UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Adaptación del proceso de desarrollo software para cumplimiento de la adecuación funcional según ISO/IEC 25000

José Antonio Martínez López

Junio, 2017

ADAPTACIÓN DEL PROCESO DE DESARROLLO SOFTWARE PARA CUMPLIMIENTO DE LA ADECUACIÓN FUNCIONAL SEGÚN ISO/IEC 25000

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

Tecnologías y Sistemas de Información

TECNOLOGÍA ESPECÍFICA DE
INGENIERÍA SOFTWARE

TRABAJO FIN DE GRADO

Adaptación del proceso de desarrollo software para cumplimiento de la adecuación funcional según ISO/IEC 25000

Autor: José Antonio Martínez López

Director: Dr. Miguel Ángel Laguna Lobato

Director: Dr. Moisés Rodríguez Monje

Junio, 2017

José Antonio Martínez López

Ciudad Real – Spain

E-mail: JoseAntonio.Martinez7@alu.uclm.es
Teléfono: 671741264

TRIBUNAL:

Presidente:

Vocal:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE	VOCAL	SECRETARIO
Fdo.:	Fdo.:	Fdo.:

Resumen

Hoy en día la calidad es una propiedad fundamental en cualquier ámbito comercial y tecnológico. Si ya de por si la calidad es un concepto difícil de cuantificar, la calidad software es aún más difícil al ser un producto intangible y relativamente nuevo. Hasta hace poco esta calidad se medía exclusivamente a través de la calidad de los procesos del ciclo de vida (ISO/IEC 15504 e ISO/IEC 12207), pero a partir de 2005 surge el estándar ISO/IEC 25000 que permite la medición de la calidad del producto software en sí mismo.

Madrija Consultoría S.L. es una empresa joven que tiene como parte de su identidad la calidad software. Este trabajo surge de la necesidad de la empresa para certificarse en calidad Software, tras su reciente certificación en mantenibilidad según la norma ISO/IEC 25000. Ahora se centrarán los esfuerzos en lograr otra de las características de la norma ISO/IEC 25000, esta característica es la Adecuación funcional. Para conseguir esto se cuenta con el apoyo (a través de uno de los directores de éste trabajo) de AQCLab, primer laboratorio acreditado para la evaluación de la norma ISO/IEC 25000.

Tras analizar qué se necesita para alcanzar un nivel adecuado de Adecuación Funcional se ha detectado y analizado la manera de adaptar y mejorar los procesos de elicitación de requisitos y pruebas. A través de este trabajo, se presentan técnicas para la captura de requisitos y se han creado unas plantillas que ayudan a formalizar y gestionar los requisitos. También se crean unas plantillas para las pruebas funcionales y de aceptación, estableciendo también instrucciones de cómo usar todo esto a través del ciclo de vida de un proyecto. Además se ha evaluado de manera informal algunos procesos de desarrollo software de la empresa buscando alcanzar el nivel 2 de madurez del modelo ISO/IEC 15504-12207.

Abstract

Nowadays the quality is an essential property whether in a comercial or technological field. If quality is a difficult concept to quantify, software quality is even more difficult as it is an intangible and relatively new product. Until recently this quality is measured exclusively through the quality of the processes of the life cycle(ISO/IEC 15504 e ISO/IEC 12207), but in 2005 appears the new standard ISO/IEC 25000. This standard allows the quality measurement of the software product itself.

Madrija Consultoría S.L. is a young company that has as part of it identity the software quality. This document arises from the need of the company to get the software quality certification. The company has already the maintainability certification according the ISO/IEC 25000 standard. Now the efforts will be on get the certification in other ISO/IEC 25000 characteristics, the functional suitability. To achieve this is counted on the help (through the help of one of the conductors of this document) of AQCLab, the first software laboratory recognized for the ISO/IEC 25000 standard evaluation.

After analyze what is needed to achieve a good level in functional suitability it is detected and analyzed the way to adapt and improve the requirements elicitation process and test process. Throug this document are presented a set of techniques for the gathering of requirements, A set of templates are done that'll help the formalization and managment of the requirements. A template for the functional and aceptance tests are done. Also there are some instructions to apply this templates in the software lifecycle of the company. In addition, an informal evaluation of certain processes is done seeking to achieve the level 2 in the maturity model ISO/IEC 15504-12207.

Agradecimientos

Me gustaría dar las gracias a cada persona que me ha ayudado a llegar hasta aquí.

Doy las gracias a todos mis profesores, y todo aquel que me ha enseñado algo, desde primaria hasta aquí, de todos he aprendido algo lo mejor que he podido.

Agradezco especialmente estos últimos meses a mis dos directores de TFG, Miguel Ángel y Moisés, sin ellos este trabajo no habría salido igual de bien, gracias por guiarme.

Agradezco a mis amigos, a todos, los buenos ratos, las charlas, los debates y las risas que sirven de oasis en medio del estrés de trabajos, exámenes y rutina. Aunque no siempre nos podamos ver, estáis ahí.

Debo dar las gracias a mi familia. A mis tíos, abuelos y primos, a quienes, por circunstancias no veo por largos periodos pero a quienes quiero y se que me quieren. De un modo u otro, en mayor o menor medida, me habéis hecho llegar hasta aquí, gracias.

Por último pero en absoluto menos importante, gracias a mis padres Jose y Julia, por apoyarme todo este tiempo, por guiarme y cuidarme lo mejor que habéis podido durante éste tiempo, lo cual ha sido mucho más que suficiente, sé que me hubieseis apoyado en cualquier dirección que hubiese tomado mi vida. Y gracias a mis hermanos, Pablo y Nacho, gracias por protegerme y por dejarme protegeros, por reíros de mi, conmigo y por hacerme reír. Por acompañarme en la vida. Os quiero.

Con humor

José Antonio Martínez López

A mis padres, por cuidarme.
A mis hermanos, por incordiarme.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

HOY en día es común para las empresas adquirir certificaciones que garantizan la calidad de sus procesos con normas como la ISO/IEC 15504 [13] y la ISO/IEC 12207 [11] pero no es tan común que estas empresas se certifiquen en la calidad del producto software que desarrollan y/o adquieren. La certificación de la calidad en productos software es un concepto bastante nuevo, pues hasta hace poco la calidad del producto se asumía como consecuencia de tener unos procesos de desarrollo software de calidad.

1.1 Calidad Software

Actualmente la calidad software ha adquirido una gran importancia, esto se debe a la proliferación de los sistemas software, la mayor dependencia depositada en ellos y el mayor tamaño y complejidad que representan ahora con respecto al pasado, si tenemos en cuenta que los fallos en software pueden causar daños severos a nivel económico y personal como puede verse en [16, 27], todo esto provoca un auge en el interés por la calidad software.

La calidad del software es lo que determina que un cliente quede satisfecho tanto con el producto entregado como con la forma en que se ha llevado a cabo, tiempos y coste. Aunque la calidad del producto no dependerá exclusivamente de la calidad de los procesos, la mejora de la calidad de los procesos facilitará obtener una buena calidad del producto.

La calidad del software tradicionalmente, hasta no hace mucho, se medía conforme a los procesos desarrollados en el ciclo de vida de los productos. Estos procesos se miden tomando como referencia modelos como CMMI [2] o estándares como la norma ISO/IEC 15504, ésta norma explica qué necesitará el ciclo de vida de la empresa para cada uno de los niveles del modelo de madurez. Los niveles de madurez van del 0 al 5 siendo el 0 el mínimo nivel (empresa inmadura) y el 5 el máximo (empresa en continuo estado de mejora e innovación en desarrollo software). Ésta norma está relacionada con la ISO/IEC 12207, indicando qué procesos de la misma son útiles para cada nivel de madurez.

La primera familia de normas ISO/IEC 15504 se publicó en 1998 y en 2010, en España, se produce un auge de certificaciones en ésta norma por parte de las empresas de desarrollo software alcanzando actualmente más de 50 compañías certificadas [18, 23].

Sin embargo, los últimos años en calidad software no se han limitado únicamente a los

procesos de desarrollo. Así, desde el año 2005 se ha estado desarrollando la nueva familia de estándares ISO/IEC 25000 [12] centrada en evaluar la calidad del propio producto software. Y en el año 2012 nace AQCLab [26], el primer laboratorio acreditado por ENAC (Entidad nacional de acreditación) para la evaluación de la calidad del producto software, lo que ha permitido que la certificación de la calidad del propio producto software sea una realidad a día de hoy.

1.2 ISO/IEC 25000

Actualmente la certificación de ésta norma en pleno auge y siendo cada día más las empresas que la empiezan a seguir como modelo para controlar y evaluar la calidad del producto software. Actualmente la certificación de ésta norma no es excesivamente popular. Ésta norma divide al producto en 8 características diferentes, estas son: Adecuación funcional, eficiencia de desempeño, compatibilidad, usabilidad, fiabilidad, seguridad, mantenibilidad y portabilidad, además, ofrece unas subcaracterísticas que serán evaluadas por separado para evaluar de esta manera las características principales. El presente Trabajo Fin de Grado (en adelante TFG) se centrará concretamente en los requisitos para poder asegurar que una empresa cumple con la característica de la Adecuación Funcional propuesta por la ISO/IEC 25000.

Esta norma sustituye a la ISO/IEC 9126[10], modelo de calidad y la ISO/IEC 14598[9], modelo de evaluación.

1.2.1 Adecuación Funcional

Uno de los problemas que aparecen en las empresas dedicadas al desarrollo de productos software suele ser el hecho de que las características del producto desarrollado no encajan perfectamente con las especificaciones del cliente, dicho de otra manera: las funcionalidades del producto no se ajustan a los requisitos [17, 24, 4], no hay una buena **adecuación funcional**.

La adecuación funcional es una característica esencial en la calidad del producto software, esta característica es la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas. Consiste en básicamente tres subcaracterísticas según la norma internacional ISO 25000 [12]:

- **Completitud funcional.** El programa cumple con todas las funcionalidades que se le exigen.
- **Corrección funcional.** El programa cumple con éstas funcionalidades correctamente.
- **Pertinencia funcional.** El programa no hace cosas innecesarias o excesivas.

1.3 La empresa y el laboratorio

La empresa Madrija Consultoría, S.L. (a partir de ahora «Madrija»), junto con la que se realiza este TFG, ha realizado un esfuerzo, inusual en otras empresas del sector, para obtener la certificación en calidad de producto software. Hasta ahora Madrija ha conseguido (en la norma ISO/IEC 25000) la certificación para calidad en mantenibilidad, su objetivo ahora, es la calidad en adecuación funcional. Esta característica es de las más importantes pues permite garantizar que el cliente recibe lo que ha pedido. De ese objetivo surge el presente TFG.

La certificación que posee la empresa fue adquirida tras la evaluación realizada por AQ-CLab, el primer laboratorio acreditado para evaluar la norma ISO/IEC 25000. Este trabajo se realiza bajo la supervisión de un miembro de éste laboratorio que será director del TFG junto con el director de I+D+i de Madrija.

Mediante el desarrollo del TFG se creará la infraestructura necesaria en una empresa real del sector del desarrollo software para poder ser evaluada y certificada en adecuación funcional.

1.4 Estructura del documento

A continuación se ofrece el listado de capítulos de este trabajo junto con una breve descripción del mismo.

Capítulo 1: Introducción En este capitulo se está presentando el tema y los principales elementos relacionados con el mismo además de la estructura del documento.

Capítulo 2: Objetivos Aquí se exponen los objetivos de este TFG. Siendo estos, el objetivo principal y otros parciales en los que se dividirá el trabajo. También se exponen aquí los objetivos docentes.

Capítulo 3: Estado de la cuestión En este capítulo se expondrá todo el conocimiento teórico utilizado para el desarrollo del trabajo. En este caso se ven métodos de desarrollo, estándares, instituciones relacionadas, etc.

Capítulo 4: Método de trabajo Aquí se explicará que método de trabajo que se ha aplicado para desarrollar el TFG y las herramientas utilizadas en el proceso.

Capítulo 5: Resultados En esta parte del documento se exponen los resultados más relevantes del trabajo realizado.

Capítulo 6: Conclusiones y propuestas Se verá aquí una reflexión final, un sumario del trabajo realizado y unas líneas base para futuros trabajos.

ANEXOS Se muestra en este apartado un listado con los acrónimos y siglas usadas a lo largo del documento y un pequeño caso práctico usando el sistema desarrollado.

Capítulo 2

Objetivos del TFG.

