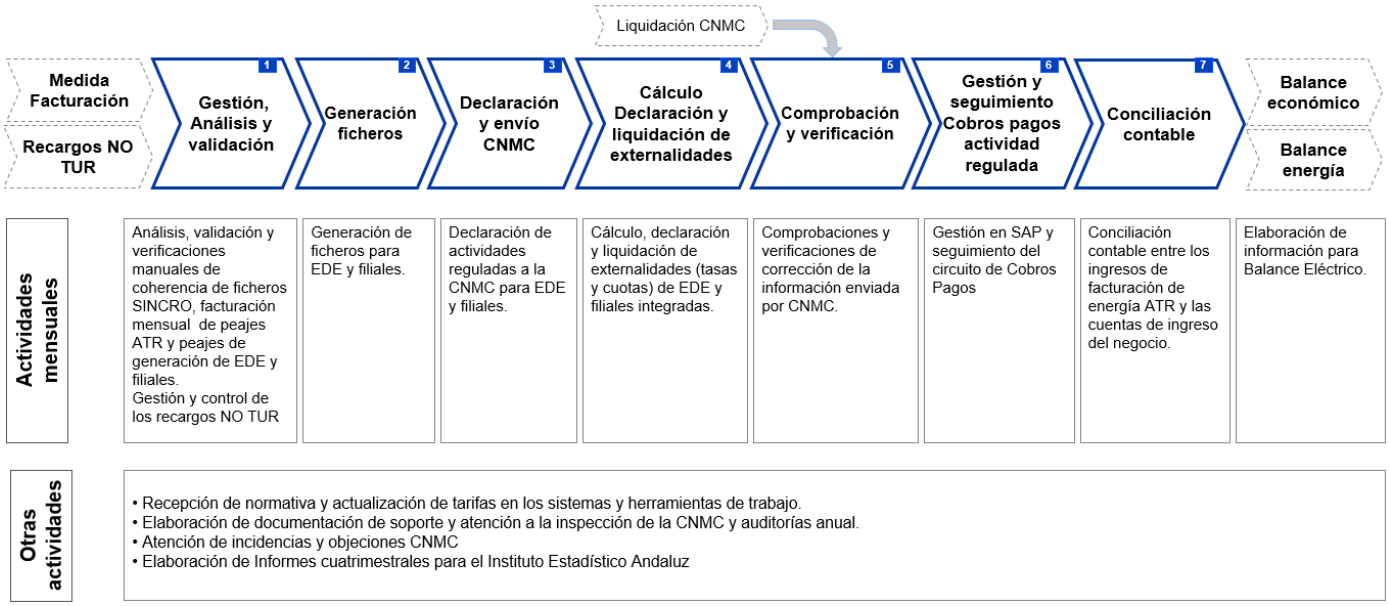


Figura 34. Baseline del proceso de Liquidaciones

El objetivo de presentar la baseline del proceso, es que todas las personas presentes en la sala puedan entender el proceso en su totalidad, de cara a entender los puntos de mejora encontrados, y/o plantear el mayor número de soluciones posibles.

4.3.1.1.2 Paint Points sobre la baseline.

Sobre la Baseline, se detectaron las siguientes carencias, en un análisis previo a alto nivel:

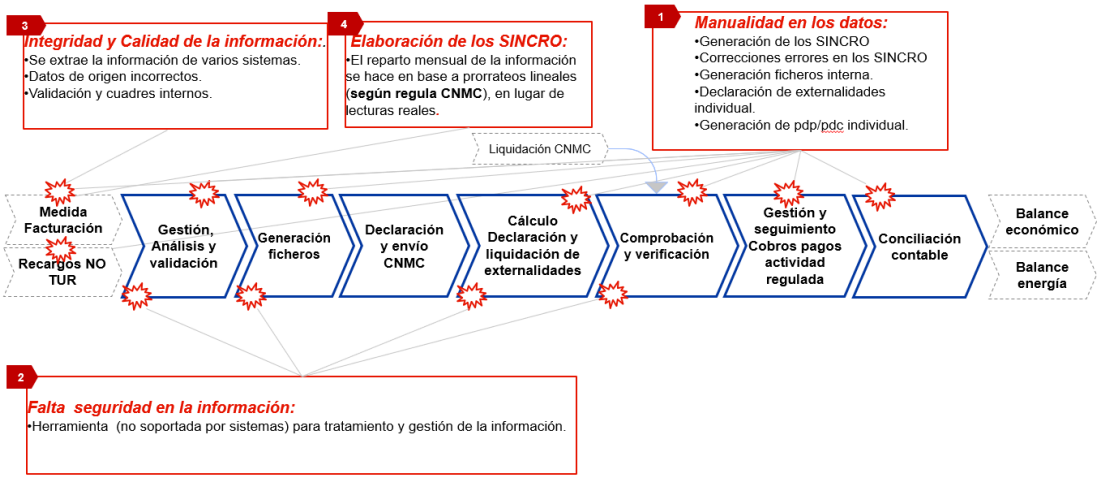
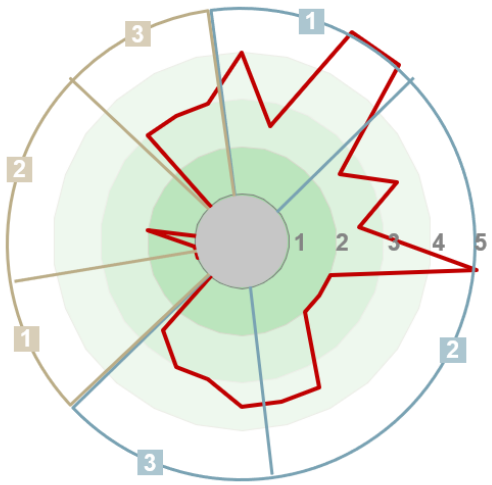


Figura 35. Lista de carencias detectadas en el proceso.

Viendo el análisis previo del product owner, en la elaboración de este *as is*, se puede observar que existe la necesidad de una gestión de los datos de forma automática, e informática, soportada por el área de sistemas informáticos de la compañía, ya que el soporte actual para la presentación de estos informes, está basado en herramientas ofimáticas, que no dejando de ser recursos de la empresa a nivel corporativo, no deja de ser una herramienta de trabajo propia (BBDD Access propia) sin capacidad de procesamiento, ni soporte.Además, la gran variedad de fuentes de información de datos de la que se nutre esta base de datos, llevan a una falta de integridad y calidad, puesto que no deja de ser un proceso manual de carga, de interpretación y de generación de los informes finales a presentar a la CNMC.

4.3.1.1.3 Estado digital del proceso de Liquidaciones:

El estado digital del proceso de liquidaciones, en su *as is* fue:



		Obtencion de datos	Validacion y declaracion
1	Difusión soporte tecnológico	5	3
	Eficacia de la infraestructura hardware	3	2
	Eficacia de la infraestructura telecom.	5	5
	Permite la gestión en remoto	5	5
2	Difusión de las Aplicaciones	4	1
	Nivel de "Automatización"	4	3
	Eficacia del proceso informatizado	3	2
	Operatividad en remoto	5	5
	User-experience	2	2
	Flexibilidad / Adaptabilidad	2	2
	Reports de síntesis automáticos	N/A	2
	Seguridad y privacidad digital	5	2
	Disaster Recovery	5	2
3	Integración Intra-proceso	4	3
	Integración con otros procesos	4	2
	Integración con el exterior	N/A	3
	Utilización de las integraciones existentes	2	3
1	Soluciones y equipos digitales innovativos	N/A	N/A
	Lanzamiento de soluciones digitales innov.	1	1
2	Procesamiento de datos (Big Data)	1	1
	Gener. autom de acciones tras anál. datos	3	1
	Puesta a disposición de info. para exterior	N/A	N/A
	Modalidad de puesta a disposición del dato	N/A	N/A
3	Nº de pasos del proceso operativo	5	1
	Nº de pasos del proceso	5	1
	Tiempo para recorrer el proceso	3	3

Figura 36. Estado Digital del proceso de Liquidaciones

El hecho de no utilizar herramientas de procesamiento masivo de datos (tipo Big Data), ni su tratamiento automático, hacen que el estado digital de este proceso no sea muy alto, siendo necesaria su actualización a un proceso automático, debido a la alta manualidad en la generación de los datos y su sensibilidad (60. MM € de sanción por errores graves en la presentación de los formularios).

4.3.1.2 Sala de Facturación y Cobros.

El proceso de facturación y cobros, compuesto por 8 personas de negocio, 2 personas de sistemas, 6 personas de software factory, y un backoffice de 300 personas, realiza como principales actividades de generación, envío y conservación de la facturación tanto en el caso de ATR[23] (incluido el alquiler de contadores), como de otros servicios: nuevos suministros, conexiones, contratación, peajes de generación, enganches directos, fraudes, y las actividades de cobro y gestión de la deuda, tanto de clientes ATR directos como de comercializadoras. Este equipo está formado por unas 10 personas del área de negocio, que, a su vez, tienen contratada en una empresa de la matriz de Endesa (Cefaco), las principales actividades de la operativa diaria (este equipo lo pueden conformar más de 200 personas).

Entre las tareas de dicha área, destacan:

- Generación de facturas y refacturaciones (De energía, de no energía, altas, bajas, modificaciones, derechos, peajes de generación, fraudes, indemnización, trabajos en campo, calidad invidual del suministro, suplementos territoriales, etc…).

[23] ATR: Acceso de Terceros a la Red.

- Edición de facturas.
- Envío y publicación de dichas facturas.
- Gestión del cobro (Cobro/Dudoso Cobro/Impagos).
- Análisis y evaluación de toma de acciones por vía judicial.
- Reporting y control del proceso.
- Atencion personalizada de comercializadoras.
- Conciliación Contable
- Soporte Fiscal.
- Atención e información de facturación para áreas internas y organismos externos.
- Seguimiento y atención a las modificaciones impulsadas por el regulador (CNMC).

4.3.1.3 Baseline de Facturación y Cobros.

El modelo presentado como Baseline del proceso es:

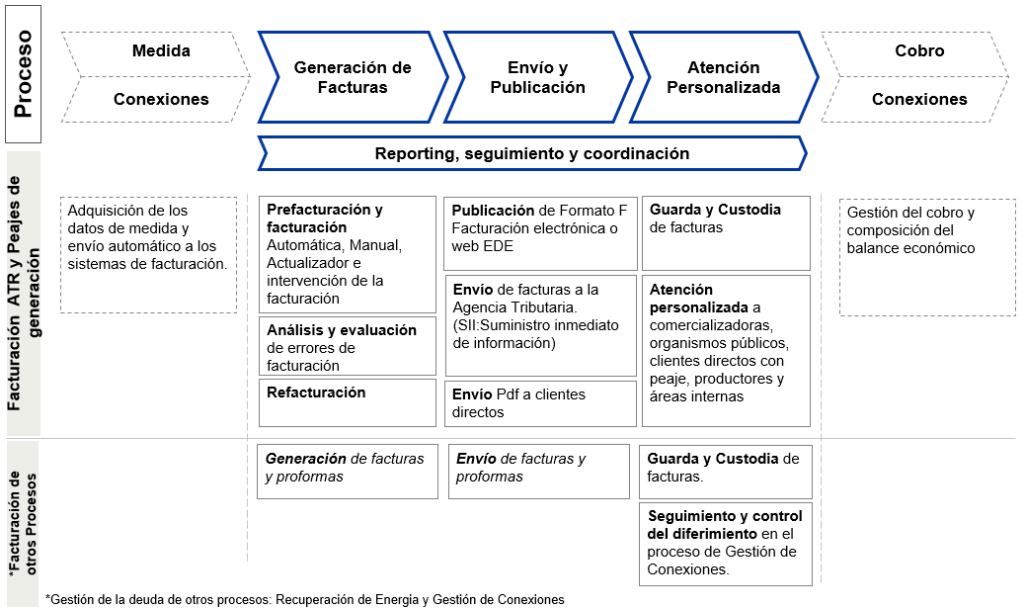


Figura 38. Baseline del Proceso de Facturación

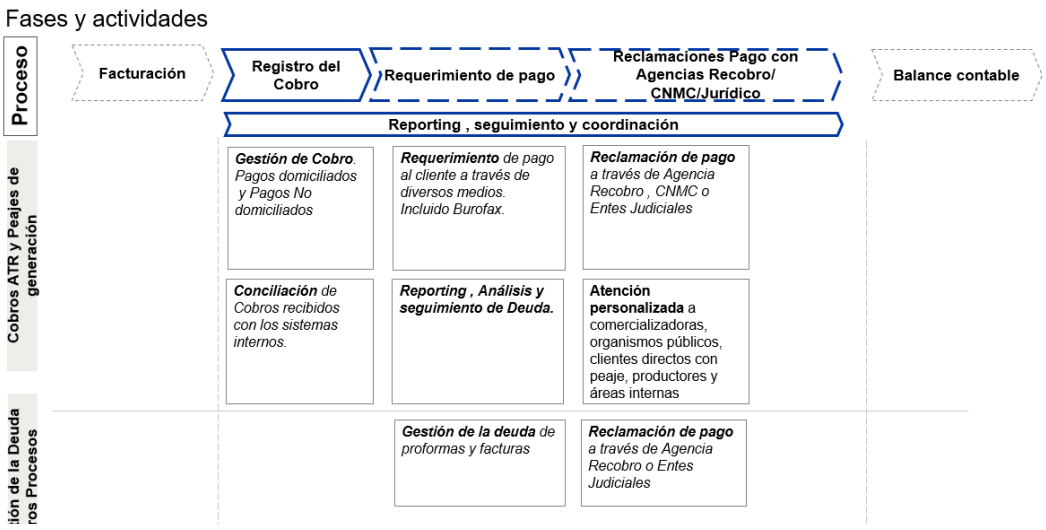


Figura 37. Baseline del proceso de cobros.

