E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Análisis de la implantación de la metodología SCRUM y la plataforma TFS en la gestión de un proyecto con integración continua en la empresa ANIMSA

Máster Universitario en
Ingeniería Informática

Trabajo Fin de Máster

Autor: Marilin Vega León

Tutor: José Javier Astrain Escola

Pamplona, 15/03/2020

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Dedicatoria

Perdonad que sea tan repetida, y dedicar este trabajo a los que seguramente dedican todos. Pero tengo una excusa, esto lo dedico a los constantes, a los que no se rinden, a los que han reído y llorado conmigo, a los que me guían desde arriba, a los de abajo por domesticarme. A mi familia y amigos.

"Tienes que actuar y estar dispuesto a fracasar. Si tienes miedo a fracasar, no vas a llegar muy lejos"

Steve Jobs, Fundador de Apple

Agradecimientos

Debo manifestar mi más sincero agradecimiento a todas las personas que durante todo este tiempo me han apoyado.

Gracias a mi tutor, por su paciencia, dedicación, motivación, criterio y aliento. Ha sido un placer y un privilegio contar con su guía y ayuda.

Gracias miles a todo el equipo de ANIMSA, por su colaboración, críticas, amabilidad y ayuda. Sin vosotros no hubiera sido posible la realización de este trabajo. Son un magnífico equipo de profesionales.

Gracias a mi familia y amigos. Unos por empujarme a esta gran aventura que ha sido la realización del máster, y otros por acompañarme durante este viaje tan maravilloso que me ha hecho crecer.

A todos gracias por creer en mí, y hacer fácil lo difícil.

"Sé que suena totalmente loco, pero como nadie es lo suficientemente loco para hacerlo, tendrás poca competencia"

Larry Page, Cofundador de Google

Resumen

En el presente trabajo se analiza y valida la viabilidad y conveniencia de implantar el desarrollo de proyectos mediante Integración Continua y metodología SCRUM en el desarrollo de la empresa ANIMSA (Asociación Navarra de Informática Municipal SA). El objeto del estudio es analizar la factibilidad y el proceso de apropiación de prácticas ágiles, específicamente SCRUM en ANIMSA, así como el uso de la plataforma TFS[1] mediante su implantación en un proyecto concreto de la empresa, permitiendo su gestión durante todo el ciclo de vida del mismo e incluyendo integración continua, con la intención de proponer su implantación. Concretamente, el objetivo es analizar el funcionamiento, factibilidad, e impacto, de la metodología, técnica y herramienta antes dicha y su relación con las prácticas ya adoptadas, desde el punto de vista de los miembros de los equipos, basados en la observación, apropiación y transferencia del conocimiento.

Palabras claves

Metodologías ágiles, Integración Continua, Servidor de Team Foundation, SCRUM.

[1] Team Foundation Server: https://docs.microsoft.com/en-us/azure/devops/server/tfs-is-now-azure-devops-server?view=azure-devops

Abstract

This work will analyze and validate the feasibility and convenience of implementing the development of projects through Continuous Integration and SCRUM methodology in the development of the company ANIMSA (Asociación Navarra de Informática Municipal SA). The object of study is to analyze the feasibility and the process of appropriation of agile practices, specifically SCRUM in ANIMSA, as well as the use of the TFS platform, through its implementation in a specific project of the company, its management throughout the life cycle of the same, and includes continuous integration with the intention of proposing its implementation. Specifically, the objective is to analyze the operation, feasibility, impact, of the aforementioned methodology, technique and tool and its relationship with the practices already detected, from the point of view of the team members, in the observation, appropriation and transfer of the knowledge.

Keywords

Agile methodologies, Continuous Integration, Team Foundation Server, SCRUM.

Índice

Índice de figuras

Índice de tablas

CAPÍTULO 1: INTRODUCCIÓN

1.1 Antecedentes

Como parte de la cumplimentación de las actividades formativas en el Máster de Ingeniería Informática en la Universidad Pública de Navarra (UPNA), se me propone la realización de las prácticas académicas externas en la empresa Asociación Navarra de Informática Municipal S.A. (ANIMSA), con el objetivo de aplicar y complementar los conocimientos adquiridos durante el máster.

Una vez dentro de la empresa, y realizando mis funciones, se me plantea la necesidad de incluir nuevas metodologías, técnicas y herramientas para el departamento de I+D. Por lo que hago una propuesta de uso de metodologías y herramientas agiles para el departamento, centrándome específicamente en la metodología SCRUM y la integración continua (IC) usando Team Foundation Server (TFS).

Aprovechando la oportunidad, decido proponer a la universidad y a la empresa ANIMSA la realización del Trabajo de Fin de Máster dentro de esta entidad y en colaboración con el departamento de "Innovación y Desarrollo" en un proyecto interno del mismo. Y de esta forma poder analizar la efectividad de la propuesta dentro del departamento de I+D de ANIMSA.

Al ser aprobada la propuesta, la realización de prácticas me permite poder evaluar experimentalmente el uso de la metodología SCRUM y la realización de IC usando TFS.

Para situar en contexto seguidamente hago una pequeña descripción de la empresa. ANIMSA es una empresa pública propiedad de entidades locales de Navarra, que tiene ya más de 30 años de experiencia en la gestión municipal de Navarra. Fue constituida en el año 1985 con el objetivo de gestionar directamente todo lo referente a los servicios informáticos de los ayuntamientos de Pamplona y de Lodosa. A posteriori, se han incorporado otros municipios y ya cuenta con 162 entidades en Navarra (Ayuntamientos, Mancomunidades, Agrupaciones), todas ellas entidades públicas. [1]

ANIMSA es el medio para asegurar el correcto funcionamiento, mantenimiento y desarrollo de todo lo que a informática y comunicaciones se refiere para cada una de sus entidades.

Misión:

(…) “prestación de servicios relacionados con las tecnologías de la información y las comunicaciones. Colaboramos activamente en su modernización, integrando tecnología y modelos organizativos con una clara orientación ciudadana.” [2]

Visión:

“Proporcionar los mejores servicios y productos TIC para la transformación digital y gestión inteligente de todas las Entidades Locales Navarras, convirtiéndolas en referentes a nivel nacional.” [2]

1.2 Justificación de metodologías y prácticas ágiles

Es prácticamente inevitable la existencia de riesgos que van surgiendo durante el transcurso del proyecto, y cuando de proyectos de desarrollo de aplicaciones informáticas se trata, estos riesgos pueden crecer considerablemente si no se adoptan metodologías de desarrollo que intenten al menos minimizar sus efectos. En muchas ocasiones no se trata de la necesidad de anticipación a los problemas que puedan surgir, porque muchos de estos momentos son ineludibles, sino de adaptarnos continuamente a ellos.

Hoy en día, las empresas que desarrollan productos de software, no solo para terceros sino para su consumo propio, viven una lucha constante en cuanto a la calidad de los mismos, aun cuando cada vez existen más técnicas y herramientas que ayudan notablemente a evitar inconvenientes dentro de toda la larga cadena productiva de un producto de software.

Hoy nos enfrentamos con una problemática que se deriva precisamente de la alta disponibilidad de diferentes soluciones para encaminar el proceso productivo del software, y es concretamente no saber qué metodología, técnica o herramienta se adapta mejor a nuestro entorno.

Por otra parte, un número considerable de proyectos de software que se están desarrollando a día de hoy, continúan usando metodologías y técnicas obsoletas que dan como resultado que los productos y servicios se entreguen muchas veces con retrasos o falta de la calidad y seguridad requerida.

AMINSA no está exenta de todos estos inconvenientes y entiende que estos problemas afectan directamente a los costos y la calidad de sus productos. Hasta hace relativamente poco tiempo los conceptos como diferenciación, calidad elevada y costes reducidos, eran los pilares en cuanto al desarrollo de software se trata, pero hoy en día la práctica demuestra que también se hace necesaria la velocidad y la flexibilidad.

A todo esto, surge una nueva tendencia en las empresas de diferentes sectores y tamaños a usar metodologías ágiles en sus modelos organizativos, cambiando en gran medida la forma en que gestionan sus proyectos, con equipos que demuestran su evolución a un alto rendimiento.

En el departamento de "Innovación y Desarrollo" de la empresa ANIMSA, el ciclo de vida del software pasa por diferentes entornos como son: desarrollo, pruebas, producción. Al desarrollar un producto dependiente del entorno de trabajo, cuando se desea desplegar en otro entorno como puede ser producción, puede suceder que éste no funcione correctamente. Desarrollar de forma manual los procesos intermedios de pruebas hasta que el software llegue a producción, en muchos casos suele ser engorroso o se dilata en el tiempo, lo cual influye en el coste.

Surge la necesidad de emplear nuevas y mejoradas técnicas, no solo para maximizar la productividad, sino para impulsar el trabajo en equipo.

Por estos motivos, considero la necesidad de plantear una metodología que impulsará la colaboración, el aprendizaje continuo, y que tenga una estructura flexible para soportar cambios o eventualidades durante todo el proceso de creación y evolución de los artefactos creados en la entidad. Vinculado a esto, emerge casi como una obligación el uso de procesos seguidos o continuos, que resuelven en gran medida las problemáticas antes planteadas, por eso planteo el uso de SCRUM e IC.

Pero una vez concebido el camino a realizar, se necesita una herramienta que sea lo suficientemente potente como para llevar a buen término lo antes planteado. Entonces propongo TFS como medio que a priori cumple con los requisitos técnicos que se piden

para desarrollar SCRUM e IC, y porque además, se cuenta con el servidor, licencias y ambiente técnico en general para su puesta en marcha.

Aprovechando la cobertura de la necesidad de modificar y ampliar un producto interno de la empresa, éste se toma como objeto de estudio y análisis para identificar la evolución e implementación del mismo (en adelante me referiré como proyecto de demostración). Por ende, esta investigación se realizará con el fin de identificar el aporte que metodologías agiles como SCRUM y la técnica de IC mediante la herramienta TFS, pueden proporcionar a los proyectos de este departamento tomando como ejemplo de prueba uno de ellos.

1.3 Objetivos

Para concluir, si es posible la implementación y adopción de la propuesta, el objetivo principal a cumplir en este Trabajo de fin de Máster es el análisis de la metodología ágil SCRUM y el uso de la IC con TFS, mediante su funcionamiento, factibilidad e impacto dentro de un proyecto de software en la empresa ANIMSA.

Este objetivo principal lleva a los siguientes sub-objetivos, que se muestran de forma estructurada para identificar y puntualizar mejor cada uno de ellos.

- Análisis de la organización y estructura de ANIMSA.
- Análisis de la implantación de la metodología SCRUM para el entorno de la empresa ANIMSA.
- Análisis de la gestión de SCRUM con TFS para la dirección y administración de proyectos en el departamento de I+D.
- Análisis de la metodología SCRUM para realizar IC con la herramienta TFS en los proyectos de desarrollo del departamento de I+D.
- Formación y Transferencia del conocimiento al departamento.
- Análisis de la apropiación por parte del departamento.

Es importante enfatizar que este estudio no se basa en el desarrollo de una aplicación, sino en el análisis de la implantación de las metodologías y procesos ya enunciados anteriormente, evidenciando posibles aportaciones al departamento de Innovación y

desarrollo de la empresa ANIMSA, demostrando así la importancia de desarrollar este trabajo.

1.4 Estado del Arte

A través de los años el proceso de desarrollo de software ha venido evolucionando metodologías que facilitan su programación. No obstante, en la década del 50 parecía que los programadores estaban más enfocados en la tarea de codificar, en vez de comprender y recoger las necesidades de los clientes que a menudo reflejaban su descontento con los resultados. Se desarrollaba prácticamente de una manera empírica, lo que conllevó que una buena cantidad de proyectos fallaran y fueran a la quiebra. A raíz de todo esto se empieza a comprender la necesidad de hacer software con calidad. La manera de lograrlo fue la adopción de modelos y metodologías clásicas que progresivamente fueron incorporando nuevos estándares y controles.

Pero la evolución no se detuvo. Fue entonces, con la llegada de Internet, que surgió toda una revolución de proyectos que se caracterizaban por entornos cambiantes y tiempos cortos de entrega, para los que las metodologías existentes no se adaptaban idóneamente.

Es entonces y ante los problemas que se venían formulando constantemente con el uso de las ya conocidas metodologías tradicionales, partiendo de las dificultades en cuanto al tiempo y la flexibilidad que planteaba la comunidad dirigida al desarrollo de software, que emergen como una necesidad las metodologías agiles, respondiendo especialmente a proyectos más bien pequeños, permitiendo una solución a medida para este tipo de entornos.

La siguiente gráfica Fig.1, muestra una línea del tiempo en la que se indica cómo ha sido la trayectoria de las metodologías de software a lo largo de los años.

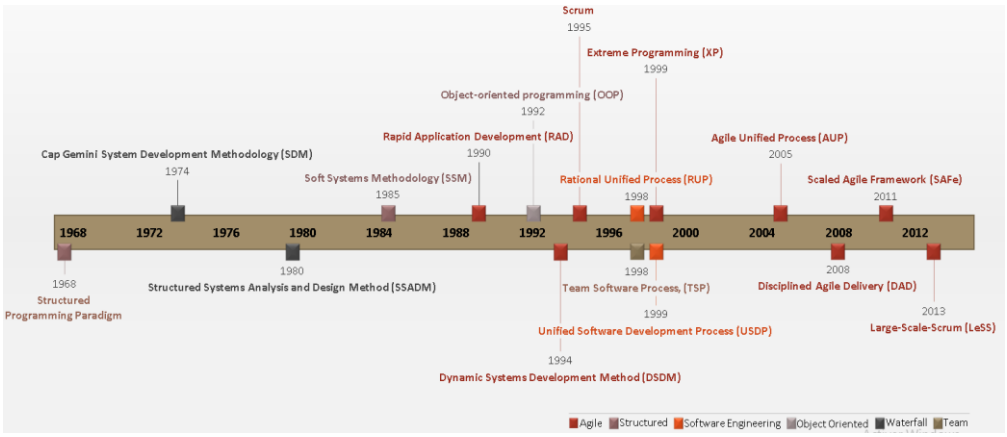


Fig. 1 Línea del tiempo del desarrollo de las metodologías de software [3]

En definitiva, aunque no se puede afirmar que un método es mejor que el otro, al menos parece que hay una tendencia al uso de metodologías ágiles por parte de las empresas. En el capítulo dos abordo con más detalles esta comparativa.

1.4.1 Desarrollo Ágil

No es hasta el año 2001 tras una reunión donde participaron un grupo de expertos, que nacen los métodos agiles y es cuando se publica el Manifiesto Ágil 2001 [4] , y enuncian así nuevas formas de crear software.

Fig. 2 Referencia al Manifiesto Ágil [5]

Los doce principios del Manifiesto Ágil [4] son:

1. Nuestra máxima prioridad es satisfacer al cliente a través de la entrega temprana y continua de software valioso.
2. Bienvenido a los requisitos cambiantes, incluso tarde en el desarrollo. Los procesos ágiles aprovechan el cambio para la ventaja competitiva del cliente.

3. Entregue software de trabajo con frecuencia, desde un par de semanas hasta un par de meses, con preferencia al menor tiempo.
4. La gente de negocios y los desarrolladores deben trabajar juntos a diario durante todo el proyecto.
5. Desarrollar proyectos en torno a personas motivadas. Bríndeles el entorno y el apoyo que necesitan, y confíe en ellos para hacer el trabajo.
6. El método más eficiente y efectivo para transmitir información a un equipo de desarrollo y dentro de él es la conversación cara a cara.
7. El software de trabajo es la medida principal del progreso.
8. Los procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deberían poder mantener un ritmo constante indefinidamente.
9. La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
10. La simplicidad, el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de equipos auto organizados.
12. A intervalos regulares, el equipo reflexiona sobre cómo ser más eficaz, luego ajusta y ajusta su comportamiento en consecuencia.

"*Agile development combines creative teamwork with an intense focus on effectiveness and maneuverability*." [6]

Traducción:

El desarrollo ágil combina trabajo en equipo creativo con un intenso enfoque en la efectividad y maniobrabilidad.

Situación Actual

El uso de metodologías ágiles es sin duda un reto también para las organizaciones que decidan usarlo, existen muchos estudios que evidencian estos desafíos. [7]

Teniendo en cuenta el estudio realizado por *Standish Group International*[2] en el informe de CHAOS del año 2015, donde presenta las cifras de éxito y fracaso de 50.000

[2] Reconocida firma internacional independiente de asesoría en investigación en Tecnologías de la Información

proyectos de IT referentes de diversas partes del mundo y centra su estudio dentro de los años fiscales 2011 a 2015. Se obtienen los resultados que se muestran en la Fig. 3.

MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

Fig. 3 La resolución moderna (OnTime, OnBudget, con un resultado satisfactorio) [8]

Es necesario aclarar que el estudio es realizado mediante una definición *OnTime*, *OnBudget*, con resultado satisfactorio. En otras palabras, manifiesta que el proyecto se ha realizado en un tiempo estimado moderado, con un presupuesto ajustado, dando el agrado y aprobación del cliente esperado. [8]

Una comparativa realizada en el informe de CHAOS 2016 [9] sobre el éxito de metodologías ágiles en proyectos pequeños evidencia claramente el éxito que poseen las mismas.

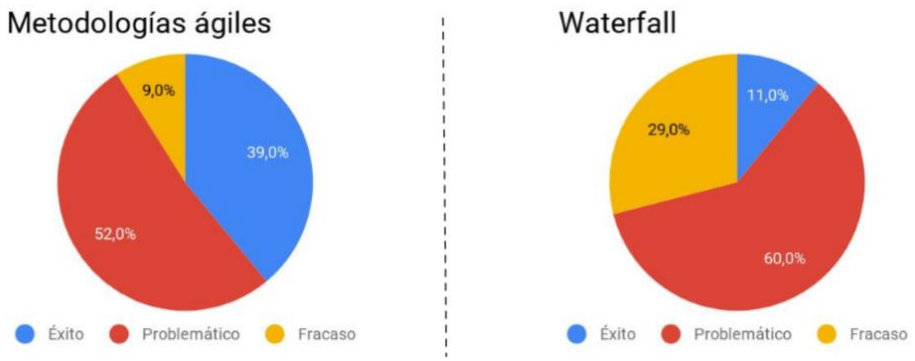


Fig. 4 Conclusiones CHAOS 2016 [9]

En la Fig.4, se muestra que solo el 9% de los proyectos agiles han fracasado, el 52% han sido problemáticos y el 39% de ellos un éxito. Paralelamente, los proyectos desarrollados con metodologías en cascada han tenido un 29% de fracaso, un 60% con problemas y un 11% de éxitos.

En 2017 se realiza un estudio [7] con un panel formado por 26 expertos que trabajan en 19 empresas distintas ubicadas en Alemania y Suiza, con gran experiencia en el sector

tanto en enfoques secuenciales como en ágiles, para determinar las dificultades del uso de metodologías ágiles, y arroja el siguiente resultado.

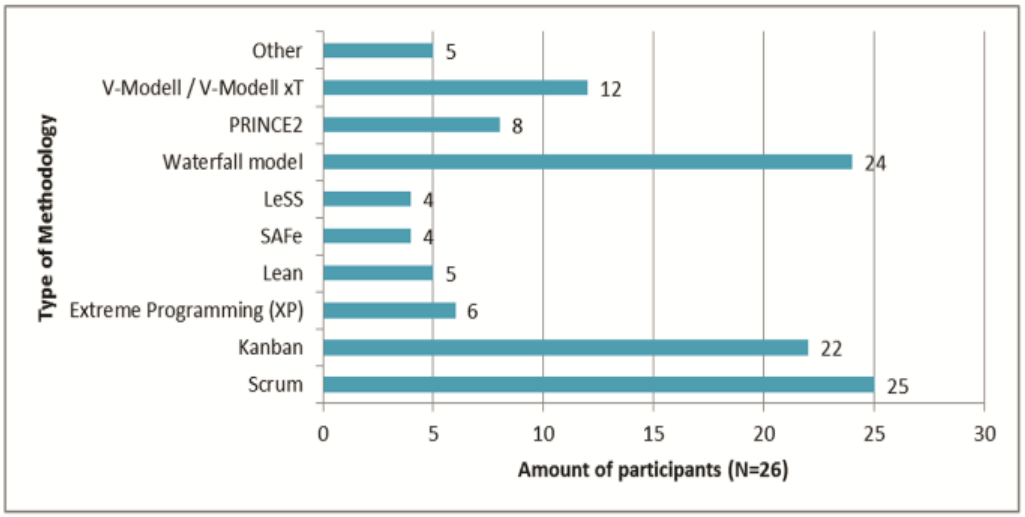


Fig. 5 Modelo de procesos usado por los expertos [7]

En la Fig.5, se evidencia un uso bastante extendido de varias de estas metodologías.