IMPULSADOS por la necesidad de mejora constante, Madrija se ha propuesto, entre otros objetivos, cumplir la norma ISO/IEC 25000, concretamente en este trabajo se desarrollará la parte de Adecuación Funcional. Debido a que la empresa tiene establecidos una serie de procesos para el ciclo de vida de sus desarrollos, será importante llegar a un acuerdo con la empresa en cuanto a los cambios que se necesiten realizar.

2.1 Objetivo principal

El objetivo de este TFG será desarrollar la infraestructura necesaria para que el ciclo de vida de una empresa de desarrollo software (en este caso Madrija) cumpla con las necesidades para que el producto software que desarrollen pueda después ser certificado en adecuación funcional según ISO/IEC 25000.

Con este proyecto se revisarán los actuales procesos de: elicitación y gestión de los requisitos, desarrollo, gestión y ejecución de las pruebas de aceptación, al amparo de las normas ISO/IEC 25000 [12], ISO/IEC/IEEE 29119[14] ISO 15504[13, 23] e ISO12207[11, 23] .

2.2 Objetivos parciales

Para hacer más fácil conseguir el objetivo principal se divide éste en otros de menor escala con la meta de hacer más fácil su comprensión y alcance.

2.2.1 Objetivo 1

Analizar, diseñar e implantar un buen **sistema de adquisición y gestión de requisitos**. Esto servirá para saber perfectamente qué es lo que el cliente quiere que haga el software.

2.2.2 Objetivo 2

Analizar, diseñar e implementar un **sistema de pruebas de aceptación** [22], partiendo de los requisitos se creará un método, lo más sencillo posible, que asegure que los requisitos se han implementado correctamente en la aplicación desarrollada.

2.2.3 Objetivo 3

Se adaptará el flujo de trabajo del equipo de desarrollo al nuevo sistema de requisitos y de pruebas.

2.2.4 Objetivo 4

Automatizar las pruebas [20], en la medida de lo posible se buscará e implementará la manera de que la mayor cantidad de pruebas de aceptación sean automáticas.

2.2.5 Objetivo 5

Se sincronizará el trabajo sobre requisitos y pruebas con el **modelo de madurez SPICE** [13]. Comprobando qué procesos de la norma procesos descritos en la ISO/IEC 12207 [11] se pueden cumplir gracias al sistema nuevo y cuales se pueden cumplir sin cambios radicales.

2.2.6 Objetivo 6

Como complemento se buscarán **herramientas de gestión de pruebas** para seleccionar la que mejor se ajuste a la forma de trabajar de la empresa. Establecer un seguimiento de las pruebas.

2.3 Objetivos docentes

Además de los objetivos previos, existen una serie de objetivos enfocados a lo que aprenderá el alumno durante la elaboración del TFG

2.3.1 Objetivo docente 1

El alumno podrá trabajar en un caso real de aplicación en una empresa del sector.

2.3.2 Objetivo docente 2

El alumno demostrará dominio de conocimientos adquiridos sobre gestión de proyectos y gestión de requisitos software.

Capítulo 3

Estado de la cuestión

BUSCANDO la calidad del software este TFG dividirá los esfuerzos en dos vías distintas, la primera de ellas perseguirá la calidad del producto según el estándar ISO/IEC 25000, concretamente la parte que trata sobre la **Adecuación Funcional**. Por otra parte, se perfeccionarán los procesos del ciclo de vida tomando como referencia los estándar ISO/IEC12207 e ISO/IEC15504.

3.1 Calidad Software

A pesar de que se han dado definiciones muy dispares de lo que significa calidad por diferentes entendidos, la mayoría de expertos históricos coinciden en que la calidad de un producto no es un valor absoluto [7], depende del uso que se le vaya a dar al mismo.

La calidad software se puede definir como el conjunto de características que consiguen satisfacer las necesidades explícitas e implícitas.

Actualmente existe un crecimiento de los sistemas software en cantidad y tamaño, así como un aumento de la influencia de los mismos en una gran cantidad de sectores profesionales. Esto provoca que haya un interés creciente en la calidad Software [27]. Si se depende de algo, mejor poder confiar en que funcionará bien.

Asimismo un aliciente extra para certificar la calidad es la larga sucesión de errores en la industria software que se pueden observar en el **informe CHAOS** [16]

A lo largo de la corta historia de la informática se han dado numerosos errores de los que se han aprendido diversas lecciones. Estos errores causaron en su momento grandes pérdidas, en algunos casos a nivel económico y en casos más graves, vidas humanas como en el caso de las máquinas de radiodiagnosis Therac-25 [19]. **El informe Chaos**, es un informe realizado por "The Standish Group" y estudia unos 50,000 proyectos (dato de 2015) poniendo de manifiesto los fracasos y éxitos de los proyectos del desarrollo software. Los proyectos son calificados como éxitos, fracasos o deficientes en base a si terminaron en tiempo, presupuesto y cumplieron con los requisitos, dejando fuera aspectos como la calidad del código generado, el riesgo del proyecto o la satisfacción del cliente.

El último informe es de 2016, si observamos la figura 3.1 comparado con años previos se observa que hay menos fracasos (31 % en 1994 y 19 % en 2016), seguramente gracias a la experiencia general de la profesión y a aprender de errores, y aunque hay una mejora en los éxitos (16 % en 1994 frente a 29 % en 2016) sigue habiendo muchos proyectos que terminan de forma deficiente (53 % en 1994 y 52 % en 2016).

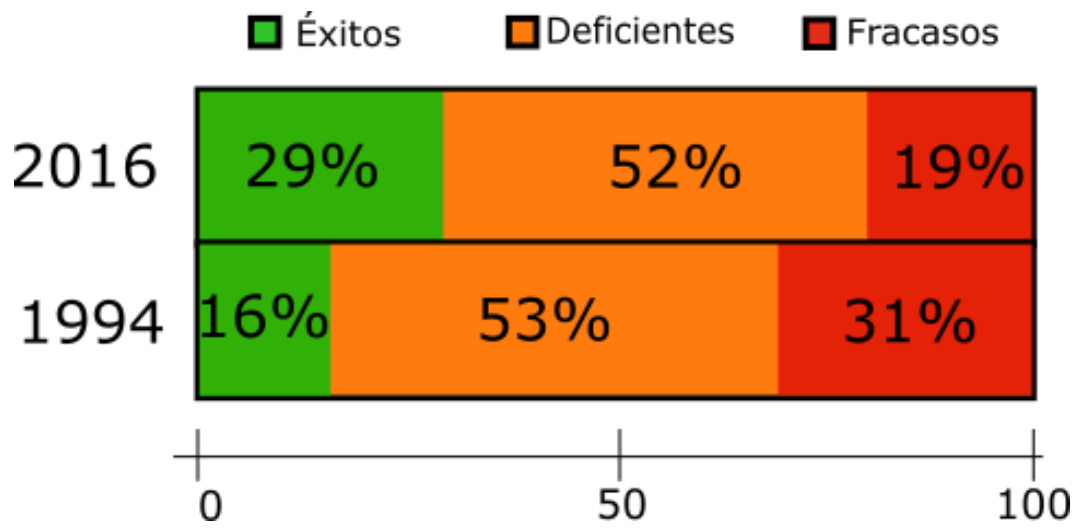


Figura 3.1: Informe Chaos 2016 frente a 1994 [16]

Para asegurar la calidad del software se disponen de dos enfoques fundamentales: calidad de los procesos utilizados en el desarrollo y calidad de los productos software en sí mismos.

Aunque la certificación para calidad de procesos si está bastante extendida no ocurre lo mismo con la certificación de la calidad de productos. Es posible que esto sea provocado por la novedad del estándar ISO/IEC 25000 (para calidad de productos software) y el estado incipiente de la calidad de los productos software (comparada con los procesos) [26].

3.2 Calidad de los procesos

Conseguir un método de desarrollo que garantice la adecuación funcional supone una serie de cambios en los procesos que se deberá controlar de alguna manera. estos procesos se pueden controlar a través del modelo de madurez, explicado en el estándar ISO/IEC 15504 [13]. El estándar ISO/IEC 15504 es una guía para la implementación, mejora y evaluación de los procesos explicados en el estándar ISO/IEC 12207 [11].

3.2.1 ISO 12207

El ISO/IEC 12207 es el estándar que describe la arquitectura de los procesos de ciclo de vida del software de la organización según ISO. El estándar no describe detalles de como

llevar a cabo estos procesos ni las tareas que componen los mismos. Esto viene determinado posteriormente por la metodología de desarrollo que cada empresa de desarrollo software implementa. Por otro lado, para evaluar los procesos de la ISO/IEC 12207 se utiliza el estándar ISO/IEC 15504.

3.2.2 ISO 15504

Comúnmente conocido como **SPICE** (*Software Process Improvement Capability Determination*). Esta norma tiene como objetivo servir de modelo para la evaluación y mejora de los procesos de desarrollo y mantenimiento de sistemas de información y productos de software.

Guiada por esta norma se llevará a cabo la evaluación de los procesos implantados en el ecosistema de la entidad evaluada y basados en los propuestos por la anteriormente comentada ISO/IEC 12207.

Modelo de Madurez AENOR para la ingeniería software

El modelo de madurez de Ingeniería de Software de AENOR [23] es una guía que tiene por objetivo favorecer la calidad en los productos software a través de controlar los procesos del ciclo de vida software.

EL modelo se estructura en niveles de madurez que van del 0 al 5 siendo 5 el máximo nivel posible. Estos niveles se evalúan para cada proceso, posteriormente el valor de cada proceso permite evaluar la madurez de la organización al completo.

Nivel 5	**Optimizado:** El proceso se mejora continuamente para cumplir los objetivos de negocio actuales y futuros.
Nivel 4	**Predecible:** El proceso se gestiona usando técnicas cuantitativas.
Nivel 3	**Establecido:** Se Utiliza un proceso adaptado pasado en un proceso estándar.
Nivel 2	**Gestionado:** El proceso se gestiona y los productos de trabajo se establecen, controlan y mantienen.
Nivel 1	**Realizado:** Existe evidencia de la realización del proceso.
Nivel 0	**Incompleto:** No hay procesos implementados

Figura 3.2: Niveles del modelo de madurez ISO 15504

El estándar tiene una estructura tal que busca poder ser adaptada a las necesidades de cualquiera que lo use. Para ello, el estándar se basa en dos principios fundamentales: modularidad y responsabilidad. El objetivo de la modularidad es conseguir procesos con un mínimo acoplamiento y una máxima cohesión. La responsabilidad permitirá facilitar la aplicación del estándar en proyectos en los que pueden existir distintas personas u organizaciones involucradas, asignando un responsable a cada proceso.

3.3 Calidad de productos software

A pesar de que la calidad en los procesos del ciclo de vida software ya disponían de un abanico de certificaciones como pueden ser (CMM, CMMI, ISO/IEC 15504, ISO/IEC 12207, ISO/IEC 9001). En 2005 surge el estándar ISO/IEC 25000 para el aseguramiento, evaluación y certificación de la calidad de los productos software. Este estándar permite a los desarrolladores de software acreditar que los productos que desarrollan son de calidad para sus clientes, y otorga a los clientes la garantía de que los productos adquiridos son de calidad.

3.3.1 ISO 25000

ISO/IEC 25000, conocida como **SQuaRE** (*System and Software Quality Requirements and Evaluation*), es una familia de normas que tiene por objetivo la creación de un marco de trabajo común para evaluar la calidad del producto software. El estándar ISO/IEC 25000 sustituye a las normas ISO/IEC 9126 e ISO/IEC 14598.

La ISO/IEC 25000 se divide en diferentes secciones, cada una de estas divisiones tiene una función determinada, estas secciones son:

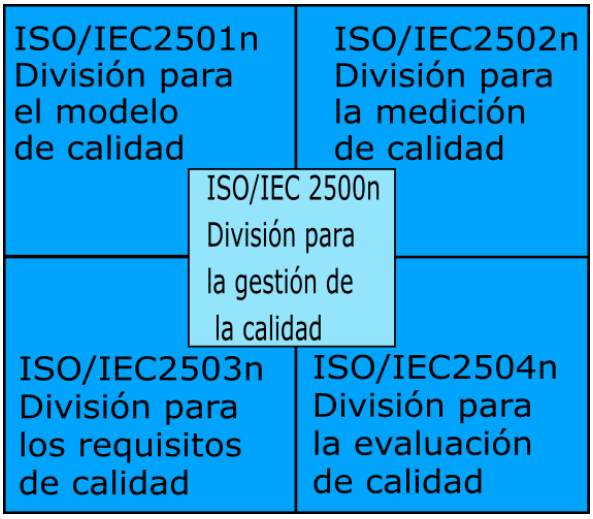


Figura 3.3: Divisiones de la norma ISO 25000 [8]

ISO/IEC 2500n – División de Gestión de Calidad

Las normas que forman este apartado definen todos los modelos, términos y definiciones comunes referenciados por todas las otras normas de la familia 25000.