4.3.1.3.1 Paint Points sobre la baseline.

Sobre la Baseline, se detectaron las siguientes carencias:

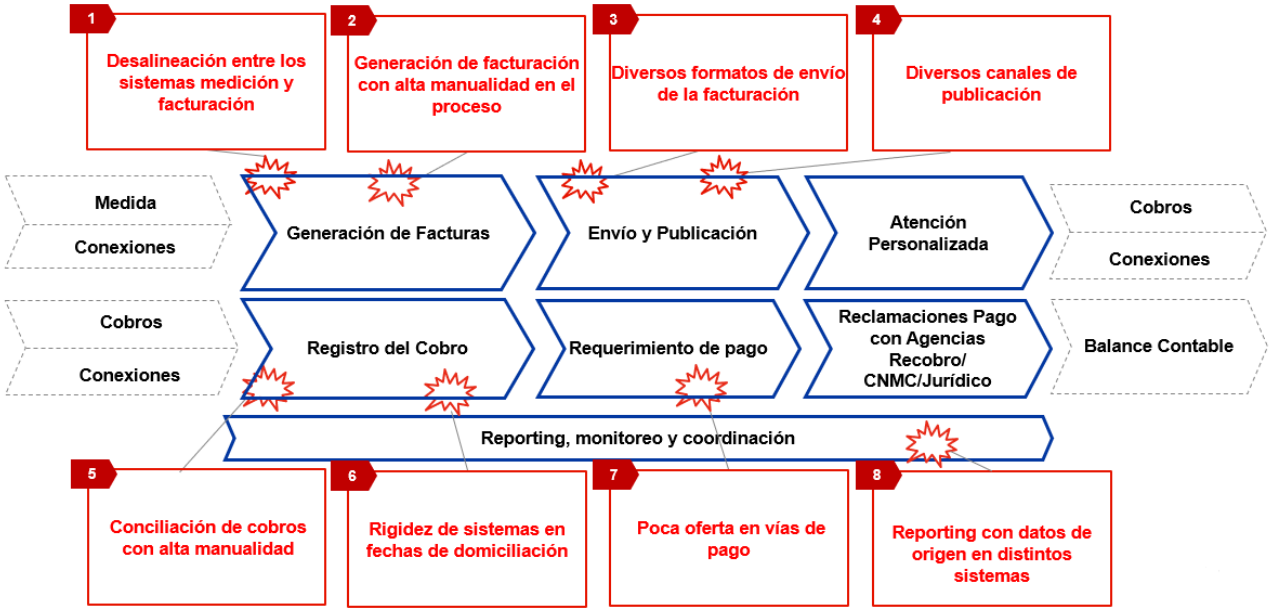
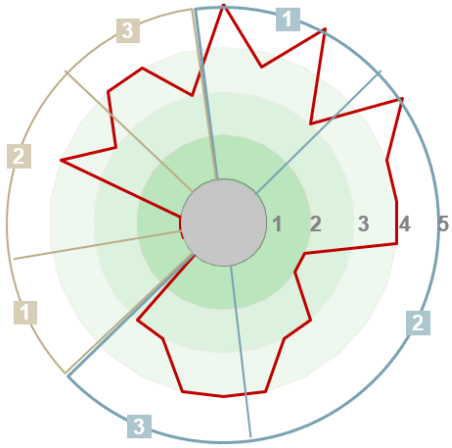


Figura 39. Lista de carencias detectadas en el proceso de Fact. y Cobros.

Viendo la lista de carencias elaborada con la baseline, vemos que se necesita una automatización en algunos pasos de los procesos (en la que los usuarios manifiestan las carencias del uso de un sistema a medida, creado con una tecnología de los años 2000 con una arquitectura MainFrame) así como utilizar nuevas tecnologías existentes en el mercado, como las vías de pago tipo pdp[24], o automatizaciones de procesos actuales, que permitan un tracking completo y automático de los procesos de facturación y del cobro.

4.3.1.3.2 Estado digital del proceso de Liquidaciones:

El estado digital del proceso de liquidaciones, en su *as is* es:



		Fact Transporto	Fact Prestaciones	Cobro
1	Difusión soporte tecnológico	5	5	5
	Eficacia de la infraestructura hardware	3	5	3
	Eficacia de la infraestructura telecom.	5	5	5
	Permite la gestión en remoto	3	3	3
2	Difusión de las Aplicaciones	5	5	5
	Nivel de "Automatización"	4	4	4
	Eficacia del proceso informatizado	4	4	4
	Operatividad en remoto	4	4	4
	User-experience	2	2	2
	Flexibilidad / Adaptabilidad	2	2	2
	Reports de síntesis automáticos	3	3	3
	Seguridad y privacidad digital	3	3	3
	Disaster Recovery	4	4	4
3	Integración Intra-proceso	4	4	4
	Integración con otros procesos	4	4	4
	Integración con el exterior	3	3	3
	Utilización de las integraciones existentes	3	3	3
1	Soluciones y equipos digitales innovativos	N/A	N/A	N/A
	Lanzamiento de soluciones digitales innov.	1	1	1
2	Procesamiento de datos (Big Data)	1	1	1
	Gener. autom de acciones tras anál. datos	1	1	1
	Puesta a disposición de info. para exterior	4	4	4
	Modalidad de puesta a disposición del dato	3	3	3
3	Nº de pasos del proceso operativo	4	4	4
	Nº de pasos del proceso	4	4	4
	Tiempo para recorrer el proceso	3	3	3

Figura 40. Estado Digital del proceso de Facturación y Cobros.

[24] Pdp: Pasarela de Pago.

Si bien este proceso es bastante automático, está basado en un sistema infórmatico que, como decíamos antes, se implantó en el año 2000 y que actualmente se encuentra en un proceso de sustitución mediante un proyecto que pretende sustituirlo por un sistema SAP, apoyado en la nube de S4 HANA que ofrece una mejora sustancial en el rendimiento de muchos procesos.

4.3.2 Design - Thinking

En esta Segundo sprint, de una duración de una semana, los objetivos fueron analizar los *pain-points*, así como encontrar nuevos, para crear nuevas ideas innovadoras a través de brainstorming, buscar soluciones tecnológicas disponibles, y preparar el portfolio backlog con el listado de nuevas iniciativas.

Para ello, se siguieron utilizando las room agile habilitadas, así como las herramientas de SCRUM, como las reuniones del *Daily meeting*, o las *retrospective meeting* diarias. Además, el uso de un Scrum Taskboard permitió ver las tareas pendientes en todo momento:

Figura 41. Explicación del proceso de Facturación en un tablero con post-its

Figura 42. Equipo de trabajo en Room Agile.

Figura 43. Scrum Taskboard

4.3.2.1 Sala de Liquidaciones.

De la fase thinking, se encontraron nuevos *pain-points* asociados al proceso, y un listado de iniciativas de forma *unconstrained*[25] teniendo como entregables de este sprint un listado de *pain-points* con indicadores asociados y medibles, un esquema del proceso *to be* y un primer listado del *portfolio backlog* con las iniciativas listadas y descritas:

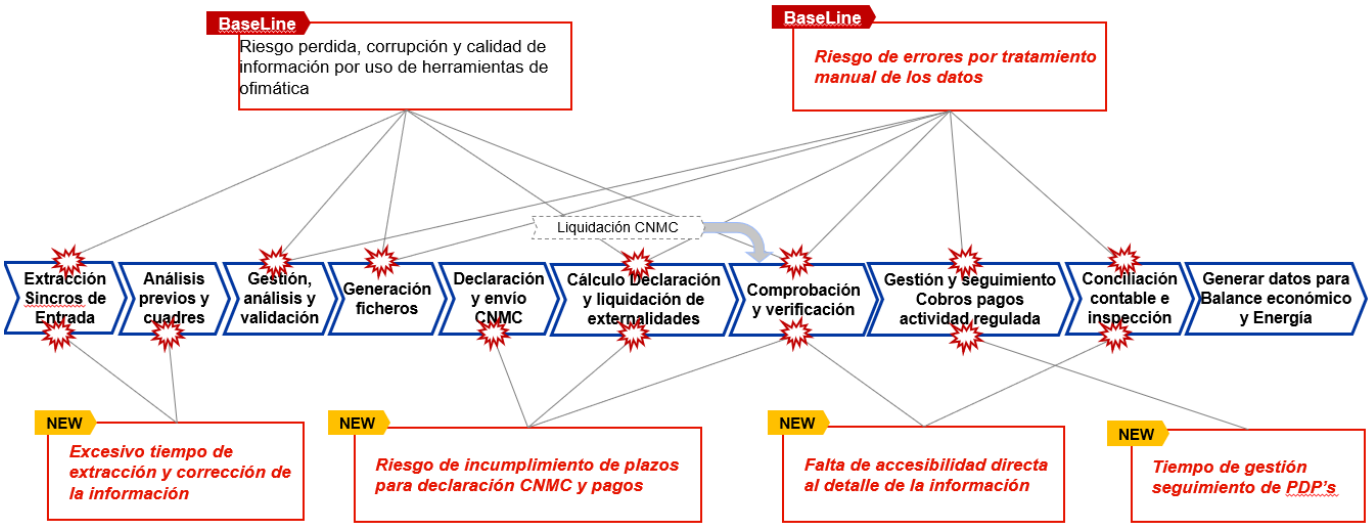


Figura 45. Proceso de Liquidaciones con nuevos pain points detectados.

Pain Point	Ineficiencia	Indicador(es)	Valor (año 2017)
1 Riesgo de perdida, falta de integridad y calidad de información por uso de herramientas de ofimática	Riesgos provocados por el uso de herramientas y sistemas no corporativos y no unificados para el tratamiento y gestión de la información.	Número de acciones a realizar (una vez que se dispone de la versión definitiva de los ficheros de entrada) en un año.	5.347 acciones
2 Riesgo de errores por tratamiento manual de los datos	Riesgos provocados por la baja automatización del proceso.	Número de emails con errores intercambiados con los contratistas al año.	2.045 mails
3 Excesivo tiempo de extracción para la declaración de la CNMC	A la hora de extraer y revisar la información para la declaración no se detectan muchos de los errores en origen ni se corrigen todos los errores reportados a la vez, sino uno a uno. La preparación y elaboración de los ficheros finales conlleva numerosas revisiones, correcciones, y tiempo para corregir los descuadres.	Tiempo desde comienzo del mes hasta que se entregan los ficheros a la CNMC	23,3 días
5 Falta de accesibilidad directa al detalle de la información	No tener a disposición de manera directa el detalle de toda la facturación provoca riesgos para el cumplimento de los plazos y riesgos para envío a la CNMC de información correcta	a) Número de facturas anuales de las que no se dispone de forma directa del detalle. b) Porcentaje de advertencias de CNMC analizados manualmente en un año.	a) 145.397.774 b) 100%
6 Tiempo de gestión y seguimiento de PDP's	Carga manual de PDPs y PDCs en el sistema que afecta a la gestión y seguimiento de cobros/pagos de la actividad regulada.	Número de PDPs+PDCs anuales	993 pdp*

*:estimación datos mensuales en base a junio 2018

Figura 44. Pain Points con los KPI´S asociados para el año 2017.

