Trabajo final de carrera

Modelo desarrollo software basado en tecnologías ágiles

Ingeniería del software

Consultor: Juan José Gallego

Autor: Enrique Abancens Bazan

Control de cambios

Versión	Fecha	Comentarios	Autor
1.0	03/01/2016	Creación documento	Enrique Abancens

Índice de contenido

1 Acerca del documento

Este documento contiene la memoria para el proyecto de fin de carrera denominado “Modelo desarrollo software basado en tecnologías ágiles”. El objetivo de este proyecto pretende recoger y procedimentar todo el ciclo de vida de desarrollo asociado a la generación de productos software, haciendo especial hincapié en el uso de metodologías ágiles, estableciendo el resultado del mismo en la presente memoria.

El documento comienza con el plan de proyecto, en donde se explica su objetivo, contexto de ejecución y con cronograma de tareas e hitos para su ejecución. Las secciones principales de las que consta son las siguientes:

- Definición del proyecto, en donde se explica el proyecto y la situación que ha motivado su ejecución.
- Recursos destinados para su ejecución.
- Riesgos y viabilidad del proyecto.
- Plan de trabajo, en donde se exponen las tareas e hitos pata su desarrollo.

Los siguientes apartados corresponde al resultado de la ejecución del proyecto, comenzando la definición del proceso ágil para el desarrollo del software. Aquí se presenta proceso para el desarrollo de soluciones software y los diferentes pasos que se requieren durante el desarrollo de aplicaciones. El objetivo es que pueda seguir de guía para el equipo de desarrollo.

El apartado correspondiente al proceso para para el análisis funcional , explica cómo ayudar a los responsables de producto a definir cuáles son los objetivos que deben conseguirse tras el proceso de desarrollo. Además, se muestra cómo traducir estos objetivo en algo que permita a los desarrolladores llevarlos a cabo, de forma que coincida con las expectativas de los usuarios. Seguido a este punto, en la sección dedicada a la auditoría funcional, se describe cómo validar que los desarrollos realizados cumplen correctamente con establecido en el proceso análisis.

Finalmente, se describe la parte de arquitectura, en donde se establece la plataforma tecnológica a utilizar y cómo utilizarla, con el objetivo de cumplir los objetivos establecidos durante el análisis funcional.

2 Plan de proyecto

2.1 Introducción

El objetivo de este apartado es establecer el plan del proyecto asociado a la definición del "Modelo desarrollo software basado en tecnologías ágiles". En éste, se definen la situación actual que motiva la ejecución del presente proyecto, un alcance general del mismo indicando los objetivos a conseguir, los recursos estimados, viabilidad del proyecto, riesgos asociados, y por último, una planificación representada a través de un diagrama de Gantt.

2.2 Definición del proyecto

2.2.1 Situación Actual

Debido a los importantes cambios de mercado existentes en los últimos años, la empresa X ha sufrido una trasformación importante en su catálogo de venta, pasando de vender productos exclusivamente electrónicos a ofrecer soluciones en la que la electrónica se combina con una importante capa software.

Esta repentina transformación, le ha obligado a habilitar de forma repentina diferente lineas de desarrollo software. Debido al poco tiempo disponible para su puesta en marcha, han aflorado problemas relacionados con las productividad de los equipos, baja calidad en la liberación de versiones, dispersión tecnológica, etc.

2.2.2 Alcance proyecto

En este proyecto se pretende recoger y procedimentar todo el ciclo de vida de desarrollo asociado a la generación de productos software, haciendo especial hincapié en el uso de metodologías ágiles. Por lo tanto, en su contenido se tratará de recoger desde la planificación y estimación del proyecto hasta la implantación del mismo, pasando por las fases necesarias para su desarrollo (análisis, desarrollo, etc), permitiendo así, que un nuevo equipo pueda proceder a su aplicación en un espacio aceptable de tiempo.

El proyecto se dividirá en tres áreas importantes, una metodológica, una destinada a los procesos de análisis y auditoría funcional, y por último, una orientada a la parte tecnológica. Aunque todos los integrantes de un equipo deben conocerlas todas, cada una está más orientada a dar respuesta a las necesidades de cada tipo de perfil que conforman el grupo de desarrollo.

El área metodológica está más centrada en guiar el jefe de proyecto en cómo gestionar a su equipo durante la ejecución del proyecto. En consecuencia, se describen conceptos como la estimación e inicio de un proyecto, tipología de roles, cómo hacer uso de Scrum como metodológica ágil, liberar

versiones para su puesta en producción, etc.

El área de análisis pretender abordar, dentro de una metodología como Scrum, la generación ágil de documentos funcionales. Esta documentación, pretende marcar los objetivos del equipo de desarrollo y permitir la validación por parte de nuestros usuarios finales. El otro punto importante de este área es la auditoría funcional, es decir, la generación de las pruebas necesarias que permitan validar los desarrollos realizados.

Por último, se tratará el área tecnológica, en donde se abordará la plataforma que soportará los desarrollos implementados. Dicha plataforma se verá fortalecida por un conjunto de patrones que permitirá su utilización de la mejor forma posible, además, se verá refrendada por auditorías técnicas que aseguren la buena calidad de los desarrollos implementados.

2.2.3 Objetivos

Aquellas empresas sumergidas en un importante proceso de desarrollo software, en el que existen múltiples proyectos ejecutados de forma paralela por varios equipos, requieren un modelo de desarrollo que de soporte a esta realidad. Éste, tendrá como objetivo asegurar la calidad de las diferentes entregas cuidando que no se desvíen conceptos tan importantes como los costes o los hitos de entrega.

Dado que los desarrollos están orientados a la generación de productos software, el ciclo de vida de los mismos se antoja largo. Por este motivo, tan importante como los costes de desarrollo vienen a ser los coste derivados de mantenimiento, en consecuencia, un software construido de una forma ordenada y estandarizada tiende a tener unos coste de mantenimiento equilibrados.

Para cumplir con estos objetivos, el proyecto contará con los siguientes entregables que permitirán cumplir los distintos objetivos:

- Proceso desarrollo software: Conjunto de documentos que deberá permitir a un equipo desarrollo implementar soluciones software de forma adecuada, cumpliendo costes, hitos de entrega, calidad, etc. Este conjunto de entregables vendrá conformado por la siguiente documentación:
    - Definición general del proceso: Mapa que permitirá conocer y seguir el proceso orientado al desarrollo de soluciones software.
    - Definición Roles: El objetivo de este entregable, es dar a conocer los roles y responsabilidad que deberá asumir cada miembro del equipo durante el proceso de desarrollo.
    - Guías detalladas asociadas a la ejecución del proceso: De la definición general del proceso se generarán una serie de guías de fácil lectura que permitirán conocer a cada integrante del equipo que debe hacer en cada fase. Es objetivo, es que todos lo equipos trabajen de forma coordinada y homogénea aumentando su productividad, garantizando la calidad del software con sus correspondiente procesos de auditoría, disminuyendo las

desviaciones en las entregas, etc.

    - Plantillas: Se generarán una serie de plantillas y herramientas que facilitarán a cada equipos la ejecución correcta del proceso.

- Análisis y pruebas: Aunque este es un apartado dentro del área general, la amplitud del mismo nos obliga a dotarlo de entidad propia de forma que permita el definir correctamente el producto a realizar, garantizando la calidad del mismo.
- Arquitectura tecnológica: Conjunto de documentos “ágiles” que definirán la plataforma tecnológica y la forma de hacer las cosas. El objetivo, evitar la dispersión tecnológica y unificar la forma de realizar las desarrollos, esto ayudará a tener un equipo que pueda dar soporte a los diferentes proyectos optimizando los futuros mantenimientos de los productos.

2.3 Recursos destinados

Para la ejecución del proyecto se va a destinar un recurso como responsable de su ejecución. Este podrá disponer de tiempos determinados de los equipos, entre el 5 y 10%, para tomar de requerimientos, validación del proceso y tareas relacionadas.

En un inicio no está presupuestada ninguna partida para subcontratación. En caso de requerirse, éstas podrá ser realizadas tras una solicitud previa, siempre y cuando se encuentres acompañas de una estimación y justificación previas .

2.4 Riesgos

Aunque sea un proyecto metodológico, éste no se encuentra exento de riesgos. Por dicho motivo, tras un análisis de los mismos se han identificado los siguientes puntos que podrían impedir la buena ejecución del proyecto:

- Urgencias: Dado que el retorno de inversión de este proceso no se enfoca a un corto plazo, teniendo como hito su implantación en el siguiente año. Se correo el riesgo que ante necesidades de urgencia para la empresa, no pueda dedicarse el tiempo necesario para la generación de un entregable de calidad, incluso para su finalización
- Reticencia al cambio: Las personas asignadas en los proyectos provienen de distintos orígenes y por lo tanto cada uno posee formas distintas de ejecutar los proyectos. La implantación de una metodología nueva puede provocar una clara reticencia al cambio, por lo que se corre riesgos en la implantación de esta metodología.
- Tiempo limitado de los equipos: Dado que los equipos poseen su hitos de entrega ya pactados, el tiempo de dedicación de estos se puede ver mermado. Esto puede tener como consecuencia una mala definición del proceso, ya que el objetivo inicial es que se conforme de forma consensuada con el objetivo de minimizar la reticencia al cambio.

2.5 Viabilidad

El proyecto se presenta viable, siempre y cuando se gestione correctamente el riesgo de la "urgencia". La viabilidad se sustenta principalmente en que inicialmente no se pretende obtener "el proceso definitivo", el objetivo es conseguir una primera versión que ayude a mejorar a la organización y que de forma iterativa éste mejore, llevando a mejorar los indicadores relacionados.

2.6 Plan de trabajo

2.6.1 Fase e hitos principales

El plan de proyecto, tal y como se expone en la tabla 1, se compone de los siguiente hitos principales, que serán expuesto con mayor detalle en este mismo apartado. Por cada hito puede observarse los correspondientes entregables, duración, y fecha de finalización estimada.

Tabla 1 Fases e hitos principales

⊟PEC 2	27 days	**29/09/15 8:00**	**4/11/15 17:00**
⊟Implantación proceso ágil desarrollo software	27 days	**29/09/15 8:00**	**4/11/15 17:00**
Definición general proceso	5 days	29/09/15 8:00	5/10/15 17:00
Definición roles	5 days	6/10/15 8:00	12/10/15 17:00
Definición Sprint 0	5 days	6/10/15 8:00	12/10/15 17:00
Generación plantillas Sprint 0	5 days	13/10/15 8:00	19/10/15 17:00
Definición Sprint n	12 days	6/10/15 8:00	21/10/15 17:00
Definición sprint review	3 days	22/10/15 8:00	26/10/15 17:00
Definición retrospectiva	2 days	27/10/15 8:00	28/10/15 17:00
Generación indicadores	5 days	29/10/15 8:00	4/11/15 17:00
Gestión Bugs	3 days	22/10/15 8:00	26/10/15 17:00
⊟PEC 3	25 days	**5/11/15 8:00**	**9/12/15 17:00**
⊟Análisis y pruebas	25 days	**5/11/15 8:00**	**9/12/15 17:00**
Generación ágil de funcionales (Historias de usuarios)	5 days	5/11/15 8:00	11/11/15 17:00
Generación plantillas funcionales	10 days	12/11/15 8:00	25/11/15 17:00
Auditoria Funcional (Pruebas)	10 days	12/11/15 8:00	25/11/15 17:00
Implantanción herramientas auditoría funcional	10 days	26/11/15 8:00	9/12/15 17:00
⊟Arquitectura Tecnológica	25 days	**5/11/15 8:00**	**9/12/15 17:00**
Seleccion plataforma tecnologica	18 days	5/11/15 8:00	30/11/15 17:00
Definición diseño arquitectura software	7 days	1/12/15 8:00	9/12/15 17:00
Componentes reutilizables	5 days	1/12/15 8:00	7/12/15 17:00
⊟Memoria y presentación	22 days	**10/12/15 8:00**	**8/01/16 17:00**
Elaboración de la memoria	8 days	10/12/15 8:00	21/12/15 17:00
Elobariación presentación	9 days	22/12/15 8:00	1/01/16 17:00
Entrega memoria y presentación	5 days	4/01/16 8:00	8/01/16 17:00

2.6.2 Diagrama Gantt

A continuación se expondrá el correspondiente diagrama de Gantt, debido a su extensión, este quedará divido para su correcta interpretación (tabla 2, 3 y 4)

Tabla 2: Diagrama Gantt 1/3 (28/09 al 30/10)

Nombre	Duracion	Inicio	Terminado
⊟**PEC 2**	27 days	**29/09/15 8:00**	**4/11/15 17:00**
⊟**Implantación proceso ágil desarrollo software**	27 days	**29/09/15 8:00**	**4/11/15 17:00**
Definición general proceso	5 days	29/09/15 8:00	5/10/15 17:00
Definición roles	5 days	6/10/15 8:00	12/10/15 17:00
Definición Sprint 0	5 days	6/10/15 8:00	12/10/15 17:00
Generación plantillas Sprint 0	5 days	13/10/15 8:00	19/10/15 17:00
Definición Sprint n	12 days	6/10/15 8:00	21/10/15 17:00
Definición sprint review	3 days	22/10/15 8:00	26/10/15 17:00
Definición retrospectiva	2 days	27/10/15 8:00	28/10/15 17:00
Generación indicadores	5 days	29/10/15 8:00	4/11/15 17:00
Gestión Bugs	3 days	22/10/15 8:00	26/10/15 17:00
⊟**PEC 3**	25 days	**5/11/15 8:00**	**9/12/15 17:00**
⊟**Análisis y pruebas**	25 days	**5/11/15 8:00**	**9/12/15 17:00**
Generación ágil de funcionales (Historias de usuarios)	5 days	5/11/15 8:00	11/11/15 17:00
Generación plantillas funcionales	10 days	12/11/15 8:00	25/11/15 17:00
Auditoria Funcional (Pruebas)	10 days	12/11/15 8:00	25/11/15 17:00
Implantancion herramientas auditoría funcional	10 days	26/11/15 8:00	9/12/15 17:00
⊟**Arquitectura Tecnológica**	25 days	**5/11/15 8:00**	**9/12/15 17:00**
Seleccion plataforma tecnologica	18 days	5/11/15 8:00	30/11/15 17:00
Definición diseño arquitectura software	7 days	1/12/15 8:00	9/12/15 17:00
Componentes reutilizables	5 days	1/12/15 8:00	7/12/15 17:00
⊟**Memoria y presentación**	22 days	**10/12/15 8:00**	**8/01/16 17:00**
Elaboración de la memoria	8 days	10/12/15 8:00	21/12/15 17:00
Elobariación presentación	9 days	22/12/15 8:00	1/01/16 17:00
Entrega memoria y presentación	5 days	4/01/16 8:00	8/01/16 17:00

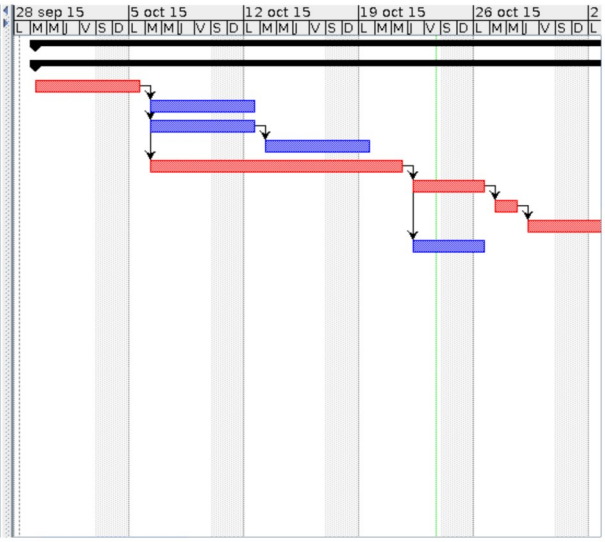


Tabla 3: Diagrama Gantt 2/3 (30/10 al 4/12)

Nombre	Duracion	Inicio	Terminado
⊟**PEC 2**	27 days	**29/09/15 8:00**	**4/11/15 17:00**
⊟**Implantación proceso ágil desarrollo software**	27 days	**29/09/15 8:00**	**4/11/15 17:00**
Definición general proceso	5 days	29/09/15 8:00	5/10/15 17:00
Definición roles	5 days	6/10/15 8:00	12/10/15 17:00
Definición Sprint 0	5 days	6/10/15 8:00	12/10/15 17:00
Generación plantillas Sprint 0	5 days	13/10/15 8:00	19/10/15 17:00
Definición Sprint n	12 days	6/10/15 8:00	21/10/15 17:00
Definición sprint review	3 days	22/10/15 8:00	26/10/15 17:00
Definición retrospectiva	2 days	27/10/15 8:00	28/10/15 17:00
Generación indicadores	5 days	29/10/15 8:00	4/11/15 17:00
Gestión Bugs	3 days	22/10/15 8:00	26/10/15 17:00
⊟**PEC 3**	25 days	**5/11/15 8:00**	**9/12/15 17:00**
⊟**Análisis y pruebas**	25 days	**5/11/15 8:00**	**9/12/15 17:00**
Generación ágil de funcionales (Historias de usuarios)	5 days	5/11/15 8:00	11/11/15 17:00
Generación plantillas funcionales	10 days	12/11/15 8:00	25/11/15 17:00
Auditoria Funcional (Pruebas)	10 days	12/11/15 8:00	25/11/15 17:00
Implantancion herramientas auditoría funcional	10 days	26/11/15 8:00	9/12/15 17:00
⊟**Arquitectura Tecnológica**	25 days	**5/11/15 8:00**	**9/12/15 17:00**
Seleccion plataforma tecnologica	18 days	5/11/15 8:00	30/11/15 17:00
Definición diseño arquitectura software	7 days	1/12/15 8:00	9/12/15 17:00
Componentes reutilizables	5 days	1/12/15 8:00	7/12/15 17:00
⊟**Memoria y presentación**	22 days	**10/12/15 8:00**	**8/01/16 17:00**
Elaboración de la memoria	8 days	10/12/15 8:00	21/12/15 17:00
Elobariación presentación	9 days	22/12/15 8:00	1/01/16 17:00
Entrega memoria y presentación	5 days	4/01/16 8:00	8/01/16 17:00

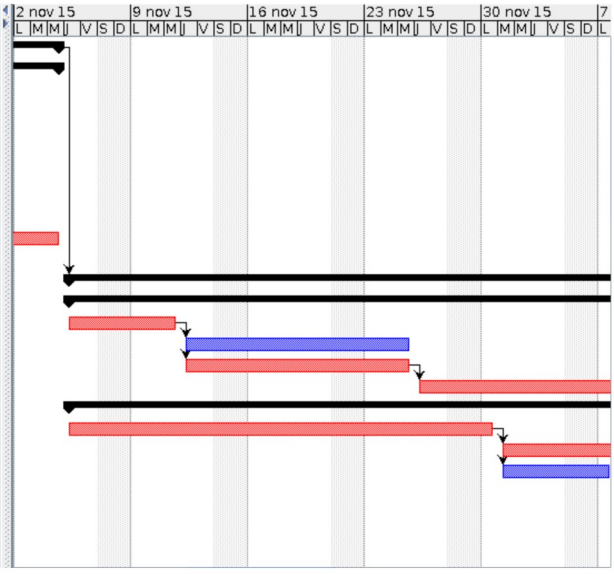
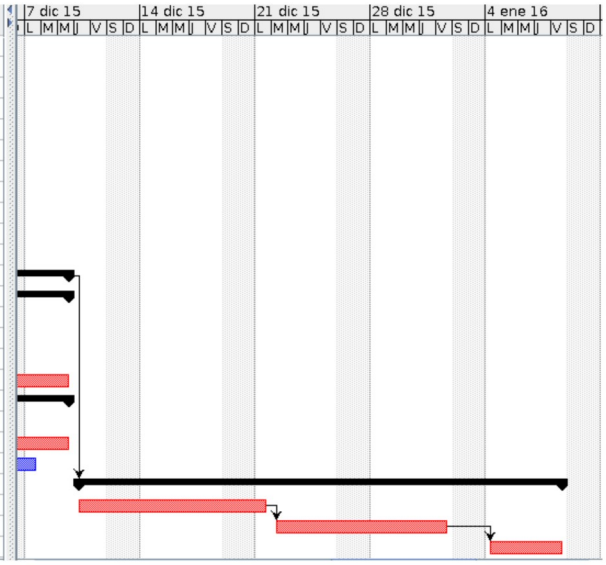


Tabla 4: Diagrama Gantt 3/3 (4/12 al fin del proyecto)

Nombre	Duracion	Inicio	Terminado
⊟PEC 2	27 days	**29/09/15 8:00**	**4/11/15 17:00**
⊟Implantación proceso ágil desarrollo software	27 days	**29/09/15 8:00**	**4/11/15 17:00**
Definición general proceso	5 days	29/09/15 8:00	5/10/15 17:00
Definición roles	5 days	6/10/15 8:00	12/10/15 17:00
Definición Sprint 0	5 days	6/10/15 8:00	12/10/15 17:00
Generación plantillas Sprint 0	5 days	13/10/15 8:00	19/10/15 17:00
Definición Sprint n	12 days	6/10/15 8:00	21/10/15 17:00
Definición sprint review	3 days	22/10/15 8:00	26/10/15 17:00
Definición retrospectiva	2 days	27/10/15 8:00	28/10/15 17:00
Generación indicadores	5 days	29/10/15 8:00	4/11/15 17:00
Gestión Bugs	3 days	22/10/15 8:00	26/10/15 17:00
⊟PEC 3	25 days	**5/11/15 8:00**	**9/12/15 17:00**
⊟Análisis y pruebas	25 days	**5/11/15 8:00**	**9/12/15 17:00**
Generación ágil de funcionales (Historias de usuarios)	5 days	5/11/15 8:00	11/11/15 17:00
Generación plantillas funcionales	10 days	12/11/15 8:00	25/11/15 17:00
Auditoria Funcional (Pruebas)	10 days	12/11/15 8:00	25/11/15 17:00
Implantanción herramientas auditoría funcional	10 days	26/11/15 8:00	9/12/15 17:00
⊟Arquitectura Tecnológica	25 days	**5/11/15 8:00**	**9/12/15 17:00**
Seleccion plataforma tecnologica	18 days	5/11/15 8:00	30/11/15 17:00
Definición diseño arquitectura software	7 days	1/12/15 8:00	9/12/15 17:00
Componentes reutilizables	5 days	1/12/15 8:00	7/12/15 17:00
⊟Memoria y presentación	22 days	**10/12/15 8:00**	**8/01/16 17:00**
Elaboración de la memoria	8 days	10/12/15 8:00	21/12/15 17:00
Elobariación presentación	9 days	22/12/15 8:00	1/01/16 17:00
Entrega memoria y presentación	5 days	4/01/16 8:00	8/01/16 17:00



Según puede apreciarse en nuestro diagrama de Gantt, el proyecto queda dividido en tres grandes bloques, “implantación proceso ágil desarrollo software”, “análisis y pruebas” y “arquitectura tecnológica”. Cada bloque, el cual se explicará a continuación, es un gran hito, existiendo dentro de éste una serie de entregables los cuales corresponden a hitos parciales.

2.6.3 Implantación proceso ágil desarrollo software

Este gran bloque tiene como objetivo establecer el proceso para el desarrollo de software, definiendo los pasos de los que queda compuesto y cómo se deben llevar acabo. Este proceso debe posibilitar a los equipos, actuales y nuevos, a conocer y llevar acabo una correcta construcción del software.

El objetivo principal es que el proceso resultante sea ágil pero efectivo, es decir, que los pasos de los que esté compuesto sean los realmente necesarios, y lo más importante, que sean realmente útiles. Es decir, se intenta huir de metodologías excesivamente burocráticas y rígidas cómo pueda ser Métrica 3 o CMMI, se busca más un proceso ágil e iterativo que mejore en cada iteración Por esta razón, unido al proceso se establecerán una serie de métricas que permitan medir la bondad del proceso y ayuden a mejorarlo poco a poco.

Por lo tanto, este gran bloque se compone de las siguientes entregas:

- Definición proceso general: Esquema que define el proceso de desarrollo a nivel general, sirviendo como guía para el resto de entregables. El objetivo es que todo aquel que participe en el desarrollo de un proyecto software identifique claramente, dónde se encuentra, cuáles son los siguientes pasos, es decir, generar un mapa que muestre el camino a seguir dentro de un proyecto de desarrollo. En este esquema se aprovechará para definir a niveles generales Scrum como metodología ágil sobre el cual se basa el modelo aquí expueto.

Tarea	Entrada	Resultado
Análisis de la situación actual de cada equipo	Reuniones con los integrantes clave.	One Note (Documento ágil) con el resumen de la situación de cada equipo, sus puntos debiles y sus puntos destacables
Análisis del proceso global de la compañía al cual debemos de satisfacer	Reuniones con los responsables de definir los productos, responsables de su implantación y de sus posterior sootasporte	One Note (Documento ágil) con el resumen de las diferentes reuiones.
Análisis de la metodología Scrum, con el objetivo de generar un mapa de procesos ágil que cubran los requerimientos de la empresa.	Análisis de la bibliografía establecida en el documento.	
Diseño del proceso general de desarrollo de software que satisfaga las	Notas de las diferentes reuniones	Mapa de proceso.

necesidades de la organización.		

- Definición roles: Roles, con su correspondiente descripción y responsabilidades, encargados de llevar acabo el proceso definido en el punto anterior. Es importante conocer los roles requeridos para llevar acabo la ejecución de un producto software, de forma que cada integrante del equipo asuma las correspondientes responsabilidades y pueda llevarlas a cabo. Es importante recalcar que cada integrante del equipo puede asumir más de un rol, esto dependerá del tamaño del proyecto.

Tarea	Entrada	Resultado
Análisis de las funciones desempeñadas por los integrantes de los diferentes equipos.	Reuniones con los diferentes equipos	One Note (Documento ágil) con el resumen de las funciones realizadas
Análisis de los roles descritos en las distintos metodologías	Análisis de la bibliografía establecida en el documento.	
Diseño y descripción de los diferentes roles requeridos	Notas de las diferentes reuniones	Documentos con las descripción de los diferentes roles y su matriz de competencia.

- Definición Sprint 0: En el proceso general se podrá identificar una serie de “cajas” responsables de conformar el denominado Sprint 0, siendo éste el responsable de agrupar las acciones necesarias para iniciar un proyecto. En muchas ocasiones al iniciar un proyecto, no se realizan las acciones iniciales requeridas que permiten su ejecución de forma eficiente, incidiendo por tanto es desviaciones temporales y económicas. El resultado de esta entrega, es conocer que se requiere para empezar un proyecto y comenzarlo cuando se cumpla unos requisitos mínimos.

Tarea	Entrada	Resultado
Análisis de las carencias que tienen los distintos grupos, identificando cuales de ellas provienen de un arranque de proyecto inadecuado.	Reuniones con los diferentes equipos	One Note (Documento ágil)
Análisis Sprint 0 según Scrum	Análisis de la bibliografía establecida en el documento.	

Diseño tareas a realizar en el sprint 0	Resultado de los diferentes procesos de análisis realizados	Presentación PPT, para una lectura ágil y amena, de los pasos a realizar en el Sprint 0.

- Definición plantillas Sprint 0: Plantillas que facilitarán la ejecución del sprint 0.

Tarea	Entrada	Resultado
Diseñar plantilla para la ejecución del Sprint 0	Resultado definición Sprint 0	One Note (Documento ágil)

- Definición Sprint N: Scrum define que la ejecución de un proyecto se compone de n ciclos iterativos (subproyectos) en el que en cada uno se genera una evolución del producto. En este apartado se procederá a su definición detallada. Esta entrega es clave, ya que define con exactitud qué debe realizar los integrares del equipo en cada iteración, de forma que generen evoluciones del software de forma ordenadas, garantizando la calidad de los entregables, los hitos marcados, etc.

Tarea	Entrada	Resultado
Análisis de los procesos que sigue cada equipo de software.	Reuniones con los diferentes equipos y seguimiento en su trabajo diario	One Note en donde se refleje su actual proceso, puntos positivos, puntos de mejora,etc. Métricas iniciales que marcan la velocidad del equipo.
Análisis de ejecución de Sprint según Scrum	Análisis de la bibliografía establecida en el documento.	
Análisis de ejecución de un proyecto según metodologías tradicionales	Análisis de la bibliografía establecida en el documento.	
Diseño tareas a realizar en el sprint N	Resultado de los diferentes procesos de análisis realizados	Presentación PPT, para una lectura ágil y amena, de los pasos a realizar en cada Sprint. Formación complementaria

- Definición Sprint review: En cada iteración se debe mostrar los objetivos definidos en cada sprint, quedando por tanto ilustrado en esta entrega. Esto permitirá que los usuarios claves vean crecer poco el proyecto permitiendo verificar en una fase temprana posibles errores de diseño, tomar las decisiones adecuadas de forma ágil, etc.

Tarea	Entrada	Resultado
Análisis Sprint review definido por Sprint	Análisis de la bibliografía establecida en el documento.	
Diseño de la fase de Sprint review	Resultado de los diferentes procesos de análisis realizados	Presentación PPT, para una lectura ágil y amena, de los pasos a realizar en el Sprint review

- Definición retrospectiva: La clave de esta metodología es la mejora continua, siendo un apartado clave en el proceso. El objetivo es marcar una fase en la que nos permita medir, mediante indicadores, cómo hemos ejecutado cada fase, identificar puntos débiles, propones mejoras y planificarlas.

Tarea	Entrada	Resultado
Análisis de la fase de retrospectiva definida por Scrum	Análisis de la bibliografía establecida en el documento.	
Análisis de los procedimientos habituales de mejora continua	Análisis del resultado de la consultoría de procesos realizada en la compañía	
Diseño de la fase de retrospectiva	Resultado de los diferentes análisis	Presentación PPT, para una lectura ágil y amena, de los pasos a realizar en la fase de retrospectiva.

- Generación indicadores: La única manera de mejorar de forma objetiva es medir, por tanto en este punto explica que indicadores debemos de tener en cuenta en la ejecución del proceso.

