Implantación de Scrum en la empresa con el soporte de la herramienta JIRA V2

Nombre del Alumno: A.C.M
Ingería Técnica en Informática de Gestión

Nombre Consultor: Xavier Martínez Munné

Entrega Final: 13-01-2016

B) GNU Free Documentation License (GNU FDL)

C) Copyright

Control de cambios:

Documento:		Implantación de Scrum en la empresa con el soporte de la herramienta JIRA Agile.	
Fecha	**Versión**	**Autor**	**Cambios**
14/10/2015	V1R0	A.C.M	Versión inicial, entrega PEC1.
21/10/2015	V1R1	A.C.M	Revisado apartado licencias.
23/10/2015	V1R2	A.C.M	Revisados los objetivos del proyecto con variables medibles y estrategia clara para conseguirlos.
24/10/2015	V1R3	A.C.M	Revisados los objetivos del proyecto con variables medibles y estrategia clara para conseguirlos.
26/10/2015	V1R4	A.C.M	Especificación de horas dedicadas al proyecto dentro del Gantt.
26/10/2015	V1R5	A.C.M	Se especifican claramente en Gantt y en la descripción de la planificación las fechas claves de las entregas.
28/10/2015	V1R6	A.C.M	Inclusión en el Gantt y en la descripción de las tareas la preparación de la defensa del proyecto y la preparación de la presentación.
30/10/2015	V1R7	A.C.M	Identificación de los recursos y los responsables de las tareas.
01/11/2015	V1R8	A.C.M	Inclusión de los costes del proyecto, según los perfiles que ejecutan las tareas.
07/11/2015	V1R9	A.C.M	Revisado el *abtract*.
15/11/2015	V1R9.1	A.C.M	Inclusión de la primera parte del contenido de la memoria relacionado con la metodología Scrum. PEC2.
13/12/2015	V1R10	A.C.M	Actualizado progreso del Gantt e incorporación de perfiles.
13/12/2015	V1R11	A.C.M	Mejora del formato e inclusión de alguna tabla, gráficos e imágenes.
13/12/2015	V1R12	A.C.M	Modificación criterios en las citas dentro del texto, revisión de gramática y ortografía.
14/12/2015	V1R13	A.C.M	Inclusión del contenido relacionado con JIRA. PEC3.
11/01/2016	V2	A.C.M	Inclusión de las conclusiones del trabajo y correcciones varias. Entrega Final.

FICHA DEL TRABAJO FINAL

Título del trabajo:	Implantación de Scrum en la empresa con el soporte de la herramienta JIRA Agile.
Nombre del autor:	A.C.M
Nombre del consultor:	Xavier Martínez Munné
Fecha de entrega (mm/aaaa):	01/2016
Área del trabajo final:	Gestión de Proyectos
Titulación:	*Ingeniería Técnica Informática de Gestión*

Resumen del trabajo (máximo 250 palabras):

En el mundo empresarial se está detectando que el uso de las nuevas tecnologías es fundamental para la optimización de los procesos de negocios y aumentar la eficiencia. Por esto, la mayoría de las compañías necesitan disponer de un *software* adecuado que cubra todas las necesidades empresariales. Este *software* puede ser tanto una solución empresarial de terceros, una solución a medida, o bien un desarrollo de la propia empresa con recursos propios. Tanto si es de una forma como de otra, es fundamental que el ciclo de vida del *software* a crear sea óptimo en su elaboración, sea eficiente en su implementación, sea coherente con lo que se necesita, sea modificable y, por su puesto, tenga una calidad en su entrega.

A día de hoy en muchas compañías que intentan crear aplicaciones informáticas tienen ciertas carencias, ya que no utilizan ningún tipo de metodología a la hora de la construcción, y tienen una serie de perjuicios muy considerable que se corregiría si utilizarán una metodología de trabajo.

En el presente trabajo se estudiará cómo implementar una metodología de desarrollo para solucionar los problemas habituales de construcción de aplicaciones que se suelen cometer en las empresas; para ello se analizará la implantación de Scrum. Para que dicha implantación sea exitosa será necesario utilizar una herramienta para gestionar el ciclo de vida de desarrollo del *software* y para ello en el presente caso se

utilizará un *software* denominado JIRA Agile como soporte a dicha metodología.

Abstract (in English, 250 words or less):

In the business world it is being detected that the use of new technologies is essential for optimizing business processes and increase efficiency. That's the reason why most of the companies need to have a proper *software* that covers all business needs. This *software* can be an enterprise solution for third, a tailored solution or a development of the company with own resources. Whether if it is one way or another, it is essential that the *software* life cycle that we are going to create would be optimum in its development, efficient in its implementation, coherent with what is needed, changeable and of course, having a quality delivery.

Nowadays, many companies that try to create applications have certain shortcomings due to they don't use any type of methodology when developing, having certain drawbacks that would be corrected with a working methodology.

In this final work we will study how to implement a development methodology to solve common faults of application development that often make the companies; Therefore we are going to study the implementation of Scrum. In order to have a successful implementation, we need to use a tool to manage the life cycle of *software* development and for this we will use a *software* called JIRA Agile, as support for this methodology.

Palabras clave (entre 4 y 8):

Scrum, JIRA, metodología de desarrollo, desarrollo para el usuario, Agile, entregables agiles, calidad de *software*

Índice

1. INTRODUCCIÓN

1.1. Contexto y justificación del trabajo

El hecho relevante de que hoy en día la tecnología avance a una velocidad muy considerable hace que dentro del proceso de desarrollo del *software* se hayan creado una serie de necesidades que hoy son inevitables en todo tipo de proyectos informáticos: las entregas rápidas, la facilidad a la hora de realización de cambios o implementar nuevas funcionalidades y, con esto, también se asume que hay que ofrecer una garantía de calidad de lo que se está fabricando. Para cumplir con estas necesidades, existe la obligación de realizar una serie de cambios en las metodologías de trabajo, utilizando las llamadas metodologías ágiles, en nuestro caso, específicamente, Scrum.

Las metodologías ágiles plantean mejorar la eficiencia en la producción y la calidad de los productos finales, y pueden tener una capacidad de respuesta rápida a los cambios en las aplicaciones y en sus definiciones. Además, pueden brindar una mayor satisfacción al cliente, a través de entregas precoces y la retroalimentación continua durante todo el proceso de desarrollo del *software*.

La implementación de la metodología Scrum aportará una serie de beneficios claros y objetivos, ya que permitirá una mayor flexibilidad que las metodologías tradicionales (en cascada e interactivas), debido a que estas son menos capaces de ajustarse a los cambios de necesidades de los clientes, del mercado y de los nuevos desafíos tecnológicos.

En la actualidad se pueden encontrar dos escenarios: uno de ellos es donde las empresas realizan la gestión del *software* sin una clara metodología, llevando las versiones de forma desordenada, sin conocer exactamente los costes de la realización de las aplicaciones y sin tener claros los tiempos de entrega, además de la dificultad a la hora de evolucionar el entorno. El segundo caso es el de aquellas empresas que utilizan metodologías tradicionales, donde se encuentran varios inconvenientes.

Algunos de los inconvenientes que se pueden encontrar en las metodologías tradicionales son que, si dentro de alguna etapa de desarrollo se observa algún problema, requiere mucho coste volver a la etapa anterior, y el usuario no ve el producto hasta la entrega final, por lo que no va a ir validando lo que realmente se está haciendo, qué es lo que se necesita. Para finalizar, también se ve que en aquellos negocios cambiantes, se puede encontrar que los requisitos iniciales cambian totalmente en el momento en el que se pase a la etapa de la construcción.

Las aportaciones que va ofrecer Scrum a la hora de ser implantada en una empresa son claras y, sobre todo, tangibles desde el primer día, por eso se describen aquí algunas de ellas:

1. *Reducción de tiempo de las entregas:* El hacer que nuestro producto vaya creciendo incrementalmente va a permitir decidir cuándo es el momento de que nuestro proyecto vea la luz y, por tanto, puedan empezar a utilizarlo los usuarios.

2. *Realización de una definición progresiva y rápida reacción al cambio:* Dado que Scrum va a ayudar a priorizar los siguientes puntos a desarrollar en función del resultado obtenido en cada una de las entregas, esto va permitir actuar en el caso de que los requisitos se descarten o sean sustituidos por otros más rentables para el producto en cualquier momento del desarrollo.

3. *Detección y análisis inmediato de los riesgos:* Aquellas tareas que pueden producir un riesgo de viabilidad del proyecto serán las primeras a afrontar. Por lo que, gracias a esto, se minimizarán las pérdidas económicas si aparece un imprevisto que impida llevar a cabo el producto tal y como había sido definido, incluso de forma que impida la cancelación del proyecto.

Los resultados que se desea obtener en cualquier compañía una vez que se aplique Scrum son los siguientes:

- Aumento de productividad.
- Aumento de la moral del equipo de trabajo.
- Aumento de la adaptabilidad de los productos.
- Aumento de la responsabilidad de cada uno de los miembros del equipo.
- Aumento de la colaboración y cooperación.

1.2. Objetivos del trabajo

Los objetivos del trabajo fin de carrera son permitir a cualquier organización ser más eficiente en sus procesos de desarrollo mediante Scrum, combinado con la utilización de la herramienta JIRA Agile.

Las metas de este trabajo son permitir que, paso a paso, cualquier departamento IT pueda llevar a cabo la implantación con éxito de la metodología, que no tenga dudas a la hora de implantar el método y la corrección de posibles errores que se pueden dar cuando se pone en marcha Scrum.

Por otro lado, se prestará ayuda para poner en marcha la aplicación JIRA para dar soporte a la metodología, explicando su uso, explicando cómo son los circuitos y fases de desarrollo, y aclarando cómo usar de modo eficiente los diferentes indicadores que existen en la herramienta, para conocer y gestionar la evolución de la construcción de cualquier proyecto de *software*.

Para poder hacer una medición y evaluar que este trabajo realmente cumple los objetivos expuestos, y para poder demostrar que gracias a este trabajo la empresa que aplica la metodología, tal y como se expone en este proyecto, mejora en su proceso productivo, se utilizará una serie de indicadores que mida el aumento de la eficiencia, la calidad y el ahorro de costes en los desarrollos de *software*.

Los indicadores claves a medir son:

1. *Disminución de los* bugs *de* software: La empresa, aplicando la nueva metodología, conseguirá que los errores de *software* disminuyan. Esto se conseguirá, primero, porque habrá una mejor planificación del proyecto y se dedicará más tiempo a aquellas tareas más complejas. El equipo estará más motivado y realizará sus tareas con una mayor calidad. Al implicarse todo el equipo en la construcción del *software*, los miembros más expertos podrán ayudar a los de menos experiencia para resolver las tareas de forma más eficaz.

2. *Desviaciones en las entregas:* Se conseguirá disminuir las desviaciones de tiempo en las entregas de cualquier *software*. Esto lo será posible ya que, al realizar interacciones más cortas, se podrá saber claramente donde están los problemas de las entregas y donde son necesarios mayores esfuerzos. También al ir haciendo entregables progresivamente se irán verificando continuamente que lo que se ejecuta es lo que se quiere hacer.

3. *Aumento de la satisfacción del cliente:* Aumentará el número de clientes que están satisfechos con las entregas realizadas y con la gestión del proyecto. Esto se conseguirá por los diferentes entregables que se realicen, por la verificación continua de los requisitos y al realizar el tratamiento del proyecto mediante una visión modo usuario en lugar de hacer una visión técnica.

4. *Ahorro de costes:* Se averiguará que también se ahorran costes, ya que se dedica el tiempo adecuado a desarrollar los productos y al utilizar los recursos justos para desarrollar cada proyecto. Con esta metodología se tendrá un ahorro de coste considerable en cada proyecto.

5. *Mejora de la vida laboral de los equipos:* Con la metodología se conseguirá un equipo más motivado, más involucrado, lo que reducirá el índice de rotación del equipo. Es decir, Scrum le sienta bien al equipo, funciona mejor y es más productivo, pero el cambio cultural es tan importante que surgen problemas con otras áreas que no han comenzado el cambio (de todos es sabido que muchos se oponen al cambio).

6. *Éxito en los proyectos:* Con la utilización de Scrum se dará un aumento de éxito en la finalización de los proyectos.

7. *Aumento de la productividad:* Según estudios realizados a varios directores de productos, se estima que con Scrum se aumenta la productividad media en un 36 %.

A continuación se detalla de qué manera Scrum permite conseguir cada uno de los beneficios anteriores (ProyectosAgiles, 2015).

Objetivos de Scrum	Cómo se consiguen
Gestión regular de las expectativas del cliente El cliente establece sus expectativas indicando el valor que le aporta cada requisito del proyecto y cuando espera que esté completado. El cliente comprueba de manera regular si se van cumpliendo sus expectativas, da feedback, ya desde el inicio del proyecto puede tomar decisiones informadas a partir de resultados objetivos y dirige estos resultados del proyecto, iteración a iteración, hacia su meta. Se ahorra esfuerzo y tiempo al evitar hipótesis.	Lista de requisitos priorizada El cliente crea y gestiona la lista de requisitos del producto o proyecto, donde quedan reflejadas sus expectativas a nivel de requisitos, valor, coste y entregas.
	Demostración de los resultados de proyecto en cada iteración Al final de cada iteración el equipo demuestra al cliente los requisitos que ha conseguido completar. Tras una inspección del resultado real del proyecto hasta ese momento, y considerando el esfuerzo que ha sido necesario para realizarlo, el cliente solicita los cambios que necesita y replanifica el proyecto.
Resultados anticipados El cliente puede empezar a utilizar los resultados más importantes del proyecto antes de que esté finalizado por completo. Siguiendo la ley de Pareto (el 20% del esfuerzo proporciona el 80% del valor), el cliente puede empezar antes a recuperar su inversión (y/o autofinanciarse) comenzando a utilizar un producto al que sólo le faltan características poco relevantes, puede sacar al mercado un producto antes que su competidor, puede hacer frente a urgencias o nuevas peticiones de clientes, etc.	Priorización de requisitos por valor y coste Al inicio de cada iteración el cliente prioriza la lista de requisitos del producto o proyecto en función del valor que le aportan, su coste de desarrollo y los riesgos del proyecto, cambiando los requisitos previstos para reaccionar a cambios de contexto en el proyecto. El progreso del proyecto se mide en función de los requisitos que el equipo completa en cada iteración.
Flexibilidad y adaptación De manera regular el cliente redirige el proyecto en función de sus nuevas prioridades, de los cambios en el mercado, de los requisitos completados que le permiten entender mejor el producto, de la velocidad real de desarrollo, etc. Al final de cada iteración el cliente puede aprovechar la parte de producto completada hasta ese momento para hacer pruebas de concepto con usuarios o consumidores y tomar decisiones en función del resultado obtenido.	Replanificación en el inicio de cada iteración Se asume que los cambios son parte natural del proyecto. Toda iteración comienza con una replanificación del proyecto. Esta replanificación no es traumática puesto que Scrum minimiza el número de objetivos/requisitos en que el equipo trabaja a los que caben en una iteración. Todavía no se ha hecho ningún esfuerzo en desarrollar los requisitos de las siguientes iteraciones. El hecho los requisitos se completen en función del valor que aportan al cliente minimiza la probabilidad de que se produzcan grandes cambios en el transcurso del proyecto.
Retorno de inversión (ROI) De manera regular, el cliente maximiza el ROI del proyecto. Cuando el beneficio pendiente de obtener es menor que el coste de desarrollo, el cliente puede finalizar el proyecto.	Priorización de requisitos por valor Cada iteración el cliente dispone de unos requisitos completados y replanifica el proyecto en función del valor que le aportan los requisitos pendientes respecto del coste de desarrollo que tienen.
Mitigación de riesgos Desde la primera iteración el equipo tiene que gestionar los problemas que pueden aparecer en una entrega del proyecto. Al hacer patentes estos riesgos, es posible iniciar su mitigación de manera anticipada. "Si hay que equivocarse o fallar, mejor hacelo lo antes posible". El feedback temprano permite ahorrar esfuerzo y tiempo en errores técnicos. La cantidad de riesgo a que se enfrenta el equipo está limitada a los requisitos que se puede desarrollar en una iteración. La complejidad y riesgos del proyecto se dividen de manera natural en iteraciones.	Desarrollo iterativo e incremental Un requisito se debe completar en una iteración. El equipo debe realizar todas las tareas necesarias para completarlo y que esté preparado para ser entregado al cliente con el esfuerzo mínimo necesario. De esta manera no se deja para el final del proyecto ninguna actividad arriesgada relacionada con la entrega de requisitos.
Productividad y calidad De manera regular el equipo va mejorando y simplificando su forma de trabajar. Los miembros del equipo sincronizan su trabajo diariamente y se ayudan a resolver los problemas que pueden impedir conseguir el objetivo de la iteración. La comunicación y la adaptación a las diferentes necesidades entre los miembros del equipo son máximas (se van ajustando iteración a iteración), de manera que no se realizan tareas innecesarias y se evitan ineficiencias. Las personas trabajan más enfocadas y de manera más eficiente cuando hay una fecha límite a corto plazo para entregar un resultado al que se han comprometido. La consciencia de esta limitación temporal favorece la priorización de las tareas y fuerza la toma de decisiones. Las iteraciones (Sprints) son regulares y de un mes para facilitar la sincronización sistemática con otros equipos, con el resto de la empresa y con el cliente. El equipo minimiza su dependencia de personas externas para poder avanzar (depender de la disponibilidad de otros puede parar tareas). La estimación de esfuerzo y la optimización de tareas para completar un requisito es mejor si la realizan las personas que van a desarrollar el requisito, dadas sus diferentes especializaciones, experiencias y puntos de vista. Asímismo, con iteraciones cortas la precisión de las estimaciones aumenta. Las personas trabajan de manera más eficiente y con más calidad cuando ellas mismas se han comprometido a entregar un resultado en un momento determinado y deciden cómo hacerlo, no cuando se les ha asignado una tarea e indicado el tiempo necesario para realizarla. El equipo se evita caminar mucho tiempo por un camino equivocado que le obligue a realizar un gran esfuerzo para llegar al objetivo esperado Se asegura la calidad del producto de manera sistemática y objetiva, a nivel de satisfacción del cliente, requisitos listos para ser utilizados y calidad interna del producto.	Mejora continua Cada iteración el equipo realiza una retrospectiva para analizar su manera de trabajar e identificar los obstáculos que le impiden avanzar
	Comunicación diaria del equipo Todo miembro del equipo conoce cómo el trabajo de los otros miembros impacta en el suyo y cuáles son las necesidades de los otros.
	TimeBoxing Cada actividad de Scrum siempre tiene la misma duración (1 mes, 4 horas, etc.), con lo que las personas aprenden lo que pueden conseguir en este tiempo, cómo organizarse, priorizar tareas y tomar decisiones.
	Equipo multidisciplinar El equipo está formado por todas las personas con las especialidades necesarias para llevar a cabo el proyecto.
	Estimación de esfuerzo conjunta En el inicio de la iteración los miembros del equipo estiman de manera conjunta el esfuerzo necesario para completar requisitos y sus tareas.
	Compromiso del equipo En el inicio de cada iteración el equipo selecciona los requisitos que se compromete a completar y entregar al final de la iteración (responabilidad). El propio equipo se organiza (autoridad) identificando las tareas necesarias, su esfuerzo y autoasignandose cada miembro las tareas que se compromete a realizar.
	Demostración de resultados preparados para ser utilizados y velocidad sostenida Por un lado, al final de cada iteración el equipo demuestra al cliente los requisitos que ha conseguido completar, de manera que están completamente operativos. Por otro lado, para tener una velocidad de desarrollo sostenida, el equipo necesita desarrollar cada incremento de producto sin tener que revisitar aspectos mal resueltos en iteraciones anteriores.

Objetivos de Scrum	Cómo se consiguen
Alineamiento entre cliente y equipo Los resultados y esfuerzos del proyecto se miden en forma de objetivos y requisitos entregados al negocio. Todos los participantes en el proyecto conocen cuál es el objetivo a conseguir. El producto se enriquece con las aportaciones de todos.	Cliente y equipo trabajando "en equipo" Cada iteración el equipo y el cliente trabajan juntos en la creación de los requisitos del proyecto (en la estimación de la lista priorizada de requisitos del proyecto), en darles detalle (en la y en el análisis del resultado obtenido (en la reunión de planificación de la iteración) demostración de los requisitos completados).
Equipo motivado Las personas están más motivadas cuando pueden usar su creatividad para resolver problemas y cuando pueden decidir organizar su trabajo. Las personas se sienten más satisfechas cuando pueden mostrar los logros que consiguen.	Equipo autogestionado El equipo es quien se compromete a completar unos requisitos determinados en una iteración y quien mejor sabe cómo desarrollarlos. Por ello es el equipo quien se autoorganiza y quien planifica cómo trabajará en la iteración.
	Demostración Cada iteración el equipo muestra al cliente los resultados que consigue. No está meses trabajando sin poder exhibir su obra.