Podemos comprobar como los propios entregables son dinámicos, como por ejemplo, en cuanto a la definición de los *pain points*, agrupándose o redefiniéndose, e incluyendo además las nuevas carencias detectadas por el grupo de trabajo durante las sesiones. Para esta fase, la recopilación de las carencias sí puede ser una tarea conjunta de todos los miembros, pero la de aportar los valores para el año 2017, es tarea del product owner, apoyado en áreas transversales de negocio.

[25] Unconstrained: sin limitaciones
[26] PDP/PDC: Petición de Pago /Petición de Cobro (Entidad SAP)

Uno de los entregables de esta fase, ya comentado anteriormente, es el proceso *to be*, que describe cómo sería el proceso en un escenario futuro, con todas las iniciativas planteadas implantadas:

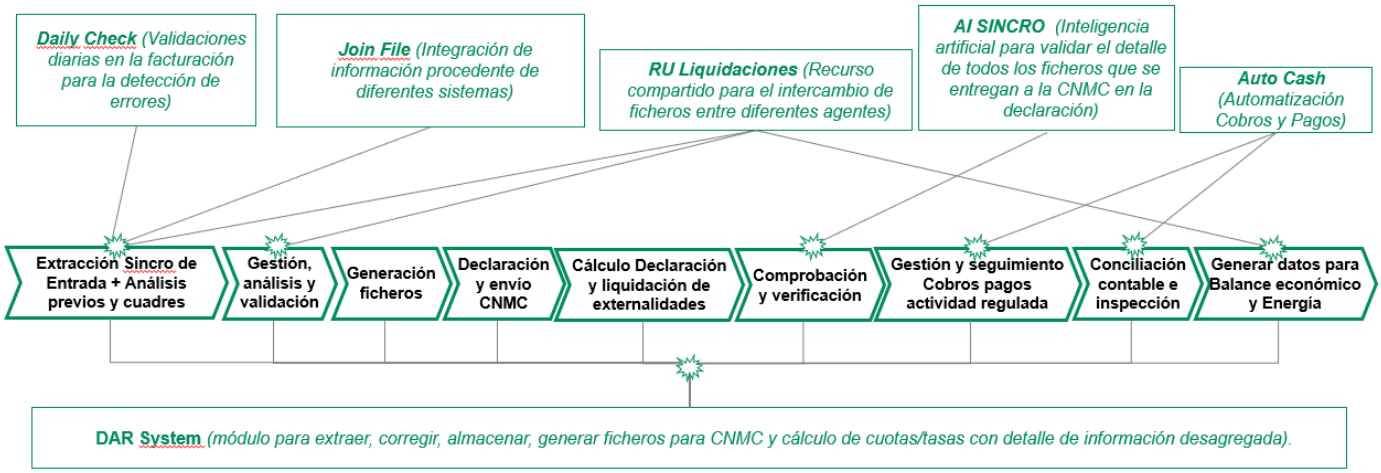


Figura 46. Proceso *To be* del proceso de Liquidaciones

Por último, el último entregable del sprint, consistió en el *product backlog* como recopilación de las ideas recogidas y sus objetivos:

	Iniciativa	Ilustración	Quick win	Descripción	Sistemas impactados	Coste	Desarrollo
Grupo de Iniciativas	Daily Check			Incluir en la extracción de los ficheros SINCRO las validaciones para detección de errores, que automáticamente identifique la/s factura/s afectada/s, con validación diaria y que se retroalimente con los errores detectados y resueltos	AIRE, SCE, BD AUTOCONSUMO, BILLING (a futuro)		
	DAR System			Creación de un sistema corporativo para la Declaración de Actividad Regulada, incluyendo la automatización de: la extracción de información, el análisis preliminar, el almacenamiento de la información por claves agregadas y factura a factura, la elaboración de los ficheros para la declaración, el cálculo de las cuotas y tasas, la declaración de externalidades y ficheros, las comprobaciones de formularios emitidos por la CNMC, la conciliación contable y la generación de ficheros para balances económicos y de energía.	AIRE, SCE, BD AUTOCONSUMO, EXABEAT, Salesforce, GIGA, , BILLING (a futuro)		
Grupos de Iniciativas	Auto Cash			Automatizar la grabación en E4E de cobros y pagos de las liquidaciones de la actividad regulada mediante la carga de una plantilla que incorpore automáticamente los datos de las PDPs/PDCs, la documentación necesaria y que reporte automáticamente información de las firmas del workflow a las personas interesadas.	E4E		
	Join Files			Integración de los peajes de Generación y Vertidos de autoconsumo en los datos de origen y reducción de ficheros. Requerimiento en marcha pendiente de ampliación de alcance.	AIRE, SCE, BD AUTOCONSUMO, BILLING (a futuro)		2018
	AI SINCRO			Utilizar A.I. (Inteligencia Artificial) para analizar automáticamente los requerimientos de detalles (advertencias) de la CNMC. Relacionado directamente con la puesta en marcha de la iniciativa DAR SYSTEM.	WEB CNMC, BBDD LOCALES, AIRE, SCE, BD AUTOCONSUMO, EXABEAT, Salesforce, GIGA, BILLING (a futuro)		
	RU Liquidaciones			Repositorio único, con control de cambios y reporte automático de los mismos a los interesados, para intercambio de ficheros de extracción y datos para Balances.	Outlook, One Drive, Share Point		

Figura 47. Listado del Product Backlog con la descripción de las iniciativas

4.3.2.2 Sala de Facturación y Cobros.

De forma análoga a la sala de liquidaciones, se analizó el proceso detectando nuevos *pain-points*, así como se definieron unos *kpi´s* asociados, y se propusieron mejoras catalogadas en el proceso *to be* ¸y en el *product-backlog*:

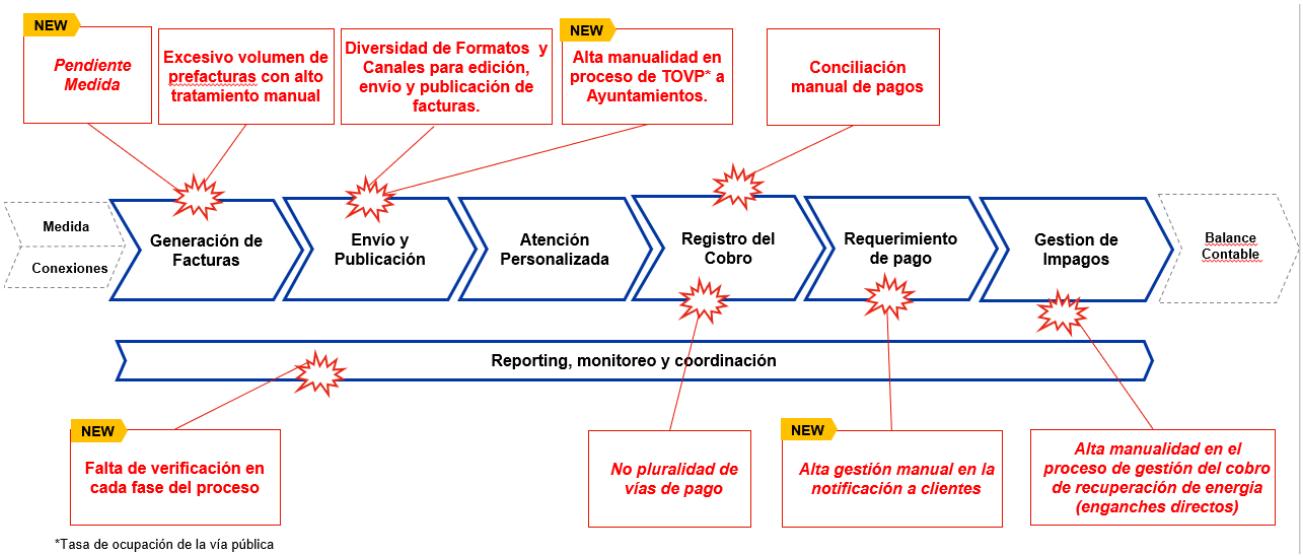


Figura 49.Proceso de Facturación y Cobros con nuevos pain points detectados.

Pain Point	Ineficiencia	Indicador(es)	Valor (valores año2017)
1.-Falta de verificación en cada fase del proceso (Medidas Pendientes/ facturas emitidas vs publicadas Salesforce).	No recepción de la medida para iniciar el proceso de facturación. Verificación de las facturas generadas frente a las publicadas, etc. Cuadro de mando incompleto.	Porcentaje de suministros monitorizadas en el proceso end to end	0%
2-Excesivo volumen de prefacturas con alto tratamiento manual.	Parametrización en el sistema obsoleta que generan exceso de prefacturas manuales. Alta gestión manual en la gestión de resolución de prefacturas.	Porcentaje de prefacturas resueltas automaticamente	18%
3-Diversidad de Formatos y Canales para edición, envío y publicación de facturas.	Existencia de 6 tipos de formatos: XML-F1(sin firma), PDF1, PDF2,WORD, XML-FacturaE, PAPELy Existen 5 canales de envío/publcación: SALESFORCE, INTERFACE, PORTALWEBEDE, PLATAFORMAS de las Administraciones Publicas, Centro de Facturación y Cobros-EDICION.	Porcentaje de facturas pendientes de migrar a formato único	Pendiente dato
4. Alta manualidad en el proceso de pago TOVP (Tasa Ocupación de Vía Publica) a los Ayuntamientos.	Falta de automatización en el proceso de pago a las Entidades Locales.	% pagos automatizados por transferencias de TOVP (Tasa de Ocupación de Vía Pública)	0%
5.Dificultad de Conciliación de los pagos con las actuales vías actuales (Domiciliación, transferencia bancaria y ventanilla).	Alta manualidad de los pagos para la conciliación del cobro con la información del banco.	%conciliaciones de cobro automáticos	38%
6. Alta gestión manual en la notificación a clientes de requerimiento del pago	Recopilación de información y notificación a clientes de requerimiento del pago de manualmente.	%notificaciones enviadas automáticamente	40%
7. Alta manualidad en el proceso de Gestion del cobro de recuperación de energia - enganches directos.	Alta manualidad en la elaboración y gestión de envíos de ficheros con Agencia de recobros (Recuperación de energía)	% de envios automáticos a la Agencia de Recobro.	0%

Figura 48 Pain Points con los KPI´S asociados para el año 2017

En este proceso, también se detectaron 4 nuevas iniciativas que no se encontraron durante la revisión del proceso en la baseline: iniciativas basadas en la gran experiencia de los usuarios en el uso de los procesos, y sobre sus conocimientos del negocio.

Como resultado del proceso de análisis, se definió un nuevo proceso *to be* así como un listado de iniciativas con su correspondiente detalle a alto nivel.

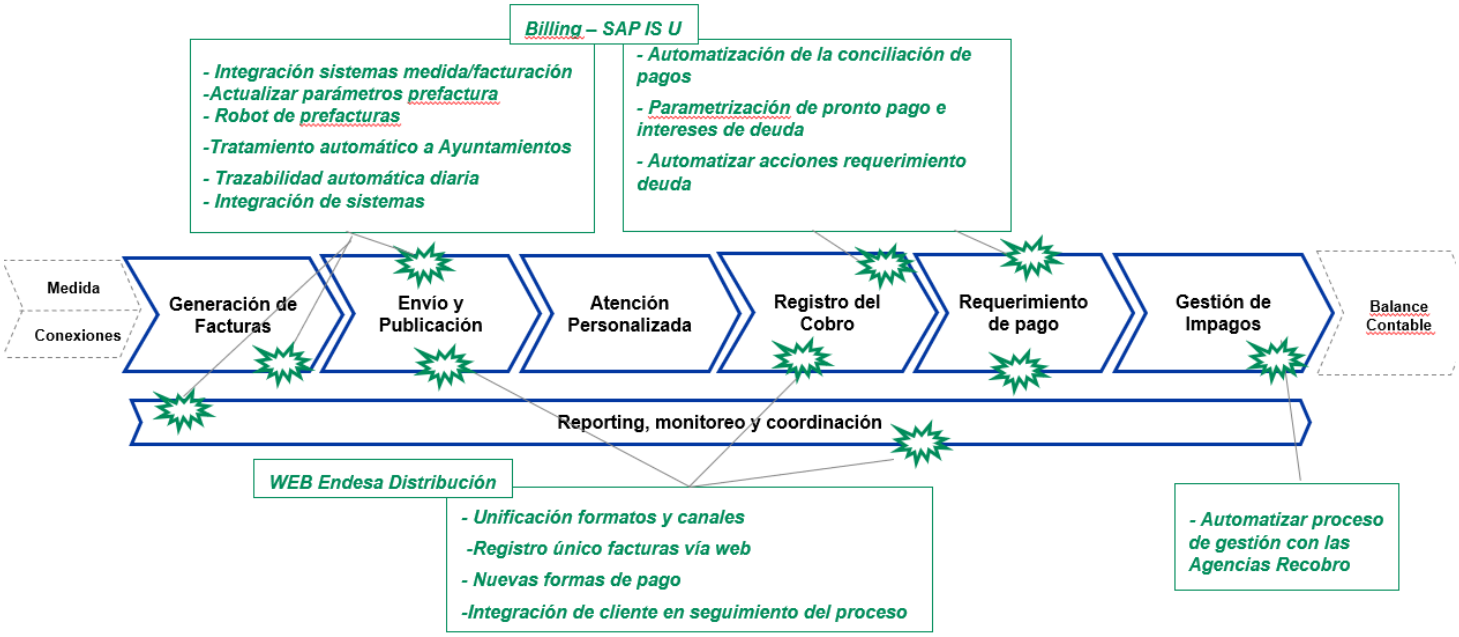


Figura 51. Proceso *to be* con las iniciativas agrupadas en varias soluciones.