Más recientemente, en el año 2018, una investigación efectuada por *CollabNote VersionOne*[3] [10], referente a los beneficios de la adopción de uso de metodologías agiles arroja el siguiente resultado que se muestra en la Fig.6**.**

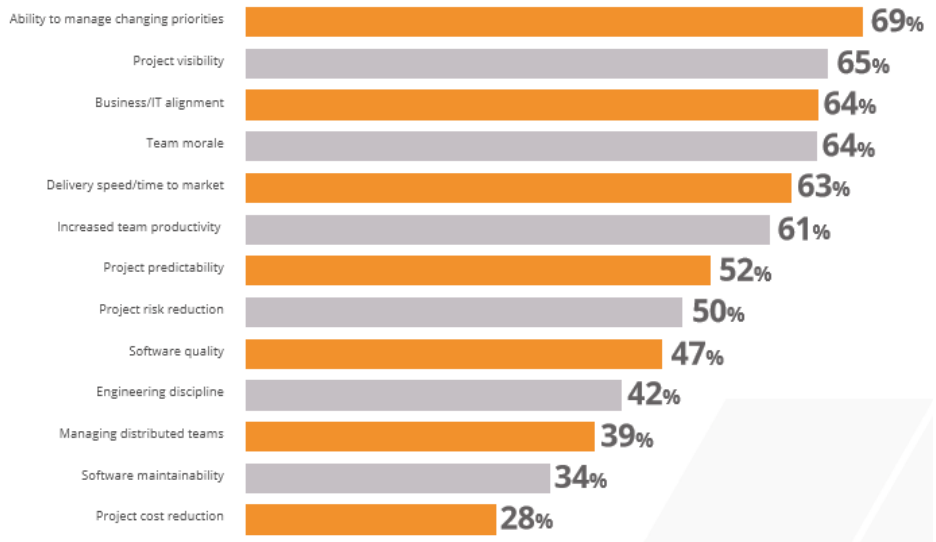


Fig. 6 Beneficios de adopción de metodologías agiles año 2018 [10]

[3] Firma de software dedicada a creación de productos y servicios en la rama de la industria de gestión de flujo de valor, gestión ágil, gestión del ciclo de vida de la aplicación y control de versiones empresariales.

Se demuestran muchos beneficios conseguidos por las empresas que adoptan metodologías agiles, y en contraste con el año pasado se produce una mejora en cuanto a moral del equipo se refiere de un 64% en comparación con el 61% del año anterior. Asimismo, aumenta de un 49% a un 52% respecto a predicción del proyecto, y lo mismo sucede con reducción en el riesgo del proyecto que aumenta a un 50% comparado con 47% el año anterior. [10]

Las encuestas arrojan que el 46% de los encuestados está utilizando prácticas ágiles para gestionar proyectos de desarrollo subcontratados y el 40% planean aumentar el uso de ágil en proyectos de desarrollo subcontratados en los próximos 24 meses. [10]

Según [10] la disminución de costos ha ganado importancia, ya que hubo un aumento del porcentaje de encuestados que lo relacionan como una buena razón para adoptar metodologías ágiles como un beneficio reportado.

Aunque todo parece indicar que la tendencia hoy día está enfocada en el desarrollo ágil, es importante aclarar que no existe una metodología mejor o peor que la otra. Precisamente porque no existe una metodología universal para hacer frente a todos los problemas. Cada una aporta ventajas que pueden ser aprovechadas, y debe ser aplicada de acuerdo al contexto del proyecto.

Entre todo el compendio de metodologías agiles, una de las más usadas es SCRUM, a continuación, veremos un poco de su historia.

1.4.2 SCRUM

SCRUM tiene sus inicios en el año 1986, que es cuando surge su idea original por los investigadores Nonaka y Takeuchi. Originalmente se basa en el papel de los equipos, de una manera auto gestionada e incentivada para abordar el desarrollo de sistemas de difícil complejidad con una visión global y solapando las fases de desarrollo [11].

Pero no es hasta el año 1995 [12] incluso antes del manifiesto ágil, que se presenta como una metodología de desarrollo de software, introducida en un inicio por Ken Swaber y Jeff Sutherland, autores además de “La Guía de SCRUM”. Esta Guía contiene la definición de SCRUM, los roles, eventos y artefactos de SCRUM, y las reglas que los relacionan.

Situación Actual

En la encuesta realizada por VersionOne en el año 2011 [13] con 6.042 respuestas, refleja que la metodología más usada con diferencia es SCRUM, que va en aumento con relación a años anteriores. La Fig.7 muestra este resultado.

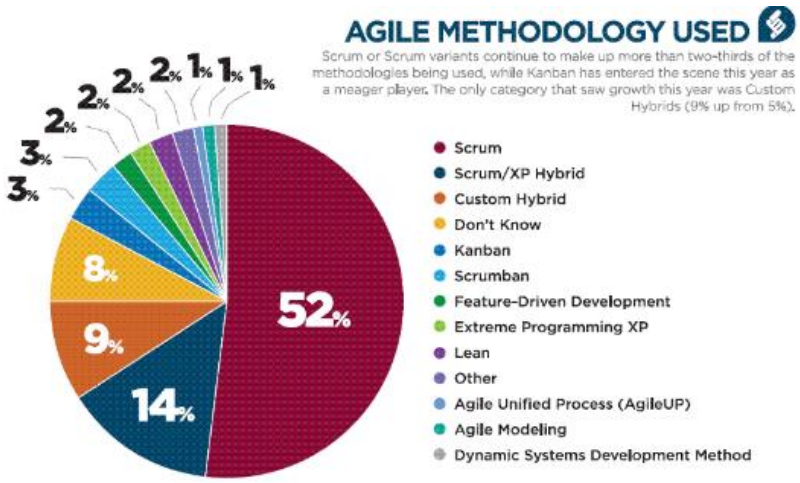


Fig. 7 Metodologías ágiles más utilizadas [13]

El informe de la decimotercera encuesta anual sobre el estado de Agile arroja que SCRUM es nuevamente la metodología ágil más ampliamente utilizada con al menos el 72% de los encuestados que usan SCRUM o un hibrido que lo incluye.

Scrum y Scrum / XP Hybrid continúan siendo las metodologías ágiles más comunes utilizadas por las organizaciones encuestadas. En la Fig.8 se muestra un gráfico con los datos del estudio. [10]

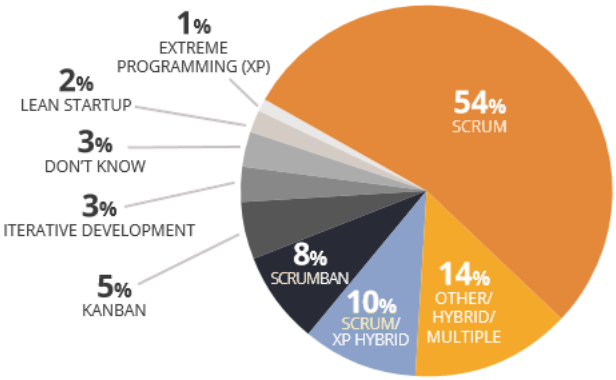


Fig. 8 Uso de metodologías ágiles más comunes utilizadas por las organizaciones encuestadas [10]

La adopción de SCRUM y los SCRUM masters están promoviendo un cambio significativo a nivel de empresa a medida que su papel madura cada vez más en las organizaciones. Así se puede evidenciar que su uso va en aumento a medida que pasa el tiempo respecto a años anteriores.

Como apoyo a la gestión interactiva de proyectos surgen herramientas como TFS, a continuación, se describe su evolución.

1.4.3 TFS

Team Foundation Server es presentado en 2005, en una época donde se empezaban a buscar herramientas que apoyaran la gestión del software. Surge entonces como una solución integrada para cubrir la gestión del ciclo de vida de las aplicaciones. Tuvo muy buena aceptación por buena parte de las organizaciones que ya trabajaban con .Net. Pero su evolución no paró, hasta que prácticamente cada año Microsoft lanza una nueva versión que goza de mejoras y nuevos productos. En la Tabla 1 se muestra esa evolución:

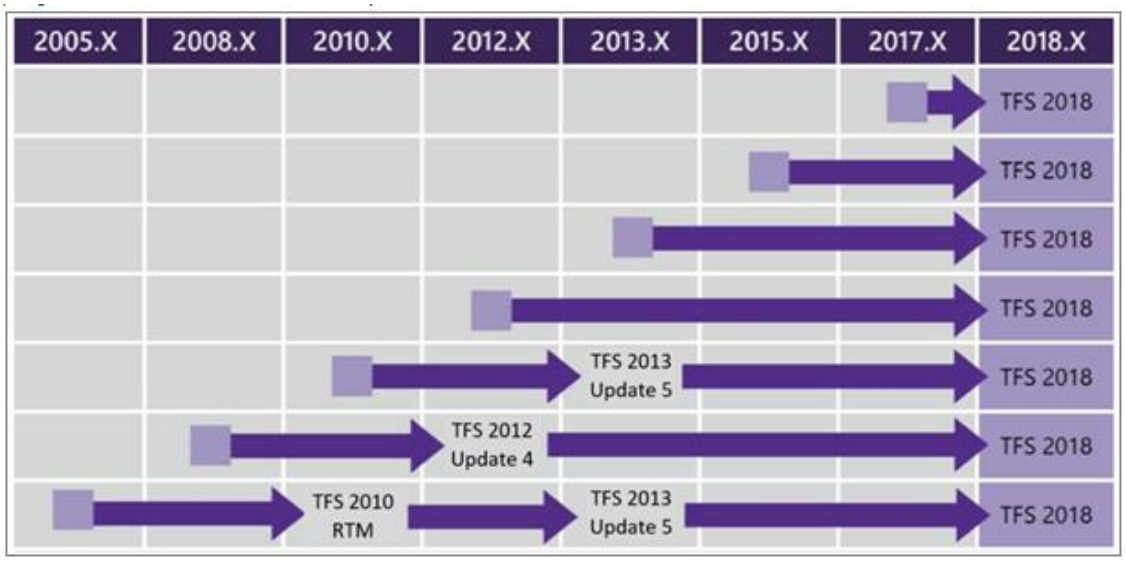


Tabla 1 Matriz de actualización de TFS [14]

La primera versión de TFS se lanzó el 17 de marzo de 2006 por Microsoft, creado en un inicio para el control de versiones de código fuente. A día de hoy es mucho más que eso. Su alcance cada vez es mayor, desde la gestión completa del ciclo de vida de las aplicaciones, comenzando por la fase de análisis y diseño hasta pruebas, y con la facilidad de IC y la calidad del código. Está disponible en dos formas, local y en línea.

Ambas versiones ofertan un entorno integrado y colaborativo, pero presentan diferencias. La oferta en local se basa en un servidor en local que las empresas generalmente eligen cuando necesitan que sus datos permanezcan en un entorno controlado dentro de su propia red. La oferta en la nube proporciona un servicio alojado escalable y disponible desde cualquier parte del mundo.

Situación Actual

Microsoft ha ido evolucionando sus propuestas y ha convertido a TFS como la oferta en local de *Azure DevOps*, incluyendo *Build* (Construcción) y *Release* (Lanzamiento) pidiéndose instalar y ser administrado en sus propios servidores.

Con *Azure DevOps Server* 2019 se cambia el nombre de Visual Studio *Team Foundation Server* a *Azure DevOps Server*, pero no es solo un cambio de nombre, viene con gran cantidad de nuevas y mejores características para trabajar desde la nube.

Entre las características más atractivas de este producto se encuentra su capacidad para la realización de Integración Continua / Entrega Continua / Despliegue Continuo (IC/EC/DC). En el siguiente apartado se relata brevemente el desarrollo y evolución de una de estas prácticas, concretamente la IC.

1.4.4 Integración Continua

La integración continua tiene sus orígenes hacia el año 1989 mediante una investigación denominada “Infuse Enviroment” [15] realizada por los profesores Dewayne E. Perry, Gail Kaiser y WM Schell. Se trataba de un entorno de desarrollo para la evaluación de proyectos y la coordinación del cambio en toda la fase de mantenimiento.

En 1994 se usa el término de IC en Análisis y Diseño orientado a objetos por el diseñador de software Grady Booch, exponiendo cómo al desarrollar mediante micro procesos se podía dar cierre a cada uno mediante esta técnica.

Al aparecer el concepto de *Extreme Programming* en 1997 surge IC como una de sus prácticas, y un año después el ingeniero de software Kent Bech realiza una publicación del tema, que más adelante en 1999 sería más elaborada al publicarse oficialmente el primer libro completo de *Extreme Programming*. [16]

No obstante, aunque esos fueron sus orígenes, la IC es una práctica que no está ligada a una metodología en concreto. El ingeniero de software británico Martin Fowler se considera el impulsor de este modelo informático, al que continuamente se le han ido realizando más aportaciones.

Situación Actual

La tendencia es a que, cada vez más, las organizaciones comprendan la necesidad de implementar IC, aunque mantienen dificultades sobre todo para entender dónde y cuándo es el mejor lugar y momento para comenzar.

En el estudio realizado por COLLABNET VERSIONONE [10], se destaca la IC como la tercera práctica de ingeniería más usada, ver Fig.9.

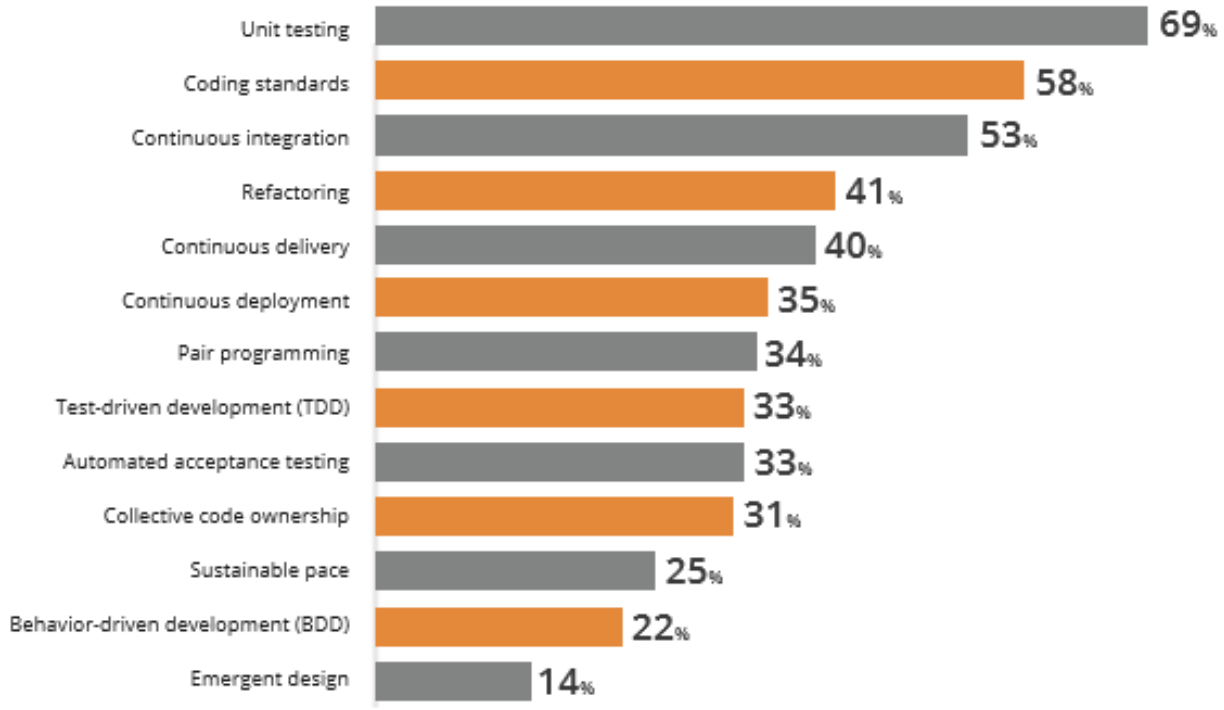


Fig. 9 Uso de prácticas de ingeniería [10]

En el mismo estudio se obtuvo la información del uso general de las herramientas y sus preferencias. Los encuestados declararon usar herramientas de aceptación automatizada con un 39% más en comparación con el 36% del año pasado. Además, declararon que planean usar herramientas de gestión de proyectos ágiles en el futuro, con un 12% este año en comparación con el 9% del año pasado. [10]

A continuación se muestra en la Fig.10, que las herramientas de IC evidencian un aumento entre el año 2018 y 2017 en cuanto a su uso futuro, encontrándose entre las 10 más usadas.

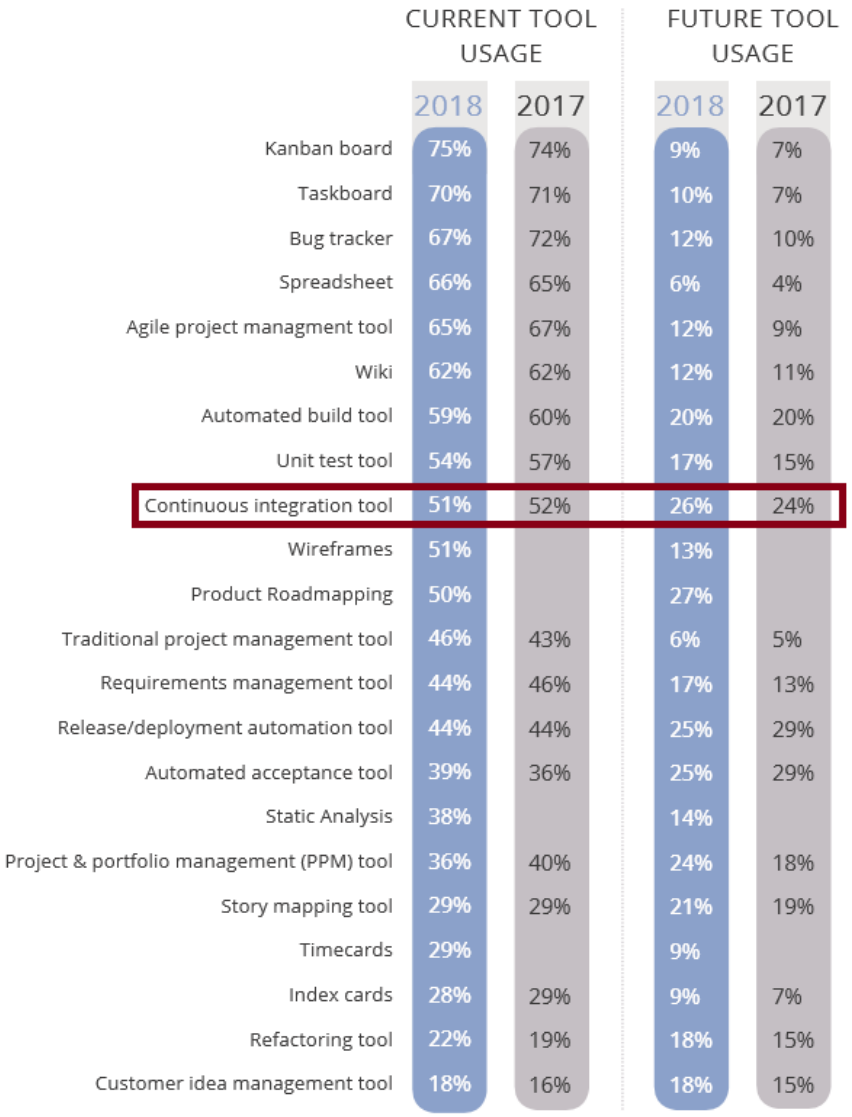


Fig. 10 Uso de herramientas ágiles de gestión de proyectos [9]

1.5 Estructura de la memoria

La memoria está compuesta por 5 capítulos que se detallan a continuación.

El capítulo 1 de introducción es el punto de partida donde se sitúa al lector en la temática a abordar, mostrando los antecedentes, justificación y objetivos del trabajo de fin de master. Por último, se muestra un estado del arte referente al tema abordado, dando a conocer todo el conocimiento acumulado en estas áreas.

Seguidamente, en el capítulo 2 se analizan la metodología, las técnicas y herramientas agiles que se proponen en el TFM. Esto se hace con el objetivo de situar al lector en el contexto para que pueda entender sus características y uso. Para un mayor entendimiento se realiza una comparativa de la propuesta con otras similares, y de esta forma se justifica su uso.

En el capítulo 3 se da a conocer el diseño y desarrollo de la propuesta. Se comienza por explicar el procedimiento para la ejecución del trabajo de fin de máster. Posteriormente se analiza la situación actual en la empresa ANIMSA. Y se concluye detallando la manera en la que han sido empleadas las técnicas y herramientas ágiles que se proponen.