Tarea	Entrada	Resultado
Análisis del proceso actual de los diferentes equipos para la resolución de bugs	Ejecución de las retrospectivas	Indicadores que permitan medir de forma objetiva a cada equipo.

- Gestión bugs: Además de dar respuesta a los nuevos desarrollos, se deberá realizar un mantenimiento correctivo de la versiones anteriormente publicadas, quedando explicado en este apartado.

Tarea	Entrada	Resultado
Definición de indicadores que permitan medir las bondades de cada equipo	Reuniones con los diferentes equipos y seguimiento en su trabajo diario.	One Note indicando sus procesos de corrección de bugs, estableciendo los puntos positivos, debilidades, etc.
Análisis de ejecución de un proyecto según metodologías tradicionales	Análisis de la bibliografía establecida en el documento.	
Modificación del proceso de ejecución de Sprint, con el objeto de albergar la gestión de bugs.	Análisis del resultado de las tareas anteriores	Proceso de ejecución del Sprint adaptado.

2.6.4 Análisis y pruebas

Este gran bloque se centra en el análisis funcional y sus correspondientes auditorías dentro del proceso de desarrollo. En él se intentará dar respuesta al complicado proceso de definir que debe hacer una aplicación, de forma que sea perfectamente comprendido por todos los actores, desde usuarios finales a desarrolladores:

- Generación ágil de funcionales: Documento que explica cómo generar documentos funcionales de una forma ágil para satisfaces la necesidades de todos los interesados. El objetivo de estos es guiar a los analistas en la generación de documentos funcionales de una forma ágil, minimizando por tanto el coste asociado, pero permitiendo que sirva a los usuario finales para confirma a funcionalidad y los desarrolladores para su implementación.

Tarea	Entrada	Resultado
Análisis de los funcionales existentes en los diferentes	Documentos funcionales existentes	One Note con un resumen de la situación de cada

grupos		grupo.
Análisis de la definición de funcionales (historias de usuario) según Scrum	Análisis de la bibliografía establecida en el documento.	
Diseño de un procedimiento para la generación de análisis funcionales.	Resultados de los análisis realizados	Documento explicativos de cómo generar los análisis funcionales

- Generación plantillas funcionales: Plantillas que servirán de utilidad para la generación de funcionales de una forma ágil.

Tarea	Entrada	Resultado
Generación de plantilla para la realización de análisis funcionales.	Procedimiento generación análisis funcionales	Plantilla para la generación de análisis funcionales.

- Auditoría funcional: Apartado que intenta clarificar la generación de pruebas que permitan validar funcionales de los desarrollos generados a partir de los documentos de análisis. Es decir, una vez realizados los documentos funcionales requeridos y realizados las correspondientes implementaciones, deberemos auditar la funcionalidad desarrollada. Aquí se establecerá los procedimientos que permitirán la correcta generación de pruebas.

Tarea	Entrada	Resultado
Análisis de los documentos de pruebas existentes en los diferentes grupos	Documentos pruebas existentes	One Note con un resumen de la situación de cada grupo.
Análisis de definición de test según TDD	Análisis de la bibliografía establecida en el documento.	
Diseño de un procedimiento para la generación de auditorías funcionales.	Resultados de los análisis realizados	Documento explicativos de cómo generar las auditorías funcionales

- Implantación herramientas auditoría funcional: Implantación de las herramientas adecuadas

que permitan llevar acabo las pruebas previamente definidas. Aquí elegiremos las herramientas adecuadas y procederemos a su instalación.

Tarea	Entrada	Resultado
Selección herramientas para la gestión de auditorías funcionales	Internet y el procedimiento obtenido en el punto anterior	Herramienta seleccionada

2.6.5 Arquitectura tecnológica:

Bloque (PEC 3) que da respuesta a la área tecnológica de los proyectos, definiendo aquí conceptos como la base tecnológica, patrones de desarrollo, etc. Este se compondrá de las siguientes entregas:

- Selección plataforma tecnológica: En esta fase se establecerá la base tecnológica a utilizar en los diferentes desarrollos. Esto se realizarán mediante reuniones conjunta con todos los equipos, compartiendo experiencias y selecciones la mejor solución tecnológica que de respuesta a las necesidades de los proyectos.

Tarea	Entrada	Resultado
Selección plataforma tecnológica	Reuniones periódicas con los distintos grupos	Plataforma tecnológica

- Definición diseño arquitectura software: Conjunto de patrones de diseño, a utilizarse dentro de los diferentes desarrollos. Esto patrones intentar dar respuesta a problemas tecnológicos, es decir, ante un desarrollo a realizar el patrón indica la tecnología a utilizar y como llevarla acabo.

Tarea	Entrada	Resultado
Definición patrones	Reuniones periódicas con los distintos grupos	Patrones de desarrollo

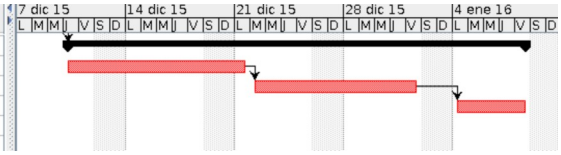
- Componentes reutilizadas: Todos los proyectos poseen un conjunto de funcionales comunes que son susceptibles de ser reutilizadas por lo diferentes equipos. En este área se intenta

definir los procesos que promueven su generación y mantenimiento.

Tarea	Entrada	Resultado
Identificar los componentes reutilizables	Reuniones con los diferentes grupos	Componentes reutilizables

2.6.6 Memoria y presentación

Nombre	Duracion	Inicio	Terminado
Memoria y presentación	22 days	10/12/15 8:00	8/01/16 17:00
Elaboración de la memoria	8 days	10/12/15 8:00	21/12/15 17:00
Elobariación presentación	9 days	22/12/15 8:00	1/01/16 17:00
Entrega memoria y presentación	5 days	4/01/16 8:00	8/01/16 17:00



3 Definición Proceso ágil para el desarrollo de software

3.1 Introducción

El objetivo de este apartado es definir el área metodológica, más centrada en guiar el jefe de proyecto en cómo gestionar a su equipo durante el proceso de ejecución, describiendo conceptos como la estimación e inicio de un proyecto, tipología de roles, hacer uso de Scrum como metodológica ágil, liberar versiones para su puesta en producción, etc.

El documento comienza exponiendo un mapa general, en el cual, se representa el proceso a seguir durante el desarrollo de software. En los siguientes puntos, se describirá a detalle cada uno de los pasos representados en el diagrama inicial, definiendo además los roles responsables de llevarlo acabo.

3.2 Definición general del proceso

3.2.1 Presentación

En este apartado se presenta el proceso para el desarrollo de soluciones software, en éste se expone, de forma general, los diferentes pasos que se requieren durante el desarrollo de aplicaciones. El

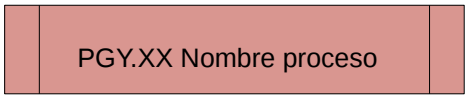
objetivo es que pueda seguir de guía para el equipo de desarrollo.

El proceso se compone por tres hitos principales, estos deberán seguirse de forma secuencial a lo largo de cada año natural. En el hito 1, el responsable de producto presenta la funcionalidad a desarrollar en el presente año y ésta se estima de forma general. Si la estimación es favorable, en el hito 2 se establece la arquitectura y el equipo responsable de su desarrollo, realizando una nueva estimación si el equipo y/o la arquitectura no se adapta a lo previsto en el hito anterior. Finalmente, se procederá a su desarrollo en el hito 3, para cerrarse el proyecto al final del año.

Antes de comenzar con las presentación de los diferentes hitos, se procede a explicar los elementos que conforman el esquema. Es decir, en cada esquema podrá identificarse los siguientes elementos:

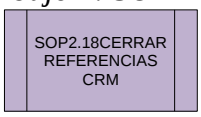
- Proceso general: Un proceso general, representado en el dibujo 1, es un paso dentro del diagrama que viene definido por una presentación y/o documento que lo describe. En el momento que el equipo de desarrollo siga el mapa y se encuentre con un proceso general, significa que debe acudir al apartado de procesos generales en el que se define su objetivo.

Dibujo 1: Proceso general



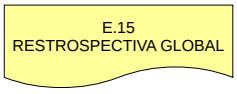
- SOP: Los SOP son pasos dentro del proceso que vienen descritos por un proceso secuencial definido punto a punto. Las principales diferencias con los procesos generales son su dimensión y su nivel de procedimentación. Respecto a su dimensión, estos son notablemente más cortos, además no requieren de ningún conocimiento adicional para llevarse acabo y por lo tanto pueden realizarse por cualquiera, ya que están perfectamente procedimentados.

Dibujo 2: SOP



- Productos: Entregables, pudiendo ser documentos, versiones de software, etc.

Dibujo 3: Entregable

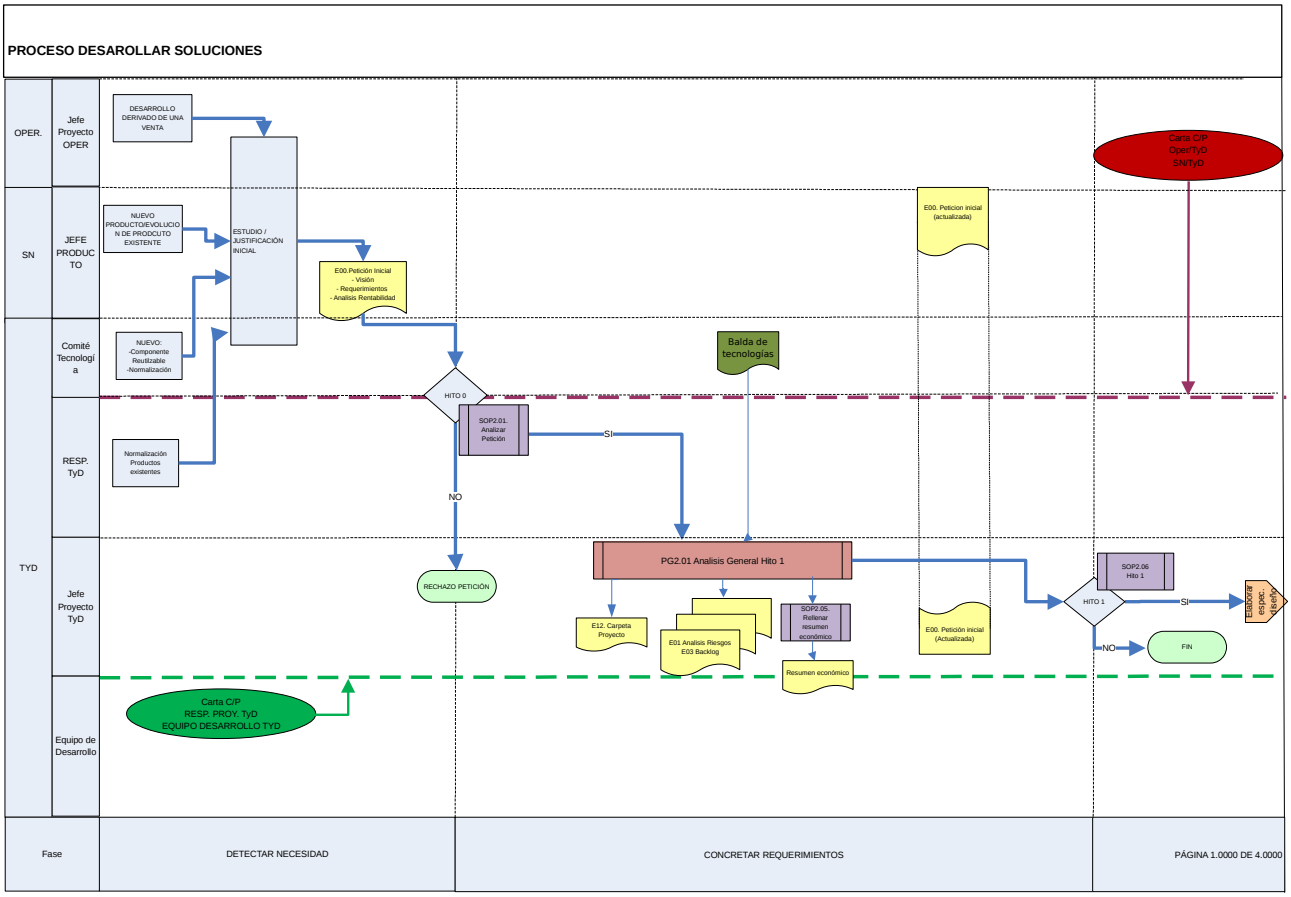


3.2.2 Hito 1

El siguiente esquema, mostrado en la ilustración 1, representa el hito1, en el que cada año el responsable de cada producto deberá establecer los objetivos a desarrollar en el presente ciclo,

además, estos objetivos podrán verse complementados por desarrollos de carácter tecnológico por el propio equipo de desaroollo. Todos estos objetivos deberá verse reflejados en un documento denominado “E00. Petición inicial”.

Ilustración 1: Figura Hito 1



Una vez recibido el E00, el responsable del departamento de desarrollo analizará que la documentación entregada posee el formato y contenido adecuado, permitiendo asignar a un responsable de proyecto que lleve acabo el análisis correspondiente a este hito. Este paso, se describe en los documentos **SOP2.01 Analizar Petición y Actualizar Índice de Proyecto** y **SOP2.03 Apertura en CRM y asignación responsable de proyecto**. En caso de que la documentación no cumpla los requisitos requeridos, se rechazará la petición hasta que se presente adecuadamente.

El responsable de proyecto, siguiendo lo definido por el proceso general **PG 2.01 Análisis general hito 1**, procederá a la estimación general del documento entregado por el responsable de producto. Gracias a este proceso, se generarán los siguientes entregables:

- Carpeta de proyecto
- E03 Backlog: Siguiendo lo marcado por Scrum, se generará un archivo en formato Excel en donde se recogerán las historias (funcionalidades), estimadas y priorizadas, identificadas

durante la lectura del documento E00. (Henrik Kniberg, 2007)

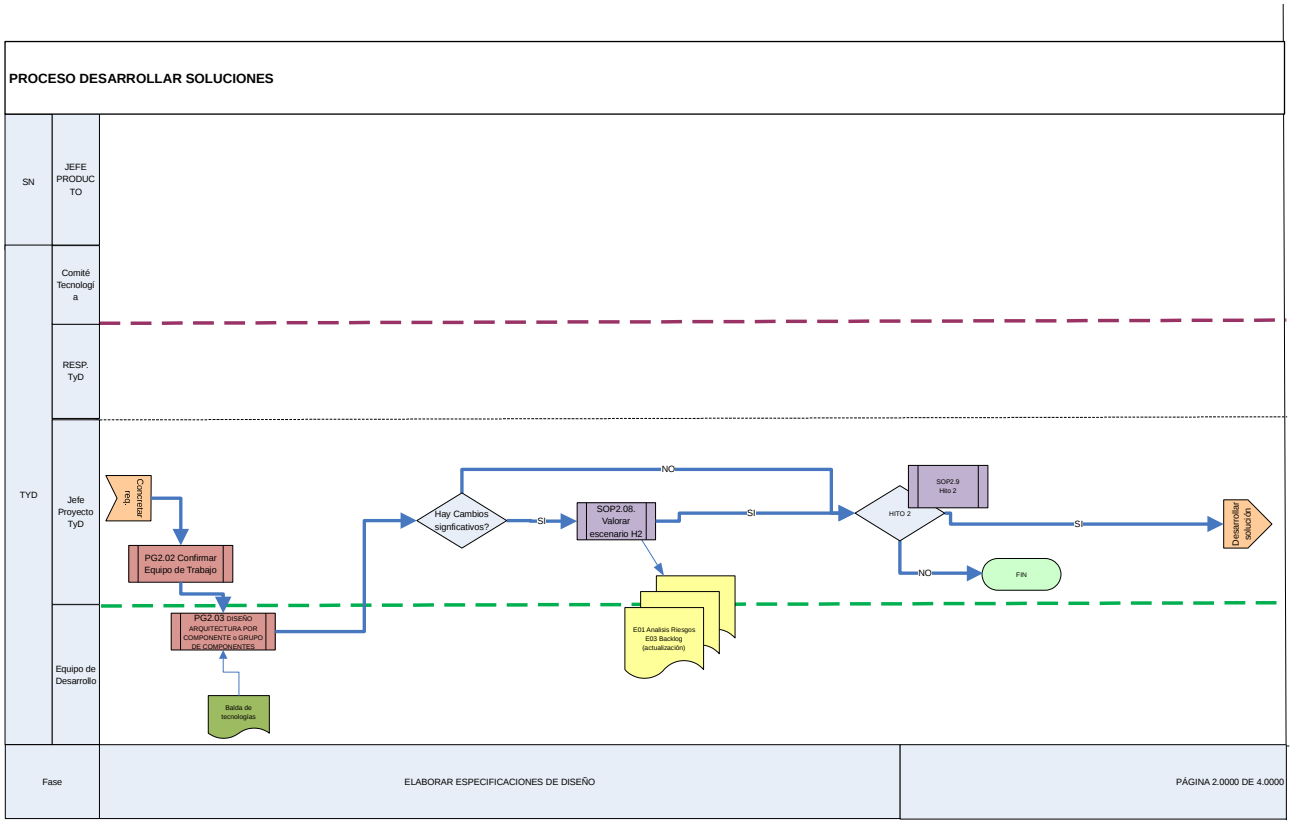
- E01 Análisis de riesgos: Riesgos asociados a la ejecución del proyecto.
- Resumen económico del proyecto.

Si la estimación determina que le desarrollo es viable, y que además éste se encuentre dentro de los margenes económicos definidos por el responsable del producto, se dará finalizado el hito 1, pudiendo pasar al siguiente, una vez que se haya seguido el SOP 2.06 Hito 1.

3.2.3 Hito 2

En el inicio del segundo hito, mostrado en la ilustración 2, significa que el proyecto parece viable y es el momento de empezar a concretar más en detalle. Por lo que en esta fase, confirmaremos el marco en donde se llevará acabo el proyecto, revisando los precios si este escenario difiere del inicialmente estimado.

Ilustración 2: Figura Hito 2



Se comienza con confirmar el equipo de trabajo, ya que el inicialmente estimado puede variar del que realmente esté disponible, este paso se realizará por medio de lo definido en el proceso general PG2.02 Confirmar equipo de Trabajo. Posteriormente, se revisará la arquitectura a nivel general, identificando si existe algún componente ya desarrollado que pueda ser reutilizado o bien vamos a desarrollar algún elemento que pueda ser utilizado a futuro por algún otro proyecto.

Una vez analizadas ambas cuestiones, se revisará si la nueva situación impacta de forma

considerable es la valoración inicial, en caso de ser así, se procederá a nueva estimación según los marcado en el SOP2.08 Valorar escenario H2. Como en el hito 1, este último paso generará una nueva revisión de los siguientes documentos:

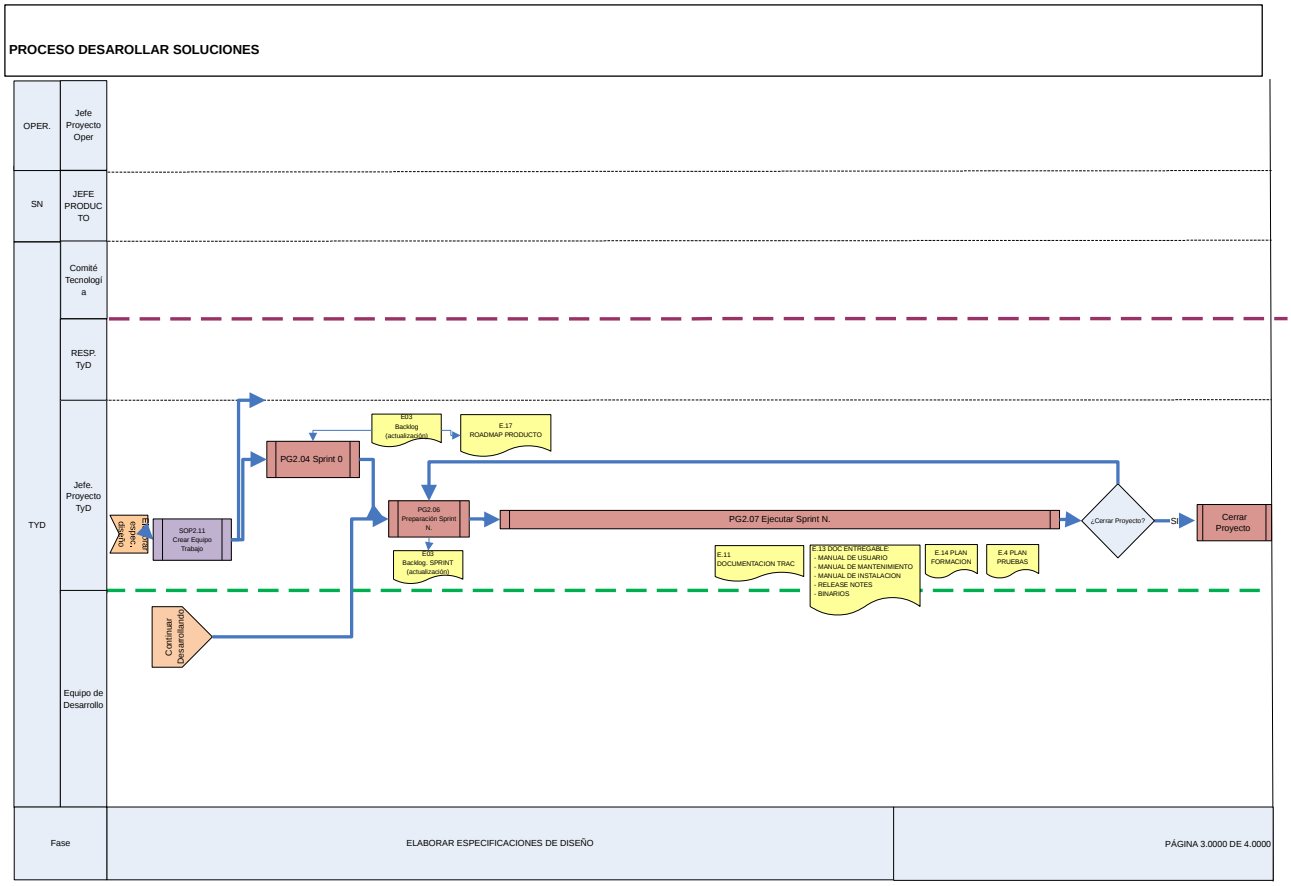
- E01 Análisis de riesgos: Riesgos asociados a la ejecución del proyecto.
- Resumen económico del proyecto.

Finalmente, en función de la estimación, se decidirá iniciar el proyecto o bien no llevarlo acabo si los costes/ riesgos hacen inviable su desarrollo.

3.2.4 Hito 3

Gracias al hito 1 e hito 2, se ha estudiado la viabilidad del proyecto, se ha determinado un coste económico inicial, se ha definido la arquitectura general y se ha confirmado el equipo de trabajo. Por lo que en el tercer hito, representado en la ilustración 3, sumerge al propio equipo en el proceso de desarrollo, por lo tanto, este hito se continuará de forma iterativa hasta la finalización del proyecto.

Ilustración 3: Figura hito 3



Se comienza creando el equipo de trabajo (SOP 2.11 Crear equipo de trabajo), evidentemente, en esta fase no es necesario la totalidad del mismo pero si es importante planificar su entrada. De forma inmediata, entraremos en el Sprint 0 (PG2.04 Sprint 0), siendo éste como luego veremos, el responsable de preparar todo lo necesario para iniciar el proyecto.

Finalizado el Sprint 0, y está todo preparado y se puede entrar ya en el ciclo iterativo (Sprint N) que gobernará el desarrollo del proyecto. Este proceso se describe con detalle en los procesos generales PG2.06 Preparación Sprint N y PG2.07 Ejecutar Sprint N. Dado que el proceso se basa en Scrum, este ciclo se repite cada 2-3 semanas (en función del proyecto) generando como resultado un nuevo entregable. (Henrik Kniberg, 2007)

Normalmente, al finalizar el año, se cerrará el proyecto para su nueva apertura el año que viene si la compañía considera útil mantener la correspondiente linea de desarrollo.

3.3 Roles

3.3.1 Introducción

Une vez descrito el proceso general y antes de entrar en detalle en cada uno de lo pasos, se describirá los roles implicados en el proceso de desarrollo. Conocer los roles implicados, ayuda a entender mejor el proceso y definir correctamente las responsabilidad de cada integrante del grupo.

Los siguientes puntos describirán en detalle cada uno de los roles existentes, definiendo claramente su objetivos. Todos los integrantes de grupo deberán conocer con detalle los diferentes roles existentes y sus responsabilidades, ya que en función de la dimensión del proyecto, una misma persona puede desempeñar múltiples roles. (Àles Alfonso I Minguillón, Eulàlia Clos Cañellas, Humberto Andrés Sanz, Isabel Domènech Puig-Serra, Jordi Schoenenberger Arnaiz, 2009)

3.3.2 Program Manager

Misión

Este puesto tiene como objetivo reunir y organizar las necesidades metodológicas y técnicas existentes en los diferentes proyectos, con el objeto definir, de forma coordinada con los jefes de proyecto, analistas y arquitectos de cada proyecto, los procesos y estándares que permitirán llevar acabo desarrollos con un alto nivel de calidad. En paralelo se dará un soporte activo a la correcta aplicación estos procesos y estándares a las diferentes líneas de producción, realizando una mejora continua de los mismos.

Finalidad 1

Identificar las necesidades metodológicas y técnicas existentes en cada proyecto, definiendo las acciones de mejora que permitan superar dichas carencias.

(Nivel de ponderación 35%)

Actividades principales

1. Analizar el estado del proyecto revisando la documentación disponible, procesos de calidad existentes, nivel de scrum aplicado, procesos de implantación, tratamiento del código, gestión de incidencias, indicadores, etc.
2. Identificar los puntos de mejora estableciendo un plan de acción asociado.
3. Exponer y consensuar, con la totalidad del equipo, los puntos de mejora definidos junto con sus correspondientes planes de actuación.
4. Priorizar, junto con el resto del equipo, las acciones consensuadas.
5. Dar soporte activo al equipo, tanto del las nuevas acciones definidas así como en los procesos existentes a nivel de departamento.

Finalidad 2

Aglutinar las diferentes acciones de mejora tomadas en los distintos proyectos, con el objeto de mejorar los procesos generales de desarrollo a nivel departamental. Estos procesos serán consensuados con los analistas, arquitectos y jefes de proyecto de cada línea de producción antes de su aplicación.

(Nivel de ponderación 30%)

Actividades principales

1. Analizar las acciones de mejora tomadas en los diferentes proyectos, identificando cuales de ellas pueden ser candidatas a forma parte de los distintos procesos de desarrollo.
2. Proponer estas acciones de mejoras a los puestos estratégicos de cada línea de producto, con el objeto de consensuar su aplicación o bien proceder a una modificación.
3. Modificar los procesos generales con las nuevas acciones de mejora consensuadas.
4. Extender la nueva versión de estos procesos a todos los equipos de desarrollo, dando además un soporte activo en su aplicación.

Finalidad 3

Mantener un catálogo activo de componentes reutilizables, ayudando a los diferentes equipos a hacer uso de los mismos o bien a aportar en su crecimiento.

(Nivel de ponderación 20%)

Actividades principales

1. Mantener un catálogo de los componentes reutilizables, apoyando su utilización y dando soporte a los diferentes equipos en su implantación.
2. En los procesos de análisis de los diferentes proyectos, identificar nuevos posibles componentes reutilizables, justificando su potencial reutilización.
3. Proponer a la línea de producto correspondiente la liberación de este desarrollo como componente reutilizable.

Finalidad 4

Establecer estándares tecnológicos en consenso con las diferentes líneas de producto.

(Nivel de ponderación 15%)

Actividades principales

1. Aglutinar las tecnologías utilizadas y requeridas en cada una de las líneas, así como su forma de uso.
2. Definir, tras previo consenso, estándares tecnológicos de desarrollo, de forma que se evite la dispersión tecnológica.
3. Definir y mantener un catálogo de estándares de diseño agrupados por tecnológica. Esto permitirá que todo desarrollador enfoque de forma similar la solución ante la implementación de una funcionalidad. Este se nutrirá a partir de los catálogos establecidos en cada equipo por el correspondiente arquitecto, siendo por tanto la labor de este puesto identificar los patrones de cada equipo e incorporar a un catálogo general aquellos que sean susceptibles de ser reutilizados.
4. Proponer el uso de frameworks tecnológicos y la incorporación de funcionalidades a estos.

Perfil de competencias del puesto

COMPETENCIA	Importancia	NIVELES			
		I	L	U	O
Competencias Técnicas					
Scrum	Alta				**X**
Trac	Alta				**X**
TestLink	Alta				**X**
Definición componentes reutilizables, arquitecturas, etc.	Alta				**X**
Diseño Técnicos	Alta				**X**
Desarrollo orientado a Test	Alta				**X**
Patrones Diseño	Media				**X**
Conocimiento Técnico Hardware / Software	Alto			**X**	
Gestión de riesgos	Alta			**X**	
Gestión de tareas	Alta				**X**
Inglés	Alta			**X**	
Competencias Gestión					
Capacidad toma de decisiones	Alta				**X**
Gestión de recursos	Alta				**X**
Capacidad de negociación	Media			**X**	
Competencias Personales					
Trabajo en equipo	Alta				**X**
Gestión de emociones	Alta			**X**	
Autoaprendizaje	Alta			**X**	

3.3.3 Responsable proyecto

Misión

Coordinar adecuadamente los recursos existentes con el objetivo de alcanzar las metas definidas en el proyecto, con los plazos y costes establecidos. Para ello, durante la ejecución del proyecto, deberá apoyar al dueño del producto en la priorización y definición en detalle de las metas fijadas.

Con estos objetivos establecidos y priorizados deberá, con el apoyo del equipo de desarrollo, definir y planificar las tareas que permitan su consecución, gestionando finalmente a su equipo para que se centre en su realización de una forma ordenada y coordinada.