ISO/IEC 2501n – División de Modelo de Calidad

Las normas de este apartado presentan modelos de calidad detallados incluyendo características para calidad interna, externa y en uso del producto software.

ISO/IEC 2502n – División de Medición de Calidad

Estas normas incluyen un modelo de referencia de la medición de la calidad del producto, definiciones de medidas de calidad (interna, externa y en uso) y guías prácticas para su aplicación.

ISO/IEC 2503n – División de Requisitos de Calidad

Las normas que forman este apartado ayudan a especificar requisitos de calidad que pueden ser utilizados en el proceso de elicitación de requisitos de calidad del producto software a desarrollar o como entrada del proceso de evaluación.

ISO/IEC 2504n – División de Evaluación de Calidad

Este apartado incluye normas que proporcionan requisitos, recomendaciones y guías para llevar a cabo el proceso de evaluación del producto software.

Concretamente la norma ISO/IEC 25010:2011 (*Quality model and guide*) es la que se tomará como referencia para la realización del TFG. El estándar ISO/IEC 25040:2011 es el estándar que se utiliza como referencia para la evaluación de la calidad de los productos Software.

DE las diferentes secciones del estándar ISO/IEC 25000, aquí se exponen la ISO/IEC 25040 y la ISO/IEC 25010.

ISO/IEC 25040

La ISO/IEC 25040 define el proceso para llevar a cabo la evaluación del producto software. Dicho proceso de evaluación consta de un total de cinco actividades.

1. Establecer los requisitos de la Evaluación
2. Especificar la evaluación
3. Diseñar la evaluación

4. Ejecutar la evaluación
5. Concluir la evaluación

En España AQCLab es el primer laboratorio acreditado para la **Evaluación de la Calidad del Producto Software** y **la Calidad de los Datos**en base al estándar internacional ISO/IEC 25000. Concretamente, AQCLab ofrece el servicio de evaluación de la Adecuación Funcional en línea con la ISO/IEC 25000 y en colaboración con la entidad certificadora AENOR, esta evaluación permite conocer el nivel de adecuación funcional del software, ya sea desarrollado o adquirido a terceros. Ayudando a diferenciarse de los competidores, asegurando tiempos de entrega y reduciendo fallos en el producto tras su implantación en producción facilitando así la obtención de la certificación entregada por AENOR para la ya mencionada norma. [6]

AENOR (Asociación Española de Normalización y Certificación) es la entidad privada española, sin ánimo de lucro, dedicada al desarrollo de la normalización y la certificación (N+C) en todos los sectores industriales y de servicios. AENOR es la responsable de otorgar, entre otros, el certificado que acredita la Adecuación Funcional conforme a el estándar ISO/IEC 25000.

El proceso de certificación que se sigue, definido en [8], es el siguiente:

1. El proceso comienza cuando la organización interesada en la calidad del producto software solicita una evaluación a un laboratorio acreditado, AQCLab por ejemplo. Para ello rellena un formulario con las características del producto software que quiere evaluar y el laboratorio lo analiza para emitir un contrato de evaluación con las condiciones del servicio. Habiendo aceptado este contrato, la organización hace entrega al laboratorio del producto software a evaluar. A partir de aquí, el laboratorio realiza la evaluación haciendo uso del entorno (modelo, proceso y herramientas) basado en ISO/IEC 25000 y acreditado por ENAC (Entidad nacional de acreditación). Este proceso suele tener una duración estimada de 2-3 semanas, dependiendo de las características del producto bajo evaluación.
2. El resultado del paso anterior es un informe de evaluación con los resultados obtenidos, este informe es entregado a la organización solicitante. En este paso, puede ocurrir que el nivel de calidad obtenido por el producto software no sea suficientemente bueno, en cuyo caso la organización solicitante, apoyada por los consultores expertos del ecosistema, deberán refactorizar el producto para mejorar el nivel de calidad. En este caso, el tiempo que puede transcurrir dependerá el número de defectos que se deben solucionar y de la cantidad de recursos que la organización pueda dedicar para tal fin. Una vez refactorizado el producto, la organización deberá repetir el paso 1 del proceso para volver a obtener un informe de evaluación favorable.
3. Cuando el producto software ha obtenido en la evaluación un nivel de calidad favo-

rable, la organización podrá contactar con una entidad certificadora, como AENOR, solicitando la certificación del producto e indicando la referencia previa de la evaluación que ha pasado realizada por un laboratorio acreditado.

4. La entidad certificadora contactará con el laboratorio evaluador para solicitar los resultados de la evaluación con la referencia indicada por la organización solicitante. Así, la entidad certificadora confirmará la veracidad de la evaluación y los resultados indicados por la organización solicitante.
5. El laboratorio colaborador revisará sus registros de evaluación y facilitará dicha información a la entidad certificadora.
6. Finalmente, la entidad certificadora analizará el informe de evaluación facilitado por el laboratorio y realizará una visita a la organización solicitante para, siguiendo con su reglamento interno de auditoría definido para el producto software, revisar el producto y las características del mismo. Como resultado de este proceso de auditoría de certificación, la entidad certificadora emitirá un informe y entregará a la organización un certificado que acredite la calidad del producto software evaluado. Este informe identifica entre otros a la organización solicitante, el producto certificado y su versión concreta, las características de calidad del modelo evaluadas y el informe del laboratorio acreditado que recoge los resultados de evaluación sobre los que se soporta el certificado emitido.

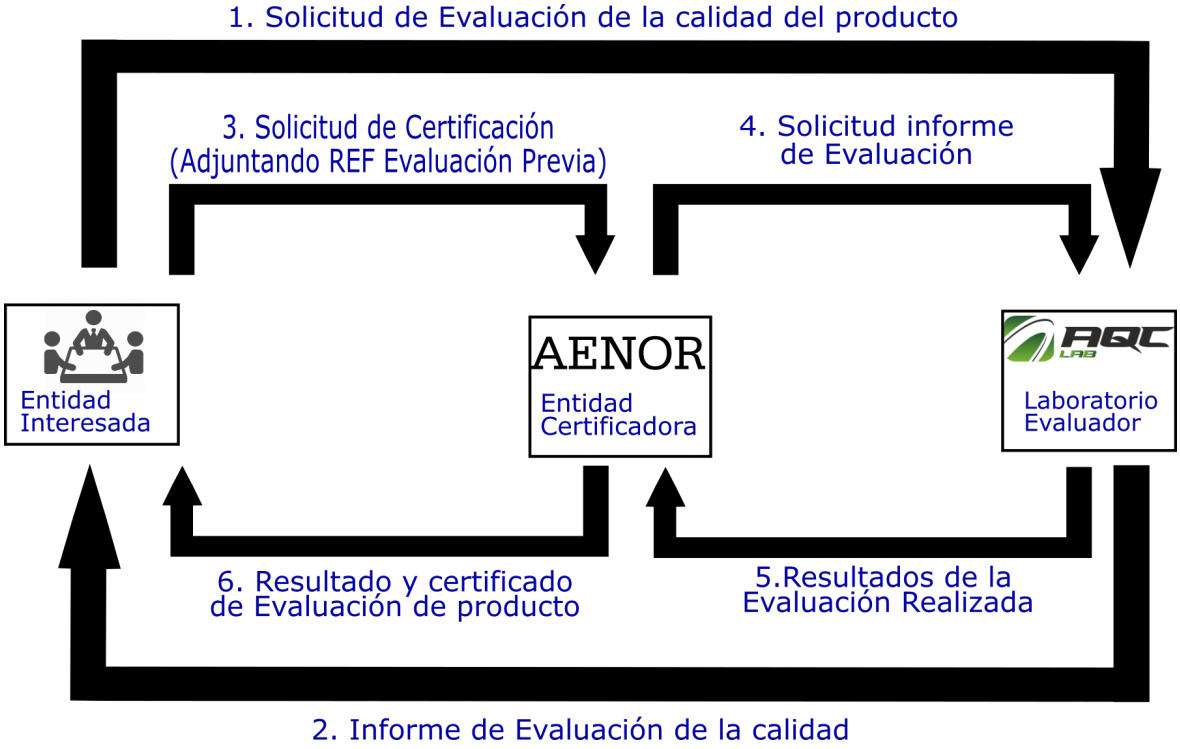


Figura 3.4: Proceso de Certificación ISO 25000 [8]

ISO/IEC 25010

La ISO/IEC 25010 representa el modelo de calidad y es la base a partir de la cual se establece el sistema para la evaluación de la calidad del producto. Este modelo describe una serie de características de calidad que se van a tener en cuenta para evaluar las propiedades de un producto software. En ésta norma es donde se puede ver la descripción de la adecuación funcional, que es la propiedad principal que interesa en la realización de éste TFG.

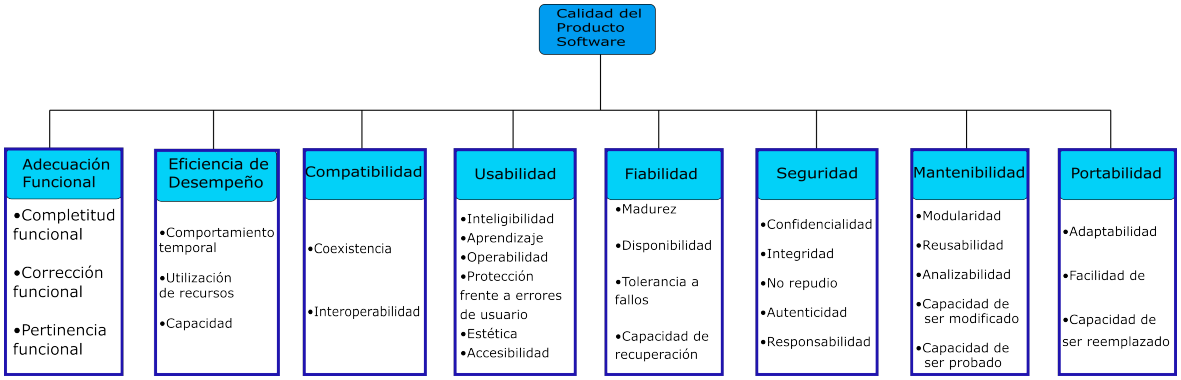


Figura 3.5: Características de la calidad según "http://iso25000.com-[8]

Adecuación funcional

La adecuación funcional consiste sencillamente en que el programa haga lo que se le pide, que lo haga bien y que haga solamente eso, nada que no se pida. Según el estándar ISO/IEC 25010 [12], la adecuación funcional: "Representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas". Esta misma norma divide esta característica en las siguientes subcaracterísticas:

Completitud funcional. "Grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos del usuario especificados". Es decir, el sistema hace lo que se pide que haga, todas funciones que el cliente/usuario pide.

Corrección funcional. "Capacidad del producto o sistema para proveer resultados correctos con el nivel de precisión requerido". Lo que significa que las funcionalidades del sistema tienen el resultado esperado.

Pertinencia funcional. "Capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados". Esta es la capacidad del sistema para ejecutar únicamente las funciones necesarias para que el sistema realice las tareas y objetivos que han sido especificados por el usuario y no otros.

Propiedades de la Adecuación Funcional

Debido a que uno de los dos tutores de este trabajo pertenece a AQCLab se cuenta con la inestimable ayuda del primer laboratorio acreditado para evaluar la calidad del producto software. Este laboratorio ha ideado una serie de propiedades las cuales complementan para valorar las diferentes características de la ISO/IEC 25000. En concreto, las propiedades de la adecuación funcional, la cual es particularmente relevante se pueden ver en [25]. Según éste artículo, las subcaracterísticas de la Adecuación Funcional poseen una serie de propiedades que pueden ser evaluadas dándoles unos valores. Una vez calculados los valores de cada subcaracterística se puede obtener el nivel de calidad de la adecuación funcional. Las propiedades de las subcaracterísticas son las siguientes.

