Trabajo de Fin de Grado
Ingeniería de Organización Industrial

La importancia de las metodologías ágiles en los proyectos de desarrollo de software

Autor: Natividad Camacho Sánchez
Tutor: José Guadix Martín

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

Sevilla, 2023

Trabajo de Fin de Grado
Ingeniería de Organización Industrial

La importancia de las metodologías ágiles en los proyectos de desarrollo de software

Autor:
Natividad Camacho Sánchez

Tutor:
José Guadix Martín
Catedrático de Universidad

Dpto. de Ingeniería de Organización Industrial II
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2023

Trabajo de fin de grado: La importancia de las metodologías ágiles en los proyectos de desarrollo de software

Autor: Natividad Camacho Sánchez

Tutor: José Guadix Martín

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2023

El Secretario del Tribunal

A mis amigos

A mi familia

A mis profesores

A mis compañeros

Agradecimientos

Estar escribiendo estas palabras significa que se cierra otra etapa más en mi vida. Gracias a todas esas personas que me han dado su apoyo, conocimiento e inspiración, ya no solo para realizar este trabajo, sino para seguir con este camino hasta el final, sobre todo en los momentos más difíciles. No podría haber llegado a este punto sin vuestra ayuda.

Natividad Camacho Sánchez

Sevilla, 2023

Resumen

Este trabajo recoge la importancia de las metodologías ágiles para la gestión de proyectos de desarrollo de software, con el fin de encontrar algunas mejoras o procedimientos que optimicen la gestión de este tipo de proyectos en una empresa específica objeto de estudio. Para ello, se exponen algunas de estas metodologías como pueden ser Scrum, Kanban o Extreme Programming indicando los beneficios que pueden aportar, y una serie de herramientas que permiten aplicar con mayor facilidad estos procedimientos en la empresa, como Jira o Trello.

Índice

Índice de Tablas

Índice de Figuras

1 INTRODUCCIÓN

Cuida los pequeños detalles. Un pequeño agujero puede hundir un gran barco.

-Benjamin Franklin-

Este proyecto nació con motivo de la finalización del Grado de Ingeniería de Organización Industrial, uniéndose también a la necesidad de establecer un procedimiento estándar o algunas mejoras para la gestión de proyectos que había en la empresa tecnológica en la que realizaba las prácticas, y en la que posteriormente me contrataron.

La función principal de esta compañía es desarrollar e implantar soluciones de software en empresas que quieran mejorar sus procesos y las experiencias de sus clientes. Tienen dos softwares estrella que son en los que me he centrado estos meses. Las empresas que requieran estas soluciones pueden obtenerlas estándar o con desarrollos a medida para cubrir sus necesidades más específicas.

Mi puesto desde empezar este trabajo es el de técnico de preventa de software, puesto para el que es importante conocer el producto y la manera de implantación en el cliente. Este es el origen del TFG.

En la empresa, los responsables de proyecto tienen bastante autonomía para gestionar sus proyectos y abarcan algunas tareas que no son especialmente propias de este puesto, cosa que les quita tiempo que podrían dedicar a analizar más en profundidad los estados del proyecto y ver si se pueden mejorar los procesos. Por eso mismo, se van a analizar las prácticas actuales y proponer un marco de trabajo o algunas mejoras que les puedan ayudar a optimizar sus operaciones

2 GLOSARIO

- GAPs: Este término lo utilizamos para hacer referencia a los desarrollos a medida, es decir, a cualquier modificación que haya que realizar sobre el producto estándar y que aporte valor al producto.
- *Disturbance*: A diferencia del anterior, este término lo usamos para aquellos desarrollos a medida que no consideramos que aporten valor a la forma de trabajar del cliente, pero que, aun así, quieren que los llevemos a cabo.
- Entregable: Resultado medible, tangible y verificable que debe realizarse para llevar a cabo un proyecto o una parte de él.
- Hito: Es un evento que indica el paso de una fase a otra, obteniéndose uno o más entregables.
- *Epic*: Concepto usado en gestión de proyectos para referirse a una gran cantidad de trabajo que se puede desglosar en tareas individuales que se deberán finalizar para llevar a cabo una función (Atlassian, 2022a). Esta tarea es una incidencia secundaria del epic, siendo este último la incidencia principal.
- *Backlog*: término inglés que hace referencia a la acumulación de algo, especialmente trabajo incompleto o cosas de las que debemos ocuparnos.
- Historia de usuario o *User Story*: es la unidad de trabajo más pequeña que se utiliza en un marco ágil y hace referencia a una tarea de desarrollo que se suele expresar como "persona + necesidad + propósito". El objetivo de esta es definir cómo un elemento de trabajo entregará valor particular al cliente. (Rehkopf, 2022a)

3 OBJETIVOS Y ALCANCE

El objetivo principal de este trabajo es definir un marco de trabajo en las prácticas de los directores de proyectos para su gestión en aquellos que contengan desarrollos software a medida, teniendo como base la metodología ágil, especialmente el procedimiento Scrum, pero adaptándolo a las herramientas que se consideren necesarias. De este marco de trabajo propuesto lo que más interesa es proponer una serie de mejoras o actividades que optimicen el rendimiento del equipo.

Las metodologías ágiles surgen en contraposición a las metodologías tradicionales ya que estas últimas no eran capaces de adaptarse a los cambios constantes que suelen darse en los proyectos de desarrollo de software durante casi todo su ciclo de vida, por lo que se presentó la necesidad de crear otros procedimientos que introdujesen esa capacidad adaptativa hacia los cambios que sugerían los clientes.

Además de esta nueva necesidad, también se identificó que las metodologías tradicionales requerían de un nivel de burocracia y documentación de procesos que ralentizaba los proyectos de software, cuando lo que se requiere en esta clase de proyectos es lo contrario. Por esto mismo, este tipo de metodologías como Scrum se centran en documentar solo la información imprescindible, agilizando los procesos.

Estas son las razones por las que este trabajo se centrará en este tipo de metodologías para buscar las actividades que permitan mejorar las prácticas de los directores de proyecto en los proyectos de desarrollo de software.

Para proponer estas mejoras para la empresa objeto de estudio, primero se deberá empezar comprendiendo bien qué son las metodologías ágiles y su importancia en el sector del software. Luego, se expondrán los tipos más destacables, y que se piensa que pueden ser de mayor ayuda en los proyectos de desarrollo e implantación de softwares, recalcando las ventajas e inconvenientes de utilizarlas, así como también se desarrollarán las distintas herramientas y técnicas que van a permitir optimizar el trabajo.

Una vez se hayan identificado qué actividades de mejora pueden ser más beneficiosas, la intención es llevarla a la práctica en el proyecto de implantación de un software en una empresa de distribución de Sevilla, con los diferentes desarrollos a media que se hayan acordado con la empresa contratante, y luego analizar los resultados obtenidos.

Fue un poco optimista el plantear que en la misma línea temporal de la realización de este trabajo daría tiempo a definir este marco de trabajo, ponerlo en práctica y analizar los resultados, contando también con que este trabajo se realiza fuera de las horas laborales y en el último año de carrera, pero se deja la puerta abierta para un próximo trabajo o investigación.

De esta forma, los objetivos primitivos fueron:

1. Explicar la importancia del uso de metodologías ágiles en el sector del software.
2. Identificar qué modelos de metodología ágil y herramientas son más beneficiosos.
3. Definir una nueva metodología o mejoras para la gestión de proyectos de desarrollo software de esta empresa.
4. Poner la metodología en práctica.
5. Analizar las mejoras y sus conclusiones.

Finalmente, se tomó la decisión de apartar el objetivo número cuatro y modificar el cinco, quedando de la siguiente manera:

1. Explicar la importancia del uso de metodologías ágiles en el sector del software.
2. Identificar qué modelos de metodología ágil y herramientas pueden ser más beneficiosos.
3. Definir una nueva metodología o mejoras para la gestión de proyectos de desarrollo software de esta empresa.
4. Analizar e identificar posibles inconvenientes que se puedan ocurrir.

4 GESTIÓN DE PROYECTOS

Si se quiere establecer un marco de trabajo para el departamento de gestión de proyectos de esta empresa mencionada, primero habrá que explicar en qué consiste y qué hacen los responsables de proyecto. Se ha optado por dividir este capítulo en dos apartados: en el primero se tratará la visión tradicional de la gestión de proyectos y en el segundo por qué se necesitan las metodologías ágiles para aquellos proyectos con desarrollo de software.

4.1. Visión tradicional

Para empezar, habrá que definir qué se entiende por 'proyecto'. Según el PMBOK (Project Management Institute, 2013), un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único. También se puede definir como que un proyecto hace referencia a un conjunto de actividades coordinadas e interrelacionadas delimitadas en el tiempo, es decir, que tienen un principio y un fin. Estas actividades son llevadas a cabo por una persona o equipo de trabajo para satisfacer un objetivo concreto, basándose en los requisitos del cliente, que determinan el alcance de este proyecto, y en los límites de presupuesto de coste y tiempo que se definen (Campo Arranz, del Campo Domínguez, & Rodrigo Raya, 2014).

Hay que destacar que el fin de un proyecto es obtener un producto o servicio único, no puede haber dos proyectos iguales, por lo que un proyecto no se puede repetir en el tiempo. Si ese fuera el caso, estaríamos hablando de un proceso u operación, no de un proyecto (Organización Internacional dee Normalización [ISO], 2017).

Se ha mencionado que el alcance de un proyecto se define en base a los requisitos de los clientes, pero ¿qué se entiende por alcance? ¿quiénes son los clientes? El alcance se trata del trabajo que se necesita para llevar a cabo el producto, servicio o resultado que se ha acordado en el proyecto, asegurando que se cumplan las funciones y características que se especificaron con el cliente. Por otro lado, los clientes no solo serán compañías externas que quieran contratar los servicios, sino que también la propia empresa cuando se quieran realizar proyectos de mejora de los propios softwares, convirtiéndose así en un cliente teórico.

Un proyecto se terminará una vez se cumplan los objetivos establecidos o bien cuando se establezca que estos no pueden ser alcanzados (Marion, 2018).

La gestión de proyectos consiste en aplicar técnicas, conocimientos, herramientas y procedimientos para planificar actuaciones y actividades que conducirán a finalizar el proyecto, poder controlar la evolución de estas en coste y tiempo y poder anticipar y prever situaciones que se puedan dar. Hay que tomar decisiones lógicas, identificar la organización adecuada, atender a la documentación y gestionar eficaz y eficientemente los recursos, entre otras cosas.

El PMBOK (Project Management Institute, 2013) propone unos fundamentos para aplicar a los proyectos. Se trata de 49 procesos (aunque en la quinta edición de esta guía mencionada solo se recogen 47) agrupados en 5 grupos de procesos y en 10 áreas de conocimiento.

Los grupos de proceso son los siguientes:

- Inicio: En este proceso se autoriza y define el proyecto o una fase del mismo.
- Planificación: Se definen los objetivos y se planifican las acciones que se necesitarán llevar a cabo para lograr estos. También se definirá en esta el alcance del proyecto.
- Ejecución: Se trata de todos los procesos que se requieren realizar para finalizar el trabajo que se ha definido anteriormente.
- Seguimiento y control: Este proceso se basan en medir, supervisar y controlar cómo va avanzando el proyecto para ver si se está cumpliendo la planificación establecida o si hay desvíos.
- Cierre: Consiste en la aceptación del resultado obtenido de este proyecto o fase.

Como se menciona antes, los procesos se agrupan también en áreas de conocimiento. Un Área de Conocimiento hace referencia al conjunto de conceptos, actividades y términos que forman un entorno profesional, un ámbito de la dirección de proyectos o un área de especialización (Project Management Institute, 2013). Las 10 áreas de conocimiento son las siguientes:

- Gestión de la integración
- Gestión del alcance
- Gestión del cronograma
- Gestión de los costes
- Gestión de la calidad
- Gestión de los recursos
- Gestión de las comunicaciones

- Gestión de los riesgos
- Gestión de las adquisiciones
- Gestión de los interesados

Relacionándose estas áreas con los grupos de procesos que sean mencionado antes de la forma que se representa en la ilustración siguiente:

Áreas de Conocimiento	Grupos de Procesos de la Dirección de Proyectos				
	Grupo de Procesos de Inicio	Grupo de Procesos de Planificación	Grupo de Procesos de Ejecución	Grupo de Procesos de Monitoreo y Control	Grupo de Procesos de Cierre
4. Gestión de la Integración del Proyecto	4.1 Desarrollar el Acta de Constitución del Proyecto	4.2 Desarrollar el Plan para la Dirección del Proyecto	4.3 Dirigir y Gestionar el Trabajo del Proyecto	4.4 Monitorear y Controlar el Trabajo del Proyecto 4.5 Realizar el Control Integrado de Cambios	4.6 Cerrar Proyecto o Fase
5. Gestión del Alcance del Proyecto		5.1 Planificar la Gestión del Alcance 5.2 Recopilar Requisitos 5.3 Definir el Alcance 5.4 Crear la EDT/WBS		5.5 Validar el Alcance 5.6 Controlar el Alcance	
6. Gestión del Tiempo del Proyecto		6.1 Planificar la Gestión del Cronograma 6.2 Definir las Actividades 6.3 Secuenciar las Actividades 6.4 Estimar los Recursos de las Actividades 6.5 Estimar la Duración de las Actividades 6.6 Desarrollar el Cronograma		6.7 Controlar el Cronograma	
7. Gestión de los Costes del Proyecto		7.1 Planificar la Gestión de los Costos 7.2 Estimar los Costos 7.3 Determinar el Presupuesto		7.4 Controlar los Costos	
8. Gestión de la Calidad del Proyecto		8.1 Planificar la Gestión de la Calidad	8.2 Realizar el Aseguramiento de Calidad	8.3 Controlar la Calidad	
9. Gestión de los Recursos Humanos del Proyecto		9.1 Planificar la Gestión de los Recursos Humanos	9.2 Adquirir el Equipo del Proyecto 9.3 Desarrollar el Equipo del Proyecto 9.4 Dirigir el Equipo del Proyecto		
10. Gestión de los Recursos de Comunicación del Proyecto		10.1 Planificar la Gestión de las Comunicaciones	10.2 Gestionar las Comunicaciones	10.3 Controlar las Comunicaciones	
11. Gestión de los Riesgos del Proyecto		11.1 Planificar la Gestión de los Riesgos 11.2 Identificar los Riesgos 11.3 Realizar el Análisis Cualitativo de Riesgos 11.4 Realizar el Análisis Cuantitativo de Riesgos 11.5 Planificar la Respuesta a los Riesgos		11.6 Controlar los Riesgos	
12. Gestión de las Adquisiciones del Proyecto		12.1 Planificar la Gestión de las Adquisiciones	12.2 Efectuar las Adquisiciones	12.3 Controlar las Adquisiciones	12.4 Cerrar las Adquisiciones

Ilustración 1. Correspondencia entre las áreas de proceso y los grupos de procesos (Project Management Institute, 2013)

Hay que tener en cuenta que esta imagen se recoge de la 5ª edición de la guía del PMBOK redactada en 2013 y que en 2021 se publicó la 7ª edición, por lo que, como se mencionaba antes, se han hecho

varias modificaciones.

Uno de estos cambios es que en la 6ª edición añadieron 3 procesos más:

- Gestionar el conocimiento del proyecto. Pertenece al grupo de Ejecución y al área de Gestión de la Integración.
- Controlar los nuevos recursos. Pertenece agrupo de Seguimiento y Control y al área de Gestión de los Recursos.
- Implementar la respuesta a los riesgos. Pertenece al grupo de Ejecución y al área de Gestión de los Riesgos

El PMBOK documenta que se deben tratar todas estas áreas para una buena dirección de proyectos y, además, el director debe tener la capacidad de decidir qué procesos aplicar y a qué nivel llevarlos a cabo.

4.2. Debilidades de esta metodología para los proyectos de desarrollo del software

En los proyectos de desarrollo e implantación de software, los procesos clave son la especificación y análisis de los requisitos. El problema que hay en este sector, es que, cuando el cliente ve el resultado, siempre quiere modificaciones y esto supone un gran coste al estar ya en las últimas fases del ciclo de vida del proyecto. Estos posibles cambios de última hora pueden dar lugar a que haya que rehacer gran parte del trabajo anterior y el proyecto se pase de costes.

¿Por qué se puede pasar demasiado de costes un proyecto si se hacen cambios en las últimas fases? Bien, pues para contestar esta pregunta, se puede mencionar la relación que existe entre el ciclo de vida de un proyecto y los costes (gracias a la profesora Elena Barbadilla por ese conocimiento transmitido).

El ciclo de vida de un proyecto se trata del desglose lógico de lo que se debe hacer para finalizar el trabajo. En ese ciclo de vida, se establecen fases en las que se dividirá el proyecto, entregables, hitos, etc. La relación entre el proyecto y los costes, como se representa en la ilustración 2, es que, a medida que va avanzando el proyecto, los costes de los cambios van en aumento también. Las metodologías tradicionales no están preparadas para aceptar cambios una vez ya está el proyecto avanzado, supondría volver a empezar el proyecto casi desde el principio y, por tanto, aumentar los costes.

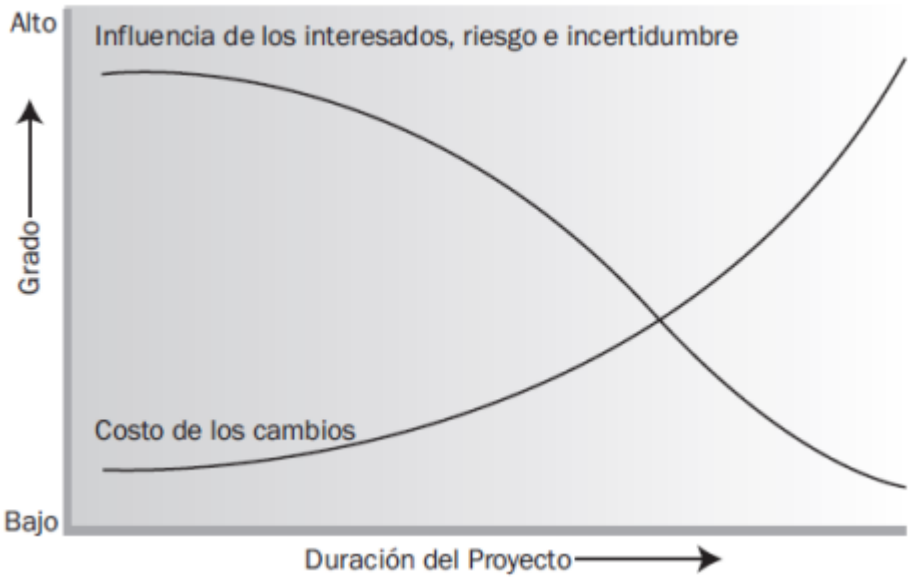


Ilustración 2. Representación del avance del coste e incertidumbre durante un proyecto (Project Management Institute, 2013)

Al principio, las metodologías tradicionales que se utilizaban para la dirección de los proyectos de desarrollo de software se basaban en la predicción para intentar conseguir los mejores resultados dentro de un calendario establecido, querían definir todos los requisitos al principio para que a lo largo del proyecto no hubiese ningún cambio y así no se incrementase el coste (Navarro Cadavid, Fernández Martínez, & Morales Vélez, 2013). Con el PMBOK también pasaba esto, pero en las últimas ediciones han ido introduciendo esta necesidad. Es decir, las metodologías que se utilizaban seguían un modelo de procesos secuencial.