Finalidad 1

Apoyar al dueño de producto en priorizar y definir correctamente los objetivos del proyecto, realizando este proceso de forma iterativa durante su ejecución. Para cumplir correctamente con esta finalidad, se deberá implicar al responsable del producto de forma activa, reportando a éste de forma ágil la situación del proyecto.

(Nivel de ponderación 40%)

Actividades principales

1. Analizar e interiorizar los objetivos (historias épicas) marcados por el dueño del producto.
2. Proponer, siempre que sea posible, dividir estos objetivos en otros más pequeños (historias) con el objeto de obtener resultados de forma más temprana.
3. Identificar riesgos que puedan poner en peligro la buena resolución del proyecto.
4. Apoyar al dueño del producto en la priorización de los objetivos existentes, teniendo como meta eliminar riesgos lo antes posible y conseguir resultandos en un corto espacio de tiempo, de forma que ratifique que la ejecución va por el camino adecuado.
5. Proponer posibles nuevos objetivos que garanticen la correcta ejecución del proyecto.
6. Informa activamente del estado del proyecto (hito conseguidos, coste, fechas, retrasos,…).
7. Exposición y aprobación, por parte del dueño del producto, de la especificación detallada asociada a cada uno de los objetivos marcados.
8. Exposición iterativa de los objetivos logrados durante la ejecución del proyecto, con el objeto de validar el resultado conjuntamente con el dueño del producto (Sprint Review).

Finalidad 2

En función de los objetivos marcados con el dueño del producto y sus correspondientes priorizaciones, el responsable de proyecto deberá coordinar adecuadamente a su equipo para cumplir estas metas en el tiempo y coste establecidos.

(Nivel de ponderación 40%)

Actividades principales

1. Gestionar correctamente el listado de objetivos marcados (Product Backlog), con sus correspondientes priorizaciones, estimaciones asociadas (Puntos de historia), esfuerzo dedicado, horas pendientes, costes, etc.
2. Identificar, con el apoyo del equipo de desarrollo, las tareas que permitirán la consecución de las metas establecidas y su coste asociado (esfuerzo / coste económico).

3. Planificar junto con el equipo de desarrollo (Spring Planning), para posteriormente gestionar las tareas existentes, teniendo identificado su estado, tiempo restante de resolución, estimación inicial, coste, etc.
4. Seguimiento activo (Daily) del estado de las tareas existentes, y por lo tanto del estado global de los objetivos marcados.
5. Realizar las acciones necesarias que permitan al equipo centrarse en las tareas y objetivos establecidos, evitando que posible "ruido" exterior desvíe a éste de la estrategia marcada.

Finalidad 3

Identificar y priorizar posibles puntos mejoras detectados durante la ejecución de los proyectos. Para posteriormente planificar y gestionar las acciones que permita su aplicación.

(Nivel de ponderación 10%)

Actividades principales

1. Realizar reuniones de retrospectiva, junto con el resto del equipo y el dueño del producto, con el objeto de identificar posibles puntos de mejora.
2. Coordinar las acciones necesarias para aplicar los puntos de mejora acordados.
3. Auditar los resultados asociados a los puntos de mejora aplicados, con el objeto de verificar su efectividad.
4. Informar al departamento de los puntos de mejora aplicados y su resultado.
5. Coordinar la aplicación de los puntos de mejora propuesto a nivel departamental.

Finalidad 4

Coordinar la documentación asociada a los productos gestionados y sus correspondientes proyectos.

(Nivel de ponderación 5%)

Actividades principales

1. Coordinar las tareas que permitan tener debidamente documentados los diferentes productos y proyectos gestionados.
2. Definir las tareas que permitan la publicación de la información adecuada asociada a los diferentes productos, de forma que esta pueda ser consumida por los departamentos de la organización que la requieran.
3. Definir las acciones necesarias para ampliar el conocimiento relativo a los productos

desarrollados, de forma que otras áreas puedan proceder a su explotación, instalación y soporte.

Finalidad 5

Estimar y planificar propuestas de proyecto.

(Nivel de ponderación 5%)

Actividades principales

1. Estima plazos, costes y recursos asociada a las propuestas realizadas por los diferentes segmentos.
2. Identificar riesgos asociados al proyecto, estimando cómo afectarán estos en coste y plazo.
3. Identificar el equipo necesario que permita la ejecución del proyecto en los tiempos y costes establecidos.

Perfil de competencias del puesto

COMPETENCIA	Importancia	NIVELES			
		I	L	U	O
Competencias Técnicas					
Scrum	Alta				**X**
Trac	Alta				**X**
TestLink	Media		**X**		
Capacidad análisis	Alta			**X**	
Conocimiento Técnico Hardware / Software	Media		**X**		
Estimación y planificación	Alta				**X**
Gestión de riesgos	Alta			**X**	
Gestión de tareas	Alta				**X**
Gestión económica proyectos	Alta			**X**	
Inglés	Alta		**X**		
Competencias Gestión					
Capacidad toma de decisiones	Alta				**X**
Gestión de recursos	Alta				**X**
Capacidad de negociación	Media			**X**	
Competencias Personales					
Trabajo en equipo	Alta				**X**
Gestión de emociones	Alta			**X**	

3.3.4 Analista

Misión

Analizar en detalle los objetivos establecidos y priorizados por el responsable del proyecto y el dueño del producto, de forma que permitan la generación de un documento que pueda ser empleado por el equipo para su consecución. Además, deberá auditar el estado funcional de los trabajos realizados de forma que garanticen la calidad final del producto final, generando indicadores que permitan el posterior desarrollo de acciones de mejora.

Finalidad 1

Analizar en detalle los objetivos (historias) establecidos y priorizados por el responsable del proyecto y el dueño del producto. Tras el análisis, generar la documentación necesaria que permita trabajar al equipo de desarrollo en estas funcionalidades, pudiendo además, ser validado por el dueño del producto.

(Nivel de ponderación 50%)

Actividades principales

1. Analizar e interiorizar los objetivos (historias épicas) marcados por el dueño del producto.
2. Apoyar el responsable de proyecto en proponer, siempre que sea posible, dividir estos objetivos en otros más pequeños (historias), con el objeto de obtener resultados de forma más temprana.
3. Apoyar al responsable de proyecto en identificar riesgos que puedan poner en peligro la buena resolución del proyecto.
4. Generar la documentación necesaria que permita al equipo de trabajo el desarrollo de los objetivos establecidos. Este deberá ser previamente aceptada por el dueño del producto, por lo que se requerirá que esté adecuada para una buena compresión para este tipo de puestos.
5. Dar soporte funcional al equipo de desarrollo durante el desarrollo de las tareas establecidas.
6. Generación de documentación requerida para extender el conocimiento de explotación del producto a otras áreas (Manual de usuario, documentación de formación,..).

Finalidad 2

Este puesto deberá llevar a cabo las auditorias funcionales que permitan certificar el buen funcionamiento de los desarrollos realizados, teniendo en cuenta la documentación de análisis previamente generada.

(Nivel de ponderación 30%)

Actividades principales

1. Definición de las pruebas que permita validar el correcto funcionamiento de los desarrollos realizados.
2. Definición de las pruebas de integración requeridas en caso de liberación de una nueva versión.
3. Soporte al responsable de proyecto para coordinar las auditorías funcionales entre los diferentes testers y el mismo.
4. Ejecución de pruebas funcionales durante los procesos de auditoría funcional.
5. Soporte funcional durante el proceso de pruebas y correcciones.
6. Identificación de pruebas susceptibles de ser automatizadas.
7. Informar del resultado final de las pruebas realizadas al responsable de proyecto y al dueño del producto.

Finalidad 3

Generación de indicadores resultantes de los procesos de auditoría funcional.

(Nivel de ponderación 10%)

Actividades principales

1. Generación de indicadores asociado a los fallos de desarrollo encontrados durante las auditorias funcionales.
2. Generación de indicadores asociados a los fallos encontrado durante las pruebas de integración.
3. Indicadores asociados a errores encontrado en producción y no detectados durante la auditoria funcional.
4. Publicación de estos indicadores, tanto a nivel de proyecto como a nivel de departamento.

Finalidad 4

Identificar posibles puntos mejoras detectados durante los procesos de análisis y auditoria funcional.

(Nivel de ponderación 10%)

Actividades principales

1. Identificar en las reuniones de retrospectiva posibles punto de mejora dentro de los procesos de análisis y auditoria funcional.
2. Identificar nuevos indicadores que aporten en la mejora del proceso, así como eliminar aquellos que ya no sean de utilidad.
3. Auditar los puntos de mejora acordados en las reuniones de retrospectiva.
4. Dar soporte funcional en los procesos de automatización de las pruebas.

Perfil de competencias para el puesto

COMPETENCIA	Importancia	NIVELES			
		I	L	U	O
Competencias Técnicas					
Scrum	Medio			**X**	
Trac	Alta			**X**	
TestLink	Alta				**X**
Capacidad análisis	Alta				**X**
Redacción especificaciones Funcionales	Alta				**X**
Elaboración Prototipos	Media			**X**	
Definición Test	Alta				**X**
Ejecución Test	Alta				**X**
Conocimiento Técnico Hardware / Software	Bajo		**X**		
Gestión de riesgos	Media		**X**		
Gestión de tareas	Media			**X**	
Inglés	Media		**X**		
Competencias Gestión					
Capacidad toma de decisiones	Alta			**X**	
Gestión de recursos	Media			**X**	
Competencias Personales					
Trabajo en equipo	Alta			**X**	

3.3.5 Arquitecto Software

Misión

Responsable de definir la capa tecnológica requerida para el cumplimiento de los objetivos establecidos, auditando durante la ejecución del proyecto que la construcción se realiza de forma adecuada. Además, deberá dar apoyo tecnológico al equipo de desarrollo, bien mediante soporte técnico como a través de la generación de componentes que mejoren los procesos de desarrollo.

Finalidad 1

Definición de la plataforma tecnológica que permita el desarrollo de los objetivos marcados en el proyecto con los costes y plazos establecidos.

(Nivel de ponderación 25%)

Actividades principales

1. Analizar y seleccionar las plataformas tecnológicas adecuadas para dar respuestas a los objetivos requeridos por el proyecto, en el plazo y coste establecidos. Esta selección se realizará en base a los estándares definidos por el departamento con el fin de evitar la dispersión tecnológica.
2. Identificar los componentes reutilizables, propios o de terceros, que permitan dar respuesta de una forma más adecuada a las necesidades del proyecto (Incluyendo aquí desde librerías y aplicaciones hasta sistemas de auditoría,..).
3. Identificar riesgos asociados a la tecnología seleccionada, con el objeto de que sean gestionados durante la ejecución del proyecto.
4. Definición de diagramas de despliegue que permita entender al equipo de desarrollo la arquitectura seleccionada y cómo ésta debe funcionar dentro del ecosistema del proyecto.
5. Proponer activamente al responsable de proyecto nuevas metas tecnológicas (historias), que garanticen el buen desarrollo del resto.

Finalidad 2

Definición de estándares de desarrollo (diseño) que permita la ejecución de los desarrollos, bajo la plataforma tecnológica seleccionada, de una forma adecuada. Posteriormente, estos desarrollos deberán ser auditados con el objetivo de verificar que siguen la normativa establecida (Auditoria técnica).

(Nivel de ponderación 25%)

Actividades principales

1. Análisis técnico de los objetivos (historias) priorizados, con el objeto de determinar si existe un estándar de diseño que pueda dar respuesta o bien requiere de la implementación de uno nuevo.
2. Desarrollo de los nuevos estándares de diseño requeridos por los objetivos del proyecto.
3. Exposición de los estándares de desarrollo definidos el equipo de desarrollo.
4. Ejecución de los procesos de auditoría técnica, con el objetivo de verificar que los desarrollos

realizados siguen el estándar adecuado. Este proceso puede ser realizado directamente por el propio arquitecto o bien con soporte de ciertos desarrolladores del equipo.

5. Para procesos de autoría en que se requiere la intervención de apoyo por parte de los desarrolladores, deberán ser coordinados por el arquitecto de software.
6. Generación de indicadores tras los procesos de auditoría en los que se refleje el grado de cumplimiento de los diferentes estándares.
7. Informar del resultado final de las auditorías realizadas al responsable de proyecto y al dueño del producto.
8. Dado que este puesto es el responsable audita la calidad técnica de los desarrollos realizados, será éste quien genere la versiones que posteriormente serán testadas funcionalmente y puestas en producción.

Finalidad 3

Generación de componentes reutilizables que mejoren el proceso de desarrollo (Facilidad de desarrollo, reutilización, fiabilidad,..), no pudiendo desarrollar en ningún caso funcionalidad alguna.

(Nivel de ponderación 25%)

Actividades principales

1. Desarrollo, bien por parte del arquitecto o a través de un equipo de desarrollo, de componente reutilizables utilizados por el equipo de desarrollo.
2. Desarrollo de ejemplos o arquetipos que asista a los desarrolladores en la implementación de partes complejas.
3. Integración de componentes de terceros (librerías, aplicaciones,..), bien por parte del arquitecto como a través de un equipo de desarrollo.
4. Automatización de pruebas, bien por parte del arquitecto como a través de un equipo de desarrollo.
5. Formación el equipo de desarrollo de los componentes desarrollados.
6. En el caso de que el desarrollo de componentes sea realizado por un equipo de desarrollo, éste deberá ser coordinado por el arquitecto bajo la supervisión del responsable de proyecto.

Finalidad 4

Identificar posibles puntos mejoras detectados durante los procesos de análisis y auditoria técnica.

(Nivel de ponderación 15%)

Actividades principales

1. Identificar en las reuniones de retrospectiva posibles punto de mejora dentro de los procesos de análisis y auditoría técnica.
2. Identificar nuevos indicadores que aporten en la mejora del proceso, así como eliminar aquellos que ya no sean de utilidad.
3. Auditar los puntos de mejora acordados en las reuniones de retrospectiva.
4. Coordinar la aplicación de los puntos de mejora propuesto a nivel departamental (Incorporación de herramientas para auditoría de código, pruebas, etc).

Perfil de competencias para el puesto

COMPETENCIA	Importancia	NIVELES			
		I	L	U	O
Competencias Técnicas					
Scrum	Medio			**X**	
Trac	Alta			**X**	
TestLink	Media		**X**		
Definición componentes reutilizables, arquitecturas, etc.	Alta				**X**
Diseño Técnicos	Alta				**X**
Estimación y planificación	Alta			**X**	
Desarrollo orientado a Test	Alta			**X**	
Patrones Diseño	Media			**X**	
Inglés	Alta			**X**	
JAVA					
J2EE	Alta				**X**
SOA	Media			**X**	
Servidor Aplicaciones	Alta				**X**
Spring	Alta				**X**
Web	Media			**X**	
ESB	Media / Baja		**X**		
SGBD	Media			**X**	
.NET					
.NET Framework	Alta				**X**
SOA (WCF)	Media			**X**	
IIS	Media			**X**	
WPF	Alta				**X**
Web	Medio			**X**	
ESB	Baja		**X**		
SGBD	Media			**X**	
C/C++					
C	Alto				**X**
C++	Alto				**X**

Estándares c++98/c++11					
Colas Mensajería(D-Bus, ZeroMQ, Active MQ)					
Frameworks (Qt5, Boots)					
Comunicaciones Serie					
Comunicación ficheros especiales (pipes / fifo)					
Concurrencia					
Sockets					
SGBD					
Sistemas					
Linux	Medio		**X**		
Windows	Medio		**X**		
Virtualización	Medio		**X**		
Alta Disponibilidad	Medio		**X**		
Competencias Gestión					
Capacidad toma de decisiones	Alta			**X**	
Gestión de recursos	Media			**X**	
Competencias Personales					
Trabajo en equipo	Alta			**X**	
Autoaprendizaje	Alta			**X**	

3.3.6 Desarrollador

Misión

Responsable de llevar acabo los desarrollos planificados según las especificaciones establecidas en el análisis funcional y bajo los patrones de diseño definidos. Estos desarrollos deberán estar debidamente documentos y testados con las pruebas unitarias y de integración que se requieran. De forma paralela, deberá servir de apoyo en las tareas de estimación, arquitectura y auditoría que se requieran durante la vida de los diferentes proyectos en los que participe.

Finalidad 1

Responsable de llevar acabo los desarrollos planificados según las especificaciones establecidas en el análisis funcional y bajo los patrones de diseño definidos. Estos desarrollos deberán estar debidamente documentados y testados con las pruebas unitarias y de integración que se requieran.

(Nivel de ponderación 70%)

Actividades principales

1. Interiorizar los análisis funcionales de forma que se tenga claro el objetivo final del desarrollo a implementar.

2. Analizar los desarrollos para establecer el patrón de diseño IKUSI que permita su implementación. En caso de no existir ninguno, informar al arquitecto para establecer la estrategia adecuada.
3. Realizar los correspondientes desarrollos según las especificaciones establecidas en el análisis funcional y bajo los patrones de diseño definidos.
4. Documentar adecuadamente los desarrollos de forma que puedan ser posteriormente mantenidos por otras personas. Esta documentación estará focalizada en comentarios dentro del código fuente y en los documentos de arquitectura global del correspondiente proyecto.
5. Realizar las pruebas unitarias que requiera el desarrollo en función de su complejidad.
6. Realizar las pruebas de integración que requiera el desarrollo en función de su complejidad.
7. Realizar los trabajos en los tiempos establecidos, informando periódicamente de los tiempos imputados y del tiempo restante para su finalización.
8. Informas al responsable de proyecto, analista y/o arquitecto de los posibles puntos de riesgo detectados durante el proceso de desarrollo.

Finalidad 2

Durante la ejecución del proyecto existirán auditorías internas del código desarrollado en las que el desarrollador puede participar con el objeto de auditar los correspondientes desarrollos.

(Nivel de ponderación 10%)

Actividades principales

1. Realizar la auditoría de lo establecido por el auditor, tomando como patrón las guías de desarrollo definidas.
2. Elaborar un informe detallando las no conformidades detectadas durante la revisión.

Finalidad 3

Durante la ejecución del proyecto el área de arquitectura requerirá de apoyo para tareas diversas, como testar ciertas tecnologías, configuración de entornos, implantación de sistemas de integración continua, etc. El desarrollador deberá colaborará en estas tareas bajo la coordinación del responsable de proyecto y el arquitecto.

(Nivel de ponderación 10%)

Actividades principales

1. Coordinar con el arquitecto las tareas a ejecutar (Desarrollo de componentes reutilizables para el proyecto, testar tecnologías,..).
2. Desarrollo de las tareas tecnológicas definidas, bajo la coordinación del arquitecto y el jefe del proyecto.
3. Realizar los trabajos en los tiempos establecidos, informando periódicamente de los tiempos imputados y del tiempo restante para finalización.
4. Informas al responsable de proyecto y arquitecto de los posibles puntos de riesgo detectados durante la ejecución de los trabajos.

Finalidad 4

Dar soporte al responsable de proyecto en los procesos de estimación y planificación en las propuestas de proyectos.

(Nivel de ponderación 10%)

Actividades principales

1. Dar soporte en la estimación de plazos asociada a las propuestas realizadas por los diferentes segmentos.
2. Dar soporte en identificar riesgos asociados al proyecto, estimando cómo afectarán estos en coste y plazo.

Perfil de competencias para el puesto

COMPETENCIA	Importancia	NIVELES			
		I	L	U	O
Competencias Técnicas					
Scrum	Medio			**X**	
Trac	Alta			**X**	
TestLink	Media		**X**		
Diseño Técnicos	Alta			**X**	
Estimación y planificación	Media		**X**		
Desarrollo orientado a Test	Alta			**X**	
Patrones Diseño	Media		**X**		
Inglés	Alta		**X**		
JAVA					
J2EE	Alta			**X**	
SOA	Media		**X**		
Servidor Aplicaciones	Alta		**X**		
Spring	Alta			**X**	
Web	Media			**X**	
ESB	Media / Baja		**X**		

SGBD	Media			**X**	
.NET					
.NET Framework	Alta			**X**	
SOA (WCF)	Media		**X**		
IIS	Media		**X**		
WPF	Alta			**X**	
Web	Medio		**X**		
ESB	Baja		**X**		
SGBD	Media			**X**	
C/C++					
C	Alto			**X**	
C++	Alto			**X**	
Estándares c++98/c++11					
Colas Mensajería(D-Bus, ZeroMQ, Active MQ)					
Frameworks (Qt5, Boots)					
Comunicaciones Serie					
Comunicación ficheros especiales (pipes / fifo)					
Concurrencia					
Sockets					
SGBD					
Sistemas					
Linux	Medio		**X**		
Windows	Medio		**X**		
Virtualización	Medio		**X**		
Alta Disponibilidad	Medio		**X**		
Competencias Personales					
Trabajo en equipo	Alta			**X**	
Autoaprendizaje	Alta			**X**	

3.3.7 Tester

Misión

Dentro del contexto de un proyecto, la misión de este puesto es apoyar al analista durante la definición de las pruebas funcionales, para después proceder a su realización. Durante esta auditoría funcional, se informará, mediante soporte documental, del resultado de los test. Además, deberá apoyar al área de arquitectura en la generación de los diferentes entornos de test.

Finalidad 1

Soporte activo al analista durante la definición de las pruebas, en función de los análisis funcionales previamente establecidos.

(Nivel de ponderación 25%)

Actividades principales

1. Interiorizar los análisis establecidos, identificando durante el proceso las posibles casuísticas funcionales.
2. Escribir juegos de pruebas resultantes del análisis realizado en el paso anterior.
3. Realizar los trabajos en los tiempos establecidos, informando periódicamente de los tiempos imputados y del tiempo restante para finalización.
4. Informas al responsable de proyecto y arquitecto de los posibles puntos de riesgo detectados durante la ejecución de los trabajos.

Finalidad 2

Ejecución de los juegos de pruebas según la planificación establecida por el analista y el responsable de proyecto.

(Nivel de ponderación 60%)

Actividades principales

1. Prepara el entorno de pruebas adecuado, antes de la ejecución de una batería de test.
2. Ejecutar los juegos de pruebas establecidos por el analista y el responsable de proyecto.
3. Ejecutar los juegos de pruebas de mayor nivel, en caso de existir una liberación de versión. Este vendrá determinado por el analista y el responsable de proyecto en función de la planificación del proyecto.
4. Documentar el resultado de las pruebas facilitando esta información al resto del equipo.
5. Realizar los trabajos en los tiempos establecidos, informando periódicamente de los tiempos imputados y del tiempo restante para finalización.
6. Informas al responsable de proyecto y arquitecto de los posibles puntos de riesgo detectados durante la ejecución de los trabajos.

Finalidad 3

Apoyo al departamento de arquitectura en la definición y preparación de los entornos de test necesarios para garantizar la calidad del proyecto.

(Nivel de ponderación 15%)

Actividades principales

1. Coordinar con el arquitecto las tareas a ejecutar (Definición y preparación de entornos de pruebas)
2. Desarrollo de las tareas tecnológicas definidas, bajo la coordinación del arquitecto y el jefe del proyecto.
3. Realizar los trabajos en los tiempos establecidos, informando periódicamente de los tiempos imputados y del tiempo restante para finalización.
4. Informas al responsable de proyecto y arquitecto de los posibles puntos de riesgo detectados durante la ejecución de los trabajos.

Perfil de competencias del puesto

COMPETENCIA	Importancia	NIVELES			
		I	L	U	O
Competencias Técnicas					
Scrum	Medio			**X**	
Trac	Alta			**X**	
TestLink	Alta				**X**
Capacidad análisis	Alta			**X**	
Definición Test	Alta			**X**	
Ejecución Test	Alta			**X**	
Conocimiento Técnico Hardware / Software	Bajo			**X**	
Inglés	Media		**X**		
Sistemas					
Linux	Medio			**X**	
Windows	Medio			**X**	
Virtualización	Medio		**X**		
Alta Disponibilidad	Medio		**X**		
Competencias Personales					
Trabajo en equipo	Alta			**X**	

3.4 Procesos generales

3.4.1 Introducción

Como se ha visto en el la definición general de proceso, descrita al inicio del documento, existe un conjunto de “cajas” denominadas procesos generales, por lo tanto, en este apartado se describe cuál es el objetivo de dicha caja y cómo cumplirlo. El grupo de proyecto deberá seguir el mapa general y

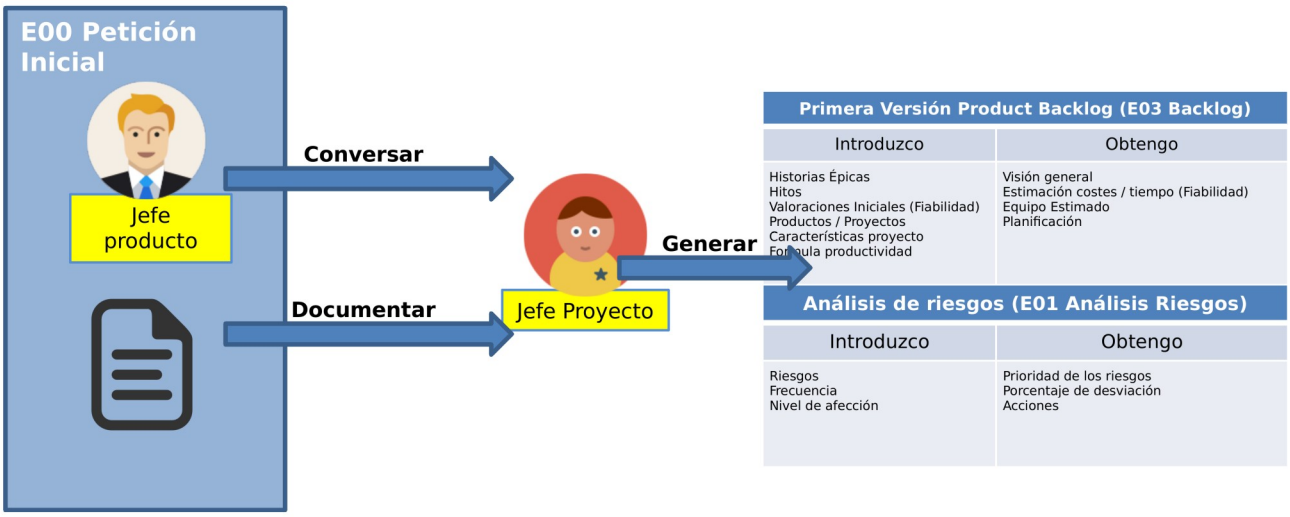
en el momento que se encuentren con un proceso general, deberán recurrir a este apartado para su consulta.

3.4.2 PG2.0.1 Análisis General Hito 1

3.4.2.1 Me cuentan el proyecto….

El responsable de producto, tras entregar el E00. Petición inicial (Las funcionalidades a desarrollar en el año), deberá explicar al responsable de proyecto los objetivos marcados en el documento. Es decir, aunque estos se encuentren escritos, Scrum recalca la importancia de la conversación en los procesos de análisis. El responsable de proyecto deberá interiorizar tanto la documentación recibida como las conversaciones mantenidas, tal y como se representa en la ilustración 4 . Si éste requiere de más procesos de conversación aclaratoria, no deberá dudar en su solicitud, como se ha comentado, la conversación es clave en esta fase del proceso. (Mike Cohin, 2004)

Ilustración 4: Análisis petición E00



El resultado de interiorizar la información recibida por el responsable de proyecto se debe traducir en un backlog (E03 Backlog) y un análisis de riesgos (E01 Aálisis de riesgos), los cuales describiremos a continuación.

Tarea	Entrada	Implicados	Resultado
Recibir información	Conversación responsable de proyecto.	Responsable de producto Responsable proyecto	Análisis inicial de los objetivos

	E00. Petición inicial		

3.4.2.2 *¿Qué definir en la primera versión del product backlog?*

En la definición del product backlog, sobre el que existirá una plantilla, se deberá reflejar la funcionalidad transmitida por el responsable de producto. Ésta deberá estar divididas en los que se denominarán historias épicas, es decir, una lista general de funcionalidades, aportando por cada una la siguiente información:

Ilustración 5: Backlog hito1

- Objetivo historias épicas: Objetivos a conseguir con un alto nivel de definición, siendo estos perfectamente entendibles por el responsable del producto. Para cada historia, además de definir su nombre y descripción, se deberá establecer una prioridad y una estimación con su grado de fiabilidad.
- Prioridad: Las diferentes historias deben estar priorizadas las una respecto a las otras, esto ayuda a determinar por qué empezar primero, de qué podemos prescindir, etc..
- Estimación: Todas las historias deben estar estimadas, aunque sea a nivel general, indicando además el nivel de fiabilidad de la estimación.
- Hitos: Normalmente todo proyecto suele tener hitos de entrega, estos suelen estar sujetos a entregas con clientes, prototipos que desean ser mostrados, etc. Es importante establecer para cada historia el hito al que se encuentra asociado, indicando, si ya lo conocemos, la fecha comprometida de entrega.
- Productos / Proyectos: Todo objetivo se lleva acabo bien porque un producto lo tiene establecido en su hoja de ruta o bien porque un proyecto lo requiere ad-hoc. Es importante conocer está información, ya que nos otorga visibilidad sobre el grado de reutilización del esfuerzo que se va a realizar. Dado que un equipo de trabajo puede llevar más de un producto es importante indicar cada funcionalidad a qué producto atañe, siendo deseable incluso que afecte a más de un producto, ya que de esta forma su grado de reutilización es mayor.
- Características del proyecto: De cara a poder definir el equipo requerido para llevar acabo el proyecto y poder aumentar la fiabilidad de la estimación, será necesario que indiquemos

algunos datos sobre las características del desarrollo que va a ejecutarse. El objetivo es dotar de una plantilla que solicite estos datos al jefe de proyecto, y en base a la experiencia, se aporte información que ayude en la toma de decisiones.

- Formula productividad: El objetivo de esta información es permitir al backlog calcular una estimación sobre cuándo finalizaremos los objetivos marcados para cada hito. Esta formula, al igual que en el punto anterior, se irá construyendo en base a la experiencia haciéndola más óptima en cada iteración.