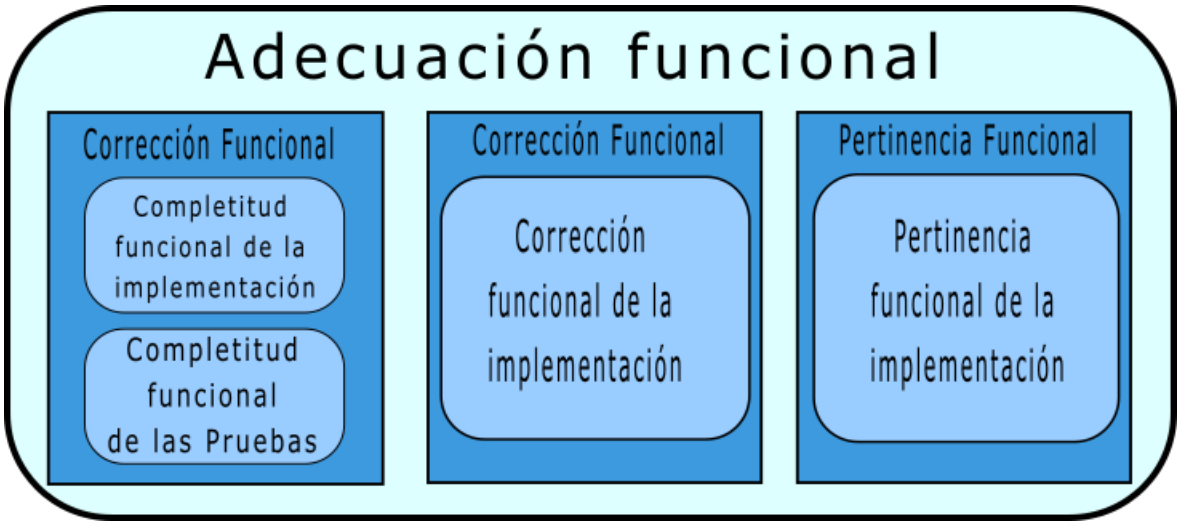


Figura 3.6: Adecuación Funcional

Completitud funcional.

Completitud funcional de la implementación. Esta propiedad tiene en cuenta las tareas específicas que el software puede llevar a cabo.

Completitud funcional de las pruebas. Esta propiedad observa si los requisitos que se describen en la especificación se someten a un proceso de pruebas. **La completitud funcional se evalúa comparando los requisitos sometidos a pruebas y los requisitos implementados.** Si un requisito no se somete a prueba no se puede saber si se ha implementado completamente.

Corrección funcional.

Corrección funcional de la implementación. Esta propiedad evalúa el nivel de corrección en la implementación de requisitos, evalúa si los requisitos implementados hacen lo que se esperaba de ellos, para ello se debe comprobar que se han completado los requisitos de la especificación de requisitos.

Pertinencia funcional.

Pertinencia funcional. Esta propiedad hace referencia a si el software complace los requisitos de cada tipo de usuario. Los requisitos para cada tipo de usuario se denominan *objetivos de uso* así que en definitiva se busca que todos los *objetivos de uso* se satisfagan.

A continuación se muestra una tabla con las subcaracterísticas y un resumen de como se evalúan las diferentes propiedades de las mismas.

Subcaracterística	**Propiedad**	**Procedimiento**
Completitud Funcional	Completitud Funcional de la implementación	Se medirá el número de requisitos implementados y el número de requisitos de la documentación específicados por el cliente o por el Product Owner (PO). Con esto se obtiene la lista de requisitos implantados en el producto final.
	Completitud Funcional de la implementación	Se usará el número de requisitos implementados junto con el número de requisitos probados y el porcentaje de cobertura para ejecutar los sistemas de casos de prueba funcional. Para medir la cobertura se necesitará usar herramientas como Emma, Opencover, Simplecov, etc. Así obtendremos la lista de requisitos probados, asegurando de esta manera que han sido completamente implementados.
Corrección Funcional	Corrección Funcional de la implementación	Se utiliza el número de requisitos implementados junto con el número de requisitos que se han probado, así como si los resultados han sido satisfactorios, y el porcentaje de cobertura de la declaración para ejecutar los sistemas de casos de prueba funcional. Se obtendría así una lista de los requisitos que presentan el comportamiento deseado por el usuario, y que fue establecido durante el análisis del producto por parte de Usuarios o clientes.

Subcaracterística	Propiedad	Procedimiento
Pertinencia Funcional	Pertinencia Funcional	Se necesita identificar los requisitos de cada objetivo de uso, asi como la proporción de requisitos donde el resultado es correcto para cada objetivo de uso y el grado en el que cada objetivo de uso es apropiado. De esta forma se puede descubrir si el producto software es adecuado

Cuadro 3.1: Propiedades de la Adecuación funcional [25]

3.3.2 Qué se necesita para conseguir la Adecuación Funcional

Una vez analizado el estándar ISO/IEC 25000 y en especial la característica de calidad adecuación funcional en la que se centra el presente TFG, se ha detectado que para poder trabajar de acuerdo al estándar, garantizar que los productos software que se desarrollan cumplen con esta característica y asegurar que posteriormente el producto podrá ser evaluado y certificado en adecuación funcional, se necesitan básicamente dos elementos.

- Un buen **sistema de requisitos** que permita conocer a la perfección qué funciones se tienen que desarrollar.
- Un buen **sistema de pruebas funcionales** que verifiquen y validen los requisitos de modo que sepamos que el producto software hace lo que debe según los requisitos definidos y que lo hace bien.

La forma en que se alcancen estos elementos dependerá de varios factores como pueden ser las metodologías que se apliquen/quieran aplicar, el tipo de software que se desarrolle, la idiosincrasia de la empresa o incluso el rango de edad de los trabajadores de la misma entre otras.

3.4 Requisitos

Uno de los elementos fundamentales que permitirán conseguir una buena adecuación funcional es la gestión de los requisitos. Los requisitos son los elementos que nos indicarán qué debe hacer el sistema a desarrollar y cómo debe hacerlo.

3.4.1 Captura de Requisitos

La captura de requisitos consiste en obtener las necesidades del programa o sistema que se va a desarrollar, ya sean requisitos funcionales (Definen una función del sistema) o no funcionales (Definen características de calidad y limitaciones del sistema). Para realizar esta captura de requisitos existen un variado número de técnicas, entre otros: Entrevistas, cuestionarios, brainstorming o revisión documental. Se exponen a continuación algunas de las técnicas más relevantes [24, 4].

Workshops

Los Workshops consisten en reuniones grupales estructuradas. Estas reuniones congregan personas implicadas de diferentes partes del negocio. En estas reuniones existirá un «facilitador» o «director» cuyo papel será el de descongestionar determinadas situaciones para que la reunión siga siendo de utilidad. Estas reuniones se suelen hacer en intervalos de trabajo cortos que permitan mantener la concentración de los participantes, otro papel importante en estas reuniones es el del «secretario» que registrará las ideas recogidas, dependiendo de quien lo aplique éste secretario será un experto analista o simplemente un escriba. Es común la participación, entre otros, de «Expertos» de materias diferentes, «representantes» de los usuarios y también de «patrocinadores» que representan la parte más ligada al Negocio.

Hay dos maneras muy comunes de aplicar los workshops, estas son los «Brainstormings» y los «JAD»

Brainstorming

El «Brainstorming» o «Tormenta de ideas» es una técnica ampliamente usada en diversos ámbitos, no solo en la ingeniería software. Esta técnica es muy útil en situaciones en las que se necesita generar nuevas ideas o conocer puntos de vista distintos de forma rápida.

La técnica consiste en reunir un grupo de unas 4-10 personas más o menos y que cada uno exponga sus ideas.

Esta Técnica se divide en cuatro fases: Preparación, generación, consolidación y documentación.

Preparación: Esta fase consistirá en seleccionar a los participantes (clientes, usuarios, ingenieros de requisitos, desarrolladores, algún experto en temas relevantes,etc.) y al jefe de la sesión, citarlos, preparar la sala donde se llevará a cabo la sesión.

Generación: Fase que se produce durante la sesión, el jefe de la sesión inicia la sesión exponiendo con claridad el enunciado o problema a resolver, este enunciado será la «semilla» a partir de la cual se irán dando ideas nuevas. Las ideas serán expuestas de forma visible para

todo el mundo, por ejemplo en una pizarra o mediante cartulinas en un tablón. El jefe de la sesión decidirá cuando finaliza la reunión, bien porque no surgen suficientes ideas y otorga un periodo de descanso/reflexión, bien porque considera que hay suficientes ideas para pasar de fase.

Los puntos claves de ésta fase son:

- Las ideas serán escuchadas por todos y se prohíbe su critica. Los participantes deben sentirse libres de hacer cualquier aporte.
- Se buscará que surjan la mayor cantidad posible de ideas. Cuantas más ideas, más posibilidades de que surjan mejores ideas.
- Se profundizará en las ideas más vanguardistas aunque se vean poco viables.
- Se animará la combinación de ideas o la mejora de ideas previas. Para ello es importante que las ideas sean visibles en todo momento para todo el mundo.

Consolidación Esta fase servirá para organizar y evaluar las ideas generadas durante la fase previa, normalmente se llevan a cabo 3 tareas:

- **Revisar ideas:** se revisan las ideas generadas para clarificarlas. Es habitual identificar ideas similares, en cuyo caso se unifican en un solo enunciado.
- **Descartar ideas:** aquellas ideas que los participantes consideren excesivamente vanguardistas se descartan.
- **Priorizar ideas:** se priorizan las ideas restantes, identificando las **absolutamente esenciales**, las que estarían bien pero que **no son esenciales** y las que podrían ser **apropiadas para una futura revisión** del sistema a desarrollar.

Documentación: después de la sesión, el jefe genera la documentación pertinente conteniendo las ideas priorizadas y los comentarios generados en la fase de consolidación.

JAD

Joint **A**plication **D**evelopment es una técnica desarrollada por IBM en 1974. Esta técnica consiste en un conjunto de reuniones en grupo durante un periodo de 2 a 4 días. En estas reuniones se ayuda a los clientes y usuarios a formular problemas y explorar posibles soluciones, involucrándolos y haciéndolos sentirse partícipes del desarrollo.

JAD consta de dos grandes pasos «JAD/Plan» cuyo objetivo es elicitar y especificar requisitos, y el «JAD/Design» en el que se aborda el diseño del software. Este documento se centrará en la fase de «JAD/Plan».

Los tipos de participantes que JAD contempla para las sesiones están claramente definidos a diferencia de otras técnicas, estos participantes son:

Líder de sesión JAD: Será el responsable del éxito del proceso, se encargará de guiar las sesiones y facilitar lo necesario para las mismas.

A pesar de que todos los participantes deberían conocer la técnica JAD el líder debe estar especialmente instruido. El líder JAD deberá tener capacidades tales como iniciar y centrar las conversaciones en lo que el proyecto necesite, entender y facilitar las dinámicas de grupo, reconducir las reuniones cuando se estén desviando del tema importante, mantener el ánimo en reuniones especialmente largas o tediosas y otras habilidades que se engloban en las destrezas de **liderazgo y comunicación.**

Analista/s: Puede haber uno o varios en la sesión. El analista es el responsables de la documentación generada en cada sesión. No es un escribano, deberá estar instruido y comprender los detalles técnicos. Es importante que el analista sea capaz organizar ideas y expresarlas de forma clara al escribir, si se necesita alguna herramienta software para la sesión (para crear mockups por ejemplo) el analista debe dominarla. Es recomendable que el analista sea un desarrollador experimentado.

Patrocinador ejecutivo: Es el último responsable a la hora de decidir si el proyecto se lleva a cabo o no. Tiene dos responsabilidades en el proceso JAD, la primera es proporcionar a los demás participantes información sobre la necesidad del nuevo sistema y los beneficios esperados al implantarlo, su otra responsabilidad es tomar decisiones a nivel ejecutivo y llevar a cabo compromisos, como asignaciones de recursos, que podrían afectar los requisitos y el diseño del nuevo sistema.

Representantes de los usuarios: Personas de la organización que usaran el sistema. Para la fase de «JAD/Plan» suelen involucrarse *managers* y personas clave de la organización. Éstos suelen tener una visión más global del sistema y de cómo se usará. Durante el «JAD/Design» incluirá un «popurrí» de usuarios, al elegir a estos usuarios se elige gente que conozca bien su departamento, las necesidades que tiene y las relaciones con otros departamentos. Estos representantes de los usuarios deberían tener ideas innovadoras y originales y no deberían temer hablar en las reuniones.