1.3. Enfoque y método seguido

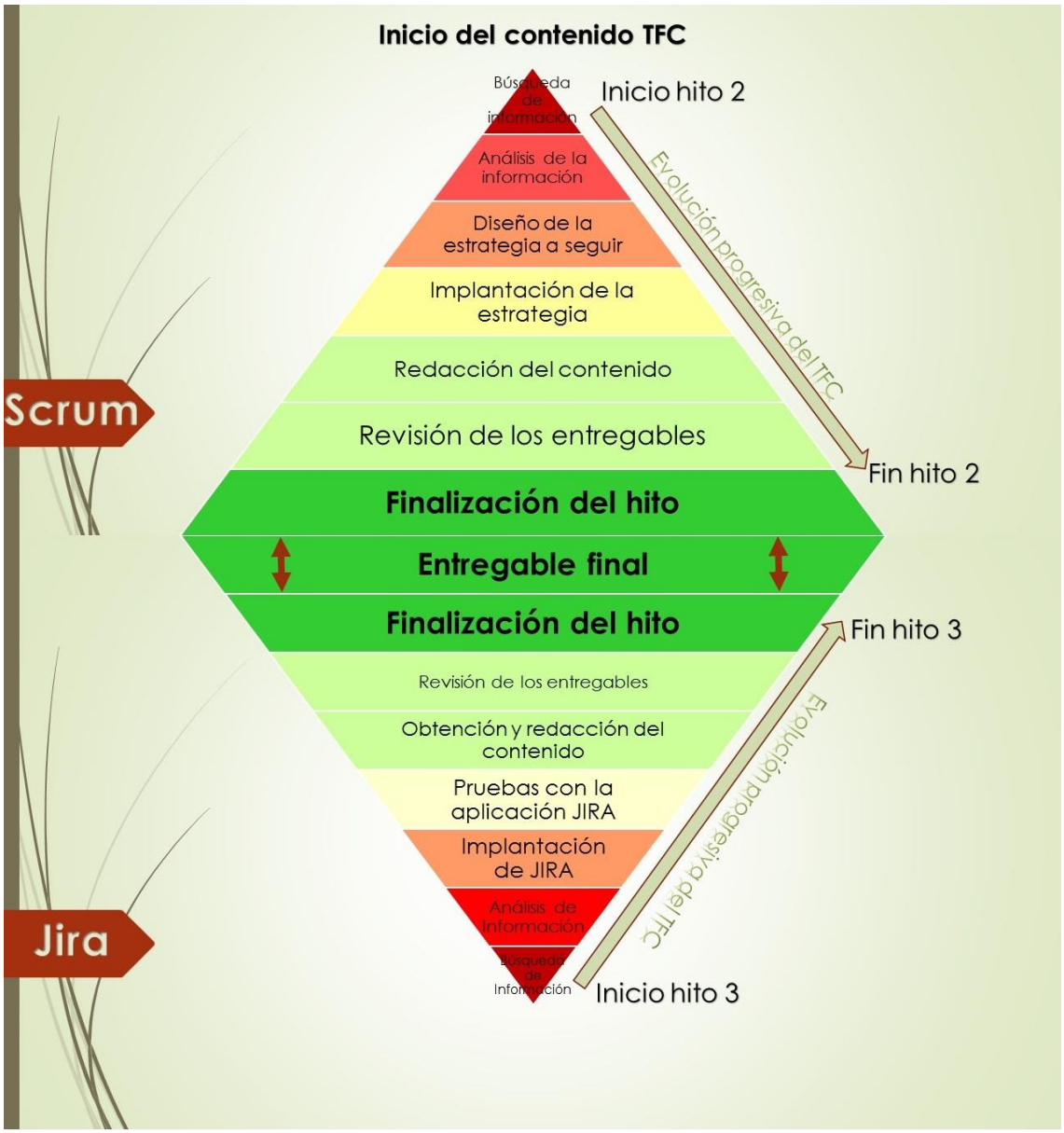
Dado que la temática elegida es una bastante teórica, se podría haber realizado el trabajo mediante la escritura de un documento con todo tipo de explicación, indicando en qué consiste la metodología Scrum y dando unas cuantas pinceladas de teoría. Sin embargo, este no es el objetivo del proyecto, ya que el objetivo es que el presente trabajo tenga una aplicación totalmente práctica y funcional de la metodología, por ello se intentará dar un enfoque diferente al enfoque teórico, con una visión de implantador del método. En el documento se hará el análisis desde el punto de vista de la persona encargada de hacer la implantación y la ejecución del proceso de cambio.

Todo proyecto final de carrera tiene unos fines ligados a la obtención de un entregable que es necesario generar una vez que se hayan realizado diversas actividades. Algunas de estas actividades pueden agruparse en hitos porque globalmente contribuyen a obtener entregables intermedios; todo esto será necesario para continuar hacia la entrega final.

El diseño de este plan ayuda a ir aumentando el valor del trabajo en cada una de las etapas, de tal forma que, con este aumento, se consiga finalmente disponer de un trabajo de calidad.

La realización del trabajo consiste en tres grandes hitos. Dos de ellos están basados en el contenido de la memoria que son: uno, «Relacionado con la metodología Scrum» y otro, «Relacionado con la herramienta JIRA». A estos dos hitos hay que sumarle el hito final, que consiste en la entrega final del trabajo, que suma todo lo generado en los hitos anteriores, y añade la realización de una presentación del trabajo.

Gráfico de la explicación de la metodología del trabajo



1.4. Planificación del trabajo

Para poder llevar a buen puerto la entrega de este trabajo se han requerido diferentes tipos de recursos, muchos de ellos encontrados en internet (artículos, documentos, noticias) y otros recopilados de la documentación oficial de la certificación Scrum Master.

También ha sido necesario adquirir una licencia de la aplicación JIRA, para mostrar su implantación y funcionamiento. La licencia ha sido adquirida en modo *demo* y no ha supuesto ningún coste económico para la realización del trabajo.

La táctica llevada a cabo en el presente trabajo es la siguiente:

En una primera fase, se realizó un plan de la gestión del proyecto donde se planificaron todas las tareas y se establecieron los mecanismos de control del mismo. Esta primera fase fue concluida con *Hito 1 del TFC, cuya fecha de entrega fue el 14 de octubre del 2015.*

En una segunda fase se realizó una investigación profunda sobre la metodología Scrum, utilizando diferentes fuentes de documentación que tuvieran relación con la temática del proyecto.

En una tercera fase se realizó el desarrollo del contenido relacionado con la metodología Scrum, incluyendo la redacción y la finalización del entregable *Hito 2 del TFC, cuya fecha de entrega fue el 18 de noviembre del 2015.*

En una cuarta fase se realizó una investigación profunda sobre la herramienta JIRA, utilizando la documentación oficial del fabricante, además de alguna documentación externa que tuviera relación con el *software* mencionado.

En la quinta fase se instaló la herramienta JIRA, se configuró para poder ser utilizada con la metodología Scrum y se fue recopilando la información de todo el proceso para la posterior redacción de la documentación.

En la sexta fase se probó la herramienta, se analizaron todas las posibilidades, se creó un proyecto de ejemplo y se investigaron todas las métricas existentes en JIRA.

En la séptima fase se realizó el desarrollo del contenido relacionado con la aplicación JIRA, incluyendo la redacción y finalizando el entregable *Hito 3 del TFC, cuya fecha de entrega fue el 16 de diciembre del 2015.*

En la última fase se unificaron los contenidos para terminar la memoria final del proyecto, se revisó toda la documentación, se realizó la presentación y la entrega de la misma. Esta última fase coincide con el último hito denominado *«Entrega final», cuya fecha de entrega fue el 13 de enero del 2016.*

Tras la entrega final, se ha dedicado un periodo de tiempo para preparar la defensa del proyecto ante el tribunal académico.

Una vez definidas cada una de las fases del proyecto, se realiza una definición detallada de tareas con un coste temporal de cada una de ellas.

Esta definición ha permitido hacer una planificación y, para cada una de las tareas, establecer una fecha aproximada de inicio y una fecha aproximada de fin. Además, todas las tareas han sido controladas a lo largo de todo el proyecto, realizando un seguimiento continuo del proyecto a través de entregables o hitos, cuyas fechas de entrega han coincidido con las diferentes fechas de finalización de las PECs.

El trabajo se planificó para tener una duración aproximada de cuatro meses.

A continuación se definen los diferentes roles que han participado en el proyecto y también el Gantt de planificación.

1.5. Roles, perfiles y responsabilidades del *proyecto*

El proyecto fin de carrera ha sido realizado por un solo recurso, pero dicho recurso ha tenido diferentes tipos de roles dependiendo de la tarea que en un momento dado se estaba realizando. A continuación se definen los diferentes tipos de roles que han intervenido en cada una de las tareas del trabajo:

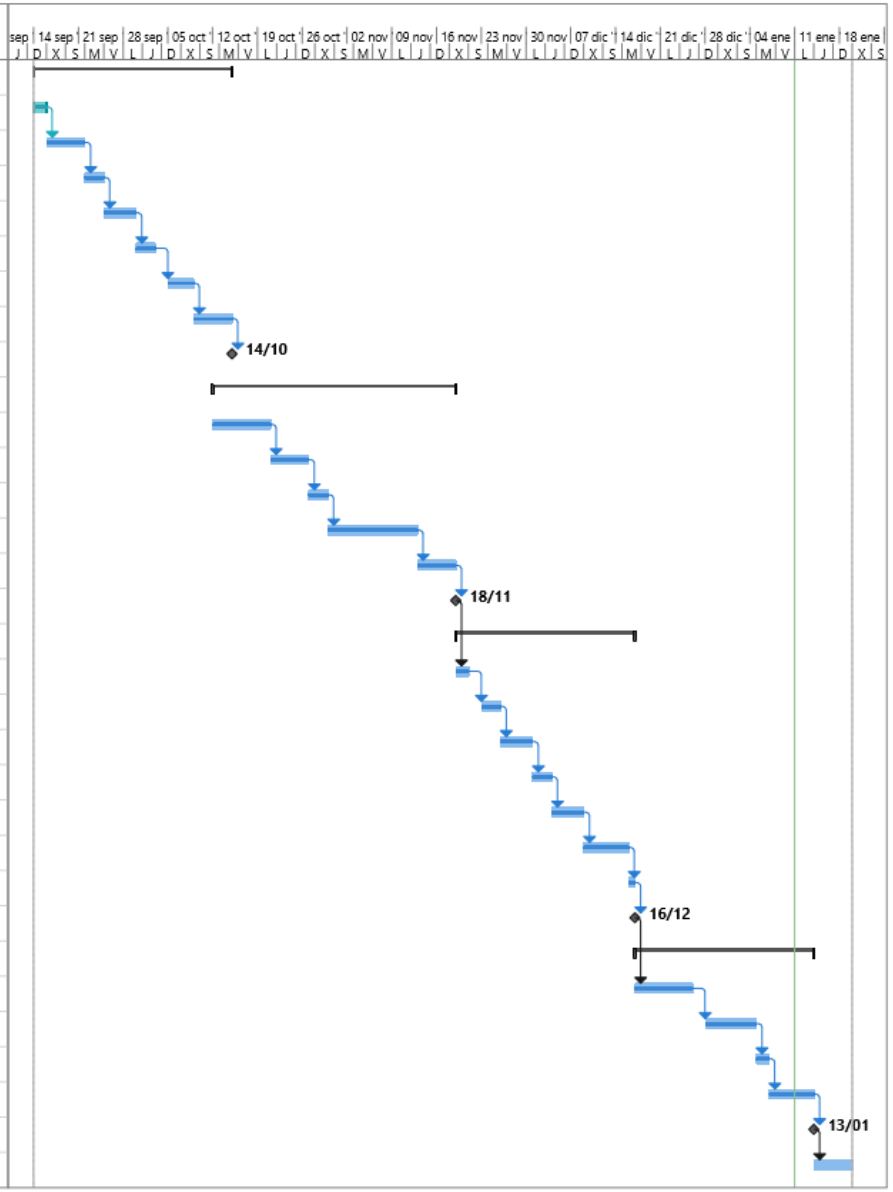
Director de Proyecto (PM): Ha ejercido las funciones de dirección, supervisión y control de las tareas programadas en el proyecto. Además, ha realizado la verificación de los productos entregados. También ha realizado la presentación final y la defensa del proyecto.

Técnico de Sistemas (TS): Ha realizado todas las tareas técnicas de instalación, testeo del *software* JIRA y la documentación sobre cómo ha de utilizarse la herramienta.

Director de Desarrollo (DD): Ha realizado toda la recopilación de la información relacionada con el proyecto, analiza todos los procesos de la metodología y la estrategia de cambio.

Diagrama de Gantt del proyecto y planificación de tareas

Id	Modo de tarea	Nombre de tarea	Rol	Horas Tarea	Duración	Comienzo	Fin
1		**Fase 1**			**23 días**	**lun 14/09/15**	**mié 14/10/15**
2		Lectura Plan de Estudios y Organización previa TFC	PM	3h	2 días	lun 14/09/15	mar 15/09/15
3		Lectura documentación TFC	PM	8h	4 días	mié 16/09/15	lun 21/09/15
4		Evaluar los posibles trabajos a realizar	PM	8h	3 días	mar 22/09/15	jue 24/09/15
5		Elección de Tematica	PM	10h	3 días	vie 25/09/15	mar 29/09/15
6		Investigación previa de la temática	PM	16h	3 días	mié 30/09/15	vie 02/10/15
7		Introducción del TFC	PM	16h	4 días	lun 05/10/15	jue 08/10/15
8		Planificación y definición de metodologia	PM	16h	4 días	vie 09/10/15	mié 14/10/15
9		**Entregable 1 - PEC 1**	PM		***0 días***	mié 14/10/15	mié 14/10/15
10		**Fase 2**			**28 días**	**lun 12/10/15**	**mié 18/11/15**
11		Busqueda Información Scrum	DD	24h	7 días	lun 12/10/15	mar 20/10/15
12		Analisis de Información Obtenida	DD	16h	4 días	mié 21/10/15	lun 26/10/15
13		Extracción de la Información a incluir	DD	8h	3 días	mar 27/10/15	jue 29/10/15
14		Redacción de la documentación a incluir	DD	40h	10 días	vie 30/10/15	jue 12/11/15
15		Revisión del entregable	PM	16h	4 días	vie 13/11/15	mié 18/11/15
16		**Entregable 2 - PEC 2**	PM		0 días	mié 18/11/15	mié 18/11/15
17		**Fase 3**			**20 días**	**jue 19/11/15**	**mié 16/12/15**
18		Busqueda Información JIRA	TS	4h	2 días	jue 19/11/15	vie 20/11/15
19		Analisis de la Información	TS	8h	3 días	lun 23/11/15	mié 25/11/15
20		Implantación y Configuración de JIRA	TS	12h	3 días	jue 26/11/15	lun 30/11/15
21		Pruebas con JIRA	TS	12h	3 días	mar 01/12/15	jue 03/12/15
22		Extracción de la información a Incluir	TS	8h	3 días	vie 04/12/15	mar 08/12/15
23		Redacción de la documentación a incluir	TS	24h	5 días	mié 09/12/15	mar 15/12/15
24		Revisión del entregable	PM	4h	1 día	mié 16/12/15	mié 16/12/15
25		**Entregable 3 - PEC 3**	PM		0 días	mié 16/12/15	mié 16/12/15
26		**Fase 4**			**20 días**	**jue 17/12/15**	**mié 13/01/16**
27		Unificación de contenidos, Glosarios, Bibliografias y Anexos	PM	32h	7 días	jue 17/12/15	vie 25/12/15
28		Implantación de mejoras planteadas en entregables pasados	PM	32h	6 días	lun 28/12/15	lun 04/01/16
29		Definición de conclusiones del TFC	PM	6h	2 días	mar 05/01/16	mié 06/01/16
30		Creación de presentación	PM	16h	5 días	jue 07/01/16	mié 13/01/16
31		**Entrega Final**	PM		0 días	mié 13/01/16	mié 13/01/16
32		Preparación de la defensa del Proyecto	PM	16h	4 días	jue 14/01/16	mar 19/01/16



1.6. Estimación de recursos del proyecto

Recursos humanos: Tal y como se ha comentado anteriormente, el proyecto ha sido realizado por una sola persona, pero que ocupaba en cada momento un rol diferente. No obstante, a nivel de coste y asumiendo que dicho trabajo podría haber sido realizado por varias personas, se estiman los costes de las horas según el rol del responsable de cada una de las tareas:

- Rol PM: 50 € por hora
- Rol DD: 40 € por hora
- Rol TS: 30 € por hora

Equipamiento y herramientas: Para la realización del proyecto se ha utilizado el *software* Microsoft Office en modo distribución gratuita. Para la utilización de la herramienta JIRA se ha adquirido una licencia de pruebas de 14 días; todos los módulos adicionales necesarios para utilizar JIRA se han utilizado en modo 30 días de prueba.

1.7. Calculo de los costes

En el diagrama de Gantt presentado anteriormente se determinó el número de horas necesarias para ejecutar cada una de las tareas y también se anexó el rol responsable de cada una de ellas; de esta forma, fue posible estimar el coste del proyecto, teniendo en cuenta las horas dedicadas y el coste hora de cada uno de los roles que intervinieron en el proyecto.

Perfil	Horas dedicadas	Coste hora	Coste total
Director de proyecto (PM)	199	50,00 €	9.950,00 €
Director de Desarrollo (DD)	88	40,00 €	3.520,00 €
Técnico de sistemas (TS)	68	30,00 €	2.040,00 €
		Total	15.510,00 €

Equipamiento y Licencias	10 €
Total Horas RR. HH.	15.510,00 €
Total proyecto	15.520 €

1.8. Breve sumario del producto obtenido

El producto que se persigue con este trabajo final de carrera es la definición de una guía que cualquier empresa pueda utilizar para poder hacer una implementación de la metodología Scrum.

Se ha definido el cambio de paradigma que tienen que hacer las organizaciones para cambiar su método de desarrollo, cómo la visión técnica pasará a un segundo plano, cambiando todo el proceso a la visión de usuario, donde este adquirirá mucho más protagonismo.

Por último, y para que la implantación de la metodología sea totalmente exitosa, será necesario el apoyo en una herramienta que controle todo el ciclo de vida del *software* y, para ello, se utilizará JIRA. De esta manera, todo el proceso estará más controlado, y será posible sacar métricas muy válidas a la hora de gestionar cualquier proyecto *software*.

Algunos de los contenidos incluidos en este trabajo fin de carrera son los siguientes:

- Definición y explicación de la metodología Scrum.
- Cambios en la empresa para trabajar con Scrum.
- Posibles errores al implantar Scrum.
- Utilización de JIRA.
- Cómo utilizar JIRA aplicando Scrum.
- Métricas a utilizar para el seguimiento de cualquier proyecto.

2. SCRUM, PUESTA EN MARCHA DEL MÉTODO

Para empezar a comprender en qué consiste la metodología Scrum hay que cambiar la forma de pensamiento, ya que se va a proceder a un cambio total de la mentalidad actual y para ello la forma nueva de trabajar se va a basar en un manifiesto ágil firmado por Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert Cecil Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland y Dave Thomas, que expone que (Wikipedia, 2015):

Con esta metodología se ponen al descubierto mejores métodos para desarrollar *software*, haciéndolo y ayudando a otros a que lo hagan.

Con este trabajo se aprenderá a valorar aspectos que hasta ahora ni si quiera se habían planteado, se aprenderá a valorar:

- *A los individuos y su interacción, por encima de los procesos y las herramientas.*
- *El* software *que funciona, por encima de la documentación exhaustiva.*
- *La colaboración con el cliente, por encima de la negociación contractual.*
- *La respuesta al cambio, por encima del seguimiento de un plan.*

2.1. ¿Qué es la agilidad?

(ScrumManager, 2014a). El entorno de trabajo de las empresas del conocimiento se parece muy poco al que originó la gestión de proyectos predictivos. Ahora se necesitan estrategias para el lanzamiento de productos orientadas a la entrega temprana de resultados tangibles, y a la respuesta ágil y flexible, necesaria para trabajar en mercados de evolución rápida.

Ahora se construye el producto mientras se modifican y aparecen nuevos requisitos. El cliente parte de una visión medianamente clara, pero el nivel de innovación que requiere, y la velocidad a la que se mueve su sector de negocio, no le permiten predecir con detalle cómo será el resultado final. Quizá ya no hay «productos finales», sino productos en continua evolución y mejora.