(Henrik Kniberg & Mattias Skarin, 2010)

Tarea	Entrada	Implicados	Resultado
Primera versión backlog	Análisis inicial de los objetivos	Responsable proyecto	Primera versión product backlog

3.4.2.3 ¿Qué me aporta esta primera versión del product backlog?

Es importante comprender por qué generamos el product backlog y qué nos aporta:

- Visión general del proyecto que vamos a llevar acabo, siendo el objetivo de ésta la tomas de decisiones en una fase temprana del proyecto.
- Estimación de costes / tiempo requerido para su ejecución, aportando además su grado de fiabilidad.
- Equipo estimado requerido para el desarrollo del proyecto. Esta estimación se irá mejorando en base a la experiencia y a la optimización del proceso llevada a cabo en cada iteración. Evidentemente, aquí se propondrá un equipo que podrá ser ajustado por jefe del proyecto si considera que esta estimación no es adecuada.
- Planificación: En base a la información introducida, el backlog deberá ser capaz de mostrar una estimación de cuando se completarán cada uno de los objetivos marcados (historias) y la fecha de finalización de los diferentes hitos. Gracias a esta información, se podrá ver en una fase temprana el grado de cumplimiento /incumplimiento de cada uno de los hitos, sin olvidar el grado de fiabilidad, pudiendo tomar decisiones en cuanto a la dimensión del equipo y la reorganización de las distintas historias.

3.4.2.4 Análisis de riesgos, ¿qué debo definir? ¿qué nos aporta?

En el análisis de riesgos, el cual se definirá en la plantilla Excel correspondiente, debe definirse la siguiente información:

- Riesgos: Riesgo que detectados que puedan surgir durante la ejecución del proyecto.

- Frecuencia: Aquí deberá analizarse la frecuencia con la que se dará el riesgo previamente identificado.
- Nivel de afección: Esta dato informa sobre cómo afecta el riesgo cada vez que se produce.

Gracias a esta información, el análisis de riesgo nos tiene que ayudar a obtener la siguiente información:

- Prioridad de los riesgos: Deberá indicarnos de los riesgos establecidos, cuáles debemos priorizar en base a sus características.
- Ayudarnos a establecer una posible porcentaje de desviación que afecte a la estimación del backlog.
- Ayudarnos a identificar y revisar las acciones que ayuden a mitigar la afección de los distintos riesgos identificados.

Ilustración 6: Análisis de riesgos

Tarea	Entrada	Implicados	Resultado
Primera versión análisis de riesgos	Análisis inicial de los objetivos	Responsable proyecto	Primera versión análisis de riesgos

3.4.2.5 ¿Cómo lo ves?

Una vez generado los documentos requeridos en este análisis general correspondiente al hito 1, debemos comenzar a sacar conclusiones, ya que ahora conocemos con cierto grado de fiabilidad ciertas cosas:

- Costes del proyecto
- Planificación sobre cuando cumpliremos los diferentes hitos de entrega.
- Riesgos que pongan en peligro la viabilidad del proyecto.

Como puede observarse en la ilustración 7, ahora es el momento de tomar decisiones de cara a pasar al siguiente hito y es importante que éstas se tomen de manera conjunta entre el responsable del producto, responsable del proyecto y el jefe de TyD. Estos momentos suelen ser delicados, sobre todo cuando los costes o riesgos impiden la ejecución del proyecto, por eso es importante que cada rol recuerde lo siguiente:

Ilustración 7: Conclusiones hito 1

- Responsable del proyectos:
  - Lo más importante ¡somos un equipo!
  - Dado que es habitual que se pidan más funcionalidades que las soportadas por el presupuesto, debo ayudar a priorizar al responsable de producto. De esta forma, podré determinar que funcionalidades no se llevan acabo sin poner en riesgo el resultado final.
  - Estimar de forma ágil, ayuda a reducir costes y poder tomar ciertas decisiones de forma rápida y ágil. Si luego se requiere una estimación a detalle deberá hacerse, pero es recomendable comenzar con una estimación ágil inicialmente.
  - Debo ayudar al jefe de producto a identificar necesidades que a él puede haber pasado por alto.
  - En ocasiones, los objetivos marcados por el responsable de producto pueden ser demasiado grandes, la división ayuda a identificar partes menos necesarias, y por lo tanto, a optimizar costes.
- Responsable del producto:
  - Lo más importante ¡somos un equipo!
  - Si priorizo y mi presupuesto es elevado, podre determinar que funcionalidades son menos necesarias sin poner en riesgo el resultado final, y por lo tanto optimizar costes.
  - Las estimaciones ágiles ayudan a comenzar cuanto antes, y por lo tanto, ser ágiles en el proceso de desarrollo. Si luego se requiere una estimación a detalle deberá hacerse, pero

es recomendable comenzar con una estimación ágil inicialmente.

- Cuanto más detalle puede aportar, el backlog mejor representará lo que quiero hacer.
- Dividir mis objetivos me ayuda a obtener resultandos antes, me ayudan a confirmas que voy por el buen camino y me ayuda a identificar funcionalidades menos necesarias, y por lo tanto, a optimizar costes.

Tarea	Entrada	Implicados	Resultado
Evaluación hito 1	Product backlog Análisis de riesgos	Responsable producto Responsable proyecto	Evaluación viabilidad proyecto.

3.4.3 PG2.02 Confirmar el equipo de trabajo

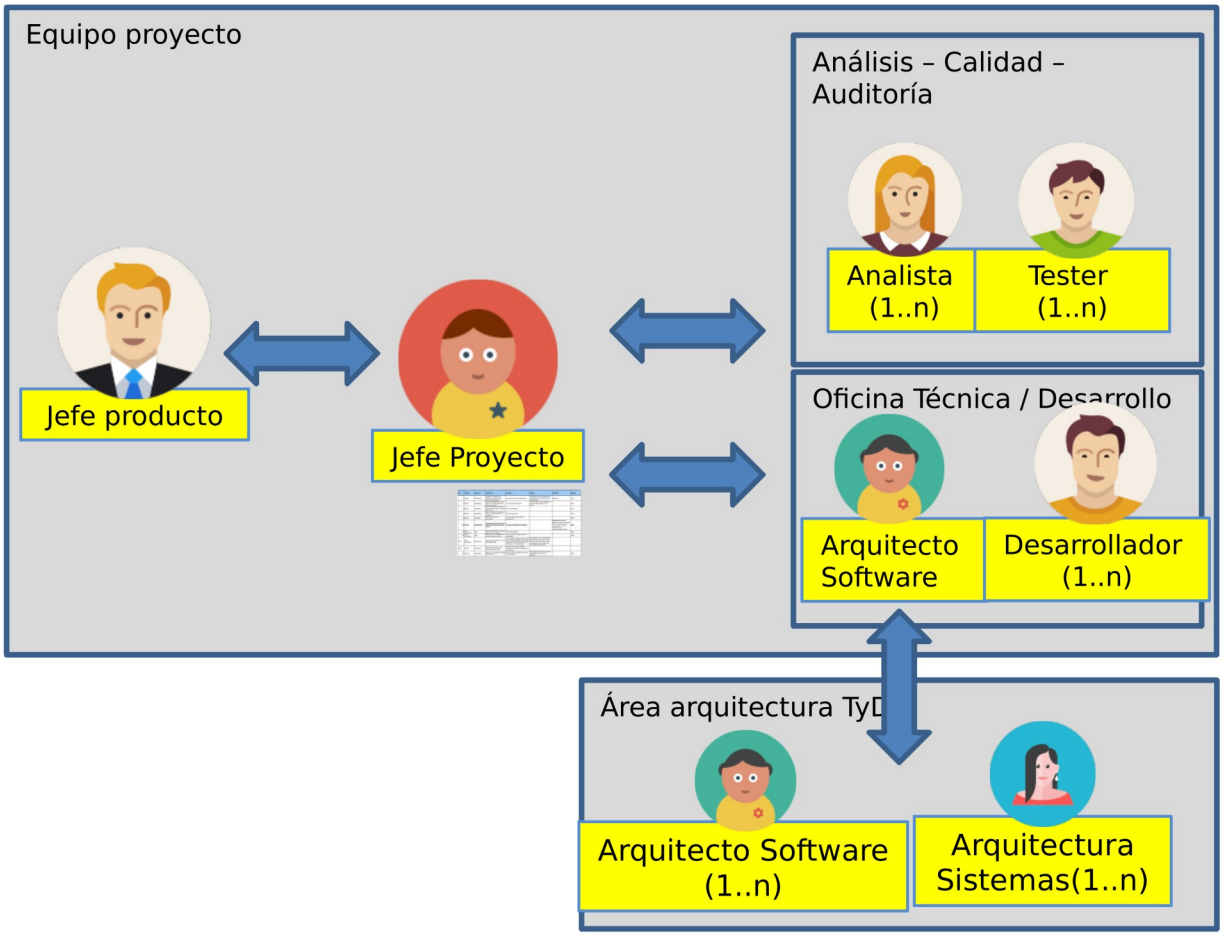
3.4.3.1 Conformar el equipo adecuado

El backlog nos ayuda a dimensionar el proyecto, unido a los hitos de entrega, deberíamos tener suficiente información para establecer al equipo de trabajo. Además de tener un equipo teórico, debemos proceder a su confirmación, es decir, debemos tener los nombre y apellidos de las personas que trabajarán en nuestro proyecto.

En un inicio, se deberán solicitar los recursos internos que den respuesta a las necesidades del proyecto, por cada uno se deberá indicar el porcentaje de uso. En caso de que la compañía no posea los recursos suficientes, se procederá solicitarse a los proveedores previamente homologados. Al igual que en la solicitudes internas, deberá indicarse al proveedor el tiempo utilizado de cada persona.

De cara a seleccionar los perfiles que darán respuesta a nuestros proyectos, tal y como puede verse en la ilustración 8, recordaremos brevemente los establecido en el apartado de Roles. Importante indicar que cada persona puede tener más de un rol.

Ilustración 8: Roles proyecto



- Jefe Proyecto: Coordinar adecuadamente los recursos existentes con el objetivo de alcanzar las metas definidas en el proyecto, con los plazos y costes establecidos. Para ello, durante la ejecución del proyecto, deberá apoyar al dueño del producto en la priorización y definición en detalle de las metas fijadas.
- Arquitecto: Responsable de definir la capa tecnológica requerida para el cumplimiento de los objetivos establecidos, auditando durante la ejecución del proyecto que la construcción se realiza de forma adecuada. Además, deberá dar apoyo tecnológico al equipo de desarrollo, bien mediante soporte técnico como a través de la generación de componentes que mejoren los procesos de desarrollo.
- Analista: Analizar en detalle los objetivos establecidos y priorizados por el responsable del proyecto y el dueño del producto, de forma que permitan la generación de un documento que pueda ser empleado por el equipo para su consecución. Además, deberá auditar el estado funcional de los trabajos realizados de forma que garanticen la calidad final del producto final, generando indicadores que permitan el posterior desarrollo de acciones de mejora.
- Tester: Dentro del contexto de un proyecto, la misión de este puesto es apoyar al analista durante la definición de las pruebas funcionales, para después proceder a su realización.

Durante esta auditoría funcional, se informará, mediante soporte documental, del resultado de los test. Además, deberá apoyar al área de arquitectura en la generación de los diferentes entornos de test.

- Desarrollador: Responsable de llevar acabo los desarrollos planificados según las especificaciones establecidas en el análisis funcional y bajo los patrones de diseño definidos. Estos desarrollos deberán estar debidamente documentos y testados con las pruebas unitarias y de integración que se requieran. De forma paralela, deberá servir de apoyo en las tareas de estimación, arquitectura y auditoría que se requieran durante la vida de los diferentes proyectos en los que participe.

(Àles Alfonso I Minguillón, Eulàlia Clos Cañellas, Humberto Andrés Sanz, Isabel Domènech Puig-Serra, Jordi Schoenenberger Arnaiz, 2009)

Tarea	Entrada	Implicados	Resultado
Conformar equipo trabajo	Product backlog Análisis de riesgos	Responsable proyecto	Equipo de trabajo confirmado

3.4.3.2 ¿Qué ocurre sino es viable el equipo ideal?

En ocasiones, conseguir el equipo ideal no es factible y debemos adaptarnos con un equipo diferente. Si este equipo es capaz de llevar el proyecto adelante, en caso contrario el proceso no podría continuar, deberemos realizar una nueva estimación para conocer el impacto de la nueva realidad. Aprovechando el backlog y el análisis de riesgos generado en el hito 1 realizaremos una estimación ágil para la totalidad de las historias identificadas de como afecta esta nueva situación.

El resultado de esta nueva situación se contemplará con el responsable de producto, requiriendo una estimación a detalle si se viera en peligro la viabilidad del proyecto. Este proceso sería similar a lo establecido dentro del apartado 2.4.2.5.¿Cómo lo ves?

Tarea	Entrada	Implicados	Resultado
Evaluación impacto equipo trabajo	Product backlog Análisis de riesgos Equipo trabajo	Responsable producto Responsable proyecto	Product backlog actualizado Análisis de riesgos actualizado Evaluación viabilidad proyecto.

3.4.4 PG2.03 Diseño Arquitectura por componente o grupo de componentes.

3.4.4.1 Generar un primer diagrama de componentes

Dentro del hito 2 se debe realizar un diseño general que represente la arquitectura que va a seguir nuestro proyecto. El método para la realización de este diseño no viene fijado, puede hacerse mediante un diagrama de componentes UML o con el método que resulte más cómodo, el objetivo es poder representar visualmente los componentes software que conforman nuestro sistema.

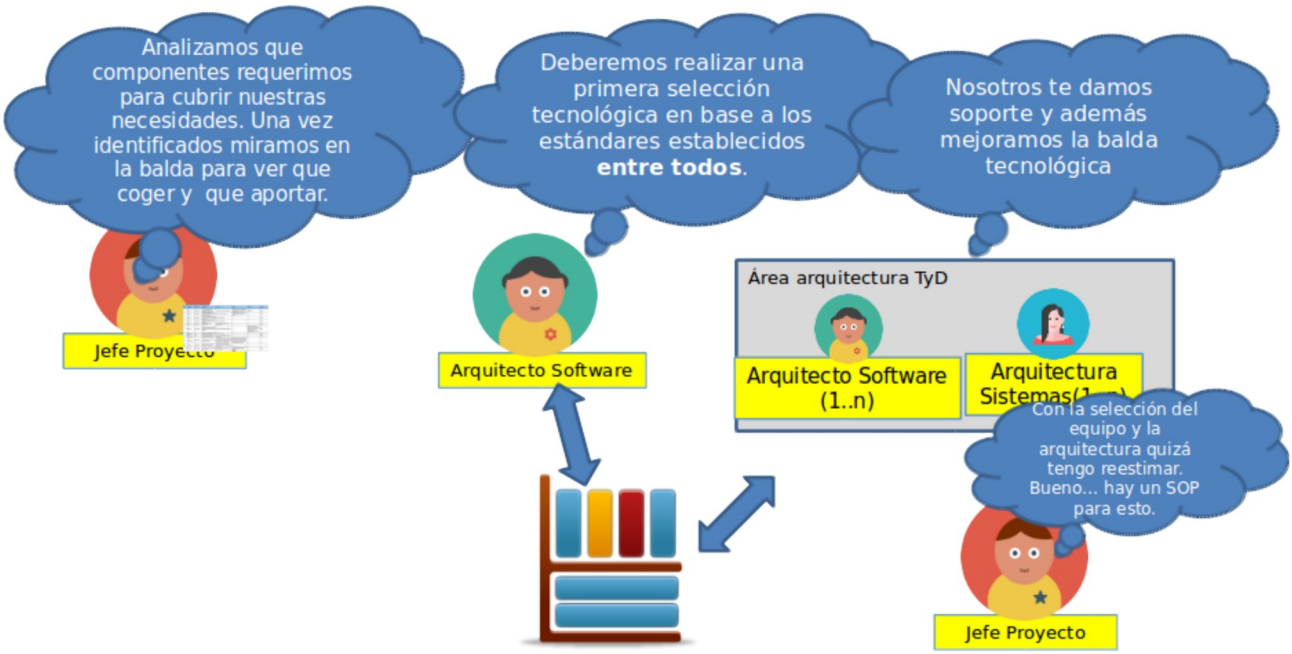
El diagrama seleccionado debería mostrar:

- Componentes software que conforman el sistema, reflejando cuales pertenecen al producto, cuales pueden ser reutilizados y cuales estoy reutilizando de la balda.
- Componentes de terceros, como bases de datos, LDAP, ERP, sistemas de terceros, etc.
- Comunicación entre los diferentes componentes y protocolo de comunicación
- Hardware asociado para cada componente y sistemas operativos soportados por cada componente.
- Requisitos requeridos por el sistema.

Por cada componente se deberá indicar las plataforma tecnológica sobre el cual serán implementados. Esta plataforma tecnológica se corresponderá con la seleccionada previamente por el área de arquitectura, en caso de que nuestro proyecto requiera de tecnologías no estandarizadas, deberá indicarse para su posterior estudio.

Una vez se disponga del diseño, éste se deberá validar con el área de arquitectura para su validación. En el la ilustración 9 puede observarse cómo sería el proceso.

Ilustración 9: Validación arquitectura



En este proceso se realizan varias acciones:

- En función de las necesidades del proyecto, se identifican posibles componentes ya desarrollados (“Balda de componentes”) que puedan reutilizarse en el presente desarrollo. De esta forma, se optimizan los costes y minimizan los riesgos.
- Se identifican posibles nuevos componentes reutilizables que puedan ser utilizados en otros proyectos, permitiendo aumentar la balda tecnológica.
- Se validan las tecnologías seleccionadas y su forma de utilización. En caso de encontrase con una nueva tecnología se validará por el área de arquitectura, esta situación podrá generar los siguientes supuestos:
    - Se ve interesante la nueva tecnología y se procede a su homologación, este proceso quedará descrito en próximos documentos (PEC 3).
    - La funcionalidad que nos aporta la nueva tecnología se solapa con otra ya homologada. En el caso de no existir ninguna razón de peso que requiera su sustitución, se solicitará el uso de la plataforma homologada.
    - La funcionalidad que nos aporta la nueva tecnología actualmente no se encuentra cubierta pero la plataforma elegida no es la idónea. En esa situación, el área de arquitectura seleccionará una nueva plataforma que se adapte a las necesidades del departamento y que a su vez de respuesta al proyecto.
- Se validan el uso de arquitectura seleccionada, pudiendo identificarse acciones correctivas

durante este proceso.

Tarea	Entrada	Implicados	Resultado
Diagrama arquitectura inicial	Product backlog Análisis de riesgos	Responsable proyecto Arquitecto	Diagrama arquitectura inicial
Validación diagrama arquitectura inicial	Diagrama arquitectura inicial	Responsable proyecto Arquitecto Área arquitectura	Diagrama arquitectura inicial validado Componentes reutilizables

3.4.4.2 Nuestra Balda de componentes

El área de arquitectura, conformado por un arquitecto de cada equipo, será la responsable por nutrir y mantener los componentes reutilizables. El objetivo de esta balda, es que sus componentes se reutilicen en los diferentes proyecto con el objeto de optimizar costes, ya que en ocasiones las necesidades de los diferentes equipo son altamente similares.

El contenidos de esta balda puede estar conformado por los siguientes elementos:

- Patrones de diseño: Formas de cómo hacer las cosas. De esta manera, cuando un equipo tenga que utilizar una nueva tecnología o enfrentarse antes un nuevo escenario de desarrollo, podrá contar con las experiencia previa de otro equipo.
- Librerías: Librerías que servirán de base para el desarrollo de ciertas funcionalidades.
- Componentes funcionales: Componentes que ofrecen una funcionalidad específica, como por ejemplo la gestión de usuarios, y que pueden incorporarse a nuestros desarrollo evitando por tanto su desarrollo.

3.4.4.3 ¿Qué ocurre si tras la revisión tecnológica hay cambios significativos?

En teoría, tras la revisión tecnológica el resultado debería ser positivo, es decir, se han identificado componentes de desarrollos anteriores que podemos reutilizar y por lo tanto se minimizan los riesgos y los costes. Por desgracias esto no siempre se cumple, en ocasiones en la primera revisión se minimizan ciertos riesgos tecnológico que tras la revisión salen a la luz, esto puede causar desviaciones significativas en la estimación inicial.

El resultado de esta nueva situación se contemplará con el responsable de producto, requiriendo una

estimación a detalle si se viera en peligro la viabilidad del proyecto. Este proceso sería similar a lo establecido en 2.4.2.5.¿Cómo lo ves?

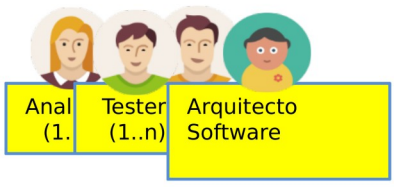
Tarea	Entrada	Implicados	Resultado
Evaluación impacto arquitectura	Product backlog Análisis de riesgos Diagrama arquitectura inicial validado Diagrama arquitectura inicial validado	Responsable producto Responsable proyecto	Product backlog actualizado Análisis de riesgos actualizado Evaluación viabilidad proyecto.

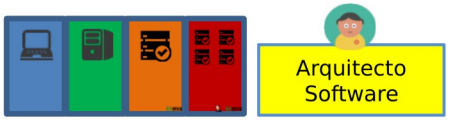
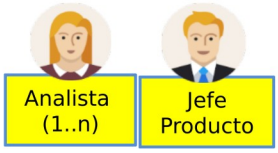
3.4.5 PG2.04 Sprint 0

3.4.5.1 Prepararlo todo

El objetivo de Sprint 0 (Fase 0), es estar preparado antes de empezar el proyecto, en ocasiones, el proyecto se inicia de forma acelerada y se detectan carencias que impiden al equipo avanzar incurriendo en importante costes. Esta fase intenta evitar esto, en este sprint se prepara todo lo necesario para que el equipo sea efectivo desde un inicio y no existan bloqueos en el desarrollo.

Normalmente, la necesidades de los distintos equipos son altamente similares, por esta razón, se ha elaborado una tabla a modo “checklist” que permitirá a cada responsable de proyecto validar que se han cumplido con los requisitos habituales antes de comenzar el desarrollo. Evidentemente, cada proyecto tiene sus particularidades, por esta razón, este checklist puede ampliarse o acortarse en función de la realidad de cada equipo.

Elemento		Descripción
Equipo		Debemos explicar nuestro backlog al equipo de trabajo, el objetivo es que a partir de este momento todos tengamos un nivel de conocimiento similar. Además debemos informar a los integrantes la fecha de inicio del proyecto (Sprint 1).
Entorno Desarrollo (Adaptar plantilla preexistente)		El arquitecto deberá preparar un entorno de desarrollo estandarizado para todos los

		miembros del equipo. Se deberá intentar basar en un estándar existente. (Guías área arquitectura)
Determinar SVN, Track, carpetas de red	SOP2.12 Rellenar solicitud SVN-Track	Deberemos de preparar el repositorio SVN para la gestión de versiones, Track para la gestión de incidencias y las carpetas de red para albergar la documentación del proyecto.
Permisos		Deberemos otorgar los permisos adecuados a los integrantes del grupo.
Entorno de pruebas		Debemos definir y documentar nuestros entornos de pruebas. Es importante contar con un entorno de integración, para pruebas durante el proceso de desarrollo, y con un entorno de preproducción para pruebas finales (FAT). Este apartado quedará descrito con mayor detalle en la PEC 3.
Product Backlog		Hasta ahora tenemos historias épicas (Objetivos de cierto tamaño), tenemos dividir estos en objetivos más pequeños, alcanzables en un sprint (iteración), para los n (n>=2) primeros. Además, debemos tener la totalidad de las historias priorizadas, estimadas (fiabilidad) y con sus correspondientes hitos (fecha).
Especificar n primeros sprints		Detallaremos las historias que vayamos a abordar en los primeros n sprints (n>=1).

		La forma de cómo realizar la documentación oportuna quedará descrita en la PEC3.
Diseño n primeros Sprint		Deberemos realizar los diseños relativos a las historias a abordar en los n primeros sprint (n>=1) La forma de cómo realizar la documentación oportuna quedará descrita en la PEC3.

Una vez se haya cumplido con este checklist, estamos en disposición de comenzar con el proyecto, es decir, comenzar con el primer 1 (primera iteración)

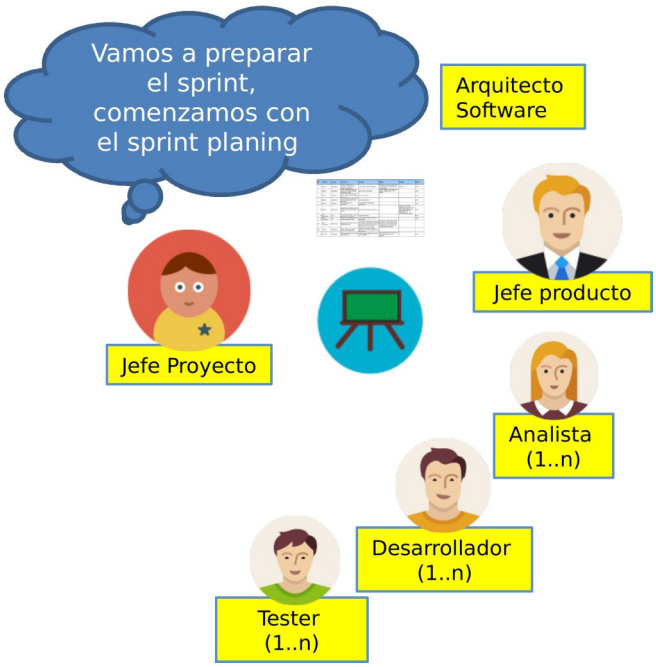
(Henrik Kniberg, 2007)

(Henrik Kniberg & Mattias Skarin, 2010)

3.4.6 PG2.06 Preparación Sprint N

Una vez superado todos los pasos previos (hito1, hito 2, sprint 0,…) ya ha llegado el momento de abordar la ejecución del proyecto. Basándonos en una metodología ágil como Scrum, aquí comienza un ciclo iterativo de pequeños proyectos, denominados sprints, de una duración fija, de 2 a 4 semanas. Cada sprint, tendrá una planificación, una fase de desarrollo y un resultado final demostrable que se enseñara responsable del producto.

Ilustración 10: Sprint planning



Es decir, cada dos semanas se dispondrá una nueva versión del producto, de forma el equipo pueda ver si va por el buen camino o bien tiene que cambiar de rumbo. En el siguiente sprint se revisa el backlog, se determina si debe cambiar sus objetivos funcionales, priorizar de nuevo sus historias, etc. Es exactamente esto lo que se realiza en este proceso general, preparar el backlog entre todos para determinar que se va a desarrollar en el presente sprint, lo que se denomina sprint planing, tal y como se muestra en la ilustración 10.

Aquí se establece una pequeña guía de cómo debe realizarse el sprint planing. Este puede hacerse bien en la pizarra o en el Excel, según las preferencias del equipo.

1. El equipo marca que historias (funcionalidades) va a abordar, el número de historias deberá determinarse por la velocidad media del equipo (puntos de historia por sprint – importante medir). En caso de no tener este dato, el equipo es nuevo o la metodología Scrum está en proceso de implantación, el grupo decidirá cuantas historias meter estimando una capacidad teórica.
2. Se coge la primera historia por orden de prioridad y se explica al resto del equipo por parte del analista/ arquitecto / jefe proyecto (según corresponda).Toda historia que decida abordarse debe poder terminarse en un sprint, en caso de no ser así es aconsejada su división en objetivos que puedan finalizarse en una iteración.
3. La historia se divide en tareas, esto debe realizarse entre todos los integrantes del equipo.
    - A veces es complicado dividir, requiere reuniones de diseño previas, **por eso es recomendable que el diseño y en análisis se realicen en sprint anteriores**. Si no ha sido así, que no cunda el pánico, se establece una tarea de diseño y se determinarán las tareas durante la propia ejecución sprint, al inicio.

- Las historias que sean complejas, bien por dificultad técnica (diseño desconocidos, alto grado de afección al código existente,..) o funcional, requerirán tareas adicionales de análisis y diseño que se traducen en reuniones entre los implicados (analistas / arquitecto / desarrolladores).
- Sospechar de las historias que tienen una única tarea, ya que este caso, puede significar que no se ha dividido lo suficiente y por lo tanto será más complicado distribuir el trabajo entre varios y ver el avance de las misma, o bien, la historia no es una historia como tal y realmente es una tarea.