Representantes de sistemas de información: Personas expertas en sistemas de información, su función en las reuniones será aconsejar que se puede o no hacer a nivel tecnológico. Quizá los usuarios no son conscientes del potencial de la tecnología disponible o por el contrario la sobrevaloran.

Especialistas: Su objetivo es proporcionar información detallada sobre aspectos específicos. Podría existir por ejemplo un especialista que domine como funciona la intranet de la empresa o cómo están organizados los servidores. Otro tipo de especialistas son aquellos que dominen el conocimiento que tratará el sistema o las necesidades de los usuarios (más allá de los representantes de usuarios).

Las fases de la técnica JAD son:

Adaptación. Para optimizar las ventajas de JAD, el líder JAD junto con el/los analistas personalizarán el proceso, los pasos de esta personalización se detallan a continuación.

1. **Dirigir la orientación:** En esta fase el líder y los analistas se familiarizarán con la compañía y conocerán el objetivo del proyecto así como la motivación inicial del mismo. Estudiarán también el estado actual y las decisiones tomadas hasta ahora si las hay.
2. **Organizar el equipo:** El líder de la sesión tiene la responsabilidad de seleccionar quien estará en la reunión, aunque el patrocinador ejecutivo puede influir y recomendar a quien él considere oportuno la decisión final es del líder.

   El líder es el encargado de comunicar fecha, hora y lugar para la reunión a todos los participantes, con el objetivo de mentalizarles también les deberá proporcionar una serie de preguntas para pensar antes de la sesión. Estas preguntas estarán enfocadas en los objetivos de alto nivel que se pretenden obtener en la reunión. Cada participante deberá centrarse en su punto de vista, por ejemplo, el representantes de sistemas de información se centrará en los asuntos tecnológicos, los usuarios en lo que necesitan, cada especialista en su campo, etc.
3. **Ajustar el proceso:** El líder de sesión determinará sobre todo, cuantas sesiones serán necesarias, cuanto duraran y también ajustar el Documento general de requisitos JAD a las necesidades que prevea.
4. **Preparar el material:** Desde reservar la sala hasta comprar bolígrafos, todo lo que prevea necesitar deberá ser preparado, ayudas visuales, tarjetas en blanco, pizarra, etc.

Sesión. Esta fase es la reunión en sí misma, aquí los participantes exponen sus ideas guiados por el líder de sesión, estas ideas se discuten, ajustan y refinan hasta que se llega a un acuerdo.

1. **Presentación y puesta en marcha:** Se presenta y se da la bienvenida a todos los participantes por parte del patrocinador ejecutivo y del jefe del JAD. El patrocinador ejecutivo expone brevemente las necesidades que han conducido al desarrollo y los beneficios que se esperan obtener. El jefe del JAD explica la mecánica de las sesiones y la planificación prevista.
2. **Definir objetivos y requisitos:** En este paso, el líder del JAD impulsará la discusión para elicitar los objetivos o requisitos de alto nivel mediante preguntas como: "¿Por qué se construye el sistema?", "¿Qué beneficios se esperan del nuevo sistema?", "¿Cómo puede beneficiar a la organización en el futuro?", "¿Qué restricciones de recursos disponibles, normas o leyes afectan al proyecto?", etc. . Según se vaya desarrollando la sesión serán unas preguntas u otras las más importantes, esto se deja a la pericia del líder del sesión.

Los requisitos recogidos según las respuestas a esas preguntas, estos serán recopilados por el analista en un documento visible para todos durante la sesión, pueden servir desde transparencias proyectadas hasta tarjetas escritas a mano.

3. **Delimitar el alcance del sistema:** Tras el debate se generaran una serie de requisitos, necesidades, peticiones, etc. . A partir de todo esto se escogerá qué es responsabilidad del proyecto, delimitando así el alcance del sistema.
4. **Temas abiertos y próximas sesiones** Si tras el fin de la sesión han quedado temas por resolver o han surgido temas nuevos tras la reunión, se estimará cuales se deben resolver, si se debe hacer una reunión para resolverlos y cuando sería la hipotética reunión. Los conflictos se asignan a una persona que deberá resolverlos antes de la próxima sesión.
5. **Conclusión de la sesión:** El líder de JAD concluye la sesión tras revisar con el resto de participantes la información elicitada y las decisiones tomadas. Se da a los participantes la oportunidad de hacer alguna aclaración final o consideración adicional. El líder de sesión deberá animar a esto con el sentimiento de propiedad del proyecto desarrollado.

Conclusión final. Tras terminar todas las sesiones se transformarán las transparencias, tarjetas, notas y el resto de documentos recogidos en ellas en otros documentos más formales.

1. **Completar la documentación:** los analistas recopilan la documentación generada durante las sesiones en documentos conformes a las normas o estándares vigentes en la organización para la que se desarrolla el proyecto.
2. **Revisar la documentación:** la documentación generada se envía a todos los participantes para que la comenten. Si los comentarios son lo suficientemente importantes, se convoca otra reunión para discutirlos.
3. **Validar la documentación:** una vez revisados todos los comentarios, el jefe del JAD envía el documento al patrocinador ejecutivo para su aprobación. Una vez aprobado el documento se envían copias definitivas a cada uno de los participantes.

Método KJ, Diagrama de afinidad

Creado por el Dr. Kawakita Jiro en 1980. La premisa de esta herramienta es sintetizar un conjunto de datos verbales (ideas, opiniones, temas, expresiones, entre otros)

Se recomienda el uso de ésta herramienta en situaciones en las que:

- Se quiere organizar un conjunto amplio de datos.
- Se pretende abordar un problema de manera directa.

- El tema sobre el que se quiere trabajar es complejo.
- Es necesario el consenso del grupo.

La virtud de esta herramienta es su potencia para organizar los datos. Para ello se siguen los siguientes pasos:

1. **Determinar la pregunta enfoque.**

   El «facilitador» (quien organiza y guía la reunión) explica en qué va a consistir la reunión, de qué fases consta y qué se espera de los participantes. El tema a analizar se expone brevemente en forma de pregunta. Éste tema debe estar visible durante el tiempo de aplicación de la técnica.

2. **Generación silenciosa de ideas.**

   Cada miembro del grupo expresa sus ideas en tarjetas, post-its, fichas de papel o cualquier otro medio, solamente una idea por cada tarjeta. Se concede un tiempo de 5 a 10 minutos en el que los participantes no deben comunicarse entre sí.

3. **Exposición de ideas.**

   Tras el tiempo concedido para la generación de ideas, el «facilitador» recupera las tarjetas escritas por los participantes y las mezcla para que éstas sean expuestas de forma aleatoria. Al sacar cada tarjeta, si la idea no se comprende bien la persona que la escribió deberá explicarla.

4. **Agrupación de ideas.**

   A continuación se agrupan las ideas en el diagrama de afinidad. Para ello puede utilizarse un panel en el que se sitúan las ideas a medida que van siendo agrupadas. Si una idea puede estar en varios grupos a la vez se duplica el papel. Cada grupo de ideas será nombrado con un título representativo que se decidirá en consenso. si hay fichas que no se han podido agrupar, se añadirán al grupo «varios» o «miscelánea».

5. **Jerarquización.**

   Se asigna una prioridad a cada grupo de ideas, con el sistema de votación preferido para llegar a un acuerdo.

6. **Conclusión.**

   Se comenta el diagrama y tras ser aceptado por los participantes, y se cierra la reunión.

3.4.2 Modelado de requisitos

Tras obtener los requisitos, estos se deben transformar de modo que puedan ser entendidos por el equipo de desarrollo y en el mejor de los casos por la totalidad de los implicados en el desarrollo (*stakeholders*, jefe de proyecto, desarrolladores, etc.). Lo normal es plasmar los

requisitos recogidos en una serie de documentos, los métodos más comunes son: **Casos de uso** (técnica ligada a los desarrollos más tradicionales) e **Historias de Usuario** (técnica más relacionada con las metodologías ágiles).

Casos de Uso

Los casos de uso [15] son descripciones de los pasos o actividades que se tienen que realizar para llevar a cabo algún proceso. Los personajes o entidades implicados en el proceso son denominados actores. En el contexto de la ingeniería software un caso de uso es una secuencia de eventos que se producen en un sistema a consecuencia de una acción que realiza un actor principal. Los casos de uso pueden ser útiles para establecer requisitos de comportamiento, pero no establecen completamente los requisitos funcionales ni permiten determinar los requisitos no funcionales. Los casos de uso deben complementarse con información adicional como reglas de negocio, requisitos no funcionales, diccionario de datos que complementen los requisitos del sistema. Sin embargo la ingeniería del funcionamiento especifica que cada caso crítico del uso debe tener un requisito no funcional centrado en el funcionamiento asociado. Un Caso de Uso lo compone información muy diversa, esta información está compuesta de los siguientes campos (puede haber casos para los que no aplique):

- ID
- Nombre
- Referencias cruzadas
- Creado por
- Última actualización por
- Fecha de Creación
- Fecha de última actualización
- Actores
- Descripción
- Trigger
- Precondición/es
- Postcondición/es
- Flujo normal
- Flujos alternativos
- Includes
- Frecuencia de uso

- Reglas de negocio
- Requisitos especiales
- Notas y asunto

Normalmente los casos de uso se organizan en diagramas que relacionan a cada actor con sus casos de uso.

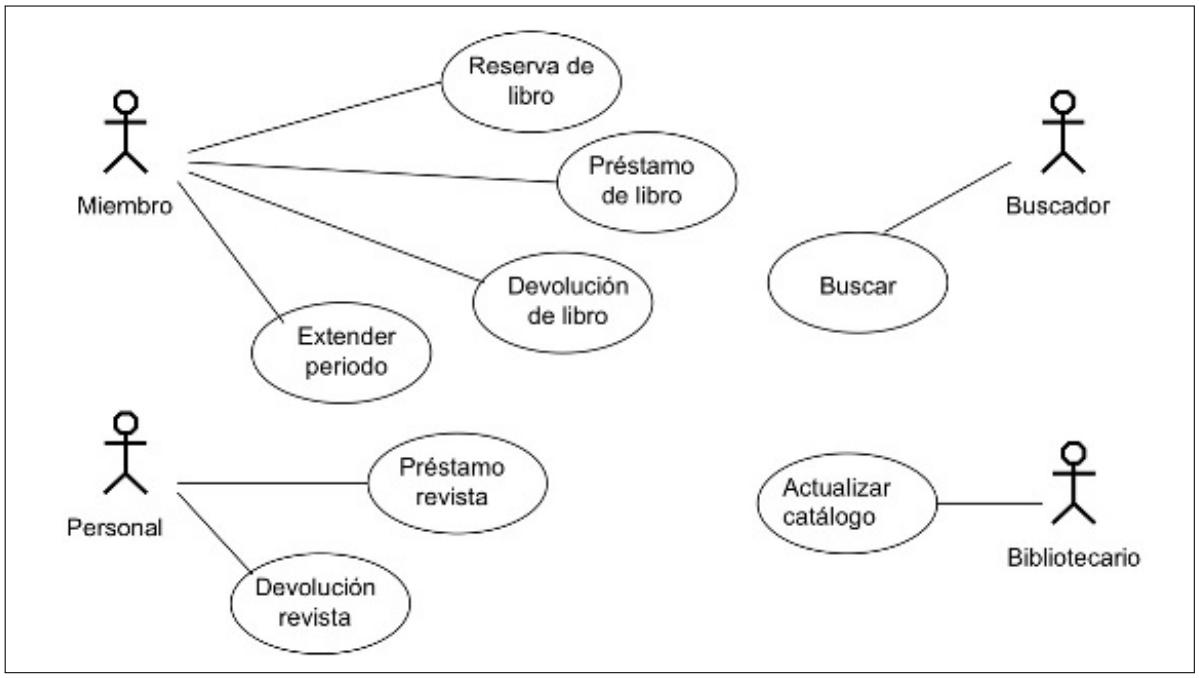


Figura 3.7: Diagrama de casos de uso [21]

Historias de usuario

Técnica muy ligada a las metodologías ágiles [3] (comienza con eXtreme Programming). Ésta técnica consiste en recoger las peticiones del cliente en forma de breves historias. Las historias deben ser comprensibles y estar bien delimitado lo que se quiere. Es común que estas historias tengan una misma estructura predeterminada, una de las estructuras más comunes para estas historias es : como, quiero, para (en inglés *as a, I want, so that*). Siguiendo esta estructura el usuario define quién es, qué quiere y por qué lo quiere. Ejemplo:

- Como: Granjero.
- Quiero: Una vaca lechera.
- Para: Tener leche merengada.