La gestión de proyectos ágil no se formula sobre la necesidad de anticipación, sino sobre la de adaptación continua.

¿La gestión de proyectos predictiva es la única posible? ¿Los criterios para determinar el éxito son siempre el cumplimiento de

fechas y costos? ¿Puede haber proyectos cuya gestión no busque realizar un trabajo previamente planificado, con un presupuesto y en un tiempo previamente calculado?

Hoy hay directores de producto que no necesitan conocer cuáles van a ser las 200 funcionalidades que tendrá el producto final, ni si este estará terminado en 12 o en 16 meses.

Hay clientes que necesitan disponer de una primera versión con funcionalidades básicas en cuestión de semanas, y no un producto completo dentro de uno o dos años. Clientes cuyo interés es poner en el mercado rápidamente un concepto nuevo, y desarrollar de forma continua su valor.
Hay proyectos que no necesitan gestionar el seguimiento de un plan, y que fracasan por haber empleado un modelo de gestión inapropiado.

La mayoría de los fiascos se producen por aplicar ingeniería secuencial y gestión predictiva tanto en el proceso de adquisición (requisitos, contratación, seguimiento y entrega) como en la gestión del proyecto, en productos que no necesitan tanto garantías de previsibilidad en la ejecución, como respuesta rápida y flexibilidad para funcionar en entornos de negocio que cambian y evolucionan rápidamente.

(Wikipedia, 2014). Para poner en marcha el método vamos a tener en cuenta 12 reglas o principios que siempre tenemos que tener como base:

1. Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.
2. Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
3. Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses como mucho, con preferencia en los periodos breves.
4. Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
5. Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.
6. La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.
7. El software que funciona es la principal medida del progreso.
8. Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica enaltece la agilidad.

La simplicidad como arte de maximizar la cantidad de trabajo que se hace, es esencial.
Las mejores arquitecturas, requisitos y diseños emergen de equipos que se autoorganizan.
En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

2.2 Introducción al modelo.

(ScrumManager, 2014b). El marco técnico de Scrum, por su sencillez, resulta apropiado para equipos y organizaciones que quieren comenzar a «avanzar en Scrum».

Está formado por un conjunto de prácticas y reglas que resultan válidas para dar respuesta a los siguientes principios de desarrollo ágil:

- Gestión evolutiva del avance, en lugar de la tradicional o predictiva.
- Trabajar basando la calidad del resultado en el conocimiento tácito de las personas, más que en el explícito de los procesos y la tecnología empleada.
- Estrategia de desarrollo incremental a través de iteraciones (sprints) y revisiones.
- Seguir los pasos del desarrollo ágil: desde el concepto o visión general de la necesidad del cliente, construcción del producto de forma incremental a través de iteraciones breves que comprenden fases de especulación —exploración y revisión—. Estas iteraciones (en Scrum llamadas sprints) se repiten de forma continua hasta que el cliente da por cerrada la evolución del producto.

Se comienza con la visión general de lo que se desea obtener, y a partir de ella se especifica y da detalle a las partes de mayor prioridad, y que se desean tener cuanto antes.

Cada ciclo de desarrollo o iteración (sprint) finaliza con la entrega de una parte operativa del producto (incremento). La duración de cada sprint puede ser desde una, hasta seis semanas, aunque se recomienda que no excedan de un mes.

En Scrum, el equipo monitoriza la evolución de cada sprint en reuniones breves diarias donde se revisa en conjunto el trabajo realizado por cada miembro el día anterior, y el previsto para el día en curso. Esta reunión diaria es de tiempo prefijado de 5 a 15 minutos máximo, se realiza de pie junto a un tablero o pizarra con información de las tareas del sprint, y el trabajo pendiente en cada una. Esta reunión se denomina «reunión de pie» o «Scrum diario» y si se emplea la terminología inglesa: «stand-up meeting», también: «daily Scrum» o «morning rollcall».

2.2. Cambiar la forma de pensar

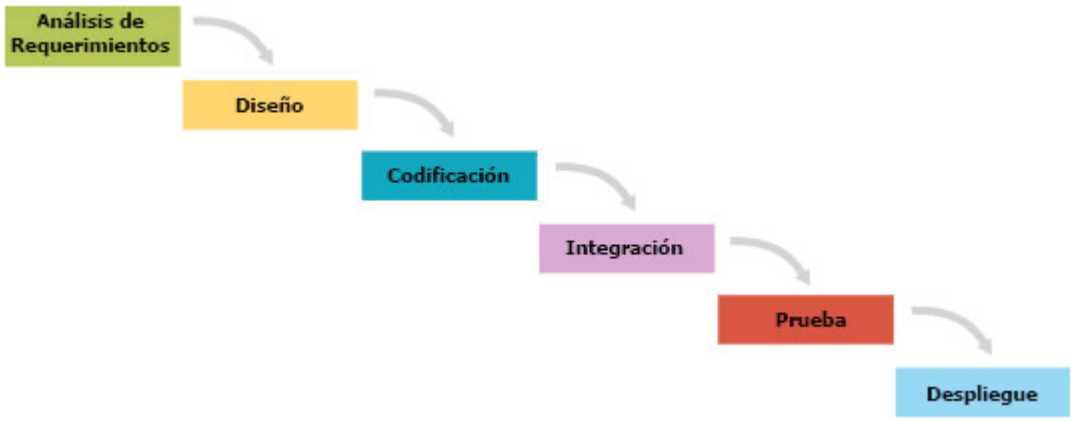
Hasta el día de hoy al comenzar el desarrollo de una aplicación, se apreciaba a nivel técnico todo lo que había que hacer para, finalmente, llegar al resultado. Como norma general se avanzaba por cada una de las fases del proyecto, pero nunca se mezclaban unas fases con otras, de tal forma que, si se realizaba un diseño, este diseño nunca se validaba a nivel de usuario, porque solo existía el diseño técnico del sistema, sin que el usuario supiera finalmente lo que se le iba a entregar.

La cantidad de requisitos funcionales de la aplicación eran la interpretación técnica que hacía un ingeniero de lo que pedía el usuario, y siempre se evaluaba técnicamente cómo hacerlo de la forma más sencilla sin tener en cuenta la usabilidad, sin tener en cuenta la realidad del usuario. Tras el análisis del ingeniero se desarrollaba todo el *software*, fase por fase, hasta llegar a su fin. Si dentro del contexto había un cambio de requisito, era complicada su adaptación, se trastocaban todos los tiempos del proyecto y el coste de las modificaciones era muy alto, por lo que se comprometían siempre la finalización del proyecto y los plazos de entrega.

Cada fase del proyecto era un todo, y hasta no finalizar completamente una fase no se empezaba con la siguiente, pero siempre sin poder enseñar previamente al usuario lo que se estaba construyendo porque no era funcional lo que se hacía, ya que, sin terminar todas las fases del proyecto, no se tenía nada.

Las fases de un proyecto tradicional se basaban en el siguiente esquema:

Esquema de las fases de un proyecto tradicional



Si en alguna de las fases había algún error de concepto, el hecho de volver a una fase anterior era muy complicado, ya que cada requerimiento dependía del resto, y si había que cambiar algo era necesario retroceder en todo el proyecto.

El nuevo planteamiento tiene como principio que el desarrollo se base en una funcionalidad concreta, se base en una visión de usuario y, que tras esta visión, se elaboren todas las labores técnicas para que el usuario pueda validar si lo que se está construyendo es exactamente lo que quiere.

En Scrum no es necesario tener un todo para tener algo, y siempre se podrá validar realmente que se anda por el camino correcto.

La visión en el proyecto de desarrollo va ser una visión de usuario y este tiene que entender lo que el ingeniero ha definido en el listado de funcionalidades recopilado en la toma de requerimientos; además, los desarrolladores tendrán que saber qué es lo que se tiene que hacer técnicamente para llevar a cabo las funcionalidades recopiladas.

Véase un ejemplo claro de cómo en el método tradicional se tomaban los requisitos:

Usuario da explicación de requerimiento: Un usuario define un requisito de un ERP, específicamente, en el módulo de facturación, donde solicita que al hacer una factura, el sistema permita buscar al cliente al destinatario de la factura.

Recogida de requerimiento técnico: El ingeniero piensa en el proceso y piensa que hay que montar desarrollo donde poder buscar a los diferentes clientes, donde habrá que consultar la tabla de clientes y donde el usuario tendrá que meter el identificador del cliente para sacar los datos específicos del cliente. También piensa en que se tiene que hacer una consulta en la base de datos utilizando la variable de entrada @idcliente.

Diseño de la pantalla: El diseñador diseña una pantalla donde el usuario mete un número de cliente y, tras meter el número de cliente, se conecta a la base de datos y trae los datos del cliente que el diseñador cree interesantes: nombre fiscal, CIF y ciudad donde está la sede del cliente.

Programación: El equipo de desarrollo hace toda la parte técnica para poner en marcha dicha funcionalidad.

Integración en la aplicación: Esta nueva funcionalidad se añade a las funcionalidades que existen con anterioridad.

Realización de las pruebas: Se realizan las pruebas para ver que realmente la nueva funcionalidad creada obtiene los datos correctos y que no da ningún error.

Incorporación de la funcionalidad: Se añade la funcionalidad a la aplicación y se asume que el usuario verá dicha funcionalidad cuando empiece a trabajar por la mañana, o bien se envía un email diciendo que dicha funcionalidad se ha incorporado.

Lógicamente, y con casi total seguridad, el usuario dará *feedback,* y posiblemente dirá que cómo él va conocer todos los identificadores de todos los clientes si hay unos mil clientes en la base de datos. Posiblemente, dirá que los datos extraídos no le valen, porque ellos por el nombre fiscal no conocen a los clientes, ya que el nombre que ellos conocen es otro, que la ciudad de la sede no les vale porque ellos facturan a oficinas distribuidoras, etc.

Esto ha provocado hacer un consumo de recursos dedicado sin que realmente esto fuera lo que quería el usuario. Ahora, hacer el cambio que el usuario necesita provoca volver a una fase inicial, ya que ni se buscará por código de cliente, ni se sacarán los mismos datos e, incluso, el tamaño de los campos en el diseño variará respecto al diseño original.

Si se cambia la metodología, todo el proceso varía sustancialmente, ya que en la fase inicial de análisis se entregará un documento funcional que trate sobre cómo va ser el proceso, cómo el usuario va ver los campos, qué campos habrá o cómo será la interactuación con la aplicación. De esta forma será más fácil detectar en las fases iniciales cómo va ser el proceso de desarrollo.

En la parte de diseño lo ideal es realizar un prototipo funcional para entregar al usuario, de tal forma que vea lo que será y así, si el usuario detecta algo que no es correcto, aún hay tiempo de modificar, ya que el proceso de desarrollo no ha empezado.

Posteriormente, tras la validación del usuario, se desarrollará e integrará y, nada más realizar dicha operación, se validará el resultado con el usuario para dar su conformidad final ayudar, incluso, a hacer las pruebas previas al paso a producción. Por último, se hará el despliegue.

Como puede verse, al aplicar la nueva metodología, todo el proceso de desarrollo gira en torno al usuario, quien valida cada fase del proceso, y los cambios son mucho más fáciles de acometer porque se detectan con anterioridad, en comparación con la metodología tradicional.

2.3. Gestión de la evolución del proyecto

Dentro de la metodología Scrum hay varios elementos que van a ayudar a entender y organizar dicha metodología; estos elementos son los *artefactos*, los *eventos* y los *roles*.

2.3.1. Artefactos

Scrum propone herramientas o «artefactos» para mantener organizados nuestros proyectos. Estos artefactos ayudan a planificar, organizar y revisar la situación del proyecto.

Todo el proyecto Scrum parte de un nuevo concepto denominado *«historia de usuario»,* es decir, cada una de las funcionalidades que describe el usuario y que formarán parte del listado funcional. El listado funcional definido por el usuario se llamará *«pila de producto»* o *Product Backlog.* La pila de producto es el instrumento metodológico del marco de trabajo Scrum, que se usa para listar las características *(features)* o *funcionalidades del software* a desarrollar, para priorizarlas de acuerdo a las necesidades del área de negocio.

La pila de producto permitirá tener una visualización de las funcionalidades a desarrollar, priorizar las características del *software* según las necesidades del negocio y dejar registrado el esfuerzo necesario para desarrollar la historia de usuario. Para ello, se deben seguir las reglas de administración de la pila de producto y saber cuándo una funcionalidad ha de entrar a la pila y cuándo no.

A continuación puede verse un ejemplo de cómo gestionar una pila de productos con los campos requeridos. Debe tenerse en cuenta que dicho control en una segunda fase se hará con la aplicación JIRA, pero en esta fase inicial se definirán los campos necesarios para poder controlar el proceso de desarrollo, antes de ver cómo JIRA gestiona el *Product Backlog*.

Ejemplo de Excel para gestionar Scrum

Id Historia	Fecha Inclusion	Usuario	Enunciado de la Historia	Estado	Esfuerzo	Prioridad	Comentarios
xx-xxx-xxxx	dd-mm-aaaa	Usuario1	Como un usuario, necesito esta funcionalidad, con la finalidad de tener este resultado				
xx-xxx-xxxx	dd-mm-aaaa	Usuario1	Como un usuario, necesito esta funcionalidad, con la finalidad de tener este resultado				
xx-xxx-xxxx	dd-mm-aaaa	Usuario1	Como un usuario, necesito esta funcionalidad, con la finalidad de tener este resultado				
xx-xxx-xxxx	dd-mm-aaaa	Usuario1	Como un usuario, necesito esta funcionalidad, con la finalidad de tener este resultado				
xx-xxx-xxxx	dd-mm-aaaa	Usuario1	Como un usuario, necesito esta funcionalidad, con la finalidad de tener este resultado				
xx-xxx-xxxx	dd-mm-aaaa	Usuario1	Como un usuario, necesito esta funcionalidad, con la finalidad de tener este resultado				
xx-xxx-xxxx	dd-mm-aaaa	Usuario1	Como un usuario, necesito esta funcionalidad, con la finalidad de tener este resultado				

En primer lugar, se encuentra el *identificador de la historia,* que puede ser un identificador numérico que vaya incrementando. Si se trabaja en varios módulos de la aplicación, sería interesante que este identificador describiera el módulo de la funcionalidad, por ejemplo, si la historia es del módulo de facturación, poner como identificador de la historia el FAC-01, donde «FAC» ya está diciendo que es una funcionalidad del módulo de facturación.

En el caso de empresas que desarrollen varios proyectos simultáneos y los recursos sean limitados, se podrá identificar también el proyecto dentro del identificador de la historia, por ejemplo: PROY1-FAC-01. De esta manera será posible llevar un *Product Backlog* de varios proyectos y hacer el seguimiento de los proyectos con un solo listado.

Es importante incorporar la *fecha de la inclusión* de la historia al *Product Backlog,* pues, de esta forma, podrán conocerse los tiempos desde que se incluye hasta que se finaliza la historia.

También se incluirá el *usuario* que ha sugerido la historia para tenerlo como referencia y que dicho usuario sea el responsable de validar la funcionalidad y todo el proceso, antes de poner la funcionalidad en la aplicación.

El *enunciado de la historia* es la descripción que hace el usuario de lo que quiere; nunca puede haber una descripción técnica, siempre tiene que haber una necesidad y un resultado de dicha necesidad.

El campo *estado* indicará en qué situación está dicha funcionalidad. Los posibles estados que podrían incorporarse a este campo podrían ser:

- Sin comenzar: La historia no ha sido empezada.
- En curso: La historia está en proceso de realización.
- Terminada: La historia ha sido finalizada y entregada.

Los diferentes estados pueden variar dependiendo del seguimiento que quiera hacerse, por lo que se podrían incluir nuevos estados, pero, a grandes rasgos, con los estados expuestos se intuye la situación de cada una de las historias.

Esfuerzo: Es la estimación de tiempo que puede costar realizar una historia; este tiempo se puede especificar en horas o días, pero cuanto más concreta sea la estimación, mayor control podrá llevarse del proyecto, para controlar los costes, las desviaciones de tiempos, etc.

Prioridad: Es la importancia de la historia para el negocio de los desarrolladores o para el negocio del cliente. Lo ideal es tener tres nivel de priorización, ya que, si se incorporan más niveles, al final, no se identificará qué es más o menos importante. Así que, la recomendación es tener prioridad 1, prioridad 2 o prioridad 3, donde la prioridad 1 es lo más importante para el negocio del usuario. Más adelante se explicará un método más preciso para asignar prioridades denominado método MoSCoW.

Comentarios: Simplemente se incorporan si es necesario tener en cuenta algo adicional o más específico como referente a la historia de usuario.

Hay algunas cosas que habrán de ser tenidas en cuenta a la hora de elaborar una buena pila de producto. En primer lugar, en la pila de producto no puede haber ningún requerimiento técnico, es decir, no se debe incluir la frase «desarrollar un procedimiento almacenado para…», solo habrá historias entendibles por los usuarios.

En la pila de producto no puede haber historias no aprobadas por el responsable del proyecto o historias que no se van a desarrollar. Si no está claro lo que quiere el usuario, no se incorpora a la historia; se aclara lo que se requiere y posteriormente se incorpora a la historia. No debe haber historias sin prioridad, ya que, a la hora de seguir desarrollando, no puede ser un técnico el que decida por dónde continuar la evolución del producto.

Otro concepto importante dentro de la metodología Scrum es el *sprint* o *interacción,* que se define como el nombre que recibe el conjunto de historias que se desarrollarán en un periodo de tiempo inferior a un mes. Este conjunto de historias se denomina *«pila de sprint»* o *Sprint Backlog* y tiene que ser totalmente funcional.

Si para llevar a cabo una historia se tienen que desarrollar varios procesos técnicos incluidos en otros módulos funcionales, habrá que acometerlos, con tal de disponer de una funcionalidad completa.

El *Sprint Backlog* ha de tener una fecha fija de entrega y esta ha de ser inamovible, al igual que las historias que incluyen dicho *sprint*. Al finalizar el *sprint* se puede hacer una entrega al usuario en modo de versión, o bien esperar a finalizar algún otro *sprint*. Lo ideal es que, al finalizar cada *sprint,* el usuario valide la interacción, para así tener la seguridad en todo momento de estar construyendo lo que se quiere.

La *versión* está definida como los entregables oficiales que se hacen al usuario, y en cada versión podrán incluir varias historias que, a su vez, habrán sido incluidas en algún *sprint* previo.

Por último, al finalizar el *sprint* resultará el *incremento*, que es lo que se incorpora a nivel funcional a la aplicación que se está desarrollando.

2.3.2. Los eventos

En Scrum se requieren momentos para asegurar que se están haciendo las cosas bien, para revisar lo que se está haciendo y para tomar decisiones oportunas que permitan mantener un proceso que siempre agregue valor. En cada uno de estos eventos se aprovecha para realizar la inspección y adaptación de algún aspecto. Además, estos eventos tienen como finalidad minimizar la necesidad de reuniones no definidas en Scrum. Cada uno tiene una duración máxima, con lo que se asegura un desperdicio mínimo de tiempo.

Cuando se han definido todos los puntos funcionales de la aplicación que se tienen que desarrollar, y cuando se definen todas las tareas técnicas que hay que realizar para poder llevar a cabo la construcción de cada una de las funcionalidades, se ejecutará el primer evento de organización del trabajo: la *reunión de planificación de sprint*. En esta reunión es donde se organizará el trabajo, siempre tomando como base las prioridades y necesidades de negocio del cliente. También se determinan cuáles son y cómo van a ser las funcionalidades que se incorporarán al producto en el siguiente *sprint*.

A esta reunión han de acudir todos los miembros del equipo Scrum, y dicha reunión es la base del comienzo de las entregas. La reunión ha de ser conducida por el responsable del proceso de Scrum, y también sería lo ideal que acudiera el propietario del producto, que es la persona que puede tomar decisiones respecto a determinados aspectos del proceso de fabricación.

También han de acudir todos aquellos miembros que tengan relevancia en el proyecto. Todos los roles serán definidos más específicamente en los siguientes puntos del presente documento.

Lo ideal es que estas reuniones tengan una duración máxima de una jornada de trabajo, aunque, lógicamente, dependerá de la duración de la entrega, pero, como se ha comentado en puntos anteriores, las entregas no han demorarse más de un mes.

La reunión tiene que tener un enfoque claro y preciso, no se deben tratar otros temas que los siguientes:

- Qué se va entregar al terminar el *sprint*. Listado de funcionalidades a incluir en la entrega.

- Qué trabajo es necesario hacer para llegar al objetivo y cómo va a organizarse en el equipo. Dentro de las funcionalidades a desarrollar, cuánto tiempo se empleará en cada una, qué recursos serán necesarios para cada una de ellas y quién las va a acometer.