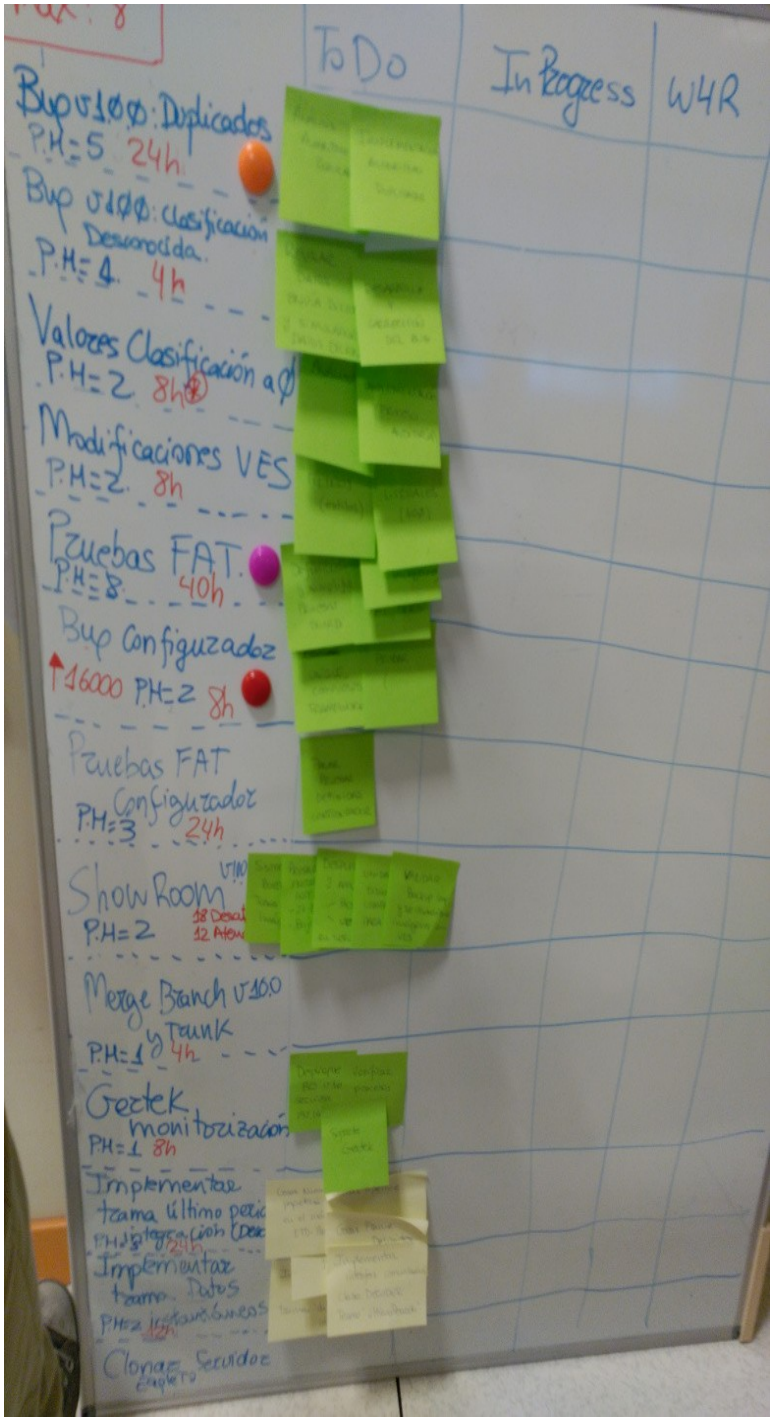
4. Una vez definidas las tareas, debe estimarse entre todos la historia de forma independiente, la estimación se realizará en puntos de historia. Los puntos de historia no es tiempo, es esfuerzo, se corresponde con un valor totalmente subjetivo que tiene mucho significado para el equipo. El equipo da un valor, siguiendo la serie Fibonacci (1,2,3,5,8,13, 21, infinito), en función del esfuerzo a una tarea habitual que conoce muy bien, por ejemplo 3. Cuando llegue otra tarea se le asignará otro valor comparando ésta a la tarea habitual, a la cual que se asigno el valor 3. Se exponen algunos consejos para llevar acabo esta labor:
   - Usar la serie Fibonacci 1,2,3,5,8,13.., consejo, no es bueno meter en un sprint historias superiores a “8”, el valor depende de la forma de medir del equipo, ya que se corre el riesgo de no terminarlas.
   - Elegir un objetivo de tamaño medio que entre en un sprint y que tengamos controlado (sabemos lo que nos cuesta) , a ese asignarle un valor de la serie Fibonacci. Ahora, todo lo que sea más sencillo estará por debajo y lo que sea más complicado estará por encima.
   - Esto nos costará hacerlo bien, pero se mejora por iteración.
   - Todos los miembros deben estimar de forma independiente y debe escucharse el valor dado por cada uno, esto se denomina <<Planning Poker>> . Un consejo es usar cartas especiales para cumplir este objetivo, ya que si alguien escucha el valor de otro miembro suele tender a cambiarlo, si ya se cogió una carta, esto no es posible. Si algún integrante del equipo da un valor muy diferente al resto, es por alguna razón, es importante preguntarnos ¿por qué?
5. Una vez se haya estimando en puntos de historia y con todas las tareas encima de la mesa, y sólo entonces, indicaremos cuantas horas necesitamos para llevar acabo las tareas identificadas (tener en cuenta un riesgo).
6. Esto se deberá repetir con cada historia (**en orden de prioridad**) que se incluya en el sprint hasta que llevemos el número suficiente de historias. El número de historias debería basarse en nuestra velocidad media (puntos historia por sprint).
7. El responsable de producto debe acudir a éste sprint planing, ya que su papel es fundamental en el proceso de definición y priorización.

El sprint planing lo podemos realizar en una pizarra o bien en Excel, ambos formatos son validos.

Lo importante es elegir aquello que permita que Scrum nos aporte lo más posible. Aquí van algunos observaciones (en base a la experiencia) para elegir uno o otro:

- Pizarra (Ilustración 11)

Ilustración 11: Ejemplo sprint planing



  - Más fácil para incorporar tareas nuevas durante el trascurso del sprint. En varios grupos que usan Excel es habitual decir, lo modifico y os aviso, no viéndose en algunos casos hasta el día siguiente.

    - Es muy visible en todos momento por todos los miembros del grupo (siempre que estén en la oficina).
    - Aumenta la participación del equipo.
    - Dinamiza los Dailys.
- Excel:
    - Permite visualizar cuando mi equipo está distribuido.
    - Facilita la automatización de indicadores.
    - Es más fácil de llevar cuando nos movemos entre ubicaciones diferentes..
    - Aumenta la descripción que puedo poner a una tarea. En la pizarra, lo que me quepa en el postit, en el excel, lo que requiera (texto, links,..)

(Henrik Kniberg, 2007)

(Henrik Kniberg & Mattias Skarin, 2010)

Tarea	Entrada	Implicados	Resultado
Preparar Sprint	Product backlog	Todo el equipo	Funcionalidades a desarrollar sprint n

3.4.7 PG2.07 Ejecutar Sprint N

3.4.7.1 Sprint N una introducción

En este proceso general se explica cómo llevar acabo la ejecución de un sprint y qué tareas deben realizar todos los integrantes de un equipo. El objetivo y explicar qué hacer y cómo hacerlo, y además que éste documento también vaya mejorando en función de la experiencia de los distintos grupos.

Una parte muy importante dentro de Scrum son la reuniones diarias,también denominadas dailys, ya que nos ayudan a la gestión ágil de problemas durante la ejecución del proyecto. Por esta razón, se dedicará todo un apartado a explicar su funcionamiento.

Aunque el equipo debe ser flexible, capaz de abordar las tarea que nuestros objetivos requieran, expondremos por rol qué tipo de tareas deben llevarse acabo. Este apartado, unido al referente a roles, ayudarán al equipo a conocer sus responsabilidad y trabajar de forma más coordinada.

Si entre los objetivos se encuentra la liberación de un versión para producción, explicaremos que

debe abordarse para su ejecución y cuál deberá ser el resultado obtenido. Esto no suele ser en ningún caso algo sencillo y puede requerir un esfuerzo importante por parte del equipo.

Una parte importante de Scrum es la mejora continuar, aquí explicaremos cómo podemos llevar esto acabo mediante las reuniones de retrospectiva.

(Henrik Kniberg, 2007)

(Henrik Kniberg & Mattias Skarin, 2010)

3.4.7.2 Daily

Cada día el equipo se junta un máximo de 15 minutos, aquí se expone lo realizado en el día anterior y lo que va a acometerse. Es importante es reflejar si tenemos o hemos tenido algún problema durante la ejecución de los trabajos. El objetivo principal es que todos los integrantes conozca como se encuentra el pryecto.

Las reuniones diarias permiten ser ágiles ante la existencia de problemas, poniendo solución en el momento que se produzcan. La resolución no debe realizarse en el daily, ya que se estaría bloqueando a todo el grupo, pero si que debe organizarse aquí (Quiénes las resuelven).

- Los problemas son del equipo y los resuelve el equipo. Personalizar un problema es negativo para el equipo.
- En los problemas es muy importante el preguntar el por qué (buscar 5 por qué) . No resolverlo de raíz provoca que vuelva.
- Tipos de remedios
    - Contenedor: Tirita para no bloquear, pero debemos planificar su resolución.
    - Corrector: Corrige el problema.
    - Preventivo: Si conocemos la razón será más complicado que vuelva a ocurrir.
- Muchas veces los problemas pueden ocurrir por carencias en las especificaciones técnicas y/o funcionales. En este caso deben juntarse (tareas nuevas) el arquitecto y/o analista con los desarrolladores correspondientes y conversar (documentando cuando sea necesario).

Cuando se acaba una tarea, se marca y se coge la siguiente en orden de prioridad, y esto puede hacerse de forma automática y repetitiva. Aunque no está de más repasar esto en el daily para detectar problemas antes de iniciar un nuevo trabajo.

Es importante llevar un control de a qué se está dedicando el tiempo, de forma que, luego pueda medirse y sacar conclusiones. De la misma forma, si en ocasiones distraen al personal de del

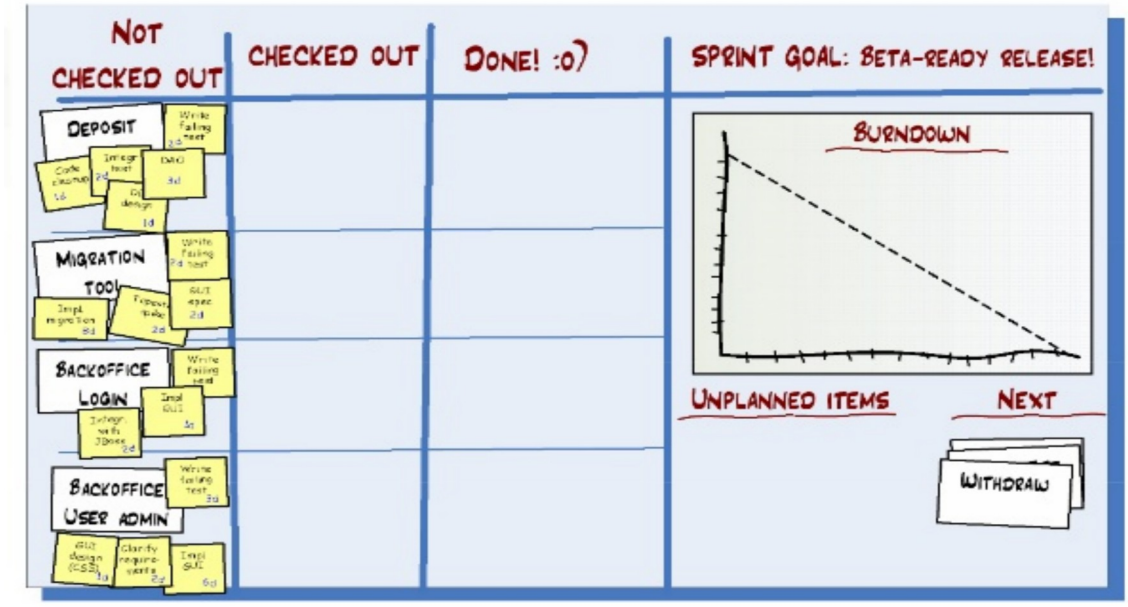
trabajo planificado y se percibe que está situación se repite con frecuencia, es bueno registrarla para ver el impacto y detectar si realmente es un problema y afecta al equipo.

3.4.7.3 Ejecución sprint

En el sprint se ha establecido los objetivos que se van a llevar acabo durante el sprint, además estos se han dividido en tareas y se puesto en la pizarra ordenados por prioridad. Si se hace uso de la pizarra(ilustración 12) cada linea horizontal corresponde con una historia (objetivo a conseguir), cada post-it es una tarea que permitirá la consecución de un objetivo. Dado que cada historia está ordenada por prioridad, el equipo de trabajo deberá coger de forma autónoma un post-it que no esté siendo ejecutado por nadie y llevarlo acabo.

Es decir, cada miembro cogerá un post-it (el de mayor prioridad) de la primera columna y lo llevará a la segunda para indicar que ese trabajo se está llevando acabo. Cuando una tarea quede finalizada, ésta se llevará a la última columna, si todas las tareas se encuentran en la última columna significará que el objetivo está conseguido.

Ilustración 12: Pizarra sprint



La pizarra indica en todo momentos cómo vamos, quién está haciendo qué, etc. Ayuda a que el trabajo se organice de forma automática, yo, cuando termino una tarea se cual puedo coger, ya que están ordenadas por prioridad. Esto lleva a otra pregunta ¿todo el mundo puede hacer todo? lo deseable es que sí, la realidad, por desgracia, en un poco diferente. Aunque la gente va evolucionando, se ha estandarizado una serie de roles y la gente asume las tareas en base a su perfil, la evolución de cada integrante permitirá que cada uno pueda asumir nuevos roles, y por lo tanto pueda gestionar nuevas tareas.

A continuación se muestra por cada rol que tipo de tareas debe llevar acabo dentro de la ejecución de un sprint.

Analista

Analizar las historias épicas para su división en historias más pequeñas, de tal forma que entren en un único sprint. Es decir, el responsable de producto genera historias épicas, es labor del analista, dividir estos objetivos grandes en objetivos más pequeños que el equipo pueda lleva a termino en un sprint.

Detallar las historias según los establecido en la correspondiente guía de análisis. Este detalle debe ser comprendido y validado por el responsable de producto, además, debe servir también a los desarrolladores para la implementación de las historias, por este motivo, debe escribirse con uno o dos sprints de anterioridad.

Además del documento escrito, debe conversarse con lo desarrolladores para explicar las distintas historias, esto aumenta el nivel de compresión y por tanto la calidad. Importante también dar soporte a las dudas funcionales que se originen durante el desarrollo de las historias.

Debe coordinar, generar y testar las pruebas asociadas a cada historia. Toda historia llevará asociado un conjunto de test que validarán su correcto funcionamiento. La generación de pruebas se explica con mayor detalle en la correspondiente guía.

Responsable de proyectos

Mantener el backlog debidamente actualizado, incorporando historias nuevas y estableciendo la división de historias (historias épicas a historias) detectadas por los analistas. Todas estás deberán estar priorizadas según las necesidades de negocio y estimadas con su correspondiente fiabilidad.

Dado que todo proyecto posee hitos pactados de entrega, deberá revisar estos analizando el grado de cumplimiento e informado de inmediato en caso de existir posibles retrasos.

El responsable debe ser un facilitador que permite que el equipo trabaje de forma fluida ante posibles problemas que puedan surgir en el día a día. Por lo tanto, debe servir de soporte antes todos los miembros del equipo, ayudando a que éstos puedan centrarse en exclusiva a las tareas definidas. Ante problemas, debe coordinar a los diferentes miembros para posibilitar un trabajo coordinado en equipo que minimice el impacto de estos.

Debe dar soporte a los responsable de producto en la priorización e identificación de historias ya que tiene una visión general del todo el proyecto.

Arquitecto

Definir los patrones de diseño (“cómo desarrollar las cosas”) que deberán seguirse por los desarrolladores en la implementación de las distintas historias. Además, deberá extender el

conocimiento de estos patrones de forma ágil para que sean reutilizados automáticamente.

Seleccionar los componentes tecnológicos de terceros , de forma coordinada con el área de arquitectura, que serán utilizados. Además de su elección, deberá extender su utilización entre los desarrolladores, bien a través de formación, de ejemplos, ambos, etc. En paralelo, servir como soporte tecnológico a los desarrolladores y analistas en la historias complejas o en la dudas que surjan en los procesos de desarrollo.

Supervisión de las historias definidas con el objeto de detectar posibles problemas técnicos que impidan su desarrollo. Además, auditar, o coordinar la auditoría, de los desarrollos realizados con el objetivo de asegurar que estos siguen lo estándares definidos las distintas guías publicadas por el área de arquitectura.

Por último, deberá potenciar la automatización de las pruebas, ya que la ejecución de éstas es una tarea costosa, tanto a nivel de esfuerzo como económico. Por esta razón, deberá apoyar y potenciar la construcción de pruebas automáticas basándose en las pruebas generadas por los analistas.

Desarrollador

Desarrollar las historias especificadas por el analista según los estándares tecnológicos y patrones definidos por el arquitecto. Solicitando reuniones técnicas (arquitecto) / funcionales (analista) en caso de detectar que una historia tiene un impacto mayor del inicialmente estimado. Cada desarrollo será validado definiendo las pruebas y test unitarios correspondientes según las guías asociadas a la gestión de pruebas.

En caso de problemas durante los diferentes desarrollos deberá reportar los problemas en las reuniones diarias con el objeto de que estos se gestionen debidamente. En ningún caso deberá asumir personalmente el problema no comunicando éste al equipo y evitando que se tomen decisiones consensuadas.

Tester

Colaborar con el analista en la generación de test que permitan validar las diferentes historias. Este proceso se explica con mayor detalle en la correspondiente apartado (Auditoría funcional). Una vez definidos, deberá pasar las distintas pruebas, informando de sus correspondientes resultados.

Preparar los entornos para la ejecución de las distintas pruebas. Además es responsable de mantenerlos “limpios y preparados”.

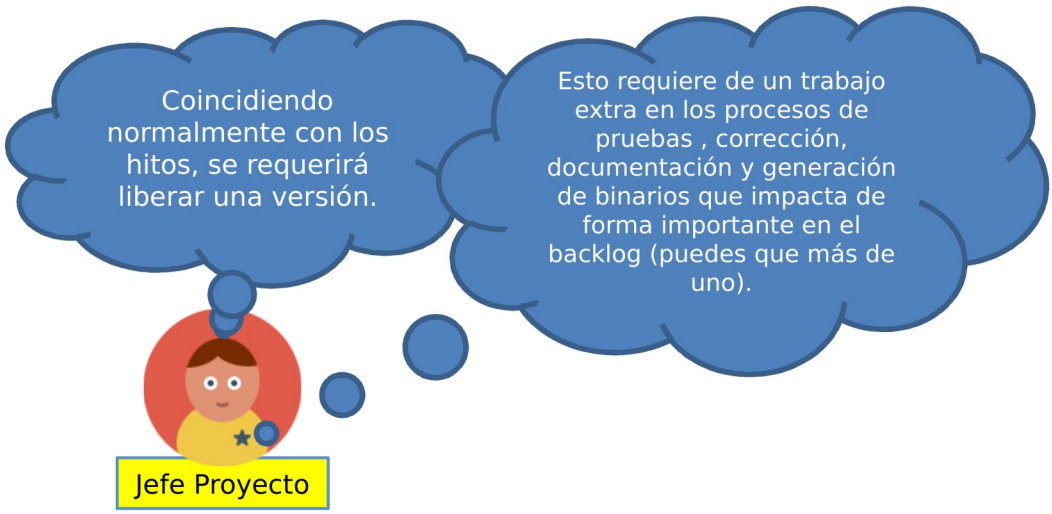
Tarea	Entrada	Implicados	Resultado
Ejecutar Sprint	Sprint planing	Todo el equipo	Objetivos marcado en el sprint planing

3.4.7.4 Liberar versión – Pruebas de integración

Coincidiendo con los hitos de entrega, normalmente se liberara una versión para su instalación en un entorno productivo. Esto requerirá de un trabajo extra en los procesos de pruebas, corrección, documentación y generación de binarios que impacta de forma importante en el backlog. Como ahora se podrá observar, se requerida de un trabajo coordinador por todo el equipo liderado por el jefe de proyecto.

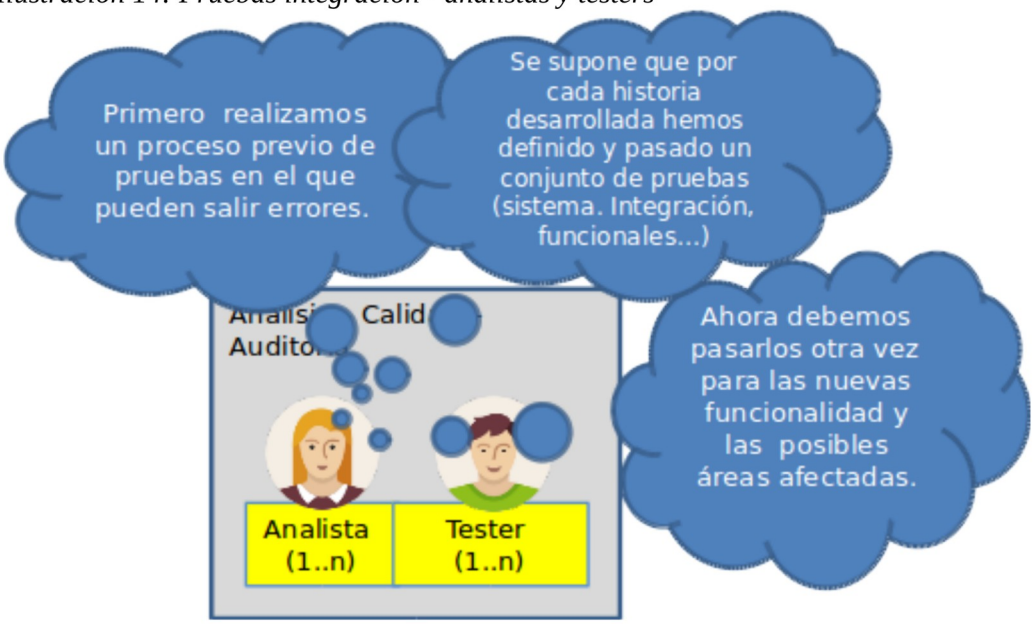
Como se puede ver en la ilustración 13, el responsable de proyecto es el encargado de gestionar este proceso que afecta de forma notable al equipo, ya que debe centrarse en asentar los diferentes evolutivos asociados a la versión que va a liberarse.

Ilustración 13: Pruebas integración - responsable proyecto



El grupo de análisis / test deberá repetir la pruebas realizadas en los nuevos desarrollos, probando además áreas que hayan podido verse afectadas. Esto se verá refrendado por una serie de test generales que validen que no existe ningún problema bloqueante. Este complicado proceso se detallará más adelante, en el área correspondiente a la auditoría funcinal . En la ilustración 4 se muestra las tareas a las que deben responder en esta fase.

Ilustración 14: Pruebas integración - analistas y testers



El equipo de desarrollo deberá reservase un tiempo importante para la realización de desarrollos correctivos en función de los errores encontrado durante las pruebas.

(Carlos Blé Jurando y colaboradores, 2010)

Tarea	Entrada	Implicados	Resultado
Pruebas integración	Desarrollos Test Documentos de análisis	Todo el equipo	Versión prevalida

3.4.7.5 Liberar versión – Pruebas FAT

Una vez que las validaciones previas se hayan realizado correctamente, deberá ejecutarse las pruebas de aceptación en fábrica (FAT). Esto requería congelar previamente el código por medio de un tag en el SVN, tal y como se especifica en el SOP 2.20 Generar Versión Test, para después, volver a ejecutar el conjunto de pruebas especificas a la funcionalidad implementada, áreas afectadas y un conjunto de pruebas generales que verifiquen el buen funcionamiento del resto de áreas teóricamente no modificadas.

En teoría, en esta fase los analistas y testers no deberían encontrar ningún error grave o bloqueante, ya que estos tenían que haber sido detectados en las pruebas de integración. Estas pruebas son determinantes porque establecen si ya puede o no liberarse una versión. Una aplicación con bugs pueden ser liberada, siempre y cuando estos sean de carácter leve o estético, estén controlados y se planifiquen su corrección en futura releases.

Aunque por prevención haya que reservar horas del equipo de desarrollo para correcciones, en teoría están no deberían ser necesarias. Si finalmente su usarán, en función del tipo de corrección podrían obligar a repetir las pruebas FAT completas.

Tarea	Entrada	Implicados	Resultado
Pruebas FAT	Versión prevalida Test Documentos de análisis	Todo el equipo	Versión validada

Si todo sale bien, liberaremos versión, siguiendo lo establecido en la SOP2.14 Crear Versión, además toda versión deberá ir acompañada de la siguiente documentación:

- Manual de usuario
- Manual de instalación
- Manual de mantenimiento
- Binarios
- Plan de formación
- Notas de la versión.

(Carlos Blé Jurando y colaboradores, 2010)

3.4.7.6 Resultado Sprint

En cada sprint, con independencia de que exista liberación, debe existir una funcionalidad demostrable y probada. Esta nueva funcionalidad debe mostrarse, a través de una demo, al responsable del producto para que valide el resultado. En caso de que el responsable no pueda asistir a la reunión, se preparará un entorno de demo para que éste pueda acceder y validar las mejoras realizadas en el sprint.

Tarea	Entrada	Implicados	Resultado
Validar Sprint	Versión demostrable	Responsable producto	Producto backlog

3.4.7.7 Retrospectiva

Antes de empezar con la retrospectiva, debe analizarse cómo ha ido el sprint.

- Puntos de historia completados / Puntos de historia estimados.
- Horas incurridas.
- Grado de fiabilidad de nuestra estimación.
- Velocidad del equipo (puntos de historia)
- Actualización hitos en función de la velocidad del equipo

  ….

Este tipo de información debe extraerse a final de cada sprint de forma que la retrospectiva sea lo más efectiva posible. Esto es lo que denominaremos indicadores y nos permitirán medir cuál es nuestra velocidad en cada sprint (puntos de historia completados), nuestro nivel de mejora de un sprint a otro, y lo más importante, conocer objetivamente como impacta cada medida adoptada.

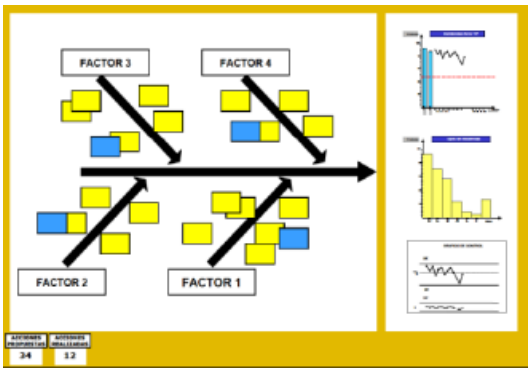
El numero de puntos de historia completados tiene gran importancia dentro de este paso, ya que si se acumula este valor sprint a sprint, se podrá obtener la velocidad media del equipo. Esta velocidad media me ayuda a posteriormente a estimar. Si se valoran todas los objetivos en puntos de historia y conozco mi velocidad media, podrá conocerse la fecha aproximada de finalización. A la hora de acumular los puntos de historia finalizados en cada sprint, sólo podrán sumarse aquellos correspondientes a historias completamente terminadas.

En la retrospectiva, se tratarán los denominados problemas crónicos, es decir, lo que siempre nos acompañan de forma constante. Todo el equipo generará una lista priorizadas con los problemas crónicos, posteriormente se escogerá aquel que tenga la prioridad más alta. Una vez que elijamos el problema que más nos afecte se deberá medir y fijar un objetivo, ya que será la única manera de realizar un seguimiento adecuado.

En un siguiente paso, cada miembro del equipo indica las causas (indicando su nombre) por las que se cree que se produce dicho problema, de entre todas las causas se eligen las más probables. En cada una establecemos las acciones que deberían poner remedio a las mismas, indicando su

responsable y fecha. Estos pasos deberá realizarse en un tablón con un formato similar al mostrador en la ilustración 15.

Ilustración 15: Problemas crónicos



Semanalmente, se revisa cómo las acciones han impactado en las productividad del equipo, estableciendo nuevas acciones si así lo vemos necesario. Como se ha comentado antes es muy importante medir y representar está mediciones en el tablón.

3.5 SOP

3.5.1 Introducción

Como se ha visto en el la definición general de proceso, descrita al inicio del documento, existe un conjunto de “cajas” denominadas SOP, por lo tanto, en este apartado se describe cuál es el objetivo de dicha caja y cómo cumplirlo. El grupo de proyecto deberá seguir el mapa general y en el momento que se encuentren con un SOP, deberán recurrir a este apartado para su consulta.

Dado que los SOP son procesos que contienen pasos a detalle, estos tienden a contener datos confidenciales o que pueden variar mucho en función de la estructura de la organización que los aplica. Por este motivo, se ha sustituido el detalle por un resumen de su objetivo, ya que el detalle del mismo no es de gran aportación para el objetivo final de trabajo.

3.5.2 SOP 2.01 Analizar Petición inicial (E00) y actualizar índice de proyectos

En este SOP se analiza la petición del responsable de producto, revisando que contiene la mínima información necesaria para comenzar el proceso.

Tarea	Entrada	Implicados	Resultado
Analizar petición y	E00. Petición inicial	Responsable TyD	Actualizar índice de

actualizar índice de proyectos.			proyectos y comenzar proceso

3.5.3 SOP 2.05 Rellenar resumen económico

En este paso, el jefe de proyecto rellena un resumen económico de la información registrada en el proceso PG2.01 Análisis General Hito 1.

Tarea	Entrada	Implicados	Resultado
Rellenar Team Charter	E01 Análisis Riesgos E03 Backlog	Responsable proyecto	Resumen económico

3.5.4 SOP 2.06 Hito 1

En este paso, el responsable de producto, proyecto y el responsable de TyD valoran si el proyecto es viable en función de los números obtenidos.

Tarea	Entrada	Implicados	Resultado
Resolución hito 1	E01 Análisis Riesgos E02 Team Charter E03 Backlog	Responsable proyecto Responsable productos Responsable TyD	¿Continuamos?

3.5.5 SOP 2.08 Valorar escenario H2

Nº	OPERACIÓN	RESPONSABLE
1	Hay nuevos riesgos identificables en el proyecto? si: en el PG analisis General Hito 1 actualizar el documento de control de riesgos. Versionar el documento como hito 2. y seguir al paso siguiente No: seguir al paso siguiente..	Responsable del proyecto
2	Hay nuevos funcionalidades identificables en el proyecto? si: en el PG analisis General Hito 1 actualizar el documento de Backlog. Versionar el documento como hito 2.y seguir al paso siguiente. No: seguir al paso siguiente	Responsable del proyecto
3	Hay cambios en la estructura del equipo prevista en el proyecto? si: reconfirmar equipo de proyecto. No: seguir al paso siguiente	Responsable del proyecto
4	Hay cambios en los costes identificados en el proyecto? si: en el PG analisis General Hito 1 actualizar el documento de control de costes. Versionar el documento como hito 2. y seguir al paso siguiente No: seguir al paso siguiente.	Responsable del proyecto
5	Hay cambios en el plan idenntificado en el proyecto? si: actualizar la planificación del proyecto. Versionar el documento como hito 2. y seguir al paso siguiente No: seguir al paso siguiente.	Responsable del proyecto

3.5.6 SOP 2.09 Hito 2

En este paso, el responsable de producto, proyecto y el responsable de TyD valoran si el proyecto es viable en función de los números obtenidos.