Además de esta descripción, la historia tendrá un nombre mediante el que hacerle referencia y podrá contener además un valor (para el negocio), una prioridad, un riesgo, una fecha o un número para organizarla.

3.5 Pruebas

El segundo elemento fundamental para obtener una buena adecuación funcional es un sistema de pruebas, este sistema de pruebas aportará la capacidad de garantizar que el sistema funciona y que lo hace tal y como se necesita. El estándar existente para el ciclo de pruebas es el estándar ISO/IEC 29119 [14]. Los tipos de pruebas más comunes son: Pruebas unitarias, pruebas de integración y pruebas funcionales.

3.5.1 ISO 29119

Independientemente del objetivo de las pruebas, éstas emplearán una serie de procesos que las dirijan, gestionen e implementen. Estos procesos están definidos en el estándar ISO/IEC 29119.

El objetivo del estándarISO/IEC 29119 es proporcionar una norma definitiva para las pruebas de software, sin dependencia del ciclo de vida de desarrollo usado. Este estándar define el vocabulario, procesos, documentación, técnicas y un modelo de evaluación del proceso de pruebas de software.

Para el presente TFG se usará principalmente el estándar ISO / IEC 29119-2. La finalidad de esta parte de el estándar es definir un modelo de procesos genérico que se ajuste a cualquier ciclo de vida de desarrollo software. Los procesos y métodos expuestos en el estándar serán aplicados a los diferentes tipos de pruebas usadas en la empresa.

3.5.2 Prueba unitaria

El objetivo de estas pruebas es comprobar el funcionamiento de cada «operación no trivial» o método del módulo. Estas pruebas serán individuales y aisladas para cada método.

3.5.3 Pruebas de integración

Estas pruebas aseguran que los diferentes métodos del sistema funcionan correctamente juntos. Si se prueba solo una parte delimitada del sistema se denominan pruebas de subsistema. Madrija utiliza un sistema especial de pruebas de integración que denomina «Pruebas de comandos».

Las «Pruebas de comandos» usan el framework basado en el patrón comando [28] de Madrija para testear la lógica de negocio de forma relativamente rápida, precisa y sin incluir la capa de presentación. Estas pruebas dan un poco más de profundidad que las pruebas de integración habituales pero sin llegar a s pruebas funcionales.

3.5.4 Prueba funcional

Una prueba funcional es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. Las pruebas funcionales se hacen, normalmente mediante el diseño de modelos de prueba, estos modelos de prueba buscan evaluar cada uno de los flujos de ejecución de la aplicación o sistema. Éstas pruebas son: específicas, concretas y exhaustivas con el objetivo de probar, verificar y validar que el software hace lo que debe y sobre todo, que hace lo que se ha especificado.

3.5.5 Prueba de aceptación

Uno de los tipos de pruebas funcionales. Las pruebas de aceptación son pruebas cuyo objetivo es acreditar que el programa cumple las expectativas del cliente. Estas expectativas serán, entre otras, las reflejadas en los requisitos y en el contrato.

Las pruebas de aceptación normalmente son ejecutadas por el cliente o por una muestra de los usuarios finales.

3.5.6 Desarrollo guiado por comportamiento

A partir de Desarrollo guiado por tests/pruebas (TDD) surge, como una evolución del mismo, el llamado Busines driven developement (BDD) [20]. BDD es un método de desarrollo software, éste consiste en definir el comportamiento que deberá seguir el software, a partir de la generación de unas pruebas que certifiquen que el comportamiento es correcto y a partir de éste momento, y no antes, generar el código de la aplicación el cual deberá pasar la prueba. Una vez la prueba es superada, el software se considera terminado (Siempre habrá lugar para mejoras).

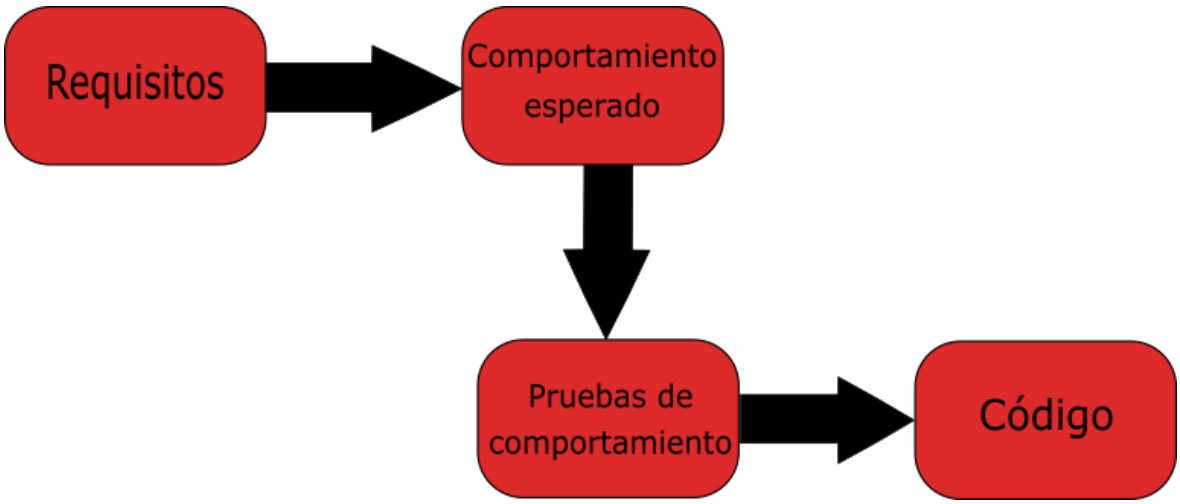


Figura 3.8: Bussiness driven development

Existen también otras técnicas denominadas Desarrollo Dirigido por Test de Aceptación (ATDD), y Story Test-Driven Development (STDD), ambas son variaciones de BDD e igualmente, tienen como base TDD.

Capítulo 4

Método de trabajo

COMO método de trabajo se ha escogido IDEAL En la primera parte de este capítulo se explicará cómo se ha seguido el método, a continuación se explicará que se planea hacer en cada una se las iteraciones previstas. Y por último se explicará que medios software y hardware se han usado.

4.1 Metodología de trabajo

Para la elaboración del TFG se ha decidido aplicar la metodología de trabajo basada en heurísticas IDEAL [1]. Éstas heurísticas serán iterativas e incrementales. Este método de trabajo se ha escogido debido a que el trabajo se desarrolla de forma individual, en parte se llevarán a cabo tareas de investigación y análisis de problemas y se tendrá muy en cuenta la opinión de la empresa en las decisiones tomadas, pues gran parte del trabajo generado será usado por ellos.

El método de trabajo IDEAL incluye cinco pasos: **I**dentificar el problema; **D**efinir y presentar el problema; **E**xplorar las estrategias viables; **A**vanzar en las estrategias; y **L**ograr la solución.

1. **Identificar el problema.**En este primer paso el objetivo será analizar la situación actual del problema, la primera vez se analiza la situación inicial. El objetivo de este paso es detectar el problema actual a solucionar

2. **Definir y presentar el problema.** Conocido el problema lo que hay que conseguir es describirlo con todo detalle posible y aportando todos los datos disponibles. Cuantos más datos se aporten mejor pues de esta manera se podrán plantear en el siguiente paso más estrategias posibles y al mismo tiempo estrategias más adecuadas. Se expondrá el problema, los datos de los que se disponen del entorno donde aplica y posibles dificultades que se prevean encontrar durante la ejecución de la iteración.

3. **Explorar las estrategias viables.** Una vez obtenida una visión sobre cuál es el problema y analizando todos los datos se deberán proponer y evaluar distintas soluciones posibles al problema planteado. Análisis de diferentes herramientas, métodos, estándares, tecnologías, etc. que sean relevantes.

4. **Avanzar en la estrategia (Actuar).** El punto crítico a pesar de que todos son muy importantes, escoger una o una combinación de varias de las soluciones y llevarla a cabo sobre el problema. Se elegirá una de las propuestas previas y una vez elegida se aplicará y se registrarán los resultados de los usuarios del sistema . Tras cada iteración se registrarán los resultados de la estrategia ejecutada.

5. **Lograr la solución.** En realidad es una traducción algo incorrecta de "look at the solution effects" , en este último paso lo que se hará es evaluar si la solución propuesta al problema es la correcta o hay que volver a algún estado anterior del proceso. Tras esta fase, se regresará a la primera a menos que sea la última iteración, en dicho caso se dará por concluido el proyecto.

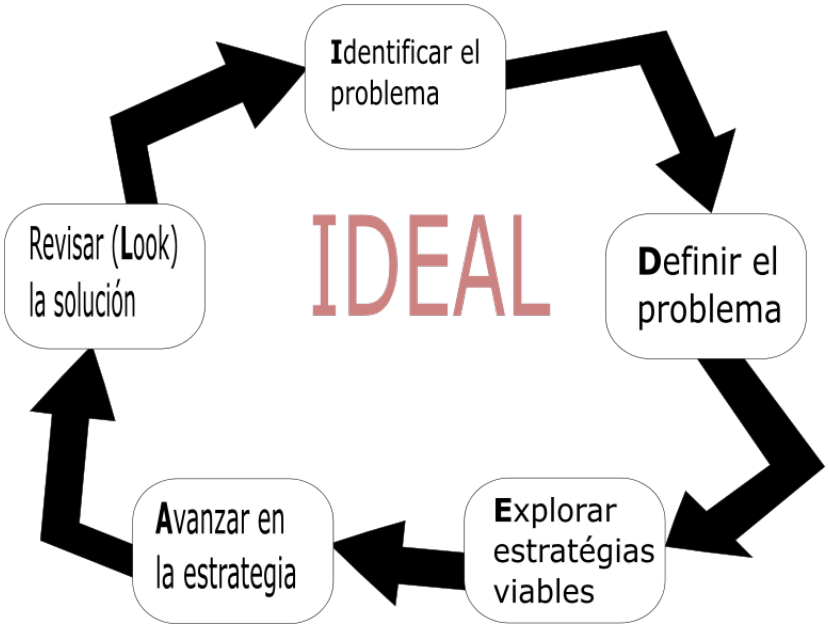


Figura 4.1: Diagrama IDEAL

Aquí se debe hacer una aclaración para no malinterpretar el esquema y es que el esquema muestra el orden habitual en el que se ejecuta este método de trabajo. Sin embargo, no es un sistema inflexible en el sentido de que se puede regresar a una etapa anterior si quien está ejecutándolo percibe que ha llegado a un callejón sin salida o, por ejemplo se le ocurre una estrategia adicional mientras está en la fase de avanzar en la estrategia.

4.2 Aplicación del método de trabajo

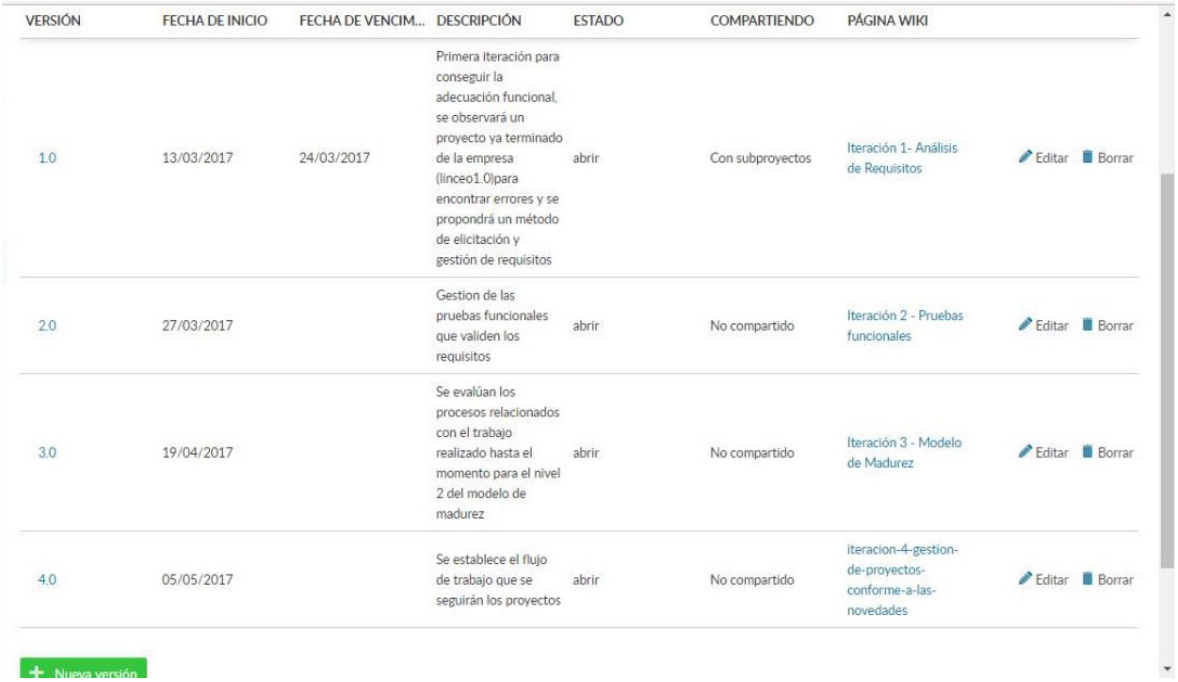
Las iteraciones y fases de cada una de las iteraciones se van a gestionar con la herramienta software OpenProject. Se creará un proyecto dentro del OpenProject corporativo de la organización para la gestión de este TFG. Para cada iteración se creará una versión y la planificación de la ejecución de las distintas fases se hará mediante tareas asociadas a dicha versión.