Un modelo de procesos es la representación simplificada de las actividades necesarias que se dan en un proceso para lograr el objetivo. Se van a explicar brevemente los 3 modelos de procesos principales que hay con ayuda de lo expuesto por Amaro Calderón y Valverde Rebaza (2007):

- Modelo secuencial o predictivo. En estos procesos no se empieza una actividad hasta que la anterior no haya terminado, lo cual dificulta y retrasa mucho el trabajo si se quieren realizar cambios en las fases de requerimientos o análisis. Otra de las opciones que se ofrecía siguiendo este modelo era la de no aceptar cambios, el cliente no podía ver el producto hasta que este estuviese completamente terminado, no había interacción con el cliente. Una de las metodologías más conocidas que utilizaba este modelo es la metodología en cascada o *Waterfall.*

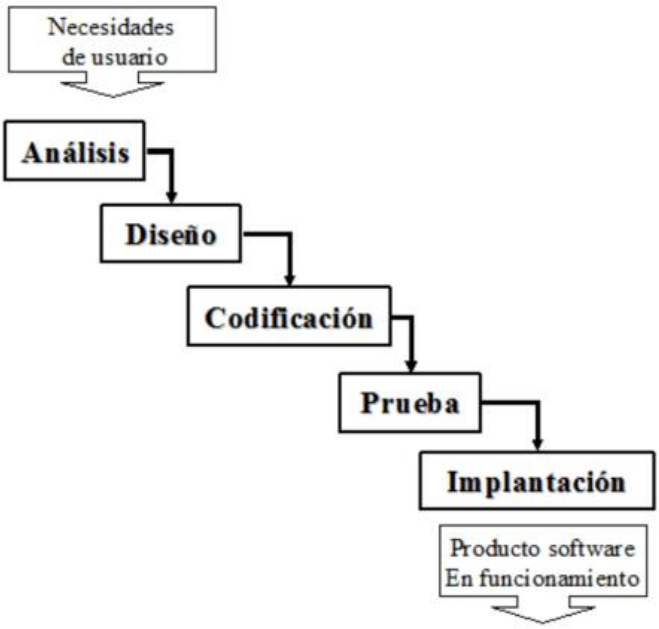


Ilustración 3. Modelo en cascada (Rojas Contreras, Esteban Villamizar, Orjuela Duarte, & CICOM, 2011)

- Modelos iterativo e incremental: rompen el ciclo de desarrollo del proceso. Se divide el proceso en fases y en cada una de ellas se hace una iteración. De cada una de estas iteraciones se obtiene un producto completo y entregable al cliente para que vaya dando su aceptación y opiniones para ir mejorando, poco a poco, el producto. El primer incremento que se obtiene es la primera versión del software y está formado solo por elementos básicos del proyecto. Luego, en los demás incrementos se irán mejorando sus funcionalidades. De esta forma, se consigue ahorrar tiempo y empezar a aceptar cambios en las distintas fases.

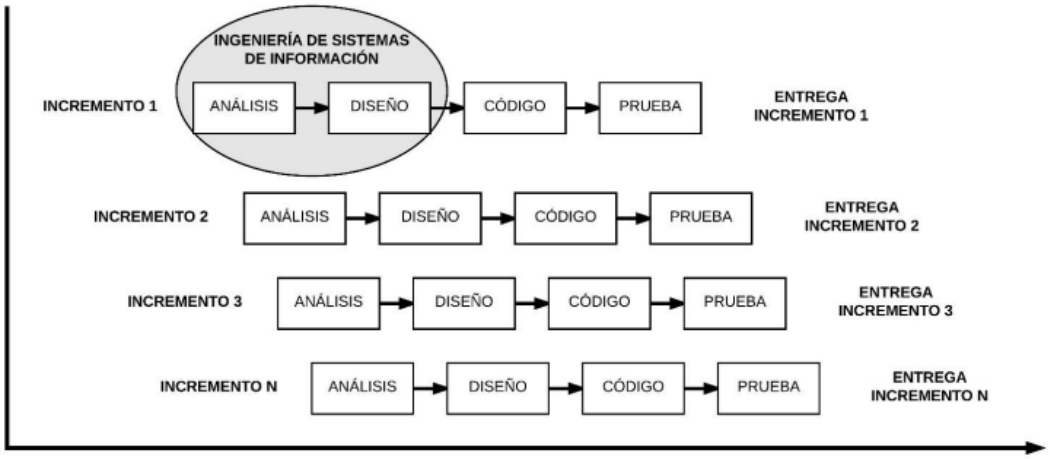


Ilustración 4. Esquema del Modelo Incremental (Pressman, 2001)

- Modelo en espiral. Mezcla el modelo iterativo en las primeras fases con el secuencial en las últimas e integra una de idea muy importante que luego adoptaron otras metodologías: el análisis de riesgos al principio del proyecto. De esta forma, este modelo planteaba la necesidad de llevar a cabo, al inicio del proyecto, prototipos desechables del producto para probar algún concepto principal que pueda suponer un riesgo crítico mediante distintas iteraciones. Después de haber conseguido un prototipo validado por el cliente, se pasaba a seguir un modelo secuencial, por lo que se plantean los mismos problemas mencionados anteriormente, no acepta cambios en las últimas fases.

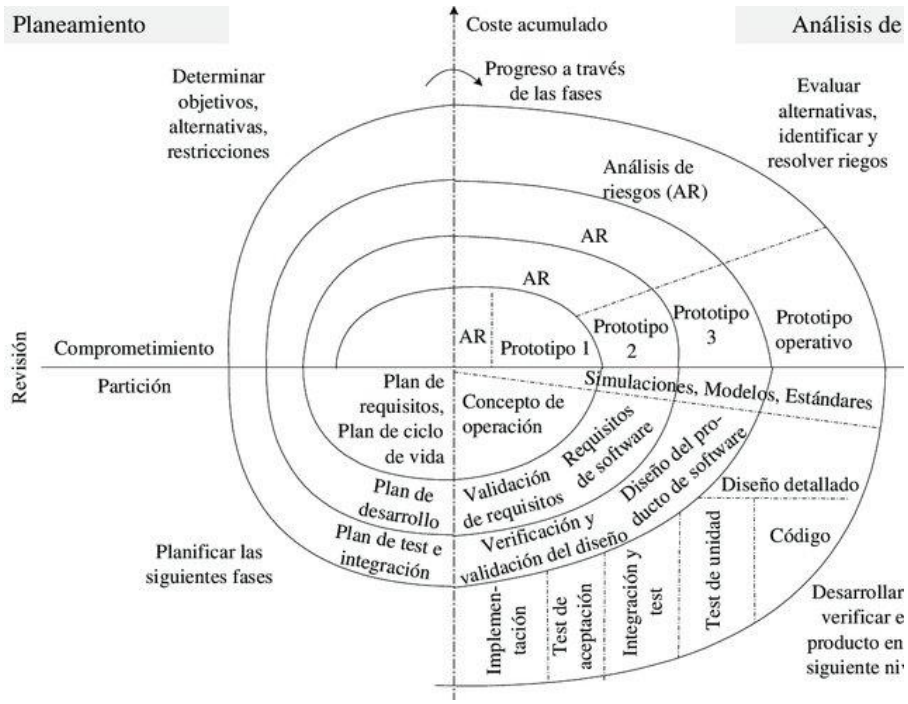


Ilustración 5. Modelo en espiral de Boehm (Boehm, 1988)

Aunque estos últimos modelos intentaban adaptarse mejor a los cambios, muchos creyeron que no eran suficientes ya que se dieron cuenta de que lo que fallaba era toda la burocracia que se tenía que llevar a cabo en las metodologías tradicionales, que se centraban, sobre todo, en las fases de planificación y análisis y en documentar absolutamente todo, constaban de muchos procedimientos burocráticos que ralentizaban el proyecto, por lo que no resultaban óptimos (Carvajal Riola, 2008).

¿Cómo interfieren las metodologías ágiles en esto? Pues sencillo, las metodologías ágiles son adaptativas, no predictivas y están orientadas a personas, no procesos. Es decir, que estas nuevas metodologías surgieron para dar solución al sobreesfuerzo que se hacía en fases que no son suficientemente productivas y para dejar de crear mucha documentación que luego no se utilizaba, es decir, rompía con toda la burocracia que ralentizaba el proyecto y se centraba en lo que realmente

generaba valor.

Un equipo ágil se centra en entregar el trabajo en incrementos pequeños, en entregas de versiones del producto que se puedan consumir para que el cliente vaya dando su visto bueno (Amaro Calderón & Valverde Rebaza, 2007). De esta forma, los requisitos, planes y resultados son evaluados de manera continua y se puede responder rápidamente a los cambios necesarios.

En esta empresa de desarrollo de software, se establecen diferentes tipos de proyectos, pero solo se van a proponer estas mejoras a aquellos que contienen desarrollo de software, ya que es en estos proyectos donde hemos visto que son necesarias las metodologías ágiles y en los que se requiere mejorar. Los 3 tipos de proyectos son los siguientes:

- Proyectos de implantación. Son aquellos en los que se implanta uno de los softwares propios de la empresa. Prácticamente nunca se suele implantar un producto totalmente estándar, sino que harán falta modificaciones para adaptarlos a la operativa de la empresa. Estos proyectos se les adjudican a un responsable de proyecto del departamento de implantación que tenga conocimientos en el sector del cliente y en el producto que se va a implantar.
- Proyectos de mejoras de proceso. Estos se dan una vez se cierra el proyecto de implantación y el cliente requiere de modificaciones en los procesos. Aquí, la figura de responsable se adjudicaría a alguien del departamento de mejoras que conozca al cliente y todos los GAPs que ya se le han hecho.
- Proyectos de continuidad. Se considera proyecto de continuidad a aquellos clientes que solicitan GAPs constantemente y que facturan entre 75 y 85 miles de euros anuales. Se les asigna un responsable de proyecto único y un equipo de desarrollo para todo el proyecto y mejoras que puedan ir saliendo. Es donde, hasta el momento, se ponen en práctica los Sprints.

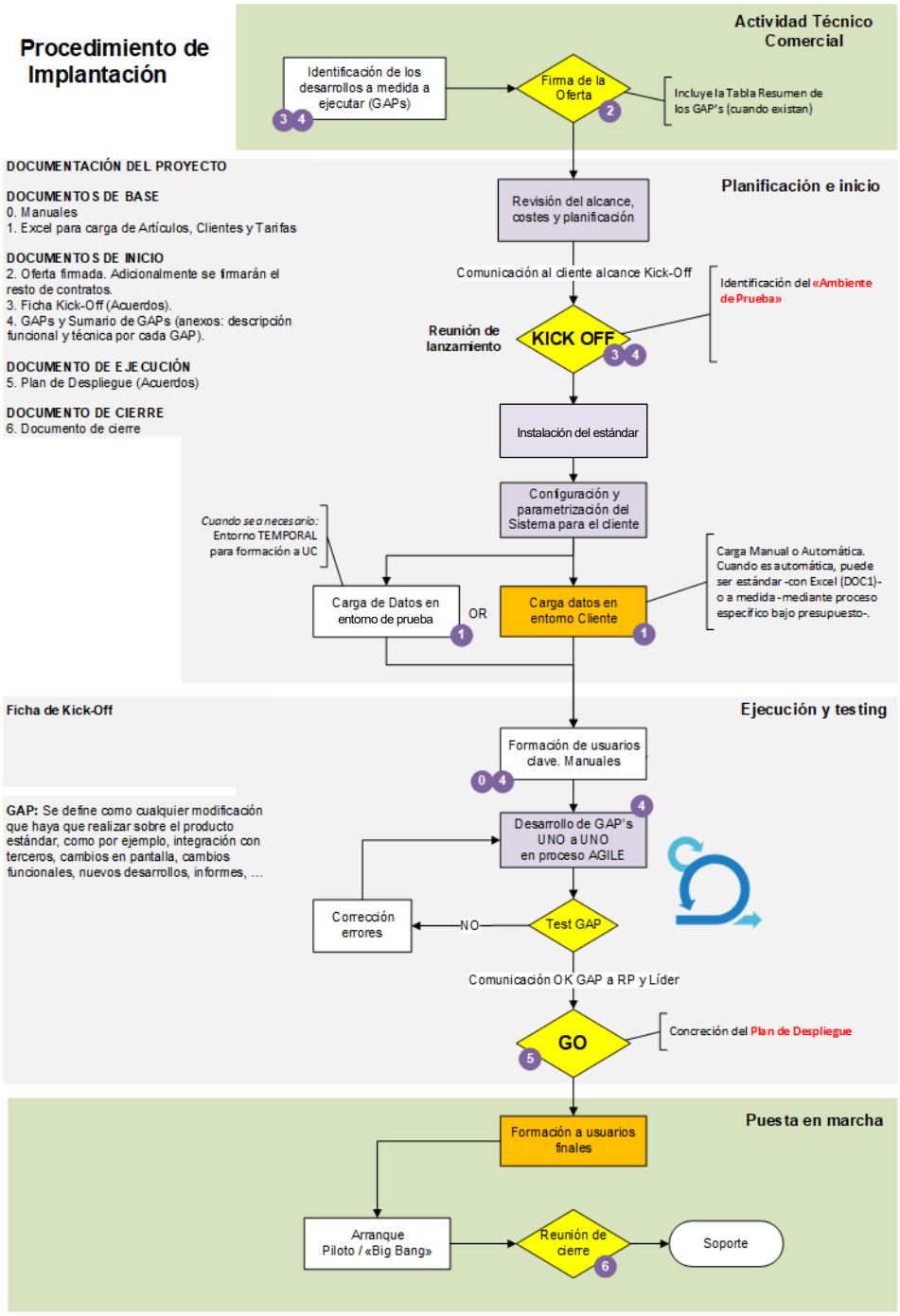


Ilustración 6. Procedimiento de implantación de un software de la empresa (propiedad de la empresa objeto de estudio)

En esta imagen, se muestra el procedimiento de implantación de una de las soluciones desarrolladas por esta empresa mencionada. Como se ve, la actividad o proceso en el que se utiliza la metodología ágil es en el desarrollo de los GAPs que se han identificado, aquel en el que se llevará a cabo un nuevo desarrollo de software.

5 METODOLOGÍA ÁGIL

Una buena retrospectiva consiste en inspeccionar y adaptar nuestra forma de trabajo.

- Alan Cyment-

En este apartado se desarrollará qué son las metodologías ágiles y por qué son tan importantes en el desarrollo del software, al igual que se expondrán algunos ejemplos de estas y se verán cuáles son las que se adaptan más a esta empresa.

5.1. Definición

Para comprender qué es una metodología ágil, primero se tendrá que exponer qué se entiende por metodología a secas en el mundo de las tecnologías informáticas. Como definen en el libro *Information systems development: methodologies, techniques and tools (3rd edition)* de 2003:

> 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 está 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 más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo (Fitzgerald & Avison, 2003).

Por lo tanto, se ve que una metodología es un conjunto de procedimientos, en los que se tienen que aplicar distintas técnicas y herramientas en las diversas fases y sub-fases establecidas para conseguir un objetivo, pero ¿qué significan estos términos en este contexto?

- Procedimiento: las normas específicas o los pasos que se tendrán que seguir para conseguir llevar a cabo el proyecto de software.
- Herramienta: instrumento que permitirá desarrollar las técnicas de la metodología.
- Fases: las divisiones de planificación que se harán dentro del proyecto.

Las metodologías ágiles son un método alternativo de gestión de proyectos, que surge en contraposición de las metodologías tradicionales para proporcionar respuestas rápidas a los cambios

dentro de los cambios basándose en el desarrollo iterativo e incremental (Rial Huerta & Montero Fernández-Vivancos, 2019). Con esto, ya se va introduciendo su definición, pero a lo largo de este apartado se irá comprendiendo mejor en qué consisten.

Estas metodologías no establecen paso a paso lo que hay que hacer. No es un guion, sino que ayudan a ir adaptando las fases u orden de ejecución en función de las situaciones que se vayan dando.

Este tipo de metodologías no es nuevo, sino que fue surgiendo conforme más desarrollos de IT se iban dando, cuando se dieron cuenta de que los ciclos de vida predictivos no eran óptimos para estos proyectos.

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.

Del artículo anterior y del desarrollado por Atlassian (2022b), se puede decir que estas metodologías presentan varias ventajas:

- Permiten dar respuesta a los cambios en las etapas de desarrollo ya que durante todo el proceso existe la posibilidad de mejorar el producto, y de esta forma también se puede mejorar la percepción del cliente hacia el mismo, debido a que es partícipe. O sea que la gestión de cambios se encuentra integrada durante todo el proceso, sin estropear la planificación, que es lo que pasaría con las metodologías tradicionales.
- Al contar con entregas continuas al cliente, se consigue tener un *feedback* continuo de todos los cambios que se realizan y se pueden introducir en la planificación futura con costes mínimos.
- Mencionando otra vez las entregas continuas, gracias a ellas, el cliente puede saber cómo va avanzando el proyecto y comprobándolo, es decir, haciéndole partícipe en todo momento. De esta forma, se mejora su satisfacción.
- Otra ventaja de estas entregas en cortos periodos de tiempo es que disminuye la dimensión de los riesgos y permite ir aprendiendo de los errores.
- Se centra en obtener una comunicación directa y fluida entre el cliente y el equipo de desarrollo. Así, se eliminan malentendidos que son el principal origen de errores.
- Solo se redactará la documentación que sea totalmente necesaria y que aporte valor. De esta forma, se optimiza el tiempo eliminando trabajos innecesarios.
- Se cuenta con la mejora continua de los procesos y del equipo de desarrollo.

5.3. Manifiesto Ágil

¿Cómo surgió este término de metodología ágil? Pues fue en 2001 cuando 17 expertos de software y algunos fundadores de algunas disciplinas de metodología de software se reunieron para discutir sobre las distintas alternativas de los procesos de desarrollo del mismo que pudiesen poner solución a los problemas planteados (crear el producto en el menor tiempo posible teniendo en cuenta los cambios que se pudiesen dar) (Highsmith , 2001).

Entre los participantes de esta reunión se encontraban Kent Beck (inventor de extreme Programming), Jeff Sutherland (cofundador de Scrum), Ken Schwaber (presidente de *Advanced Development Methds*), Jim Highsmith, Alistair Cockburn, Mike Beedle, Ward Cunningham, …

Kent Beck, declaró en ese momento "Todo en el 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".

En esto, estuvieron todos de acuerdo, el continuo cambio en este sector y la alta variabilidad en estos proyectos hacían que las metodologías tradicionales no fuesen suficientes. Por eso mismo, de esta reunión se obtuvo la terminología "Ágil" quedando reflejada en el Manifiesto Ágil de 'Desarrollo de Software' (Alianza Ágil, 2001).

Los valores que se definen en este mencionado manifiesto no se enfocan en prácticas, metodologías o procedimientos de trabajo, sino defienden un cambio de mentalidad, una nueva cultura organizativa basada en cuatro pilares:

1. Favorecer a los individuos y sus iteraciones en vez de fomentar los procesos y herramientas.
2. Centrarse en obtener un software que funcione en lugar de documentación exhaustiva que luego no se va a utilizar.
3. Beneficiar la colaboración con el cliente sobre la negociación contractual.
4. Ser reactivo ante el cambio en lugar de seguir un plan.

En definitiva, establece que los equipos de trabajo que quieran seguir un enfoque ágil tienen que tener el foco en una comunicación abierta, la colaboración, la adaptación y la confianza entre los miembros que lo conforman.

Por otro lado, este manifiesto consta de 12 principios que debe cumplir el software ágil: en los dos primeros se define la idea de agilidad y en los demás, se desarrollan las consideraciones que hay que tener en cuenta a la hora del desarrollo en cuanto al equipo de programadores, su organización y cómo conseguir los objetivos. Citando textualmente el manifiesto (Alianza Ágil, 2001), los principios son

los siguientes:

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Esta agrupación declara que en la metodología ágil se acepta el modelado, la documentación y la planificación, pero solo cuando sea de total utilidad. Ya se mencionó antes que uno de los inconvenientes de las metodologías tradicionales era la burocracia, toda la documentación y los procedimientos que se tenían que llevar a cabo obligatoriamente, aunque luego no se utilizasen para nada. Esto lo que hacía era ralentizar los proyectos, destinar más tiempo a la tarea de la planificación y documentación, tiempo que se quita de lo que realmente aporta valor. La metodología ágil lo que busca, como bien indica su nombre, es agilizar los procesos, es decir, optimizar. Para ello, reflexiona sobre qué documentos y procedimientos son totalmente necesarios y ayudarán a mejorar el proyecto, eliminando todos los demás.

Como se puede observar, este manifiesto no dice qué metodologías son ágiles y cuáles no ni tampoco menciona cuál usar en cada caso. Solo indica una serie de valores y principios para que luego cada uno los aplique, ya sea siguiendo las instrucciones de una metodología ágil a rajatabla o combinando

elementos de metodologías distintas (Atlassian, 2022b).

5.4. Tipos de Metodologías ágiles

Algunas de las distintas metodologías que están referenciadas como ágiles son:

- Adaptative Software Development
- Agile Modeling
- Agile Model Driven Development
- Agile Project Management
- Agile Unified Process
- Crystal Methods
- Dynamic Systems development methods
- Feature driven development
- Internet Speed Development
- Lean software development
- Pragmatic programming
- Scrum
- Test Driven Development
- XBreed
- Extreme Programming
- Win Win Spiral.
- Evolutionary Project Management.
- Story cards driven development.
- Open Unified Process
- Scrumban (mezcla de Scrum y Kanban)
- G300
- 6D-BUM
- PMI Agile
- Kanban
- Agile RUP

Como se mencionaba antes, no son las únicas que existen, ya que cada equipo o empresa suele crear la suya propia utilizando los procedimientos que considere adecuados de cada metodología ya definida y desarrollando otros que les sean más convenientes, siempre teniendo en cuenta que no se pueden

perder de vista los valores y principios establecidos en el manifiesto si realmente se quiere que sea un marco de trabajo ágil.