Tarea	Entrada	Implicados	Resultado
Resolución hito 2	E01 Análisis Riesgos E02 Team Charter E03 Backlog	Responsable proyecto Responsable productos Responsable TyD	¿Continuamos?

3.5.7 SOP 2.11 Crear equipo trabajos

En este paso se confirma el grupo de trabajo, bien haciendo uso de recursos internos o bien recurriendo a la subcontratación mediante proveedores homologados.

Tarea	Entrada	Implicados	Resultado
Crear equipo trabajo	E01 Análisis Riesgos E02 Team Charter E03 Backlog	Responsable proyecto Responsable TyD Proveedores	Equipo trabajo confirmado

4 Proceso ágil de análisis funcional

4.1 Introducción

En la apartado dedicado al análisis, se explica cómo ayudar a los responsables de producto a definir cuáles son los objetivos que deben conseguirse tras el proceso de desarrollo. Además, se muestra cómo traducir estos objetivo en algo que permita a los desarrolladores llevarlos a cabo, de forma que coincida con las expectativas de los usuarios. Finalmente, se describe cómo validar que los desarrollos realizados cumplen correctamente con establecido en el proceso análisis.

4.2 Cuándo hacer el análisis

En muchas metodologías, al inicio del proyecto se reserva un tiempo considerable, variable en función del tipo de proyecto, en donde se intentan definir la totalidad de los requisitos que se realizarán en durante la posterior fase de desarrollo (Ministerio de Hacienda y Administraciones Públicas, 2001).

Este enfoque es arriesgado y difícil de conseguir. Arriesgado, porque al inicio es cuando menos se conoce de un proyecto, en consecuencia, si se toman en ese momento la totalidad de las decisiones existirá un alto riesgo de equivocación. Es difícil de conseguir, ya que es muy habitual que muchos requisitos aparezcan durante el desarrollo, y es normal, ya que en esta fase se crean pantallas, se realizan pruebas, etc, es decir, que para que estos requisitos aparecieran al inicio del proyecto, debería contarse con todos estos elementos, y esto es prácticamente imposible.

Por lo tanto, esto presenta una pregunta ¿cuándo se realiza el análisis?. La respuesta es sencilla, durante toda la vida del proyecto, es decir, se va haciendo gradualmente. Si se retoma el proceso de desarrollo descrito en el apartado 2, en el hito 1 hacemos un trabajo previo con el responsable de producto, estableciendo una hoja de ruta de los objetivos funcionales. Posteriormente, sprint a sprint, se detallan únicamente los objetivos que vayan a abordarse durante la siguiente iteración. En cada sprint podrá aparecer objetivos nuevos, o bien, modificar lo que establecimos en un inicio porque al final resulta que no era prioritario o incluso necesario, esto ayuda a rectificar sin coste los

errores que se pudieran cometer al definir la hoja de ruta (Mike Cohin, 2004)

En conclusión, dado que definir la que debe realizar una aplicación es una tarea compleja, debe hacerse mejor cuando más conocimiento se tiene. Es decir, se va definiendo lo que tiene mayor prioridad, procediendo a su desarrollo de forma casi inmediata en el siguiente sprint, realizando esto de forma iterativa.

4.3 Historias épicas e historias

En metodologías más tradiciones, como métrica 3, las necesidades se expresan mediante requisitos, casos de uso, etc. Estos son documentos que intentar recoger todo el detalle posible, dado que al realizarse un único proceso de análisis al inicio sin posibilidad de error, la mejor forma de abordarlo es mediante documentación detallada y un listado requisitos. El listado de requisitos exige además un laborioso proceso de trazabilidad, ya que como se define al inicio del proyecto, debe comprobarse que no se deje nada por alto al final del proyecto (Ministerio de Hacienda y Administraciones Públicas, 2001).

En las metodologías ágiles como Scrum, los análisis funcionales se establecen mediante otras técnicas que faciliten el análisis secuencial, es decir, que se ejecute durante toda la vida del proyecto. Principalmente las funcionalidad se transmite a través de historias épicas e historias.

Las historias épicas representan a funcionalidades de cierto tamaño, o mejor dicho, objetivos de cierta envergadura que debe cumplir la aplicación. Cuando al inicio del proceso, en el hito 1, el responsable de producto nos transmite la hoja de ruta para el presente año, realmente está trasladando un conjunto de historias épicas, todavía pendiente de definición. En conclusión, las historias épicas son objetivos que nuestra aplicación debe cumplir pero que todavía no se han analizado en detalle y por lo tanto todavía no se puede comenzar con su desarrollo. Para ilustrar mejor la explicación, se muestran algunos ejemplos:

- Almacenar datos de clientes
- Emitir una factura
- Emitir abono
- ...

Por esto, cuando comienza el proyecto en el sprint 0, normalmente el backlog se compone únicamente de historias épicas, dado que está todo pendiente de detallar. Es en este sprint cuando se escogen únicamente las épicas más prioritarias y se comienza con su análisis detallado. Este proceso comienza con la división de la historia épica en historias más pequeñas y manejables, es decir, el objetivo representado por la historia épica se divide en objetivos más pequeños

representados por las historias. En el backlog sustituiremos la historia épica por las nuevas historias generadas, estableciendo en cada historia, tal y como se indicaba en el apartado 2, la prioridad y la estimación.

El problema suele presentarse en cómo dividir, ya que no es una tarea fácil, el objetivo es partir nuestra épica en objetivos lo suficientemente pequeños que se puedan implementar en 3-4 días por un desarrollador. Para entender mejor el proceso, se ilustrará mediante un ejemplo haciendo uso de la historia épica "Emitir factura":

Ilustración 16: Backlog historia épica

Id	Historia épica	Historia	Importancia	Puntos de historia
100	Emitir factura		1000	21

1. La historia épica "Emitir factura" está así representada en el backlog dentro de la ilustración 16.

Ilustración 17: Backlog historia épica dividida

Id	Historia épica	Historia	Importancia	Puntos de historia
101	Emitir factura	formatear factura	1000	3
102	Emitir factura	Generar número correlativo de factura	990	3
103	Emitir factura	Imprimir datos empresa factura	980	1
104	Emitir factura	Imprimir datos cliente factura	970	3
105	Emitir factura – Generar detalle factura		960	13

2. A la hora de dividir, no es obligatorio que se dividan directamente en historias, es decir, si lo que se está tratando es muy grande podemos generar un conjunto de historias, que serán tratadas en el siguiente sprint, y de una nueva historia épica de menor tamaño que la anterior y que será tratada más adelante, puede verse una posible división en la ilustración 17.

3. Como puede observarse en el ejemplo anterior, se ha dividido la emisión de la factura en tres historias los suficientemente pequeñas para que sean desarrolladas en el siguiente sprint, además, se he generado una nueva épica que contiene el resto del objetivo todavía pendiente de dividir. La nueva épica será analizada más adelante, posiblemente durante el siguiente sprint.

Este proceso de división no sólo se realiza en el sprint 0, sino que se va realizando poco a poco en cada iteración. En el sprint anterior se analizan las historias que se tratarán en el sprint siguiente, evidentemente, no hay que esperar justo al sprint anterior y podemos abordar está división antes. Además, según se vaya avanzando en el desarrollo se detectarán nuevas historias, algunas de las inicialmente identificadas se desecharán y otras cambiarán de prioridad. Esto normal, y además es bueno, ya que según avanzamos en el proyecto el nivel de conocimiento del grupo aumenta y poco a poco se tienen más claro lo que se quiere, es decir, es contraproducente ceñirse estrictamente a un

guión que se define al inicio, cuando el conocimiento es menor. Es mejor trazar un plan de acción flexible (historias épicas identificadas en el sprint 0) que se adapte al avance del proyecto.

En las divisiones indicadas hasta ahora, estás sen centran principalmente en épicas de cierto tamaño que se dividen en objetivo más pequeños, abarcables estos dentro de un sprint. A veces, una historia puede transformarse en épica no sólo por su tamaño sino por su complejidad, por ejemplo, imaginemos la historia “pagas factura al banco”. El desarrollo de esta historia puede ser sencillo, ya que banco suele ofrecer las herramientas para facilitar su implementación, pero existe un desconocimiento total sobre este ámbito en el equipo de desarrollo, en consecuencia, se procederá a crear un historia destinada a la investigación que permita eliminar la incertidumbre y una, o más historias (en función del resultado del estudio), para su desarrollo (Mike Cohin, 2004).

Las historias generadas deberán cumplir las siguientes características:

- **Independiente**: Las historias (los objetivos) deben ser independientes los unos de los otros, ya que en en caso de no ser así, podrían generarse problemas con las prioridades. Por ejemplo, si tenemos dos historias una con una prioridad muy alta y una muy baja, pero la primera posee una dependencia con la segunda, no obligaría a empezar con la segunda a pesar de ser menos importante para el usuario. En caso de que se den estas situaciones se puede proceder a su unión, si ambas son pequeñas, o dividir la parte de la segunda que lo hace imprescindible y crear una nueva historia con prioridad más alta.
- **Negociable**: Las historias deben ser siempre negociables, como se ha comentado repetidas veces, en cada sprint el conocimiento del equipo aumentar y se toman mejores decisiones. Esto conlleva a que las historias se deben eliminar si se detecta que realmente el producto no las necesita; modificar en caso de que la aplicación requiera realmente otra cosa; cambiar de prioridad en caso de que detectemos que la aplicación necesite cumplir unos objetivos antes que otros; crear nuevas historias si se detectan nuevas necesidades durante el proceso de desarrollo.
- **Valor**: Las historias deben ser comprendidas perfectamente por todo el equipo, incluido el dueño del producto, ya que debe establecerse la prioridad de forma conjunta. Esto significa que las historias no pueden tener un carácter técnico, por ejemplo, “generar múltiples colas de mensajería para la recepción de información”, ya que sería ininteligible para un perfil no técnico y por lo tanto sería complejo establecer una prioridad. En cambio, “aumentar el rendimiento en la recepción de la información” sí puede ser entendida y priorizada por todo el equipo. Una vez la historia es entendida por la totalidad del equipo, está debe ser susceptible de ser valorada para analizar cómo de importante es para el proyecto y por lo tanto poderle asignar un valor dentro del backlog.
- **Estimable**: El equipo debe poder estimar la historia, es decir, ésta debe encontrarse lo suficientemente definida para que se permita su estimación (puntos de historia / tiempo)
- **Pequeña**: El objetivo asociada a una historia debe ser los suficientemente pequeño para ser

alcanzado por un desarrollador en un plazo de 3 – 4 días (Jorge Hernán Abad Londoño , 2013).

- **Testeable**: Las historias deben poderse testear, de forma que permita validar al equipo de que ésta se ha cumplido correctamente.

4.4 Cómo documentar

Una vez que se ha explicado como obtener y gestionar las historias se expondrá cómo proceder a su documentación. En pura teoría, la historias deben ser los suficientemente pequeñas para que su descripción funcional entre en una pequeña nota, en éstas además se podrán incluir pequeñas anotaciones derivadas de los procesos de conversación con los responsables de producto y los desarrolladores (Mike Cohin, 2004).

Esta técnica puede ser suficiente para el puro proceso de desarrollo, pero se antoja insuficiente cuando se intenta usar como guía de consulta sobre la cual examinar la funcionalidad concreta que contiene una versión específica del aplicativo. En cambio, regresar a los métodos de documentación tradicionales minimizan este problema, aunque no lo eliminan del todo, ya que cuando la aplicación evoluciona la documentación tienda a no actualizarse por el elevado coste que esto conlleva. Otro problema de aplicar los métodos más tradicionales, es que el tiempo invertido en generar toda la documentación provoca que en ocasiones el desarrollo sea más ágil que la propia escritura, y que a veces se tenga que desarrollar con un documento a medio completar, minimizando mucho el retorno de inversión asociado al proceso de escritura.

Ante esta situación, se propone evolucionar el puro método teórico de forma que permita dotar al proyecto de la agilidad necesaria, pero sin renunciar a un repositorio de consulta en el que pueda obtenerse de forma clara la funcionalidad existente para cada versión del aplicativo. En función del tipo documentación que necesite el proyecto se seleccionará un soporte documental diferente, existiendo un documento Word por historia épica en los proyectos con necesidades importantes de documentación, en cambio, los proyectos con una necesidad baja de documentación almacenarán todo su contenido en un One Note de Microsoft.

4.4.1 Proyecto con altas necesidades documentales

Con independencia de la documentación que requiera el proyecto, el backlog será llevado por medio de un Excel, éste sera responsable de contener la totalidad de historias e historias épicas existentes, tal y como puede verse en la ilustración 18.

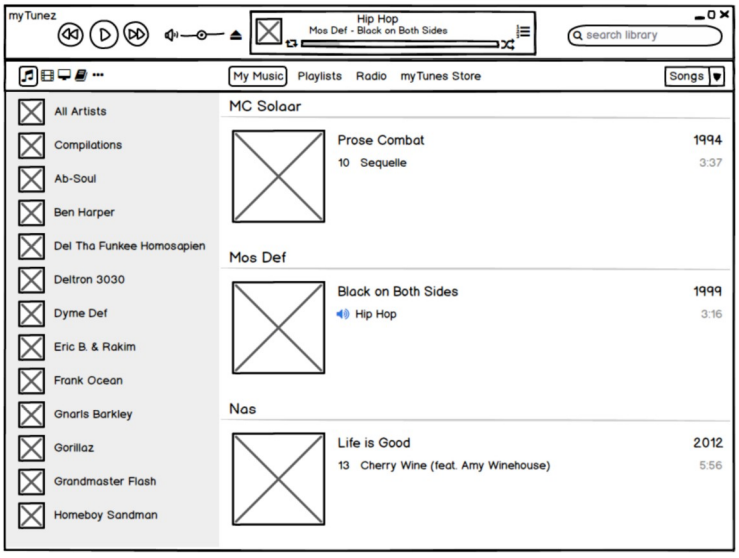
Ilustración 18: Listado historias para documentar

Id	Historia épica	Historia	Importancia	Puntos de historia
101	Emitir factura	formatear factura	1000	3
102	Emitir factura	Generar número correlativo de factura	990	3
103	Emitir factura	Imprimir datos empresa factura	980	1
104	Emitir factura	Imprimir datos cliente factura	970	3
105	Emitir factura – Generar detalle factura		960	13

El objetivo, es contar con un documento Word que describa a nivel general el objetivo que debe cumplir la historia épica, este documento no es necesario generarlo cuando se identifica la historia épica, sino que puede llevarse a cabo uno o dos sprint antes de su desarrollo. Este documento será desarrollado por los analistas y contrastado por la totalidad del equipo. El formato del mismo no está fijado y será pactado entre el grupo de análisis y el jefe del producto, aunque aquí se enumeran algunas recomendaciones:

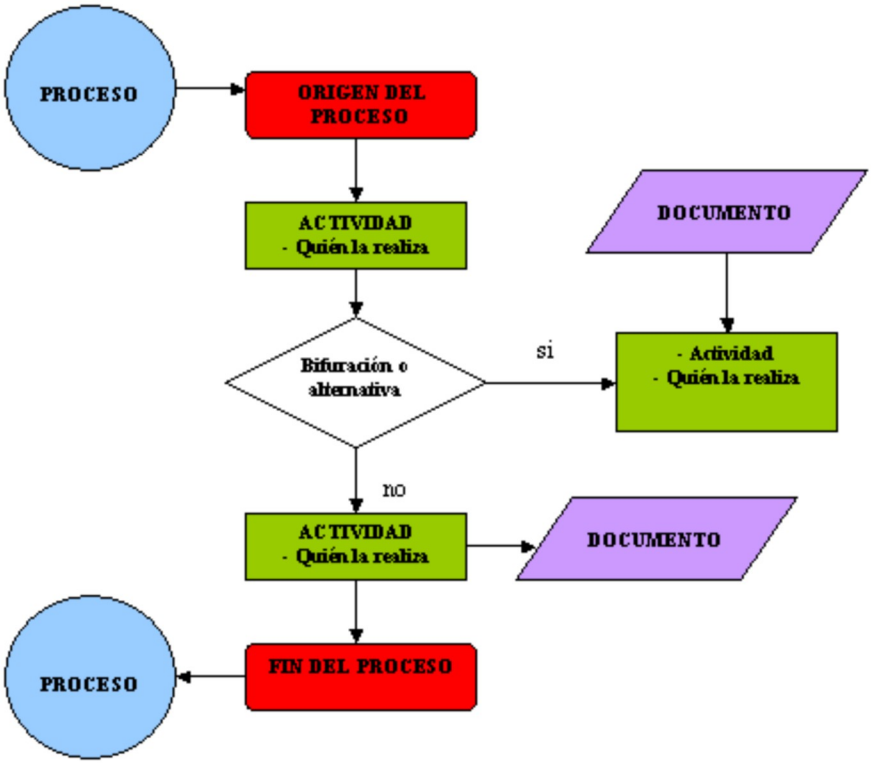
- Se recomienda encarecidamente apoyar el funcional en prototipos realizados mediante mockups. Una imagen de la futura pantalla ayudará mucho más al usuario, y serán más fáciles de mantener, que multitud de páginas descriptivas. Puede verse un ejemplo en la ilustración 19.

Ilustración 19: Mockup



- Se recomienda se breve y directo en la descripción del objetivo buscado en la historia épica, muchos más datos no significan necesariamente más precisión.
- En caso de que meta asociada a la historia épica sea la consecución de un proceso de negocio, se recomienda describir el proceso en pasos breves y claros, apoyando a éste en un diagrama representativo del proceso. El formato de dicho diagrama será libre, aunque se muestra un pequeño ejemplo en la ilustración 20.

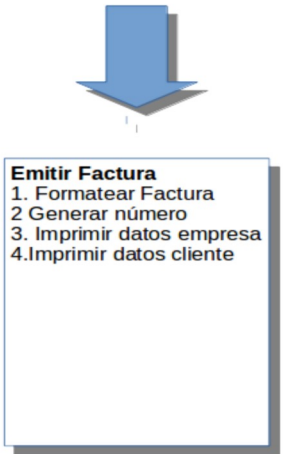
Ilustración 20: Diagrama proceso negocio



En el proceso de división de la historia épica, cada historia será un apartado dentro de la historia épica y vendrá descrito de forma breve cual es el objetivo concreto de dicha historia. Por lo que el documento queda estructurado de la siguiente forma, existirá un capítulo generar en donde se describe la historia épica, contando con un aparado por cada historia asociada correspondiente épica. Se muestra un ejemplo en la tabla 5.

Tabla 5: Historias épicas e historias

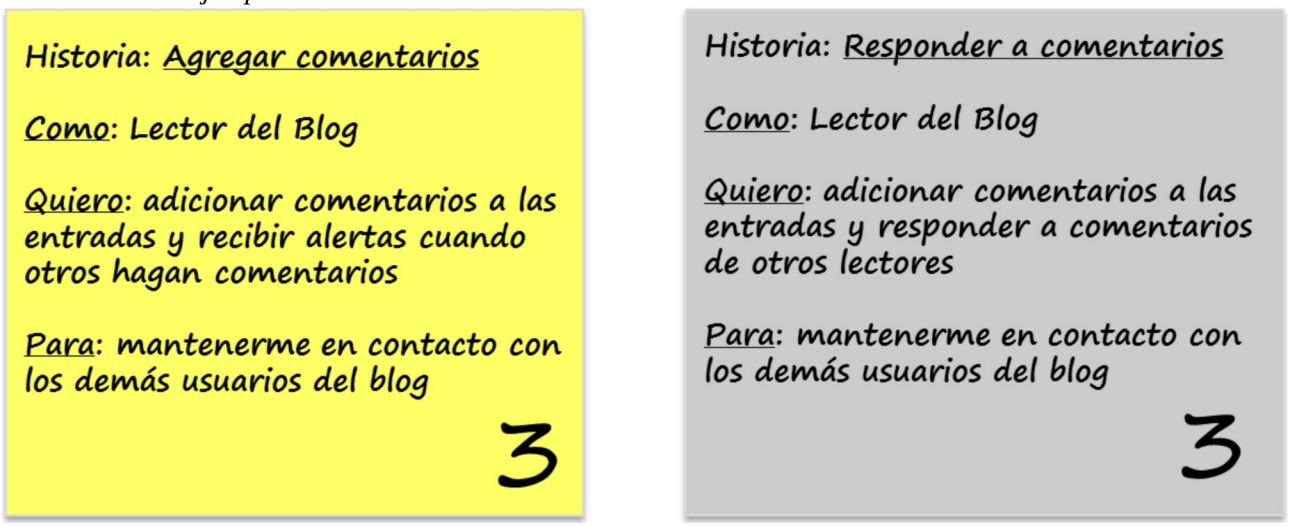
Id	Historia épica	Historia	Importancia
101	Emitir factura	formatear factura	1000
102	Emitir factura	Generar número correlativo de factura	990
103	Emitir factura	Imprimir datos empresa factura	980
104	Emitir factura	Imprimir datos cliente factura	970



Durante el proceso de análisis y desarrollo se irán detectando nuevas historias asociada a la épica, esto implicará que se vayan creando nuevos apartados dentro del correspondiente documento en donde relejar el detalle asociado a las nuevas historias. Si una historia modifica el comportamiento de la épica, deberá actualizarse la descripción general de ésta, el objetivo, es que el apartado asociada a la épica esté actualizado respecto a todas las historias contenidas en el documento. Es decir, es muy habitual que se incorporen, borren o modifiquen historia,si esto impacta en la descripción de la épica, deberá actualizar la descripción general, teniendo como objetivo que ésta describa la funcionalidad real.

En los diferente apartados que van asociados a las historias, se describirá de forma breve y concisa su objetivo específico, dentro del objetivo global marcado por la épica, que quedará cubierto tras su desarrollo. Es importante que en la descripción de la historia se indique a quién va dirigida, qué va ha hacer y con que objetivo, para ello se suelen usar tres preguntas cómo, qué y para qué, tal como se muestra en la ilustración 6.

Ilustración 21: Ejemplo historia



El desarrollador, tras la lectura de la épica y el objetivo marcado por la historia, deberán conocer exactamente cuál es su cometido. Posteriormente, durante conversaciones mantenidas entre el programador, el analista y el arquitecto,previas al desarrollo de la historia, se podrá aprovechar su correspondiente apartado en el documento para realizar las anotaciones técnicas y funcionales que sean de utilidad en su implementación.

Todas los documentos se agruparán en lo que se denominan temas, los temas son una forma de categorizar y agrupar las historias épicas. Es decir, se creará una carpeta por tema y dentro de cada una ser albergará un documento por historia épica existente, alojándose en el propio documento un apartado por cada historia asociada a la épica correspondiente.

Toda la documentación se agrupará en una única carpeta que indica la versión del aplicativo que se está desarrollando. Cuando se cumplan con los objetivos de la versión y se vaya a comenzar con

una nueva, se crearán una carpeta con la nueva versión, por ejemplo 2.3, y se copiará todo el contenido de la versión anterior, que es desde donde se parte. En esta carpeta recién creada, se actualizarán las historias épicas que vayan a ser modificadas, se crear nuevas épicas (nuevos documentos) y se elimina las épicas que vayan a desaparecer.

Desde el archivo Excel responsable de conformar el backlog, representado en la Ilustración 3, se permitirá acceder fácilmente a la documentación. Si pinchamos en la historia épica, se abrirá el documento correspondiente, en cambio, si pulsamos en la historia se abrirá la entrada asociada a la historia.

4.4.2 Proyecto con bajas necesidades documentales

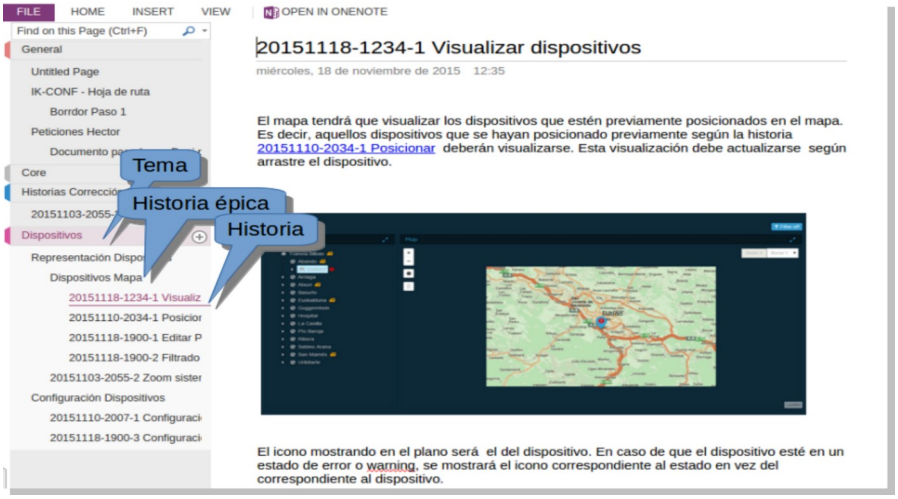
Al igual que en el caso anterior, el backlog será llevado por medio de un Excel responsable de contener la totalidad de historias e historias épicas existentes, tal y como puede verse en la Ilustración 22.

Ilustración 22: Listado historias para documentar

Id	Historia épica	Historia	Importancia	Puntos de historia
101	Emitir factura	formatear factura	1000	3
102	Emitir factura	Generar número correlativo de factura	990	3
103	Emitir factura	Imprimir datos empresa factura	980	1
104	Emitir factura	Imprimir datos cliente factura	970	3
105	Emitir factura – Generar detalle factura		960	13

En este caso, tal y como se muestra en la Ilustración 23, existirá un único documento OneNote de Microsoft, sobre el cual habrá una pestaña por cada tema (agrupación de historias épicas) existente. En cada pestaña, se albergará una entrada por cada historia épica donde se describirá a nivel general el objetivo que ésta debe cumplir, recordar que dicha descripción no es necesaria generarla cuando se identifica la historia épica, sino que puede llevarse a cabo uno o dos sprint antes de su desarrollo. Esta descripción será desarrollada por los analistas y contrastada por la totalidad del equipo. A diferencia del punto anterior, el nivel de documentación requerido será menor, de ahí la utilización de otro tipo de soporte.

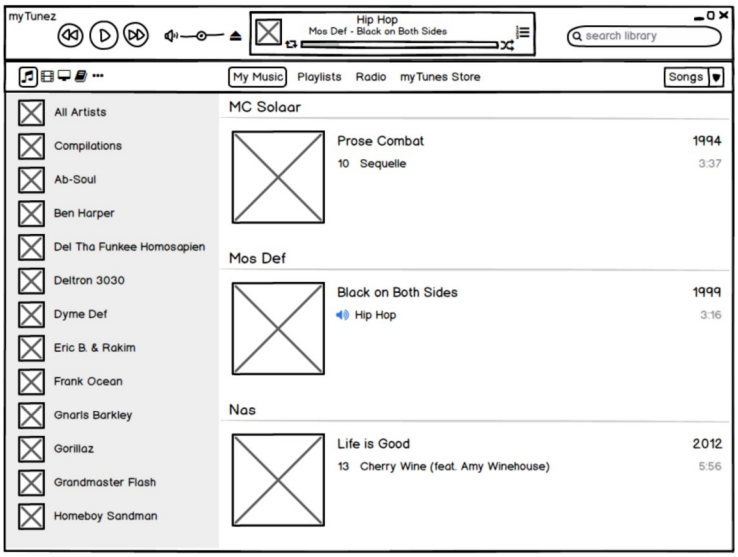
Ilustración 23: One Note



El formato del mismo no está fijado y será pactado entre el grupo de análisis y el jefe del producto, aunque aquí se enumeran algunas recomendaciones:

- Se recomienda encarecidamente apoyar el funcional en prototipos realizados mediante mockups. Una imagen de la futura pantalla ayudará mucho más al usuario, y serán más fáciles de mantener, que una amplia descripción. Puede verse un ejemplo en la Ilustración 24.

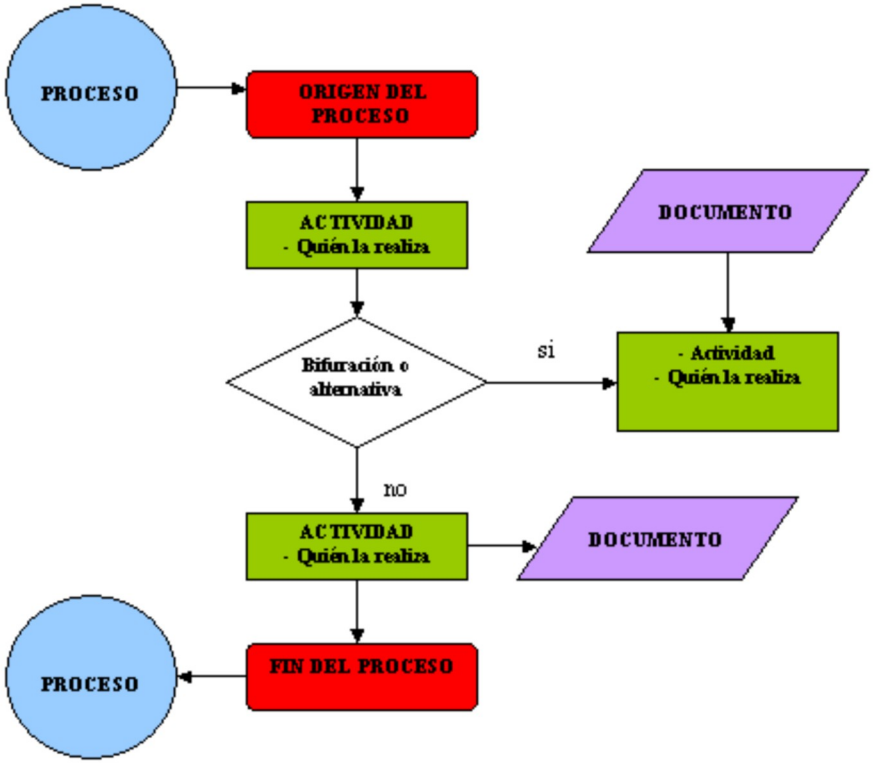
Ilustración 24: Mockup



- Se recomienda se breve y directo en la descripción del objetivo buscado en la historia épica, muchos más datos no significan necesariamente más precisión.
- En caso de que meta asociada a la historia épica sea la consecución de un proceso de

negocio, se recomienda describir el proceso en pasos breves y claros, apoyando a éste en un diagrama representativo del proceso. El formato de dicho diagrama será libre, aunque se muestra un pequeño ejemplo en la ilustración 25.