Después de planteada la propuesta y su forma de ser ejecutada, en el capítulo 4 se hace un estudio de la implantación referente al diseño y desarrollo propuesto en el capítulo 3. Para esto se hace un examen detallado para extraer resultados de la conveniencia o no de su uso dentro de la entidad.

Para completar el estudio se presenta en el capítulo 5 las conclusiones del análisis de los datos obtenidos durante toda la investigación. Finalmente, se detallan las líneas de trabajo para posibles investigaciones de interés en el futuro.

CAPÍTULO 2: METODOLOGÍA SCRUM Y HERRAMIENTAS DE IC

En general, las metodologías ágiles están diseñadas de manera que permitan dividir el software en pequeños entregables y estas partes puedan irse mostrando al cliente, lo que permite una retroalimentación e implicación del cliente en el proyecto y a su vez el trabajo y la comunicación más accesible en el equipo. Se distingue por formarse de una gran cantidad de ciclos que tienen como característica distintiva que son relativamente cortos, y permiten de manera natural la IC, influyendo así positivamente no solo en la detección temprana de errores, sino también en poder adaptarse a los cambios de manera sencilla.

A pesar de que las metodologías ágiles han respaldado el manifiesto ágil y conjuntamente se corresponden con los principios anteriormente listados, cada una tiene características propias y distintivas. El uso de una u otra depende de cada proyecto y de sus características, que podrán variar indistintamente ya sea por el número de integrantes, el carácter del producto, el tipo de cliente, etc. Dentro del compendio de metodologías ágiles se encuentra SCRUM.

2.1 SCRUM

"Scrum: Un marco de trabajo por el cual las personas pueden acometer problemas complejos adaptativos, a la vez que entregar productos del máximo valor posible productiva y creativamente. (…) no es un proceso o una técnica para construir productos; en lugar de eso, es un marco de trabajo dentro del cual se pueden emplear varias técnicas y procesos. (…) consiste en los Equipos Scrum, roles, eventos, artefactos y reglas asociadas. Cada componente dentro del marco de trabajo sirve a un propósito específico y es esencial para el éxito de Scrum y para su uso." [17].

Existe una teoría de control de procesos empírica o empirismo en la cual SCRUM se basa. Dicha teoría afirma que para obtener conocimiento se debe partir de la experiencia, y de la toma de decisiones desde lo que se conoce. [17]

Valores y principios básicos de SCRUM: [18]

(…)

● Pequeños equipos de trabajo auto-organizados, buscando maximizar la comunicación y el intercambio de conocimiento tácito e informal, y minimizar el sobreesfuerzo.

● Adaptación a las solicitudes de cambios técnicos o cliente / servidor, lo que garantiza la entrega del mejor software posible.

● Entregas frecuentes de versiones que pueden ser probadas, ajustadas, implementadas, documentadas y publicadas para la producción.

● La división del trabajo y las responsabilidades de equipo del proyecto para pequeñas entregas.

● Capacidad para ofrecer software listo cuando lo necesite el cliente o negocio.

(…)

2.1.1 Los Roles en SCRUM

***Team*:** Un Equipo en SCRUM se compone por el *Product Owner, Development Team y Scrum Master*. Son auto-organizados, auto-gestionados y versátiles pudiendo realizar múltiples funciones. Si pensamos en flexibilidad, productividad y creatividad pensamos en un equipo SCRUM, son tres palabras que definen íntegramente el modelo del equipo SCRUM. La entrega de productos de este equipo se realiza de manera iterativa e incremental, permitiendo de esta forma que en todo momento se encuentre una versión que sea funcional y operativa del producto realizado, lo que a su vez posibilita que haya una especie de *feedback* para mejoras futuras. [11]

***Product Owner*:** Dueño del producto, es la persona encargada de incrementar la productividad del equipo de desarrollo y la valorización del producto. Es también el responsable de administrar y manejar el *Product Backlog*. Para que todo fluya de la manera debida el equipo debe respetar el juicio y la toma de decisiones del dueño del producto, que son reflejadas con posterioridad en el *Product Backlog*. [11]

***Development team*:** Se refiere al equipo de desarrollo, encargados de hacer crecer el producto luego de cada sprint. Se caracterizan por ser auto-organizados y multifuncionales, aunque cada miembro pueda tener una especialidad determinada todos son catalogados como desarrolladores y por ende las cargas de trabajo recaen totalmente en el equipo. El tamaño de los equipos de desarrollo puede variar dependiendo del

proyecto, pero siempre que se cumpla que sean tan reducidos como para mantenerse ágiles y al mismo tiempo que puedan cubrir todo el trabajo sin carencias. [11]

***SCRUM Master*:** Se considera como "un líder que está al servicio del equipo SCRUM" [11]. Comprueba y verifica que la metodología es adquirida y aplicada como corresponde y se describe en las buenas prácticas de SCRUM.

2.1.2 Eventos de SCRUM

***Sprint*:** Es contemplado como el evento principal de SCRUM, en concreto es un intervalo de tiempo variable (nunca mayor de un mes) en el cual se crea un aumento del producto que a su vez estará listo para utilizar. [11] Se considera que el tiempo debe ser limitado de esta manera para así minimizar los riesgos y poder visualizar con mayor claridad el avance del proyecto, sin que el proceso se vuelva complejo.

Se compone de:

***Sprint Planning*:** Reunión colaborativa donde queda plasmado todo el trabajo a realizar durante el sprint.
***Daily Scrum*:** Reunión de 15 min. máximo para coordinar el trabajo realizado y el que está por realizar durante las próximas 24 horas.

***Sprint Review*:** Se realiza al concluir cada sprint, analizándose el cumplimiento o no de las tareas propuestas, y de esta forma verificar el incremento del producto y si fuera el caso adaptar la lista del producto.

***Sprint Retrospective*:** Se considera un *feedback* o retroalimentación dentro del equipo donde se analizan posibilidades de mejoras para el siguiente sprint, de manera que se denota una constante adaptación del equipo para cambiar y mejorar en todo momento.

2.1.3 Framework SCRUM

El siguiente gráfico Fig.11, nos muestra SCRUM desde la planificación hasta la entrega del software.

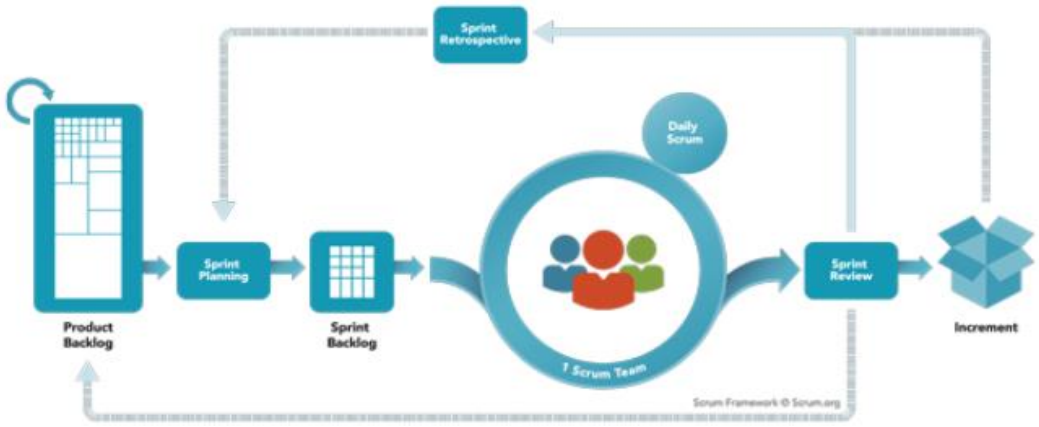


Fig. 11 Framework de SCRUM [19]

El cliente (partes interesadas) define los requerimientos (*Product Backlog*). Éste es un documento de alto nivel que contiene las funcionalidades, descripciones de los requisitos, etc. Todas ellas con un nivel de prioridad que define cómo se toman en cuenta en cada reunión de planificación del sprint para seleccionar el trabajo a realizar. La reunión de planificación se realiza en un ciclo que como norma general oscila de 15 a 30 días, depende del proyecto.

De la planificación del sprint queda constituido el *sprint Backlog*, que consiste en un documento detallado donde se especifica cómo serán implementados los requisitos durante todo el sprint.

El equipo de desarrollo se encargará de implementar las tareas del *sprint Backlog*, así como estar presente en cada una de las reuniones diarias de Scrum, revisión del sprint, y Retrospectiva del sprint.

Al final de cada sprint se debe evidenciar un crecimiento del producto y mejoras del proceso. Sucesivamente, se realiza el ciclo hasta que el *Product Backlog* se agote.

2.2 TFS

Team Foundation Server es un producto de Microsoft que se integra con el IDE de desarrollo Visual Studio, lo que posibilita considerablemente el manejo de los elementos de trabajo para su seguimiento. Crea un ambiente colaborativo para el equipo pudiendo obtener y compartir documentos, informes, y todo lo referente al proyecto.

Características y novedades [20]:

- Paneles que pueden ser configurables por el usuario
- Control de fuente de dos tipos:
    - Git (distribuido)
    - TFVC[4] (centralizado)
- Planificación y seguimiento del trabajo
    - Administración agrupada (epopeyas contienen características y características contienen historias)
    - Gestionar elementos de trabajo
    - Planificación de sprint
    - Tableros KANBAN para visualizar el flujo de trabajo
- Integración continua y despliegue
- Pruebas continuas, manuales y exploratorias
- Servicios de colaboración
- Ganchos de servicio
- Servicios alojados en la nube basados en el uso y servicios en la nube de Azure
- Servicios Administrativos

2.2.1 Sistema de control de versiones

Como parte de la garantía de fiabilidad y seguridad del código, es guardado en una base de datos de SQL. Es un sistema centralizado, pero posibilita la distribución en ramas que más tarde podrán combinarse. No es necesario estar conectados al servidor para poder trabajar, los conflictos son tratados en el momento del *check-in*.

[4] Team Foundation Version Control

2.2.2 Elementos de trabajo

TFS dispone de tres plantillas de trabajo (CMMI, Agile, SCRUM), cada una brinda sus tipos de elementos de trabajo. No obstante, se pueden establecer sus propias plantillas customizadas para poder adaptarse a su propia metodología de trabajo.

Los elementos de trabajo (Prueba, Error, Requisito, Caso de uso, otros) se pueden corresponder al código fuente, de tal manera que se pueda cubrir el código con una tarea determinada y se puede realizar directamente desde el propio Visual Studio. [21]

2.2.3 Gestión de informes

Entre las características más atractivas de TFS se presenta la generación de informes (SQL Server Reporting Services, Exel, otras).

Los servicios de informes pueden ser creados y modificados usando Report Builder. Abarcan desde la gestión de proyectos, compilación, resultados y proceso de pruebas realizadas, informes ágiles, hasta informes más especializados para pruebas de carga.

2.3 Integración Continua

Hoy en día existe una amplia documentación sobre IC, pero cabe destacar que se encuentran divergencias en cuanto a dónde empieza y termina su alcance. Para definirla se hará uso de la definición realizada por la empresa ATLASSIAN[5], líder en su sector.

(…) “es la práctica de automatizar la integración de los cambios de código de múltiples contribuyentes en un solo proyecto de software. El proceso de IC se compone de herramientas automáticas que afirman la corrección del nuevo código antes de la integración. Un sistema de control de versión de código fuente es el quid del proceso de IC. El sistema de control de versiones también se complementa con otras verificaciones como pruebas de calidad de código automatizadas, herramientas de revisión de estilo de sintaxis y más. ” [22]

[5]Empresa de software que crea productos para empresas y desarrolladores de software en particular.

2.3.1 Integración continua, Entrega continua y Despliegue continuo

Es importante definir la diferencia que existe entre Integración continua, Entrega continua y Despliegue continuo para situarnos en el contexto de implementación del TFM. La definición que se tomará es la realizada igualmente por ATLASSIAN. [22]

Integración continua: Primera fase que cubre todo el proceso del grupo de desarrolladores que pretenden integrar los cambios del código en local con el repositorio principal de la solución e integran a este proceso pruebas automatizadas.

Entrega continua: Segunda fase, basada en la entrega mediante el empaquetado del artefacto para hacer entrega al usuario final. Permite que se pueda generar la aplicación mediante herramientas que lo permitan formar de manera automatizada. Luego de esta fase el artefacto estaría preparado para poder ser desplegado en todo momento ante los usuarios.

Despliegue continuo: Última fase donde básicamente se implementa el proceso mediante el inicio y distribución de forma automática del artefacto a cada uno de los usuarios finales. Llegados a este punto se implementa y distribuye de manera automática el artefacto.

2.3.2 Arquitectura

En la fig.12 se muestra la arquitectura de la integración continua, evidenciando cada una de sus fases.

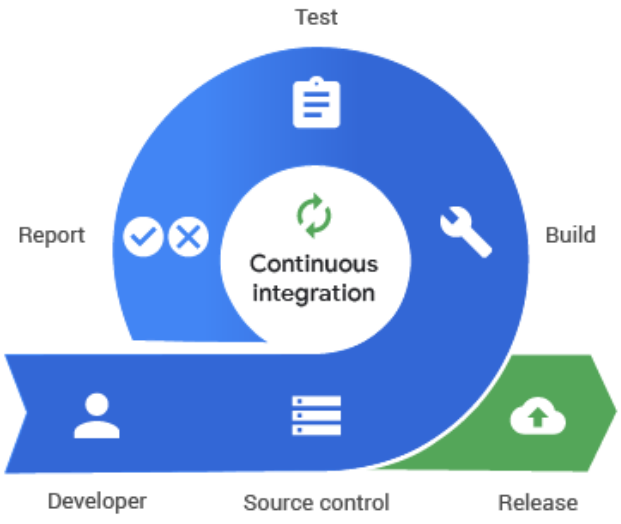


Fig. 12 Fases de integración Continua [23]

2.4 Comparativas

Es importante partir de la idea de que no se trata de definir cuál es mejor o peor. Un contexto bastante usado es el de la comparación entre el cuchillo y el tenedor. ¿Cuál es mejor? Pues no existe una mejor respuesta, surgirán muchos "Depende", y eso viene relacionado con la idea de que para poder seleccionar una u otra herramienta dependerá siempre del contexto.

2.4.1 Metodologías

Actualmente considero que entre las metodologías más usadas por la comunidad del desarrollo software se pueden evidenciar:

- XP
- KANBAN
- SCRUM

Para la selección de la Metodología parto de un análisis de las tres anteriores, y comienzo conociendo sus características, para la elección de la apropiada.

En dicho análisis obtengo los siguientes resultados:

SCRUM vs KANBAN

- La metodología SCRUM regula, establece y formula todo su proceso mucho más claro que KANBAN. Esta última no define roles, ni estructura de los equipos, aun cuando se pueden incorporar iteraciones no las prescribe.
- Una de las características más usadas de KANBAN es su famoso tablero empleado como técnica de visualización. Aunque no en todos los contextos pero de manera general esta metodología tiene como requisito que los equipos deban estar situados en la misma localización de trabajo. En este sentido SCRUM ha adoptado esta práctica usando un tablero donde se encuentran todas las tareas a realizar en el sprint y de esa forma medir su avance.
- KANBAN aplica un límite de la regla de trabajo en su progreso, lo que es más adecuado para equipos que posean recursos más limitados, mientras que SCRUM restringe el cúmulo de trabajo a realizar en cada sprint.

- Considero que si se desea ampliar el proceso, SCRUM es más adecuado que KANBAN, esto se debe a los sprints sucesivos que mejoran la salida y la velocidad del equipo, lo que facilita su predicción, serenando a las organizaciones más grandes.

SCRUM vs XP

- La metodología XP centra más su atención en la ingeniería, mientras que SCRUM lo hace en la productividad.
- En relación al trabajo, XP determina un conjunto de prácticas básicas que, de ser una persona sin mucha experiencia en proyectos ágiles, puede parecer angustioso. En cambio, SCRUM deja a los equipos que determinen con libertad cómo realizar el trabajo.
- Ambos requieren de una reunión diaria de base.
- Son muy similares los roles de propietario del producto en SCRUM y el cliente en XP, realizan prácticamente las mismas tareas.
- SCRUM no tiene obligatoriamente que comenzar a trabajar un elemento de trabajo por la prioridad, en cambio XP sí debe seguir ese orden de manera estricta.

Conclusiones

Todas las metodologías cumplen con los principios del manifiesto ágil. La diferencia se basa en el contexto donde se desee emplear. Por una parte, XP hace especial hincapié en la calidad del código para la estabilidad del producto final. Por otro lado, KANBAN tiene mucha utilidad cuando se tienen retrasos en los proyectos, limitando la cantidad de trabajo en progreso. Por último, SCRUM está especialmente pensado para que los equipos logren mayor productividad y ganancia, por lo que se ajusta más a equipos que puedan destinar tiempo colectivo a un proyecto.

En [24] se realiza un análisis inicial de las empresas líderes del mercado en la planificación ágil, ver fig.13.

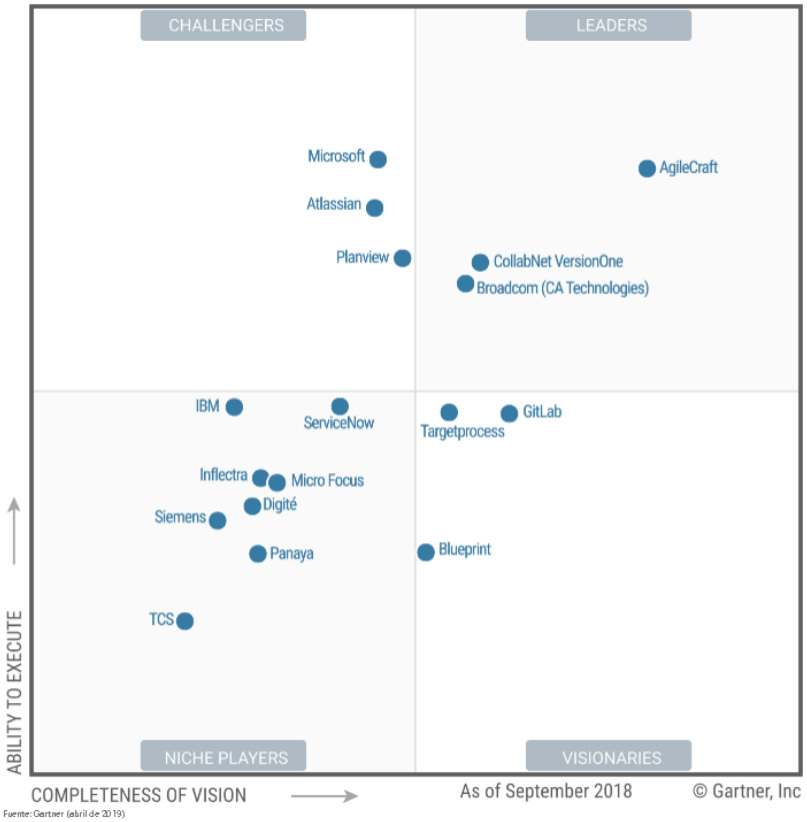


Fig. 13 Cuadrante mágico empresas líderes del mercado en la planificación ágil [24]

Desde hace algunos años Atlassian se venía mostrando como líder en su sector. En la última actualización realizada por Gartner[6], AgileCraft se ha posicionado a la cabeza, aunque para el año 2019 Atlassian ha anunciado un acuerdo definitivo para adquirir el producto también llamado AgileCraft.

De manera general el mercado está compuesto por Atlassian, AgileCraft, Microsoft, Planview, CollabNet VersionOne como principales líderes. Por ende, sus herramientas están igual de bien posicionadas.

[6] Empresa líder en investigación y asesoramiento

2.4.2 Herramientas gestión de proyecto, control de versiones, integración continua.

A continuación en la Fig.14 se muestra una comparativa de varias de las herramientas más usadas para la gestión de proyectos según una investigación realizada por CollabNote VersionOne[7] [10]. Se basa en una encuesta realizada a personalidades en una extensa gama de la industria del software.