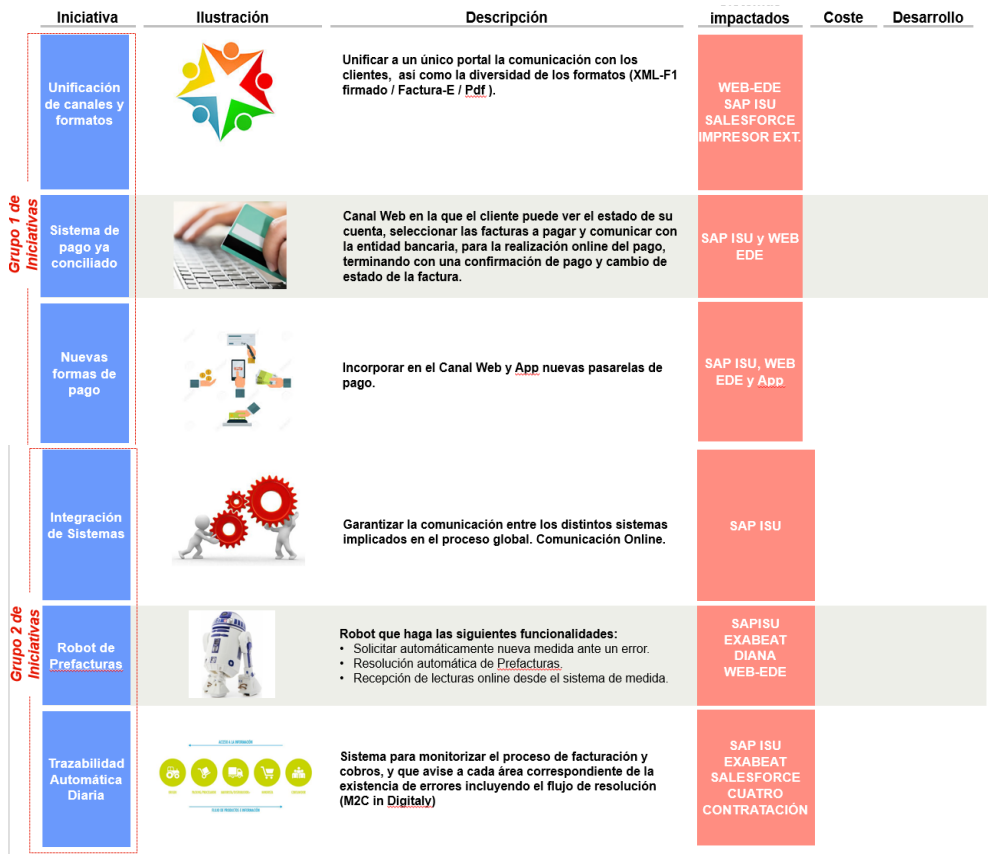


Figura 50. Listado de iniciativas del *Product Backlog*

4.3.3 Fase de Definition

En esta tercera etapa, de una duración de una semana aproximadamente, se volvió a realizar el análisis de las iniciativas, tratando de agruparlas si tenían objetivos comunes y/o por su viabilidad técnica, además de establecer los diferentes indicadores para los *pain points,* y realizar al menos un model business canvas para una de las iniciativas del *product-backlog*, que explicase en qué consistía la iniciativa de una forma más desarrollada.Uno de los principales de esta fase, es pasar de una visión *unconstrained* donde cualquier mejora es viable más allá del punto de vista técnico y económico, a una visión más realista, agrupando iniciativas con objetivos comunes.

4.3.3.1 Sala de Liquidaciones.

En este periodo se refinaron las presentaciones a presentar, así como se agruparon las distintas iniciativas o se renombraron, presentando aquellas diapositivas que presentaron cambios:

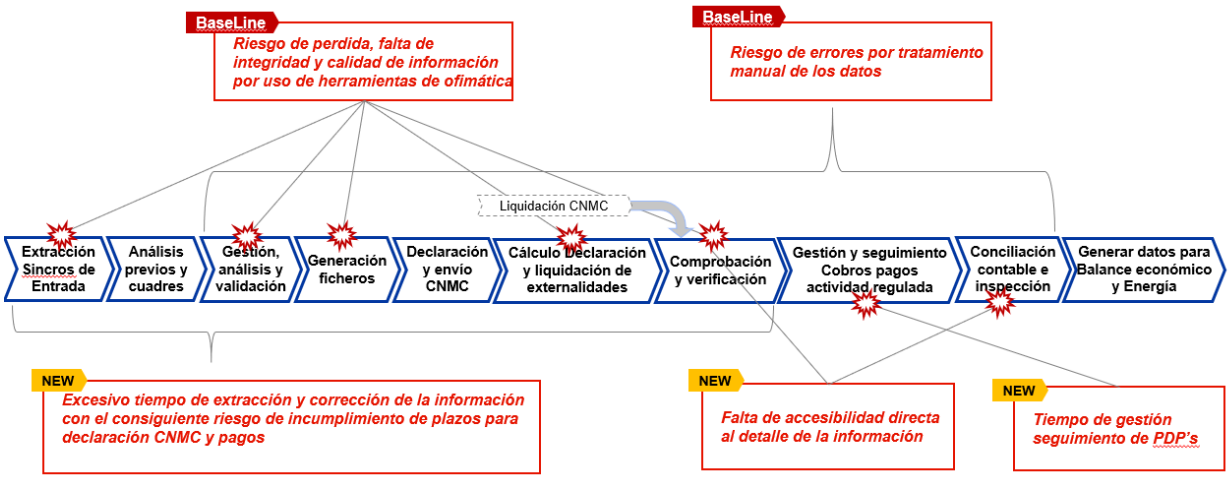


Figura 52. *Pain Points* del proceso de liquidaciones reagrupados.

Pain Point	Ineficiencia	Indicador(es)	Valor (año 2017)
1 Riesgo de perdida, falta de integridad y calidad de información por uso de herramientas de ofimática	Riesgos provocados por el uso de herramientas y sistemas no corporativos y no unificados para el tratamiento y gestión de la información.	Número de acciones a realizar (una vez que se dispone de la versión definitiva de los ficheros de entrada) en un año.	5.347 acciones
2 Riesgo de errores por tratamiento manual de los datos	Riesgos provocados por la baja automatización del proceso.	Número de emails con errores intercambiados con los contratistas al año.	2.045 mails
3 Excesivo tiempo de extracción y corrección de la información con el consiguiente riesgo de incumplimiento de los plazos para la declaración de la CNMC	A la hora de extraer y revisar la información para la declaración no se detectan muchos de los errores en origen ni se corrigen todos los errores reportados a la vez, sino uno a uno. La preparación y elaboración de los ficheros finales conlleva numerosas revisiones, correcciones, y tiempo para corregir los descuadres.	Tiempo desde comienzo del mes hasta que se entregan los ficheros a la CNMC	23,3 días
4 Falta de accesibilidad directa al detalle de la información	No tener a disposición de manera directa el detalle de toda la facturación provoca riesgos para el cumplimento de los plazos y riesgos para envío a la CNMC de información correcta	a) Número de facturas anuales de las que no se dispone de forma directa del detalle. b) Porcentaje de advertencias de CNMC analizados manualmente en un año.	a) 145.397.774 b) 100%
5 Tiempo de gestión y seguimiento de PDP's	Carga manual de PDPs y PDCs en el sistema que afecta a la gestión y seguimiento de cobros/pagos de la actividad regulada.	Número de PDPs+PDCs anuales	993 pdp*

*:estimación datos mensuales en base a junio 2018

Figura 53. Nuevos *kpi´s* definidos y valorados.

Iniciativa DAR System

3 Stakeholders implicados

- CNMC
- ICT
- Proveedores ICT
- P&C
- Áreas EDE (D.A.R., Medida, Facturación y Cobro, Balance de energía y Contratación)
- Otras áreas ENDESA CEFACO, Regulación y Filiales)
- Proveedores D.A.R

1. Problema

Actualmente el proceso gestiona 6.000M€/año para la liquidación de ingresos con el regulador con una herramienta ofimática, Lo cual genera los siguientes problemas:

- Riesgo pérdida y calidad de información
- Riesgo de errores por manualidades
- Excesivo tiempo ->riesgo de incumplimiento de los plazos para la declaración de la CNMC
- Falta de accesibilidad directa al detalle de la información

5. Impacto en el negocio

- Disminución del riesgo de posibles sanciones (hasta 60 M€), gracias a:
  1. Reducir los riesgos del proceso
  2. Garantizar la correcta liquidación de la retribución de EDE y Filiales
  3. Asegurar el cumplimiento normativo (prevención riesgos legales)
- Reducción Opex por ahorro de contratista externos
- Aumento grado de digitalización
- Menos información sensible fuera del negocio

4. Recursos necesarios

- ICT y Proveedores ICT
- Infraestructura software
- Cloud
- Desarrollo aplicación
- Equipo DAR
- Repositorio
- Servicio mantenimiento
- Formacion

2. Solución

Sistema corporativo para la declaración de actividad regulada, incluyendo la automatización de:

- Extracción de información
- Análisis preliminar y cuadres
- Almacenamiento de datos por claves agregadas y factura a factura
- Elaboración de los ficheros y formularios para la declaración
- Cálculo y declaración de externalidades
- Comprobaciones de formularios emitidos por la CNMC
- Conciliación contable
- Generación de otros informes

6. Implicaciones legales / regulación

- Especificaciones CNMC
- Riesgo sanciones por retrasos e incorrecciones
- Riesgo incumplimientos protección de datos (RGPD)

7. Viabilidad técnica

- Desarrollo sistema DAR SYSTEM
- Sinergias uso de tecnologías ya utilizadas

8. Costes y beneficios

COSTES:

- Coste desarrollo
- Licencia software y mantenimiento
- Coste de servidores
- Formación

BENEFICIOS:

- Reducción Opex por ahorro de contratista externos
- Sinergias con otros sistemas. Maximizar uso de tecnología existente
- Reducción riesgo de sanción
- Asegurar calidad de los datos
- Mayor capacidad de análisis
- Uso herramienta unificada y soportada por sistemas

Figura 54. Business Model Canvas para la iniciativa del DAR System

En estos canvas, se presenta de manera pormenorizada, el desarrollo de la iniciativa desde el punto de vista del desarrollo de un negocio. De esta forma, se presenta la iniciativa como parte de un proceso que surge de un problema, al que se le dota de una solución, y se analizan los principales aspectos relacionados, ya sean personas relacionadas, viabilidades de la iniciativa, o su impacto en el negocio, así como los costes y beneficios, de forma que una persona totalmente ajena al proceso, pueda, simplemente leyendo dicho Canvas, saber qué se plantea como solución a un problema, qué costes tiene, y que beneficios implica.