1. **Identificar el problema.**

   En el caso a tratar se observará, de las fases de desarrollo, cuales se van a modificar para asegurar la adecuación funcional y se decidirá cuál será el proceso modificado en la iteración a ejecutar. Se documentarán en la wiki del proyecto OpenProject los fallos encontrados en el proceso estudiado. Una vez identificado se registrará una nueva versión para la iteración que da comienzo con esta fase.

2. **Definir y presentar el problema.**

   Una vez escogido el proceso a modificar, se expondrá el objetivo de dicho proceso en la respectiva wiki en OpenProject, los datos de los que se disponen del entorno donde aplica, posibles dificultades que se prevean encontrar durante la ejecución de la iteración, etc.



VERSIÓN	FECHA DE INICIO	FECHA DE VENCIM...	DESCRIPCIÓN	ESTADO	COMPARTIENDO	PÁGINA WIKI	
1.0	13/03/2017	24/03/2017	Primera iteración para conseguir la adecuación funcional, se observará un proyecto ya terminado de la empresa (linceo1.0)para encontrar errores y se propondrá un método de elicitación y gestión de requisitos	abrir	Con subproyectos	Iteración 1- Análisis de Requisitos	Editar Borrar
2.0	27/03/2017		Gestion de las pruebas funcionales que validen los requisitos	abrir	No compartido	Iteración 2 - Pruebas funcionales	Editar Borrar
3.0	19/04/2017		Se evalúan los procesos relacionados con el trabajo realizado hasta el momento para el nivel 2 del modelo de madurez	abrir	No compartido	Iteración 3 - Modelo de Madurez	Editar Borrar
4.0	05/05/2017		Se establece el flujo de trabajo que se seguirán los proyectos	abrir	No compartido	iteracion-4-gestion-de-proyectos-conforme-a-las-novedades	Editar Borrar

+ Nueva versión

Figura 4.2: Pantalla versiones Open Project

3. **Explorar las estrategias viables.**

   Conforme al análisis del paso anterior se elaborará un plan de acción para llevar a cabo. De éste plan se generarán varias acciones o pasos. Las acciones derivadas del plan elaborado serán traducidas a tareas asociadas a la versión del proyecto en OpenProject.

4. **Avanzar en la estrategia (Actuar).**

   Una vez establecido el plan y creadas las tareas correspondientes se llevarán a cabo estas tareas documentando los avances en la wiki de OpenProject asociada a la misma.

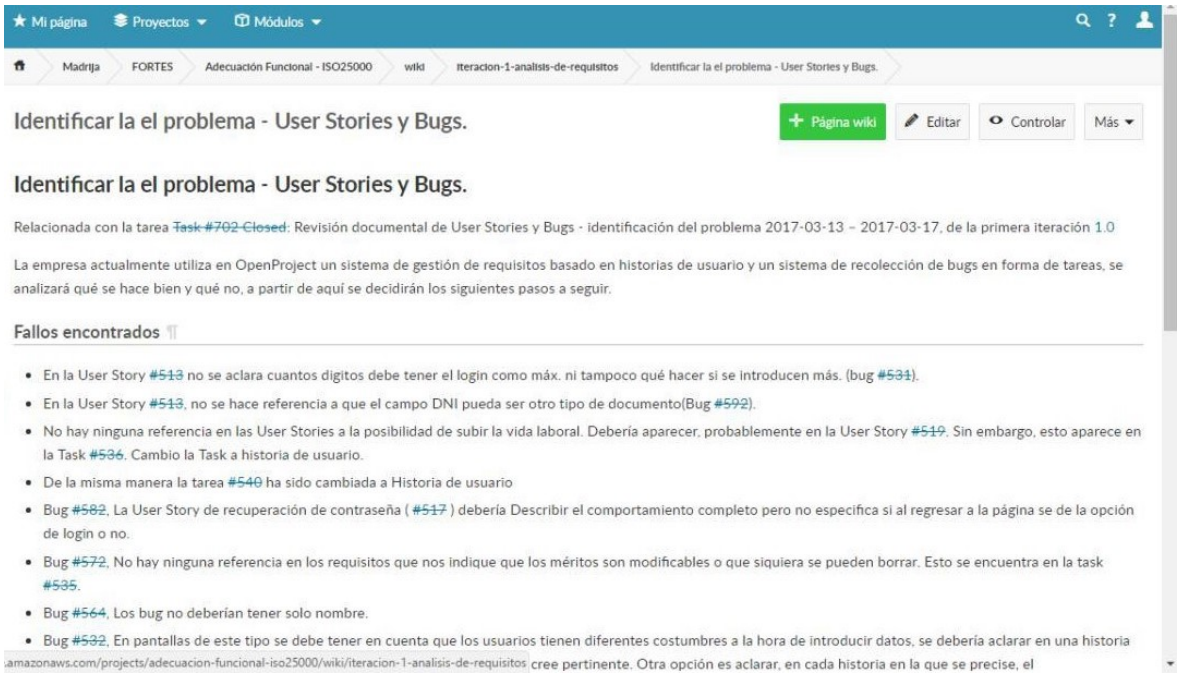


Figura 4.3: Ejemplo wiki Open Project

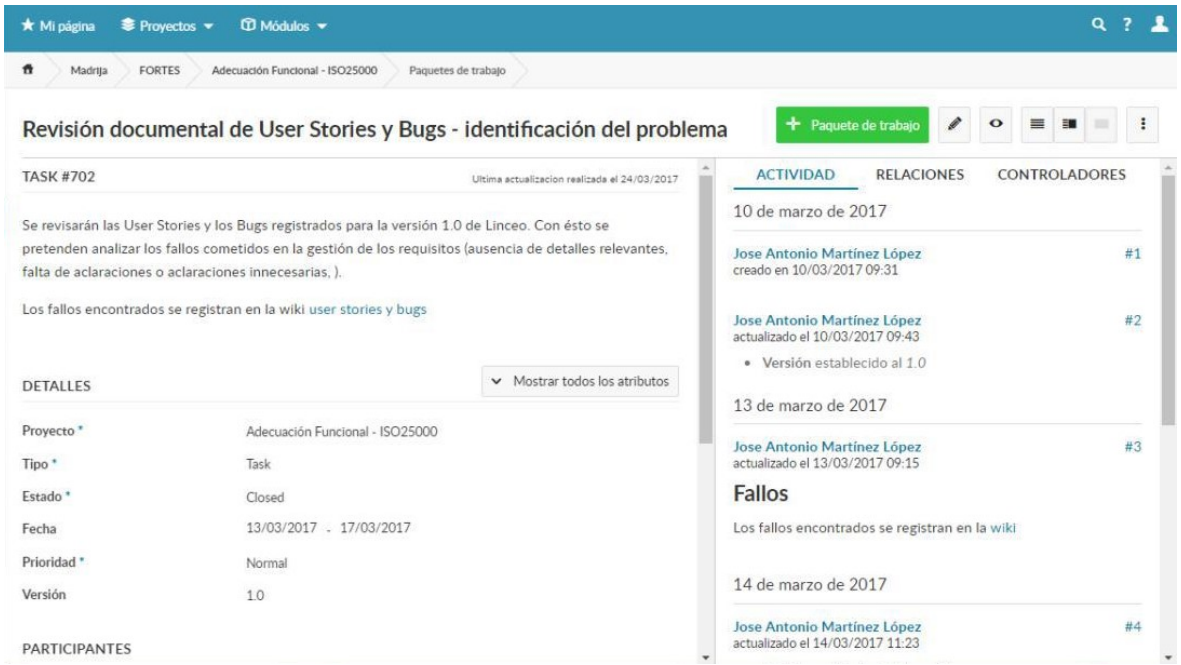


Figura 4.4: Ejemplo tarea Open Project

5. **Lograr la solución.**

   Se comprobará si el plan llevado a cabo es el correcto y si la solución es la que necesitábamos o si hay que retomar algún punto previo, modificar algo, evaluar otra propuesta, o avanzar a otra iteración y mejorar otra parte del desarrollo. Ésta fase cerrará las tareas creadas en OpenProject una vez se considere que la iteración ha concluido.

4.3 Planificación de iteraciones

Se desarrollan unas líneas base para determinar qué se pretende obtener tras finalizar cada iteración. El desarrollo de estas iteraciones está sujeto a cambios debido a imprevistos.

4.3.1 Iteración 1

El objetivo para esta iteración será encontrar un sistema de adquisición y gestión de requisitos que altere lo menos posible la forma de trabajar de la empresa pero que facilite conocer las necesidades de los *stakeholders*, lo que actualmente es algo difuso.

4.3.2 Iteración 2

El objetivo de esta iteración será crear un sistema de pruebas que verifiquen que los requisitos se cumplen. Se buscará que este sistema de pruebas sea sencillo de gestionar y usar.

4.3.3 Iteración 3

El objetivo en esta iteración será evaluar el trabajo ya realizado contra la norma ISO 15504 (SPICE) para saber qué elementos ayudan a conseguir el nivel de madurez 2, cuales se podrían añadir y cuales se añadirán a continuación.

4.3.4 Iteración 4

Se evaluará el impacto de las novedades sobre el flujo de trabajo de la empresa y se desarrollarán, si es necesario, unas guías para el nuevo flujo de trabajo, el cual se intentará no varíe mucho del actual.

4.3.5 Iteración 5

Se buscará una herramienta de gestión de pruebas.

4.3.6 Iteración 6

Se buscará un sistema de automatización de las pruebas.

4.4 Marco tecnológico

4.4.1 Herramientas software

4.4.2 OpenProject

Para la gestión del esfuerzo y los tiempos así como para la planificación se ha usado la herramienta software OpenProject, esta herramienta es la que usa Madrija, se ha usado una cuenta con acceso al servidor de la empresa.

Capítulo 5

Resultados

PARA exponer los resultados del trabajo realizado se dividirá este apartado, por una parte se explicará, de cada iteración, los detalles más importantes y las eventualidades surgidas en la ejecución de las mismas, por otra parte se expondrán los recursos generados para la mejora de los procesos (nuevo flujo de trabajo, plantillas para los requisitos, checklists, resultados del nivel de madurez...)

5.1 Recursos generados

Como resultado de todo el trabajo realizado durante las iteraciones se consiguen una serie de recursos. Cada recurso será explicado a fondo más adelante. Estos recursos son:

- Un guión de flujo de trabajo que ayudará a incorporar las novedades al proceso del ciclo de vida software de la empresa,
- Una serie de plantillas para requisitos y unas checklists que comprueben si están correctamente redactado.
- Una serie de plantillas para pruebas y unas checklists que comprueben si están correctamente redactadas.
- El documento Definition of Ready (DoR)
- El documento Definition of Done (DoD).
- Tabla resumen con las evidencias para el modelo de madurez.

5.1.1 Flujo de trabajo

Debido a las novedades introducidas en los procesos del ciclo de vida y para facilitar su correcto uso, se establece un flujo de trabajo estándar que funcionará en gran medida para la mayoría de situaciones. Este nuevo flujo de trabajo no será inquebrantable, la empresa se dirige por metodologías ágiles y eso significa que se adaptará en la medida de lo posible a las limitaciones del cliente y de cada proyecto.