Ilustración 25: Diagrama proceso negocio

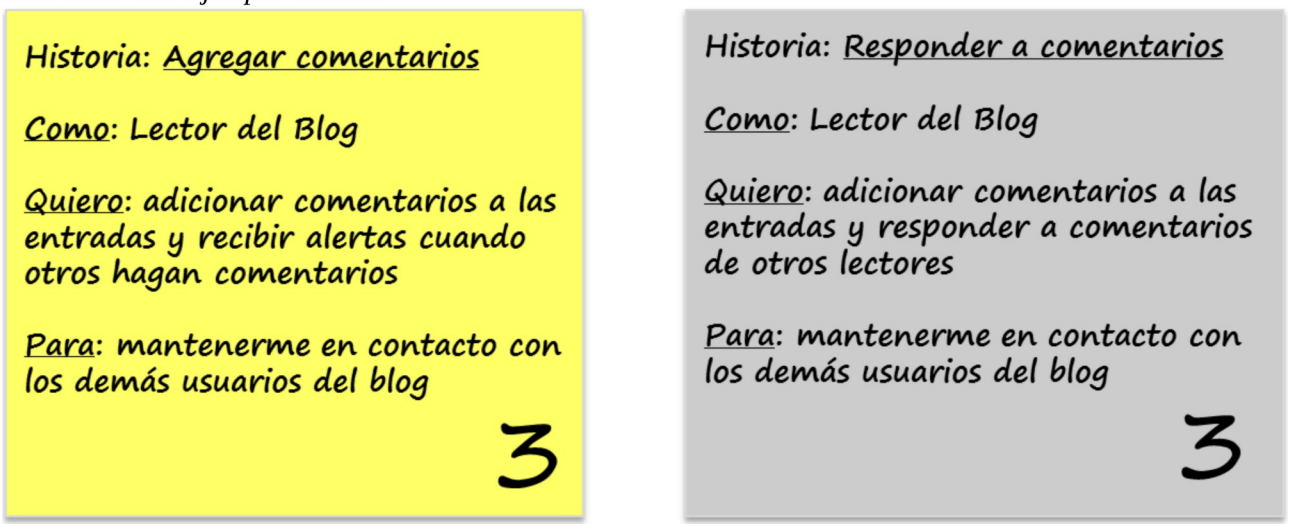


En el proceso de división de la historia épica, cada historia será una entrada dentro de la épica, tal y como se muestra en la Ilustración 23, y vendrá descrito de forma breve cual es el objetivo concreto de dicha historia.

Durante el proceso de análisis y desarrollo se irán detectando nuevas historias asociada a la épica, esto implicará que se vayan creando nuevas entradas donde relejar el detalle asociado a las nuevas historias. Si una historia modifica el comportamiento de la épica, deberá actualizarse la descripción general de ésta, el objetivo, es que el apartado asociada a la épica esté actualizado respecto a todas las historias contenidas en el OneNote. Es decir, es muy habitual que se incorporen, borren o modifiquen historias,si esto impacta en la descripción de la épica, deberá actualizar la descripción general, teniendo como objetivo que ésta describa la funcionalidad real.

En los diferente apartados que van asociados a las historias, se describirá de forma breve y concisa su objetivo específico, dentro del objetivo global marcado por la épica, que quedará cubierto tras su desarrollo. Es importante que en la descripción de la historia se indique a quién va dirigida, qué va ha hacer y con que objetivo, para ello se suelen usar tres preguntas cómo, qué y para qué, tal como se muestra en la ilustración 26.

Ilustración 26: Ejemplo historia



El desarrollador, tras la lectura de la épica y el objetivo marcado por la historia, deberán conocer exactamente cuál es su cometido. Posteriormente, durante conversaciones mantenidas entre el programador, el analista y el arquitecto,previas al desarrollo de la historia, se podrá aprovechar su correspondiente apartado para realizar las anotaciones técnicas y funcionales que sean de utilidad en su implementación.

Toda la documentación se agrupará en una único fichero, indicando en su nombre la versión del que se está desarrollando, por ejemplo, Aplicación1_0. Cuando se cumplan con los objetivos de la versión y se vaya a comenzar con una nueva, se copiará el fichero y se renombrará, Aplicación1_1. Sobre el nuevo documento, se actualizarán las historias épicas que vayan a ser modificadas, se crear nuevas épicas (nuevos entradas) y se elimina las épicas que vayan a desaparecer.

Desde el archivo Excel responsable de conformar el backlog, representado en la Ilustración 22, se permitirá acceder fácilmente a la documentación. Si pinchamos en la historia épica, se abrirá el apartado del OneNote correspondiente, en cambio, si pulsamos en la historia se abrirá la entrada asociada a la historia.

4.5 Cómo obtener la información

En este apartado se procederá a describir ciertas técnicas que tienen como objetivo ayudar a analistas y responsables de producto a identificar, priorizar y documentar historias.

- El equipo de análisis debe ayudar al responsable de producto en la identificación de historias. El responsable de producto, aunque conozca cuáles son los objetivos a conseguir no sabe expresarlos , necesita ayuda por parte del equipo para generar las correspondientes historias y priorizarlas. En la mayoría de las veces el usuario no sabe exactamente lo que quiere, pero cuando le sugerimos ejemplos sin ambigüedad ni definiciones, generalmente

sabe decir si es o no eso lo que busca (Carlos Blé Jurado, 2010)

- La forma más efectiva de generar historias viene dada directamente a través de la conversación. Además de toda la documentación e información que pueda facilitar el responsable de producto, estará incompleta si no viene refrendada por una conversación que la confirme y la complete.
- La observación es un punto muy importante en la obtención de historias. Si existe la oportunidad de observar a lo usuarios finales en los procesos que debe cubrir la aplicación, podrá entenderse la visión del usuario, identificar historias que para ellos puedan resultar obvias, dar un soporte mucho más efectivo en la asignación de prioridades y describir las historias de una forma más efectiva.
- La generación de prototipos, conformado mediante mockups, es una herramienta muy efectiva que ayuda mucho a los usuarios a entender y detallar la historias. El tener un apoyo visual les ayuda a comunicarse con el resto del equipo y a identificar historias que podrían haberse obviado.

5 Auditoría funcional

5.1 Introducción

La auditoría funcional, las denominadas pruebas, es la manera de verificar que la aplicación hace lo que se espera que haga. Este es un proceso tan importante como el de desarrollo, ya que una aplicación sin una auditoría funcional adecuada provoca que los errores se detecten en el entorno productivo, esto genera importantes costes, mala imagen delante del cliente, etc. En este apartado se describe cómo generar las pruebas que validarán los desarrollos asociados de la diferentes historias.

5.2 Pruebas de verificación por historia

Durante el proceso de análisis, además de identificar y detallar las historias, deben generarse los test que permitan validar su comportamiento. Es decir, la generación de las pruebas de verificación es parte del análisis y debe realizarse al inicio, en la definición de la historia.

A la mayoría del equipo esto le resulta complicado, cómo puede generarse las pruebas al inicio de todo. Esto, aunque suene raro, dentro de Scrum es habitual, ya que la definición de las pruebas no es más que definir mejor la historia, como consecuencia, el desarrollador trabaja en un escenario más favorable porque además de la propia definición de la historia cuenta con una serie de test de verificación que va a confirmar que ha entendido debidamente el funcional y que su desarrollo

funcionar correctamente (Mike cohin, 2004)

Por cada historia generaremos una serie de test que confirmarán su funcionamiento. Esto son afirmaciones en un lenguaje humano que tanto el dueño del producto, como los desarrolladores, entiende. Dichas afirmaciones deben centrarse en el qué y no en el cómo y deben definir el funcionamiento exacto de la historia. Puede verse un ejemplo en la ilustración 27.

Ilustración 27: Ejemplo test

- *Buscando que el precio sea inferior a 600€, e introduciendo el texto "Santa Cruz de Tenerife", el sistema muestra una lista de pisos que no superan los 600€mensuales de alquiler y que se encuentran en la ciudad de Santa Cruz de Tenerife*
- *Buscando que el precio esté entre 500€y 700€y que tenga 2 habitaciones e introduciendo el texto "Santa Cruz de Tenerife", el sistema muestra una lista de pisos que cumplen las tres condiciones*
- *Buscando que tenga 3 habitaciones y 2 cuartos de baño, e introduciendo el texto "Santa Cruz de Tenerife", el sistema muestra una lista de pisos que cumplen las tres condiciones*
- *Buscando con el texto "Tenerife", el sistema muestra la lista de pisos de toda la provincia de Santa Cruz de Tenerife*
- *En la lista, cada piso se muestra mediante una fotografía y el número de habitaciones que tiene*

5.3 Pruebas de verificación por historia épica

Como se ha expuesto en el punto anterior, cada historia dispondrá de un pruebas de aceptación que validarán que su implementación se ha desarrollado correctamente. En ocasiones, los bugs no se encuentra exclusivamente en la funcionalidad establecida por una historia, sino que se encuentran en la combinación de la casuística de una historia con otra.

Para solventar esto se definirán una serie de test de integración, de forma que pueda probarse la aplicación de forma horizontal. Es decir, con los test asociados a cada historia probamos una parte concreta del aplicativo, en cambio, con los test de integración validamos de forma horizontal varios

módulos verificando que estos trabajan de forma coordinada (Carlos Ble Jurado, 2010).

Los test de integración deben estar escritos y verificados antes de las pruebas FAT (descrita en 2.4.7.5 Liberar versión - Pruebas FAT) asociada a la liberación de una versión. En estos, se analizarán la posibles casuísticas que puedan darse entre un módulo estableciendo las pruebas que verifique su correcto comportamiento.

5.4 Cómo organizar las pruebas

Las pruebas se organizarán por medio de la herramienta Testlink, ésta servirá como repositorio para aunar los test de las diferentes versiones de nuestro aplicativo (Qaustral ,2010). Para ello se seguirá la siguiente estructura:

Por cada producto se creará un proyecto de Testlink, tal y como puede verse en las Ilustraciones 28 y 29.

Ilustración 28: Creación proyecto Testlink

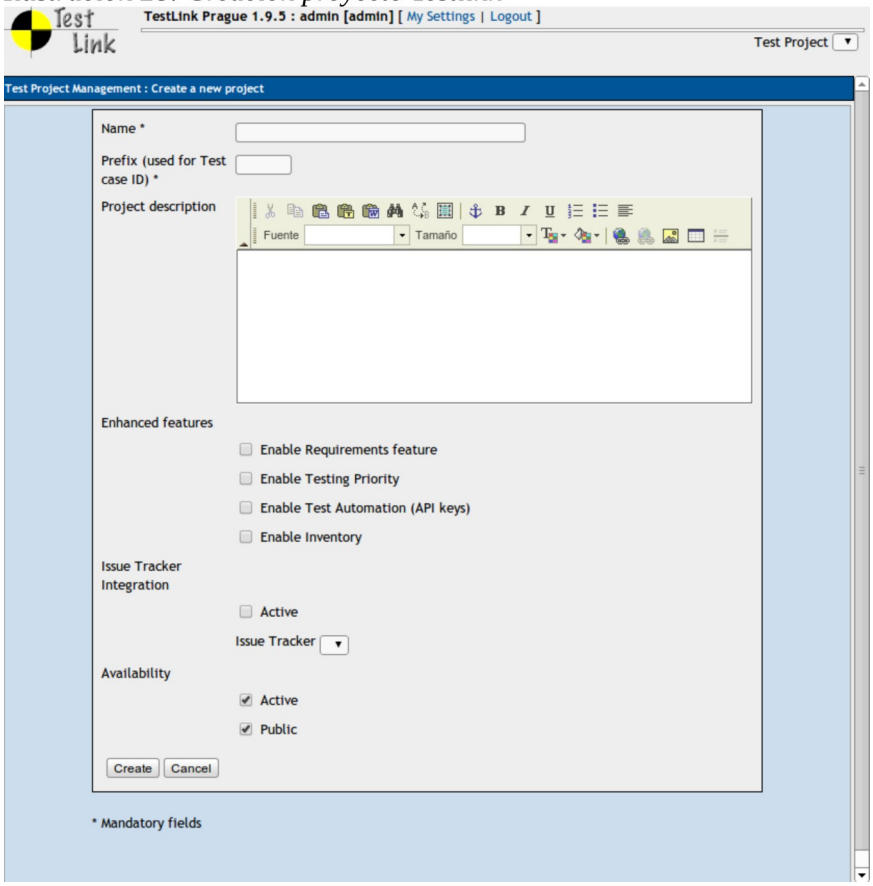
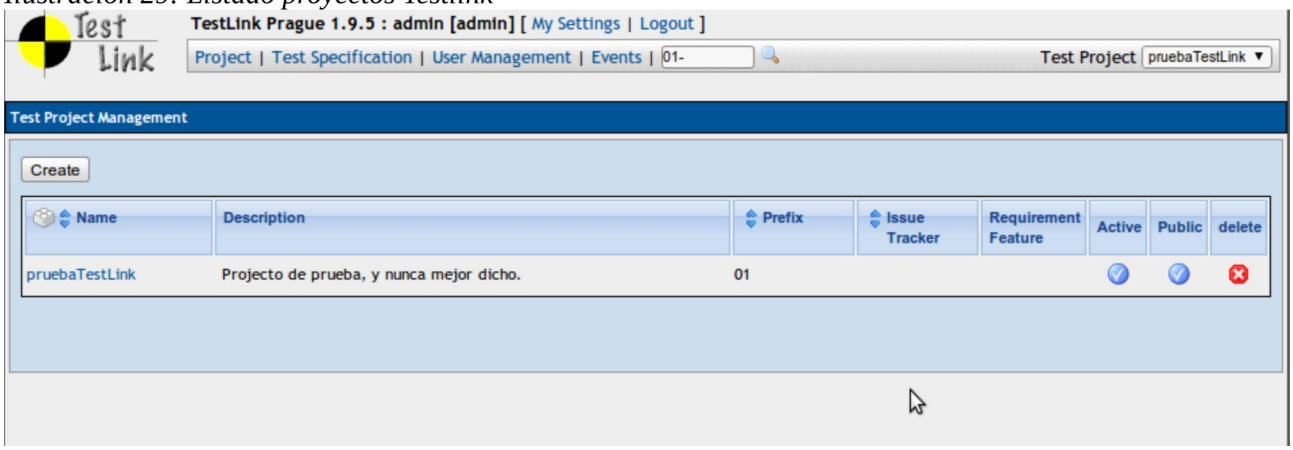


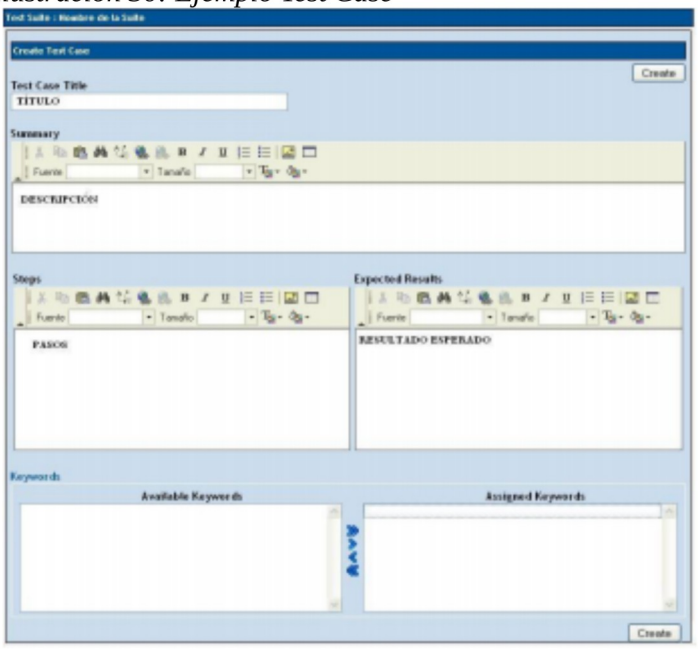
Ilustración 29: Listado proyectos Testlink



Una vez creado el proyecto, se creara un Test Suite, para TestLink es un conjunto de test, por cada tema existentes en el backlog. Dentro de cada Test Suite asociado a cada tema, se creará una nueva agrupación de test para cada historia épica, incluyendo en éste uno nuevo para cada historia. Una vez realizadas estas acciones, se contará con la estructura de test correcta para albergar las diferentes pruebas asociadas a cada historia e historia épica.

Los test se crear por medio de los Test Case, ubicando estos en el Test Suite correspondiente, es decir, en la historia épica o historia asociada. Por cada test se deberá aportar la información mostrada en la Ilustración 15.

Ilustración 30: Ejemplo Test Case



- Título: Título descriptivo de aquello que se desea probar.
- Resumen: Resumen descriptivo de lo que verifica el test.
- Condiciones previas: Condiciones que deben cumplirse para poder realizar la prueba
- Pasos y resultado: Se podrá establecer una serie de pasos a realizar durante la ejecución de la prueba, indicando para cada uno de ellos el resultado esperado.
- Adjuntos: Se podrán adjuntar archivos que ayuden durante la ejecución de la pruebas, como ejemplos de pantallas, scripts, etc.

Por cada una de las pruebas escritas, éstas se podrán asociar a una versión del aplicativo, de esta forma, dispondremos de las pruebas que deben realizarse para una versión u otra. De forma similar, el sistema también permita asociar pruebas a un entorno de test, por ejemplo, no pueden realizarse las mismas pruebas en un laboratorio que en equipos reales ubicados en campo. El sistema permita gestionar el versionado de las pruebas, así si una misma prueba cambiar de una versión de un aplicativo a otra, podremos gestionar su trazabilidad. El sistema permite monitorizar la ejecución de las diferentes pruebas, permitiendo de esta forma generar el correspondiente reporte de resultado (Test correctos, incorrectos, tiempos requeridos para su ejecución…)

5.5 Entornos de pruebas

Todo producto contará con los siguientes entornos de pruebas

Dibujo 4: Entornos

- Desarrollo: Este es el entorno de trabajo utilizado por los desarrolladores para la implementación de las historias de usuario y la realización de sus pruebas de desarrollo. Es propiedad del desarrollador por lo que tiene pleno control sobre éste.

- Integración: Este entorno en donde el equipo de pruebas realiza las pruebas de aceptación por historia e historia épica. Aquí además se realizan las pruebas de instalación, con su correspondiente documentación, por esto, este entorno es propiedad del equipo (esto requiere coordinación por pate del mismo).
- Preproducción: Aquí se realizan las pruebas finales, lo que en el apartado 2 se denominaba pruebas FAT. Es decir, aquí se debería probar la aplicación una vez esté ya totalmente testada y libre de errores. Dado que este entorno debe ser idéntico a producción, aquí se podrán reproducir y corregir errores reportados por los usuarios finales en producción. Sobre este ambiente, se requiere de un control más estricto, por esta razón, será gestionado por el personal de sistemas, además, podrá ser ubicado en Amazon, de esta forma, posibilitará simular multitud de entornos productivos sin un gran coste asociado, sólo de uso.

5.6 Automatización de pruebas

Las pruebas, además de ser un proceso importante, también son un proceso muy costoso, ya que se tienen que repetir y actualizar en función de la evolución del aplicativo. Por lo que cada vez que se genera un nueva versión debe revisarse si las pruebas definidas anteriormente siguen siendo válidas para posteriormente pasarlas de nuevo.

Este proceso pueden verse mitigado de forma importante mediante la automatización de pruebas. Por un lado los desarrolladores deberán realizar test unitarios, de cada uno de los métodos albergados en la fachada del modelo del aplicativo, tal como se expone en el apartado 5.3 Modelo. Además, sobre las pruebas manuales asociadas a historias e historias épicas generadas en los procesos de análisis deberán revisarse cuales de ellas son susceptibles de ser automatizadas para proceder a su implementación.

La ventajas de las pruebas automáticas es que pueden ejecutarse todos los días si requerir por ello una dedicación expresa por parte del equipo, por lo que diariamente se generara un reporte con el resultado de las diferentes pruebas automáticas. Gracias a este informe, el equipo de desarrollo puede identificar sin existente errores o bien si se debe actualizar un test porque esté ya no es valido tras los nuevos desarrollos. Para posibilitar este escenario se contarán con herramientas de integración continua como las siguientes:

- Nexus: Repositorio de componentes (librerías) con código que debe probarse o bien que la aplicación a probar requiere para su funcionamiento.
- SVN: Repositorio en donde se ubica la totalidad del código fuente a testar.
- Jenkins: Será el encargado de gestionar las pruebas de integración continua.
- Testlink: Encargado de albergar los test a realizar y el resultado de los mismos.
- Selenium: Realización de pruebas en entornos Web.

Este herramientas trabajarán de forma coordinada para que diariamente a partir del código fuente se generen versiones ejecutables que sean testadas de forma automática, generado los correspondientes informes.

6 Arquitectura Tecnológica

6.1 Introducción

En el apartado anterior se ha expuesto las herramientas que permitirán el equipo determinar qué deben de hacer, este capitulo en cambio se centra en el cómo. Es decir, aquí se definirá la plataforma tecnológica que dará respuesta a los requisitos funcionales a abordar y cómo hacer uso de ésta.

Las aplicaciones se implementaran mediante un desarrollo basado en capas, haciendo uso del patrón modelo / vista / controlador, tal y como se muestra en la Ilustración 31. El modelo corresponde a la capa en donde se ubica la inteligencia de nuestra aplicación, está se expone por varios canales a través de un controlador, mediante una capa Web a los usuarios finales o por medio de Web Services en caso de que el consumidor de la lógica de negocio sea otra aplicación. La capa de persistencia será la responsable de almacenar la información gestionada por el aplicativo, además, la aplicación podrá obtener información de otros sistema por medio de colas de mensajería u otros canales como Web Services, etc.

Ilustración 31 MVC

Como podrá verse en los siguientes puntos, la capa tecnológica se basa principalmente en tecnología Java. En cada área se expondrá los frameworks Java utilizados para facilitar el proceso de desarrollo.

6.2 Persistencia

La capa de persistencia es la responsable de almacenar los datos gestionados por el aplicativo. Esta viene conformada por la base de datos, pudiendo elegir entre las siguientes plataformas:

- SQL Server: En caso de que nuestro desarrollo requiera de una plataforma de base de datos con servicio de soporte, esta es la alternativa más económica. En función de los requerimiento de almacenamiento y de alta disponibilidad deberemos seleccionar la versión adecuada (Standar, Enterprise,..)
- Oracle: Base de datos, con soporte asociado, altamente extendida especialmente en entornos críticos o con un alto volumen de datos. En función de los requerimientos de almacenamiento y de alta disponibilidad deberemos seleccionar la versión adecuada (Standar, Enterprise)

- Postgre SQL: Base de datos de código libre que además se adapta correctamente a entorno de críticos o con un alto volumen de datos. El principal problema de esta opción es que el soporte está dado por la comunidad, por lo que en algunos entornos esto puede ser un impedimento.

La elección de la base de datos es una decisión muy importante, ya que tiene implicaciones en muchos ámbitos. La elección de una base de datos propietaria implica que asociado a nuestra aplicación tendremos unos costes importantes que se repetirá anualmente, sobre todo en caso de elegir Oracle. En cambio, la elección de una base de datos soportada por la comunidad puede traer problemas en ciertas instalaciones, en donde niveles de soporte urgente son un requisito indispensable.

Otro punto a tener en cuenta, es el uso que se hace de la base de datos, sobre todo en aquellas que poseen coste de licenciamiento. Conceptos como el volumen de datos, usuarios nominales requeridos , ala disponibilidad, etc, pueden impactar de forma considerable en el coste de licenciamiento.

Dado la tecnología utilizada en el desarrollo de aplicaciones, la generación de aplicaciones que soporten múltiples base de datos es algo totalmente viable sin impactar de forma considerable en la estimación final. Es decir, la aplicaciones deberían soportar de serie las plataforma de base de datos anteriormente

6.3 Modelo

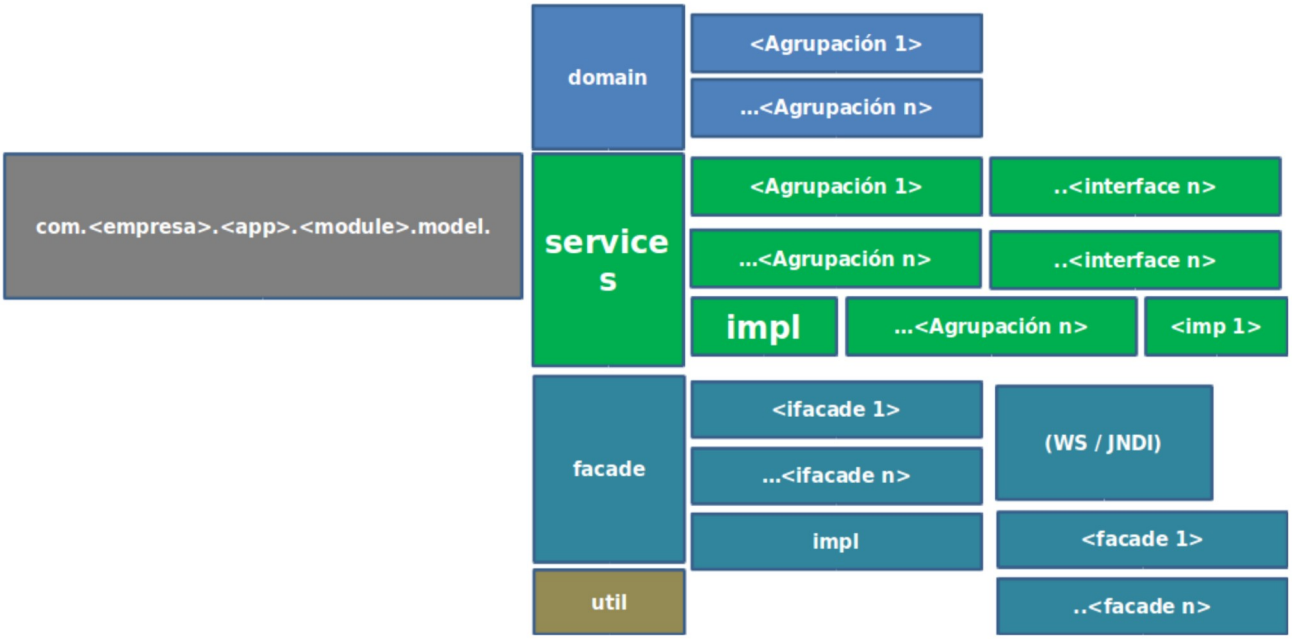
6.3.1 Tecnología

En este capa se desarrolla la inteligencia de nuestra aplicación, basando su desarrollo en tecnología Java. En este capa de la aplicación se hará uso principalmente de los siguientes Frameworks:

- Spring Core: Esta capa hará uso de este framework para su implementación, obteniendo de la misma características tan interesantes como ya inyección de independencia y la gestión de aspectos.
- JPA 2: La integración con la capa de persistencia se hará haciendo uso de esta tecnología, facilitando el uso de múltiples base de datos por parte de nuestra aplicación. Su hará uso de la implementación ofrecida por Hibernate la utilización de esta tecnología.
- Spring Data: Spring Data es un proyecto de SpringSource cuyo propósito es unificar y facilitar el acceso a distintos tipos de tecnologías de persitencia, tanto a bases de datos relacionales como a las del tipo NoSQL (Jose Manuel Sánchez Suárez, 2014).

6.3.2 Patrón de diseño

Ilustración 32: Estructura modelo



Los desarrollos asociado a esta capa deberán seguir el patrón de diseño marcado en la Ilustración 32:

- **Domain**: Clases de dominio responsables de contener la información gestionada por el sistema, es decir, a través de estas clases la aplicaciones almacena y gestionar la información que requieren para ofrecer las correspondiente funcionalidad. Éstas se agruparán en paquetes en función de la naturaleza de la clase. Dichas clases serán la unión con la capa de persistencia correspondiente al aplicativo.
- **Services**: Esta clases contendrán las funcionalidades del sistema dividas en funcionalidades unitarias, que posteriormente serán coordinadas por la fachada. Los servicios quedarán divididos por áreas en caso de que se prevea un crecimiento considerable de éstos. Todo servicio deberá tener como mínimo una interface, que será la utilizada por el resto del sistema con el objeto de reducir el acoplamiento. Estos servicios serán implementados mediante spring framework.
- **Fachada**: Las fachadas serán las responsables de coordinar y exponer las funcionalidades asociados a los servicios**.** No deberá existir un número elevado de fachadas, por lo que no será necesario su agrupación. Éstas deberán poseer su correspondiente interface, siendo ésta última quien sea usada por el resto de capas. Es decir, en ningún caso deberá hacerse uso directo de la implementación sino siempre de su correspondiente interface.
- **Util**: Paquete responsable de contener constantes y utilidades utilizadas por el aplicativo.

6.3.3 Integración con sistemas externos

En caso de que nuestra aplicación haga uso de otros sistemas para complementar su lógica de negocio, se integrará con estas hacienda uso de las siguientes tecnologías, tal y como puede observarse en la Ilustración 31.