Como es obvio, no se pueden analizar todas las metodologías ágiles que hay en este documento, por lo que se van a desarrollar aquellas que tienen más documentación y que se piensa que pueden ser de mayor utilidad en este sector y para esta empresa, y comentar algunas en las que destacan algunas cualidades. Serán las siguientes:

1. Agile Project Management (que surge del ASD).
2. Crystal Methods.
3. Scrum.
4. Extreme Programming.
5. Kanban.
6. Scrumban

5.4.1. Agile Project Management (APM)

Esta metodología surge de otra mencionada, Adaptive Software Development. Fue desarrollada por primera vez por Jim Highsmith en 1999 en su libro "*Adaptive Software Development: A Collaborative Approach to Managing Complex System*" y, un año más tarde, en otro libro suyo con el propio nombre de la metodología: *Agile Project Management*. Estas son las dos principales fuentes que se van a utilizar para exponer esta metodología.

En plena búsqueda de metodologías que satisficiesen las necesidades de aceptar cambios durante todo el proyecto, Highsmith (1999) defiende que el proceso de desarrollo debe ser acorde a los objetivos del proyecto, de modo que, si los objetivos pueden ser definidos en su totalidad al principio del proyecto, se podría usar un proceso prescriptivo. Sin embargo, si los objetivos son innovadores, entonces el marco de trabajo tendría que ser ágil, flexible y fácil de adaptar. Es decir, proponía una nueva perspectiva de gestión de proyectos, que un par de años más tarde se recogería en el manifiesto ágil.

El modelo APM, como se describe en *Agile Project Management* (Highsmith, 2000), tiene una estructura que se basa en cinco fases: Previsión, Especulación, Exploración, Adaptación y Cierre, como se puede observar en la figura.

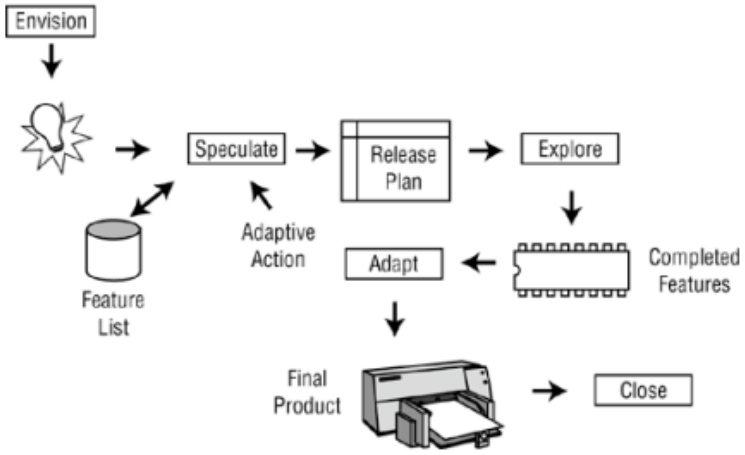


Ilustración 7. Fases del modelo APM (Carvajal Riola, 2008)

- Previsión. Como bien indica el nombre, en esta primera fase, hay que determinar la visión del producto, así como definir el alcance del proyecto, quién conformará el equipo de trabajo y cómo se trabajará de forma conjunta.
- Especulación. En esta fase, se hace un plan de entregas que se base en el conjunto de funcionalidades del producto y no en actividades a llevar a cabo. Esta etapa reemplazaría a la de planificación tradicional ya que esta última tiene una connotación que hace referencia a predecir con cierta certeza, mientras que, si se especula, se está dando por hecho que lo que va a pasar es incierto.
- Exploración. Durante esta fase, se obtienen las funcionalidades requeridas, se prueban y se validan o desechan por el cliente. Para que se alcance este objetivo, el equipo de trabajo debe estar auto-organizado y deben des auto-disciplinados. Su nombre se debe a que este proceso, que son iteraciones de entregas, es exploratorio al basarse en especulaciones, no en hechos.
- Adaptación. El objetivo de esta fase es revisar los resultados de la fase anterior, el estado y el rendimiento, pudiendo hacer algunas modificaciones si fuera necesario.
- Cierre. Fase final en la que se termina el proyecto aprendiendo de las experiencias que este ha generado. Lo importante aquí no es solo cerrar el proyecto, sino que el conocimiento que se ha ido construyendo a lo largo del proceso se transmita y sirva para algo.

También, se tienen que mencionar los roles que plantea esta metodología. Para hacerle mención al nombre de esta metodología, Highsmith defiende que los roles que se adopten también tienen que ser flexibles y adaptativos debido a las diversas interpretaciones que se les puedan dar. Por esto mismo, da una breve descripción de estos mismos roles, para que luego cada persona los adapte a sí misma y a sus capacidades.

- Patrocinador ejecutivo: aquella persona o grupo de personas que lideran el proyecto y son las encargadas de tomar las decisiones clave acerca de los objetivos y restricciones del proyecto.
- Gestor de proyectos o *Project manager*: persona encargada de liderar el equipo que entrega el producto.
- Gestor del producto o *Product Manager*: responsable de liderar el equipo que se encarga de especificar qué resultados hay que entregar.
- Ingeniero jefe: encargado de guiar al equipo en los aspectos técnicos del proyecto.
- Gestores: grupo de personas (ya sean de la organización que se encarga del proyecto o de empresas que tengan relación con este, *stakeholders*) que puedan participar en la toma de decisiones del proyecto.
- Equipo del cliente: son individuos destinados por el cliente para encargarse de plantear y priorizar las características que tiene que cumplir el producto que quieren.
- Equipo del proyecto: son los individuos seleccionados por la empresa contratada que se encargarán de realizar el proyecto.
- Proveedores: empresas externas que proveen de servicios o productos necesarios para el proyecto.
- Gobierno.

Otra de las cosas a destacar que se menciona en este libro también es que Highsmith presenta algunas prácticas ágiles que se pueden llevar a cabo en las distintas fases que se han mencionado anteriormente (y que también se pueden adaptar para los marcos de trabajo que se quieran crear) (Highsmith, 2000). Algunas de ellas son:

- Arquitectura del producto. Esta práctica se realiza al inicio del proyecto y consiste en exponer las partes del proyecto que ocupan más recursos para ver dónde hay más criticidad.
- Listado de funcionalidades del producto. El objetivo de esta práctica es realizar una lista evolutiva de los requisitos del producto.
- Tarjetas de especificación de rendimiento. Son documentos que recogen las operaciones primordiales y los requisitos del producto.
- Reuniones diarias del equipo del proyecto para coordinar las actividades y ver el progreso.
- Coaching y desarrollo.

5.4.2. Crystal Methods

Desarrollada por Alistair Cockburn para IBI. Defiende la idea de que las características de un proyecto varían dependiendo del número de personas que compongan el equipo encargado de llevarlo a cabo y

del nivel de complejidad del mismo. Es decir, un equipo formado por pocas personas puede realizar el proyecto sin necesidad de redactar demasiados documentos de información, cosa que no podrá pasar en los equipos numerosos porque no se tiene tanta comunicación (Mrsic, 2017).

Para poder alcanzar todo tipo de proyectos, Alistair propone distintos métodos en función del tamaño del equipo:

- Claro: equipos formados por ocho personas o menos.
- Amarillo: entre 10 y 20 personas.
- Naranja: entre 20 y 50 personas.
- Rojo: entre 50 y 100 personas.
- Oscuro: aquellos que constan de más de 100 personas.

Las características generales de estos métodos son:

- Entregas frecuentes del código probado.
- Mejora continua de los procesos del equipo.
- Comunicación osmótica entre los miembros del equipo que trabajan en un mismo lugar. Esto consiste en que todos escuchen las conversaciones en las sesiones de trabajo, aunque estas conversaciones no les involucre directamente. Es decir, se basa en la compartición de la información. El problema de esto es que es muy complicado llevarlo a cabo en equipos grandes o que trabajen en distintas localizaciones.
- Seguridad total a la hora de comunicarse con el equipo, ya sea exponiendo nuevas ideas, posibles situaciones problemáticas o soluciones para estas. El equipo no debe juzgar, hay que crear un buen ambiente de trabajo.
- Equipo centrado y enfocado en su trabajo propio.

La característica que más destacable parece es la de la comunicación osmótica, y es una de las razones por las que me he decidido a mencionar un poco esta metodología. Puede que muchas metodologías ágiles compartan esta idea, sobre todo Scrum con la reunión diaria, pero los métodos crystal son los que más importancia le da. No es solo compartir información de la situación del proyecto, es compartir conocimiento durante todo su desarrollo (Mrsic, 2017).

Me ha llamado la atención porque es algo que se realiza en el día a día, que a mí me gusta particularmente, y que no se le pone nombre. Es como realmente se empieza a aprender, escuchando a los demás sobre los temas que dominan y las dudas que les surgen. Una reunión con un cliente, con el equipo de trabajo, una simple charla entre compañeros, … ¡Aunque no tenga nada que ver con tu área de trabajo! Esto es una de las cosas que me gustaron más en mi formación y desarrollo en la

empresa: todas estas reuniones en las que no participaba de forma directa, pero que me hacían aprender muchísimo y de las que, poco a poco, iba obteniendo más conocimiento.

5.4.3. Extreme Programming

El eXtreme Programing (XP) es una de las metodologías ágiles más populares y usadas entre la comunidad de desarrolladores.

Esta práctica fue definida en 1999 por Kent Beck con el mismo objetivo que las demás metodologías ágiles, el de poder adaptarse a los cambios que se diesen.

Los roles que propone esta metodología según Wells (2009) son:

- Cliente. Se encargará de definir las funcionalidades que requiere para el producto con sus respectivas prioridades.
- Desarrollador. Responsable de realizar todo el desarrollo necesario y llevar a cabo las pruebas de funcionamiento.
- *Coach.* Es el intermediario entre el cliente y el equipo, resuelve conflictos y problemas además de ser el encargado de controlar el método y su cumplimiento.
- *Tracker.* Responsable de evaluar el avance y la calidad de cada ciclo de desarrollo. Debe comunicar al equipo y al cliente sus conclusiones y sus propuestas de mejora.

Es muy similar a la metodología Scrum, pero esta es más prescriptiva y obstinada. Scrum se centra mayormente en la organización del equipo de trabajo, mientras que XP lo hace en el código. Por esto mismo, esta última es una de las más usadas en el sector de desarrollo de software.

XP consta de 5 valores: sencillez, comunicación, retroalimentación, coraje y respeto. Nada fuera de lo que ya se haya mencionado en otras metodologías (Wells, 2009).

Algunas prácticas que propone esta metodología para un mejor desarrollo del código son las siguientes (Olic, 2017):

- Programación en parejas: todo el código es desarrollado por dos personas que trabajen en el mismo puesto. De esta forma, es revisado y discutido en todo momento lo que aumentará la calidad y no conllevará una pérdida de eficiencia (aunque pueda parecer lo contrario al tener a dos personas trabajando en lo mismo).
- Propiedad colectiva del código: todos los miembros del equipo tienen acceso al código y pueden arreglar fallos. Esto obliga a que el código sea limpio y pueda ser leído fácilmente, lo recomendable es adoptar un estándar de codificación.

- Equipos multifuncionales: los equipos de trabajo que siguen una metodología XP suelen ser pequeños (menos de 10 personas), por lo que deben ser capaces de encargarse distintas tareas, deben tener múltiples habilidades.
- Compilación en 10 minutos: Se debería poder ejecutar las pruebas en este tiempo. Si no es así, no es óptimo.

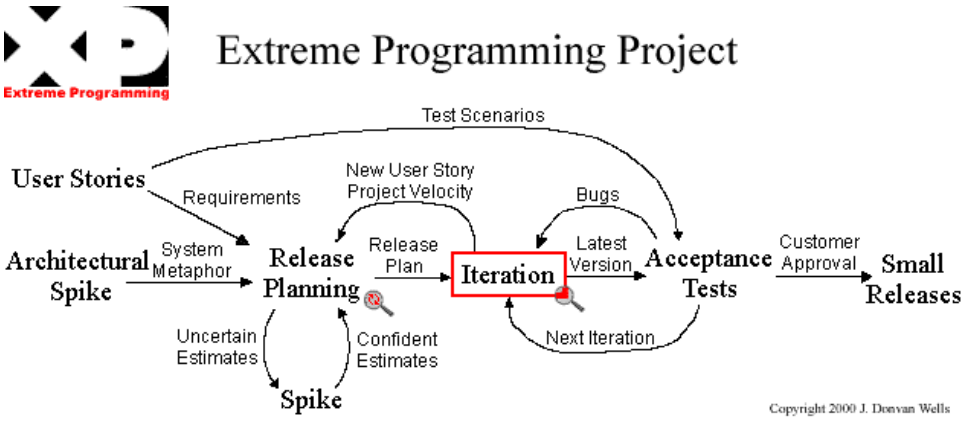


Ilustración 8. Flow Chart de un proyecto que sigue una metodología de XP (Wells, 2000)

Una de las desventajas que tiene este marco de trabajo es que hay muchas reglas que se deben cumplir para que se obtenga el resultado óptimo. Algunas de estas reglas son (Wells, 1999):

- El cliente siempre estará disponible. El cliente tiene un papel fundamental en el proyecto y será necesario durante gran parte de este. Esto es la teoría, pero, en la práctica, el cliente tiene otras muchas cosas que hacer y no siempre va a tener tiempo para dedicarle a esto y se puede retrasar el proyecto.
- Todo el código estará escrito según los estándares definidos. Obliga a que todos los desarrolladores trabajen el código igual, así como también garantiza que todos puedan leerlo fácilmente.
- Todo el código programado por parejas. Ya se mencionaba antes que obliga a tener a dos personas trabajando en lo mismo.
- Utilizar tarjetas CRC (Clase, Responsabilidades y Colaboración) para diseñar el sistema de forma grupal.

5.4.4. Scrum

Este marco de trabajo es muy utilizado por los equipos de desarrollo de software ya que se centra en la mejora continua. Es decir, se basa en el empirismo y el pensamiento lean. Permite aprender de las experiencias y reflexionar sobre ellas para ver qué se puede mejorar, aceptando los cambios necesarios de manera natural ya que reconoce que el equipo no lo sabe todo al comienzo del proyecto (Drumond,

2022).

La información que se va a exponer a continuación es una recopilación de la guía de Scrum (Schwaber & Sutherland, 2020) y los artículos de Atlasian sobe este mismo tema. Si se quiere aplicar o saber más sobre este marco de trabajo, se pueden consultar dichos enlaces que están en la bibliografía.

En 1991 fue cuando se le dio nombre a esta metodología, pero no fue hasta 1995 que Ken Schwaber (antes mencionado) y Mike Beedle empezaron a describir y definir los fundamentos de esta misma en el libro "Agile Software Development With Scrum" que se publicó en 2001, y en 2011 fue cuando se publicó la primera versión de la "Scrum Guide".

El nombre de Scrum significa "melé", que es una formación característica del rugby en la que los miembros del equipo construyen una formación humana para hacerse con la pelota. Es decir, el nombre hace referencia a uno de los aspectos fundamentales de este marco, que es el hecho de basarse en la cohesión del equipo por encima del individualismo.

Para comenzar, hay que decir que Scrum propone una serie de actividades, roles y artefactos para llevar a cabo este proceso, todos centrados en crear un buen flujo de comunicación. El elemento clave en el que se basa esta metodología es el Sprint. Todo el proceso se llevará a cabo mediante Sprints. De esta forma, se puede decir que se entiende como Sprint un evento de duración concreta (suele ser un mes o menos) en el que se realizará un determinado trabajo. Hasta que no se da por terminado un Sprint, no se comienza con el siguiente (Schwaber & Sutherland, 2020).

Se va a comenzar mencionando cómo se debe conformar un equipo Scrum. Este está formado por un número pequeño de personas (10 o menos), donde no hay jerarquías y todos los miembros deben tener todas las habilidades necesarias para crear valor en cada Sprint. Todo equipo se autogestiona, de forma que se decide internamente quién hace qué, cuándo y cómo (Drumond, 2022).

Se diferencian tres roles dentro de cada equipo Scrum:

- Desarrolladores. Son los encargados de crear el backlog de Sprint, de adherirse a la definición de "Hecho", de adaptar su plan diario para conseguir el objetivo y de responsabilizarse unos a otros como profesionales.
- Propietario del producto (*Product owner*). Responsable de maximizar el valor del producto. Para ello, debe asegurarse de que se desarrolle y comunique claramente el objetivo del producto, además de que se definan, comuniquen y ordenen los elementos del backlog del producto, y que este backlog sea transparente, visible y comprensible. Las decisiones que tome este propietario deben ser respetadas por toda la organización, por lo que, si se quiere cambiar algo respecto al backlog, tiene que darle el visto bueno antes esta persona.

- Scrum master. Encargado de que la organización y el equipo comprendan la teoría y práctica de Scrum. Entre todas las funciones que tiene este rol están las de eliminar todos aquellos impedimentos al progreso del equipo, ayudar a encontrar técnicas para definir correctamente la meta del producto y facilitar, cuando se solicite o requiera, la colaboración de las partes interesadas.

Ahora mismo, en la empresa objeto de estudio, los roles que se definen son:

- Responsable del proyecto (RP). Encargado de la ejecución del proyecto en calidad, plazos y margen.
- Director de proyecto (DP). Se encuentra por encima del RP y es el responsable último del proyecto de implantación, de planificar los recursos necesarios y de velar por la calidad y plazo de las implantaciones, así como de prever los riesgos antes de que se produzcan. Esta figura será necesaria solo en proyectos que lo exijan, ya sea por volumen o complejidad.
- *Key account manager* (KAM). Es el comercial que ha cerrado la operación y debe participar en la primera reunión de lanzamiento del proyecto (Kick-Off), en la reunión del GO y en la de Cierre. En función de la tipología de proyecto podrá hacer de DP. Es responsabilidad del KAM la identificación de los GAP's en fase comercial, para lo cual podrá apoyarse en Desarrollo.
- Programadores (P). Serán las personas encargadas de hacer los desarrollos de los GAP's

Como se puede observar, en el método Scrum no se define estos roles de responsable y director de proyecto, esta noción no existe. Esto se debe a que el equipo se auto-organiza siempre que sea posible o, si no es el caso, tiene una gran autonomía, por lo que algunas de las funciones tradicionales asociadas al jefe de proyecto desaparecerían. Además, se puede ver que otras de estas funciones recaen en la figura del Product Owner.

El éxito de esta metodología se basa en que el equipo se vuelva más competente en cinco valores: compromiso, enfoque, apertura, respeto y coraje. Concretamente, la guía lo indica de la siguiente forma:

> El Equipo Scrum se compromete a lograr sus objetivos y a apoyarse mutuamente. Su enfoque principal está en el trabajo del Sprint para lograr el mejor progreso posible hacia estos objetivos. El Equipo Scrum y sus partes interesadas están abiertos sobre el trabajo y los desafíos. Los miembros del Equipo Scrum se respetan entre sí como personas capaces e independientes, y son respetados como tales por las personas con las que trabajan. Los miembros del Equipo Scrum tienen el coraje de hacer lo correcto, de trabajar en problemas difíciles (Schwaber & Sutherland, 2020).