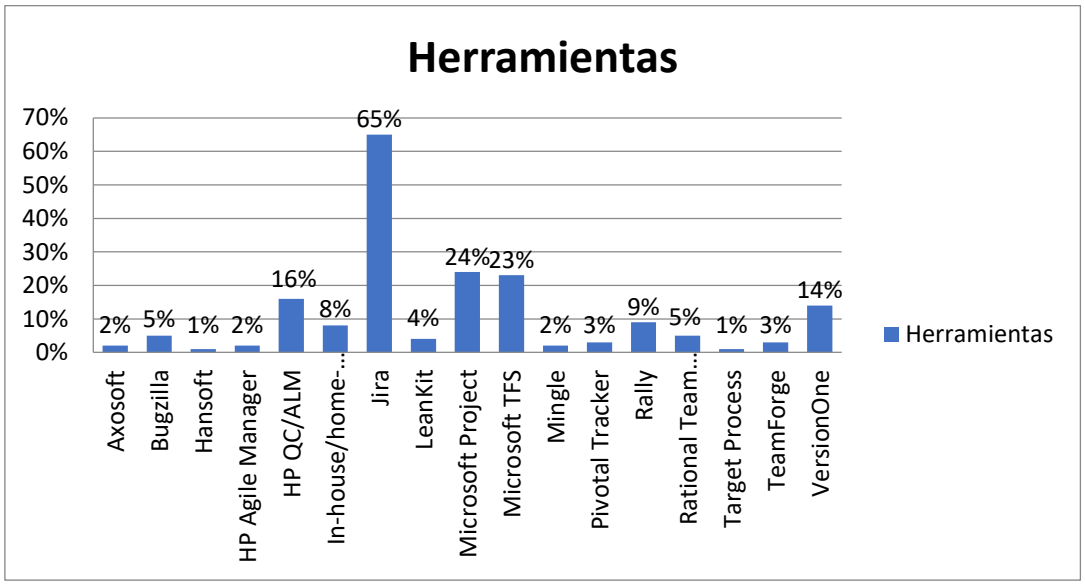


Fig. 14 Comparativa de uso herramientas gestión de proyectos

Seguidamente, se muestra un gráfico Fig.15, del uso de sistemas VCS[8] por tipo. Para el estudio se encuestaron 7643 programadores. [25]

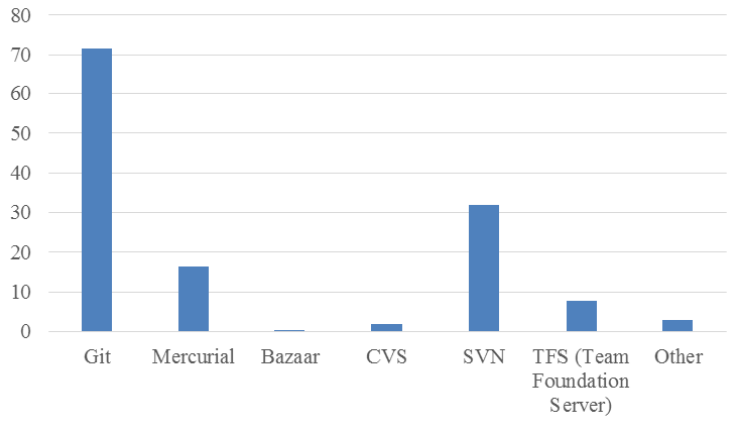


Fig. 15 Estadísticas sobre el uso de sistemas VCS por tipo [25]

[7] Firma de software dedicada a creación de productos y servicios en la rama de la industria de gestión de flujo de valor, gestión ágil, gestión del ciclo de vida de la aplicación y control de versiones empresariales.
[8] Sistema de control de versiones

Aunque no se puede generalizar el uso de estos sistemas porque para decidir su empleo se deben analizar las características del proyecto y la institución en la que se implantará, se puede de igual modo inferir que el uso de repositorios distribuidos como Git encabeza la lista de los más usados.

En G2[9] [26] realiza un estudio para ver el comportamiento del uso de diferentes herramientas según su uso. Para realizar la puntuación, G2 se basa en revisiones que han sido recopiladas por la comunidad de usuarios, fuentes en línea y redes sociales. De esta forma, se aplica un algoritmo que calcula los puntajes de satisfacción y presencia en el mercado en tiempo real. A continuación, en la Figura 16, se muestra el resultado en cuadrantes (océanos azules). El caso de estudio solo se centrará en modelos de Empresa y las variables que determinan los cuadrantes serán: Contendiente, Líderes, Nicho, Alto rendimiento. Los ejes X – Nivel de satisfacción, Y – Presencia en el mercado.

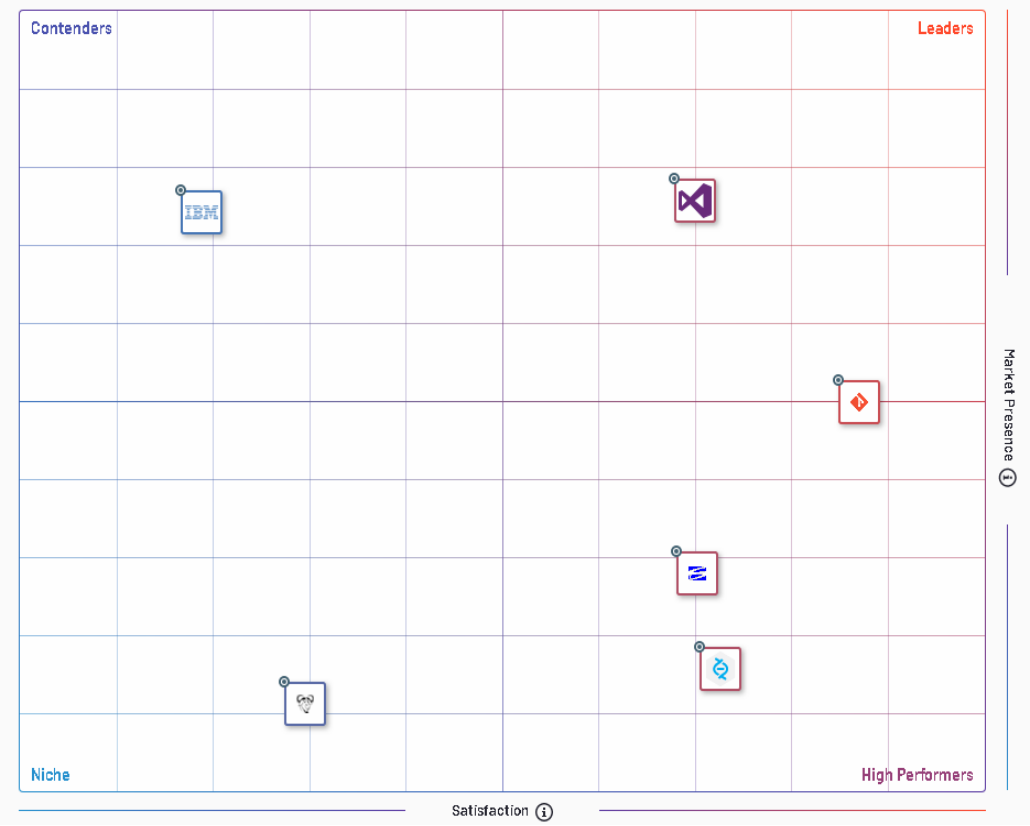


Fig. 16 Cuadrante representativo de VCS para Empresa [24]

[9] Mercado tecnológico donde las empresas pueden descubrir, revisar y administrar la tecnología que necesitan para alcanzar su potencial.

Se analizan las herramientas con mayor competencia respecto a TFS (Git, Subversion, Relational ClearCase) en este sector.

La Tabla 2 muestra un resumen de las categorías de uso de cada herramienta, ya que uno de los requisitos fundamentales era la selección de herramientas con múltiples funciones:

	Git	Subversion	Rational ClearCase	Microsoft Team Foundation
Categorías comunes	Sistema de control de versiones			
Categorías particulares	----------	----------	Gestión de configuración	Despliegue Continuo, Revisión de código de pares y alojamiento de control de versiones.

Tabla 2 Categorías de uso TFS, Git, Subversion, Rational ClearCase [26]

En la Fig.17, se analizan las herramientas con mayor competencia respecto a TFS (CircleCI, CloudBees Core, Jenkins) en este sector.

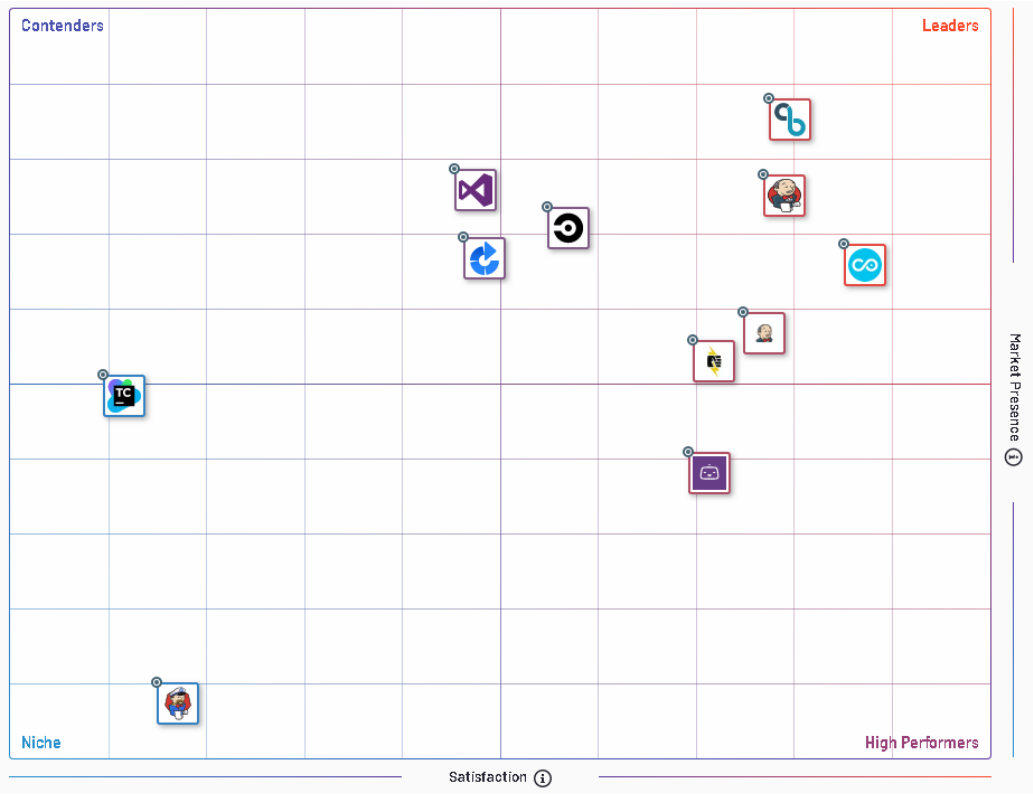


Fig. 17 Cuadrante representativo de IC para Empresa [24]

La tabla 3 muestra un resumen de las categorías de uso de cada herramienta:

	CircleCI	CloudBees Core	Jenkins	Microsoft Team Foundation
Categorías comunes	IC y automatización de compilación.			
Categorías particulares	Despliegue continuo	Gestión de configuración y Despliegue continuo	Entrega continua	Sistema de control de versiones, Despliegue continuo, Revisión de código de pares y alojamiento de control de versiones.

Tabla 3 Categorías de uso TFS, CircleCI, CloudBees Core, Jenkins [26]

Conclusiones:

La elección de una u otra herramienta, técnica o metodología estará sujeta y condicionada a las características de cada institución de manera general y del proyecto en cuestión de forma más específica.

A continuación se explican algunos detalles a tener en cuenta:

ANIMSA tiene un formato empresarial, con tres departamentos dedicados al: desarrollo, despliegue y mantenimiento de las soluciones y servicios que brinda. Por tanto, han de considerarse herramientas de alcance empresarial. En cuanto al desarrollo de sus productos, los equipos de trabajo son pequeños y auto-gestionados por lo que el uso de metodologías ágiles es una opción viable.

Es necesario aclarar que se ha partido del análisis de la herramienta de TFS por requerimiento de la empresa, basado en el amplio uso de .NET, contándose con las licencias correspondientes de antemano. Además, se valora el uso extendido que poseen del IDE de desarrollo Visual Studio, y el nivel de conocimiento general del equipo en el *stage* de .Net, así como capacitaciones recibidas con anterioridad con la herramienta de TFS. Todos ellos son factores de peso en la toma de decisiones.

Teniendo en cuenta todo lo anteriormente expuesto y el análisis realizado de las metodologías, técnicas y herramientas existentes en el mercado, se realiza la propuesta de implantar la metodología Ágil SCRUM, usando la herramienta de TFS para IC como objetivo principal, pero teniendo en cuenta el uso de la misma para emplear todo su

potencial no solamente en los proyectos actuales sino también en los venideros, teniendo una base ya creada para poder seguir evolucionando en el futuro.

Una vez analizada la metodología y las herramientas ágiles, en el siguiente capítulo se detalla la manera en la que serán empleadas dentro del departamento de I+D de ANIMSA.

CAPÍTULO 3: DISEÑO Y DESARROLLO

3.1 Análisis de la situación actual

Para la realización del estudio he partido de la elaboración de una encuesta online (ver Anexo #1), realizada de manera anónima a personal del departamento de Innovación y Desarrollo (I+D). De esta manera pretendo percibir su conocimiento, dominio y nivel de adopción referente a las herramientas, técnicas y metodologías de gestión y desarrollo de software que conocen. No obstante, además de la encuesta he realizado entrevistas personales para indagar en otros aspectos que considero convenientes de cara al trabajo en equipo.

El departamento de I+D se compone de 10 miembros con perfiles de análisis y desarrollo en su mayoría. Teniendo en cuenta esta información, creo la encuesta valorando los aspectos más importantes de cara a obtener resultados que me permitan analizar la situación vigente en el departamento.

La encuesta se compone de tres grupos de preguntas:

Grupo1 – Preguntas sobre metodologías de desarrollo

Grupo2 – Preguntas sobre herramientas de gestión de proyecto

Grupo3 – Preguntas sobre IC

A continuación, muestro algunos de los resultados más generales. Por motivos de confidencialidad con la empresa no hago públicos resultados más específicos.

Resultados: En la Fig.18 se evidencia la cantidad de personas por los diferentes roles que han desempeñado o bien tareas que se puedan corresponder con los roles.

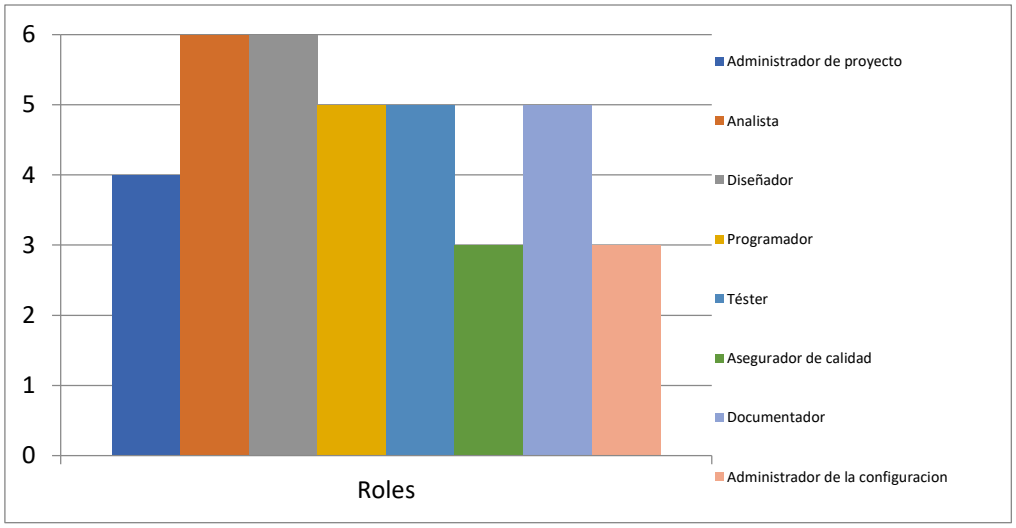


Fig. 18 Rol o Roles que desempeñan o han desempeñado

En la Fig.19 Se muestran por cantidad de miembros, qué metodología conocen.

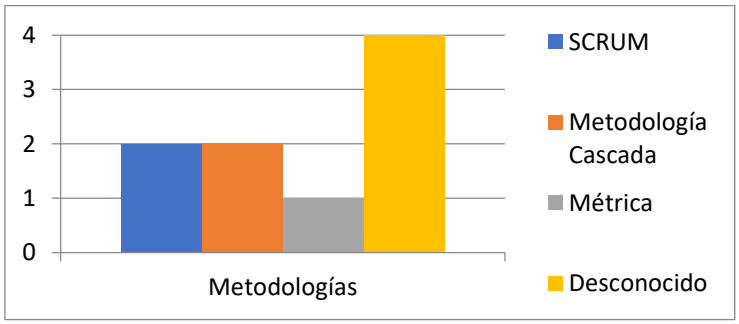


Fig. 19 Conocimiento Metodologías de desarrollo

Una vez analizados de manera exhaustiva los datos de la encuesta, llego a las siguientes conclusiones. No se evidencian amplios conocimientos sólidos y comunes entre todos los miembros. Los niveles de adopción de las metodologías y herramientas son limitados y no se explotan todas las capacidades. Por otra parte, no existe desarrollo de técnicas de IC de manera automatizada.

Se ha de resaltar que los equipos son pequeños (una media de 3 personas) y multifuncionales en gran medida, además de auto-organizados y con bastante autonomía.

No se desarrollan reuniones diarias, en cambio realizan reuniones bajo demanda del gestor de proyecto. El gestor de proyecto puede llevar el control de uno o varios proyectos concurrentemente, y será el encargado de garantizar la supervisión, coordinación y análisis y diseño de cada uno de sus proyectos. Se procura que la figura

del gestor de proyecto siempre sea la misma una vez designado, pero excepcionalmente bajo circunstancias que lo ameriten podrá ser sustituido por otra persona.

Los equipos se constituyen por: un gestor de proyecto que además es el analista principal y el encargado de las pruebas de integración, e1 equipo de desarrollo (máximo tres desarrolladores) encargados del desarrollo y la implementación de pruebas unitarias. Resulta importante resaltar que la empresa trabaja con instituciones públicas, por lo que es necesario que los implicados posean un conocimiento de la manera de trabajar en las mismas. Por ende, normalmente existe una mezcla en los equipos de personal más experimentado en la manera de trabajar de estas entidades, aportando un conocimiento importante para el buen desempeño del trabajo.

En ANIMSA los equipos de desarrollo son subcontratados a externos a través de concurso público. Lo que hace que sea complejo el uso de metodologías enfocadas en el trabajo en equipo.

Actualmente no está instaurado ningún sistema de IC o de pruebas de manera automatizada. Se puede evidenciar que los equipos han estado realizando en ocasiones, y de manera inconsciente, determinadas prácticas ágiles. Sin embargo, no se aplican muchas de las prácticas más importantes, así como el uso de técnicas y herramientas para su aplicación. Sus técnicas de pruebas se realizan de forma manual. Aún no se basan en ciclos con objetivos alcanzables a corto plazo para solucionar problemas de manera más eficaz.

Debe existir un mecanismo de integración más dinámico entre los equipos, donde se conozca en qué trabaja cada individuo y fluya la colaboración. Esto es importante de cara a la reutilización de código. Ser ágiles implica tener capacidades de adaptación y en este sentido aún se tiene que trabajar para lograr la mentalidad y capacidad necesarias en el departamento.

Luego de haber realizado un análisis exhaustivo y detallado, describo cómo propuse la implantación de metodologías ágiles, específicamente SCRUM dentro del departamento.

3.2 Implantación de SCRUM

La primera acción a tomar para poder implementar una metodología ágil era cambiar la filosofía del equipo, ya que es necesario crear una mentalidad ágil. Aunque usualmente suele ser un reto para los equipos, el resultado obtenido compensa el esfuerzo.

En [27] se determinan cinco actividades principales (ADAPT) para la adopción de la metodología, las cuales propongo desarrollar:

Conciencia: Aceptar que el proceso actual no está dando los mejores resultados. La resistencia al cambio suele ser normal cuando en tiempos pasados obtuvimos buenos resultados, pero se debe ser crítico cuando ya se evidencia que el estatus actual no es el más deseable. En este punto propongo a la dirección realizar un estudio y generar un pequeño informe con gráficos y tablas donde se vean los resultados a nivel de comparativa, de todos los proyectos que se han realizado en el departamento. De esta forma se podrá tener una visión más clara para poder crear conciencia al personal, de que funciona mejor o peor.

Deseo: Ganas de adoptar como suyas las prácticas que se ajusten o estén más acordes con nuestros procesos dentro de la empresa. No solo basta con tener la conciencia de qué se debe hacer, sino que además es importante tener ganas de hacerlo. ¿Pero qué pasa si esas ganas no surgen por sí solas? Existen mecanismos para aumentar el deseo del cambio, aunque se pueden ir ajustando dependiendo de los resultados. Se puede: comunicar que hay una mejor manera, crear una sensación de urgencia, construir los momentos, crear una experiencia rápida de agilismo con el equipo, incrementar los incentivos, intentar mitigar el miedo, dar tiempo para dejar ir el pasado y no desacreditarlo, involucrar al equipo en el trabajo.