4.3.3.2 Sala de Facturación y cobros.

Para este proceso, el resultado obtenido fue similar, presentando algunas diferencias entre las iniciativas encontradas inicialmente, y las que posteriormente se fueron definiendo como se puede ver en los siguientes esquemas:

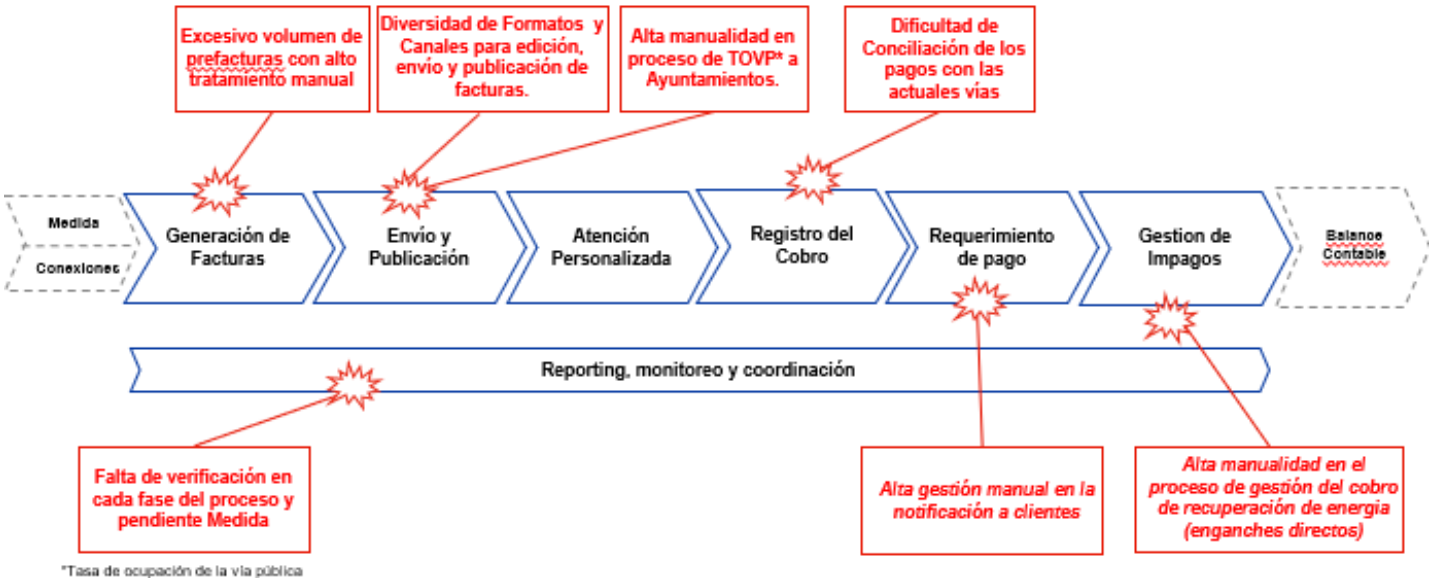


Figura 55. Proceso de Facturación y cobros con *pain-points* agrupados

Pain Point	Ineficiencia	Indicador(es)	Valor (valores año2017)
1.-Falta de verificación en cada fase del proceso (Medidas Pendientes/ facturas emitidas vs publicadas Salesforce).	No recepción de la medida para iniciar el proceso de facturación. Verificación de las facturas generadas frente a las publicadas, etc. Cuadro de mando incompleto.	Nº suministros monitorizadas en el proceso end to end	0
2-Excesivo volumen de prefacturas con alto tratamiento manual.	Parametrización en el sistema obsoleta que generan exceso de prefacturas manuales. Alta gestión manual en la gestión de resolución de prefacturas.	Porcentaje de prefacturas resueltas automaticamente	18%
3-Diversidad de Formatos y Canales para edición, envío y publicación de facturas.	Existencia de 6 tipos de formatos: XML-F1(sin firma), PDF1, PDF2,WORD, XML-FacturaE, PAPELy Existen 5 canales de envío/publcación: SALESFORCE, INTERFACE, PORTALWEBEDE, PLATAFORMAS de las Administraciones Publicas, Centro de Facturación y Cobros-EDICION.	Porcentaje de facturas pendientes de migrar a formato único	Pendiente dato (próximo lunes)
4. Alta manualidad en el proceso de TOVP (Tasa Ocupación de Vía Publica) a los Ayuntamientos.	Falta de automatización en el proceso de elaboración y envío de información a las Entidades Locales.	% pagos automatizados por transferencias de TOVP (Tasa de Ocupación de Vía Pública)	0%
5.Dificultad de Conciliación de los pagos con las actuales vías actulaes (Domiciliación, transferencia bancaria y ventanilla).	Alta manualidad de los pagos para la conciliación del cobro con la información del banco.	%conciliaciones de cobro automaticos	38%
6. Alta gestión manual en la notificación a clientes de requerimiento del pago	Recopilación de información y notificación a clientes de requerimiento del pago de manualmente.	%notificaciones enviadas automaticamente	40%
7. Alta manualidad en el proceso de Gestion del cobro de recuperación de energia - enganches directos.	Alta manualidad en la elaboración y gestión de envíos de ficheros con Agencia de recobros (Recuperación de energía)	% de envios automáticos a la Agencia de Recobro.	0%

Figura 56. Tabla con valores de los *kpi´s* redefinida.

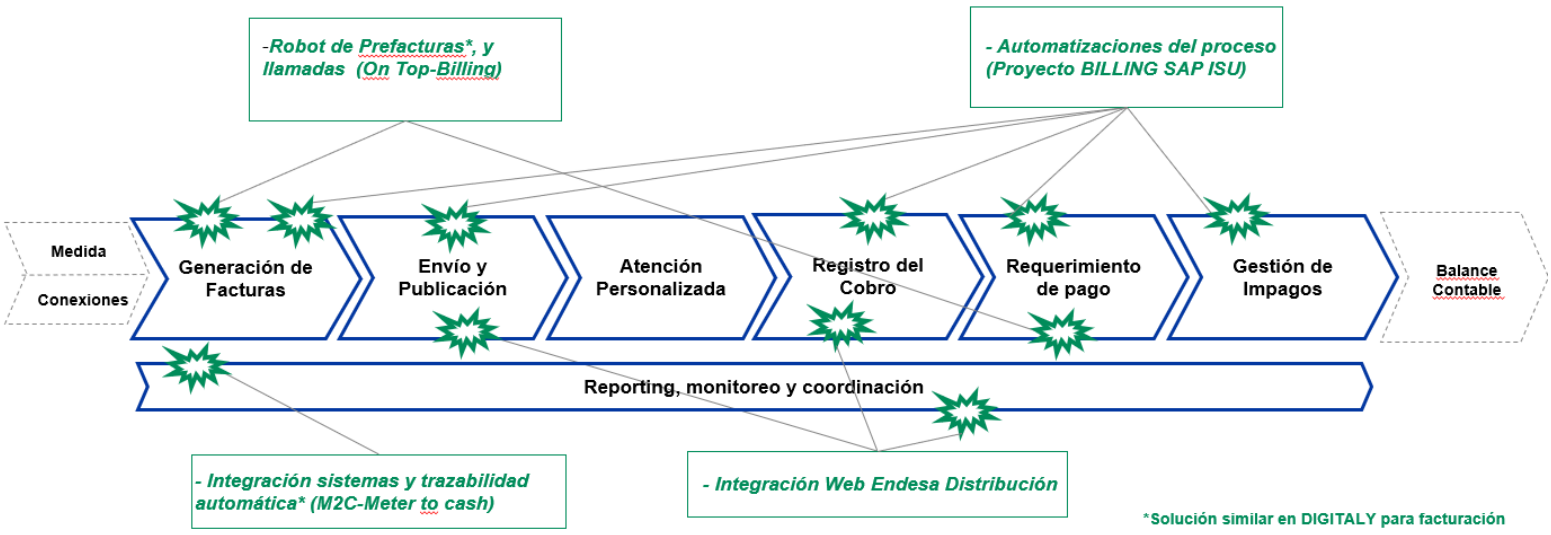


Figura 57. Business Model Canvas de la Iniciativa Robot de Prefacturas.

Con esta fase, se consigue una mayor definición de los procesos, al plantear cuestiones sobre los mismos, llevando a cabo de forma necesaria, una reflexión sobre los principales problemas, y las principales soluciones necesarias y prioritarias para un buen funcionamiento de los sistemas, y de la explotación de la gestión diaria (cabe mencionar, que este sistema en concreto (el SCE-MR[27]) es utilizado por varios cientos de usuarios, y que una mejora en el funcionamiento del mismo, es una mejora en el cash Flow operativo de la empresa, y una reducción de costes de capex, asociada al ahorro en personal externo subcontratado para operar el sistema)

[27] SCE-MR : Sistema Comercial de Endesa- Mercado Regulado.

Robot de pre-facturas

3. Stakeholders implicados • Áreas EDE • ICT • Proveedores ICT • EOSC	1. Problema • Alto volumen de pre- facturas • Alta Manualidad • Retraso en la facturación • Petición manual medida	5. Impacto en negocio (4,5/5) • Adelanto de la facturación • Automatización de tareas
		6. Implicaciones legales/ regulación • No afectado por regulación • Pendiente legal
4. Recursos necesarios • Desarrollo aplicación • Integrar con sistemas • Expertos de EDE • ICT y proveedores ICT • Formación • Expertos de EOSC • Mantenimiento.	2. Solución • Automatización en la resolución de las pre-facturas, con generación de informes sobre los resultados. • Petición automática de medidas. • Aprendizaje sobre los parámetros de pre-factura.	7. Viabilidad técnica • Blue Prims. (plataforma de algoritmo de vía software) • SAP.
8. Costes y beneficios •Costes: • Creación • Mantenimiento • Licencias	•Beneficios: • Reducir demoras plazos de facturación • Reducción Backoffice	

Figura 58. Business Model Canvas de la Iniciativa Robot de Prefacturas.

4.3.4 Fase de Tuning

En esta cuarta fase, también de una duración de una semana, el objetivo consistió en elaborar el entregable final de la fase de Diseño, incluyendo un *porfolio backlog* final, un análisis de beneficio – costes, y tiempos para cada cada iniciativa, el business model canvas para todas las iniciativas, y el análisis del futuro estado digital con la inclusión de todas las iniciativas implementadas.

4.3.4.1 Sala de Liquidaciones.

En esta última entrega se definieron los *pain-points* definitivos, el valor de los *kpi´s* asociados, el proceso *to be* final, el listado de iniciativas con su coste y plazo asociado para su implementación.

Además de esto, se terminaron de elaborar todos los business model canvas, así como el cálculo de los ahorros (FTE[28]) y retornos de las posibles inversiones.

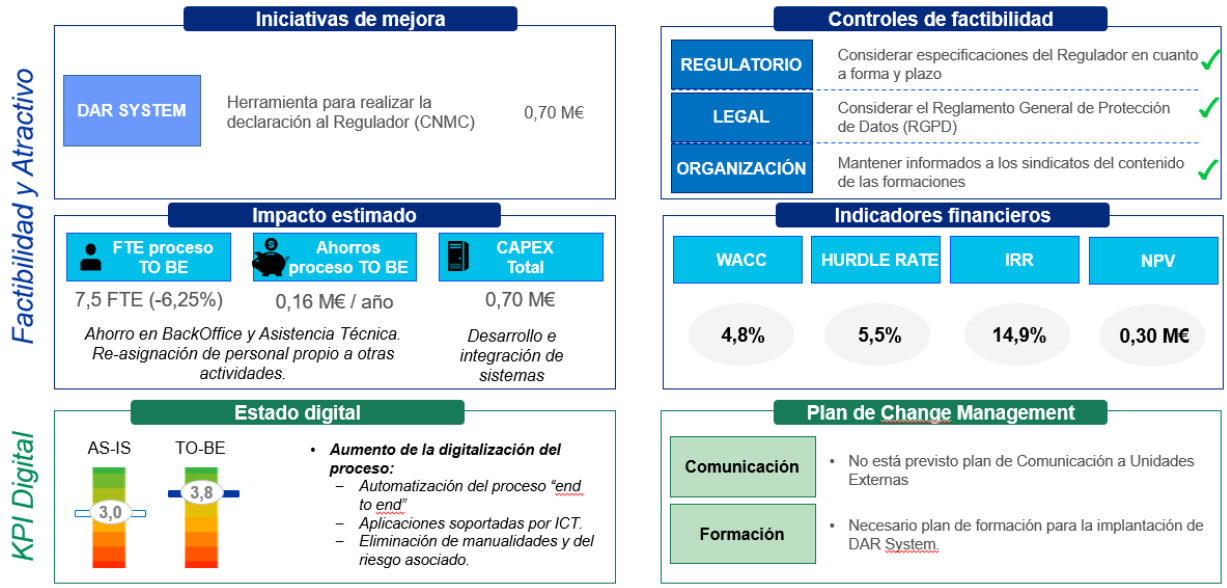


Figura 59. Resumen de la iniciativa que se presentó.