Según esta metodología, las decisiones importantes se basan en el estado de tres artefactos formales que representan trabajo o valor para ofrecer transparencia, inspección y adaptación, siendo estos tres últimos los pilares empíricos de Scrum (Schwaber & Sutherland, 2020). Los artefactos son:

- Backlog del producto (*Product Backlog*). Se trata de una la lista dinámica principal de todo el trabajo que se debe realizar para crear el producto ordenado por prioridad. En ella deben de recogerse todas las funciones, requisitos, mejoras y correcciones respecto al producto, teniendo en cuenta que hay que estar constantemente revisándolo por si hay que cambiar las prioridades o algún elemento ha quedado obsoleto. Los elementos de esta lista tendrán que pasar por un evento denominado "planificación de Sprint" (*Planning Spring*) para que les den el visto bueno y ser parte del siguiente artefacto (Radigan, 2011).
- Backlog de Sprint (*Sprint Backlog*): es la lista de elementos, historias de usuario o correcciones de errores que los desarrolladores tienen que llevar a cabo durante el Sprint. Es decir, se trata de una imagen visible y en tiempo real del trabajo que se planea realizar en el Sprint para lograr el objetivo (*Sprint Goal*). Esta lista hay que ir actualizándola a medida que se avanza en el Sprint y se va aprendiendo más. Con esto se quiere decir que esta lista es flexible, puede evolucionar durante el Sprint, pero no se debe quedar en el olvido el objetivo de este. Otro dato por mencionar en este punto es que, a la hora de redactar o actualizar esta lista, debe hacerse con el suficiente detalle para que luego se pueda inspeccionar su progreso en el *Daily Scrum* (Rehkopf, 2022b). Por último, hay que decir que el backlog de Sprint puede presentarse, controlarse y actualizarse utilizando distintos soportes como pueden ser hojas de cálculo, pizarras, tableros o soportes digitales.
- Incremento: Es el producto final utilizable del Sprint, un peldaño concreto hacia el objetivo del producto. Se le puede denominar de esta forma, hito, epic lanzado, objetivo del Sprint o como la definición del equipo de "Hecho". Cada incremento de cada uno de los Sprints se suma a todos los incrementos anteriores y se verifica que funcionen juntos. Esta suma se presenta en la revisión del Sprint (*Review Sprint*).

Se mencionaba antes, en el apartado de incremento, que el equipo tenía que establecer la definición de "Hecho", "Finalizado" o "Terminado", según lo quieran denominar. Con esto se hace referencia a una descripción formal de la situación del Incremento cuando cumple con las medidas de calidad necesarias para el producto (Drumond, 2022). Esto se hace para crear transparencia, para que todos puedan saber qué trabajo se ha completado como parte del Incremento. Si un elemento no cumple con la definición de "Hecho", no se pude lanzar ni presentar en la fase de "Revisión del Sprint".

Como se comentaba al principio de esta metodología, el trabajo se fragmenta en Sprints, lo que nos

permite mejorar el enfoque y la consistencia del equipo si se trabaja a un ritmo adecuado. Dentro de cada Sprint, se tendrán todos los eventos que se han ido mencionando como son la planificación de Sprint, Scrums diarios, Revisión de Sprint y Retrospectiva de Sprint. Se puede ver esto reflejado en la ilustración 9 junto con los artefactos descritos anteriormente. De esta forma, se ve cómo se organiza un Sprint.

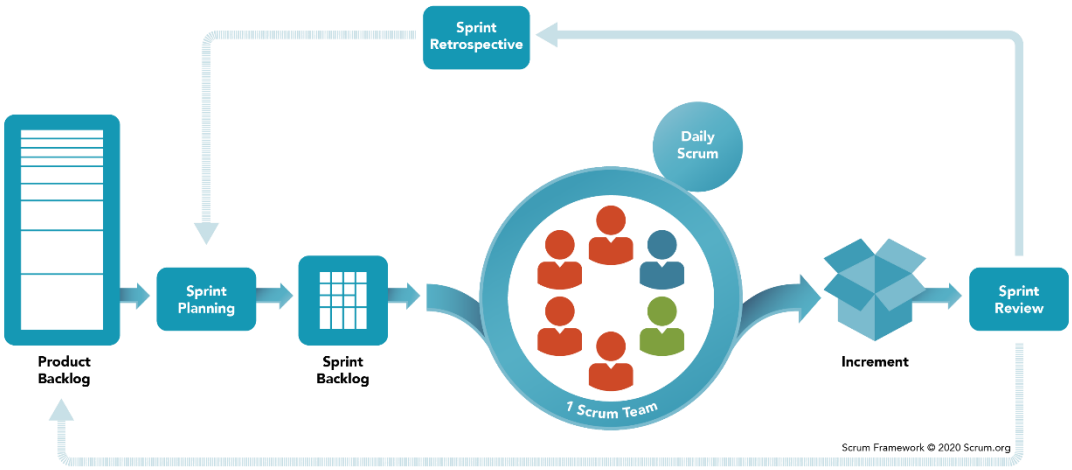


Ilustración 9. Ciclo de un proyecto Scrum (Scrum.org, 2022)

Como se ve en la figura, el ciclo de vida de un proyecto Scrum es el siguiente (Subra & Vannieuwenhuyze, 2018):

1. El Product Owner se encargará de definir las tareas o historias que conformarán el proyecto y las colocará en el Product Backlog.
2. En el siguiente paso, el Product Owner tendrá que priorizar los elementos que ha definido y ordenará el Product Backlog en concordancia.
3. El equipo Scrum se juntará en la reunión de planificación del Sprint (que durará un máximo de ocho horas), para diseñar el trabajo que se llevará a cabo durante el Sprint y que se organizará en el Sprint Backlog. En esta planificación habrá que decidir cuál será el objetivo principal de este Sprint, qué elementos del Backlog del Producto se pueden realizar y se dividirán en tareas para el equipo de desarrolladores.
4. A partir de este momento, el Sprint podrá iniciar con una iteración de 2, 3 o 4 semanas.
5. El siguiente paso sería el Scrum diario. Esta mini reunión de 15 minutos la hacen los desarrolladores todos los días a la misma hora para analizar el progreso hacia el objetivo, ver si necesitan la ayuda de alguien en alguna tarea pendiente y adaptar el Sprint Backlog si lo ven conveniente, lo que permite un mayor enfoque.

6. Una vez se da por "Terminado" un Sprint, se pasa a la fase de revisión donde se analiza el resultado obtenido y se determinan futuras adaptaciones.
7. Por último, en la fase de Retrospectiva de Sprint (*Sprint Retrospective*) se planifica cómo aumentar la calidad y la eficacia del equipo analizando su funcionamiento a lo largo del Sprint.

Con este evento, se finalizaría el Sprint y se volvería a empezar el ciclo hasta que se finalizase el proyecto.

Una vez desarrollada esta metodología, solo queda resumir algunas de sus ventajas y desventajas. Como bien presenta Drumond (2022) en uno de los artículos de Alliance, estas ventajas y desventajas son las siguientes:

Ventajas:

- Marco de trabajo sencillo, semiprescriptivo y fácil de entender.
- División del trabajo en historias de usuario que permiten un mayor manejo.
- Al realizar lanzamientos rápidos y continuos con los Sprints, se consigue mantener motivado al equipo y una mayor relación de confianza con el usuario.
- Transparencia

Desventajas:

- No permite cambios durante la iteración.
- El tablero Scrum (backlog del Sprint) se cambia en cada iteración.
- Hay que hacer pruebas con frecuencia porque se deben hacer entregas de funcionalidades terminadas correctamente cada 2, 3 o 4 semanas.

Si se quiere obtener una formación y certificación en este método, se puede hacer a través de Scrum Alliance (http://www.Scrumalliance.org/) o mediante la propia organización de Scrum creada por Ken Schwaber (http://Scrum.org)

5.4.5. Kanban

Esta es otra de las metodologías favoritas de las empresas que se dedican al desarrollo de software y DevOps. La información que se va a exponer ahora es contrastada entre el artículo realizado por Radigan (2022) y la guía expuesta por Kanban Tool (2009).

Este marco de trabajo tiene su inicio en la historia a finales de los años 40, cuando Toyota introdujo el conocido *Just In Time* (JIT) para ajustar los niveles de existencia al consumo real. Los trabajadores de la fábrica japonesa comunicaban los niveles de capacidad pasándose una tarjeta o "Kanban", al igual

que hacían para pedir la cantidad exacta de materiales al almacén o a los proveedores cuando se terminaban. El concepto Kanban viene del japonés y significa "tarjeta visual" (kan es "visual" y ban es "tarjeta").

En la actualidad, se ha adaptado este método para ajustar la cantidad de trabajo en curso (o más reconocido con el término inglés, *Work In Process* (WIP)) del equipo de trabajo. Para ello, se utiliza un tablero en el que se representan, en forma de tarjetas, todas las tareas que están pendientes de realizar, las que están en proceso y las que se han finalizado para optimizar el flujo de trabajo entre los miembros del equipo de forma que puedan controlar todo el progreso del proyecto de manera muy visual.

Para que esta metodología funcione con éxito, se necesita comunicación en todo momento sobre la capacidad del equipo y total transparencia con respecto al trabajo. Por ello, este tablero debe ser la única fuente fiable de información para el equipo.

Los elementos de trabajo se representan visualmente en un tablero de Kanban, lo que permite a los miembros del equipo ver el estado de cada uno en cualquier momento. Los equipos tienen así opciones de planificación más flexibles, resultados más rápidos, un enfoque más claro y transparencia a lo largo del ciclo de desarrollo.

Los objetivos y fundamentos de Kanban son los siguientes (Kanban Tool, 2022a):

- Visualizar el flujo de trabajo. Al igual que en la metodología Scrum antes explicada, esta también propone descomponer el proyecto en tareas más pequeñas o historias de usuario y establecer el flujo de trabajo en el tablero Kanban. Para esto último, tenemos que definir qué columnas vamos a poner en el tablero, es decir, las fases por las que van a pasar las tareas. Antes, se mencionaba que se tenían por lo general tres fases "Pendiente", "En proceso" y "Hecho" pero se pueden añadir subetapas en la fase de progreso si se piensa que aporta valor y agiliza el proceso, quedando algo parecido a lo que se ve en la figura.

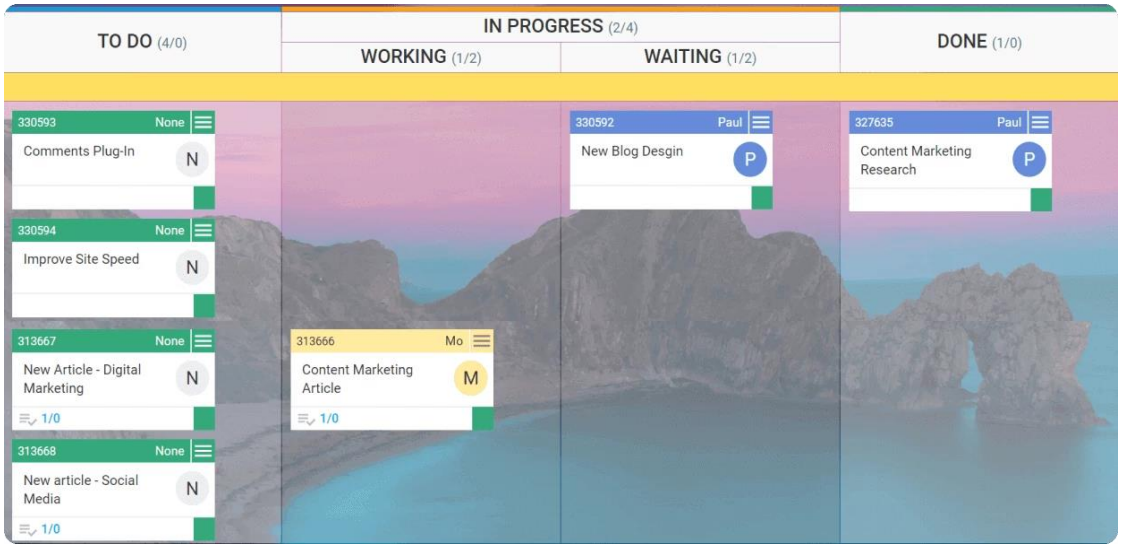


Ilustración 10. Ejemplo de tablero de Kanban (Kanbanize, 2022)

Por ejemplo, los estados que se han definido en esta empresa son los siguientes:

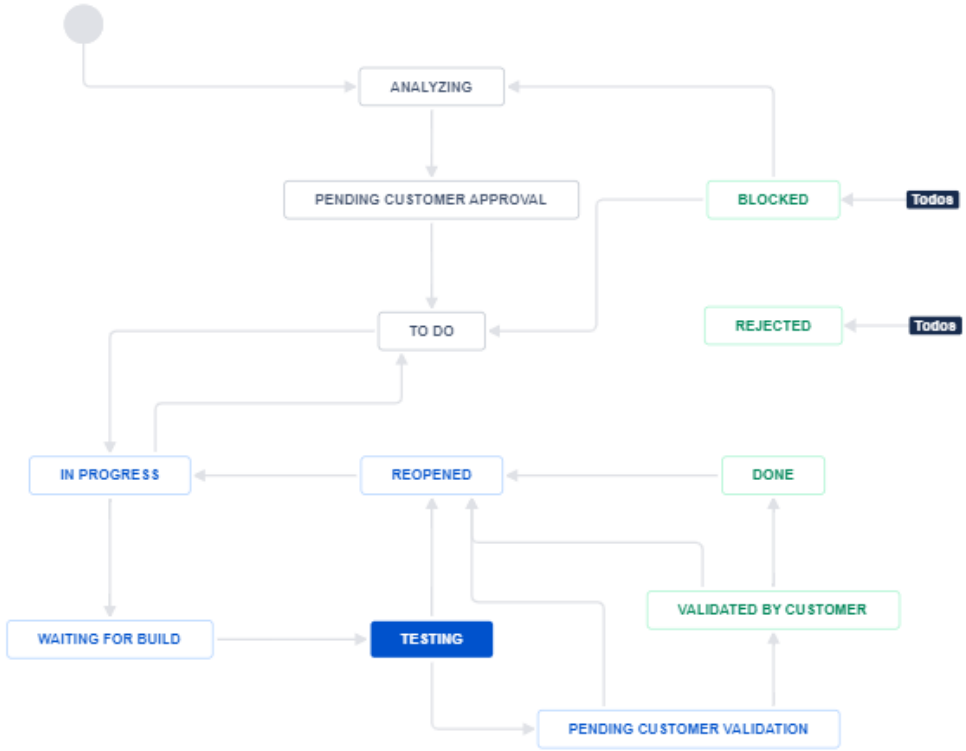


Ilustración 11. Estados de la empresa (Cedido por la empresa)

Otra de las cuestiones que hay que definir aquí, es establecer los tipos de trabajos que se pueden tener en el tablero (si son tareas, análisis, informes, solicitudes, …) y qué color de tarjeta se va a asociar a cada uno de ellos para poder llevar un seguimiento más visual.

- Determinar el límite del trabajo en curso. Algo fundamental de este marco de trabajo es su hincapié en que hay que identificar el número máximo de tareas que se pueden estar llevando a cabo en cada fase. Además, para que un trabajador comience una nueva tarea, debe haber terminado antes la otra. De esta forma, permite al equipo centrarse en terminar el trabajo y mejorar los procesos (Andersen, 2011).
- Administrar el flujo de trabajo. Al igual que en Scrum, se propone la idea de hacer breves reuniones diarias donde el equipo ve el tablero y analizan el flujo. Con estas reflexiones, se puede ver si los elementos se bloquean en ciertas zonas (cuellos de botella) y se puede plantear qué hacer para que no ocurra más. En este apartado, son de mucha ayuda los diagramas de fujo acumulativo y las gráficas de los tiempos de ciclo y de espera (o gráfico de control).

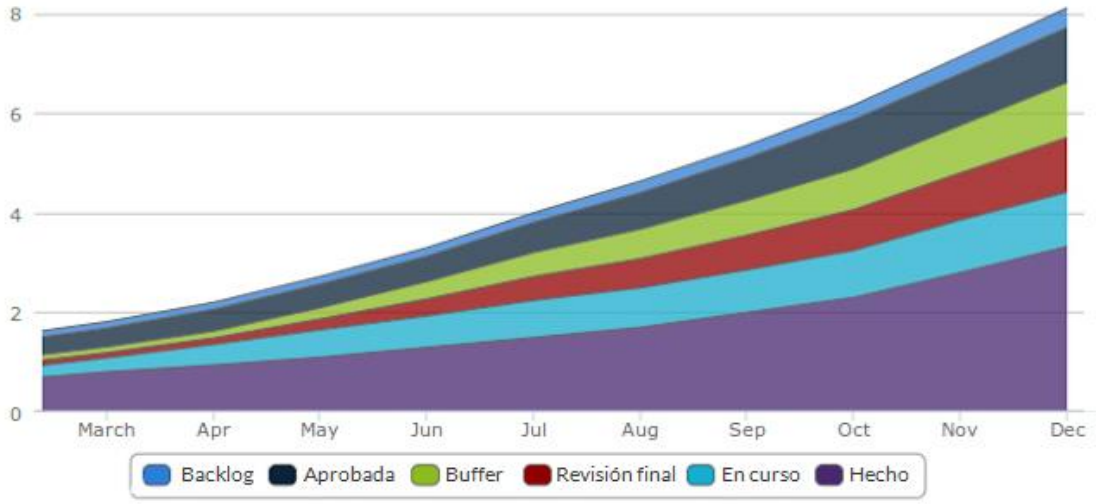


Ilustración 12. Diagrama flujo acumulativo (Kanban Tool, 2022b)

Este diagrama de flujo acumulativo muestra, por etapas, la cantidad total de trabajo en curso y ayuda a analizar la estabilidad del flujo y las tendencias para su finalización. En el eje "x" se representa el tiempo, mientras que en el eje "y" se visualiza el número de elementos de trabajo. Si se ve que hay saltos y líneas planas en la gráfica, significa que hay cuellos de botellas o limitaciones. Por otro lado, si las líneas suben suavemente como en la figura anterior, esto representa que el flujo de trabajo es constante y da una mayor fiabilidad para predecir la fecha de finalización del proyecto (Atlassian, 2022c).

La comparación entre el tiempo de entrega y el tiempo que realmente se trabaja da el rendimiento del proceso. Este rendimiento sirve para analizar si los límites del trabajo en proceso que se ha establecido son correctos o si, por el contrario, se debería hacer alguna modificación.

- Explicar las políticas de gestión. Hay que definir todo el proceso que se va a llevar a cabo, cuáles son los objetivos y estar seguros de que todo el equipo lo ha comprendido correctamente.
- Identificación de mejoras. Una vez se ha terminado un proceso, se analiza lo que se ha hecho para ver qué se puede hacer mejor.

Tener un flujo de trabajo transparente y definir eficazmente los límites del trabajo en curso ayudará al equipo a detectar y eliminar rápidamente los cuellos de botella. Las tarjetas presentan información vital sobre el elemento como quién es el responsable de llevar a cabo esa tarea, una descripción del trabajo que se está realizando, la duración estimada y pueden llegar a contener algún elemento gráfico que pueda servir de ayuda para su elaboración (Andersen, 2011).

Puede parecer que este modelo es muy similar al Scrum, pero hay que destacar que tiene bastantes diferencias. Para mostrarlas de forma más visual, se ha decidido presentar una tabla basada en lo expuesto por Kniberg y Skarin (2010):

Scrum	**Kanban**
Se definen iteraciones de duración limitada (Sprints).	La duración definida de la iteración es opcional (puede ser "eventual").
El equipo se compromete a entregar un determinado contenido funcional durante el Sprint.	El compromiso es opcional.
Establece equipos multi-competenciales.	Se prefieren equipos especializados, aunque no se prohíben los equipos multi-competenciales.
Una historia hay que descomponerla para llevarla a cabo en un Sprint.	No hay tamaño definido para un "ítem" de desarrollo (esto es lógico porque no hay Sprints por defecto).
Se limita el trabajo en curso por iteración, por Sprint.	Se limita el trabajo en curso por estado en flujo de trabajo
Se deben planificar las tareas en el Sprint	No es necesaria una planificación.

planning.	
No se puede modificar el contenido de un Sprint en curso.	Se pueden añadir/eliminar aspectos a tratar en cualquier momento, siempre que se mantengan en los límites permitidos de cada etapa del proceso.
El backlog del Sprint pertenece a un equipo y solo a uno.	El tablero Kanban se puede compartir.
Define 3 roles concretos (PO/Scrum Master/Equipo).	No establece ningún rol.
El tablero Scrum (tablero de las tareas del equipo durante un Sprint) se reinicializa en cada Sprint.	El tablero Kanban es persistente.
Se necesita un Backlog de Producto priorizado.	La priorización es opcional.

Tabla 1. Comparación Scrum - Kanban (Kniberg & Skarin, 2010)

Por último, algunas de las ventajas que puede presentar esta metodología son las siguientes (Andersen, 2011):

- Flujo constante de trabajo, pero sin sobrepasar el límite. Se consigue más haciendo menos.
- Ayuda a identificar los cuellos de botella.
- Acorta el tiempo de ciclo desde la solicitud del cliente hasta la entrega.
- Transparencia al ver en el tablero todo el flujo de trabajo obteniendo una visión general de cada etapa.
- Se trabaja en la mejora continua.
- Gran solución para proyectos de distintas urgencias, complejidades y tamaño.