La reunión ha de ser una reunión abierta, donde todos los miembros del equipo deben opinar, donde hay que tener un punto de realismo en cuanto a los objetivos y donde el cliente tiene que respetar las decisiones que se han tomado en la misma. En esta reunión se establecen las bases funcionales que hay que desarrollar y, una vez determinadas estas bases, no se pueden alterar.

Las conclusiones de la reunión deben ser:

- Pila del *sprint,* o funcionalidades a incorporar al proyecto para la próxima entrega.

- Duración del *sprint,* con fecha estimada de finalización del mismo.

- Fecha de reunión de revisión del *sprint.*

- Objetivo perseguido en el negocio una vez finalizada la entrega.

El soporte para hacer el seguimiento de los temas a tratar ha de ser, por ejemplo, una hoja Excel, o bien una herramienta de gestión de proyectos, en nuestro caso, JIRA.

De forma posterior a la reunión de planificación del *sprint* se realizará una serie de reuniones diarias denominada *Scrum diario;* esta reunión es la reunión diaria de seguimiento, y no ha de durar más de quince minutos. El objetivo de esta reunión es sincronizar con el equipo la situación de cada una de las tareas y es donde se establecen todas las tareas que cada miembro va hacer desde esta reunión diaria hasta la reunión diaria del día siguiente.

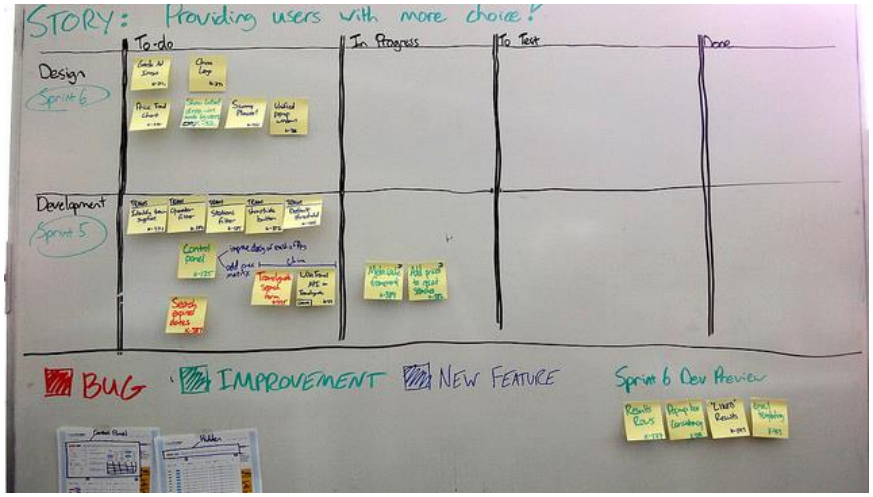
Se recomienda que dicha reunión se haga todos los días y siempre a la misma hora. No es necesario celebrarla en una sala, pues se recomienda que los asistentes estén de pie para dar rapidez al evento, y sobre un tablero o pizarra se va viendo la evolución de las tareas. Pueden utilizarse notas adhesivas tipo pósit que identifiquen cada una de las tareas a realizar; de esta manera se podrá observar cómo van evolucionando las mismas en el avance del *sprint*.

El resultado de la reunión es la pila del *sprint* y el gráfico de avance actualizado. También es importante identificar si hay algún elemento que impida el avance.

Por último, en la reunión diaria cada miembro del equipo habrá de explicar qué ha logrado desde la anterior reunión, lo que va hacer para la siguiente reunión y también comentará los problemas que haya encontrado o prevea encontrar.

Formato de la reunión de Scrum

Pizarra de Scrum



A continuación se describen algunos de los campos que podrían incluirse en la pizarra de Scrum.

- *Nombre del sprint:* Identificación del nombre del *sprint* en el que se trabaja; se puede diferenciar el *sprint* por tipo de trabajo (diseño, desarrollo), o bien, en caso de hacer el seguimiento de varios módulos o proyectos, pueden identificar los mismos.

- *To do:* Son todas las tareas a realizar en el *sprint* que está siendo organizado.

- *In progress:* Tareas que en un momento dado están en curso y en las que se está trabajando.

- *To test:* Tareas que se han finalizado y están siendo probadas. Si dentro de algunas pruebas falla la funcionalidad, permanecerán en este estado.

- *Done:* En este apartado aparecerán todas las tareas que se han finalizado y están listas para la entrega.

En este tipo de pizarra, aparte de definir las tareas, también se identifica si dichas tareas son: errores *(bugs),* mejoras sobre una funcionalidad existente *(improvement)* o nuevas funcionalidades *(new feature),* en función de lo cual se escribirá la tarea en el pósit en diferentes colores.

Dicha pizarra también se encontrará en la aplicación JIRA de forma virtual, por si se desea hacer el seguimiento utilizando un proyector o una pantalla de ordenador para, así, poder sustituir la pizarra.

Tras varias reuniones diarias de *sprint*, se llegará a la fecha de entrega del *sprint,* cuando todos los avances serán presentados en la revisión del *sprint*.

Como se ha mencionado anteriormente, una de las conclusiones que se obtenían en la reunión de planificación del *sprint* era la fecha de la reunión de *revisión del sprint,* que es la fecha en la que se revisarán los incrementos realizados respecto a lo tratado en la reunión de panificación.

Dicha reunión no ha de durar más de cuatro horas para validar *sprint* largos; para *sprint* de una o dos semanas, con dos horas de reunión ha de ser suficiente.

El objetivo de la reunión es que el *propietario del producto* compruebe el progreso del producto; se identifican las funcionalidades implementadas y se evalúan para determinar si se pueden dar por finalizadas o, por el contrario, no se ha entendido lo que se quería y hay que volver a completar dicha funcionalidad de una manera correcta. En esta reunión se recomienda hacer una *demo* de las funcionalidades realizadas, para que se dé la completa validación de las mismas.

Lo importante de la reunión es conocer el *feedback* de propietario del producto, determinar cuándo va ser la reunión de planificación del siguiente *sprint* y también se puede tratar cuándo se desea que dichas funcionalidades puedan salir en versión o se pueden incorporar a producción.

En caso de que no se haya entregado alguna de las funcionalidades comprometidas, se han de explicar los motivos. Además, si hay algún hecho claro que se pueda mostrar, ha de enseñarse para afianzar claramente que se ha hecho todo lo posible con el fin de conseguir los objetivos, pero no ha sido posible llegar a tiempo por los motivos expuestos.

Por último, hay un evento importante que ha de acometerse, ya que es la reunión para ir incrementando la calidad del proceso de desarrollo. Esta reunión se denomina reunión *retrospectiva,* y se realiza tras finalizar cada *sprint* y antes de la reunión de planificación del nuevo *sprint*. Esta reunión debe tener una duración de no más de tres horas y a ella acuden solo los miembros del equipo de desarrollo. Consiste en un autoanálisis sobre cómo se ha trabajado en el *sprint* entregado. De esta manera y entre todo el equipo se puede evaluar qué es lo que ha fallado o cómo se puede mejorar para las siguientes entregas, y permite planificar mejoras en las entregas. Dicha reunión se considera una reunión técnica para mejorar el marco del trabajo y la calidad de los procesos.

Un tema a tener en cuenta respecto a todos estos eventos, y en el que se suele caer en todos los equipos de trabajo, es el hecho de realizar reuniones poco productivas cuando la idea de los eventos es todo lo contrario: tener reuniones para mejorar el rendimiento y la calidad de cada uno de los avances que se dan en el proyecto. Esto es uno de los cambios que toda empresa que quiera aplicar dicha metodología ha de variar respecto a como lo realiza en la actualidad. Estos puntos son básicos para aplicar la metodología. Por eso se recomienda para hacer las reuniones eficaces llevar a cabo estas pautas:

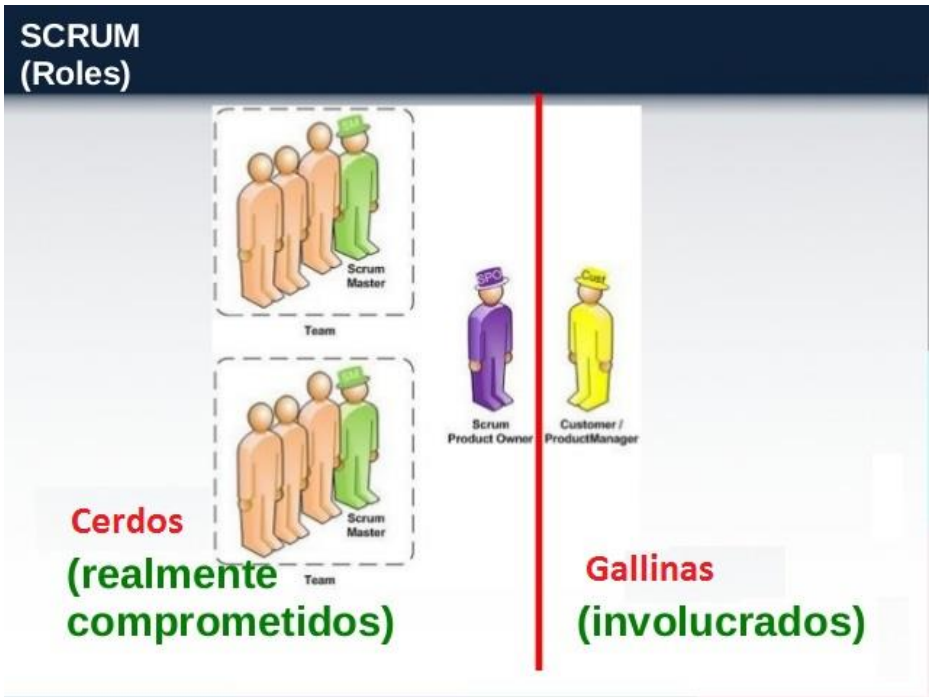
1. Que todos los asistentes tengan claridad respecto a los temas a tratar y conozcan el objetivo de la reunión.
2. Tener claro quién ha de acudir a la reunión y evitar oyentes que no estén directamente ligados a los temas que se van a tratar.
3. Saber calcular la duración de la reunión y no sobrepasar lo calculado.
4. Dejar siempre que la reunión sea fluida y haya un largo turno de ruegos y preguntas.
5. Es conveniente redactar un acta donde se especifiquen claramente los acuerdos alcanzados.
6. El moderador de la reunión ha de centrase en los puntos de los cuales se extraerán conclusiones y evitar tratar temas de manera difusa.

2.3.3. Los roles

Hasta ahora se han mencionado algunas de las personas que intervienen en el proceso Scrum, pero sin explicar claramente cuáles son su rol y su función dentro de la metodología. Por eso, en este apartado se definirá claramente el *rol* de los miembros de Scrum y qué labor tienen dentro del proyecto.

> Todas las personas que intervienen, o tienen relación directa o indirecta con el proyecto, se clasifican en dos grupos: comprometidos e implicados. En círculos de Scrum es frecuente llamar a los primeros «cerdos» y a los segundos «gallina».
>
> El origen de estos nombres está en la siguiente metáfora que ilustra de forma gráfica la diferencia entre «compromiso» e «implicación» en el proyecto:
>
> Una gallina y un cerdo paseaban por la carretera. La gallina preguntó al cerdo: «¿Quieres abrir un restaurante conmigo?».
>
> El cerdo consideró la propuesta y respondió: «Sí, me gustaría. ¿Y cómo lo llamaríamos?».
>
> La gallina respondió: «huevos con jamón».
> El cerdo se detuvo, hizo una pausa y contestó: «Pensándolo mejor, creo que no voy a abrir un restaurante contigo. Yo estaría realmente comprometido, mientras que tu estarías sólo implicada».
>
> (ScrumManager, 2014c).

Representación gráfica del equipo Scrum



Respecto a los roles, sería conveniente hacer antes una pequeña matización sobre los cambios que tiene que acometer aquella empresa u organización que quiere aplicar Scrum. La empresa tiene que empezar a cambiar la forma de ver al equipo; desde el momento en el que se quiere aplicar Scrum, la empresa ya no tiene recursos, sino equipo, y ya no tiene ejecutores que hacen lo que se les dice, sino aportadores.

Como buen equipo Scrum, se debe conseguir que todos y cada uno de los miembros del equipo se sientan parte del proyecto, que no se sientan excluidos y, para eso, los aportes de cada uno de ellos han de tenerse en consideración.

El aplicar Scrum no es solo una metodología, sino que se podría decir, incluso, que es un método de motivación, organización y un método para conseguir el cien por cien de cada persona que forma parte del proyecto, por eso se define al *team* como los «realmente comprometidos». Si se diera la circunstancia de que alguna persona del equipo no esté en esta situación, en primer lugar, ha de intentarse reconducirla y, en el caso de ser una situación imposible de reconducir, ha de prescindirse de ella, porque puede desmotivar al resto del grupo y todo el esfuerzo puesto en la aplicación del método se puede tirar por la borda.

En primer lugar, entre los roles dentro de Scrum, está el *propietario del producto,* quien toma las decisiones del cliente. Su responsabilidad es el valor del producto, en otras palabras, es el encargado de que el producto que se está construyendo vaya evolucionando y se vayan implementando todas las funcionalidades requeridas para acometer todas las necesidades del negocio. Cuando es un desarrollo interno para la propia empresa, este rol normalmente ha de asumirlo una persona que conozca el negocio, que conozca los objetivos de la empresa y que tenga relación con todos los departamentos corporativos de la organización. Lo ideal es que sea una persona cercana a la dirección de la compañía y que tenga poder de decisión a la hora de tomar caminos en el desarrollo, y pueda defender el proyecto en cualquier circunstancia y frente a cualquier persona en la organización.

Se puede dar el caso de que se forme un comité de producto, cuando entre varias personas se puedan tomar decisiones respecto a la evolución del producto, pero, al fin y al cabo, debe de recaer toda la responsabilidad final en una sola persona.

En el caso de una construcción para un cliente externo, el puesto de propietario del producto podrá recaer sobre alguna persona específica del cliente o de un *project manager* de la empresa de desarrollo que tenga comunicación directa con el cliente para tomar decisiones específicas sobre los temas que se vayan tratando. No obstante, siempre es conveniente ir enseñando *demos* al cliente final para que también vaya validando la progresión de la construcción.

Independientemente de la persona que sea el propietario del producto, esta ha de conocer la metodología de trabajo para que desempeñe con éxito sus funciones dentro de Scrum.

Otro aspecto muy importante y que debe defender el propietario del producto es respetar las decisiones tomadas y nunca modificar las prioridades ni elementos de la pila de productos que existan en el *sprint* en curso. Es habitual que este punto sea cambiante en las empresas que organizan el trabajo de manera tradicional, y es importante en este nuevo método de trabajo respetar las entregas planificadas para no poner en riesgo la entrega del *sprint*. Si se determina alguna nueva funcionalidad importante a incluir, ha de implementarse en el siguiente *sprint* y no en el que está en curso.

El rol fundamental dentro del Scrum y el más implicado en todo el desarrollo es el *equipo de desarrollo* o *team*. Normalmente está compuesto por un equipo técnico, de diferentes perfiles profesionales, entre los que podrá haber programadores, diseñadores, analistas funcionales, analistas técnicos, arquitectos o probadores. Los miembros del equipo no se especializan, sino que todos deben tener, en mayor o menor medida, habilidades en todas las actividades implicadas. Los miembros del equipo deciden entre todos ellos cómo se organiza el trabajo y nadie (ni siquiera el *Scrum master*) puede decir cómo lo deben hacer.

El equipo debe estar compuesto por entre cuatro y ocho personas. Más allá de ocho resulta más difícil mantener la comunicación directa, y se manifiestan con más intensidad los roces habituales de la dinámica de grupos (que comienzan a aparecer a partir de seis personas). En el cómputo del número de miembros del equipo de desarrollo no se consideran ni el *Scrum master* ni el propietario del producto. En el caso de que haya un equipo superior en número de personas, se han de formar varios equipos, de tal forma que se puedan dividir los trabajos del proyecto, por ejemplo: en módulos funcionales, de tal forma que un módulo será asignado a un equipo Scrum y otro modulo a otro; en este caso, el *Scrum master* podría ser la misma persona, al igual que el dueño del producto.

Es importante destacar que la responsabilidad total del incremento del proyecto recae ahora sobre todo el equipo y no sobre personas concretas. Cada miembro aportará su conocimiento al grupo y este conjunto de conocimiento será utilizado por el grupo de la mejor manera. Cada uno tomará responsabilidad de sus tareas y buscará ayuda en el grupo para resolver los inconvenientes.

El equipo deberá conocer en todo momento la visión del propietario del producto; aportarán y colaborarán con el propietario del producto, compartirán de forma conjunta el objetivo de cada uno de los *sprints* y la responsabilidad de conseguir las entregas. Todos los miembros participan en las decisiones, todos respetarán las opiniones y los aportes de todos y, lógicamente, deberán conocer la forma de trabajo mediante Scrum.

Por último, está la figura del *Scrum manager* o *Scrum master,* cuyo perfil profesional podría ser el de un *project manager* con conocimientos técnicos y conocimiento también del negocio. Es el encargado de cumplir las normas de Scrum asegurando que se trabaje utilizando dicha metodología. Es el asesor tanto del dueño del producto como del *team*. Sus funciones principales son: liderar el proyecto, asesorar y acompañar tanto al equipo como al dueño del producto.

Por otro lado, el *Scrum master* ha de validar la pila de producto; es el que modera las reuniones, el que resuelve los problemas para llegar a las entregas de los *sprint* y, sobre todo, ha de ser líder a la hora de implementar las prácticas del Scrum en toda la organización.

2.3.4. Cambio de cultura organizativa

Está claro que el Scrum va ayudar a dar un salto significativo organizacional dentro de cualquier empresa que quiera empezar a aplicar dicha metodología, pero es necesario un cambio cultural dentro de la organización.

Los directores y gestores de la compañía tienen que tener una total creencia en la metodología ya que hay puntos que les puede resultar inasumibles, puesto que hasta ahora siempre se tenía mentalidad de «jefe dice y empleado hace», y ahora se cambia a una visión donde es el propio equipo el que se autoorganiza y el que toma sus propias decisiones.

También los propios miembros del equipo tienen que cambiar el paradigma, pues ahora cada compañero debe confiar en el otro, deben respetar cada uno sus conocimientos y sus capacidades y la responsabilidad del resultado final recae completamente en ellos.

También cada miembro ha de imponerse su autodisciplina pues no se va a imponer desde fuera ninguna disciplina. Por último, el equipo ha de tener una total transparencia; va tener toda la información necesaria y también la visibilidad del desarrollo que están acometiendo; conocen en todo momento lo que se está haciendo mal y por qué, y también lo que se está haciendo bien, sin escatimar en las felicitaciones al grupo.

A continuación se expone cómo por norma general se organizan las empresas según una encuesta realizada por la Scrum Alliance en este mismo año. El universo de la encuesta han sido *5 000 personas de 108 países diferentes que representan a más de 14 industrias* distintas: Desarrollo de *software*, IT en general, financieras, de salud, etc.

- Los equipos formados dentro del Scrum son de como máximo siete personas.
- Los *sprints* suelen duran unas dos semanas.
- Un proyecto suele constar de siete o más *sprints* (para el 55 % de los encuestados).
- Se suele hacer casi siempre un *sprint planning* antes del *sprint* (para el 83 % de los encuestados).
- Se suelen hacer reuniones diarias de Scrum para ver las tareas diarias (para el 81 % de los encuestados).
- Se suelen hacer reuniones de retrospectiva al terminar el *sprint* o el proyecto (para el 81 % de los encuestados).

3. ¿CÓMO GESTIONAR LA *METODOLOGÍA?*

Hasta este momento del presente trabajo se ha ido exponiendo, punto por punto, cómo hay que organizar el departamento de desarrollo para aplicar Scrum. Cómo es cada uno de los pasos que se deben dar para tener claridad respecto a la metodología y también algunos aspectos que habrá que tener muy en cuenta para no caer en errores y conseguir el éxito de trabajar con Scrum.

Para que la implantación de la metodología sea exitosa habrá que utilizar alguna herramienta de *software* para poder hacer el seguimiento de los proyectos y tener controladas las fases de desarrollo. Para ello se utilizará la aplicación informática JIRA.

3.1. La herramienta JIRA

JIRA es una aplicación *software* que permite gestionar proyectos. Inicialmente era una herramienta para gestionar incidencias, pero gracias a la evolución del producto hoy en día permite manejar el ciclo de vida de cualquier proyecto de desarrollo.