Capacidad: Se trata de desarrollar capacidades ágiles, no solo por el hecho de aprender nuevas, sino porque de esa manera ayudamos a que desaparezcan las deficientes. Existen herramientas para desarrollar estas habilidades: Proporcionar coaching y capacitación, crear sentido de responsabilidad, compartir la información y los problemas, establecer objetivos que sean razonables y empezar a trabajar.

Promoción: Consta de tres pasos principales. Primero, promover el éxito para la creación de conciencia de que es bueno. Segundo, reforzar el agilismo difundiendo las

cosas buenas que se han logrado. Tercero, trabajar en la conciencia de terceros (RRHH, compras, ventas, sistemas, mantenimiento), ya que su influencia es primordial para el éxito de la transición.

Transferencia: Si no se transfiere el conocimiento muere. Si los demás departamentos no son compatibles con el "agilismo", emerge una fuerza tal que impide su avance. No significa que el resto de la organización deba usar SCRUM, pero sí deben conciliar o ir acorde en la medida de lo posible.

Una vez definidas las actividades y conocido el estado de las mismas en la organización se debe decidir cómo se incorporarán estas prácticas. Esto puede ser decisivo para el éxito o fracaso de la implantación. En [27] se plantean algunas preguntas claves para realizar la toma de decisiones:

¿Deberíamos empezar con uno o dos equipos, o deberíamos convertir todos los equipos al mismo tiempo?

¿Deberíamos anunciar nuestra intención (quizás sólo a algunos en la empresa, pero tal vez también públicamente), o deberíamos mantener el cambio en silencio por ahora?

3.2.1 Comienzo Poco a poco o apostando por todo

Poco a Poco	Apostar por todo
Es más barato	Puede minimizar la resistencia al cambio que suele ser bastante común.
El éxito de manera temprana está prácticamente garantizado	Los equipos trabajarían de manera homogénea, y evitamos las diferencias.
Se evitan los riesgos al apostar todo por todo	El cambio sería prácticamente inmediato.
Estresa menos	-
Evitamos la restructuración abrupta y traumática de la empresa	-

Tabla 4 "Empezar poco a poco" VS "Apostando todo"

Tras analizar los diferentes puntos de vista como se muestra en la Tabla 4, en el caso de ANIMSA considero conveniente una postura quizás más conservadora. Comenzar con un proyecto de prueba, e ir aprendiendo de él sin arriesgar mucho, y paulatinamente ir expandiendo luego por todos los demás proyectos si así lo requieren y en dependencia de los resultados que se vayan obteniendo.

Suelen existir dos maneras de ir expandiéndose. Una puede ser que tras terminar un proyecto se comience inmediatamente con el siguiente. Otra forma es de manera solapada, es decir, luego de varios sprint y aún sin terminar, comenzar el siguiente proyecto y así sucesivamente con los demás.

En nuestro caso particular propongo un enfoque cauteloso, luego de implantar el proyecto actual se valorará los resultados para comenzar con otros proyectos. Esto se debe también a que de esta forma se puede ir mostrando el éxito de manera pequeña para ir convenciendo e involucrando a los más escépticos. En el caso de ANIMSA un progreso a pequeña escala no significa necesariamente que dicho progreso sea lento, ya que se puede avanzar rápidamente una vez se vean los resultados positivos.

3.2.2 Anunciar Públicamente o no

Público	No público
Al saber todo el mundo lo que se está empleando, existen más posibilidades de que se unan a ello.	Se gana terreno, en otras palabras, hay ventaja de avanzar antes que empiecen a resistirse los elementos.
Es interesante que se genere debate sobre el motivo del cambio, y de esta forma obtener puntos de vista que podrían ser útiles.	No hay tanta tensión, al no ser conocido no se siente la misma presión sobre los involucrados. No se crean expectativas.
El compromiso es importante. Al trabajar sin ocultaciones, refuerza y afianza ese compromiso involucrando a todos.	El anonimato se mantiene hasta que se determine que puede ser revelado. Si algo sale mal se tiene tiempo para intentarlo nuevamente, hasta que se obtengan éxitos y pueda ser mostrado.
Se pueden obtener ideas y ayuda de otras partes que pueden ser útiles.	Existe una frase popular que dice: “Hay cosas que para lograrlas han de andar

	ocultas". Si nadie sabe que se hace, nadie intentará detenerlo.
Es más impactante sobre todo para los escépticos decir lo que vas hacer y luego hacerlo, que anunciar un objetivo después que ha sido logrado, ya que puede crear dudas en cuanto a su veracidad.	-

Tabla 5 "Anunciar Públicamente" VS "Discreción"

Luego de analizar los puntos mostrados en la Tabla 5 llego a la siguiente conclusión. Dentro del departamento, sus principales activos confían en las potencialidades de SCRUM, es por tal motivo que se hace público su uso. Primeramente, expliqué su uso, implicaciones y potencialidades al cuadro de mando del departamento y al ser aprobado, hice extensiva la información a todos los departamentos que están involucrados. Pidiendo de esta forma la máxima colaboración y aportación de los mismos.

3.2.3 Difusión del Agilismo

Una vez adoptada la manera y anunciada nuestra posición, es imprescindible difundir el agilismo, si no se corre el riesgo de que expire.

En [27] se determinan las siguientes pautas:

1- **Dividir y Sembrar**: Se trata de tomar miembros claves de los proyectos que han tenido éxito y usarlos dentro de otros equipos y de esta forma transmitir los conocimientos adquiridos.
2- **Crecer y Dividir**: Se considera una variación del enfoque anterior, en este caso basta con ir agregando miembros al equipo. Cuando éste haya crecido lo suficiente y el agilismo esté completamente adoptado por todos, el equipo principal se dividirá formando dos equipos equivalentes.
3- **Coaching Interno**: Se trata de identificar en los proyectos que más avanzan un miembro que entiende perfectamente el agilismo, y esa persona entonces tomará el papel de coach para los equipos que presentan más retraso.

Aunque aplicar "Dividir y Sembrar" gusta por la acelerada propagación que suele tener al incorporar de manera bastante más rápida nuevos proyectos, considero conveniente al

no existir varios grupos de desarrollo muy definidos dentro del departamento optar por la solución de un coach interno, que será la figura de SCRUM máster para todos los demás proyectos que sigan esta línea. El coach aportará siempre nuevas ideas a cada uno de los equipos, ya que no es una figura estática sino más bien un ente que se va moviendo y retroalimentando de todas las experiencias, y en función de los resultados va adoptando mejores medidas para cada uno de los equipos convirtiéndose en un experto del agilismo.

3.2.4 Aplicar prácticas técnicas

Cuando pensamos en implantar SCRUM van de la mano prácticas técnicas que se pueden adoptar y que en gran medida ayudan a que los equipos sean aún más ágiles. Pero, ¿cuándo es momento de adoptarlas?

Se puede comenzar pronto o retrasar su uso. Ambas maneras tienen sus ventajas, en [27] se determinan algunas que se muestran a continuación, Tabla 6.

Pronto	Retrasar
Permite realizar mejoras rápidas El uso de estas prácticas rápidamente proporcionará victorias tempranas, que suelen ser motivadoras.	Resistencia al cambio. Probar cosas nuevas no es siempre del gusto de todos, ya que requieren tiempo y esfuerzo. Existen personas más reacias al cambio, y probarlas tempranamente puede aumentar la predisposición a rechazar el agilismo.
Si no es ahora puede que no sea nunca. Es común al adoptar un mínimo de las prácticas de SCRUM que se cree una especie de conformismo porque las cosas van funcionando “bien” y dejamos para más adelante probar cosas nuevas. Queda demostrado que la gran mayoría de prácticas que se dejan a futuro luego no se realizan.	Puede que sea demasiada información para empezar. Comenzar a usar metodologías como SCRUM conlleva de por sí un esfuerzo. Si a eso le sumamos tener que aprender nuevas prácticas, puede sobresaturar en muchos casos a los miembros que pueden sentirse abrumados en el inicio, cuando aún están en período de adaptación y evolución.
Su uso puede solucionar muchas de las problemáticas más incidentes. Estas	-

prácticas surgen por una razón, y estas razones son resolver problemas que suelen ser bastante recurrentes en los proyectos: ciclos de entregas muy largos, problemas de mala calidad, malos diseños, etc.	

Tabla 6 "Adopción temprana" VS "Retrasar Adopción"

En muchos casos suele ser común que no se elija una u otra opción de manera exacta. En nuestro caso, entiendo la necesidad del uso de prácticas técnicas prontamente, pero de manera paulatina. Es decir, se van incorporando poco a poco. Existen muchas prácticas a usar, pero ir eligiendo consecuentemente cada una de ellas puede ser la clave. He decidido comenzar con las prácticas de IC con pruebas automatizadas y *Feedback*. Su uso traerá por consiguiente una agilidad que saldrá de manera natural y casi sin esfuerzo en el equipo. Más adelante se pueden ir introduciendo otras como programación en pares, refactorización, etc.

En general se propone realizar un trance iterativo hacia el agilismo y hacia SCRUM por consiguiente. Considero que evolucionar poco a poco hacia el agilismo puede ser menos costoso para la empresa, además evita la restructuración abrupta de la misma, porque el cambio se da de forma más natural. Al seleccionar solo un subconjunto de prácticas evitará el stress de los miembros que están más acostumbrados a prácticas tradicionales, pero que podrán conciliar el nuevo método de trabajo con prácticas que no serán costosas de integrar y evidenciarán rápidamente logros. Esto incentivará el deseo por continuar e ir introduciendo otras prácticas, este proceso se procura que además sea ágil también.

Para adoptar SCRUM propongo la herramienta TFS que puede ser usada gracias a su capacidad de adaptación para diferentes contextos. A continuación, explico la propuesta.

3.3 SCRUM con TFS

En la actualidad, se evidencia un cambio radical en la manera que se gestionan los proyectos de desarrollo de software. Resulta casi imposible no relacionarlo con metodologías ágiles o SCRUM propiamente. A raíz de todo este movimiento surgen como una necesidad herramientas como TFS.

Este software será el que nos ayude a la gestión del ciclo de vida de todo nuestro proyecto (en su distribución *On-premise*). Emplearemos la plantilla de la metodología a usar, en este caso SCRUM.

Podremos realizar las siguientes actividades:

- Definir marco de trabajo (creación de proyecto, control de miembros, tiempos, capacidades, tipo de actividades por miembro).
- Gestión de procesos (Epopeyas, funcionalidades).
- Planificación del sprint (Elementos de trabajo, tareas).
- Control de versiones de código fuente (en *check-in* asociar una tarea, *feedback*).

Existen diversas funcionalidades a tener en cuenta para mejorar resultados como la búsqueda de código en todos los proyectos, administración de paquetes, seguimiento de un elemento de trabajo, actualizaciones directas y filtrado en el panel KANBAN, edición masiva de etiquetas, uso de correo electrónico y widgets.

Con TFS el seguimiento y planificación del trabajo diario, la revisión del sprint y la retrospectiva están prácticamente resuelto mediante todas las actividades antes expuestas. En el caso de la reunión de retrospectiva le sacaremos muchísimo partido a TFS, a través de toda la información recolectada y *feedbacks* realizados se tendrá suficiente material para debatir cuán veloz ha ido el equipo, qué no ha ido bien en el sprint y por qué. Y lo más importante, qué haremos de forma diferente, siempre buscando maximizar el resultado.

A continuación, veremos las actividades antes expuestas de manera más concreta en el proyecto de demostración en el que he trabajado dentro del departamento de innovación y desarrollo.

3.3.1 Caso práctico dentro del departamento:

Hasta este instante he evidenciado todas las potencialidades que de manera general brinda TFS para poder realizar cada uno de los procesos que determina SCRUM en su marco de trabajo. Pero considero importante evidenciar cómo se propone la puesta en marcha dentro del departamento, de forma que se vea cómo se ha trabajado en el proyecto modelo.

Para que el proceso de comprensión sea mejor, a continuación muestro la Fig.20, que servirá como apoyo visual para centrar cada proceso de SCRUM a cada una de las prácticas realizadas con la herramienta.

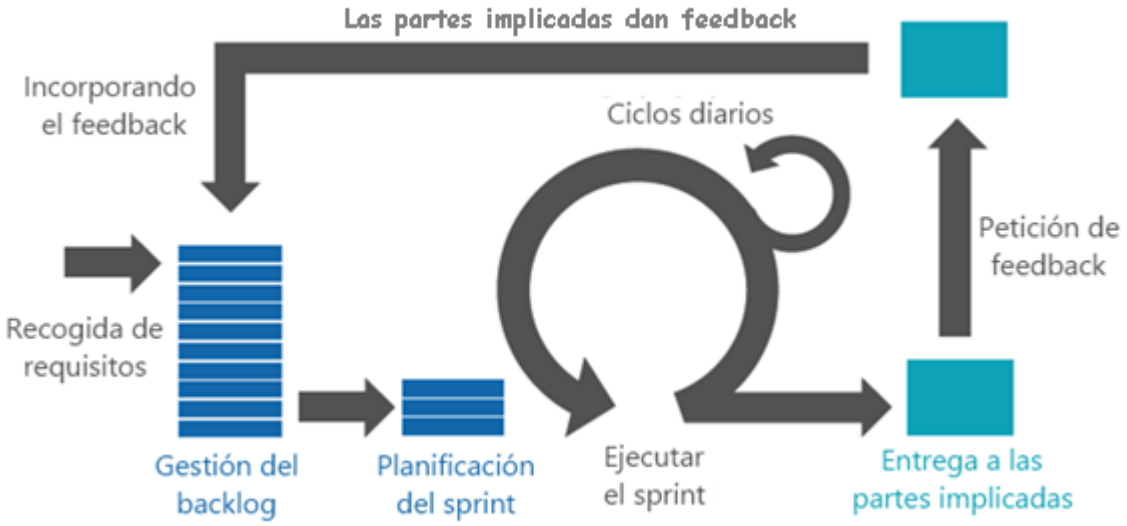


Fig. 20 Proceso de SCRUM [fuente desconocida]

Como parte de las tareas de las prácticas realizadas dentro de la empresa realicé un análisis funcional de los sistemas que se debían implementar. De este análisis surge el documento funcional del proyecto, que entre otras cuestiones recoge un levantamiento de requisitos que sirve como inicio del proceso.

3.3.2 Gestión del Backlog

Una vez definidos debidamente los requisitos se comienza la "**Gestión del Backlog**". En este caso la figura del *Product Owner* fue definida bajo mi criterio personal. Teniendo en cuenta que se trata de un miembro dentro del equipo que conoce muy bien el negocio dentro de ANIMSA y posee una visión general del proyecto, representa muy

bien cada una de las partes y su compromiso es total con el enfoque ágil que se quiere dar. Es necesario aclarar que es una persona que no tenía amplios conocimientos sobre este rol. Parte de mi trabajo ha sido formarlo en este sentido e ir supervisando cada una de sus actividades.

Para su formación era importante transmitirle los conocimientos necesarios para desarrollar sus funciones:

- Organizar y priorizar el backlog.
- Crear las historias de usuario con la estructura recomendada y buenas prácticas que se definen para su creación.

Ejemplo de formato propuesto para crear la historia de usuario:

  - *Como [***usuario***] quiero [***algo***] para que [***pueda lograr eso***]*.
  - *[***usuario***]* – rol de la persona que lo solicita
  - *[***algo***]* – lo que desearía que pasara
  - *[***pueda lograr eso***]* – el objetivo que se busca

Con TFS dentro de la sección de */Trabajo/Trabajo pendiente*, se tienen las herramientas para introducirlo de manera bastante sencilla. A continuación, se muestra un ejemplo de cómo se realizó en el proyecto modelo.

En la Fig.21 se puede ver la lista de historias de usuarios ordenadas según la prioridad. Seguidamente, en la Fig.22 se muestra la misma información, pero en forma de tablero KANBAN, en donde se puede visualizar en todo momento la evolución de estas historias de usuario.

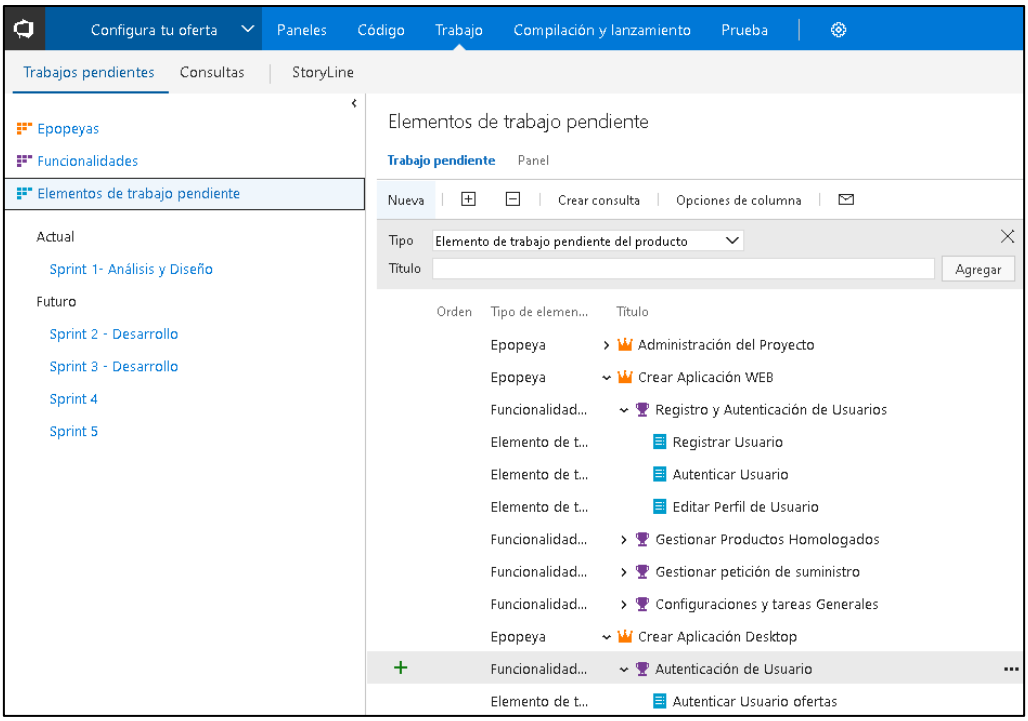


Fig. 21 Ejemplo del Backlog, proyecto modelo

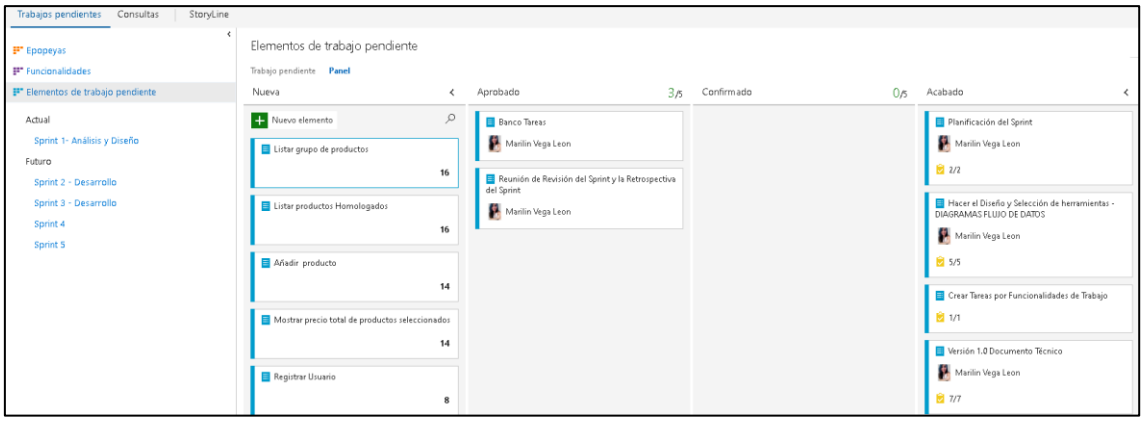


Fig. 22 Ejemplo Backlog visto desde el panel (Tablero)

3.3.3 Preparación del entorno

Una vez creado el backlog se prepara el entorno de trabajo. Para la realización de esta tarea se define el tiempo que dura cada sprint. En el proyecto en cuestión se definieron en principio sprints de quince días o dos semanas laborales. Se considera que al no ser una aplicación tan grande se pueden ir haciendo entregables en cortos intervalos de tiempo. En el momento inicial fueron creados tres sprints, pero esto puede variar (tanto la duración de los sprints como la cantidad) a medida que el desarrollo vaya avanzando.