5.4.6. Scrumban

No se va a desarrollar esta metodología, pero quería mencionarla porque se trata de un modelo híbrido entre las metodologías Scrum y Kanban que han adoptado muchas empresas.

Como se exponía antes, las empresas suelen adaptar estas metodologías y modificarlas según les funcione mejor. Por eso mismo, varias empresas vieron el potencial que tenían estas dos nombradas

metodologías y decidieron mezclarlas para quedarse con lo que les aportaba más valor de cada una de ellas.

Esto es un claro ejemplo de que no se deben quedar solo con lo que ya está creado, sino que hay que ir buscando la mejor forma de funcionar propia de cada empresa.

6 HERRAMIENTAS PARA METODOLOGÍAS ÁGILES

En este apartado se van a exponer algunos ejemplos de herramientas que pueden ser de mucha utilidad a la hora de aplicar un marco de trabajo ágil. Se verá en qué consisten, qué aportan y algunas ventajas e inconvenientes.

6.1. Herramientas

Existen varios softwares que pueden ayudar a gestionar los proyectos utilizando metodologías ágiles. Algunos de ellos son:

- Jira
- Trello
- ActiveCollab
- Pivotal Tracker
- GitLab
- Zoho Sprints
- Asana
- Axosoft
- IceScrum
- Planview
- Flow-e
- Planning Poker
- Taiga
- Slack
- Azure Boards
- Version One
- Scrumblr

Se expone esta lista por si se quiere investigar sobre otras herramientas que existan y se adecue más a otras ideas porque, como es comprensible, no se pueden explicar todas en este trabajo. Por tanto, se desarrollarán aquellas de mayor utilidad y de otras se mencionarán sus principales características.

6.1.1. Jira

Es una herramienta ideal para empresas que se dedican al software y actualmente es la que se usa en la empresa que es foco de estudio en este trabajo desde principios de 2022.

Según la información que viene en su página oficial y la que se ha recopilado utilizando este programa, se puede decir que es compatible con cualquier metodología ágil, ya sea Scrum, Kanban o una propia creada (Atlassian, 2022d).

La plantilla de Scrum, entre otras cosas, ayuda a organizar el trabajo en cortos y delimitados periodos de tiempo denominados "*Sprints*", así como también a estructurar y clasificar el trabajo del backlog y del tablero según su prioridad, gestionar las tareas de realización asociadas a las Stories a través del tablero de Scrum configurable, ... Por otro lado, la plantilla de Kanban cuenta con el tablero, antes mencionado, en el que se basa esta metodología.

Su interfaz es muy cómoda e intuitiva porque muestra todas las actividades venideras, en desarrollo, finalizadas o en incidencias desde un mismo módulo principal organizado por columnas.

¿Su inconveniente? Es de pago.

Las principales funcionalidades de este software son las siguientes (Atlassian, 2022e):

- Planifica. Jira posee unas plantillas de tablero Scrum o Kanban que permiten dividir los proyectos grandes y complejos en incidencias, tareas e historias de usuario por equipos. De esta forma, se puede trabajar por Sprints.

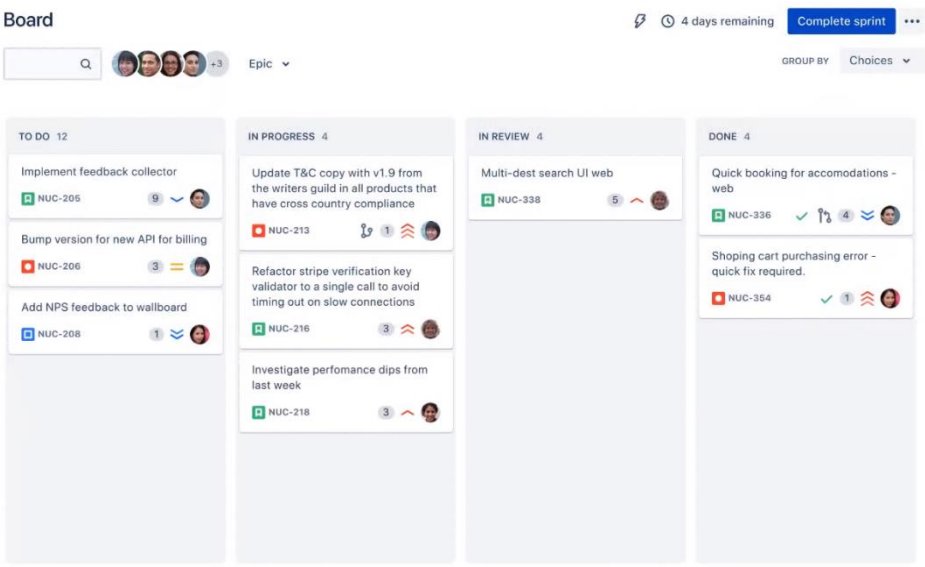


Ilustración 13. Tablero de Jira (Atlassian, 2022c)

- Supervisa. Consta de unas hojas de ruta que permiten hacer un seguimiento del panorama general a nivel de epics. Además de esto, también se pueden visualizar las dependencias que se puedan dar para adelantarse a ellas, ver si los equipos cuentan con la capacidad suficiente para completar el trabajo en el tiempo establecido, crear varios escenarios para ver las distintas situaciones y ver cuál beneficia más a la empresa.
- Lanza. Permite ver el avance de las versiones del software, teniendo de esta forma toda la información actualizada y poder lanzar estas versiones con confianza.
- Informe. Con esta función, se pueden identificar las tendencias y conseguir mejorar el rendimiento del equipo, estando siempre al día y permitiendo la toma de decisiones basadas en datos.
    - Informes para equipos de Scrum (Atlassian, 2022f):
        - Informe de Sprint o burndown chart: Este diagrama se puede aplicar sobre varios casos: un proyecto completo, una entrega o un Sprint. Este último caso es donde más se utiliza, y el que se va a describir, aunque es similar a los otros. Se trata de un gráfico que ayuda a determinar si los compromisos y la corrupción del alcance son excesivos, y qué tareas se ha terminado en cada Sprint para ver si se podrá completar el trabajo en el tiempo establecido. En el eje de abscisas se indica la duración del Sprint y en el de ordenadas la suma de la cantidad de trabajo de cada *User Story* que contiene el Sprint. Luego, se agrega una progresión lineal que representa cuándo se terminaría el proyecto de forma ideal (representado en la ilustración 14 por el número 3). Por último, conforme se vayan realizando las tareas, se mostrará la tendencia de progreso que está siguiendo el equipo (línea 2), que es lo que va a aportar más información.

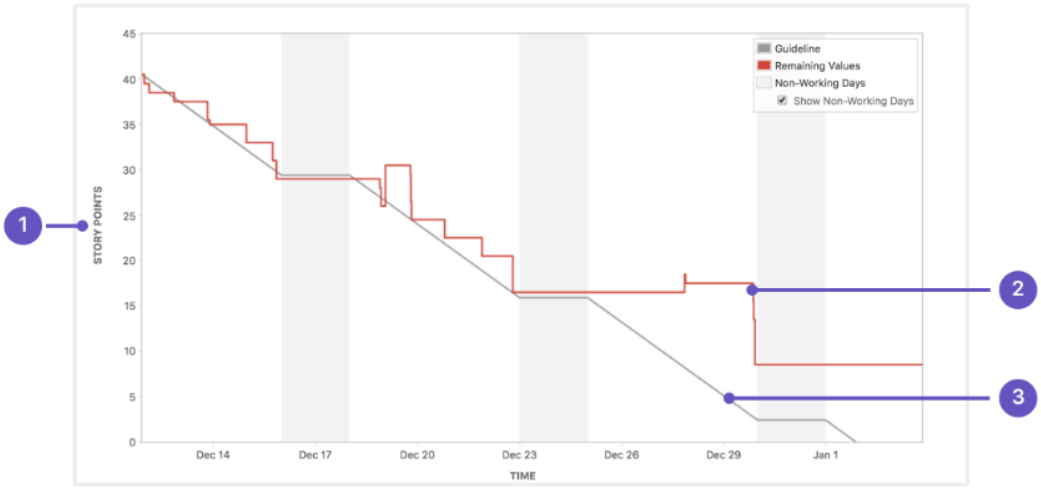


Ilustración 14. Burndown chart de un Sprint (Rehkopf, 2022c)

Es decir, esta gráfica permite decidir qué acciones tomar dependiendo de en qué situación se esté.

- Diagrama de evolución hacia los objetivos de los Sprints o *burnup chart*: permite visualizar el progreso hacia los objetivos de los Sprints, a su alcance total. También ofrece advertencias para identificar problemas con la desviación.

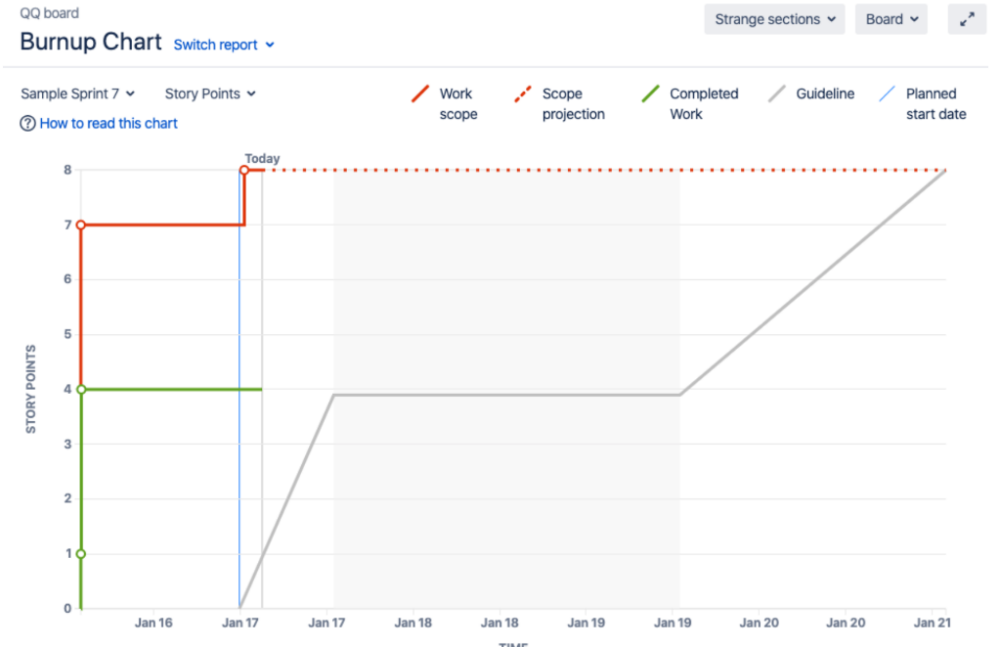


Ilustración 15. Ejemplo de gráfico burnup chart de Jira (Atlassian, 2020)

- Evolución de publicaciones: monitoriza la fecha definida para lanzar las versiones y, si el trabajo está retrasado, ayudará a tomar las medidas adecuadas.
- Gráfico de velocidad en cuanto a los Sprints.

- Informes para equipos de Kanban:
  - Diagrama de flujo acumulado: muestra el nivel de estabilidad del flujo y ayuda a entender dónde se necesita más enfoque para que el proceso sea más predecible. Da una visión de los problemas presentes y pasados y podemos ver el tiempo de ciclo, el rendimiento y el trabajo en curso.
  - Gráfico de control: puede ayudar a determinar si los datos del Sprint que se está llevando a cabo se pueden utilizar para identificar el futuro rendimiento del equipo. Este gráfico muestra el tiempo de ciclo del producto, versión o Sprint. Recoge el tiempo empleado en solucionar una incidencia y lo mapea durante un periodo de tiempo concreto (Atlassian, 2022g).

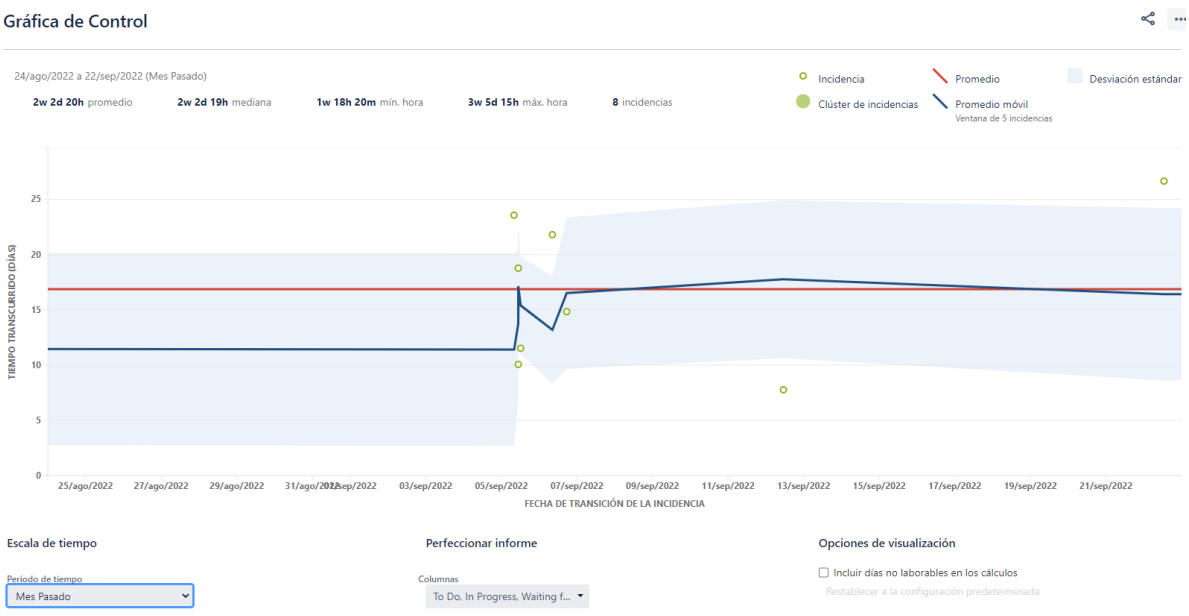


Ilustración 16. Gráfica de Control de un proyecto de la empresa objeto de estudio. (Elaboración propia)

Aparte de estos informes mencionados, también hay otros generales como:

- Informe de edad media
- Informe de incidencias creadas vs resueltas
- Informe de incidencias creadas recientemente
- Informe de tiempo de resolución
- Informe de tiempo desde incidencias

- Informe de carga de trabajo del usuario
- Informe de seguimiento del tiempo

- Automatiza. Jira Automation permite automatizar cualquier tarea o proceso según unas plantillas como pueden ser (Atlassian, 2022c):
  - Asignación automática de incidencias
  - Sincronizar incidencias: si se finaliza un epic, todas las historias o tareas que lo compongan se macarán también como finalizados.
  - Resumen diario: se puede enviar diariamente por Slack o MS Teams una lista de incidencias abiertas en un Sprint o incidencias bloqueadas, entre otras cosas.
  - Cierre automático de incidencias antiguas: se puede establecer un número máximo de días para que, si el cliente no contesta a una incidencia, esta se cierre automáticamente.

Otra funcionalidad para comentar es que desde este tipo de herramientas también se puede llevar directamente la imputación de horas. ¿Por qué es tan importante llevar el control del tiempo real? Las razones son las siguientes:

- El tiempo es dinero. Y más cuando se facturan esas jornadas de desarrollo. Todas las empresas quieren tener controlado su dinero para ver dónde se invierte más o menos y dónde tienen que hacer cambios si algo no funciona.
- Controlar el tiempo invertido ayuda a ver y analizar la diferencia entre lo teórico y lo real y, de esta forma, poder tomar decisiones fundamentadas. Esto es lo que permite luego poder sacar todos los informes que se han expuesto antes.
- Con el histórico de imputaciones será más sencillo calcular una estimación más cercana a la realidad de cada desarrollo nuevo.
- Se podrá ofrecer una transparencia total sobre el progreso del proyecto.

En cada tarea/historia que se cree, da la opción de agregar el tiempo dedicado a ella, seleccionando día y las horas que se han invertido en esa tarea ese día.

6.1.2. Trello

Al igual que Jira, se trata de una herramienta gestionada por Atlassian. Se puede acceder a ella desde el navegador o desde dispositivos móviles, aunque no se tenga conexión a Internet (Atlassian, 2022h).

Se trata de un tablero similar al de Kanban, en el que se definen las tareas, se las asignan a miembros del equipo, se definen las fechas de acción y de vencimiento, se pueden establecer recordatorios para que no se pasen estas fechas, … Además, se pueden adjuntar archivos a cada una de estas tareas.

Permite separar estas Historias de Usuario en tareas con un checklist. Cuando todas las tareas estén realizadas, la Historia pasará a la columna de 'Terminada'.

Este módulo puede ser un gran complemento a Jira. Se puede consultar el estado, el responsable y la prioridad de las incidencias que tenemos creadas en Jira, y vincularlas a las tarjetas de Trello, entre otras cosas.

6.1.3. Axosoft

Este software es parecido a Jira también. Está muy orientado a Scrum y permite llevar un buen seguimiento de los errores (Subra & Vannieuwenhuyze, 2018). Es bastante conocido internacionalmente, pero no especialmente en España, por eso se va a explicar un poco.

Tiene una pequeña versión gratuita para que pueda probarse en proyectos reales, aunque sea en menor medida (solo se pueden dos usuarios).

Su interfaz de usuario se resume en tres apartados principales:

- Work Items, destinado principalmente a la gestión de los diferentes niveles de Backlog (Product Backlog, Release Backlog, Sprint Backlog).
- Release Planner, dedicado a poder ver la capacidad del Sprint y del equipo de trabajo.
- *Dashboard*. Permite construir cuadros de control basados en distintas gráficas que pueden servir de mucha ayuda para comprender cómo se está funcionando (algunas de estas ya se han ido mencionado en otros apartados):
    - Gráfico de artículos
    - Burndown chart
    - Gráfico de velocidad
    - Trabajo restante
    - Diagrama de flujo
    - Estimado frente real

Como con Jira, dispone de una gran flexibilidad de configuración (Axosoft, 2022):

- Configurar su workflow
- Configurar los atributos de sus ítems.
- Configurar notificaciones sobre diferentes eventos.

Para terminar, un punto muy interesante es la posibilidad de configurar un portal de soporte para sus clientes. Además, con los workflows es posible rellenar tickets de incidencias que entran directamente y puede convertir los correos electrónicos en tickets también para llevar un total seguimiento, una

funcionalidad muy requerida hoy en día.

6.1.4. IceScrum

Se trata de una herramienta de código abierto también dirigida principalmente a Scrum, como bien indica su nombre. Está disponible para Cloud u *on premise* (Subra & Vannieuwenhuyze, 2018).

Este software tiene las funcionalidades básicas de las demás herramientas de este tipo, como son la creación y gestión de tareas, creación de los backlogs (tanto del producto como de os Sprints), seguimiento, ... Pero además de esto, se pueden destacar algunas otras muy interesantes (Kagilum SAS, 2022a):

- Se pueden crear características, que son funciones del producto que aportan valor al usuario. Estas suelen requerir mucho tiempo para su total desarrollo (varios Sprints), por lo que se dividen en *Stories* conectadas.
- Divide las Stories en tres tipos:
    - Historias de usuario: son las que aportan valor a los usuarios u *stakeholders.*
    - Historias de defecto: aquellas que eliminan valor, es decir, que representan un error.
    - Historias técnicas: aquellas que aportan valor al equipo de trabajo.
- Las Historias pueden empezar en un "área de ensayo", es decir, empiezan estando en un estado de prueba en el esperan la aprobación del *Product Owner* para pasar al *Product Backlog*. Así se asegura de que todos los elementos estén correctamente definidos.
- Como se introduce en el punto anterior, las Historias pasarán por distintos estados que son los siguientes:
    - Sugerida: cuando se encuentra en el "área de ensayo" que se comentaba anteriormente.
    - Aceptada: cuando el *Product Owner* da el visto bueno y se introduce en el Backlog del Producto.
    - Estimada: cuando se determina el esfuerzo que puede conllevar realizar esta Historia.
    - Planeada: se adjudica la Historia a un *Sprint.* Es decir, se introduce en el backlog del *Sprint*.
    - En progreso: pasa a este estado en el momento en el que se inicia la tarea. Es común a las otras herramientas.
    - En revisión: es el estado en el que se comprueba su validez. Para ello, hay que vincular la Historia con las pruebas de validación, lo que permite también llevar un buen seguimiento.