- Colas de mensajería: Un sistema externo puede comunicarse con nuestra aplicación por medio de colas de mensajería en caso que se requiera la realización de operaciones asíncronas. Las tecnología asociada a las colas de mensajería usará las siguientes reglas:
  - AMQP: El sistema con el que nos queremos integrar usa AMQP o se puede seleccionar la tecnología a utilizar en ambos sistemas
  - JMS: El sistema con el que nos debemos integrar usa este estandar de Java.
  - MQTT: Sistema con recursos limitados o redes con recursos limitados, o bien, el canal de comunicación es WebSocket

Ilustración 33: Comparación colas de mensajería

Carácterística	Estándar		
	AMQP	JMS	MQTT
Interoperatividad	★★★		★★
Priorización de mensajes	★★★	★★★	
Uso avanzado de patrones de mensajería	★★★	★★	
IOT	★★★		★★★
Ancho de banda limitado / redes de gran latencia	★★		★★★
Dispositivos con recursos limitados / sistemas embebidos	★★		★★★
Escalabilidad	★★★	★★★	★★★
Rendimiento	★★★	★★	★★★
Concurrencia	★★	★★	★★★
Seguridad	★★★	★★★	★★★★
Transaccionalidad	★★★	★★★	★★★
Gestión avanzada de colas: restricciones, enrutamiento	★★★	★	

- Web Services: En el caso de que se requiera la realización de operaciones síncronas y el sistema con el que hay que integrarse esta diseñado para entornos heterogéneos, este es un buen canal de comunicación. En caso de poder elegir, se hará uso de JSON ya que optimiza la comunicaciones de este tipo.
- Spring HTTP Invoker: En el caso de que se requiera la realización de operaciones síncronas y el módulo con el que deba integrarse nuestra aplicación no requiere su utilización en entornos heterogéneos, usaremos esta tecnología, ya que permite su integración de forma fácil y transparente

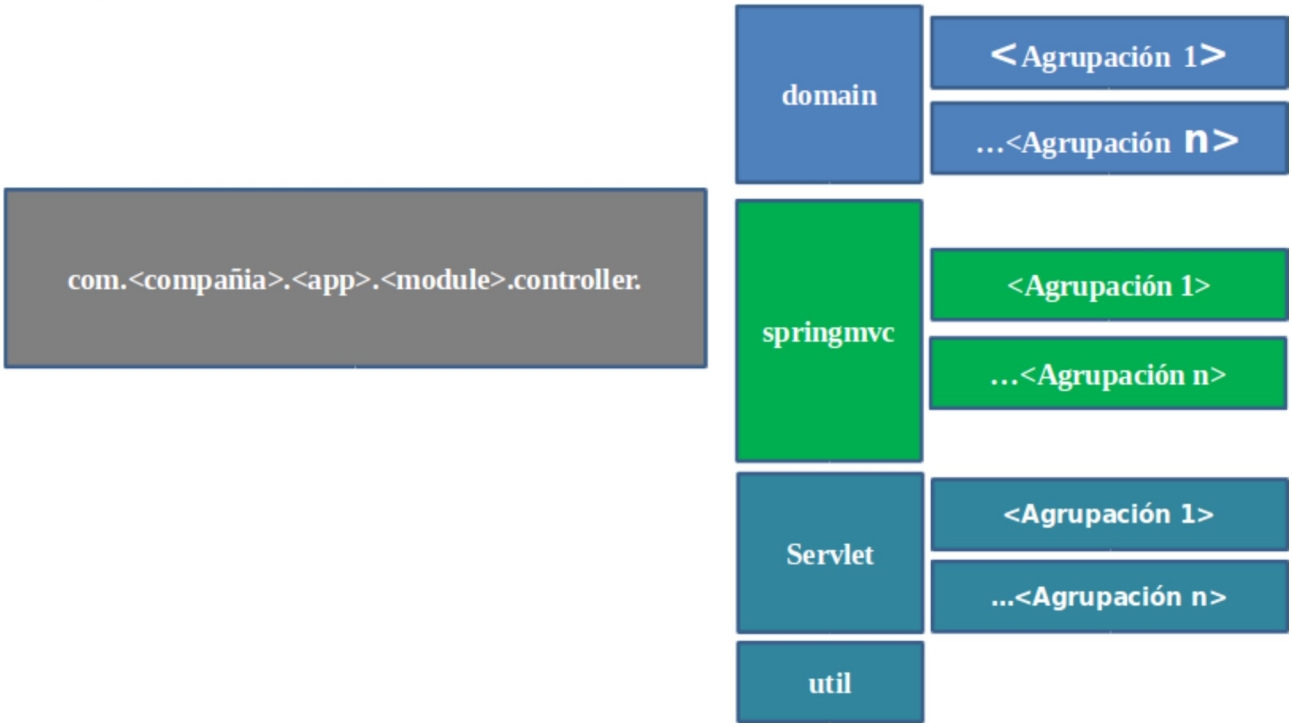
6.4 Controlador

6.4.1 Tecnología

Esta capa es la unión entre la lógica de negocio y la interface de usuario. Al igual que el modelo, se hará uso de la tecnología Java y más en concreto, se utilizará el framework Spring MVC.

6.4.2 Patrón de diseño

Ilustración 34: Estructura Controlador



Los desarrollos asociado a esta capa deberán seguir el patrón de diseño marcado en la Ilustración 34:

- **Domain**: Esta clases serán las responsable de adaptar el dominio proveniente del modelo a las necesidades de la capa visual. Éstas también quedarán agrupadas por áreas en caso de que existan o se prevea un número considerable de clases.
- **Controlador SpringMVC**: Los controladores Spring MVC será los responsables de atender a las peticiones de la capa visual, existirá aun agrupación por área (mantenimiento)
- **Servlet**: En el caso de que nuestra aplicación requiera el uso de Servlets podrá hacerse uso de estos, de cara a la organización, quedarán agrupados por áreas en el caso de que se prevea la existencia de un número considerable de clases.

6.5 Vista

6.5.1 Tecnología

Esta capa que tendrán interacción directa con el usuario, por lógica, está capa será implementada al igual que el resto, mediante tecnología Web. Las aplicaciones a desarrollar serán Web y se implementarán por medio de JSP aunque la interface con el usuario será ”**single-page application** (**SPA**)”

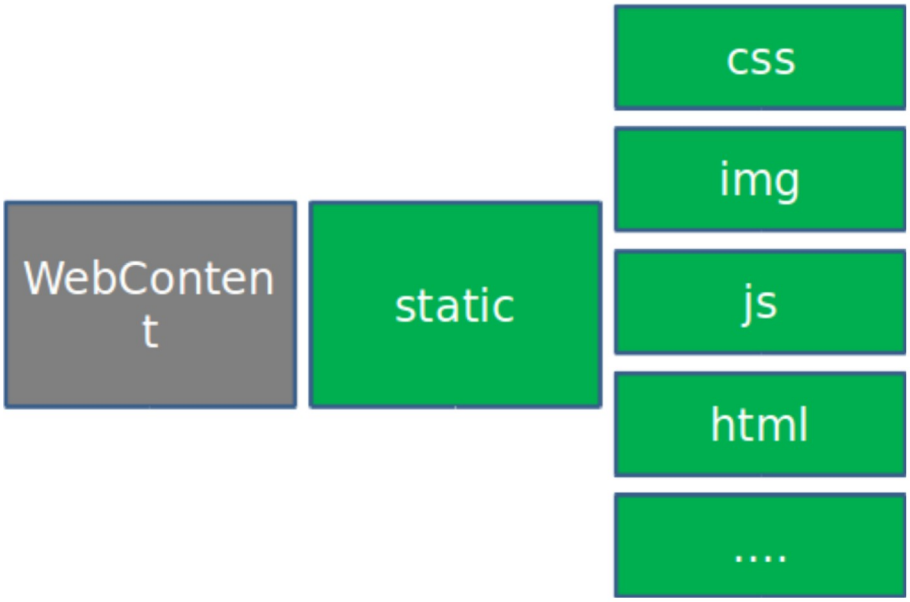
La aplicación de página única (SPA) es una aplicación web o es un sitio web que cabe en una sola página con el propósito de dar una experiencia más fluida a los usuarios como una aplicación de escritorio. En un SPA todos los códigos de HTML, JavaScript, y CSS se carga en de una vez o los recursos necesarios se cargan dinámicamente como lo requiera la página y se van agregando, normalmente como respuesta de los acciones del usuario. La página no tiene que cargar otra vez en ningún punto del proceso tampoco se transfiere a otra página, aunque las tecnologías modernas (como el pushState API del HTML5) pueden permitir la navegabilidad en páginas lógicas dentro de la aplicación. La interacción con las aplicaciones de página única pueden involucrar comunicaciones dinámicas con el servidor web que está detrás (Wikipedia, 2015).

Los controladores Spring MVC, además de generar las SPA a petición del usuario al conectarse a una URL, también serán los responsable de responder a las llamadas AJAX realizadas por el HTML albergado en el navegador del usuario. Este controladores retornarán el JSON correspondiente a la página permitiendo realizar la funcionalidad adecuada, otorgando la fluidez de una aplicación de escritorio. Dado que ésta es un parte compleja, existirá una aplicación ejemplo que muestre la tecnología Javascript utilizada y cómo debe aplicarse.

Las diferentes aplicaciones tendrán un estilo visual uniforme, para conseguir este objetivo, se contará con un componente reutilizable que se encargue de definir el aspecto visual de las aplicaciones y el comportamiento de ciertas áreas de usuario. Esto permitirá que los desarrolladores no se deban preocupar por este punto, ya que esto les vendrá dado automáticamente.

6.5.2 Patrón de diseño

Ilustración 35: Estructura vista



Los elementos estáticos ubicados en el servidor deberán seguir el patrón de diseño marcado en la Ilustración 34:

- css: Hojas de estilo.
- img: Imágenes.
- js: JavaScript.
- html: Páginas HTML estáticas
- ….: Resto de elementos de carácter estático.

Esta estructura permitirá la creación de reglas en el reverse proxy que permitan una mejor gestión de estos elementos (Cacheo, reubicación…).

6.5.3 Cuando nuestro usuario final es otra aplicación

Existirán algunos casos en que nuestro usuario final es un aplicación. En esos casos no dispondremos de capa de vista pero si de controlador, los métodos asociados a éste serán los responsables de exponer la funcionalidades que nuestra aplicación desee publicar a través de Web Services.

6.6 Convecciones de desarrollo

6.6.1 Código fuente

Dado que la totalidad de los desarrollos están basado en tecnología Java, en este apartado se van a definir una serie de convecciones que deben tenerse en cuenta en el proceso de desarrollo. El objetivo es tener un código claro y fácil de mantener por parte de todos los miembros del equipo y no sólo por la persona que llevo a cabo el desarrollo.

Con el objeto de simplificar y no inventar de nuevo estas buenas prácticas, se harán uso de las definidas por SUN en el documento *Convenciones de Código para el lenguaje de programación Java*.

6.6.2 Base de datos

Normal Generales

- Mantener nombres cortos y descriptivos.
- Todos los nombres de Tablas, Columnas, Constraints, Índices, Vistas y Triggers deben de indicarse en ingles
- Mantener nombres de objetos únicos, por ejemplo evitar crear la tabla WAREHOUSE y una o vista con el mismo nombre.
- Por defecto, no se aceptan espacios en blanco en medio de los identificadores.
- Para la definición de nombre de objetos de base de datos de acuerdo al caso se usará el carácter underscore "_" para separar las palabras_del_nombre. En caso de que se van a agrupar dos nombres iguales, se usará el doble underscore "__".
- No se utilizan acentos, ni caracteres especiales
- En los nombres de tablas/columnas donde se hace referencia a otras tablas y debido que en algunas bases de datos hay restricciones de longitud de nombre de tablas o columnas, limitar la referencia a las tablas a cuatro caracteres. Una opción puede ser coger los primeros cuatro caracteres.

Notación Base de datos

El nombre de la base de datos debe estar en **mayúsculas**.

Ejemplo:

Base de Datos: TEST

Notación tablas

- El nombre de la tabla debe ser descriptivo, en singular y en mayúsculas.
- Las tablas identifican una entidad del sistema con un nombre completo.
- Una tabla hija debe llevar al inicio el nombre de la tabla padre.
- Si la entidad o proceso que genera la tabla tienen más de una palabra se deberá colocar cada una de ellas en singular y deben ser separadas por un "_".
- Si una tabla es fruto de una relación M:N, la tabla que se genera debe de tener como nombre la combinación de ambas tablas.
- Si la entidad o proceso representa a una tabla de uso temporal, el nombre de la misma debe ser precedida por el siguiente prefijo “TEMP_”.

Ejemplo:

Nomenclatura de una tabla padre:	SUPPLIER
Nomenclatura de una tabla hija:	SUPPLIER _OBSERVATION
Nomenclatura de tablas fruto de una relación M:N :	SUPPLIER _CUSTOMER
Nomenclatura de una tabla temporal:	TEMP_XXX

Notación tablas agrupadas por aplicación o módulo

En algunos casos, dado el volumen del proyecto, será necesario identificar las tablas por aplicación o módulo. Por ese motivo, y sólo en aquellos proyectos que se requiera, la aplicación o módulos se identificarán mediante un alias de dos caracteres.

Ejemplo:

Alias de la aplicación:	AA
Nomenclatura de la tabla:	AA_SUPPLIER
Nomenclatura de una tabla hija:	AA_SUPPLIER _OBSERVATION
Nomenclatura de una tabla temporal:	AA_TEMP_XXX

Notación para campos

<Tipo>_<Nombre_columna>

Donde,

- <Tipo>: Indica el tipo de constraint, donde puede coger lossiguientes valores: PK

(primary key) o FK (foreing key). En los campos donde no se haga referencia a un campo constraint, no se indica nada. Opcional

- <Nombre_columna> :
    - Los nombres de los **campos** deben ser descriptivos, en singular y en mayúsculas. Obligatorio.
    - En caso de FK, se indica el nombre de la tabla foránea, más el nombre de la columna: <NombreTablaOrigen_Columna>

Ejemplo:

Unique:	UQ_EMPLOYEE
Foreing Key:	FK_COMPANY_PK_COMPANY
Columnas comunes:	NAME
	DOCUMENT

Notación para las constraint

<Tipo>_<Nombre>_<NN>

Dónde,

- <Tipo> :Tipo de constraint: PK, FK, CH (check), UQ (unique) o DF (default). Obligatorio
- <Nombre> :Nombre del la Constraint. Obligatorio
- <NN> :Número correlativo. Opcional

Notación para claves primarias:

PK + Nombre de la tabla.

Puede darse el caso que se quiera definir un primary key dentro de una tabla generada fruto de una relación M:N, en esos casos el nombre de la tabla, será la combinación del nombre de ambas tablas: <NombreTablaA_NombreTablaB>.

Ejemplo:

PK_SUPPLIER

PK_SUPPLIER_CUSTOMER

Notación para claves foráneas

Índices con llaves foráneas o relacionadas.

Nomenclatura:

FK_*NombreTablaOrigen_NombreTablaReferenciada*

Ejemplo:

FK_SUPPLIER_PURCHASEORDER

Notación para Unique

UQ_NemónicoTabla_NombreUnique

Ejemplo:

UQ_CUSTOMER_COD_CUSTOMER

Notación para valores por defecto

DF_NemónicoTabla_NombreColumna

Ejemplo:

DF_CUSTOMER_DATE

Notación para Check

CH_NemonicoTabla_NombreCheck

Ejemplo:

CH_CUSTOMER_COD_CUSTOMER

Notación para índices

I<XX>_ <Nombre>_<NN>

Dónde:

- <XX> :Tipo de índice (Unique (UQ), Clustered (CL), NonClustered (NC)).
- <Nombre> :Nombre del Índice. Obligatorio
- <NN> :Número correlativo. Obligatorio

Ejemplo:

IUQ_CODEMPLOYEE_01 (índice Unique)

ICL_CCUSTODY_01 (índice Clustered)

Notación para Trigger

<AA>_TRG_<Nombre>_<C><T>

Dónde:

- <AA> : Alias de la aplicación. Opcional
- V : Identificador para las Vistas
- <Nombre> : Nombre de la tabla en Mayúsculas. Obligatorio
- <C> : Es la clase de trigger: A (after) o B (before). Obligatorio
- <T> : Es el tipo de trigger: I (insert), U (update), D (delete). Obligatorio

Ejemplo:

TRG_TABLENAME_D (Cuando se realiza una eliminación en la tabla).

TRG_TABLENAME_U (Cuando se realiza una actualización en la tabla).

TRG_TABLENAME_I (Cuando se realiza una inserción en la tabla).

TRG_TABLENAME_IU (Cuando se realiza una inserción o actualización en la tabla).

Notación para vistas

<AA>_V_<Nombre>

Dónde:

- <AA> : Alias de la aplicación. Opcional
- V : Identificador para las Vistas
- <Nombre> : Nombre de la Vista en Mayúsculas. Obligatorio

Ejemplo:

V_CUSTOMER_CHECK

Para el nombre de las Vistas utilizar verbos en Infinitivo.

6.7 Control de versiones

6.7.1 Introducción

El código asociado a los productos desarrollados debe estar correctamente almacenado y gestionados para la totalidad de las versiones existente. Para este cometido se contará con un sistema SVN, existiendo un repositorio diferente por producto. La interacción con éste se realizará mediante las siguientes aplicaciones:

- Eclipse: Subversive
- Sin IDE: TortoiseSVN

6.7.2 Estructura

Todo repositorio del SVN contendrá la siguiente estructura:

- Trunk: Esta es la rama principal destinada al almacenar el código fuente que está siendo objeto de un desarrollo evolutivo.
- Tags: Cada vez que se generé un nueva versión destinada liberada a producción, deberá congelarse en código generando lo que se denominada un tag. En esta rama se albergan los diferentes tags del correspondiente producto. El código aquí ubicado no es susceptible a ser modificado.
- Branches: Este rama se alberga el código que se encuentre en una de las siguientes situaciones
    - Mantenimiento correctivo: Código fuente que se encuentra en mantenimiento, por ejemplo, correcciones sobre versiones liberadas, etc.
    - En caso de que existan desarrollos paralelos de un mismo producto, evidentemente uno saldrá antes y otro después, dado que ambos no pueden encontrarse en el Trunk, el que tenga una fecha más temprana se albergará en una rama y el que vaya a salir más tarde se albergará en el trunk.
    - En el caso de que vaya a realizarse un desarrollo experimental, sobre el cual el resultado es incierto, deberá crearse una rama con el objetivo de no dañar la rama principal.

Dentro del trunk se estructura el código de la siguiente forma:

- Database
- <módulo 1>
- <módulo 2>

- …
- <módulo n>

Dentro de branches se estructura el código de la siguiente forma:

- Versión de dos digitos X.Y (por ejemplo 1.0)_
  - Database
  - <módulo 1>
  - <módulo 2>
  - …
  - <módulo n>

Dentro de tags se estructura el código de la siguiente forma:

- Versión de dos digitos X.Y.Z (por ejemplo 1.0.0)_
  - Database
  - <módulo 1>
  - <módulo 2>
  - …
  - <módulo n>

Ahora se describirá que se alberga en cada carpeta (Database, en cada módulo, etc).

- **database**
  - Incluye los scripts de creación de la base de datos
  - Deberá existir un script de creación por versión del aplicativo
  - Deberá existir un script de actualización por versión del aplicativo
- **<módulo>**
  - Estructura de carpetas del proyecto que variará en función de la naturaleza del mismo.
  - Estructura típica proyecto java
    - src
    - properties
  - Estructura típica de un proyecto java con maven
    - src/main/java
    - src/main/resources
    - src/test/java
    - src/test/resources
- Estructura típica de un proyecto web java

- src
- properties
- WebContent

6.7.3 Subir código al SVN

A la hora del subir código al SVN, lo que normalmente se denomina COMMIT, se deberá seguir una serie de reglas descritas a continuación:

- Código sin errores
- Compilación correcta
- Uso de comentarios: Identificar la tarea en la que se está trabajando
- Desarrollo en grupo: Nuestras subidas no deben afectar negativamente al resto
- Contenidos:
  - Contenidos a almacenar en SVN
    - Código fuente
    - Scripts
    - Archivos de configuración (.properties, .ini, xml, etc.)
    - Contenido estático: imágenes, css, javascript, etc
  - Contenidos que no se deben almacenar en SVN
    - Binarios
    - Documentación (.doc)
    - Vídeos, imágenes pesadas
    - Imágenes de máquinas virtuales

6.7.4 Generación Tags

Como se indicado anteriormente, los tags son fotos de nuestro código o también se denomina “congelar el código”. Cuando se va a liberar una versión debe congelarse el código de forma indeleble, gracias a esto, tendremos acceso al código fuente correspondiente a liberación de una versión a producción.

A la hora de generar un tag el nombre del mismo deberá seguir la siguiente nomenclatura:

- Formato de la etiqueta: vX.X.X[sf] (Ejemplos: v1.0.0rc1, v 1.4.1b8, v2.1.1)
- Versiones de 3 dígitos X.X.X:
  - Dígito 1: Cambios significativos (rediseño, etc.)
  - Dígito 2: Nuevas funcionalidades
  - Dígito 3: Corrección de errores
- Sufijos:
  - rcX: Versión release candidate (rc1, rc2 … bn)
  - bX: Versión beta (b1, b2 … bn)

Al pasar al entorno de pre-producción se deberá generar la primera versión 'release candidate' (por ejemplo 1.0.0rc1), se irán incrementando las versiones rc si se encuentran errores en las pruebas asociadas a este entorno. Una vez validado el proyecto en el entorno de pre-producción, se generará la versión definitiva para el entorno de producción partiendo de la última rc probada y validada, para ello se renombrar la versión rc con el nombre de la versión para producción, por ejemplo, 1.0.0. Opcionalmente, después de crear la versión definitiva se podrán eliminar de SVN las versiones rc y beta que se hayan generado durante el desarrollo.

En los comentarios del código deberá aparecer el nombre de la versión en curso y sus notas de la versión. Si al cerrar una versión se están solucionando incidencias codificadas, se deberá especificar los códigos corregidos en las notas de la versión.

6.7.5 Generación Branches

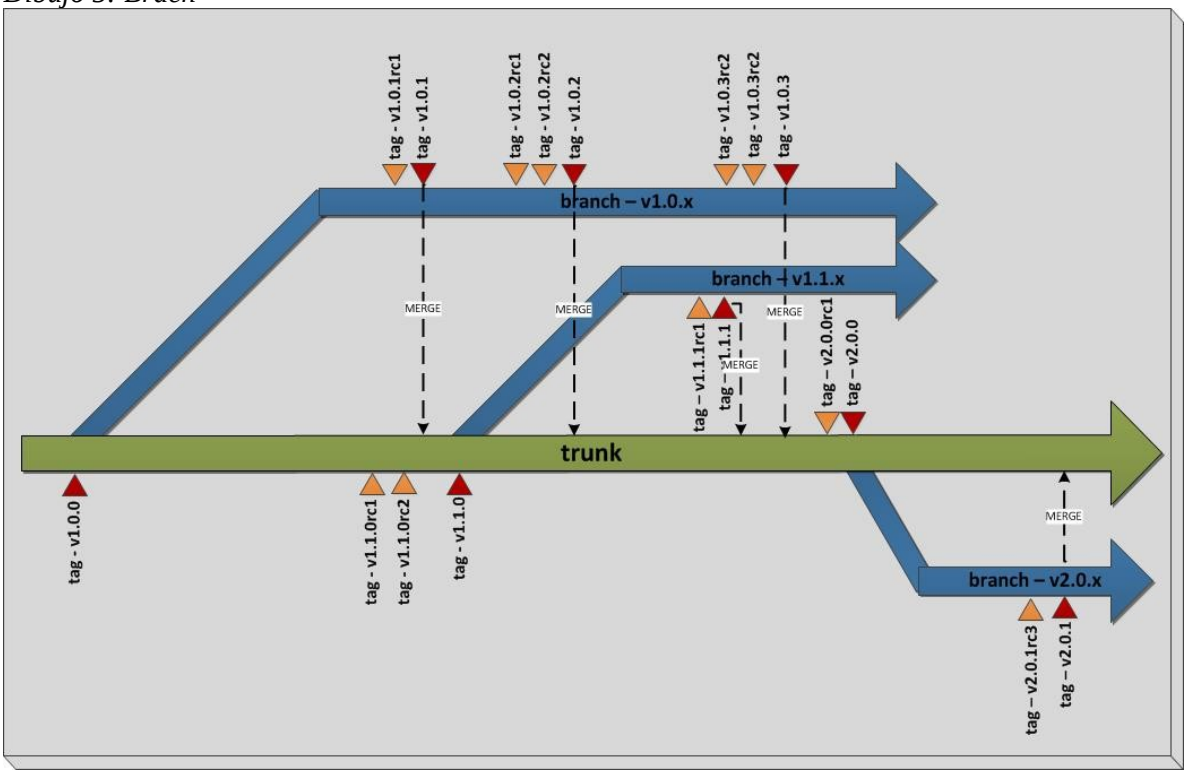
Como se ha comentado antes, el desarrollo principal (evolutivo) estará en trunk, por lo que se creará una rama para los mantenimientos correctivos, desarrollos experimentales y evolutivos que se den de forma concurrente. Las ramas creadas deberán seguir la siguiente nomenclatura:

- Para los desarrollos correctivos se escogerán los primeros dos dígitos de la versión, es decir,

  se creará una nueva rama por cada versión que contenga evolutivos, por lo que se tendrá versiones de 2 dígitos X.X:
  - Dígito 1: Cambios significativos (rediseño, etc.)
  - Dígito 2: Nuevas funcionalidades
- Ramas experimentales: Nombre descriptivo + sufijo '-exp'. Por ejemplo: 'rest-ws-exp'.

- Desarrollos en paralelo: Nombre de la versión a desarrollar en paralelo

Dibujo 5: Brach



Tal y como puede verse en el Dibujo 5, tanto los desarrollo correctivos como los evolutivos deberán estar sometidos a integraciones (merge) continuas y coordinadas con el trunk.

6.8 Auditoría Técnica

Al igual que existe una auditoría funcional, en donde se comprueba que la aplicación hace realmente lo que se ha determinado en las diferentes historias, debe existir una auditoría técnica en donde se valide que los desarrollos se hacen como se deben de hacer. Esta auditoría debe realizarse con cierta regularidad y será promovida por el área de arquitectura, como resultado se generarán una serie de no conformidades que deberán subsanase por el equipo afectado.

Uno de los primeros puntos a revisar, es que el proyecto sigue la estructura definida en en este documento dentro de los puntos 4.3 Modelo, 4.4 Controlador y 4.5 Vista. Además de visar lo establecido en el 4.6 Convenciones de desarrollo. Esta revisión en una inicio se realizará manual, siendo automatizada en un futuro por medio de Sonar (http://www.sonarqube.org/) el cual generará un reporte diario sobre las no conformidades existentes en el proyecto.

Otros punto a revisar, es el destinado a evitar la dispersión tecnológica, es decir, auditar que no se utiliza tecnologías no homologadas por el área de arquitectura. Además se deberá revisar la correcta

gestión del SVN, con el objetivo de que no exista código fuente no gestionado.

6.9 Documentación Técnica

El objetivo es encontrar un método de documentación ágil para los desarrollos asociados a los diferentes productos, de esta forma, los productos estarán debidamente documentados sin requerir un esfuerzo excesivo en esta tarea.

Deberá documentarse el código fuente a través de comentarios de Javadoc siguiendo estas sencillas normas:

- Deberá comentarse cada clase, indicando su objetivo y añadiendo toda información que pueda ser de interés en futuros mantenimientos.
- Deberá documentarse cada método, indicando su objetivo y añadiendo toda información que pueda ser de interés en futuros mantenimientos.
- Si el código fuente contiene algoritmos complejos difíciles de entender por un tercer desarrollador, este también deberá ser comentado.
- Si poseemos paquetes con una algorítmica compleja, estos también deberán ser comentados incluyendo toda la documentación necesaria que pueda ser de utilidad en futuros mantenimientos.

A partir de estos comentarios y mediante la herramienta Doxygen, se generará la documentación de arquitectura necesaria para cada versión liberada a producción.

7 Referencias Bibliográficas

Àles Alfonso I Minguillón, Eulàlia Clos Cañellas, Humberto Andrés Sanz, Isabel Domènech Puig-Serra, Jordi Schoenenberger Arnaiz, Proceso ingeniería del Software. Barcelona: UOC

Carlos Blé Jurando y colaboradores (2010), Diseño ágil con TDD (1a Edición). Safe

Creative

Henrik Kniberg (2007), Scrum y XP desde las trincheras (1ª Edición). Estados Unidos de

América: InfoQ

Henrik Kniberg & Mattias Skarin (2010) ,Kanvan y Scrum obteniendo lo menor de ambos (1ª Edición). Estado Unidos de América: C4Media

Jorge Hernán Abad Londoño (2013), Características de una buena historia de usuario. Colombia [Fecha de consulta: 5 de diciembre del 2015]

https://www.scrumalliance.org/community/articles/2013/august/caracteristicas-de-una-buena-historia-de-usuario

Jose Manuel Sanches Suarez (2014), Introducción a Spring Data: soporte para JPA. España [Fecha de consulta: 6 de diciembre de 2015]

http://www.adictosaltrabajo.com/tutoriales/spring-data-jpa

Lucho Salazar (2013), Escribiendo historias de usuarios altamente efectivas 2. Colombia [Fecha de consulta: 5 de diciembre de 2015]

htt://www.gazafatonarioit.com/2013/08/escribiendo-historias-de-usuario.html

Mike Cohin (2004), User Storis Applied For Agile Software Development. Estados Unidos

de América: Addison-Wesley Professional.

Ministerio de Hacienda y Administraciones Públicas (2001), Metríca 3. España [Fecha de consulta: 4 de diciembre de 2015]
http://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html#.VmMMFY3hDWU

Qaustral (2010), Manual de TestLink. Argentia [Fecha de consulta: 6 de diciembre de 2015]:

http://www.qaustral.com.ar/descargas/MTestLink.pdf

Santiago Codolá Vilahur y Pere Mariné Jové (2011), Metodología y gestión de proyecto (1ª Edición). Barcelona: informáticos (UOC)

Wikipedia (2015), Single page Application. [Fecha de consulta: 6 de diciembre de 2015]

https://es.wikipedia.org/wiki/Single-page_application