Dentro de sus módulos JIRA dispone del complemento JIRA Agile; este complemento permite hacer una gestión ágil de proyectos y aporta la pizarra Scrum, que permite hacer seguimiento de los *sprints,* hacer seguimiento de las versiones y llevar el control del *Product Backlog*.

La aplicación es una herramienta comercial que se puede adquirir para ser instalada en los propios servidores *onsite,* o bien adquirir una licencia *cloud* donde se utilizan los servidores del fabricante del producto para su utilización.

El precio del *software* es realmente económico si el equipo de desarrollo no está compuesto por más de diez personas. A continuación se muestra una tabla de precios de la solución JIRA.

Producto *onsite*	Número de usuarios	Precio año
JIRA *Software*	10	10 $
JIRA *Software*	25	1 800 $
JIRA *Software*	50	3 300 $
JIRA *Software*	Ilimitados	36 000 $

Producto *cloud*	Número de usuarios	Precio mes
JIRA *Software*	10	10 $
JIRA *Software*	25	150 $
JIRA *Software*	50	300 $
JIRA *Software*	2000	1 500 $

Una de las ventajas de JIRA es que la herramienta dispone de varios módulos que se pueden adquirir aparte, y que mejoran aún más el control a la hora de desarrollar y mantener cualquier aplicación que se haya fabricado o se esté fabricando.

Algunos de los módulos que se pueden encontrar en JIRA son:

JIRA Service Desk: Es un módulo para poder gestionar el soporte informático de cualquier empresa. Incidencias de *software*, incidencias de *hardware,* gestión de SLAs, herramienta para el centro de atención al usuario, etc.

Bitbucket: Es un módulo que permite integrar JIRA con el repositorio de código fuente. Dicho modulo es compatible con Git y con Mercurial. Este módulo JIRA no solo permite controlar el ciclo de vida del *software,* sino también controlar el código fuente de cada uno de los proyectos en los que se esté trabajando.

Confluence: Este módulo permite gestionar toda la documentación del proyecto, de una forma colaborativa. Es una aplicación *web* donde se dispone de toda la documentación, donde se pueden subir y controlar todos los documentos necesarios para que sean consultados por el usuario, así como los documentos internos del equipo técnico.

Tempo Timesheets: Este módulo se utiliza para poder llevar una mejor gestión de los tiempos de cada uno de los proyectos, así como para conocer la dedicación de cada uno de los miembros del equipo, lo que permite hacer un *traking* de cada una de las historias o tareas en las que se esté trabajando.

De todos los módulos comentados, el más interesante es el *Tempo Timesheets* y se recomienda también realizar la adquisición del mismo, con un coste de 10 $ al mes para 10 usuarios en modo *cloud* o 10 $ al año en modo *onsite*.

3.2. Comenzando a utilizar JIRA

Lo primero que habrá que hacer antes de empezar a trabajar con JIRA es decidir si utilizar una solución *onsite* o utilizar la versión *cloud*. Tras esta decisión empieza el trabajo con la herramienta.

Dentro de la página *web* oficial del fabricante de JIRA está disponible de toda la documentación para realizar la instalación de la aplicación en caso de utilizar la solución *onsite* o, simplemente, desde la misma página *web* es posible activar la solución *cloud,* donde se habilitará el entorno en cuestión de minutos.

La documentación de la instalación *onsite* de la herramienta está disponible en este enlace:
<https://confluence.atlassian.com/adminjiraserver070/installing-jira-applications-749382623.html>.

Por otro lado, si lo que se desea es activar la solución *cloud,* es posible hacerlo en este otro enlace:
<https://es.atlassian.com/*software*/jira/pricing>.

Para empezar a trabajar con la aplicación, en primer lugar, habrá que crear los usuarios. De principio, habrá un usuario administrador de la plataforma que puede ser o no miembro del equipo Scrum y, por otro lado, tiene que estar dado de alta cada uno de los miembros del equipo de desarrollo. El *dueño del producto* no tiene por qué estar dado de alta en la herramienta, pero, si fuera requerido, podría tener también un usuario de acceso en modo consulta.

El elemento de trabajo básico en JIRA es el *proyecto de tipo software*, por lo que debe crearse un proyecto por cada aplicación en la que se trabaje o si, como se ha comentado anteriormente, el equipo de trabajo es muy grande, se podrán crear tantos proyectos como equipos Scrum haya en nuestra organización. Cada equipo Scrum ha de trabajar sobre un proyecto y el proyecto puede ser un *software* completo o un módulo específico.

Dado que JIRA dispone de toda la documentación a nivel de usuario, no será necesario centrar la atención aquí en explicar cómo funciona la aplicación, sino en cómo utilizar la herramienta para gestionar Scrum.

En este enlace puede consultarse toda la documentación de usuario para ver cómo crear proyectos, crear incidencias, configuración del entorno, etc.:
<https://confluence.atlassian.com/jira*software*server070/jira-*software*-documentation-762877196.html>.

Como recomendación si en la actualidad se está llevando la gestión del software con otra aplicación, se deben de traspasar a JIRA únicamente las historias de usuario que están pendientes de hacer o las que están en curso. Las historias ya terminadas se recomienda no migrarlas. También se tendrá que decidir a partir de qué fecha se deja de utilizar la antigua herramienta y cuando se empieza a trabajar con JIRA, para formar a todo el equipo en la herramienta.

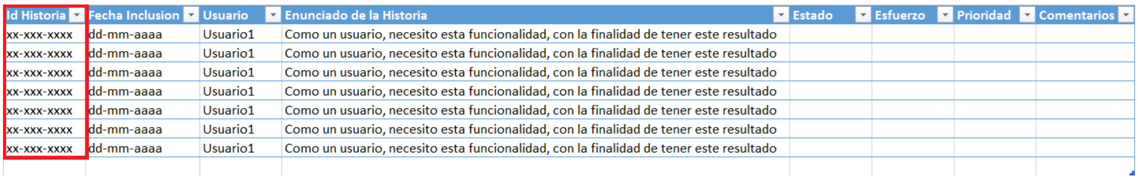
3.2.1. Las incidencias en JIRA

El elemento básico que utiliza JIRA para gestionar los trabajos son las *incidencias*, pero no hay que confundir este término con el registro de los errores que tiene nuestra aplicación. JIRA denomina a cada elemento de trabajo «incidencia», independientemente que sea un error, una nueva funcionalidad, una historia o una tarea técnica.

Las incidencias en JIRA son cada uno de los trabajos que se tienen que realizar dentro de un proyecto, por lo que dentro de un proyecto se tendrá un conjunto de incidencias.

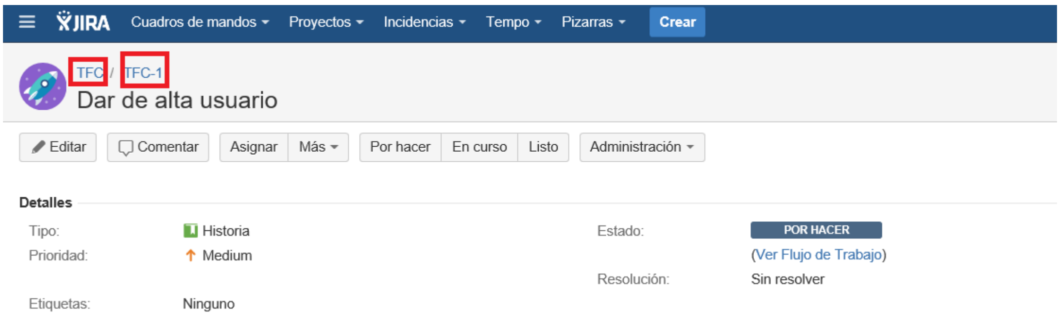
Toda incidencia tiene un identificador único que se le asigna cuando se crea la incidencia; este identificador es el mismo identificador que anteriormente había en la hoja Excel de seguimiento de Scrum, y utiliza la clave del proyecto para componer dicho identificador. A continuación se puede ver un ejemplo de identificador TFC-1:

Id. Historia: Es el identificador de las historias que se habían definido en el punto 2.3.1. del presente trabajo



Id Historia	Fecha Inclusion	Usuario	Enunciado de la Historia	Estado	Esfuerzo	Prioridad	Comentarios
xx-xxx-xxxx	dd-mm-aaaa	Usuario1	Como un usuario, necesito esta funcionalidad, con la finalidad de tener este resultado				
xx-xxx-xxxx	dd-mm-aaaa	Usuario1	Como un usuario, necesito esta funcionalidad, con la finalidad de tener este resultado				
xx-xxx-xxxx	dd-mm-aaaa	Usuario1	Como un usuario, necesito esta funcionalidad, con la finalidad de tener este resultado				
xx-xxx-xxxx	dd-mm-aaaa	Usuario1	Como un usuario, necesito esta funcionalidad, con la finalidad de tener este resultado				
xx-xxx-xxxx	dd-mm-aaaa	Usuario1	Como un usuario, necesito esta funcionalidad, con la finalidad de tener este resultado				
xx-xxx-xxxx	dd-mm-aaaa	Usuario1	Como un usuario, necesito esta funcionalidad, con la finalidad de tener este resultado				
xx-xxx-xxxx	dd-mm-aaaa	Usuario1	Como un usuario, necesito esta funcionalidad, con la finalidad de tener este resultado				

Pantalla de incidencias de JIRA



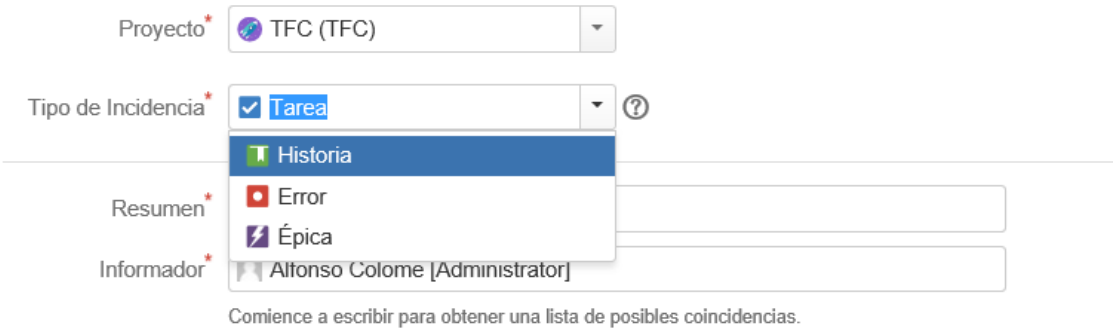
TFC: Es la clave del proyecto (nombre de proyecto trabajo fin de carrera)

TFC-1: Es el identificador de la primera incidencia creada

3.2.2. Los tipos de incidencias

Cuando se crea una incidencia en JIRA, por defecto, hay que especificar de qué tipo es. En el presente caso, la mayoría de las incidencias, al estar implantando la metodología Scrum, serán de tipo *historia,* que serán las *«historias de usuario»* que se han comentado y explicado en el punto 2.3.1, y se representan en JIRA mediante este icono . Aparte de las historias, se podrá identificar algún otro tipo de incidencia, para que de una forma visual sea posible reconocer si es un requerimiento funcional, o bien, otro tipo de trabajo. Por esto habrá algún tipo de incidencia adicional a la historia que va a ayudar a la hora de gestionar los trabajos.

Ejemplo de cómo se selecciona el tipo de incidencia en JIRA



Los tipos de incidencias que se utilizarán en JIRA aparte de las historias son:

- *Error:* Es una problema del *software* que se está desarrollando que impide o dificulta el correcto funcionamiento del producto.

- *Tarea:* Este tipo de incidencia se utiliza para representar trabajos técnicos o historias técnicas. Podría haber algún tipo de confusión con este tipo de incidencias respecto a la historia de usuario. Para aclarar este tipo de incidencia sería conveniente añadir que se utilizará cuando la historia que se crea es del tipo no funcional. Son aquellos aspectos técnicos que hay que realizar y son independientes de las historias funcionales de los usuarios (seguridad, alta disponibilidad, lentitud de la aplicación, instalación de la plataforma, etcétera).

- *Épica:* Este tipo de incidencia se utiliza para una historia de usuario muy grande que haya descomponer para que sea mucho más manejable. Con este tipo de incidencia se definirá un conjunto de historias agrupadas en una historia mucho más grande, para cuya finalización sean necesarios varios *sprints.* Las épicas no pueden formar parte de otras épicas y una incidencia no puede formar parte de dos épicas a la vez. Otro aspecto que hay que tener en cuenta es que en JIRA se pueden crear incidencias directamente desde la propia épica, y estas incidencias estarán ya incluidas directamente dentro de la épica.

Pantalla para crear incidencias dentro de épicas

3.2.3. Las subtareas

Las subtareas son trabajos específicos que hay que realizar cuando se trabaja en una incidencia, por lo que se puede inferir que la incidencia es la descripción que hace el usuario de la funcionalidad y las subtareas son todo aquello que es necesario realizar para llevar a cabo dicha incidencia.

El poder gestionar subtareas, será útil saber qué es lo que hay que hacer técnicamente para entregar una funcionalidad a nivel de historia. Por eso, cuando un usuario solicita un requerimiento funcional habrá que analizar qué es lo necesario técnicamente y esto asociarlo a la historia funcional. En todo momento, con esta división, se podrá observar lo que queda pendiente técnicamente para terminar la incidencia.

Las subtareas son muy similares a las incidencias, pero estas se crean directamente desde dentro de una incidencia previamente existente. Estas subtareas no serán visibles desde el *Product Backlog,* pero sí se pueden visualizar si se consulta la información de una de las incidencias. Las subtareas, a diferencia de las incidencias, no se podrán clasificar en tipos de subtareas.

Con el ejemplo que se daba al inicio de este documento (apartado 2.2) se explicará a continuación la manera de organizarlo en JIRA.

Usuario da explicación de requerimiento: Un usuario que define un requisito en un ERP, específicamente en el módulo de facturación, donde solicita que, al hacer una factura en el sistema, sea posible buscar el cliente destinatario de la factura.

Incidencia de JIRA de tipo historia: Permitir buscar a un cliente cuando se vaya a hacer una factura. Dentro de la incidencia se crean las siguientes subtareas:

- Subtarea 1: Creación de documento funcional.
- Subtarea 2: Validación de documento por parte del usuario.
- Subtarea 3: Diseño de prototipo funcional.
- Subtarea 4: Validación prototipo funcional.
- Subtarea 5: Diseño final de entorno.
- Subtarea 6: Desarrollo funcional.
- Subtarea 7: Pruebas funcionales.
- Subtarea 8: Validación del usuario previa a la entrega.
- Subtarea 9: Despliegue de la funcionalidad.

3.2.4. Otros datos específicos de las *incidencias*

A igual que existe un campo tipo a la hora de crear incidencias, también se dispone de otros campos que se utilizarán para poder tener más información específica sobre la incidencia que se esté dando de alta.

Existen unos campos específicos en JIRA a la hora de crear incidencias, pero la aplicación permitirá configurar campos adicionales si en el proyecto en curso fuese necesaria información adicional a la estándar.

A continuación se describirán algunos de los campos importantes y necesarios a la hora de crear las incidencias, y que también son necesarios para poder gestionar Scrum con JIRA.

Resumen: Será el título de la incidencia. Este campo también estaba especificado en la hoja de Excel de gestión de Scrum que se manejaba en los apartados anteriores, denominado como «Enunciado de la historia». Este campo resumen servirá para que todo el equipo de Scrum sepa identificar cuál es el objetivo de esta incidencia.

Prioridad: Este campo se utilizará para saber el impacto de cada uno de los trabajos que hay dentro del *backlog*, y así poder ordenar cada uno de los trabajos a planificar en el proyecto. Adicionalmente, luego se verá que se usa un método específico de priorización de tareas basado en etiquetas. Las posibles opciones de priorización que se den en el campo son los que se muestran en la siguiente tabla:

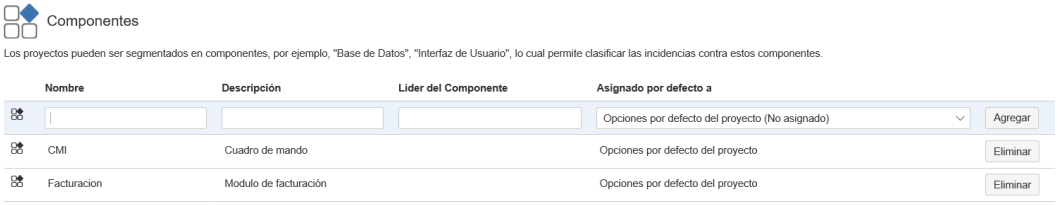
Tabla de la prioridad de las incidencias en JIRA

Highest	Es una incidencia que hace bloquear las tareas de desarrollo e impide seguir avanzando.
High	Indica que la incidencia está causando un problema grave y requiere una atención urgente.
Medium	Indica una pérdida importante en la funcionalidad, pero no impide seguir trabajando con la aplicación de forma general.
Low	Indica una pérdida menor en la funcionalidad, pero con una forma alternativa para seguir trabajando. El impacto es relativamente pequeño.
Lowest	Es el problema con la prioridad más baja. No existe pérdida de servicio a nivel funcional.

Componente: Normalmente, cuando se trabaja en un proyecto *software*, la aplicación que se está desarrollando está formada por varios tipos de componentes o módulos. Por ello, cuando se crea una incidencia en JIRA, es posible identificar a qué componente pertenece esta incidencia.

Por defecto en JIRA, cuando se crea un proyecto, aparece sin componentes y es el administrador de la herramienta el que tiene que dar de alta los diferentes tipos de componentes que tendrá el proyecto.

Pantalla para crear componentes del proyecto



Versiones: A la hora de crear las incidencias dentro de JIRA, hay dos campos relacionados con las versiones del proyecto. Se suele utilizar a la hora de gestionar las incidencias de tipo error, ya que no tiene mucho sentido utilizarla en las nuevas funcionalidades. Hay dos campos: *versión afectada,* que es la versión donde fue detectado el error del *software,* y el campo *versión correctora,* que se usa para especificar en qué versión fue corregido el error. El campo versión correctora también se utilizará cuando se incluye una nueva funcionalidad al proyecto, ligándolo a una versión.

Responsable: Es la persona responsable de resolver la incidencia y encargada de reportar y explicar todo lo relacionado con esta incidencia. En las reuniones diarias de Scrum, el responsable es el que dará las explicaciones sobre la evolución, problemas o demás situaciones existentes, relacionados con la incidencia en la que está trabajando.

Informador: Es la persona que registra la incidencia en el sistema. En la mayoría de los casos será el *Scrum master* el que registre todas las incidencias. Se puede dar el caso de que los propios usuarios puedan ser los que abran las incidencias, desde el portal de usuarios.

Descripción: Es la descripción más detallada de lo que solicita el usuario. Será necesario tener cuidado de no poner las tareas que hay que hacer en la incidencia dentro del campo descripción y poner dichas tareas como subtareas.

Estimar: Es un campo muy importante a la hora de crear las incidencias en JIRA. Dicho campo se utiliza para poder planificar los tiempos necesarios para finalizar cada una de las incidencias que habrá que afrontar en el proyecto. A la hora de gestionar el proyecto es esencial saber lo que queda para terminar, lo que se desvía y lo que se avanza en el *sprint*.

Según la metodología Scrum, dicha estimación se realiza mediante la asignación de puntos a la historia, pero, bajo nuestro punto de vista, la estimación en tiempos reales es más clara y se ajusta más a las planificaciones temporales que se hacen en la gestión de los proyectos, ya que los puntos de historia serían más operativos si existiera una experiencia previa de lo que tarda el equipo en realizar los trabajos. En el presente caso de análisis, al iniciarse el equipo en la metodología, se recomienda utilizar los tiempos reales.

Por defecto, JIRA utiliza la estimación utilizando puntos de historia, pero será necesario modificar la configuración para utilizar estimaciones en tiempos reales. En el siguiente apartado se explica cómo hacer dicha configuración y cómo estimar las incidencias creadas en JIRA.

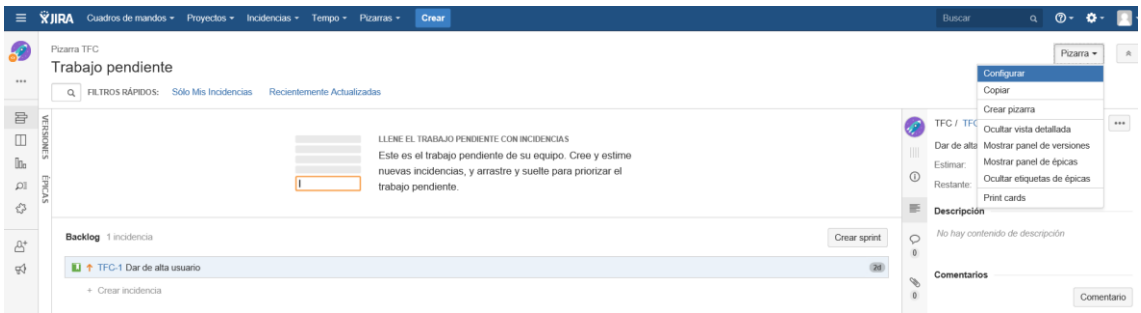
3.2.5. Estimación de las incidencias en JIRA

Tal y como se ha comentado en el punto anterior, cuando se crean las incidencias en la aplicación es necesario hacer estimaciones temporales de estas para poder gestionar adecuadamente el proyecto.