    - Terminada: pasa a este estado cuando su validez ha sido aprobada por el *Product Owner*.
    - Congelada: pasa a estar en este estado cuando ha sido validada, pero se decide no adjuntarla a la versión.
- A su vez, las Historias se dividirán en tareas, que solo podrán estar adjudicadas a una persona. Cuando una Historia pase a 'Terminada', todas las tareas que estén asociadas a ella también pasarán a ese estado.
- Nos proporciona un *Timeline* con los diferentes lanzamientos de versiones que tenemos programados, con sus respectivos *Sprints.*

Haciendo clic en una Release, se visualizan todos los Sprints de esta y fácilmente se pueden planificar las Stories simplemente arrastrando y soltando, aunque también se puede automatizar este proceso. El software tiene en cuenta la velocidad y la prioridad de las Stories para producir un plan de Sprints automáticamente.

En modo estándar, iceScrum permite adjuntar al Sprint los documentos con la "Definición de terminado", lo que es bastante inteligente, o un acta de la Retrospectiva o cualquier otro documento útil a nivel del Sprint.

Los informes e indicadores que se pueden visualizar con esta herramienta son los siguientes, algunos de ellos ya se han mencionado anteriormente (Kagilum SAS, 2022b):

- Burndown chart
- Burnup chart
- Diagrama de flujo acumulado
- Gráfico de estacionamiento: representa el grado de finalización de las características en función de sus Historias asociadas que hayan pasado al estado 'Terminada'. De esta forma, se puede ver el progreso general comparando las distintas áreas.

  En la figura 17, se muestra un ejemplo de este gráfico. Como se puede observar, en el eje de ordenadas están agrupadas las distintas 'Características' que se han definido para ese proyecto, y en el de abscisas se encuentran el porcentaje de finalización de cada una de ellas.

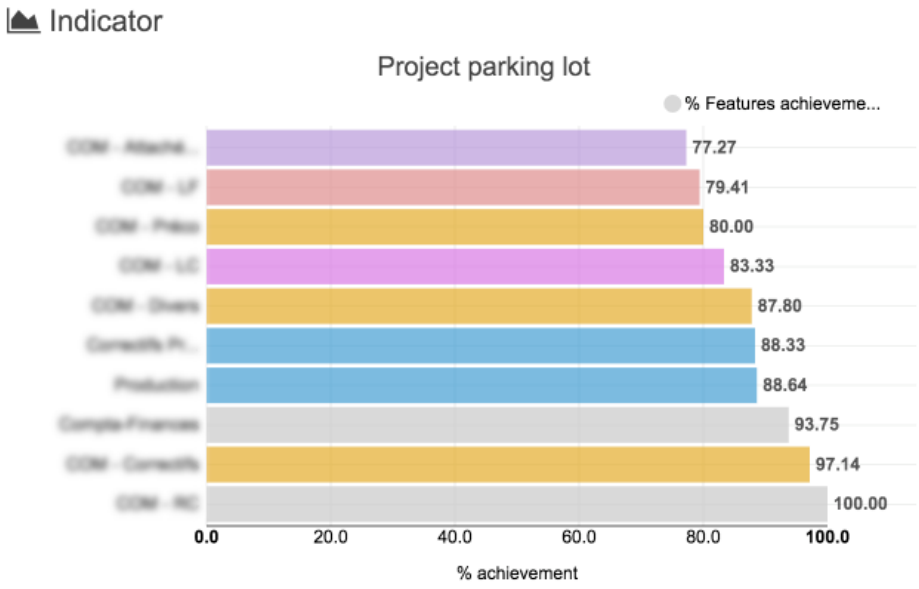


Ilustración 17. Gráfico de estacionamiento (Kagilum SAS, 2022b)

- Gráfico de velocidad
- Gráfico de velocidad frente a velocidad planificada: muestra la variación entre la velocidad que ha llevado el equipo en los anteriores *Sprints* (es lo que se representa en verde en la figura 18) y la velocidad que se ha planificado (representado en azul). Para decir que se lleva una excelente gestión del proyecto, las dos líneas tendrían que coincidir en el final de cada *Sprint*. Esta gráfica ayuda a reflexionar sobre el tiempo que se estima para las Historias, si se inician demasiadas Historias en un mismo *Sprint*, ...

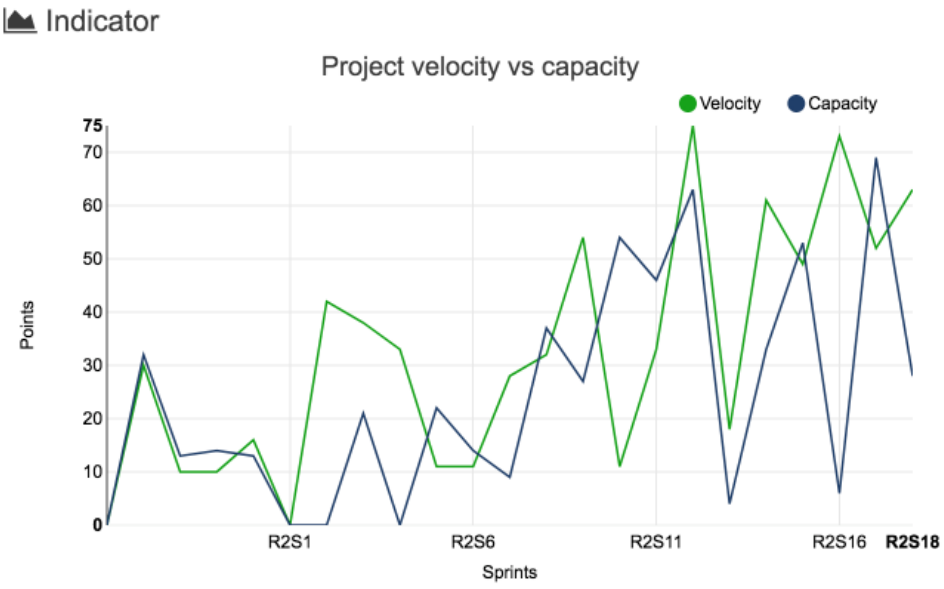


Ilustración 18. Gráfico de velocidad frente a velocidad planificada (Kagilum SAS, 2022b)

- Notas de lanzamiento y notas de Sprint: permiten generar automáticamente las historias incluidas en un lanzamiento mediante plantillas personalizables para HTML y MarkDown.

6.1.5. Zoho Sprint

Otra herramienta similar a las anteriores que se centra en proyectos Scrum. Una de las características a destacar de este software es que te permite establecer unos límites en el trabajo en curso como se propone en Kanban, como se muestra en la ilustración 19. Proporcionará informes como el gráfico de velocidad, burndown y burnup, diagrama de flujo acumulativo, gráfico de progreso del epic, informes de partes de hora (para ver las horas de registro de un Sprint, del tipo de trabajo, por usuario y por mes) y notas de lanzamiento (Zoho Corporation, 2022a).

Para obtener más información acerca de esta herramienta, consulte la página web Online Agile Project Management Software | Zoho Sprints que es de donde se ha obtenido estos datos.

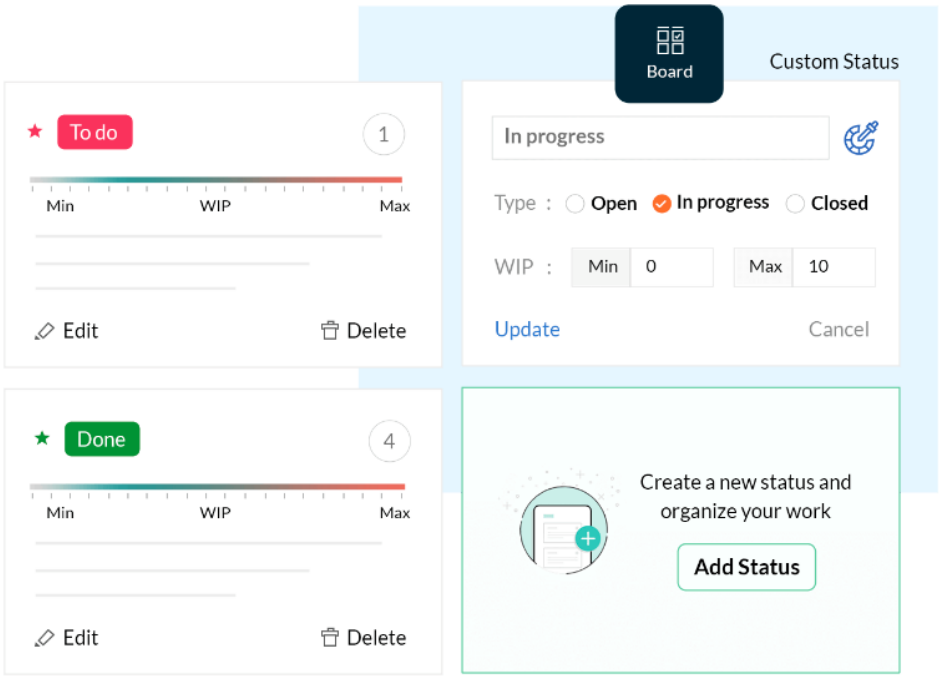


Ilustración 19. Personalización de estados con Zoho Sprints. (Zoho Corporation, 2022b)

6.1.6. Asana

Esta herramienta no solo se puede utilizar para la gestión de proyectos.

Lo más diferenciable de este software es su cronograma. Se pueden crear todas las tareas e historias de usuario de forma muy visual y asignarles sus fechas, recursos y dependencias. De esta forma, cuando se representan en el cronograma, se pueden ver todos los conflictos o incompatibilidades que se puedan dar (Asana, 2022). Además, se podrán ir modificando y ajustando, de forma muy sencilla, las tareas a medida que el trabajo vaya cambiando. También hay que mencionar que, si ya se tienen

las tareas planificadas en una hoja de cálculo, se podrán importar el archivo CSV para que se cree directamente el cronograma.

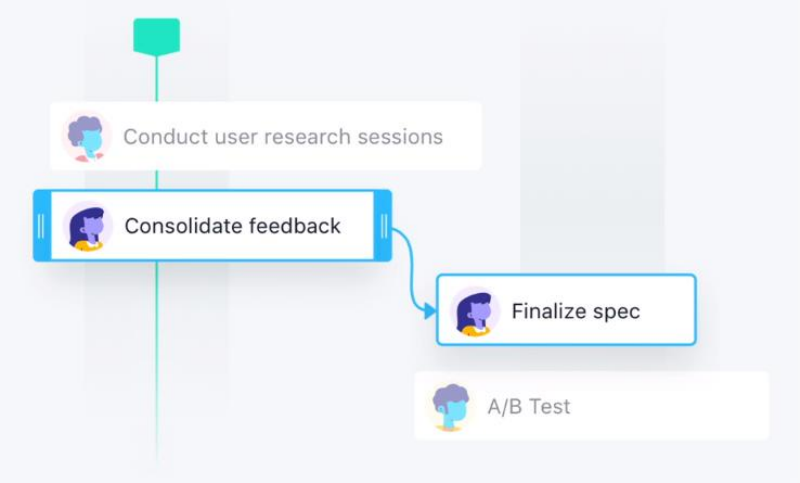


Ilustración 20. Cronograma de Asana (Asana, 2022)

6.1.7. PlanView

Se trata de un software para la gestión del trabajo y la cartera que ha sido referenciado por Gartner como líder para informes y gestión adaptativa de proyectos (Gartner, 2022), entre otras cosas.

Esta herramienta consta con un tablero Kanban, con la posibilidad de establecer límites en el número de tarjetas que se pueden colocar en cada estado o en la cantidad que se puede asignar a cada trabajador. También ofrece ver las dependencias entre equipos para poder identificar los retrasos que sen puedan dar en las entregas (Planview, 2022).

7 MARCO DE TRABAJO

A continuación, se va a exponer cómo se trabaja en esta empresa objeto de estudio, cuál es la función que desempeño y qué mejoras se proponen para aquellos proyectos que constan de GAPs o disturbances.

7.1. Contexto de la empresa objeto de estudio

Esta empresa que se ha ido mencionando a lo largo de este documento, se trata de una compañía cuya función es proveer de soluciones de software a aquellas empresas que quieren mejorar sus procesos. Estos softwares están totalmente diseñados por ellos, por lo que tienen total libertad y conocimiento para hacer todas las modificaciones que las empresas clientes quieran (ya que siempre habrá algo que no les guste, que ellos lo hagan de otra forma o que necesiten unos procesos específicos y haya que adaptar el software). Por esto mismo, la empresa está enfocada en desarrollo del software.

Actualmente, la compañía está dividida en varios departamentos que son:

- Operaciones o desarrollo del negocio. Este departamento se centra en la captación de clientes y de partners, asuntos relacionados con campañas comerciales, desarrollo del negocio, etc.
- Comercial. Aunque el nombre del departamento sea este, no solo incluye a los comerciales, sino que también se encuentran dentro de este los preventas (tanto de software como de infraestructura), el equipo de compras y el personal de marketing. Los comerciales se encargan de la relación con el cliente y de aterrizar las oportunidades de ventas. Los preventa, por otro lado, se encargan de la valoración de los desarrollos a medida y los aspectos más técnicos en estas primeras fases con el cliente, el equipo de compras lleva toda la relación con los proveedores, y el equipo de marketing se encarga de llevar todas las redes sociales de la empresa, preparar los eventos, ...
- Software. Este a su vez se subdivide en varios equipos: implantación, desarrollo y personalización, y postventa. En la parte de implantación se encuentran todos los responsables y directores de proyecto. En la de desarrollo y personalización se encuentra el equipo que se encarga de hacer todos los desarrollos y personalizaciones de software que se hayan acordado con el cliente. Y, por último, la parte de postventa se encarga del soporte hacia el cliente, de gestionar las distintas incidencias que se van dando.

- Infraestructura. Aquí se engloba todo lo relacionado con la administración de redes y sistemas, mantenimiento de los equipos, telesoporte y DevOps.
- Customer Support and Tracking (CST). Departamento de atención al cliente.
- Gestión corporativa. Engloba recursos humanos, consultoría jurídica y equipo financiero.

Mi departamento es el comercial, siendo mi puesto el de preventa de software, como se ha comentado al principio. A primera vista puede parecer que este cargo no está nada relacionado con la mejora de los procesos de desarrollo de software, pero este puesto es el nexo entre los comerciales y el departamento de implantación (donde se encuentran todos los responsables de proyecto).

Para conseguir un buen ciclo de venta, debemos estar todos alineados. En la fase de venta, se le ofrece al cliente un presupuesto, una oferta técnico-económica en la que se define el alcance del proyecto y su valoración económica, teniendo en cuenta los desarrollos a medida que quiere el cliente.

Esto último que se ha mencionado es lo que más nos interesa en este momento. El comercial es el encargado de recoger la necesidad que tiene el cliente, pero esta necesidad hay que llevarla a un nivel más técnico. Esta es una de las tareas principales de la preventa, ver qué desarrollos habrá que hacer para satisfacer la necesidad del cliente. Además de analizar esto, tenemos que estimar cuánto costaría y qué tiempo se requeriría para llevarlo a cabo.

Aquí es donde entra ya en juego nuestro tema principal. La preventa es la que define el tiempo máximo que tiene el equipo de desarrollo para que el código cumpla con el objetivo establecido. Y también es una de las encargadas en identificar todas las necesidades correctamente junto con el comercial, que es crucial para que luego aparezcan los mínimos cambios posibles.

Lo ideal es que estas mejoras las desarrollase un responsable de proyecto que, al fin y al cabo, es el que va a llevar el control de todo el proceso y es a quién más le va a beneficiar, pero el Grado de Ingeniería de Organización Industrial dota de la capacidad y los conocimientos para desarrollar este trabajo.

Con todo esto, se ha introducido la empresa que es objeto de análisis y su estructura.

7.2. Situación actual

Ahora, se va a exponer qué es lo que se hace hoy en día en esta empresa con respecto a estos proyectos con desarrollos a medida.

Una vez se contrata el proyecto, se asigna un responsable de proyecto que es el que tendrá que estimar la fecha de entrega, definir los distintos hitos y entregas que se van a establecer y comunicar al cliente

los pasos que se van a seguir. Desde este momento, la responsabilidad pasa a ser de esta persona, es la que queda al mando. Recuerden que, en el segundo apartado, se comentó que en esta empresa se diferenciaban tres tipos de proyectos que tenían desarrollos a medida: los de implantación que tenían pequeños desarrollos, los de mejora que solo contaban con uno o dos desarrollos y aquellos de implantación que tenían un gran volumen de desarrollos.

Bien, pues actualmente, solo se lleva a cabo una metodología más o menos Scrum en los proyectos con un gran volumen de contratación que piden muchos desarrollos a medida. A estos proyectos se les asigna un director de proyecto (una figura superior a la del responsable) y, además, un equipo de desarrollo propio, que será el que siempre haga todos los GAPs y *disturbances* que se definan (puede estar formado por varias personas o ser una sola).

En los otros proyectos, el responsable pondrá en carga el desarrollo que se tenga que hacer y el desarrollador que esté libre o que tenga menos carga será el asignado para llevarlo a cabo. Puede que cada uno de los desarrollos que se necesiten para un cliente se realicen por personas distintas, no hay un equipo cerrado.

Para mejorar la gestión de los proyectos en esta empresa e intentar hacerla más ágil y óptima, se han ido introduciendo algunas técnicas. Estas son:

- Herramienta para la gestión de proyectos. A principios de 2022 se empezó a usar una de las herramientas software para la gestión de proyectos ágiles mencionada anteriormente, Jira. Esto supuso un gran avance para los responsables de proyecto, ya que se podía llevar un control de todos los desarrollos que se llevaban a cabo de una forma más sencilla. Además, optimizaba la comunicación entre el responsable y los desarrolladores que iban a participar en esos desarrollos y todo se queda registrado. Esto último es muy útil también por si alguien deja el equipo y tiene que ser sustituido por otra persona, es decir, no se pierde todo el conocimiento. Aunque sí que es verdad que cada uno tiene su forma de trabajar y de gestionar sus proyectos, no hay algo estándar definido.

- Hacer una reunión diaria. Sin embargo, estas reuniones son solo entre todos los desarrolladores para ver en qué está trabajando cada uno y cómo avanzan, no de todo el equipo que trabaja en un proyecto como se indica en la metodología Scrum. No se facilita la comunicación directa entre todas las personas que conforman el proyecto.

- En los proyectos de continuidad se trabaja con Sprints. En las distintas reuniones que se van teniendo con los clientes, se definen todos los desarrollos que van queriendo y se introducen en el Backlog del Producto para luego, poco a poco, irlos añadiendo en los backlogs de los Sprints.

- El responsable de proyecto debe planificar una reunión semanal con el cliente para informarle de cómo van avanzando los desarrollos y el proyecto en sí. Como se ha mencionado durante este documento, el cliente y la comunicación con este, son una parte fundamental de las metodologías ágiles para conseguir optimizar los procesos.

  Se ha hecho una encuesta a 14 responsables de proyecto de la empresa para analizar las distintas prácticas que se llevan a cabo (Adjuntada en el anexo I). Cuando le preguntamos sobre esta en cuestión, la respuesta fue bastante positiva: más del 85% tienen al menos una reunión a la semana con el cliente para informarle sobre el proyecto.

Ilustración 21. Periodicidad de reuniones cliente-responsable del proyecto (elaboración propia)

- Estas reuniones que se han mencionado en el punto anterior y todas las reuniones que se hagan con respecto al proyecto, desde hace unos meses, se hacen con un perfil técnico en cada área concreta por parte del cliente, no con la dirección que suele ser quien contrata el proyecto. Esto se debe a que un perfil de dirección no siempre puede responder o seguir las cuestiones que surgen en este tipo de reuniones y tampoco consiguen transmitir totalmente esta información a su equipo técnico. Por esto mismo, normalmente, se suele volver a tener la reunión con la persona correspondiente, aunque se pierde tiempo, o se perderá información que puede dar lugar a malentendidos. Es el cliente quien tiene la responsabilidad de designar a las personas concretas de su empresa para cada área del proyecto que se haya definido.
- Otro punto muy importante que mencionar aquí es que la imputación de horas de los responsables de proyecto se lleva a cabo en una herramienta propia de la empresa y, además, no suelen imputar en el mismo día en el que realizan las tareas, ni siquiera en la semana, suelen imputar a final de cada mes como vemos en los resultados de la encuesta de responsables de proyectos expuestos en figura 22.