[28] FTE: Equivalente a tiempo completo (full-time equivalent, en inglés). Es una forma de medir la cantidad de empleados a tiempo completo que se necesitarían para llevar a cabo el trabajo realizado en una empresa. Se obtiene dividiendo las horas de trabajo de varios trabajadores o empleados a tiempo parcial por la cantidad de horas de un período laboral

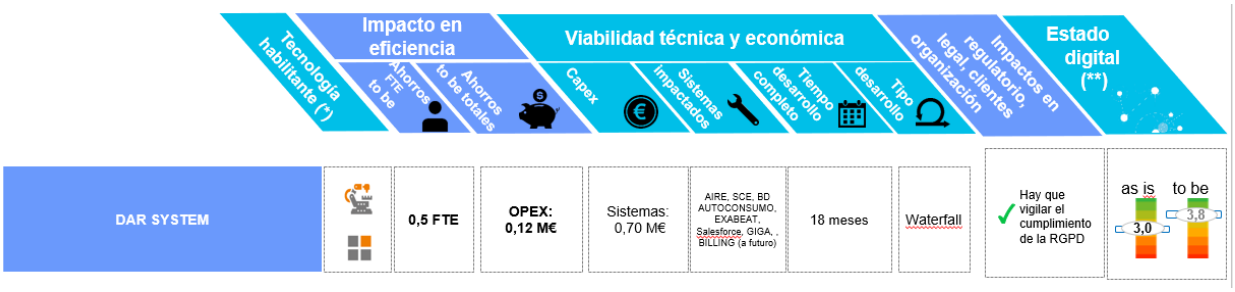


Figura 60. Resumen de la iniciativa con costes, ahorros y estrategia de desarrollo.

Figura 61. Análisis del impacto de la herramienta en el estado digital del proceso.

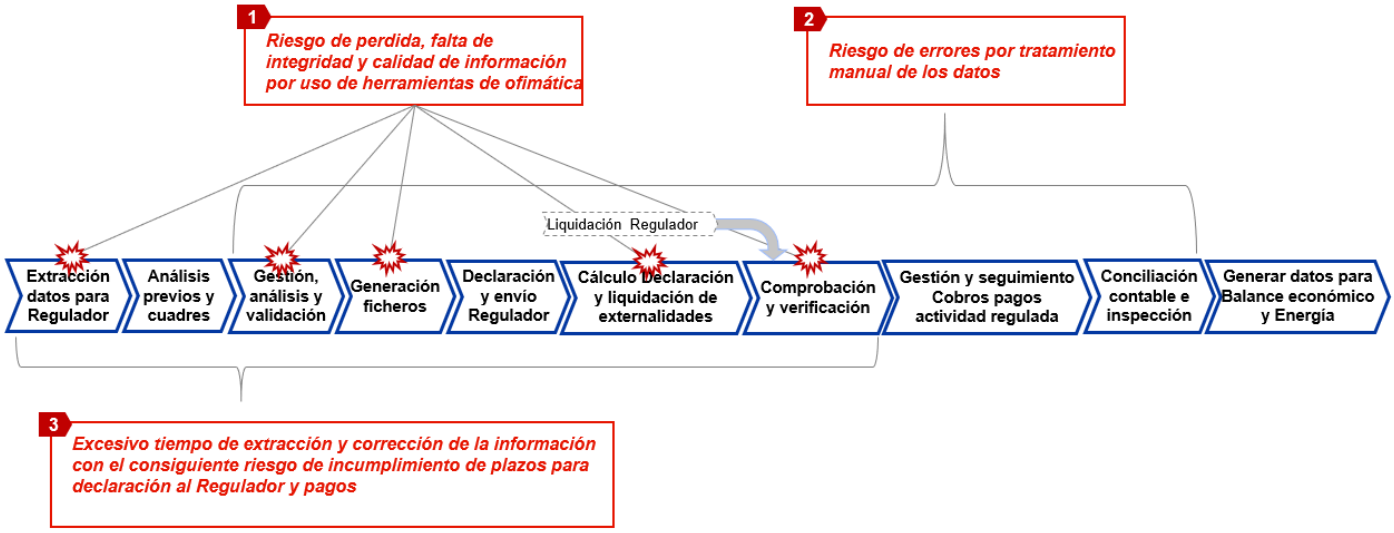
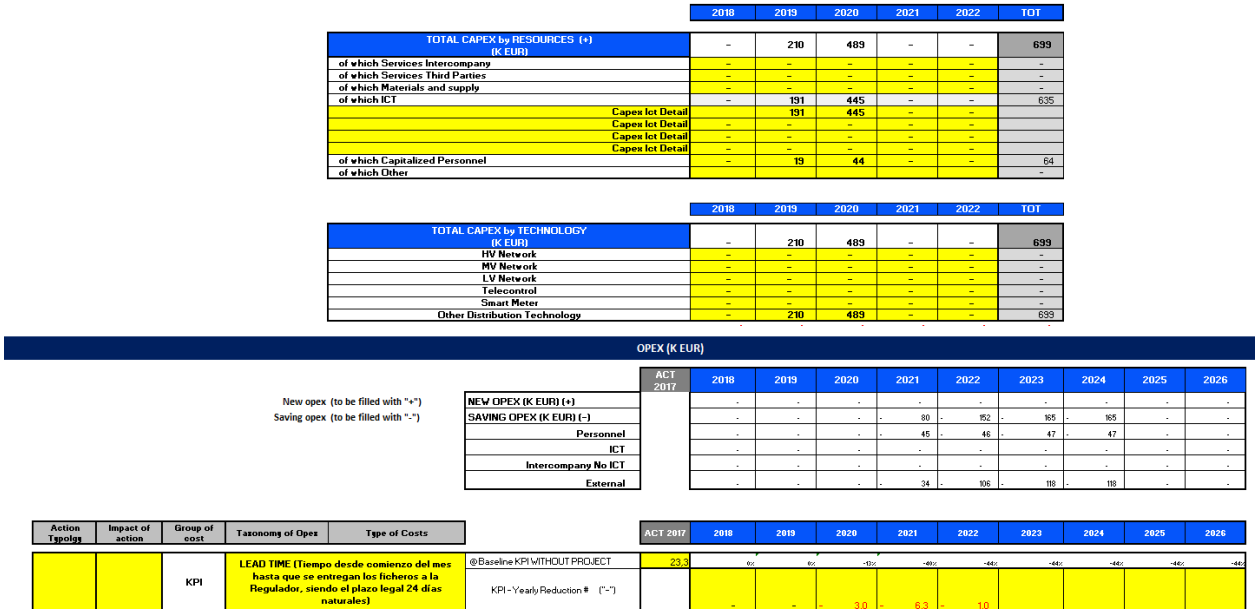


Figura 62. Resumen de la iniciativa con costes, ahorros y estrategia de desarrollo.



TOTAL CAPEX by RESOURCES (+) (K EUR)	2018	2019	2020	2021	2022	TOT
	-	210	489	-	-	699
of which Services Intercompany	-	-	-	-	-	-
of which Services Third Parties	-	-	-	-	-	-
of which Materials and supply	-	-	-	-	-	-
of which ICT	-	191	445	-	-	635
Capex Ict Detail		191	445	-	-	
Capex Ict Detail	-	-	-	-	-	
Capex Ict Detail	-	-	-	-	-	
Capex Ict Detail	-	-	-	-	-	
of which Capitalized Personnel	-	19	44	-	-	64
of which Other						-

TOTAL CAPEX by TECHNOLOGY (K EUR)	2018	2019	2020	2021	2022	TOT
	-	210	489	-	-	699
HV Network	-	-	-	-	-	-
MV Network	-	-	-	-	-	-
LV Network	-	-	-	-	-	-
Telecontrol	-	-	-	-	-	-
Smart Meter	-	-	-	-	-	-
Other Distribution Technology	-	210	489	-	-	699

OPEX (K EUR)

New opex (to be filled with "+")
Saving opex (to be filled with "-")

	ACT 2017	2018	2019	2020	2021	2022	2023	2024	2025	2026
NEW OPEX (K EUR) (+)		-	-	-	-	-	-	-	-	-
SAVING OPEX (K EUR) (-)		-	-	-	- 80	- 152	- 165	- 165	-	-
Personnel		-	-	-	45	46	47	47	-	-
ICT		-	-	-	-	-	-	-	-	-
Intercompany No ICT		-	-	-	-	-	-	-	-	-
External		-	-	-	- 34	- 106	- 118	- 118	-	-

Action Typolgy	Impact of action	Group of cost	Taxonomy of Opex	Type of Costs		ACT 2017	2018	2019	2020	2021	2022	2023	2024	2025	2026
		KPI	LEAD TIME (Tiempo desde comienzo del mes hasta que se entregan los ficheros a la Regulador, siendo el plazo legal 24 días naturales)		@Baseline KPI WITHOUT PROJECT	23,3	0%	0%	-13%	-40%	-44%	-44%	-44%	-44%	-44%
					KPI - Yearly Reduction # ("-")		-	-	- 3,0	- 6,3	- 1,0				

Figura 63. Excel económico para calcular costes, ahorro y tasas de amortización para las iniciativas.

Estos ahorros, se basaron en varios aspectos:

- Ahorro en coste de mantenimiento de los sistemas actuales.
- Ahorro en coste de explotación de los sistemas actuales.
- Ahorro en personal propio, como resultado de la automatización de los procesos.
- Ahorro en personal externo, como resultado de las mejoras implementadas.
- Se presupone un horizonte de amortización de 5 años.

4.3.4.2 Sala de Facturación y Cobros.

De forma análoga a la otra sala, se terminaron de detallar los dossieres finales a presentar, terminando de definir los *pain-points* con sus *kpi´s* correspondientes, el proceso *to be* con las mejoras identificadas, la preestimación de tiempo, coste y beneficios para las distintas iniciativas, los business model canvas de cada una de las iniciativas, y la mejor metodología para desarrollar las distintas iniciativas.

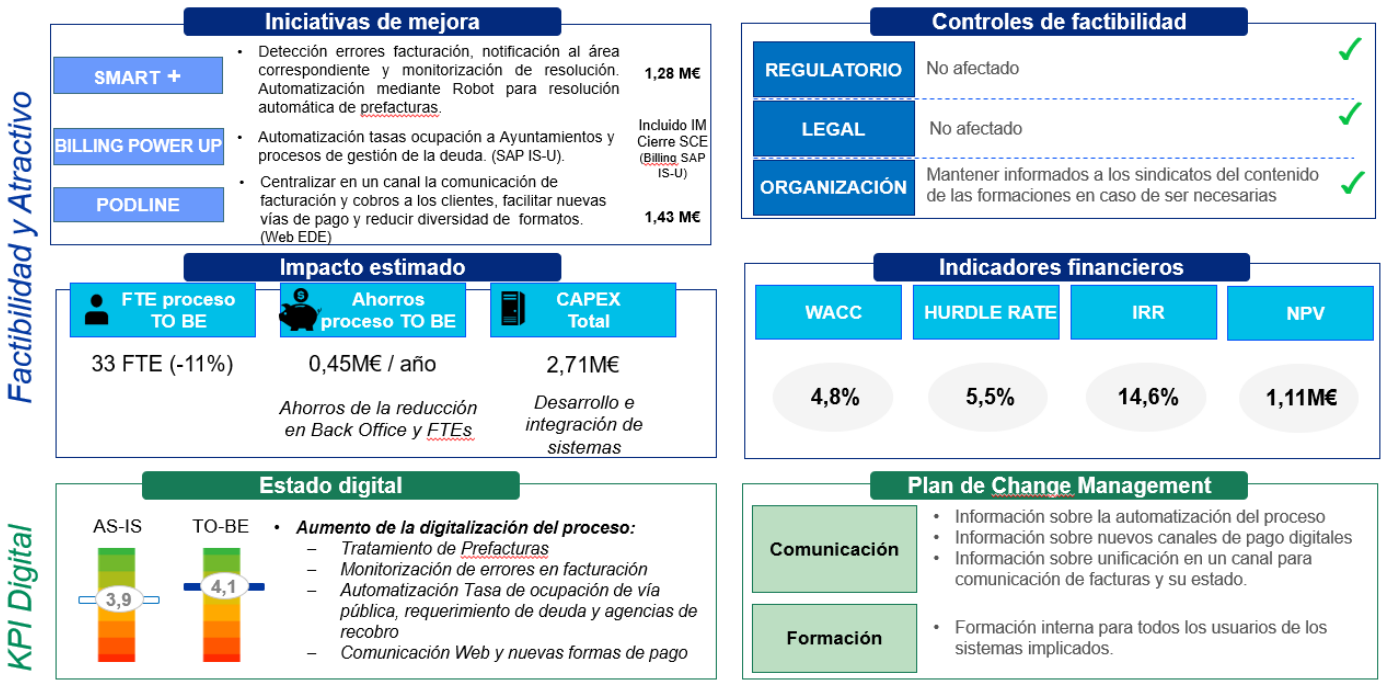


Figura 64. Resumen de iniciativas del proceso de Facturación y Cobros.