Por defecto, en JIRA las estimaciones están definidas mediante la asignación de puntos en la historia, pero en el caso que se comenta se va a configurar la estimación mediante la asignación de tiempos reales y la utilización de este tipo de estimaciones a lo largo de todo el proyecto. Dado que la configuración de la estimación en modo temporal es algo que no se ve claramente, se indicará cómo se habilita esta opción en JIRA y será explicada a continuación.

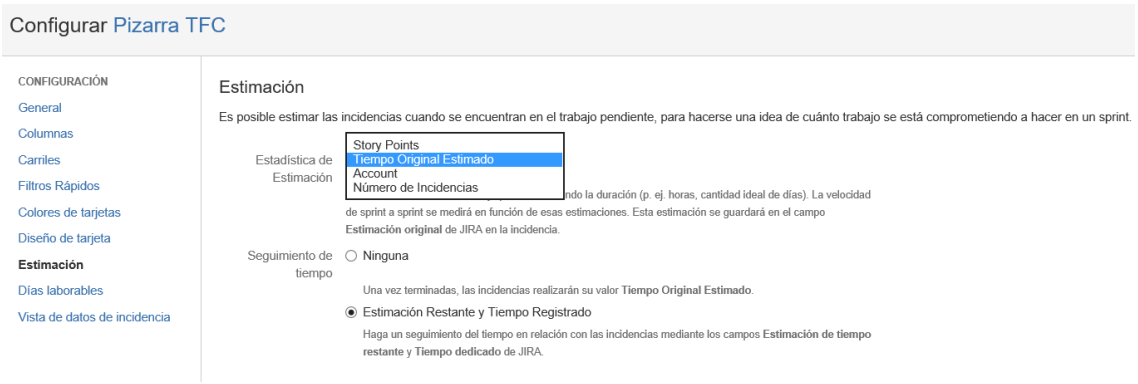
Una vez que se ha creado el proyecto dentro de JIRA, hay que acceder al mismo y en la visión normal de trabajo ir al menú de la derecha llamado *Pizarra* y luego pinchar en configurar.

Pantalla de *backlog* de JIRA



Una vez que se ha clicado en configurar, en las opciones de la izquierda se puede ver la opción *Estimación* y desde esa misma opción habrá que configurar para que la estimación, en lugar de ser mediante puntos de historia, sea mediante tiempos reales.

Pantalla para cambiar el tipo de estimación de incidencias

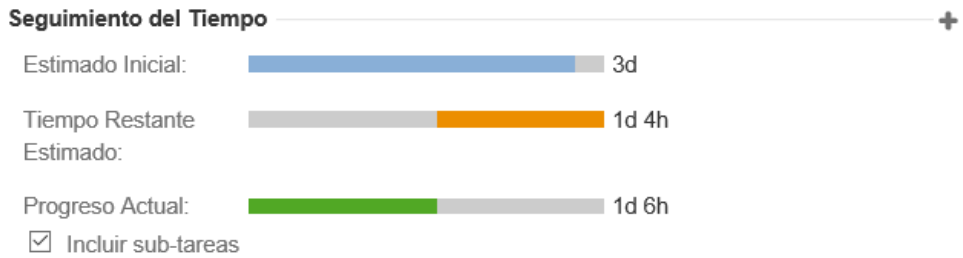


La *Estadística de la estimación* se pondrá para que tenga en cuenta el tiempo original estimado y para que en el *seguimiento del tiempo* se vaya calculando automáticamente, de tal forma que el tiempo de *Estimación restante* se calcule utilizando las variables de la estimación original y las horas que se vayan registrando de cada una de las incidencias en las que se esté trabajando.

Por lo tanto, la fórmula que utiliza JIRA para calcular el tiempo de estimación restante será *Tiempo de estimación restante* (TER) es igual al *Tiempo de estimación original* (TEO) menos el sumatorio de los tiempos registrados en la incidencia (TR).

Como ejemplo, se muestra cómo se obtienen los valores y cómo estos se pueden ver gráficamente.

Gráfico de evolución temporal de incidencia



Como se puede observar en el ejemplo, se ha realizado una estimación original de una incidencia de tres días. El tiempo de trabajo realizado es de un día y seis horas (progreso actual) y queda un tiempo estimado de finalización de un día y cuatro horas (tiempo restante estimado).

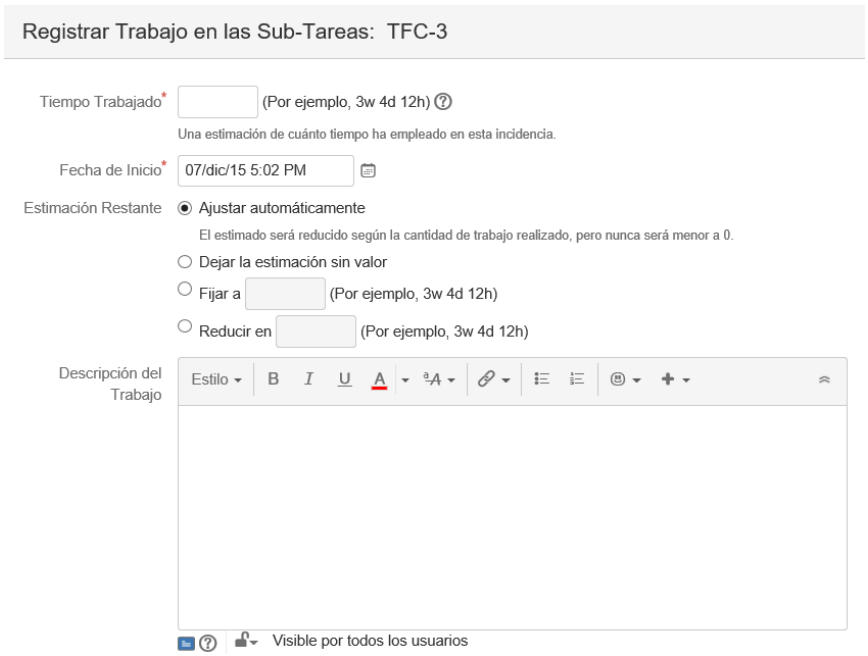
A la hora de trabajar con tiempos reales se dispondrá de una precisión más exacta de lo que queda para finalizar cada una de las tareas, siempre y cuando se estime correctamente. Se recomienda que toda estimación sea realizada por el equipo Scrum; si no, estas estimaciones no serán realistas ni serán asumidas por los miembros del equipo de desarrollo.

Para ir incrementando el progreso actual, los miembros del equipo han de ir registrando su actividad diaria en JIRA; de esta forma irá aumentando el progreso y se irán recalculando los tiempos restantes estimados.

El equipo podrá registrar los tiempos directamente en la incidencias donde está trabajando, o bien podrá registrar sobre las subtareas. Se recomienda detallar el registro del tiempo a nivel de subtarea, para así saber con más exactitud qué trabajo es el que más tiempo está llevando y, por lo tanto, supone una mayor dificultad. Si se imputan los tiempos sobre las subtareas, estas serán sumadas a nivel de incidencia padre, haciendo el sumatorio de los tiempos de cada una de las subtareas.

Desde la incidencia o desde la subtarea existe una opción para registrar los tiempos de trabajo, que se denomina «Registrar horas de trabajo» y se pueden añadir semanas (w), días (d) y horas (h).

Pantalla de registro de horas



Como recomendación, respecto al registro de horas, es necesario matizar que todos los miembros del equipo han de registrar las horas efectivas de trabajo, no se han registran vacaciones, ni los cursos de formación, ni los descansos, etcétera. Si se registran todas estas tareas no se tendrá una visión real de lo que ha costado hacer un producto, ya que realmente lo interesante son las horas efectivas de desarrollo y conocer el rendimiento del equipo.

Por otro lado, también se recomienda que el registro de horas sea diario, pues de esta forma se dispondrá de la evolución y los avances en tiempo real. Si se dejan de imputar las horas al final de la semana, se pierde claridad respecto al proyecto, se difumina la realidad y seguramente dicha imputación dejará de ser realista porque nadie se puede acordar un viernes lo que ha realizado el lunes.

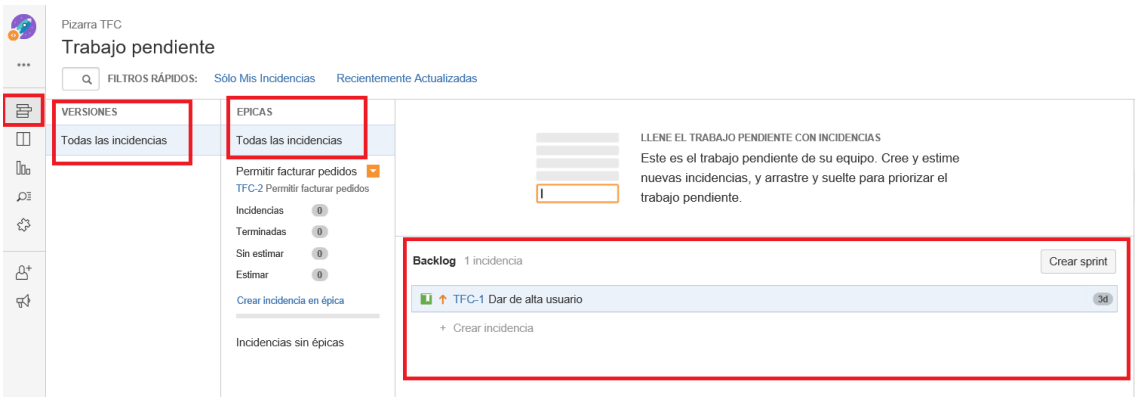
3.3. El *Product Backlog* en JIRA

En el apartado 2.3.1 se definía el *Product Backlog* como la base de Scrum y como parte de los artefactos que se utilizaban en la metodología; en resumen, era una lista ordenada de requerimientos y necesidades que hay que desarrollar para que en cierta forma vayan aportando valor según se vayan acometiendo.

En JIRA, cuando se crea el proyecto automáticamente, la aplicación crea el *backlog* que es como en JIRA se llama al *Product Backlog*. La forma de poder gestionar el *backlog* es desde la opción *«Trabajo pendiente»*. Desde este apartado permitirá ir creando nuevos requerimientos; permitirá modificar los requerimientos ya creados y obtener así la lista de requerimientos para realizar la planificación de los nuevos *sprints*.

Desde esta misma opción de pantalla, la aplicación también va permitir gestionar las versiones del proyecto y gestionar las épicas creadas, asignando o desasignado incidencias tanto a la épica como a la versión.

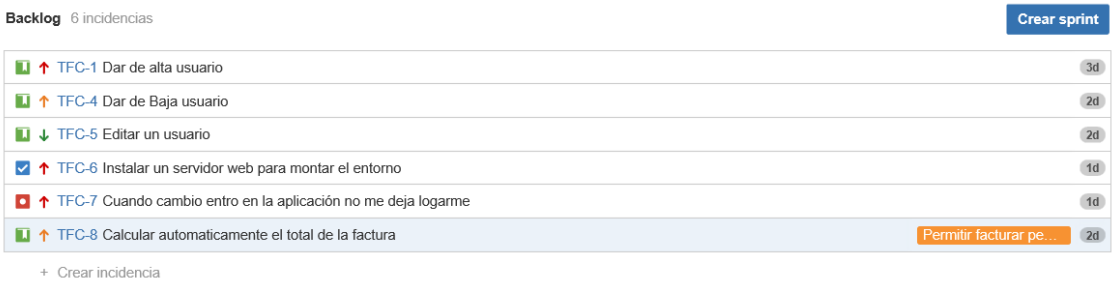
Pantalla Trabajo pendiente donde poder gestionar el backlog



Dentro del *backlog,* como se ha comentado con anterioridad, podrá haber incidencias de diferentes tipos y todas ellas han de planificarse según la prioridad de las mismas. En dicho *backlog* puede haber historias de usuario, tareas, errores, pero este no puede contener épicas, ya que son las historias incluidas en una épica las que aparecen en el *backlog*.

En el *backlog* de JIRA se dispone de información muy importante que ayudará a hacer el seguimiento del proyecto y a planificar el trabajo. Si se observa la hoja Excel anterior, y que se proponía utilizar para el seguimiento del proyecto, se pueden observar muchas similitudes.

Ejemplo de *backlog* en JIRA



En el siguiente ejemplo se observa un posible *backlog* del proyecto analizado en el presente trabajo y pueden observarse diferentes campos que se irán utilizando continuamente a lo largo del tiempo.

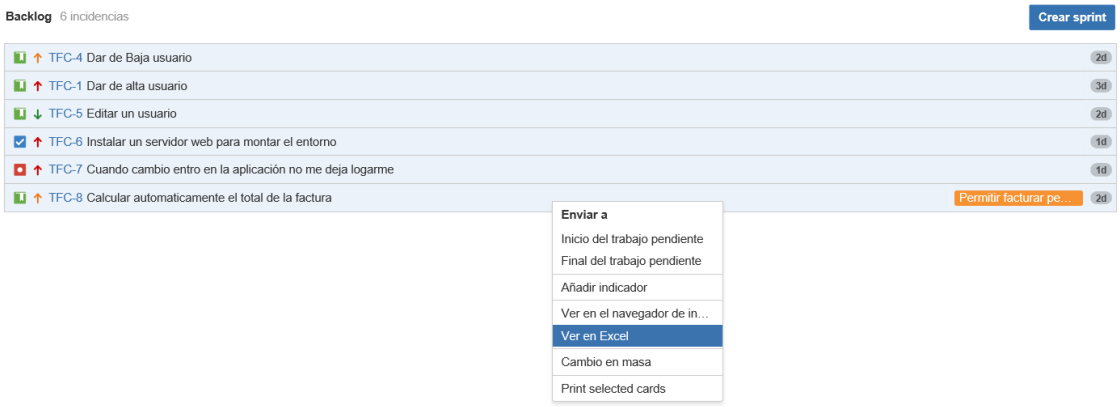
En el primer campo aparece un icono que ayuda a conocer qué tipo de incidencias (se habían especificado anteriormente los diferentes tipos de iconos según el tipo de incidencia). En segundo lugar, aparece el icono que muestra la prioridad de la incidencia (también se han comentado anteriormente qué tipos de prioridades existen en JIRA). Seguidamente, están el identificador y el resumen de la incidencia.

Respecto al tema de la estimación, se puede encontrar en una última columna en la que se pueden observar los días necesarios para realizar cada uno de los trabajos que se listan en el *backlog*.

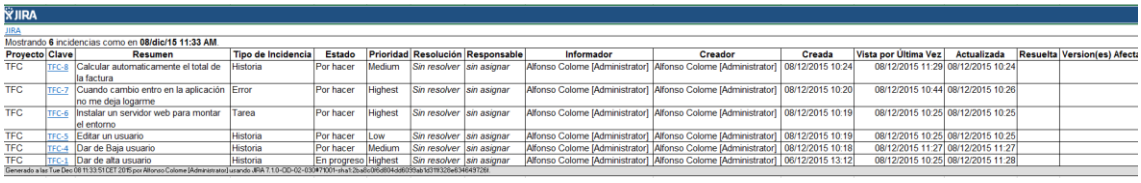
Por último, puede observarse que, en la última de las incidencias que se muestran en el ejemplo, aparece una etiqueta en naranja; esto quiere decir que dicha incidencia pertenece a una épica del proyecto y aparece el nombre de la misma.

El listado de *backlog* es el que hace sustituir a la hoja Excel mencionada con anterioridad, por lo que en todo evento Scrum se ha de obtener un listado del *backlog* directamente extraído de JIRA y utilizar el mismo para la gestión del trabajo que tiene que realizar el equipo y mostrar los avances al dueño del producto. Para obtener dicho listado desde el propio *backlog*, solamente hay que seleccionar todas las incidencias que existen y, con el botón derecho, indicar «Ver en Excel»; de esta forma se obtiene el *backlog* de JIRA en el mismo formato en el que estaba con anterioridad al introducirse en Scrum.

Ejemplo de cómo exportar el *backlog* a Excel



Hoja Excel extraída en JIRA



Proyecto	Clave	Resumen	Tipo de Incidencia	Estado	Prioridad	Resolución	Responsable	Informador	Creador	Creada	Vista por Última Vez	Actualizada	Resuelta	Version(es) Afecta
TFC	TFC-8	Calcular automaticamente el total de la factura	Historia	Por hacer	Medium	*Sin resolver*	*sin asignar*	Alfonso Colome [Administrator]	Alfonso Colome [Administrator]	08/12/2015 10:24	08/12/2015 11:29	08/12/2015 10:24		
TFC	TFC-7	Cuando cambio entro en la aplicación no me deja logarme	Error	Por hacer	Highest	*Sin resolver*	*sin asignar*	Alfonso Colome [Administrator]	Alfonso Colome [Administrator]	08/12/2015 10:20	08/12/2015 10:44	08/12/2015 10:26		
TFC	TFC-6	Instalar un servidor web para montar el entorno	Tarea	Por hacer	Highest	*Sin resolver*	*sin asignar*	Alfonso Colome [Administrator]	Alfonso Colome [Administrator]	08/12/2015 10:19	08/12/2015 10:25	08/12/2015 10:25		
TFC	TFC-5	Editar un usuario	Historia	Por hacer	Low	*Sin resolver*	*sin asignar*	Alfonso Colome [Administrator]	Alfonso Colome [Administrator]	08/12/2015 10:19	08/12/2015 10:25	08/12/2015 10:25		
TFC	TFC-4	Dar de Baja usuario	Historia	Por hacer	Medium	*Sin resolver*	*sin asignar*	Alfonso Colome [Administrator]	Alfonso Colome [Administrator]	08/12/2015 10:18	08/12/2015 11:27	08/12/2015 11:27		
TFC	TFC-1	Dar de alta usuario	Historia	En progreso	Highest	*Sin resolver*	*sin asignar*	Alfonso Colome [Administrator]	Alfonso Colome [Administrator]	06/12/2015 13:12	08/12/2015 10:25	08/12/2015 11:28		

3.3.1. Priorizar el *Product Backlog* en JIRA

En la definición del producto *backlog,* se comentaba que era un listado de requerimientos o necesidades que iban aportando valor al proyecto, pero no se ha profundizado en la necesidad de priorizar cada uno de los trabajos.

En este punto se destacará la necesidad de que todo trabajo incluido en el *backlog* ha de tener una prioridad para que el equipo Scrum sepa la importancia que da el dueño del producto a cada una de las incidencias que se tienen que realizar. Por ello, se detallará cómo debe priorizarse el *backlog* para saber qué incidencias se habrán de incluir en cada una de las fases del trabajo.

Por el momento, cuando se registraban las incidencias, únicamente se especificaba la prioridad de la misma indicando si era muy alta, alta, media, etcétera, pero esta priorización carece de efectividad cuando el listado de *backlog* es muy grande. Por ello, se indicará ahora cómo se priorizan las incidencias con un método más claro y adecuado.

La técnica que se recomienda utilizar es la técnica de MoSCoW:

> *El método MoSCoW es una técnica de priorización de requisitos basada en el hecho de que aunque todos los requisitos se consideren importantes es fundamental destacar aquellos que permiten darle un mayor valor al sistema, lo que permite enfocar los trabajos de manera más eficiente. (Jummp, 2013).*

Existe una clara diferencia de esta técnica respecto a otras, ya que en esta escala los significados son claros y el dueño del producto sabe exactamente lo que conlleva hacer una priorización u otra.

A continuación se identifican los diferentes tipos de priorización utilizados en la técnica MoSCoW en una tabla:

Tabla de priorización MoSCoW

Valor	Significado
M	Must - Debe incluirse
S	Should - Debería incluirse
C	Could - Podría incluirse
W	Won't - No incluir

Valor	Descripción
M	Requisito que tiene que estar implementado en la versión final del producto para que la misma pueda ser considerada un éxito.
S	Requisito de alta prioridad que en la medida de lo posible debería ser incluido en la solución final, pero que, llegado el momento y si fuera necesario, podría ser prescindible si hubiera alguna causa que lo justificara.
C	Requisito deseable pero no necesario; se implementaría si hubiera posibilidades presupuestarias y temporales.
W	Hace referencia a requisitos que están descartados de momento, pero que en un futuro podrían tenerse de nuevo en cuenta y reclasificarse en una de las categorías anteriores.