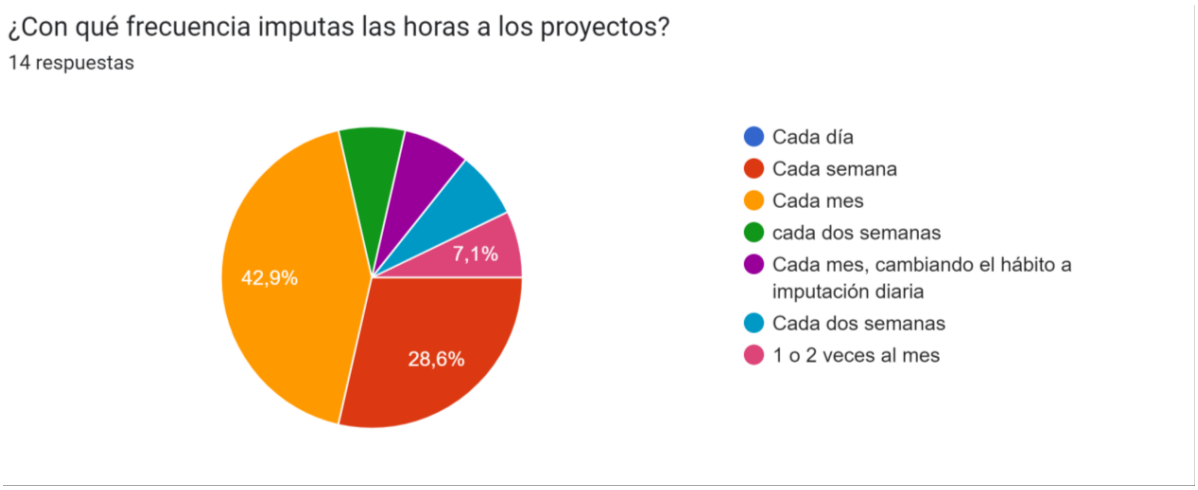


Ilustración 22. Frecuencia de imputación de horas de responsables de proyecto (elaboración propia)

En cambio, los desarrolladores imputan en Jira, supuestamente a final de cada semana, aunque podemos ver que realmente casi la mitad imputan cada día, como muestran los resultados de una encuesta que le he hecho a los desarrolladores descrita en el Anexo II y que podéis ver en la figura 23. Esta imputación se manda cada lunes a la herramienta de la empresa que es donde se lleva el control económico de los proyectos.

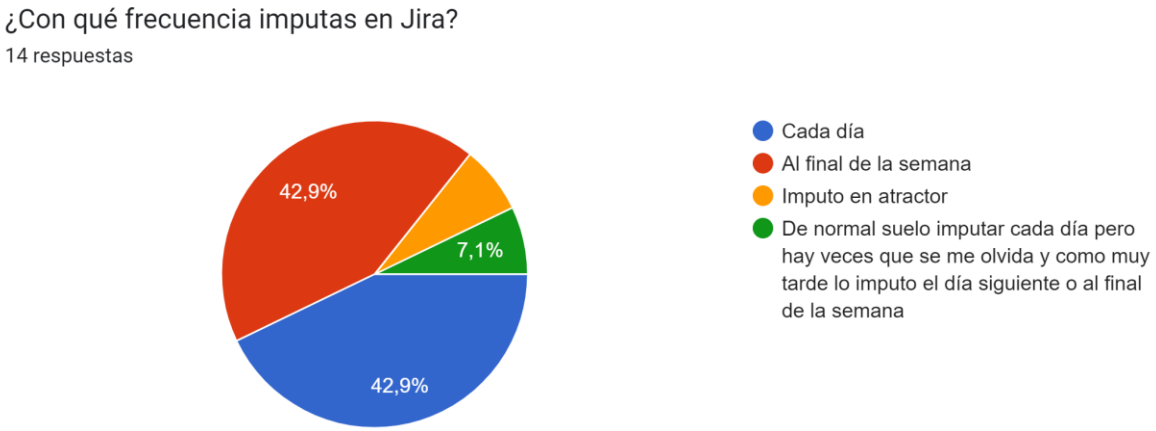


Ilustración 23. Diagrama con las respuestas de los desarrolladores sobre la frecuencia de imputación. (elaboración propia)

Los responsables de proyecto no imputan en Jira porque no suelen tener tareas ahí. Casi todas las tareas que hay en Jira están asociadas al equipo de desarrollo.

7.2. Carencias

Una vez expuestas las prácticas que se llevan a cabo actualmente, los defectos o las faltas que he apreciado en este procedimiento son:

- No hay una mejora continua ya que los responsables de proyecto no analizan cuál es la situación del proyecto. Es decir, si una incidencia ha estado más tiempo de la cuenta en espera o en resolución, si hay más elementos de los que corresponden en el product backlog, si se tarda mucho en una tarea, … Todo esto que se ha mencionado, debería revisarse cada cierto tiempo y, para hacerlo de una manera más sencilla y rápida se podrían usar algunos de los informes que tiene Jira y que se han descrito en apartados anteriores, el problema es que no se utilizan. No se aprovechan los recursos que se tienen y que se están pagando (contamos con la version de pago de Jira).
- No se identifican cuellos de botella con antelación. Como no se analiza la información, tampoco se puede reconocer cuáles son los recursos que están más saturados, ni cuál es el proceso que requiere más trabajadores.
- Se pierde información y tiempo al no tener un equipo concreto para un proyecto. Si cada desarrollo es realizado por una persona distinta y no se hacen reuniones diarias con el responsable, cada vez que se quiera informar de algo, se perderá el doble de tiempo.
- No se puede llevar un buen control si no se imputa en el mismo día. Actualmente, como se ha mencionado antes y como se ha demostrado con las encuestas a los responsables de Proyecto y a los desarrolladores, la mitad de los responsables imputan a final de mes y de los desarrolladores al final de la semana. Como se sabe, la memoria no es perfecta y, cuanto más tiempo pasa, la capacidad de recordación es menor y acabamos olvidando detalles importantes como cuánto tiempo hemos dedicado a cada tarea. Esto hace que, si se analiza la situación del momento, no se tenga una visión real de la misma.

  Esto no solo repercute en el análisis del proyecto, sino que esta imputación de horas se utiliza también a nivel de preventa para ver cuántas horas se han requerido para un desarrollo si tiene que valorar uno similar para otro cliente. Si la imputación no es real, puede dar lugar a presupuestos mal hechos que pueden llevar a proyectos con pérdidas de costes.
- Ahora mismo, la herramienta de Jira se utiliza solo para asignar los desarrollos a los trabajadores y tener la información recogida en un mismo sitio, pero está muy poco aprovechada. Contamos con la version de pago, por lo que se debe decidir si interesa sacarle más provecho a esta herramienta o cambiar a otra más sencilla y con menos costes.

7.3. Mejoras propuestas

En función de estas carencias identificadas, vamos a proponer las siguientes mejoras:

- Mayor uso de Jira. Se opta por intentar usar más esta herramienta debido a toda la información que se ha recogido y a que es una propuesta relativamente nueva. Para ello, todos los responsables de proyecto deben introducir todos los desarrollos que haya que hacer especificando si se trata de un GAP o un *disturbance*. Además de esto, se debe acompañar cada uno de estos con una descripción técnica de su alcance (no vale con el trabajo que hizo preventa, sino que hay que llegar a un nivel mucho mayor). Debemos inculcar a los responsables y a los desarrolladores que Jira puede ser su mayor aliado, ya no solo para llevar el control del proyecto, sino que también les puede servir para optimizar procesos y tiempo en el momento en el que se tenga que hacer un nuevo desarrollo, si este ya se ha hecho en otro cliente o se ha realizado uno con gran similitud. Esto último que he mencionado, hace referencia a la gestión del conocimiento.
- Conseguir que todos imputen al final del día, o en su defecto a final de semana, tanto los responsables de proyectos como los desarrolladores para que se pueda analizar la información real. Esta es una de las propuestas más importante, ya que, si no se cumple esta, no se podrá llevar un control económico real de los proyectos en todo momento ni podremos conseguir información exacta sobre la situación de estos.
- Análisis de la información. En relación también con el primer punto, vamos a intentar aprovechar la capacidad de análisis de la herramienta que tenemos. Los responsables deberán revisar y analizar, por lo menos una vez a la semana (o a mitad del Sprint), los siguientes informes:
    - Diagrama de flujo acumulado. Como se ha mencionado en la teoría expuesta, este diagrama nos ayuda a identificar los problemas presentes y pasados y podemos ver el tiempo de ciclo.
    - Burndown chart. Este gráfico permitía ver el avance del proyecto o sprint y la tendencia de este hasta su finalización. El responsable deberá fijarse en la línea de tendencia para obtener esta información. Recomiendo volver a ver la ilustración 14 donde se puede observar un ejemplo de este gráfico. Revisando este informe, el responsable se puede encontrar con tres casos distintos:
        - El equipo podría terminar anticipadamente (la línea de tendencia llega al eje de abscisas antes que la progresión ideal). En este caso podría integrar una o varias tareas antes de que finalizase el Sprint para poder ir adelantando trabajo.
        - El equipo piensa que no podrá terminar todas las tareas en el tiempo propuesto (la línea de tendencia sobrepasa la progresión ideal, como podemos ver en la

ilustración 14). En este caso el responsable deberá identificar qué tareas son menos prioritarias o necesarias y retrasarlas al Sprint siguiente.

- El equipo va acorde con lo planificado (la línea de tendencia va acorde a la progresión ideal).

- En relación también con Jira, propongo el uso de Jira Automation para conseguir las siguientes funcionalidades:
  - Asignación automática de incidencias. Las incidencias se asignarán por defecto a la persona etiquetada como responsable en Jira que forme parte del equipo del proyecto. Esto lo que nos permite es optimizar el proceso de asignación, ya que, normalmente mientras el proyecto siga abierto, será el responsable el que tendrá que encargarse de las distintas incidencias que se den. Una vez se cierra el proyecto, el departamento encargado de resolver estas incidencias es postventa y se hace a través de otra herramienta desarrollada por la empresa.
  - Sincronizar incidencias. Cuando se finalice un epic, todas las historias o tareas que lo compongan se macarán también como finalizados, con doble validación. Esto lo qe nos permite es asegurarnos de que no se deje ninguna incidencia abierta por error ni que cerremos un epic con incidencias sin cerrar.
  - Resumen diario. Jira Automation permite enviar un resumen diario con la lista de incidencias abiertas en un Sprint por MS Teams o Slack. Como en esta empresa se utiliza MS Teams, esta lista diaria de incidencias abiertas llegará por este canal y nos permitirá llevar un mejor control del estado de nuestros proyectos y asegurarnos de no dejar de lado las incidencias.
  - Cierre automático de incidencias antigua. Se fijarán unos días máximos de espera de respuesta del cliente con respecto a una incidencia para proceder al cierre automático de esta misma.
- Para las reuniones con clientes, antes de cada reunión, tenemos que hablar con el responsable del proyecto o de área por parte de ellos mencionándole el tema a tratar para que designen a la persona concreta para esa reunión y, de esta forma, no se pierda nada de información ni de tiempo. No debemos tener una misma reunión dos veces con personas distintas, siempre tiene que ser con los usuarios clave del tema a tratar.
- Crear equipos de trabajo. Crear equipos de desarrollo lo que nos permite es centrar el conocimiento de un proyecto en un círculo reducido.
- Reuniones semanales con el equipo de trabajo. Establecer una reunión semanal entre los

responsables de proyecto y el equipo o responsable de desarrollo de, como máximo, 30 min nos permite potenciar la comunicación entre ambas partes y garantizar la correcta evolución del proyecto.

8 ANÁLISIS Y CONCLUSIONES

En este apartado, se va a exponer el análisis de las mejoras propuestas en la sección anterior y las conclusiones del trabajo.

El principal inconveniente de esta propuesta es que, para obtener unos resultados buenos y reales, todos los implicados deberían llevar al día su tarea de imputación. Esto es muy complicado y podemos ver que no todos los responsables de proyecto están dispuestos a imputar cada día (como vemos en la ilustración 24), pero la forma de conseguirlo es comunicándoles los beneficios que pueden tener. En el caso de los responsables de proyecto, hay que tener en cuenta que tienen un plus variable en función de la productividad de sus proyectos, al imputar cada día e ir revisando cómo va el proyecto a nivel de costes, puede tomar decisiones más rápidas y con más tiempo de margen. Y, además, puede revisar que todas las personas involucradas han imputado bien sus costes y no ha añadido horas, y por tanto costes, de más.

Ilustración 24. Encuesta a responsables de proyecto sobre la predisposición a imputar diariamente (elaboración propia).

Por otro lado, los desarrolladores llevan una imputación más frecuente, semanalmente, aunque deberían imputar también diariamente. Como vemos en el gráfico de la figura 25, poco más del 70% de los desarrolladores estarían dispuestos a imputar cada día.

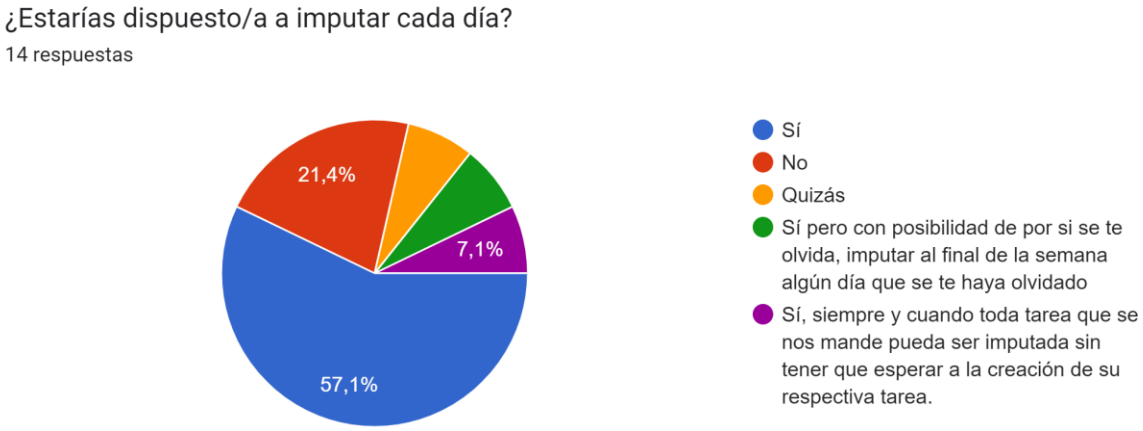


Ilustración 25. Encuesta a los desarrolladores sobre la predisposición a imputar diariamente (elaboración propia).

En cuanto al mayor uso de Jira, al principio podría costar un poco más empezar a aplicar nuevas funcionalidades, a insertar toda la información de cada GAP, pero a futuro va a optimizar el trabajo. Es similar a la incorporación de un nuevo miembro al equipo de trabajo cuando se está saturado y con gran carga. Al principio va a necesitar formación, que se le dedique gran parte del tiempo, pero al final va a beneficiar y va a ayudar a repartir esa carga de trabajo.

En definitiva, sabemos que hoy en día tenemos que estar revisando constantemente nuestro trabajo para estudiar nuevas formas de optimizarlo. Esto pasa sobre todo en el mundo de las nuevas tecnologías, del software y de la gestión de proyectos, que están en constante avance. Hay que buscar qué procedimientos y herramientas nos conducen a trabajar con mayor calidad y eso es lo que se ha hecho en este trabajo. Se han determinado algunas prácticas que pueden beneficiar tanto a la empresa como a sus trabajadores, pero lo principal es encontrar la forma de que esos nuevos procedimientos, tecnologías o metodologías que se impongan, sean cumplidos por los empleados. Hay que incentivarlos y motivarlos para que no tengamos que recordarles qué es lo que deben hacer, sino que ellos mismos se interesen por ello.

Estas mejoras propuestas se pondrán a prueba sobre 3 proyectos para ver qué impacto tienen realmente en la gestión de proyectos y si se deben mantener en el tiempo o no. Se propone probarlas en 3 proyectos para conseguir unas conclusiones con mayor fundamento, además de ver las diferencias entre las distintas formas de analizar la información. En un proyecto el responsable de proyecto utilizará el diagrama de flujo acumulado, en otro el burndown chart y en el último se utilizarán los dos.

Además de esto, se revisará cada dos días que tanto los responsables como los desarrolladores hayan imputado todas sus horas. Si se va cumpliendo esta imputación diaria, se irá ampliando el periodo de revisión de forma paulatina.

Se deja esta segunda fase de análisis de la efectividad de estas mejoras propuestas para un próximo trabajo.

REFERENCIAS

Alianza Ágil. (2001). *Principios del Manifiesto ágil*. Obtenido de http://agilemanifesto.org/iso/es/principles.html

Amaro Calderón, S. D., & Valverde Rebaza, J. C. (2007). Metodologías Ágiles. Universidad Nacional de Trujillo.

Andersen, D. J. (2011). *Kanban. Cambio evolutivo exitoso para su negocio de tecnología.* Blue Hole Press.

Asana. (2022). *Presentamos el cronograma*. Obtenido de https://asana.com/es/product/timeline

Atlassian. (2020). *Gráfico de quemado*. Obtenido de Asistencia técnica de Jira: https://confluence.atlassian.com/jirasoftwareserver/burnup-chart-983794938.html

Atlassian. (2022a). *Guía para hojas de ruta de Jira Software*. Obtenido de Atlassian: https://www.atlassian.com/es/software/jira/guides/roadmaps/basic-roadmaps

Atlassian. (2022b). *¿Qué es ágil?* Obtenido de https://www.atlassian.com/es/agile

Atlassian. (2022c). *Ver y comprender el diagrama de flujo acumulativo*. Obtenido de https://support.atlassian.com/jira-software-cloud/docs/view-and-understand-the-cumulative-flow-diagram/

Atlassian. (2022d). *Jira Software*. Obtenido de https://www.atlassian.com/es/software/jira

Atlassian. (2022e). *Funcionalidades de Jira Software*. Obtenido de https://www.atlassian.com/es/software/jira/features

Atlassian. (2022f). *Los informes en Jira Software*. Obtenido de https://www.atlassian.com/es/software/jira/features/reports

Atlassian. (2022g). *Ver y comprender el gráfico de control*. Obtenido de https://support.atlassian.com/jira-software-cloud/docs/view-and-understand-the-control-chart/

Atlassian. (2022h). *Trello*. Obtenido de https://trello.com/

Axosoft. (2022). *Scrum y seguimientos de errores*. Obtenido de https://www.axosoft.com/scrum-bug-tracking

Campo Arranz, R., del Campo Domínguez, M., & Rodrigo Raya, V. (2014). *Gestión de Proyectos.* Ra-Ma Editorial, S.A.

Drumond, C. (2022). *Scrum*. Obtenido de Atlassian: https://www.atlassian.com/es/agile/scrum

Fitzgerald, G., & Avison, D. E. (2003). *Information systems development: methodologies, techniques and tools (3rd edition).*

Gartner. (2022). *2022 Gartner® Magic Quadrant™ for Adaptive Project Management and Reporting.* Obtenido de https://info.planview.com/gartner-adaptive-pm-reporting-mq-_report_prm_en_reg.html?_ga=2.145161716.1794785838.1664989224-813184177.1664989223

Highsmith, J. (2001). *Manifiesto por el Desarrollo Ágil de Software.* Obtenido de http://agilemanifesto.org/history.html

Highsmith, J. A. (1999). *Adaptive software development : A collaborative approach to managing complex systems.* New York: Dorset House.

Highsmith, J. A. (2000). *Agile Project Management.*

Kagilum SAS. (2022a). *Documentación. Características*. Obtenido de IceScrum: https://www.icescrum.com/es/documentation/features-stories-tasks/

Kagilum SAS. (2022b). *Documentación. Indicadores e informes*. Obtenido de IceScrum: https://www.icescrum.com/documentation/indicators-and-reporting/

Kanban Tool. (2009). *Guía Kanban*. Obtenido de https://kanbantool.com/es/guia-kanban/fundamentos-de-kanban/hacer-explicitas-las-politicas-de-procesamiento

Kanban Tool. (2022a). *Fundamentos de Kanban*. Obtenido de https://kanbantool.com/es/guia-kanban/fundamentos-de-kanban

Kanban Tool. (2022b). *Gestión y medición del flujo*. Obtenido de https://kanbantool.com/es/guia-kanban/fundamentos-de-kanban/gestionar-y-medir-el-flujo

Kanbanize. (2022). *31 Ejemplos de Tableros Kanban para Diferentes Equipos*. Obtenido de https://kanbanize.com/es/recursos-de-kanban/software-kanban/ejemplos-de-tableros-kanban

Kniberg, H., & Skarin, M. (2010). *Kanban y Scrum - obteniendo lo mejor de ambos.* C4Media Inc.