Dentro del entorno de trabajo se define además el tiempo del que dispone cada miembro del equipo y qué tipo de especialidad tiene cada uno. Teniendo en cuenta que una de las características del equipo es que es multidisciplinar, se hace de vital importancia calcular las capacidades. Dentro de TFS es muy sencillo poder hacerlo, puesto que la herramienta brinda un panel en el cual una vez introducidos cada uno de los miembros del equipo se le asignan la o las actividades que va a desarrollar. Además, se indica el tiempo que dedicará a trabajar diariamente en cada una de las actividades definidas. Inclusive, se puede introducir el tiempo de vacaciones durante el sprint si procede.

A continuación, en Fig.23, se muestra un ejemplo de cómo se hizo esta distribución en un primer sprint que nombré "Análisis y diseño". Este Sprint es conocido también como el sprint 0. En este sprint dejo listo el entorno, se trabaja en el producto backlog (lista de historias de usuarios, prioridad, estimación), se reparten las historias de usuario por iteración y se estudia la arquitectura.

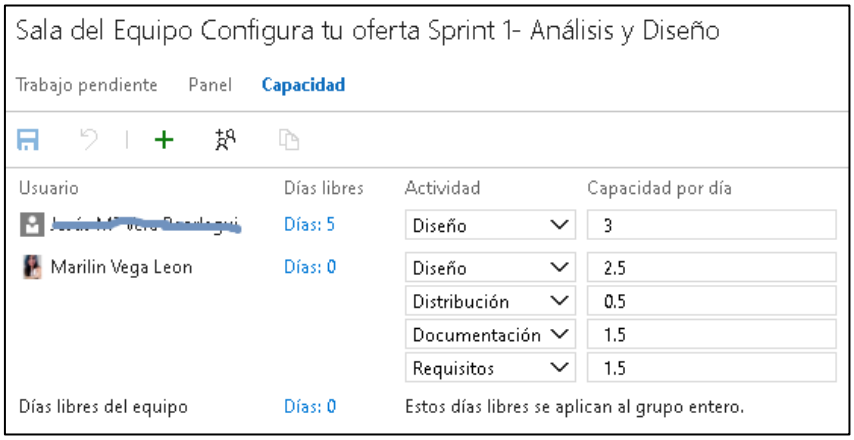


Fig. 23 Ejemplo de asignación de capacidades

3.3.4 Planificación del sprint

Esta actividad comienza dentro de la reunión de planificación del sprint, la cual he determinado que teniendo en cuenta que son sprints de 15 días cada uno, tendrán una duración de cuatro horas. Al terminar dicha reunión se tendrá el listado de las tareas que se desempeñarán durante todo el sprint.

Desde TFS podremos arrastrar los elementos de trabajo planificados sobre el sprint que corresponda. Es importante que, dentro de cada una de las historias de usuario seleccionada, quede siempre recogida toda la información que se obtenga durante la explicación del requisito por el dueño del producto.

Para una mejor fluidez del trabajo, he tomado la decisión de que el propio equipo de desarrollo se auto-asigne las tareas. Como norma general propongo que si una tarea dura más de un día, deberá ser subdividida en tareas más pequeñas.

La estimación de la duración de las tareas se realizará en horas. Las características del equipo de desarrollo subcontratado, no permite poder hacer estimaciones por ejemplo mediante puntos de historia. Esto se debe a la contratación regida por tiempo, alcance y costo fijo.

Con respecto a esta parte del proceso, he determinado además algunas pautas a seguir de manera general en cada una de las reuniones de planificación que se realicen:

- El *Product Owner* deberá llevar una propuesta de las historias de usuarios que serán implementadas en el sprint. La prioridad debe estar debidamente señalada en cada una de ellas.
- Se recomienda que el *Product Owner* lleve preparada cada una de las historias con la mayor claridad posible, de esta forma se ahorra en tiempo.

3.3.5 Ciclos diarios - Reuniones diarias

Dentro de un sprint el trabajo se divide por días, y para su seguimiento propongo las reuniones diarias con una duración de no más de cinco minutos por participante. Lo ideal sería poder hacerlo siempre en una misma localización, pero como en ANIMSA las salas son comunes y coexisten varios departamentos los recursos son comunes. Por tanto, con un día de anterioridad se reservará una sala para realizar el encuentro a primera hora de la mañana del siguiente día.

He decidido que la mañana sea la franja horaria para realizar las reuniones porque estimo que será mejor ya que las personas están frescas y podrán fijar mejor los objetivos.

Para optimizar mejor el tiempo propongo las típicas preguntas en los *daily meeting* de SCRUM:

- ¿Qué hice el día anterior?
- ¿Qué problemas he encontrado?
- ¿Qué voy a hacer hoy?

Era necesario poder hacer entender que estas reuniones lejos de parecer que se realizaban para controlar a las personas, tenían el objetivo de que se fomentara la comunicación dentro del equipo. La comunicación es fundamental para estar en sintonía y que el trabajo salga bien.

De las características mejor valoradas que brinda TFS es su tablero, donde muestra de forma visual el desarrollo del trabajo. Hay que tener en cuenta que en este tablero una vez cambiada una tarea al estado terminado, no se puede volver atrás. Si esto sucede se tomará como norma crear una nueva historia que subsane el problema generado.

Para poder mantener el trabajo actualizado y que la gráfica de "*burn down chart"* esté debidamente al día, recomiendo las siguientes pautas:

- Día a día, el equipo debe ir actualizando las tareas en el TFS, tanto si fueron concluidas como el tiempo restante para su culminación, y siempre antes de la próxima reunión diaria.
- Si se realiza un cambio desde el proyecto en Visual Studio, cuando el programador vaya a realizar un *check-in*, dentro del panel de "*Pending Changes*" de la ventana de "*Team Explorer*", debe asociar mediante arrastrar y soltar a la tarea. En Fig.24 muestro el procedimiento.

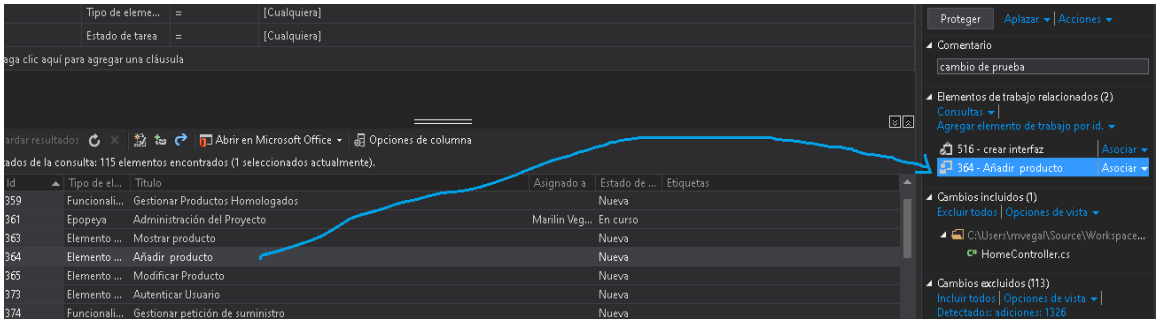


Fig. 24 Procedimiento para asociar un cambio a la tarea

3.3.6 Revisión del sprint / Reunión de retrospectiva (*feedback*)

La revisión del sprint y la reunión de retrospectiva son dos procesos diferentes y claramente definidos, pero por cuestiones de tiempo, disponibilidad del personal y de los recursos, he propuesto hacerlas juntas. Que vayan unidas no significa que se solapen, se debe respetar el tiempo destinado a cada una de ellas.

Para la recolección de los feedback que surjan de cada una de estas importantes reuniones, se usará una herramienta gratuita denominada "*Microsoft Feedback Client for TFS*" ver Fig.25. Esta aplicación una vez instalada, se conecta con nuestro TFS y nos permite almacenar en él todos los datos que han sido recogidos.

Procedemos a la revisión del trabajo realizado. En nuestro caso considero oportuno que estén todas las partes implicadas y la duración será de dos horas. Se presentará el producto tal y como está hasta este punto.

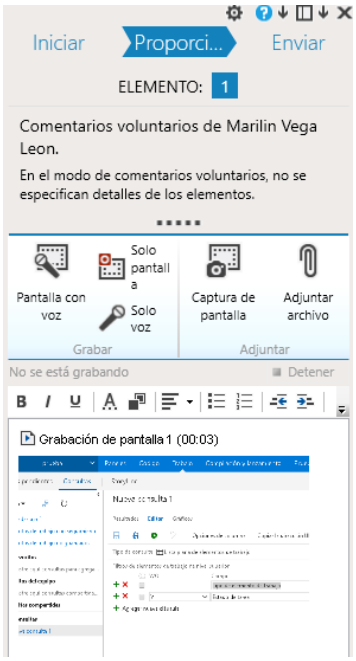


Fig. 25 Vista del cliente "Microsoft Feedback Client"

Para esta parte del proceso he determinado las siguientes pautas a seguir de manera general en cada una de las reuniones que se realicen de revisión del sprint:

- Se realizará una presentación real y práctica del producto, verificando su incremento. Dicha presentación estimo conveniente que sea el propio equipo de desarrollo quien la realice. Esto lo creo beneficioso para que el cliente no solo se vincule con el *Product Owner* sino también con el equipo en general y fluya una colaboración más estrecha.
- Se tendrá un tiempo destinado a que el o los clientes realicen cuantas preguntas estimen convenientes al equipo.
- El trabajo no culminado deberá ser recogido y pasado al siguiente sprint.
- Se analizarán las métricas obtenidas en el sprint (ver aparatado 4.3).

Una vez concluido el tiempo para la realización de la reunión de revisión se toma un descanso de unos quince minutos y se procede a comenzar la reunión de retrospectiva. Esta reunión propongo que tenga una duración de no más de una hora.

Para hacer esta última parte más amena propongo hacer una especie de juego, que denominaré (*circle feedback*). A continuación, explico su proceder.

Todos los implicados crearán una especie de círculo (en la medida posible). La idea es que todos puedan mirarse las caras. Se seleccionará un miembro el cual deberá comenzar a dar su opinión sobre cómo ha sido el proceso, qué se puede mejorar y si tiene alguna propuesta de mejora (plateará su propuesta desde dos ópticas: una desde el punto de vista de equipo, y otra desde su punto de vista personal). Luego, desde su derecha se pasará la palabra al siguiente compañero. Se procura que en próximas reuniones comience un miembro diferente cada vez.

Una vez más el objetivo es promover la participación y el trabajo en equipo. Otro objetivo que persigo con este método es generar ideas para identificar posibles acciones que ayuden a mejorar el rendimiento del equipo.

De lo más novedoso para el departamento, y que será de vital importancia, es el empleo de IC. En este sentido, TFS nos brinda un entorno bastante fácil de manejar y que nos permitirá aprovechar al máximo su capacidad. A continuación, expongo cómo será ejecutado.

3.4 SCRUM y TFS para integración continua

Usaremos IC como una actividad más dentro del proceso de SCRUM, ya que encaja perfectamente con la metodología, integrándose de forma muy natural en los sprints. Ayuda a mejorar notablemente la velocidad de cada sprint, agilizando las entregas de los artefactos debidamente testeados, y proporcionando un *feedback* del cliente, permitiendo mejorar continuamente el producto, solucionando bugs de manera eficiente antes que el proyecto crezca y cree dependencias que puedan ser difíciles o incluso imposibles de corregir.

La IC en el desarrollo de un equipo ágil viene ligada a múltiples factores a tener en cuenta. Es por tal motivo que usar herramientas que estén pensadas con ese fin es

primordial. En este sentido entra en juego TFS sirviendo de puente con el propósito de sacar el mayor provecho a estas prácticas.

El portal de TFS brinda una interfaz bastante íntegra e intuitiva. Mostrará el proceso desde las líneas de código, la integración de todo el desarrollo, incluyendo pruebas de calidad del código, hasta toda la información recogida gracias al equipo de desarrollo.

TFS brinda además el control de código fuente que, mediante una centralización del mismo, permite hacer un seguimiento constante, posibilitando además establecer métricas de calidad requeridas. Por su parte, el servidor de IC al interactuar constantemente con el repositorio de código estará diariamente actualizando a medida que él o los desarrolladores vayan cargando los cambios.

Con relación a subir código al TFS (*check-In*) propongo seguir las siguientes pautas:

- Siempre que se realice un *check-in* se debe hacer un comentario que indique el motivo de los cambios. Muy útil cuando en el futuro se quiera ver por qué se realizó un cambio en un momento dado.
- Al realizar un *check-in* debe asociarse a una o varias tareas (a criterio del desarrollador según el trabajo realizado).
- Al inicio de la sección de trabajo obtener siempre la última versión del repositorio y también antes de realizar un *check-in*.
- No acumular mucho código en los *check-in* para evitar demoras y prevenir conflictos. Tampoco se realizarán *check-in* muy seguidos porque esto desencadenará el procedimiento de integración continua cada vez y sería ineficiente. Se propone entonces siempre que el cambio esté asociado a generar una modificación de estado en una tarea, o si la modificación tiene un tamaño tal que influya en la disminución de horas de la misma.

Para la realización de pruebas, mediante TFS podemos crear las definiciones de compilación. Se crearán varias definiciones de compilación que incluirán diferentes tareas en el proceso de compilación. Algunas de ellas serán: análisis con Sonarqube, testAssemblies (ejecutar pruebas unitarias), compilar con MSBuild, entre otras. A continuación, muestro en Fig.26 un ejemplo de definición de compilación que propuse usar al desencadenar el proceso de IC.

Sonarqube: Es una plataforma para evaluar código fuente, que brinda una gran cantidad de funcionalidades para la detección de código duplicado, no adecuación a estándares y convenciones de código, vulnerabilidades conocidas de seguridad, código spaghetti, complejidad ciclomática, alto acoplamiento, entre otras.

testAssemblies: Se usará para ejecutar las pruebas unitarias y funcionales de visual studio.

Compilación de Visual Studio: Se usará para compilar con MSBuild y establecer la propiedad de versión de Visual Studio.

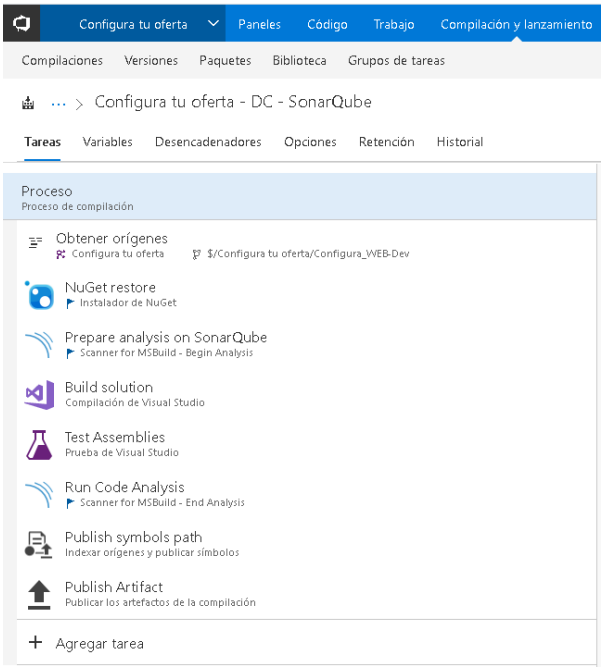


Fig. 26 Definición de compilación

La definición de compilación está sujeta a cambios que vendrán dados por posibles adiciones de nuevas tareas que puedan ser útiles de cara a mejorar el proceso de depuración del código (calidad, seguridad).

Una de las tareas dentro de la definición de compilación mostrada es la de Sonarqube. Propongo usar este servidor que está montado desde hace dos años en la empresa y no se ha implantado su uso. Considero que es una oportunidad para ver todo lo que nos puede brindar, teniendo en cuenta que se integra perfectamente con TFS, y puede servir como el complemento perfecto para poder desarrollar IC de una manera más integra de cara a la calidad del código y su seguridad.

3.4.1 Sonarqube

Hacer uso de esta nueva práctica técnica implica que dentro del servidor de Sonarqube se deba crear el proyecto que se vinculará con el de TFS. Para realizar dicha actividad impartí una formación explicando paso a paso su uso a todo el departamento de I+D y el departamento de mantenimiento.

En principio propongo usar las reglas por defecto que propone el servidor. Una vez finalizada la definición de compilación y por ende concluido el análisis con sonar, se podrá acceder a las métricas (tamaño, porcentaje de código duplicado, complejidad) y reporte que brinda Sonar de nuestro proyecto. Con toda esta información podremos sacar conclusiones que pueden ser luego debatibles en reuniones de sprint.

En la Fig. 27 muestro un ejemplo de un análisis obtenido con la herramienta.

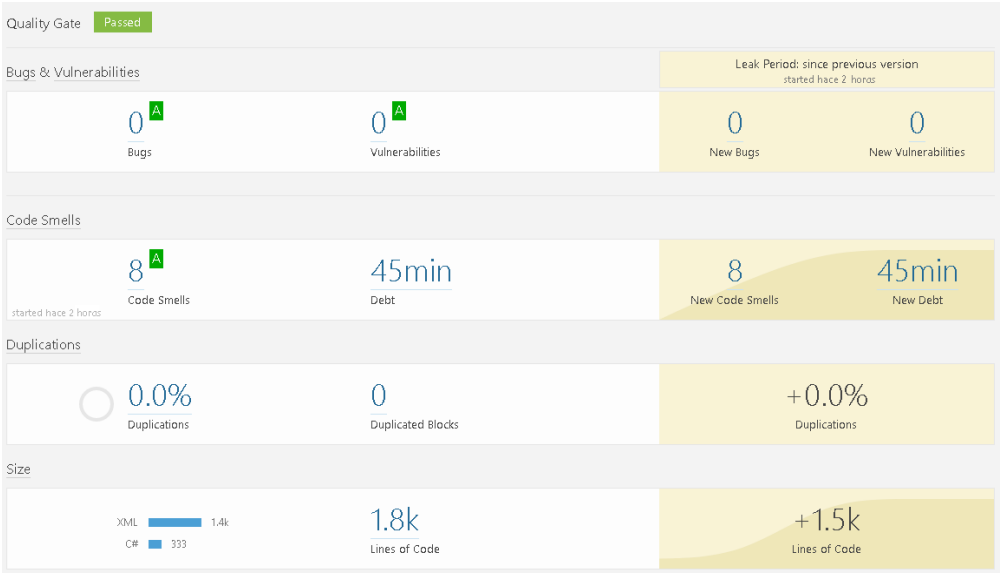


Fig. 27 Resultado general obtenido con servidor Sonarqube

En general sería muy difícil implementar IC si no se dispone de una automatización de *Builds* que pueda permitir la realización de procedimientos de forma repetitiva, pero gracias a TFS podemos hacerlo, de lo contrario se tendrían que hacer de manera manual.

Una vez definidas las pautas a seguir con las herramientas se describirá el plan concreto de implantación de prácticas ágiles que propongo para el departamento.

3.5 Diseño general del plan de implantación

El plan de implantación consta de cuatro pasos principales detallados a continuación, y de vital importancia para poder insertar la propuesta dentro del departamento de I+D.

Paso 1: Sesiones de formación para las personas implicadas. Se trata de divulgar e introducir la cultura del agilismo para evitar que exista un rechazo. Es importante dar a conocer las ventajas e implicaciones que trae consigo este tipo de prácticas.

Se realizarán dos sesiones de 1:30 horas con todos los departamentos implicados, I+D, Compras, Sistemas.

Paso 2: Se parte de un autodiagnóstico con varios de los responsables de los proyectos. El objetivo de estas sesiones es discutir y definir las prácticas ágiles que en principio son las más adecuadas para implementar en el departamento.

Las sesiones tendrán una duración entre 1 y 1:30 horas. Se estiman tres sesiones con diferentes responsables que serán convocados paulatinamente: jefa departamento I+D y principales jefes de desarrollo, responsables del departamento de sistemas (comisionistas).

Paso 3: Sesiones de formación de las técnicas y herramienta (IC, TFS, Sonarqube, Test-Feeback).

En este caso se realizarán tres sesiones (una teórica y dos prácticas), de 1:30 y dos horas respectivamente. En las sesiones prácticas se realizará una demo del uso de las herramientas.

Paso 4: Seguimiento y evolución. No basta con decir que lo que se hace es importante, hay que verificar que se haga bien y que funciona. ¿Qué no es apropiado? ¿Qué se puede cambiar? E ir integrando paulatinamente más herramientas.

El seguimiento será constante. Propongo usar la propia reunión de retrospectiva del sprint para verificar y evaluar la implantación. Al concluir la implantación del proyecto usado como modelo, se realizará una reunión donde se analizará la efectividad o no de las prácticas y se evaluará el proceso de manera global.

Una vez hecha la propuesta para el diseño y desarrollo, en el siguiente capítulo se estudiarán los resultados de la misma y se llegará a conclusiones sobre la efectividad de su implantación.