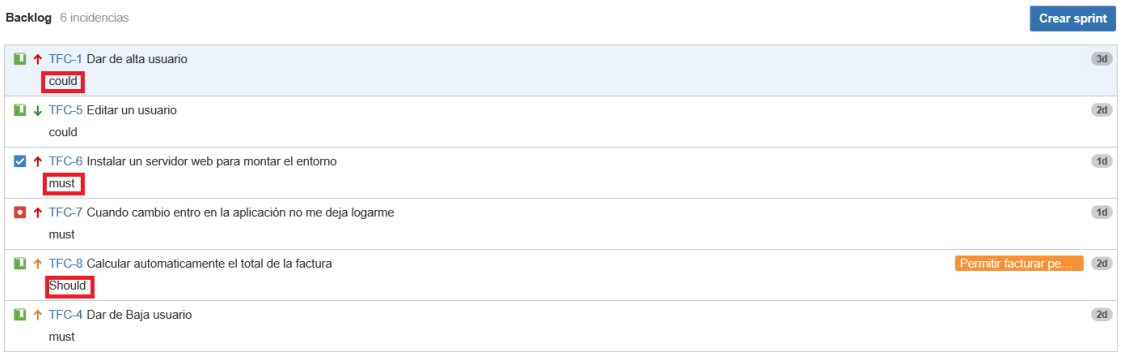
Cabe destacar que esta priorización puede ir variando a la largo del proyecto, por lo que las incidencias van a cambiar sus prioridades a lo largo del tiempo.

Para utilizar la regla MoSCoW en JIRA se usará un campo nuevo dentro de las incidencias: las etiquetas. Este campo permitirá clasificar las incidencias por categorías y realizar búsquedas de una forma más rápida, además de utilizar filtros, de tal forma que ayudará a identificar las incidencias más prioritarias facilitando la organización del trabajo.

Por defecto en JIRA en la visión que se muestra del *backlog*, no existe el campo etiqueta, pero desde la pizarra puede configurarse e incluirse dicho campo, de tal forma que de una mirada puede conocerse la priorización que se ha realizado de cada incidencia. Para configurar este campo, hay que ir al botón «Pizarra –> Configuración» y dentro de las opciones pinchar en «Diseño de tarjeta» e incluir tanto en «Trabajo pendiente» como en «Sprints activos» el campo «Etiquetas».

Pantalla de configuración del campo Etiquetas en el *backlog*

Visión del campo Etiquetas en el *backlog*



3.4. El *sprint* en JIRA

Tal y como se comentó en el apartado 2.3.1, el *sprint* en Scrum se define como el nombre que recibe el conjunto que historias que se desarrollarán en un periodo de tiempo no superior a un mes. Para dar por finalizado el *sprint* habrá que hacer una entrega o prototipo completamente funcional y este, cuando sea necesario, se decidirá incorporar en una nueva versión o *release*.

El JIRA habrá que hacer una ordenación del trabajo para poder empezar a desarrollar las funcionalidades, por esos se ejecutarán una serie de pasos para asignar el trabajo y previamente dar de alta el *sprint* en la herramienta.

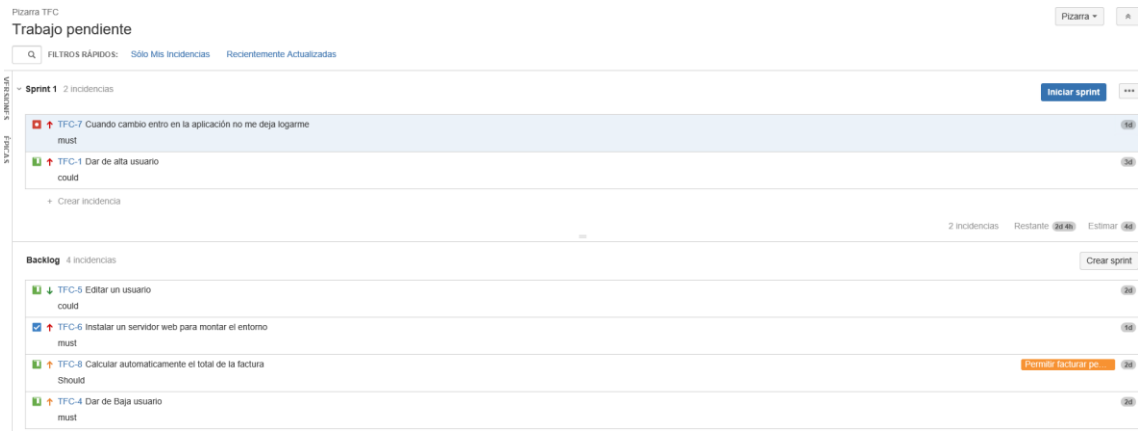
En primer lugar, como se ha comentado con anterioridad, habrá que saber qué incidencia se debe incorporar al *sprint;* para ello, se convocará la primera reunión de planificación de *sprint* con el fin de identificar qué incidencias se incorporarán. Antes de ello, habrá que tener todas las incidencias creadas en JIRA priorizadas mediante el método MoSCoW hablando previamente con el dueño del producto y, por supuesto, estimadas temporalmente, tarea que ha de realizar el equipo Scrum. Sin todas estas tareas previas no sería posible realizar la planificación del *sprint*, porque no estarían todas las variables del proyecto actualizadas.

Una vez que se tenga todo el proyecto actualizado en JIRA, se convocará la *reunión de planificación* del *sprint*, donde se seleccionarán las incidencias que se van a incorporar.

Para la reunión habrá que disponer de una serie de documentación que han de revisar previamente todos los asistentes a la reunión; esta documentación será una hoja Excel con todo el *backlog* pendiente de realizar. Dicha hoja Excel se ha comentado en apartados anteriores y deberá ser extraída de JIRA.

En la reunión es interesante utilizar un PC con acceso a JIRA y, a ser posible, proyectado en una pantalla, para hacer el seguimiento e ir actualizando las decisiones directamente en la herramienta. En la reunión o previamente a la reunión se crea el *sprint;* para ello, hay que ir al *backlog* del proyecto, pinchar en «Crear sprint» y de forma inmediata JIRA dividirá la pantalla. En la parte de arriba se irán arrastrando las incidencias que se vayan a incorporar al *sprint,* y en la parte de abajo estará la relación de incidencias del *backlog*.

Pantalla de planificación del *sprint*



Tras haber incorporado todas las incidencias que vayan a incluirse en el *sprint* se puede iniciar este. Para ello, en la misma pantalla, debe clicarse el botón en el que pone *«Iniciar sprint».* Tras activar el botón pedirá la duración de la entrega y, gracias a la estimación temporal hecha, será posible saber cuánto tiempo será necesario. Por defecto JIRA propone que la duración sea de una, dos, tres o cuatro semanas, pero excepcionalmente se podrá poner más de cuatro semanas (opción que no se recomienda).

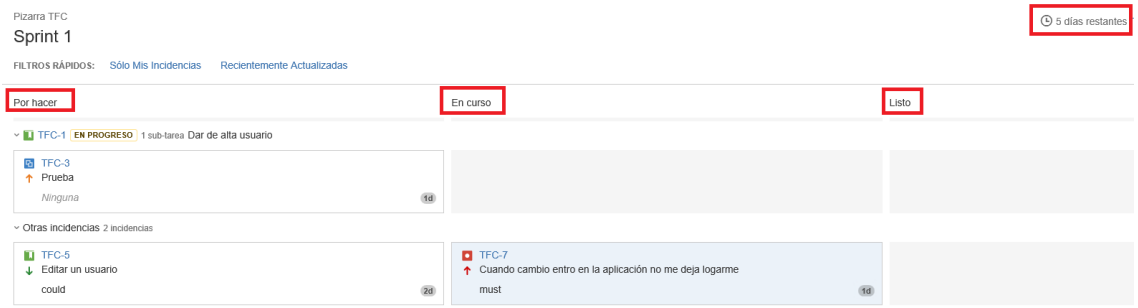
Para poder dar por finalizada la reunión de planificación, tendrán que estar resueltas todas las dudas que hayan podido surgir respecto a las incidencias que se quieren incluir en el *sprint,* y tras la reunión el equipo debería ser capaz de ponerse a trabajar sin ningún inconveniente.

Una vez finalizada la reunión de planificación, seguidamente o a primera hora del día siguiente, se realizará la primera reunión de trabajo o *reunión diaria de Scrum*. En esta primera reunión se asignarán las incidencias a cada uno de los miembros del equipo técnico y se pondrá en marcha la pizarra de Scrum con los pósits, aunque en JIRA ahora ya aparecerá la pizarra Scrum dentro de la opción *sprints* activos.

En la pizarra Scrum aparecerán tanto las incidencias como las subtareas y también habrá varias columnas que definen el estado de cada una de las incidencias o subtareas: «Por hacer», «En curso», «Listo». Dichos estados pueden variar y se podrán añadir otros estados nuevos o quitar existentes.

Otro dato importante que aparece en la pizarra es el número de días que restan para cerrar el *sprint* y se puede observar arriba a la derecha. A partir de este momento el equipo técnico es el responsable del *sprint* y tiene que hacer todo lo posible para llegar a las entregas comprometidas.

Pizarra Scrum en JIRA



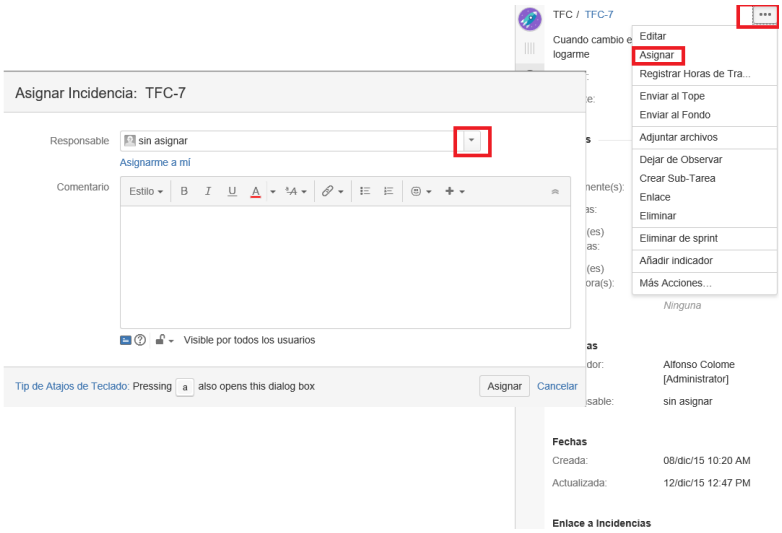
3.4.1. Asignación de las tareas

Anteriormente se ha comentado que en la primera *reunión diaria de Scrum* es donde se asignan los trabajos a los miembros del equipo y para ello hay que tener en cuenta una serie de aspectos relacionados con esta asignación.

En primer lugar, si una incidencia se tiene que realizar entre varias personas habrá que dividir la misma en varias subtareas; de esta forma se sabrá qué tareas va hacer cada uno de los miembros del equipo, y estas subtareas se asignarán a cada una de las personas.

Si una incidencia tiene varias subtareas y estas van ser realizadas por una única persona, se podrá asignar la incidencia y no las subtareas; luego el propio técnico al que se le ha asignado la incidencia, irá asignándose él mismo cada una de las subtareas e irá avanzando por cada una de ellas.

Pantalla de asignación de incidencias



Es importante recordar que una vez que se comience el trabajo los miembros del equipo deberán de imputar el tiempo de trabajo dedicado a cada una de las incidencias o subtareas de forma diaria y lo más precisa posible.

Las tareas asignadas serán revisadas en las reuniones diarias de Scrum, donde cada miembro del equipo explicará los avances del trabajo, inconvenientes encontrados y el trabajo que va hacer hasta la siguiente reunión, y se resolverán las dudas con la ayuda de todo el equipo. Dichos avances se podrán incluir como comentarios en las incidencias o subtareas para tener un registro de los avances, y esta información deberá ser registrada por cada responsable de la tarea.

El *Scrum master* deberá dirigir la reunión diaria, revisar los trabajos de cada miembro del equipo, que se imputan las horas correctamente, que se actualizan los estados de las tareas, e ira ayudando a resolver los inconvenientes que se vayan encontrando.

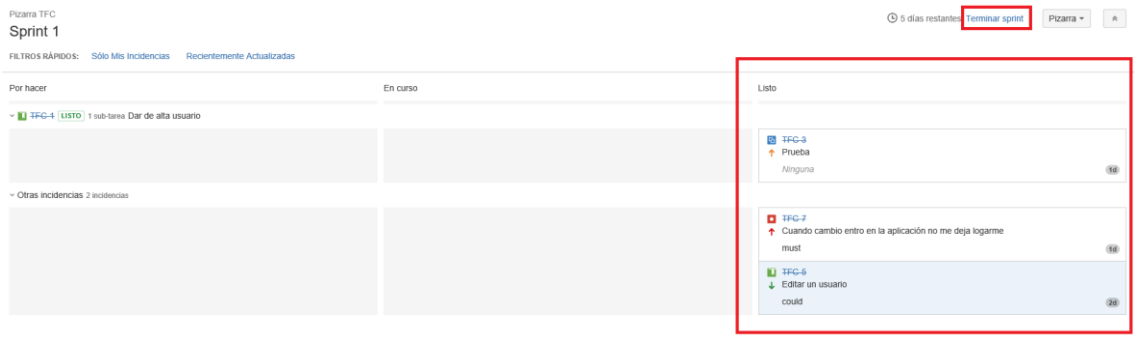
Para finalizar, tras varias reuniones diarias de Scrum y la finalización de cada una de las incidencias, llegará el momento de la entrega y del cierre del *sprint*.

Para cerrar el *sprint* deberá de disponerse de un entregable, que deberá haber probado y tendrá que ser totalmente funcional. Si alguna de las incidencias no se ha terminado a tiempo no se deberá incorporar ninguna de las subtareas al prototipo funcional, y se dejará para el siguiente *sprint*.

Tras haber validado el entregable funcional por el equipo técnico y el equipo de test, se deberá convocar la reunión de *revisión del sprint,* donde se validará la entrega por parte del dueño del producto. Se deberá explicar qué funcionalidades se han resuelto en el *sprint*, qué no ha dado tiempo a resolver y los motivos. Se decidirá si se crea versión nueva para desplegar y se planificará el despliegue o, por el contrario, se acumularán *sprints*, para decidir en la siguiente reunión de revisión de *sprint* si se crea versión o se sigue acumulando.

Tras esta reunión será posible terminar el *sprint,* y las incidencias pendientes pasarán al *backlog,* si no hay *sprint* nuevo, o pasarán al nuevo *sprint,* si ya está creado. Tras este evento se convocará la nueva reunión de planificación del nuevo *sprint* y la *reunión de retrospectiva* (solo para el equipo Scrum, donde se revisará a nivel técnico y a nivel de organización del grupo el *sprint* entregado).

Pantalla con incidencias listas y terminar *sprint*



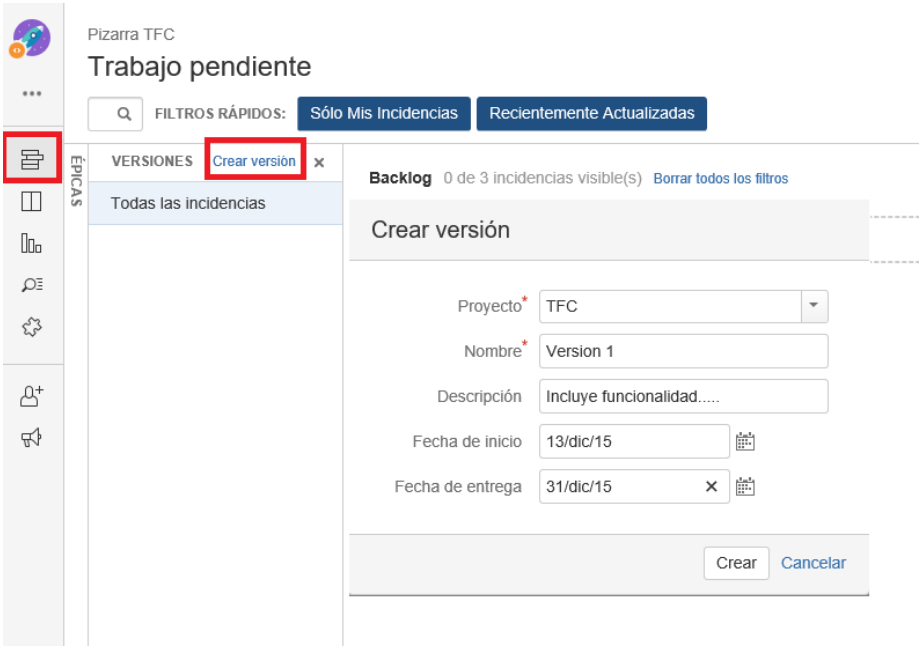
3.4.2. Creación de versiones

Como se ha señalado anteriormente, en la reunión de revisión del *sprint*, se decide si el *sprint* se incorpora a una nueva versión o bien se acumula para que luego hacer una versión más grande. (Como recomendación, no se deberían hacer despliegues muy grandes de nuevas funcionalidades, ya que pueden surgir varios *bugs,* y cuanta más funcionalidad, más *bugs* se pueden producir. Por esto se recomienda por cada uno o dos *sprint* hacer un despliegue de versión).

En JIRA cuando se utiliza Scrum las versiones se gestionan a partir de los *sprints;* para ello, deben darse una serie de pasos que se explicarán a continuación, y ayudarán a asignar todos las incidencias de un *sprint* a una versión.

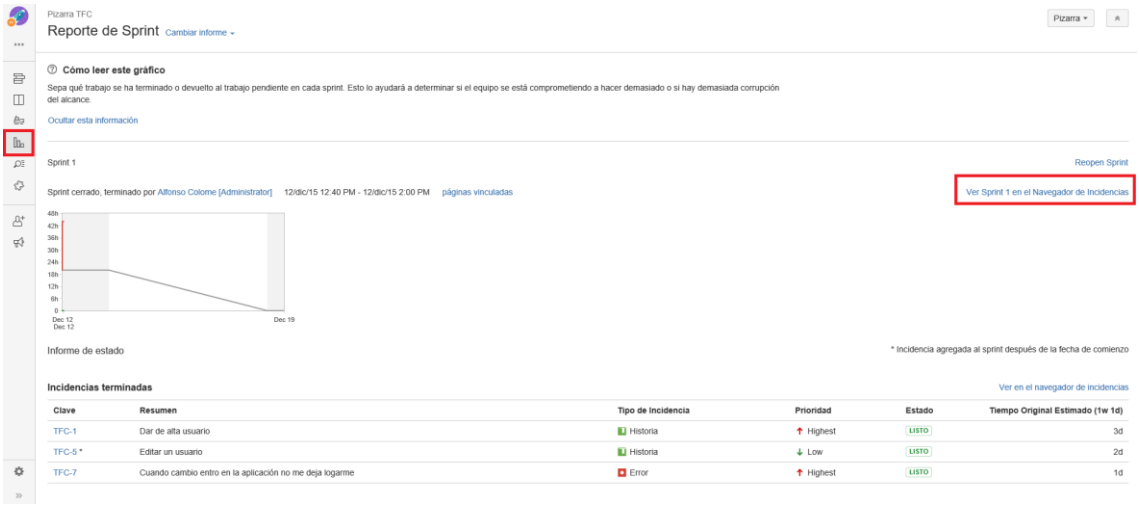
1. Se crea la versión en JIRA, desde la pantalla de «Trabajo pendiente», pestaña «Versiones».

Pantalla para crear versión



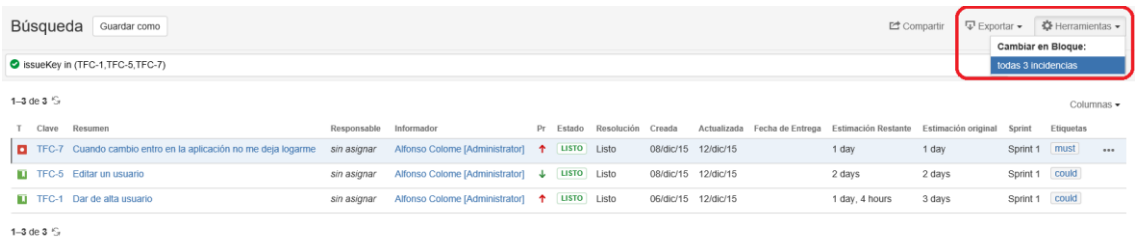
2. Se introduce el nombre de la versión, una descripción de la versión (por ejemplo: un resumen de funcionalidades incluidas), la fecha de inicio, que sería cuando se decide desplegar versión, y la fecha entrega, que es la fecha del despliegue o cuando se pasará a producción.
3. Seguidamente, hay que asignar todas las incidencias de los *sprints* que van en la versión y, para ello, se hará una asignación en masa de incidencias.
4. En la opción informes puede verse información de los *sprints* cerrados y posteriormente seleccionar la opción de ver el *sprint* en el navegador de incidencias.