Marion, J. (2018). *Project Management: A Common Sense Guide to the PMBOK, Part One-Framework and Schedule.* New York: Momentum Press.

Mrsic, M. (2017). *Crystal Methods*. Obtenido de ActiveCollab: https://activecollab.com/blog/project-management/crystal-methods

Navarro Cadavid, A., Fernández Martínez, J. D., & Morales Vélez, J. (2013). Revisión de metodologías ágiles para el desarrollo de software. *PROSPECTIVA*, 30-39.

Olic, A. (2017). *Extreme Programming*. Obtenido de ActiveCollab: https://activecollab.com/blog/project-management/extreme-programming-xp

Organización Internacional dee Normalización [ISO]. (2017). *Gestión de la calidad — Directrices para la gestión de la calidad en los proyectos*. (ISO 10006:2017).

Planview. (2022). *Planificación ágil empresarial*. Obtenido de https://www.planview.com/products-solutions/solutions/enterprise-agile-planning/

Pressman, R. S. (2001). *Software Engineering: a Practitioner's Approach.* New York: McGraw-Hill Higher Education. .

Project Management Institute. (2013). *Guía de los fundamentos para la dirección de proyectos (Guía del PMBOK). Quinta edición.* Project Management Institute.

Radigan, D. (2011). *El backlog del producto: la lista de tareas pendientes definitiva*. Obtenido de Atlassian: https://www.atlassian.com/es/agile/scrum/backlogs

Radigan, D. (2022). *Kanban*. Obtenido de Atlassian: https://www.atlassian.com/es/agile/kanban

Rehkopf, M. (2022a). *Historias de usuario con ejemplos y plantilla*. Obtenido de Atlassian: https://www.atlassian.com/es/agile/project-management/user-stories

Rehkopf, M. (2022b). *Scrum sprints*. Obtenido de Atlassian: https://www.atlassian.com/es/agile/scrum/sprints

Rehkopf, M. (2022c). *Aprende a utilizar diagramas de evolución en Jira Software*. Obtenido de Atlassian: https://www.atlassian.com/es/agile/tutorials/burndown-charts

Rial Huerta, J. S., & Montero Fernández-Vivancos, G. (2019). Aplicación de metodologías ágiles a desarrollo de proyectos.

Rojas Contreras, M., Esteban Villamizar, L. A., Orjuela Duarte, A., & 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*, 97-105. Obtenido de https://repositorio.unal.edu.co/bitstream/handle/unal/38745/26729-93655-1-PB.pdf

Schwaber, K., & Sutherland, J. (2020). *Scrum Guide*. Obtenido de https://scrumguides.org/scrum-guide.html

Scrum.org. (2022). *¿Qué es Scrum?* Obtenido de https://www.scrum.org/resources/what-is-scrum

Subra, J.-P., & Vannieuwenhuyze, A. (2018). *Scrum- Un método ágil para sus proyectos.* Barcelona: Ediciones ENI.

Wells, D. (1999). *The Rules of Extreme Programming*. Obtenido de

http://www.extremeprogramming.org/rules.html

Wells, D. (2000). *Extreme Programming Project*. Obtenido de http://www.extremeprogramming.org/map/project.html

Wells, D. (2009). *Los valores de la Programación Extrema*. Obtenido de http://www.extremeprogramming.org/values.html

Zoho Corporation. (2022a). *Zoho Sprints*. Obtenido de https://www.zoho.com/sprints/

Zoho Corporation. (2022b). *Tablero ágil de Scrum*. Obtenido de https://www.zoho.com/sprints/scrum-board.html

ANEXOS

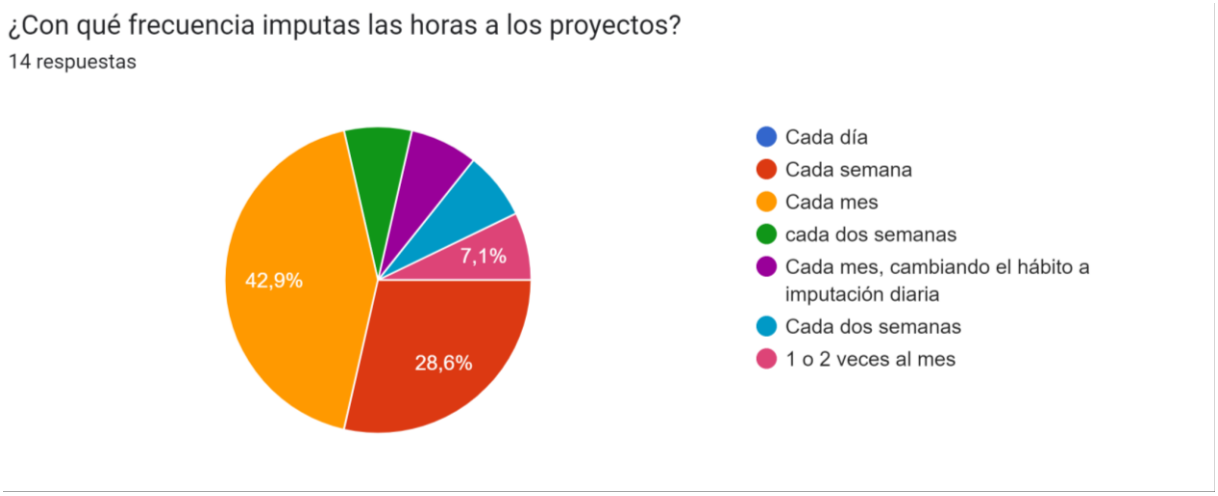
ANEXO I. ENCUESTA A LOS RESPONSABLES DE PROYECTO

Se les ha realizado una encuesta a todos los responsables de proyecto que hay ahora mismo en la empresa objeto de estudio para ver qué se puede mejorar en cuanto a las tareas que llevan a cabo, las herramientas que utilizan y su relación con los clientes y los desarrolladores.

Se puede consultar la encuesta en el siguiente enlace: https://docs.google.com/forms/d/e/1FAIpQLSdx7Lp4-LrF5c0s4cEcfJhpKn-CqK2C33CFLyyZVSU8Iq1v4w/viewform?usp=sf_link

Aquí exponemos las preguntas que se les han hecho con sus respectivas respuestas.

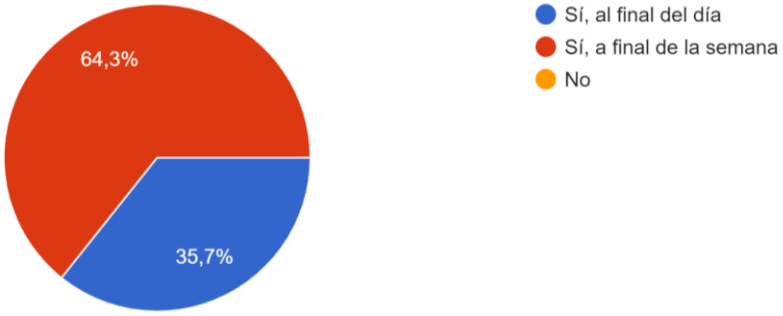
1.



2.

¿Estarías dispuesto/a a imputar cada día o, en su defecto, al final de la semana?

14 respuestas



3.

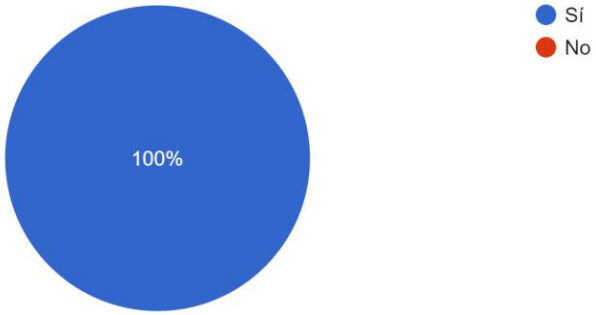
¿Crees que sería de utilidad tener una reunión diaria o a principios de semana con los desarrolladores que participan en un proyecto?

14 respuestas

4.

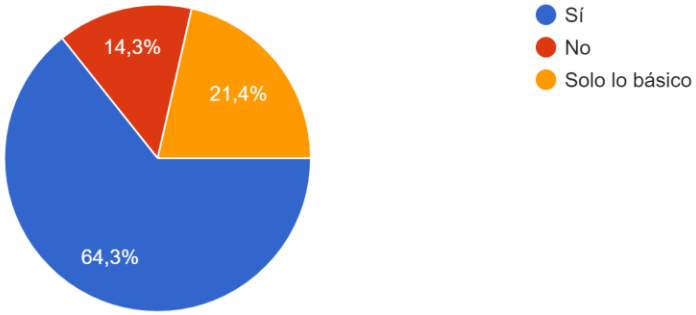
¿Te parece útil la herramienta Jira?

14 respuestas



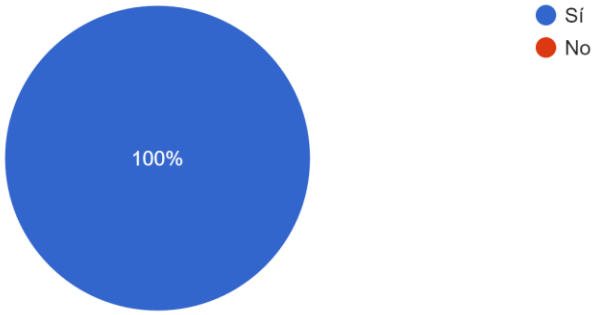
5.

¿Sueles documentar toda la información en Jira?
14 respuestas



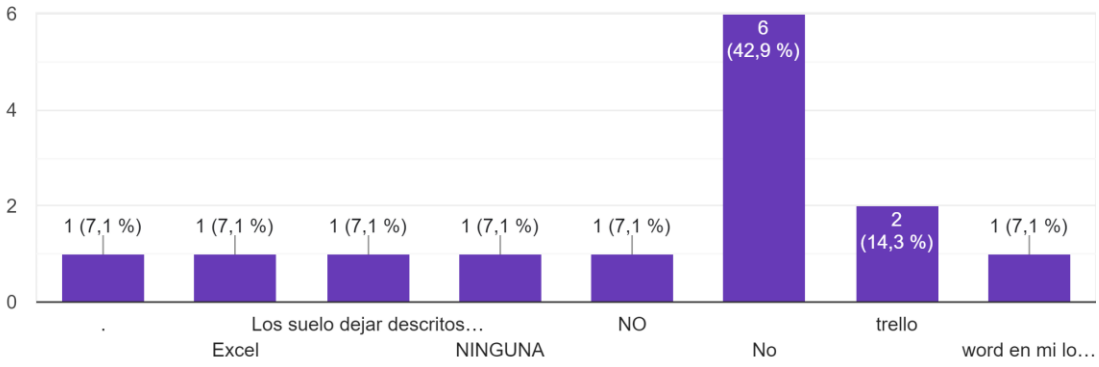
6.

¿Ves importante que se pueda analizar la información que se recoge en Jira? Relativo a la carga de trabajo, los cuellos de botella, los tiempos de entrega, ...
14 respuestas



7.

¿Utilizas otra herramienta para guardar más información acerca de los desarrollos a medida? Indicar cuál o cuáles
14 respuestas



8.

¿Sueles tener reuniones periódicas con el cliente (perfil técnico) para informarle del estado de los desarrollos?

14 respuestas

9.

¿Hay algo que quieras cambiar de tu labor como responsable de proyecto?

14 respuestas

No sabría decirte

Tener reuniones semanales con los team leader de desarrollo

TRABAJAR EN EQUIPO

Echo en falta un lugar en el que poder consultar a nivel funcional todos los procesos involucrados en una solución. Igualmente, también me gustaría disponer de un registro de gaps de otros proyectos que tal vez pueda poner en práctica en el propio.

No hay particularmente nada que quiera cambiar, pero si que deberíamos saber un poco más a cerca de cómo hacer consultas en base de datos y sobre programación java.

Tener que imputar horas.

No

No sabría decir qué

Mayor implicación del equipo técnico

Si

.

La capacidad de poder enfocar mi trabajo unicamente al papel de jefe de proyectos y no a veces como un desarrollador más

mejor formación

Nada

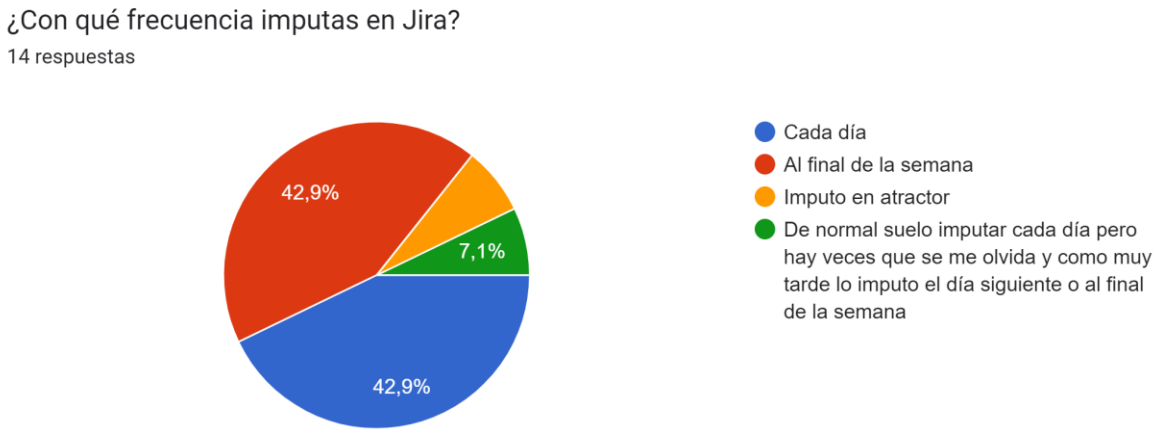
ANEXO II. ENCUESTA A LOS DESARROLLADORES

Se les ha pasado una encuesta a todos los desarrolladores que hay ahora mismo en la empresa objeto de estudio para ver qué se puede mejorar en cuanto a las tareas que llevan a cabo, las herramientas que utilizan y su relación con los responsables de proyecto.

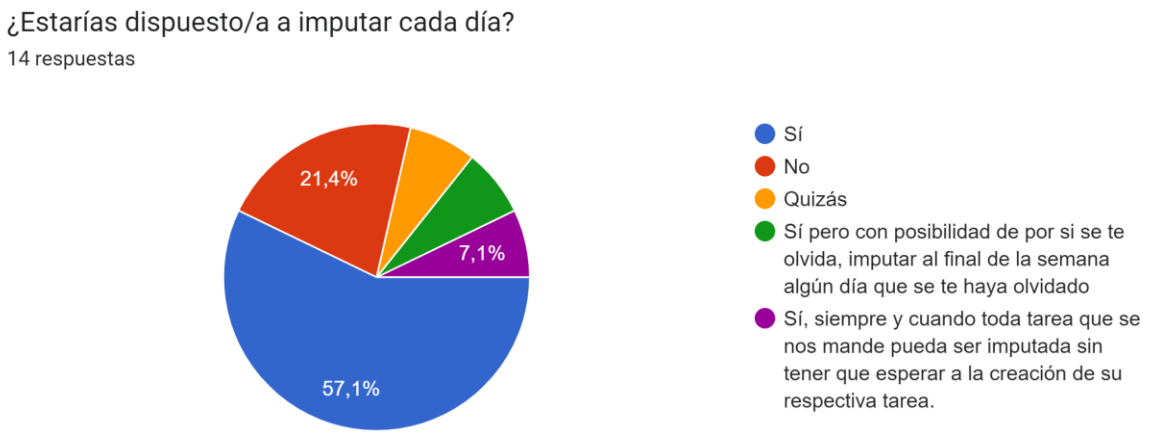
La encuesta se puede consultar en el siguiente enlace: https://docs.google.com/forms/d/e/1FAIpQLScd3B2VzMJ1hAhDijJqcobUjkY-ocQR5J9ltMtqNMIOBEh9Lg/viewform?usp=sf_link

A continuación, se exponen las preguntas que se les han hecho con sus respectivas respuestas.

1.



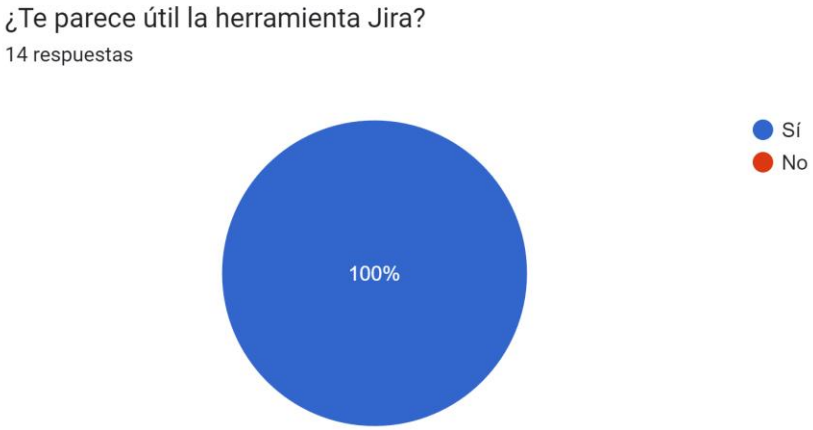
2.



3.


¿Crees que sería de utilidad tener una reunión diaria o a principios de semana con el responsable de proyecto?
14 respuestas
28,6%
7,1%
35,7%
21,4%
Sí, diaria
Sí, a principios de semana
No
Con un par de días a la semana estaría bien
Idealmente una a principios de la semana si. Si el desarrollo tiene complicaciones es neceario mantener la comunicación de manera diaría. A veces no es neceario reunirse y basta con in…

4.


¿Te parece útil la herramienta Jira?
14 respuestas
100%
Sí
No

5.

¿Ves algo que haya que mejorar o simplificar de la forma en la que se trabaja con Jira?

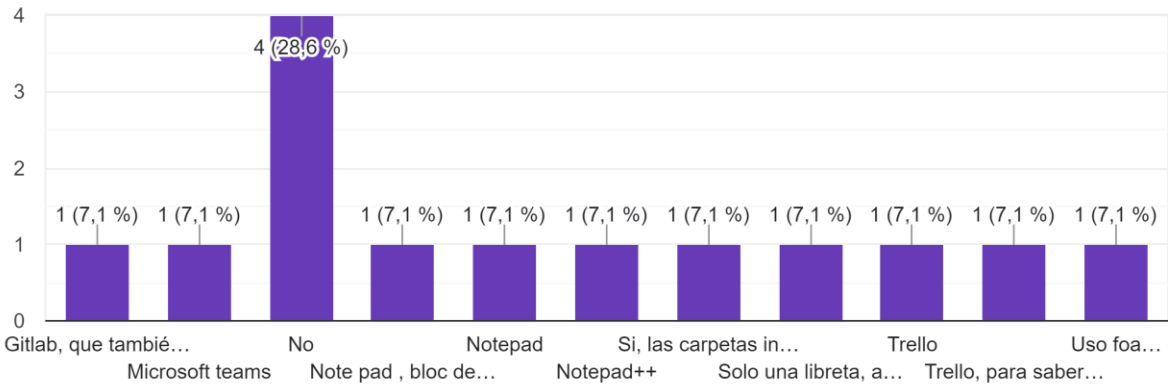
14 respuestas

No
La gestion de archivos adjunto a los tickets
Mínimo de caracteres para los GAP
Poner mas información en los gap e ir actualizándolos con las decisiones que a veces se toman durante el desarrollo
Quizás se podría usar de forma más activa el sistema de control de ramas que integra.
Se aumenta la productividad al quedar toda la información registrada en el mismo sitio y reducirse la cantidad de información que se transmite de forma oral
Actualmente no, el trabajo con Jira es sencillo y cómodo.
Cambiar el tipo de ventana para poder verlo algo mas grande a la hora de la imputación
Para muchos desarrollos falta definición o la redacción es confusa, lo que invita a tener reuniones. Esto resta poder a la herramienta Jira.

6.

¿Utilizas otra herramienta para guardar más información acerca de los desarrollos a medida? Indicar cuál o cuáles

14 respuestas



7.

Si te atascas con un desarrollo, ¿se lo dices al responsable o esperas a que vaya a pedir reporte del estado del desarrollo?

14 respuestas

8.

¿Qué vía de comunicación con el responsable de proyecto ves más óptima?

14 respuestas