Figura 65. Mejoras en el proceso con la implantación de las iniciativas.

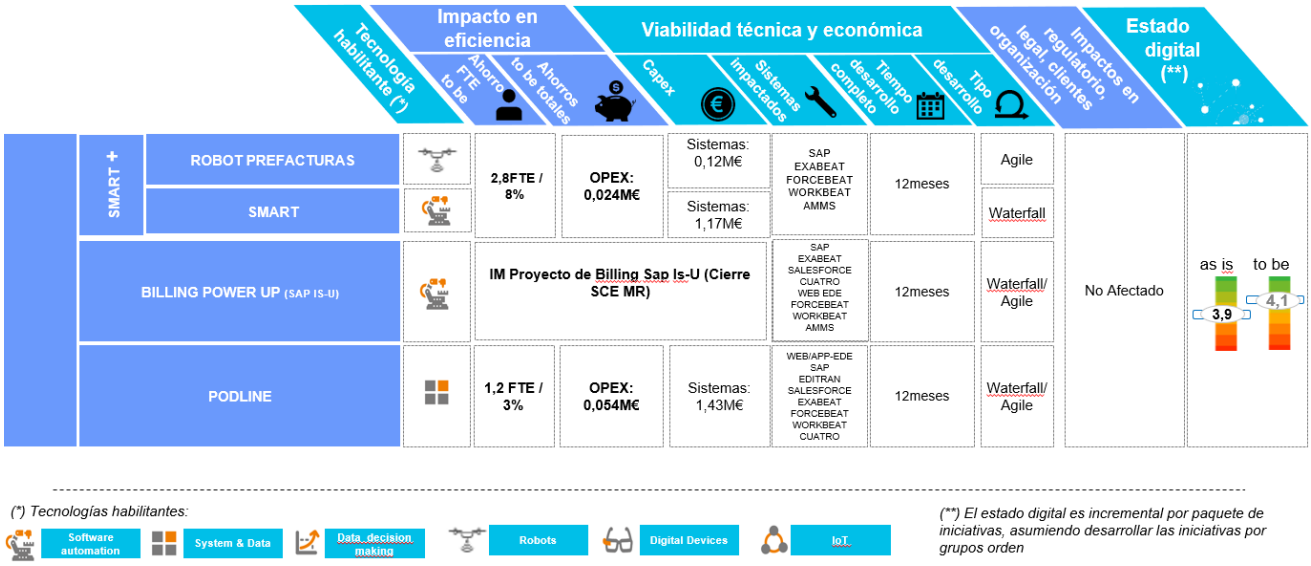


Figura 66. Análisis Coste-Beneficio, plazo, metodología, y estado digital de las iniactivas.

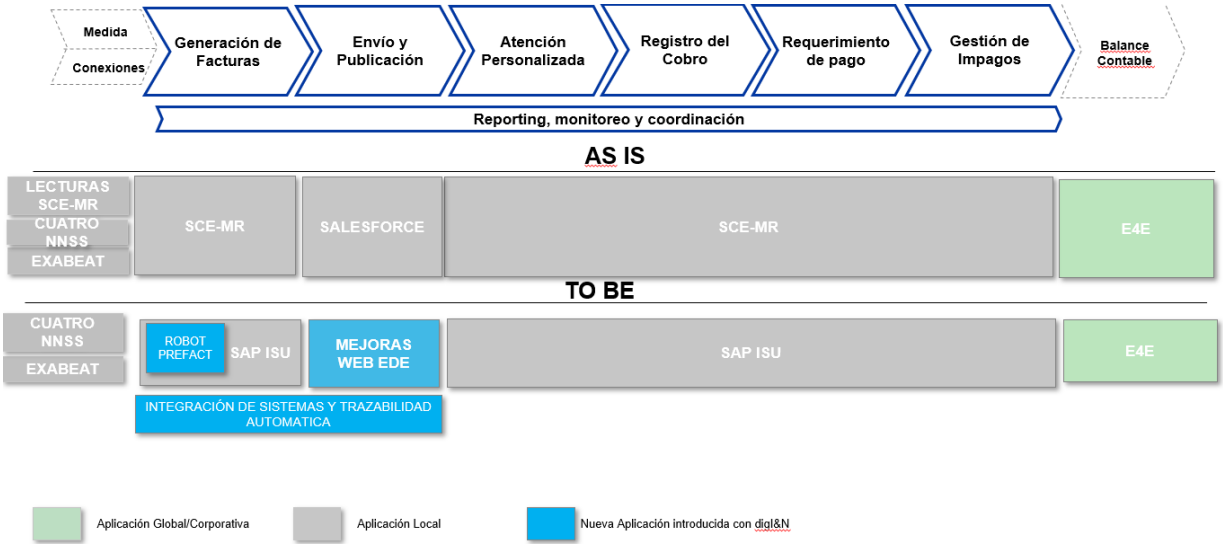


Figura 67. Proceso *AS IS* y *TO BE*, con la introducción de los nuevos sistemas.

4.3.5 Fase de Priorization + Retrospective (Ceremony)

En esta siguiente fase, celebrada mediante un evento en Barcelona, el objetivo, fue el de intercambiar ideas con personas de otros procesos de negocios, de cara a encontrar nuevas iniciativas, y hacer una reflexión conjunta de todo el proceso llevado a cabo.

Hora	Fase	Descripción	Ponentes
10:30 11:00	Opening sesión y objetivos	Apertura de la sesión y explicación de la agenda del día	Sr Caccialupi y Sr. Brogi introducen el día Siag y Opinno explican las dinámicas
11:00 11:30	Inspiración y Disruption view	Presentación de global trends	Opinno
11:30 13:15	Gallery room – World Cafe	Identificación de una nueva iniciativa y desarrollo del BMC de la misma a alto nivel,.	Siag y Opinno coordinan la dinámica
13:15 14:15	Metodología y DigItaly	Fase de Inception Referencias coordinación DigItaly	Digital Hub DH presenta los próximos pasos la planificación y explica la fase de Inception dentro de la metodología Agile
14:15 15:15	Big Agile Retrospective	Retrospectiva grupal regreso al pasado. Actividad en equipo de reflexión y feedback.	Siag y Opinno coordinan la dinámica
15:15 15:30	Cierre	Cierre de la sesión	Biondi

Figura 68. Resumen de la agenda del evento.

Durante el evento, pudimos aportar nuestro punto de vista en un proceso de negocio totalmente diferente a los ya trabajados, y junto a personas que trabajaban en áreas totalmente diferentes.

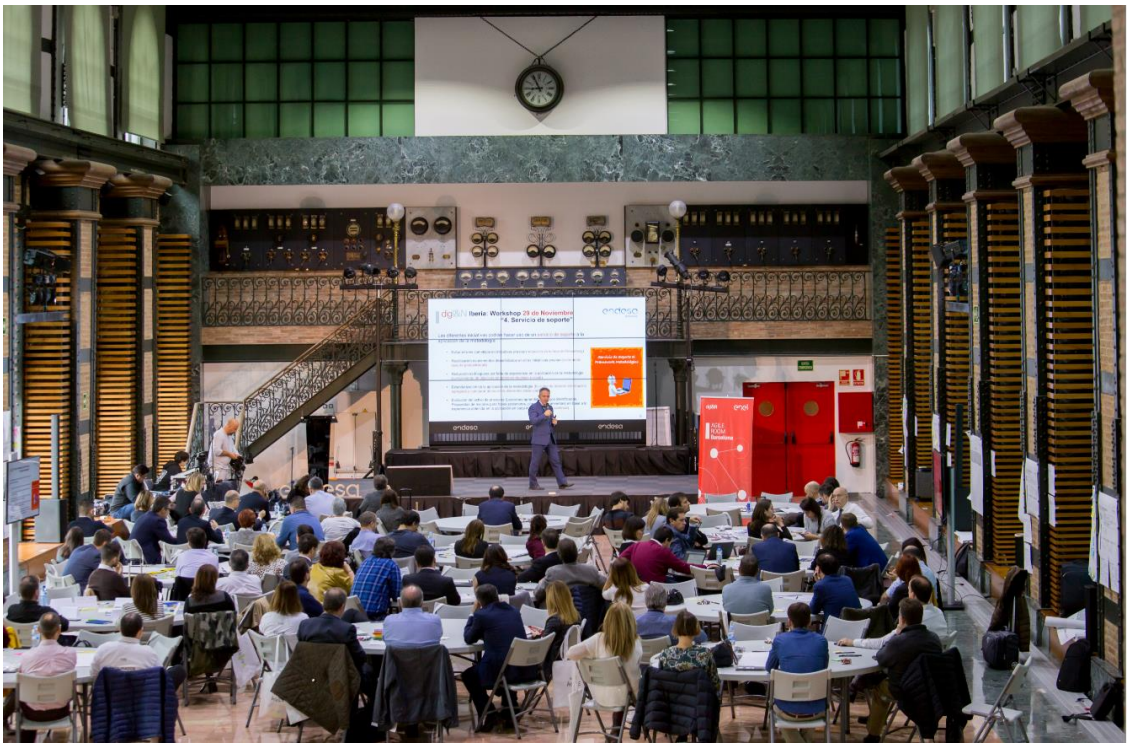


Figura 69. Sala del evento en Barcelona.

Figura 70. Distintos participantes revisan un proceso de negocio.

Figura 71. Conclusiones extraídas de la retrospective.

Como resultado del encuentro pudimos compartir conclusiones, llegando entre todos los asistentes a encontrar conclusiones comunes tanto en beneficios aportados por el uso de la metología, como en carencias presentes a la hora de llevarlas a cabo.

Todas estas conclusiones, fueron recogidas en unos cuadros de trabajo que fueron enviados por correo, y comentados a la dirección global del proyecto DIG&IN, para que sean tenidos en cuenta en el desarrollo de nuevas iniciativas, o análisis de procesos de negocio que no hayan sido revisados hasta ahora.

De esta forma, podemos enumerar algunos de las conclusiones extraídas por todos los miembros:

- **Ventajas:**
    - Mejora del trabajo en equipo.
    - Mayor conocimiento de procesos de negocio.
    - Nuevos contactos.
    - Mayor motivación.
    - Autoorganización.
- **Desventajas:**
    - Dedicación exclusiva.
    - Carga de trabajo.
    - Falta de formación.
    - Necesidad de mayor planificación.
    - En ocasiones, alto stress.

Como resultado de la presentación, de todas las iniciativas en los dossieres de todos los procesos de negocio, desde la dirección general, se priorizaron las iniciativas y se liberaron las partidas presupuestarias correspondientes a las distintas iniciativas que tuvieron el visto bueno.

En lo referente a las dos salas que aquí se tratan (Liquidaciones, y Facturación y Cobros), salieron adelante las iniciativas de:

- **Sala de liquidaciones**: DAR System, por un valor de 700.000 € y un plazo de ejecución de 18 meses, con una modalidad de implementación waterfall
- **Sala de Facturación y cobros:** Smart+, por un valor de 1.280.000 € y un plazo de ejecución de 12 meses, con una modalidad de implementación híbrida waterfall/agile.

4.3.6 Fase de Inception.

Para esta fase de inception, centrada ya en el desarrollo de los proyectos asociados, seguiremos con el ejemplo del modelo de facturación y cobros, ya que el de liquidaciones seguirá una metodología tradicional en su desarrollo, que no aporta nada adicional sobre este trabajo.