Pantalla de revisión de *sprints* cerrados



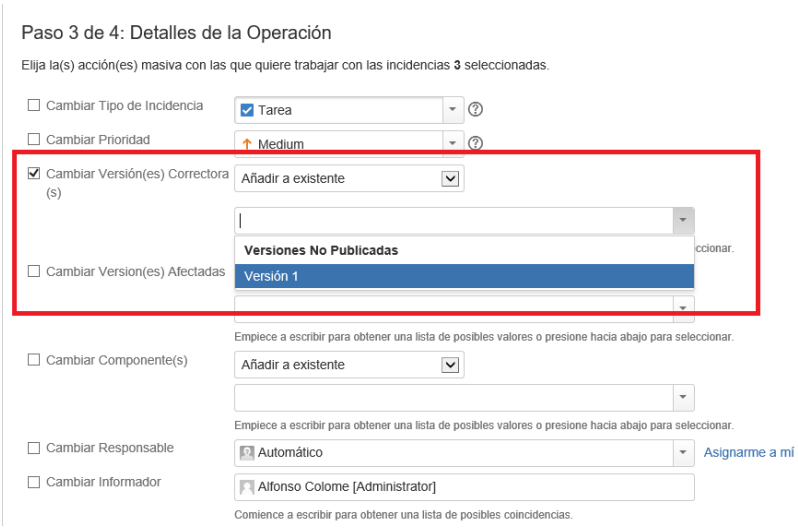
5. Ahora se hace la asignación en bloque de la versión, y para ello se clica la opción herramientas, todas las incidencias. (En este caso solo hay tres).

Pantalla de incidencias



6. Se seleccionan todas las incidencias y se pincha en «siguiente».
7. Posteriormente, se marca la opción «Editar incidencias» y se clica «siguiente».
8. Se marca la versión correctora, se introduce la versión donde se quieran incluir las incidencias y se confirma.

Pantalla de selección de versión correctora



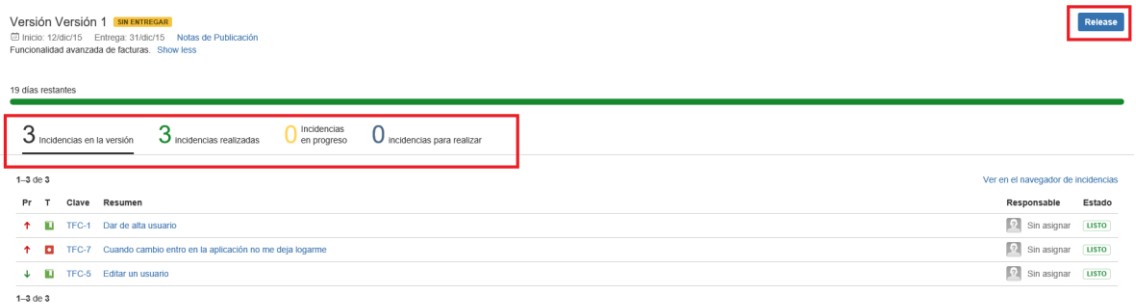
Estos pasos descritos deben hacerse tantas veces como *sprints* se tengan, y asignar todas las incidencias de aquellos *sprints* que se desea incorporar a la versión.

Tras realizar esta operativa con todos los *sprints*, se puede ver ya en JIRA una nueva opción donde aparecen todas las incidencias incluidas en una versión. Para acceder a dicha visualización se observa que aparece un icono nuevo en la pantalla de proyectos, que permite consultar todas las versiones del proyecto y saber si dichas versiones están o no están publicadas (es decir, puestas en producción).

Pantalla de visualización de las versiones del proyecto

Para ver el contenido de las versiones se accede a una de ellas y aparecen todas las incidencias que han sido incluidas además del detalle de si las mismas están totalmente finalizadas o aún hay alguna incidencia «En curso».

Pantalla de revisión de versión



Una vez que se ha realizado el despliegue de la versión y esta ya esté en producción, se podrá hacer la publicación de la mismas en JIRA pinchando en la opción *«Release»,* indicando en qué fecha finalmente ha sido puesta en producción y fueron liberadas las nuevas funcionalidades. Ahora la versión aparecerá como publicada.

3.5. Métricas del proyecto objeto de estudio

Hasta este momento es posible decir que ya está disponible la primera versión de nuestro proyecto con JIRA, pues se ha realizado la gestión completa utilizando metodología Scrum y gestionando todo con la herramienta JIRA. Sin embargo, para ser óptimos en el proceso, no solo vale hacer una estrategia de gestión utilizando la metodología, sino que ahora de una forma objetiva se deberán observar y analizar los resultados de la eficiencia del equipo y, para ello, habrá que analizar una serie de indicadores facilitados por JIRA y que son imprescindibles para saber que realmente la aplicación de la metodología es totalmente efectiva.

3.5.1. Evolución del trabajo en el equipo Scrum

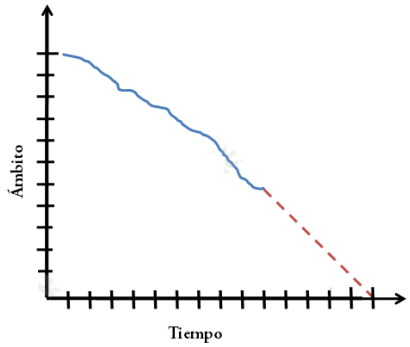
Para saber si el equipo lleva una evolución correcta respecto a las entregas hay que medir la evolución del trabajo, dicha evolución se calcula a partir de los trabajos que el equipo ha de realizar en el *sprint* y cómo van evolucionando las mismas a lo largo del *sprint*.

Es cierto que algunos autores requieren hacer el seguimiento del equipo a partir de puntos de la historia, pero estos valores son totalmente técnicos y no funcionales, ya que para el dueño del producto lo que interesa es el número de incidencias que se van incorporando y no la dificultad de las mismas.

Por esto, el equipo tendrá la referencia de las incidencias que se van entregando respecto a las fechas comprometidas de entrega y habrá una gráfica que mostrará dicha información de una manera visual. Para este seguimiento se utilizará la «Gráfica de trabajo» dentro de la opción «Informes del proyecto», y consistirá en una gráfica donde el eje vertical refleje el número de incidencias y el eje horizontal, el tiempo.

Con este gráfico podrán detectarse las desviaciones y prever cuántas incidencias será posible acometer en el siguiente *sprint*. Ayuda a saber si hay tiempo de llegar a las entregas según el trabajo que se vaya realizando.

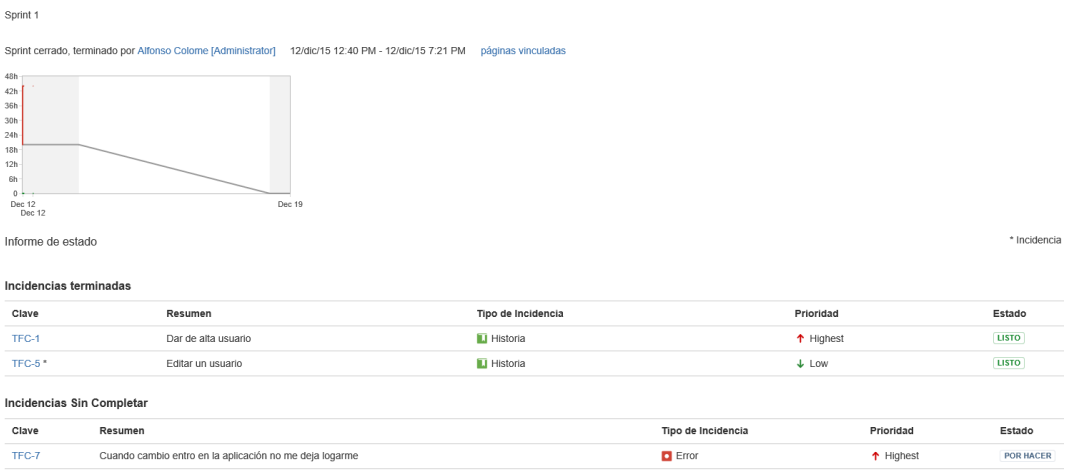
Gráfica de trabajo restante-burndown



3.5.2. Evolución de las incidencias completadas

En JIRA hay otro indicador que irá indicando si se completan en cada uno de los *sprints* las incidencias comprometidas, por lo que mostrará aquellas incidencias que sí se han completado y qué incidencias no se han completado en el momento de cerrar el *sprint*. Para obtener esta información se puede utilizar el informe «Reporte de *sprint*».

Informe de incidencias completas en *sprint*



Sprint 1

Sprint cerrado, terminado por Alfonso Colome [Administrator] 12/dic/15 12:40 PM - 12/dic/15 7:21 PM páginas vinculadas

48h
42h
36h
30h
24h
18h
12h
6h
0
Dec 12
Dec 12
Dec 19

Informe de estado

* Incidencia

Incidencias terminadas

Clave	Resumen	Tipo de Incidencia	Prioridad	Estado
TFC-1	Dar de alta usuario	Historia	Highest	LISTO
TFC-5 *	Editar un usuario	Historia	Low	LISTO

Incidencias Sin Completar

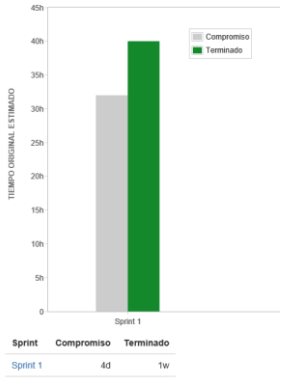
Clave	Resumen	Tipo de Incidencia	Prioridad	Estado
TFC-7	Cuando cambio entro en la aplicación no me deja logarme	Error	Highest	POR HACER

3.5.3. Velocidad respecto a lo *comprometido*

Existe otro informe en JIRA que ayudará de una forma visual a saber, respecto a todo el trabajo y tiempos estimados de los *sprints*, cuánto trabajo se ha terminado según lo previsto y, así, saber si se están cumpliendo objetivos con el cliente. Este gráfico se llama «Gráfica de velocidad» y es un gráfico de barras donde una barra es la cantidad de tiempo estimado y otra barra indica la cantidad de tiempo real utilizado.

En el siguiente gráfico se puede observar que había una estimación del *sprint* con un tiempo de finalización de treinta horas y se ha terminado el trabajo en cuarenta horas, por lo que hay una desviación de diez horas.

Gráfico de velocidad del *sprint*

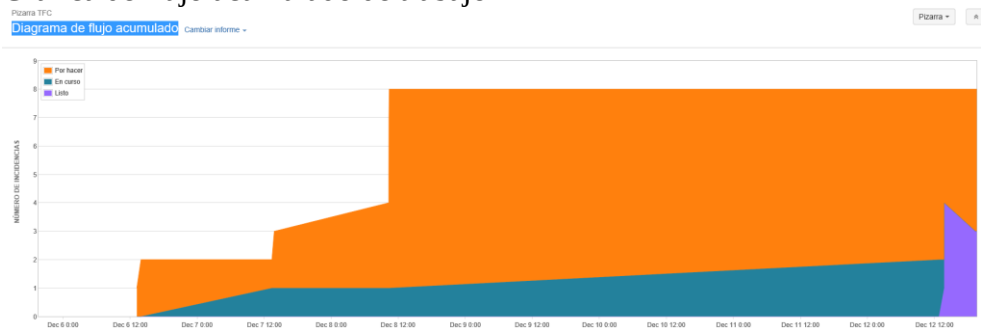


Sprint	Compromiso	Terminado
Sprint 1	4d	1w

3.5.4. Diagrama de flujo acumulado

Este gráfico en JIRA ayuda a hacer un seguimiento del trabajo que queda por hacer teniendo en cuenta todo el *backlog* del proyecto. En este gráfico se identifica la cantidad de trabajo que queda pendiente, la cantidad de trabajo terminado y la cantidad de trabajo en curso. En el eje vertical está el número de incidencias del *backlog,* y en la gráfica horizontal se muestran las fechas de progreso del proyecto.

Gráfica de flujo acumulado de trabajo



3.5.5. Otros informes y gráficos

Aparte de estos indicadores e informes importantes para gestionar el proyecto, JIRA aporta muchos otros informes que pueden irse utilizando según corresponda, pero en estos apartados se han explicado los más utilizados y los que más información aportan para gestionar el proyecto aquí analizado.

Otros informes existentes son:

Timesheet Report: Es un informe relacionado con el complemento que se recomendaba instalar en JIRA, y que aporta de una manera más clara la dedicación de cada uno de los miembros del equipo en cada una de las tareas del proyecto. Indica el porcentaje total de tiempo de trabajo de cada miembro, lo dedicado al desarrollo y los tiempos por tareas. También ayudará a llevar un control de costes del proyecto, teniendo en cuenta el precio por hora de trabajo de cada perfil técnico.

Gráfico de incidencias creadas vs resueltas: Este indicador también es interesante para conocer si, con los recursos de que se dispone, es posible asumir la carga de trabajo del proyecto o, por el contrario, si se tienen recursos sobredimensionados, ya que no hay tanta carga de trabajo como para disponer de tantos recursos. El gráfico muestra si entra más trabajo del que sale, o bien no entra casi trabajo y se estima que el proyecto está cerca de la finalización, gracias a lo que se podría dedicar algún recurso a otro proyecto.

Gráfico de tarta: Muestra de una forma gráfica a modo de tarta datos de cualquier criterio, por ejemplo: incidencias por estado, gráfico de incidencias asignadas, etcétera.

4. CONCLUSIÓN

Para concluir este trabajo fin de carrera solo queda dar unas cuantas anotaciones respecto al contenido del mismo.

En el inicio del trabajo se ha realizado una explicación sobre qué es Scrum, cómo hay que plantear la implantación del método y cada uno de los pasos necesarios para empezar el trabajo mediante esta metodología. También se ha intentado plantear la implantación de la metodología como un cambio de visión del proceso de desarrollo y cómo la empresa tiene que cambiar su visión y pasar la responsabilidad del desarrollo al equipo Scrum.

En la segunda parte del trabajo se ha definido cómo gestionar Scrum utilizando una herramienta de gestión de proyectos *software* llamada JIRA, que es una aplicación que, como tal, sirve para gestionar incidencias de *software,* pero diferente a la hora de trabajar con ella y muy útil para ayudar con la implantación de la metodología Scrum.

Si se lee solo este apartado de conclusiones se podría pensar que hay mucha documentación de Scrum en internet, pero toda esta documentación encontrada es puramente teórica, pues no comenta de manera práctica cómo aplicar la metodología; simplemente explica qué es la metodología. En este trabajo, a diferencia de otra documentación encontrada, se explica, por ejemplo, qué perfiles profesionales son necesarios para cada uno de los roles dentro de Scrum, cómo utilizar la pizarra Scrum, los campos necesarios para el documento de seguimiento, cómo organizar la reuniones, etcétera. Esta parte puramente práctica es la que se intenta mejorar respecto a lo encontrado, lógicamente, explicando también la parte teórica de la metodología.

Otra de las aportaciones que se pretenden en el trabajo es la de saber si se ha mejorado con Scrum respecto al desarrollo tradicional, incorporando indicadores de verificación que ayudan a justificar la implantación de la metodología.

Por último, y otra mejora respecto a otros trabajos encontrados en internet, es combinar la aplicación de la metodología con la utilización de una herramienta para llevar un seguimiento de los proyectos de desarrollo. La herramienta que mejor considerada para gestionar Scrum es JIRA, por ello, se ha explicado cómo utilizar JIRA cuando se implanta la metodología, para así cerrar el círculo del proceso de desarrollo.

Como reflexión final, el proyecto en sí ha resultado sumamente interesante, pues permite dar apoyo a las empresas u organizaciones al realizar la implantación de Scrum en todo sus aspectos, sin dejar de lado ningún punto y cubriendo todas las necesidades.

Con este documento será posible poner en marcha Scrum, conociendo la teoría, la parte práctica, qué herramienta utilizar para el seguimiento de proyectos y, por último, conociendo qué se debe medir para saber si Scrum ha aportado mejoras en el proceso de desarrollo.

5. BIBLIOGRAFÍA Y REFERENCIAS

- Wikipedia, definición de «Scrum», 2015.Disponible en internet en: <https://es.wikipedia.org/wiki/Manifiesto_%C3%A1gil>. Fecha de visita: 12-10-2015.
- Cómo otros desarrolladores utilizan Scrum. Disponible en internet en: <http://www.desarrolloweb.com/articulos/artefactos-scrum.html>. Fecha de visita: 14-10-2015.
- Conocer más en profundidad el uso de los eventos. Disponible en internet en: <http://desarrollandowebapps.blogspot.com.es/2013/04/eventos-scrum.html>. Fecha de visita: 16-10-2015.
- Artículo sobre la gestión ágil de proyectos. Disponible en internet en: <http://www.marblestation.com/?p=663>. Fecha de visita: 18-10-2014.
- Información de otros tipos de metodologías agiles. Disponible en internet en: <http://www.marblestation.com/?p=661>. Fecha de visita: 20-10-2015.
- Conocer información de como IBM desarrolla Scrum. Disponible en internet en: <https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Rational+Team+Concert+for+Scrum+Projects/page/SCRUM+como+metodolog%C3%ADa>. Fecha de visita: 22-10-2015.
- Varios artículos relacionados con Scrum. Disponibles en internet en: <https://swnotes.wordpress.com/category/scrum>. Fecha de visita: 24-10-2015.
- Artículo sobre los beneficios aplicados a la tecnología. Disponible en internet en: <http://www.i2btech.com/blog-i2b/tech-deployment/los-beneficios-de-implementar-la-metodologia-agil>. Fecha de visita: 26-10-2015.
- Conocer las metodologías agiles para diferenciar con Scrum. Disponible en internet en: <http://inventtatte.com/metodologia-tradicional-vs-agil>. Fecha de visita: 28-10-2015.
- Artículos para minimizar los riesgos al aplicar Scrum. Disponibles en internet en: <http://startmeup.es/tenemos-que-movernos-mas-rapido-metodologias-agiles-scrum>. Fecha de visita: 30-10-2015.
- ScrumManager, Introducción a la Agilidad, 2014a. Disponible en internet en: <http://www.scrummanager.net/bok/index.php?title=Introducci%C3%B3n:_La_agilidad>. Fecha de visita: 1-11-2015
- ScrumManager, Conocer y documentarnos sobre la metodología en la certificación oficial. Introducción al modelo, 2014b. Disponible en internet en: <http://www.scrummanager.net>. Fecha de visita: 1-11-2015.

- ScrumManager, Roles, 2014c. Disponible en internet en: <http://www.scrummanager.net/bok/index.php?title=Roles>. Fecha de visita: 1-11-2015.
- Encuesta de la Scrum Alliance, 2015. Disponible en internet en: <http://www.laboratorioti.com/2015/10/12/informe-del-estado-de-scrum-2015>. Fecha de visita: 15-11-2015.
- ProyectosAgiles, artículo sobre cómo obtener los objetivos de Scrum, 2015. Disponible en internet en: <http://proyectosagiles.org/beneficios-de-scrum>. Fecha de visita: 15-11-2015.
- JIRA Agile y Scrum, cómo combinarlos de manera correcta. Disponible en internet en: <http://www.javiergarzas.com/2014/07/jira-agile-scrum-1.html>. Fecha de visita: 19-11-2015.
- ¿Conoces la técnica de priorización MoSCoW? Disponible en internet en: <http://joaquinoriente.com/2015/03/04/conoces-la-tecnica-de-priorizacion-moscow>. Fecha de visita: 21-11-2015.
- Ingeniería de requerimientos: La priorización MoSCoW. Disponible en internet en: <https://viviendo20.wordpress.com/2014/05/16/ingenieria-de-requerimientos-la-priorizacion-moscow>. Fecha de visita: 21-11-2015.
- Jummp, método MoSCoW, 2013. Disponible en internet en: <https://jummp.wordpress.com/2013/04/27/metodo-moscow>. Fecha de visita: 21-11-2015.
- Documentación de JIRA. Disponible en internet en: <https://confluence.atlassian.com/alldoc/atlassian-documentation-32243719.html>. Fecha de visita: 23-11-2015.
- Aplicación JIRA entorno de prueba. Disponible en internet en: <https://acolome.atlassian.net/secure/Dashboard.jspa>. Fecha de visita: 23-11-2015.