Es evidente que no existen fórmulas mágicas ni algoritmos perfectos que se amolden a todos los casos, es cuestión de probar e ir ganando experiencia. A medida que nuestro equipo va madurando y todos los departamentos van conjuntamente sincronizados, se hace más fácil ir ajustando la maquinaria para poco a poco ir obteniendo los resultados deseados.

Al concluir mi tiempo estimado para prácticas en la empresa, y por la demora en la contratación del equipo encargado del desarrollo, solo he podido estar en el inicio de la implantación, y no he podido seguir de la mano toda la adopción sobre todo de las practicas técnicas, pero he estado al tanto de los resultados y experiencias obtenidos.

Luego de implantada la propuesta, he realizado entrevistas de satisfacción a los principales activos dentro del departamento. Lo anteriormente dicho, unido a una observación por mi parte del comportamiento del personal (para comprobar si la propuesta es de agrado o no), y como eran desarrolladas las tareas, ha hecho posible que pueda llegar a conclusiones que detallo más adelante.

Por otra parte, al no poder estar durante todo el proceso, he procurado la definición de métricas o KPI (*key performance indicator*), que se podrán usar para medir el nivel de rendimiento de varias partes del proceso.

Comenzaré analizando la implantación de SCRUM y TFS.

4.1 Implantación de SCRUM y TFS

Un reto importante para ANIMSA son las contrataciones tradicionales que se mantienen con costo, tiempo y alcance fijos. Esto se debe a la burocracia de las organizaciones públicas para la realización de las contrataciones. Teniendo en cuenta que el proceso de agilidad en las empresas abarca la contratación, se hace difícil la adopción de prácticas ágiles en este sentido en todo su esplendor, concretamente para la contratación del equipo de desarrollo.

No obstante, una vez creadas las pautas iniciales se pudieron mantener otras prácticas ágiles, en el desarrollo de aplicaciones dentro del departamento de I+D.

La decisión de implantar prácticas ágiles poco a poco, se considera acertada. Los logros obtenidos sirvieron para crear un ambiente positivo en el departamento. Se estima que próximos proyectos se realicen con estas prácticas.

Por otra parte, el anuncio del agilismo de manera pública si bien no trajo inconvenientes, no se puede decir que haya sido del todo satisfactorio, ya que no se evidencia que otros departamentos quieran de momento sumarse a estas prácticas. Aunque sí se mostró un interés por el departamento de Mantenimiento para la implantación de TFS como gestor de proyectos y código fuentes.

Otra situación que no se ha concretado es la del uso de un coach interno para la difusión de prácticas ágiles. Esto se debe a que no se encuentra una persona del todo capacitada para la realización de estas funciones.

La falta de conocimiento por no tener un adecuado entrenamiento hace que se tenga una concepción aún errada del sprint y sobre el seguimiento que se debe tener sobre él. En un inicio no se era consciente de que el sprint es el centro de SCRUM y todos los procesos a su alrededor son de vital importancia, pero conforme fueron involucrándose en el proceso se fue ganando en conciencia.

La propuesta realizada para la creación del entorno de desarrollo se ha cumplido en la medida de lo posible. Algo a aportar en este sentido es que llegamos a la conclusión de que no deberíamos obsesionarnos con configurar todos los sprints. Se pudo prever que los mismos podrían cambiar si el equipo en su momento considera que son, o muy largos, o demasiado breves. Por eso creo que es mejor para el equipo ir configurando todo sprint a sprint.

La actividad para estimar las capacidades ha sido muy efectiva ya que, en principio no nos toma más de diez minutos y a cambio, gracias a estos datos, más adelante, somos capaces de medir el trabajo que podemos asumir a lo largo de un sprint.

El dar la capacidad de auto-asignar las tareas al equipo de desarrollo, ha tenido buenas críticas, ya que dota al mismo de cierta autonomía y holgura para realizar el trabajo. Es una muestra además de confianza en las capacidades del equipo, y hace que el ambiente de trabajo sea más ameno.

Algo que se debe señalar es que no siempre se respetaba el orden de prioridad que tenían las tareas. Además, no se hacía una constante reordenación del listado de tareas pendientes.

Un problema habitual fue el no cumplimiento de la disciplina de las reuniones diarias, lo que traía como resultado la postergación de tareas por no tomarse la decisión con la agilidad requerida. Es importante resaltar que en este sentido al encontrarse el equipo de desarrollo fuera de la entidad de ANIMSA y al ser además subcontratados de otra empresa y tener su propia metodología, se hizo sumamente complejo el poder realizar las reuniones diarias, por lo que se toma la decisión de no realizarlas.

Estimar el tiempo de desarrollo de un proyecto de forma exacta es una labor que se torna casi imposible en muchos casos. Por eso, usar puntos de historia es una buena técnica de estimación. Pero dentro del departamento no se reconoce su efectividad y sencillez para planificar los sprints. Prefieren trabajar en base a horas, y estimar la capacidad bajo ese criterio. Esto sucede porque la filosofía de la empresa para la contratación de los equipos de desarrollo es por horas fijas, y esto no permite poder hacer estimaciones de esta manera. Es importante aclarar que esto no es una cuestión que atañe solo a ANIMSA sino también implica a las organizaciones que se relacionan con ella, y que están sujetas a cumplir determinadas normas por tratarse de administración pública.

A criterio personal, creo que es hora de darle una vuelta a todo esto, y empezar a cambiar el modo en que las organizaciones públicas gestionan las contrataciones. Quizás no es necesario cambiar todo, pero sí mejorar procesos ineficientes que afecten a los tiempos y los costos.

En cuanto a la implantación de prácticas ágiles de manera paulatina, se ha evidenciado una buena aceptación. Las prácticas propuestas han ido siendo aceptadas y adoptadas por algunos de los miembros de otros proyectos.

Surgieron varios inconvenientes sobre todo al inicio, que radicaban principalmente en la resistencia al cambio que se notaba en algunos miembros de otros equipos, ligado más que nada a un escepticismo de nuevas prácticas que suelen ser complejas de desarrollar cuando no se adoptan con la conciencia necesaria. Se percibe que la definición del proceso de SCRUM es una tarea que conlleva tiempo.

Un aspecto importante es que no es fácil constituir equipos de trabajo de tres o más personas dentro del departamento por la propia dinámica de los proyectos, por lo que en ocasiones figuras como *SCRUM máster* o *Product Owner* no están del todo definidas.

En ocasiones las ideas eran anotadas, pero no debidamente subidas al TFS como parte de los *feedback* resultado de las reuniones pertinentes. Esto viene ligado a que cuesta trabajo hacer la conciencia por tener que dedicar tiempo para subir la información al servidor. Ligado a esto se debe trabajar más en la buena práctica para crear historias de usuario mejor estructuradas y detalladas.

Por otro lado, entre los logros obtenidos podemos considerar de gran importancia que se gana una mejor comunicación del equipo, obteniéndose un conocimiento global de todo el trabajo que se realiza.

El uso de nuevas tecnologías como TFS evidencia mayor motivación, y se muestra bastante colaboración de los miembros para el aprendizaje de la nueva herramienta. La propia adopción de SCRUM hace que se esté a la vanguardia con nuevas tecnologías. En este sentido, estimo que los logros más significativos están enfocados en el control y seguimiento de las actividades del proyecto. Esto ayudará a tomar acciones correctivas para prevenir problemas potenciales.

La asignación del tipo de actividad de una tarea fue bien valorada, ya que automáticamente se podía ver si se excedía en el tiempo de trabajo planificado para dedicar a esa actividad o por el contrario si podemos comprometernos a realizar aún más trabajo.

Las entregas frecuentes y revisiones del producto han hecho que se minimicen posibles riesgos de una entrega completa.

El uso de la herramienta de *feedback*, aunque en el inicio no era muy utilizada poco a poco fue gustando, sobre todo porque mantenía al personal informado, aun cuando algún miembro no estuviera presente en las reuniones.

La herramienta TFS ha sido de gran utilidad para organizar el trabajo, y para demostrar que en muchas ocasiones se desperdiciaban recursos por la mala planificación. La gráfica conocida como *"burn down chart"* tiene gran aceptación porque muestra claramente el avance o estancamiento del sprint. En la Fig.28 muestro un ejemplo del primer sprint del proyecto de prueba.

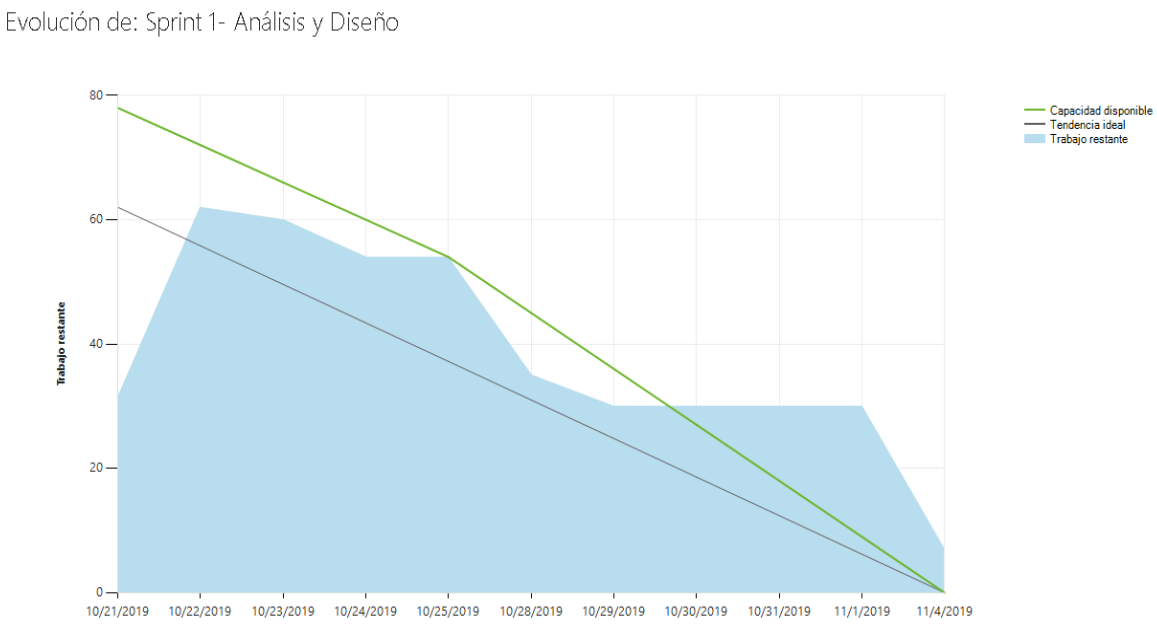


Fig. 28 Gráfica burn down chart (Sprint1 - Análisis y Diseño)

Como se observa en Fig.28 si estamos por debajo de la línea de tendencia ideal es probable que se haya sobre-estimado el esfuerzo de las tareas. En cambio, si estamos por encima significa que vamos con retraso. En este caso se nota sobre todo al final del sprint que se va muy por encima de la tendencia ideal. Luego de analizar, se llega a la conclusión de que esto se debe a que no se respetó la pauta de actualizar diariamente las tareas. Precisamente una de las dificultades encontradas ha sido la falta de uso de las normas planteadas para mejorar este proceso.

Hasta el momento los tiempos establecidos para la realización de las diferentes reuniones se consideran adecuados y suficientes.

4.2 Prácticas técnicas

La IC ha sido una de las prácticas más novedosas para el departamento. Entre las dificultades presentadas se encuentra la resistencia por el cambio de los procesos habituales.

Precisa de servidores y entornos adicionales para su puesta en marcha. El servidor de Sonarqube fue montado por un externo que ya no se encuentra en la entidad por lo que el conocimiento del mismo es prácticamente nulo, lo que ha desembocado en la necesidad de una nueva preparación del mismo.

Uno de los inconvenientes es que la licencia de Sonar para Visual Basic, al ser de pago en el momento que se monta el servidor, ya se encontraba caducada, por lo que solo se podía hacer uso de las tareas de Sonar en algunos tipos de proyectos. En el caso del proyecto de demostración no era posible su uso. Gracias a que la licencia fue liberada hace poco tiempo para versiones mayores a la 7.4 de Sonar, solo nos vimos en la necesidad de actualizar el servidor de Sonarqube.

Por otra parte, el uso de Sonarqube ha sido de incalculable valor demostrando principalmente su capacidad para no solo reportar errores en el estilo o bien en las prácticas de programación, sino también para la detección de vulnerabilidades de diversos tipos, como inyección SQL, entre otras. Su implantación aún está en proceso de desarrollo, ya que el uso de esta nueva técnica implica nuevos conocimientos que pueden volverse complejos para alguno de sus miembros.

Entre las valoraciones positivas que se obtienen de IC por los miembros, está la facilidad de detección temprana de errores. Además, la existencia de un registro exacto de cada una de las modificaciones ha sido positiva en el departamento.

Se avizora una evidente interacción entre el cliente y el equipo, ya que éste se involucra mucho más en todo el proceso y va recibiendo incrementos del producto de manera paulatina sin tener que esperar hasta la finalización total del mismo. Cabe destacar que en el caso del proyecto de demostración, el cliente es el propio departamento de compras de ANIMSA lo que hace que sea aún más fácil la comunicación encontrándose en la misma localización.

Los participantes consideran que se debe tomar lo mejor de cada práctica para crear un proceso adecuado a las necesidades puntuales de ANIMSA y no solo implantar el framework de SCRUM como única solución a todo.

4.3 Propuesta para medir resultados

Al dilatarse en el tiempo la contratación y al culminarse mi período de prácticas, me encuentro en una situación en la que no podía sacar muchas más conclusiones que las que hasta el momento he podido realizar. Pero eso no significa que el departamento no pueda continuar con el estudio de la propuesta.

Es por eso que planteo varias maneras para poder medir los resultados.

Para evaluar el desempeño del trabajo de los equipos, se medirá primeramente mediante la satisfacción del cliente y la calidad con la que se entreguen los productos. Lógicamente, un equipo que se adapta a los cambios y es capaz de realizar entregas siempre incrementando el valor del producto, consigue que el resultado se adapte a lo que el cliente quiere y aumente su satisfacción. Esto se puede obtener de los *feedback* que se extraen en cada una de las reuniones donde el cliente exprese su criterio del trabajo realizado.

Pero esas medidas pueden no ser suficientes, quizás porque no son tan cuantificables como otras que propongo a continuación.

4.3.1 Métrica de calidad

Para medir la calidad del producto propongo usar el número de incidencias (bugs) detectadas en cada sprint. Sugiero registrar, por una parte, los incidentes que surgen en ambiente productivo, y por otra la cantidad de incidentes descubiertos durante las pruebas. Finalmente, se puede obtener una media de cada tipo de incidencia por cada ambiente y comparar con otros proyectos. Para la comparación entre proyectos hay que tener en cuenta el tamaño del mismo, ya que mientras más grande sea ese proyecto (en relación al tiempo) mayor será la probabilidad de incidencias. Es por eso que propongo establecer una proporción cantidad de incidencias / cantidad de sprints.

Ilustro en Fig.29 y Fig.30, un ejemplo hipotético:

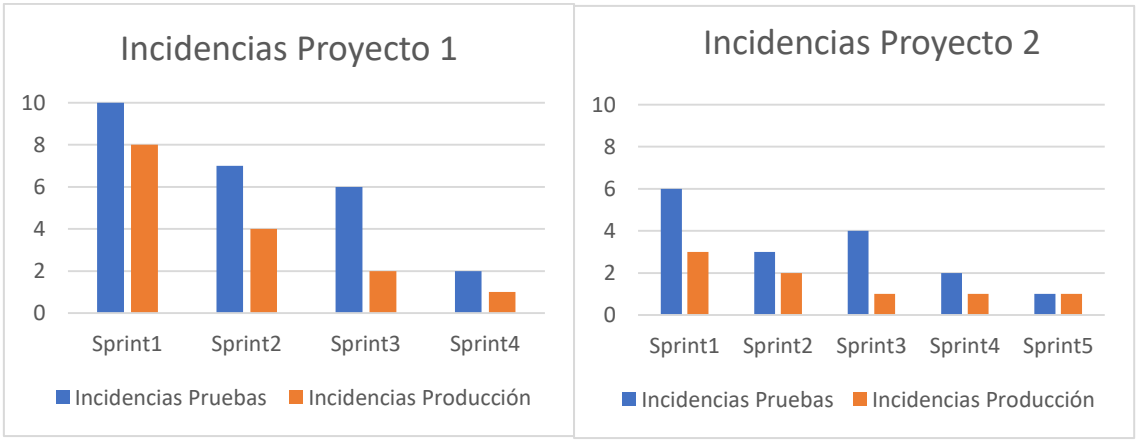


Fig. 29 Ejemplo incidencias por proyecto

Incidencias de prueba Proyecto 1 = 25 / 4

Incidencias de producción Proyecto 1 = 15 / 4

Incidencias de prueba Proyecto 2 = 16 / 5

Incidencias de producción Proyecto 2 = 8 / 5

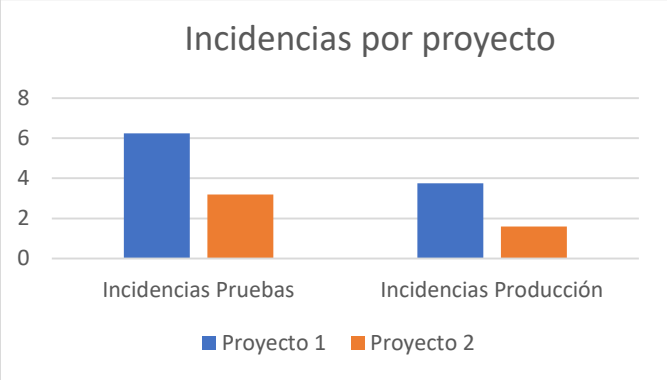


Fig. 30 Ejemplo comparativa de incidencias generales de los dos proyectos

En el ejemplo anterior (Fig.30), aun cuando el proyecto dos tiene una iteración más que el proyecto uno, sus niveles de incidencias tanto de pruebas como de producción son menores.

4.3.2 Métrica Valor agregado

Considero necesario este indicador para poder medir el valor de negocio entregado por el equipo en el tiempo.

El gráfico de "*burn down chart*" será nuestro principal aliado para poder medir si la planificación realizada es o no la correcta. Mediante el registro, visualización y actualización diaria de la cantidad de horas que van restando para finalizar todas las tareas para entregar las historias de usuario comprometidas en el sprint.

Paralelamente al esfuerzo restante registrado en el "*burn down chart*", TFS brinda otro gráfico estilo *Burnup*, ver Fig.31 (ejemplo de una planificación inicial del proyecto), que representa en el eje X la cantidad de Valor de Negocio (por estado) y en el eje Y los días transcurridos del sprint.

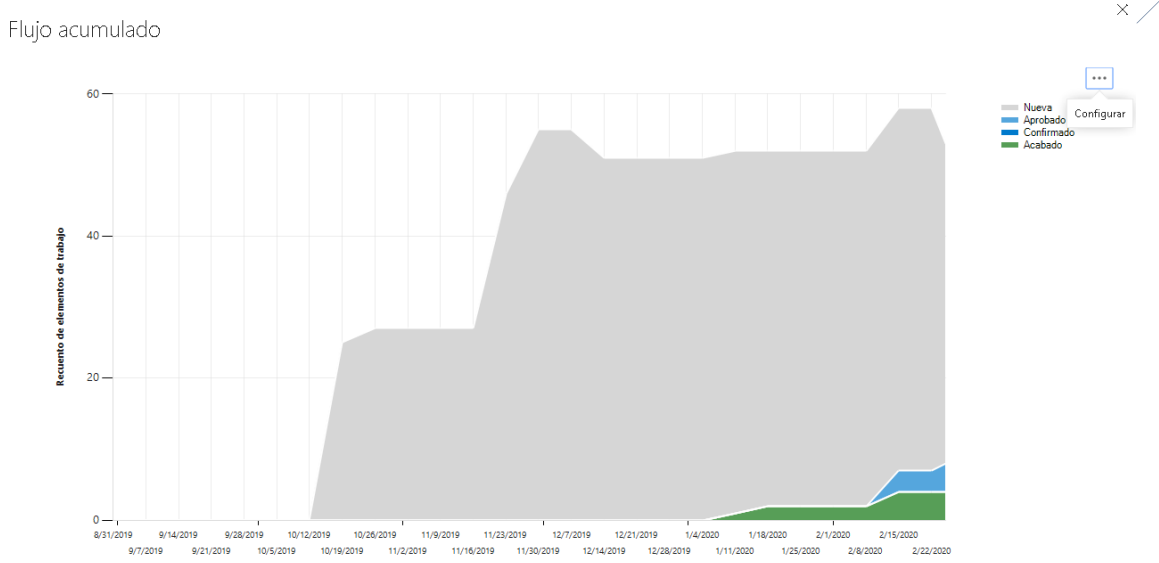


Fig. 31 Diagrama del flujo acumulado inicial del proyecto

Si se observa poco valor entregado es probable que lo que es de valor no se esté priorizando como es debido. En este sentido el *Product Owner* debe hacer una buena asignación de prioridad de las historias de usuario y reflexionar sobre qué mecanismos está usando para dar prioridad, ya que quizás no son los más apropiados. Si además comienzan a surgir problemas que impiden reiteradamente la entrega de requisitos importantes, es mejor en el próximo sprint no comprometer tantas historias y así aseguramos su entrega.