En esta fase, de una duración aproximada de 2 a 3 semanas por cada sprint (como anteriormente comentábamos en el cuadro esquemático) pero iterativa hasta finalizar todas las historias de usuario, se definirán:

- Las diferentes *historias de usuario*[29] de acuerdo con el *product backlog* priorizado por el *product owner*.
- Las tareas asociadas a cada historia, así como posibles impactos.
- Las pruebas que validaran las historias de usuario.
- La velocidad o capacidad del equipo de trabajo sobre cada sprint en puntos de esfuerzo (valor que se irá ajustando en cada sprint).
- Número de sprints necesarios para completer todas las historias de usuario.
- Fecha de revisión del sprint realizado.

Para llevar a cabo todas estas tareas, se utilizarán las herramientas comentadas como el *Sprint Planning Poker*, el *diagrama de Burndown*, o el *daily meeting,*entre otras.

[29] Cada historia de usuario (*user story)* hace refencia a una parte/funcionalidad potencialmente entregable dentro del producto global

Figura 73. Equipo analizando el producto backlog.

Figura 72. Equipo de trabajo trabajando sobre una historia de usuario.

4.3.1 Fase de Execution.

En esta última etapa de desarrollo, de una duración aproximada de 2 a 3 semanas por ejecución, el trabajo a desarrollar estará centrado en desarrollar las distintas tareas asociadas a cada historia de usuario sobre la que se esté trabajando, y su presentación al product owner al finalizar cada ciclo.

De esta forma, en cada iteración, el usuario podrá ser partícipe directo del proyecto, pudiendo comprobar cómo el equipo en su conjunto va generando cada parte del proyecto, aportando nuevas funcionalidades en cada entrega, y, en definitiva, generando valor sobre el producto final.

Todo esto, unido a la capacidad por parte del *product owner*, de modificar en todo momento, sus propias necesidades, y/o el orden de las nuevas funcionalidades a aportar, no hace sino entregar más valor añadido sobre el producto.

Figura 74. Otro equipo Agile en fase de análisis de una historia de usuario.

En el momento de redacción y entrega de este trabajo, el desarrollo de los proyectos con metodología ágil aún no había llegado a la fase de liberación de versiones y puesta en producción, por lo que no se pueden plasmar en este documento, pero con el desarrollo expuesto a lo largo del documento quedaría cubierta la forma de desarrollar cada hito, y su explicación.

Una similitud para explicar esta entrega de valor por hitos, podría presentarse por ejemplo, a la hora de hablar de cualquier aplicación para el móvil: en dichos entornos, se ha pasado en los últimos años, de una versión definitiva sin cambios en una app, a una aplicación, que cada 2-3 semanas, genera una nueva actualización que los usuarios descargan y utilizan, y en la que se le entrega un nuevo valor añadido al producto como mejora de la anterior versión, no existiendo nunca una versión final, porque hace tiempo que dejó de existir.

5 CONCLUSIONES

En esta última sección, trataremos de exponer las conclusiones en relación con el uso de este tipo de metodología aplicado a proyectos, así como las ventajas e inconvenientes encontradas durante su aplicación a lo largo de todo el proceso.

5.1 Conclusiones generales.

El uso del método ágil aplicado al desarrollo de proyectos trae consigo una serie de ventajas, experimentadas a lo largo de su uso como pudieran ser:

- Acelera la entrega de productos digitales.
- Gestionar prioridades cambiantes.
- Aumenta la eficacia de los equipos.
- Alineamiento entre Negocio y Tecnologías de la Información (TI).
- Mejora el *time-to-market*.
- Aumenta la satisfacción del cliente.
- Captación y retención de talento.
- Entorno colaborativo y de innovación.
- Entrega constante de valor.
- Equipos empoderados y autoorganizados.
- Feedback constante del cliente.

Sin embargo, a pesar de traer todas estas mejoras, existen otros factores en los cuales se podría incidir, para tratar de mejorar:

- **Síndrome del Burn Out:** trabajar con métodos ágiles puede aumentar el estrés de algunos de los desarrolladores.
- **Falta de participación del cliente:** a pesar del entusiasmo por trabajar en base a métodos ágiles, a la hora de la verdad, el cliente no lo pone tan fácil para mantener el ritmo de comunicación necesario.
- **Aumento del riesgo:** el ritmo del trabajo y el entusiasmo por los logros alcanzados puede hacer perder la aversión al riesgo que se necesita en cualquier proyecto.
- **Falta de controles de calidad:** la calidad no puede obviarse. De nada sirve avanzar a toda velocidad, si el producto no reúne las condiciones necesarias.
- **Falta de documentación:** Las metodologías ágiles apuestan por un software que funciona por etapas, dejando en un segundo plano la gestión documental, porque supuestamente ésta se intregra como control de cambios en el proceso.
- **Fuerte dependencia de los líderes:** Los equipos de trabajo dependen del liderazgo de la persona responsable del proyecto. Las continuas reuniones y evaluaciones periódicas hacen que la persona responsable del proyecto centralice gran parte de las responsabilidades y decisiones.

5.2 Próximas áreas de desarrollo.

Se proponen como futuras áreas de desarrollo en proyectos relacionados con las metodologías ágiles, los siguientes temas:

- Comparativas de estudio *waterfall- agile* para un mismo proyecto.
- Escalabilidad de *Agile* en otros niveles dentro de una organización empresarial.
- Business Agility.
- El futuro de Agile.
- Lean como resultado de la aplicación de agile.
- Uso de metodologías agiles dentro y fuera de España.
- Uso de Watergile, o Agilefall.
- Nuevas corrientes post-agile.

En definitiva, se propone continuar el desarrollo e investigación sobre estas nuevas formas de trabajar que, siendo no tan novedosas- pues llevan más de un par de décadas entre nosotros- están aún por desarrollar, estandarizar y dar a conocer de una forma sencilla, práctica y cercana tanto para el desarrollador que las utiliza en su día a día, como para el usuario que se transforma en una parte activa, e involucrada del proceso en el uso de estas nuevas metodologías.

El hecho de vivir en la era post-metodológica, en la que la tecnología nos rodea en nuestro día a día, evoluciona, y se revoluciona de la noche a la mañana, nos lleva a pensar que quizá estemos viviendo actualmente una nueva etapa de cambio metodológico que nadie ha descrito aún, en la que la preparación al cambio constante es la principal necesidad de un negocio, que nunca ha vivido una etapa tan inestable de cambios, y donde el diálogo y la colaboración en equipo será clave para la supervivencia en el mercado.

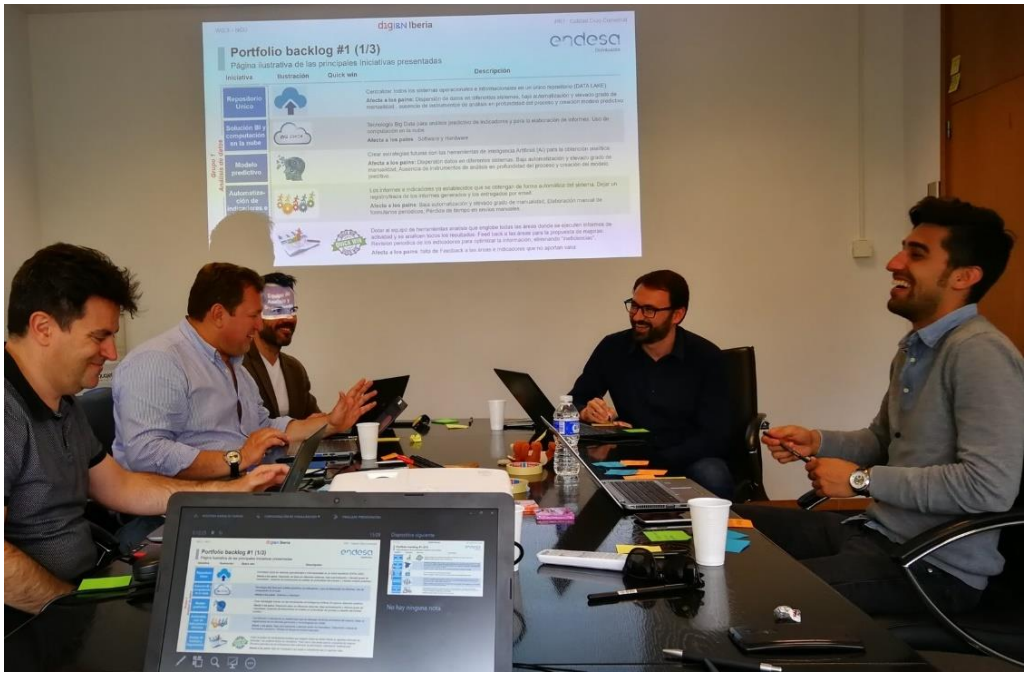


Figura 75.Equipo agile, debatiendo sobre la priorización de las historias de usuario.

6 REFERENCIAS

ADDISON-WESLEY, 2006. *Software Engineering, 9th Edition*. S.l.: s.n. ISBN 9780137035151.

ANDERSON, D.J., 2010. *Kanban: Successful Evolutionary Change for Your Technology Business*. S.l.: Blue Hole Press. ISBN 9780984521401.

AVISON, D.E. y FITZGERALD, G., 1998. *Information Systems Development: Methodologies, Techniques, and Tools*. 2nd. S.l.: McGraw-Hill Higher Education. ISBN 0077092333.

BECK, K., 2000. *Extreme Programming Explained: Embrace Change*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN 0-201-61641-6.

BOEHM, B.W., 1979. Software Engineering-as It is. *Proceedings of the 4th International Conference on Software Engineering*. Piscataway, NJ, USA: IEEE Press, pp. 11-21.

CARVAJAL RIOLA, J.C., 2008. METODOLOGÍAS ÁGILES: Tesis Final de Máster. ,

DERNIAME, J.-C., KABA, B.A. y WASTELL, D.G., 1999. Software Process: Principles, Methodology, Technology. En: . S.l.: Springer, ISBN 3-540-65516-6. DOI 10.1007/3-540-49205-4.

GIMSON, L., 2012. Metodología ágiles y desarrollo basado en conocimiento. , pp. 97.

HIRSCH, J., 2004. Globalización. , pp. 1-23. DOI doi: 10.1016/j.physleta.2005.04.058.

JOSÉ, H., CANÓS, P. L., & CARMEN, P., 2003. «Metodologías Ágiles en el Desarrollo de Software». ,

KNIBERG, H., DE, P., SUTHERLAND, J. y COHN, M., 2007. *Una historia de guerra Ágil SCRUM Y XP DESDE LAS TRINCHERAS*. S.l.: s.n. ISBN 978-1-4303-2264-1.

LINDVALL, M., BASILI, V., BOEHM, B., COSTA, P., DANGLE, K., SHULL, F., TESORIERO, R., WILLIAMS, L. y ZELKOWITZ, M., 2002. Empirical Findings in Agile Methods. En: D. WELLS y L. WILLIAMS (eds.), *Extreme Programming and Agile Methods --- XP/Agile Universe 2002*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 197-207. ISBN 978-3-540-45672-8.

LOPEZ.R.E., 2015. Management of Business Projects. , no. Concapan Xxxv.

ORTIZ, M., 2010. Métodos y técnicas para la gestión de proyectos de software. , pp. 106-127.

ROJAS CONTRERAS, M., ESTEBAN VILLAMIZAR, L.A., ORJUELA DUARTE, A. y CICOM, C.C., 2011. Modelo de integración de las actividades de gestión de la guía del PMBOK, con las actividades de ingeniería, en proyectos de desarrollo de software. *Revista Avances en Sistemas e Informática*, vol. 8, no. 2, pp. 97-105. ISSN 1909-0056. DOI 10.1007/s11202-007-0050-0.

ROYCE, W.W., 1970. Managing the Development of large Software Systems. *IEEE Wescon: Technical Papers of Western Electronic Show and Convention (WesCon)*. Los Alamitos, CA, USA: IEEE Computer Society Press, pp. 1-9. ISBN 0-89791-216-0. DOI 10.1016/0378-4754(91)90107-E.

SATPATHY, T., 2017. *Una guía para el CUERPO DE CONOCIMIENTO DE SCRUM (Guía SBOK™) 3ra Edición Una guía integral para la entrega de proyectos utilizando Scrum*. S.l.: s.n. ISBN 9780989925204.

SCACCHI, W., 2002. Process Models in Software Engineering. *Encyclopedia of Software Engineering*, no. February, pp. 1-24. ISSN 00189162. DOI 10.1002/0471028959.sof250.

SCHWABER, K. y BEEDLE, M., 2001. *Agile Software Development with Scrum*. 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR. ISBN 0130676349.

TAKEUCHI, H. y NONAKA, I., 1986. The new new product development game. *Harvard business review*, vol. 64, no. 1, pp. 137-146.