4.3.3 Métrica Percepción del equipo

Creo interesante una métrica que evalúe el producto desde la apreciación del equipo. Este indicador se puede ejecutar sobre el producto entregado en cada sprint, pero desde el punto de vista técnico.

Propongo una escala con diferentes variables, y cada miembro del equipo propone un valor, ver Fig.32.

Variables:

Orgullo: Estamos orgullosos de lo que hicimos. Deseamos contar a todos lo bueno que es. No podría ser mejor.

Bomba de tiempo: Muchos fallos técnicos que pueden explotar en cualquier momento. No deseamos estar presentes cuando se comience su uso.

Adecuado: Hay posibilidades de mejoras, pero hemos entregado algo técnicamente robusto. Funciona acorde a los requisitos y no falla.

Utilizable: El producto funciona, pero es más que probable que en el futuro traiga problemas. Habría que rehacer algunas cosas para mejorar su funcionamiento.

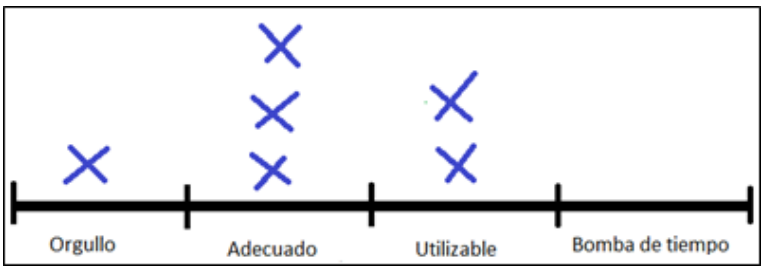


Fig. 32 Ejemplo de uso de escala percepción del equipo

Es necesario velar por la evolución de este indicador. A continuación, expongo algunas ideas para su evaluación.

Si se observa un aumento de las variables de "Bomba de tiempo" o "Utilizable", es probable que el equipo no esté teniendo en cuenta la necesidad de la calidad del producto, o se estén pasando por alto muchos problemas técnicos.

Propongo:

Emplear nuevas prácticas que mejoren la efectividad técnica del equipo. También se deben evaluar las prácticas desarrolladas hasta el momento, ya que es probable que no se estén desarrollando de la mejor manera. Se debe cuidar la motivación del equipo, pues influye directamente en su desempeño.

Si por otra parte se observa un aumento de las variables "Orgullo" y "Adecuado", significa que el equipo está evolucionando e incluso perfeccionando su trabajo. No obstante, hay que tener especial cuidado de no crear un ambiente de falsa seguridad. Hay que estar en constante mejora.

4.3.4 Métrica de compromiso

Con este indicador lo que persigo es aproximar el porciento de compromiso que mantiene el equipo sobre los objetivos de valor para el negocio acordados en el sprint. En otras palabras, se puede evidenciar si las nuevas prácticas están ayudando a mejorar la responsabilidad de trabajo del equipo y que tan bien cumplen la planificación.

Para poder realizar el ejercicio es necesario registrar las horas introducidas en las tareas de cada historia de usuario del *backlog* y sumarlas para todas las tareas trabajadas en el sprint. Luego se podrán comparar contra las horas disponibles del equipo.

Fórmula:

C: Compromiso

HS: Horas metidas en tareas del sprint que fueron entregadas.

HT: Horas de trabajo planificadas del equipo en el sprint.

C = HS / HT * 100
C = 80 / 112 * 100
C = 71.42%

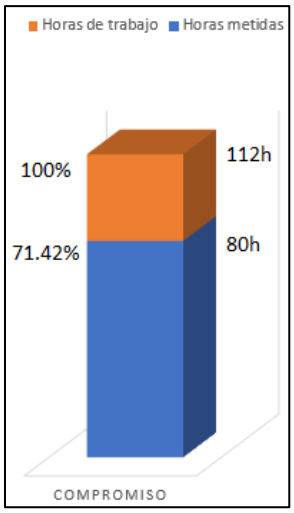


Fig. 33 Ejemplo que ilustra (Horas de trabajo / Horas metidas)

El ejemplo de la Fig.33 se basa en un total de 112 h que se calculan como base de trabajo para los sprint del proyecto de demostración (teniendo en cuenta horas ideales). Las HS (horas metidas) no son horas reales en el proyecto, son un número aleatorio para entender mejor el ejemplo.

Este indicador se debe revisar constantemente. Si el compromiso sube significa que el equipo está trabajando bien enfocado, y que están hablando un mismo idioma. Es por tanto un equipo bien encaminado y que funciona.

Si en cambio el compromiso baja, probablemente no se esté planificando de la mejor manera. Puede que estén surgiendo muchas tareas que no estaban en el plan o que no se esté respetado la prioridad. También puede estar pasando que haya muchas distracciones en el equipo. Si esto sucediera, propongo replanificar y registrar todas las

interrupciones y tareas para poder analizar el problema. Velar por que el equipo se centre más en las tareas prioritarias.

4.3.5 Métrica resolución de incidencias

Esta métrica resulta bastante sencilla de implementar. Consiste en evaluar el tiempo medio de la resolución de una incidencia (bugs registrados en el sistema), ver Fig.34. De esta forma se evalúa el tiempo de reacción del equipo para solventar los problemas.

Fig. 34 Ejemplo de Bugs registrados en el proyecto

En general, sugiero evaluar estos indicadores al finalizar cada una de las reuniones de retrospectiva como parte de la técnica *circle feedback* propuesta en la reunión de retrospectiva. Una vez finalizados todos los sprints, estos datos pueden ser interesantes para la toma de decisiones, detectando patrones y tendencias en el equipo.

Al ser este un primer proyecto de prueba, no se tiene un umbral definido para poder comparar y valorar que tan bien o mal está nuestro proyecto con respecto a otros. En cualquier caso, se podría realizar un estudio y con estos elementos contrastar con datos recabados de proyectos pasados.

De todas formas, en adelante usar estas técnicas y herramientas nos marca una nueva forma de trabajo que nos brinda de manera muy sencilla la posibilidad de obtener elementos para, en el futuro, poder hacer mejores comparaciones y tener métricas más acertadas para evaluar los procesos y comenzar un camino de mejora continua.

5.1 Conclusiones

En este trabajo se ha presentado la experiencia sobre el uso de prácticas SCRUM dentro de un proyecto piloto dentro del departamento de I+D de la empresa ANIMSA. Para lograr esto, se realiza un proceso de evaluación y diagnóstico de prácticas ágiles y se crea una correcta formación del equipo.

En este informe se muestra que el uso de SCRUM e IC fue un desafío ya que la cultura de la empresa no tenía una clara definición de la agilidad. La resistencia al cambio fue notable sobre todo en un inicio, lo que fue cambiando a medida que se evidenciaban resultados.

No obstante, aunque la intención de adoptar estas prácticas en otros proyectos era positiva, entraba en una disyuntiva sobre qué persona sería la encargada de promover y velar por estas prácticas. Esto se debe principalmente a problemas culturales dentro de la organización para poder escalar ágilmente. En el futuro se propone la presencia de un coach interno que trabaje en pos de lograr más resultados no solo en el departamento de I+D sino también en los demás departamentos de ANIMSA.

Con respecto a la herramienta TFS, se considera adecuado adoptarla no solo como control de código fuente sino para gestionar todo el ciclo de vida de los proyectos. Por tanto, quedará implantado su uso en los departamentos de Innovación - Desarrollo y Mantenimiento - Soporte.

Es importante que en el futuro cada sprint esté debidamente aprobado por el cliente para evitar retrocesos que puedan ser costosos.

Aunque se concluye que no se puede afirmar categóricamente que la metodología de gestión de proyecto SCRUM sea mejor que otra para el departamento, sí se puede determinar que SCRUM es una opción viable a usar en el mismo, sin descartar otras metodologías.

Se consideran oportunas más capacitaciones al personal sobre todo en el uso de las herramientas ágiles.

En tal caso, se considera oportuno el uso de SCRUM aunque no en su totalidad por las características de la empresa al tratarse de una entidad pública, y por ende sujeta a determinadas normas que pueden no compatibilizar con el marco de trabajo de SCRUM para adoptarlo de manera íntegra. Implantar SCRUM en su totalidad implica una serie de procesos para los que el departamento no se encuentra preparado. La propuesta es realizar algunas de las prácticas que esta metodología propone e ir incorporando otras progresivamente a medida que se vaya evolucionando.

Aún luego de todos los impedimentos, se ha logrado la implantación real de prácticas ágiles en el departamento. Esto se debe en gran medida a que se entiende la importancia de la adopción de las mismas. Se ha logrado además crear una conciencia de la necesidad del agilismo dentro del departamento.

Por otra parte, queda implantado el uso de TFS para la gestión de los proyectos dentro del departamento.

Las organizaciones año tras año hacen más hincapié en la necesidad de implantar metodologías y técnicas ágiles, porque comprenden su necesidad en el mundo tan cambiante al cual nos enfrentamos hoy, donde el aumento de la automatización de las pruebas se hace necesario para transformar la cultura entre el desarrollo y las operaciones en las organizaciones.

De manera general, los resultados del TFM demuestran que el uso de SCRUM con prácticas de IC y herramientas como TFS, no solo para esta empresa sino de manera general, evidencia un futuro prometedor para este tipo de procedimientos que buscan continuamente la mejora de los procesos en las organizaciones.

5.2 Líneas Futuras

Luego de obtener conclusiones a lo largo de todo el proceso, se identifican líneas de trabajo futuro que se podrían trabajar más adelante.

Se debe estudiar la inclusión de nuevas herramientas técnicas para integración continua, e ir adaptándolas a los procesos de manera paulatina. En este sentido considero oportuno la evolución de manera natural a la entrega continua. En el futuro se puede considerar si el despliegue continuo es una opción viable en ANIMSA.

A nivel personal considero conveniente realizar un estudio que puede servir como tema de doctorado, para encontrar posibles modificaciones de metodologías ágiles para compatibilizar mejor en entornos de la administración pública en España.

Bibliografía

[1] A. d. R. l. ANIMSA, «Manual de Bienvenida,» Pamplona, 2019.

[2] E. d. d. d. i. y. d. d. ANIMSA, «ANIMSA,» [En línea]. Available: https://www.animsa.es/mision-y-vision/.

[3] Office Timeline, «Office Timeline,» 2020. [En línea]. Available: https://www.officetimeline.com/blog/software-development-methodologies-timeline. [Último acceso: 23 01 2020].

[4] K. Beck, M. Beedle, K. Schwaber y J. Sutherland, «Manifiesto Ágil,» 2001. [En línea]. Available: http://agilmanifesto.org. [Último acceso: 16 11 2019].

[5] Varios, «Wikipedia,» Fundación Wikimedia, Inc, 23 09 2019. [En línea]. Available: https://es.wikipedia.org/wiki/Manifiesto_%C3%A1gil. [Último acceso: 16 11 2019].

[6] A. Cockbur y J. Highsmith, «Agile Software Development: The Business of Innovation,» Barry Boehm, Computer Science Department, University of Southern California, Los Angeles, CA 90089; boehm@ sunset.usc.edu, Los Angeles, 2001.

[7] E. M. Schön, D. Winter, M. J. Escalona y J. Thomaschewski, «ResearchGate GmbH,» 2008-2019. [En línea]. Available: https://www.researchgate.net/publication/316116540_Key_Challenges_in_Agile_Requirements_Engineering. [Último acceso: 17 11 2019].

[8] The Standish Group International, Inc., «CHAOS REPORT 2015,» 2015.

[9] Ó. AGUDO, *GESTIÓN ÁGIL DE PROYECTOS CON SCRUM*, Pamplona, 2019.

[10] COLLABNET VERSIONONE, «The 13th annual State of Agile Report,» EEUU, 2018.

[11] A. Menzinsky, G. López y J. Palacio, Guía de Scrum Manager. Manual para consulta y formación de Scrum Manager, Safe Creative, 2016.

[12] N. Zya y M. Suaib, « ResearchGate GmbH,» 2008-2019. [En línea]. Available: https://www.researchgate.net/publication/272944740_AGILE_METHODOLOGIES_IN_SOFTWARE_DEVELOPMENT. [Último acceso: 17 11 2019].

[13] COLLABNET VERSIONONE, «6th Annual State of Agile Report,» EEUU, 2011.

[14] Microsoft, «Microsoft,» 2019. [En línea]. Available: https://docs.microsoft.com/es-es/visualstudio/releasenotes/media/upgradematrix.png. [Último acceso: 20 12

2019].

[15] IEEE - Instituto de Ingenieros Eléctricos y Electrónicos, «IEEE Xplore,» 2019. [En línea]. Available: https://ieeexplore.ieee.org/document/65147/authors#authors. [Último acceso: 12 12 2019].

[16] M. Fowler, «MartinFowler.com,» 2019. [En línea]. Available: https://www.martinfowler.com/articles/continuousIntegration.html. [Último acceso: 12 12 2019].

[17] . K. Schwaber y J. Sutherland, «SCRUM GUIDES,» 2018. [En línea]. Available: https://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-es.pdf. [Último acceso: 17 11 2019].

[18] L. M. de Souza Mariz , . A. C. C. Franca y . F. . Q. da Silva, «An Empirical Study on the Relationship between the Use of Agile Practices and the Success of Software Projects that Use Scrum,» 2013.

[19] Equipo de desarrollo, «Scrum.org,» 2019. [En línea]. Available: https://www.scrum.org. [Último acceso: 04 12 2019].

[20] Equipo Microsoft, «Microsoft,» 2019. [En línea]. Available: https://docs.microsoft.com/en-us/azure/devops/user-guide/services?view=azure-devops. [Último acceso: 05 12 2019].

[21] Grupo Certia Desarrollo y Consultoría, «Certia Desarrollo y Consultoría,» 2019. [En línea]. Available: https://www.certia.net/TFS. [Último acceso: 05 12 2019].

[22] ATLASSIAN, «ATLASSIAN,» 2019. [En línea]. Available: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment. [Último acceso: 18 12 2019].

[23] Google, «Google Cloud,» 2019. [En línea]. Available: https://cloud.google.com/solutions/continuous-integration/?hl=es-419. [Último acceso: 20 12 2019].

[24] Gartner, «Gartner.,» 2019. [En línea]. Available: https://www.gartner.com/doc/reprints?id=1-6K0PBLO&ct=190422&st=sb. [Último acceso: 23 12 2019].

[25] P. Y. Khudyakov, A. Y. Kisel'nikov y I. M. Startc, «Research Gate,» 2019. [En línea]. Available: https://www.researchgate.net/publication/325288940_Version_control_system_of_CAD_documents_and_PLC_projects. [Último acceso: 05 12 2019].

[26] G2, «G2,» 2019. [En línea]. Available: https://www.g2.com/categories/continuous-integration#grid. [Último acceso: 20 12 2019].

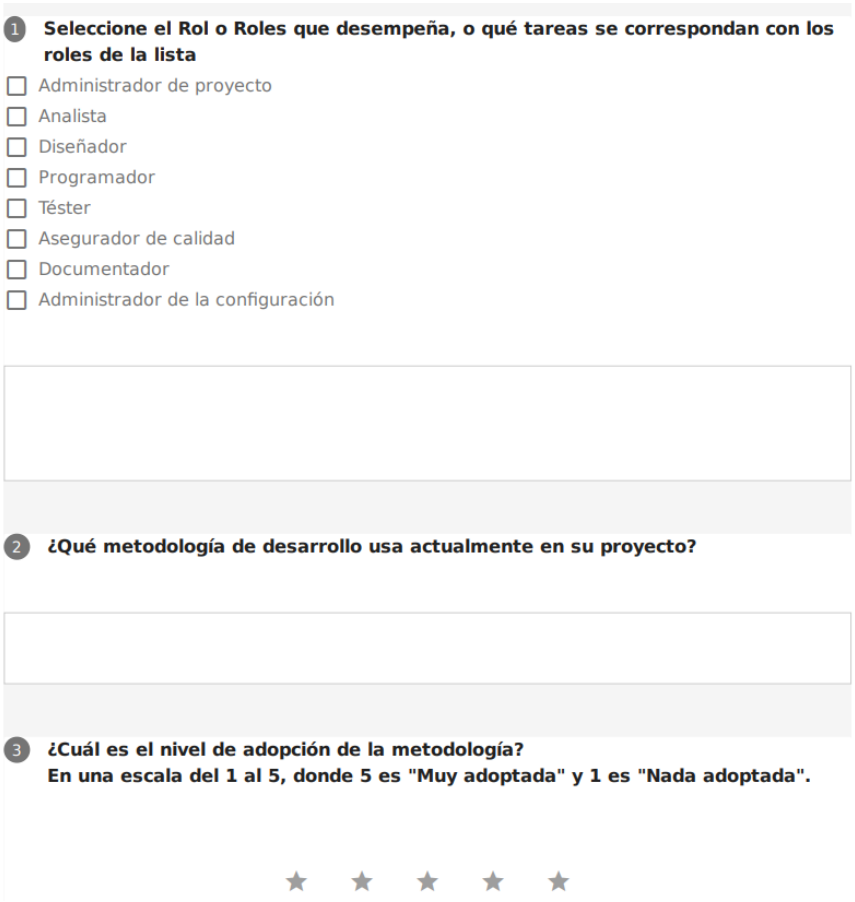
[27] M. Cohn, «Chapter 2. Adapting to Scrum,» de *Succeeding with agile. Software development using Scrum*, Addison-Wesley, 2009.

[28] J. García Navarro, «Estudio comparativo de metodologías, herramientas y WIKI de soporte para la gestión de proyectos de desarrollo de software,» Catalunya, 2018.

[29] P. Rodríguez, «Researchgate,» 2020. [En línea]. Available: https://www.researchgate.net/publication/275654650_COMBINING_LEAN_THINKING_AND_AGILE_SOFTWARE_DEVELOPMENT_How_do_software-intensive_companies_use_them_in_practice/figures?lo=1. [Último acceso: 22 01 2020].

Anexos

Anexo #1 Encuesta realizada por autor, para el análisis de la situación actual en ANIMSA



4 **¿Qué tareas realiza mediante esta metodología?**

5 **¿Cuál o cuáles de los siguientes aspectos se corresponden con la metodología actual usada?**

- [ ] Simplicidad
- [ ] Facilidad de uso
- [ ] Es la adoptada por la empresa
- [ ] Esta de moda
- [ ] Llevo muchos años usándola
- [ ] No conozco otra
- [ ] Buenos resultados
- [ ] Otro

6 **¿Siempre usa el mismo tipo de metodología?, si la respuesta es No, diga cuál ha usado además.**

7 **Seleccione concretamente qué metodología ha usado**

- [ ] XP
- [ ] SCRUM
- [ ] CMMI
- [ ] KANBAN
- [ ] RUP
- [ ] Otra

8 **¿Qué herramienta de gestión de proyecto usa actualmente?**

9 **¿Cuál es el nivel de adopción de la herramienta usada?**
En una escala del 1 al 5, donde 5 es "Muy adoptada" y 1 es "Nada adoptada".

★ ★ ★ ★ ★

10 **¿Qué tareas realiza mediante esta herramienta?**

11 **¿Cuál o cuáles de los siguientes aspectos se corresponden con la herramienta actual usada?**

- ☐ Simplicidad
- ☐ Facilidad de uso
- ☐ Es la adoptada por la empresa
- ☐ Está de moda
- ☐ Llevo muchos años usándola
- ☐ No conozco otra
- ☐ Buenos resultados
- ☐ Otro

12 **¿Siempre usa la misma herramienta?, si la respuesta es No, diga cuál ha usado además.**

13 **Seleccione concretamente qué herramienta ha usado**

- [ ] OpenProject
- [ ] Collabtive
- [ ] TFS
- [ ] Intranet ANIMSA
- [ ] Microsoft Project
- [ ] Otra

14 **¿Alguna vez ha realizado integración continua en alguno de sus proyectos? Si la respuesta es SI, comente su impresión.**

15 **¿Cree usted importante el uso de integración continua en los proyectos? ¿Por qué?**

FIN