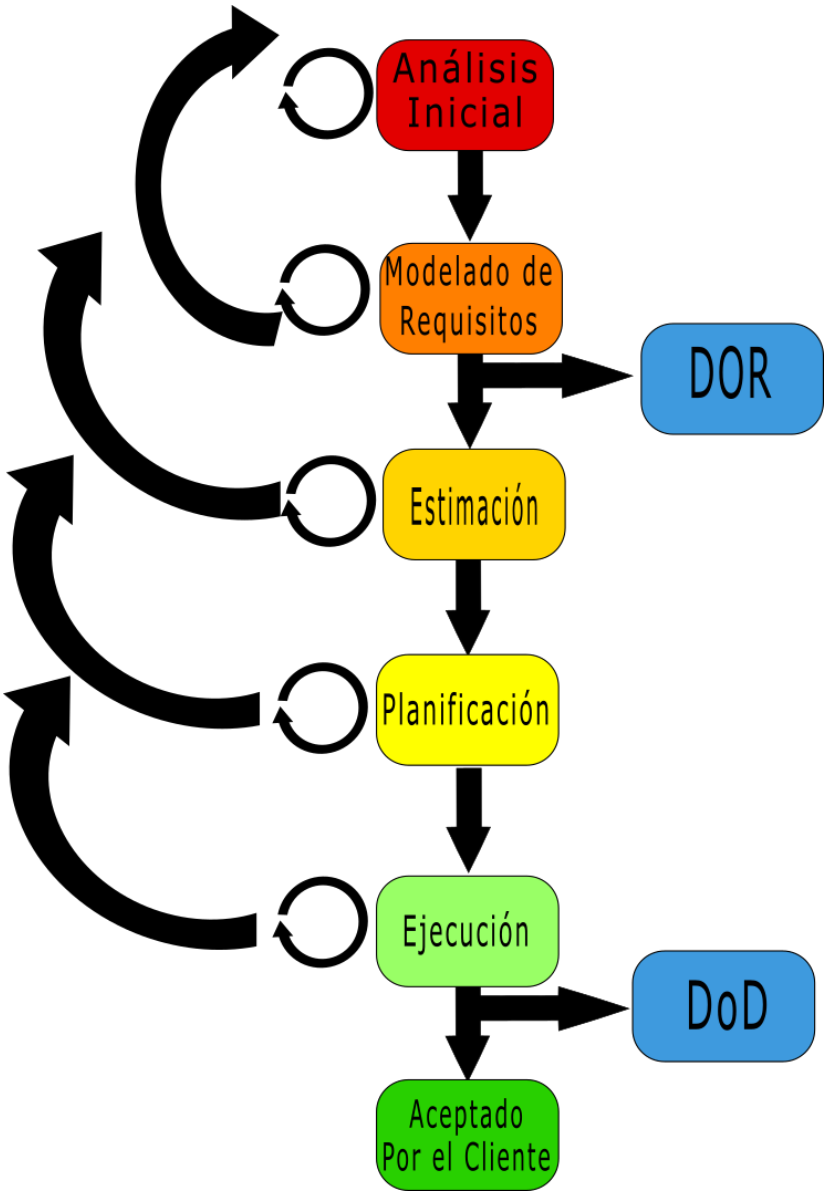


Figura 5.1: Esquema de flujo de trabajo

Análisis inicial

Todo proyecto tiene un comienzo, en este caso tras un contacto inicial y las fases previas como la incepción del proyecto, necesitaremos una base de documentos que nos indiquen que nos pide el cliente/*stakeholder*, estos documentos serán:

Actas de Reuniones. Tras una reunión con un cliente se obtendrá un documento donde el encargado de la reunión escribirá todo lo que sea capaz centrándose en lo más importante. Las reuniones se llevarán a cabo preferiblemente dentro del entorno de técnicas grupales como pueden ser JAD Brainstormings o método KJ.

Documentos entregados por el cliente. Ya sea la propuesta inicial u otro tipo de documentos que profundicen en la especificación de requisitos.

Puede ser que se obtengan ambos tipos o que solo se disponga de un solo tipo de documentos. En cualquiera de los casos, tras recoger estos documentos que reflejaran las peticiones de los clientes/*stakeholders* se procederá a crear el conjunto de **necesidades del cliente**

Acuerdo con el cliente

Se redactarán las **necesidades del cliente** (usando la plantilla desarrollada en este trabajo) conforme a lo que se entiende quiere el *stakeholder*. Tras esto, se presentarán al *stakeholder* para que las valide o muestre sus desacuerdos.

Una vez el cliente valide todas las necesidades de cliente (NDC)

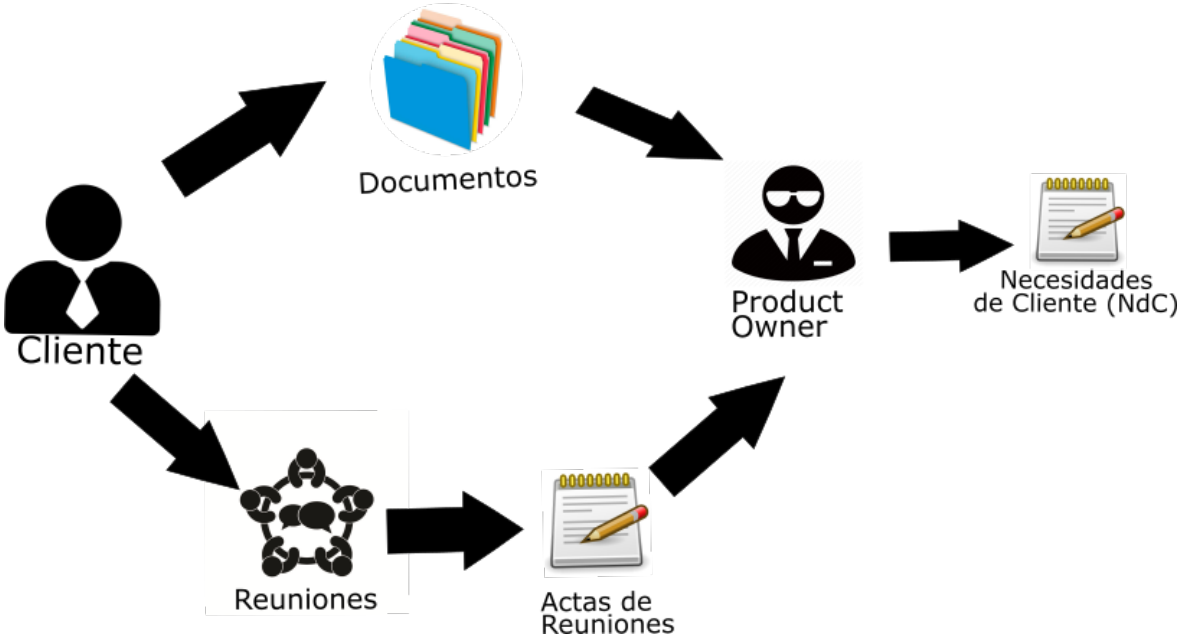


Figura 5.2: Captura de Requisitos

Modelado de Requisitos

Una vez obtenemos las Necesidades del cliente y tras ser validadas, el Product Owner (PO) las transformará en los documentos que se usarán como requisitos, estos son: **HDU**, **entidades** y **Casos de prueba (CDP)**

Las Entidades y las HDU se obtendrán de forma más o menos simultanea. Los CDP se obtendrán a posteriori, pues se basarán en los criterios de aceptación y en los escenarios de las HDU.

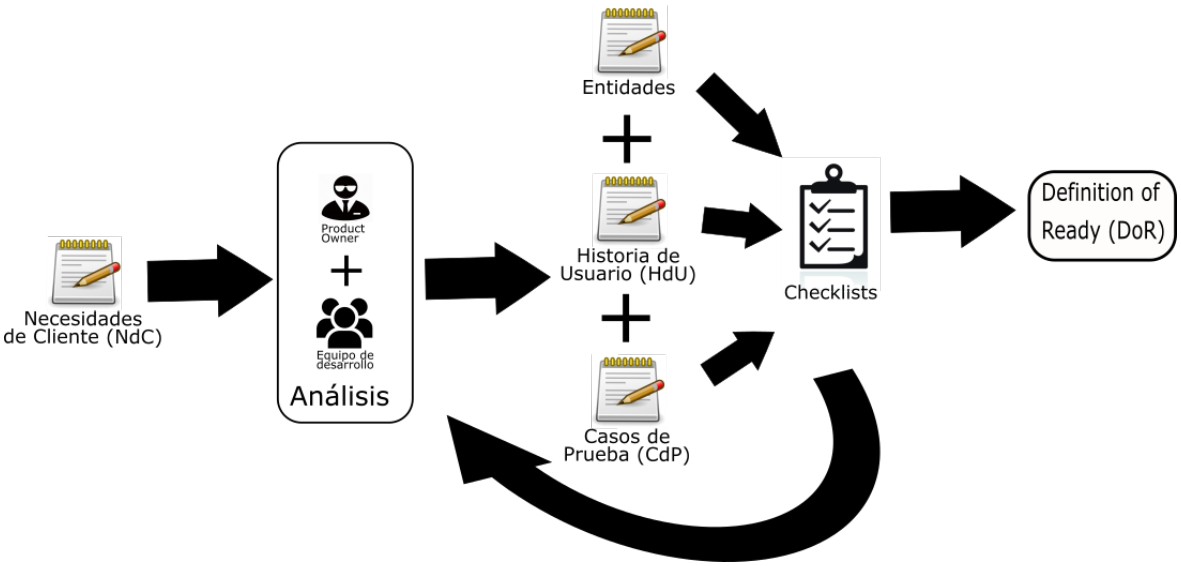


Figura 5.3: Modelado de Historias de Usuario

DoR

Tras terminar las descripciones de los requisitos el PO trasladará las responsabilidad al equipo de desarrollo, el equipo de desarrollo se encargará, usando las checklists proporcionadas, de comprobar que están redactadas tal y como se necesitan, además de esto se aplicará un checklist de DoR, proporcionado más adelante.

Estimación, planificación y ejecución

Una vez Superado el DoR se estimará el tiempo y el esfuerzo que supondrá llevar a cabo la implementación de las historias de usuario. Obtenido el coste de cada HDU se planificará la ejecución de las mismas usando para ello la estimación realizada previamente y dividiendo las historias en tareas. Después de planificar la ejecución se comenzará la fase de codificación del proyecto.

DoD

Una vez se considere finalizada una tarea por parte del equipo de desarrollo, se procederá a aplicar la checklist de DoD, se puede ver más adelante . Si el DoD es superado, los resultados de la ejecución de la tarea se podrán presentar al cliente. Por lo general el DoD solicitará superar uno o varios CdP

5.1.2 Plantilla para necesidades del cliente

Se valoran dos vías de información para establecer la línea base de necesidades del cliente/*stakeholder*. Estas dos vías son: documentos proporcionados por el cliente y actas de reuniones con el cliente.

Sea a través de una o con la unión de ambas vías, esta información será estructurada

en un proceso posterior generando unos documentos denominados necesidades de cliente (NDC).

La estructura para estos documentos es la siguiente:

ID: Otorgado por OpenProject, servirá para hacer referencia a esta tarea dentro de toda la documentación.

Nombre: Nombre representativo de la necesidad, se debe buscar simpleza y concreción sin perder significado irá acompañado de la "etiqueta" [NECESIDAD]

Texto libre: Se escribirá aquí la necesidad del cliente de la forma más detallada posible. A partir de este texto surgirán las HDU que determinarán el comportamiento del sistema a desarrollar. Todo aquello que se considere relevante para la posterior realización de las HDU deberá escribirse aquí. El uso del formato Dado-Cuando-entonces (*Given-When-Then*) facilitará la redacción y comprensión de estas descripciones, así como su posterior transformación a HDU.

Condición/es de aceptación: Una primera versión de los criterios de aceptación. Nos dará información no detallada sobre que necesitamos hacer para cumplir con lo que solicita el cliente.

Relaciones: Cada necesidad de cliente tendrá subordinadas una serie de HDU. El número de HDU relacionadas en indefinido. Se usarán los campos de relación de Open Project.

El documento de Necesidades de cliente servirá de primer acuerdo con el cliente. De esta manera se establecerá que se hará y como determinar que está hecho.

Ejemplo

En la figura 5.4 observamos un ejemplo de historia de Necesidad de cliente creada en la Aplicación OpenProject, en la zona izquierda está la navegación por el proyecto, en el centro está la información de la NDC y a la derecha se ven las relaciones, concretamente con las HDU, además en la pestaña «Actividad» se pueden ver los últimos cambios realizados en el paquete de trabajo. El uso de la herramienta se verá en más profundidad en el anexo B.

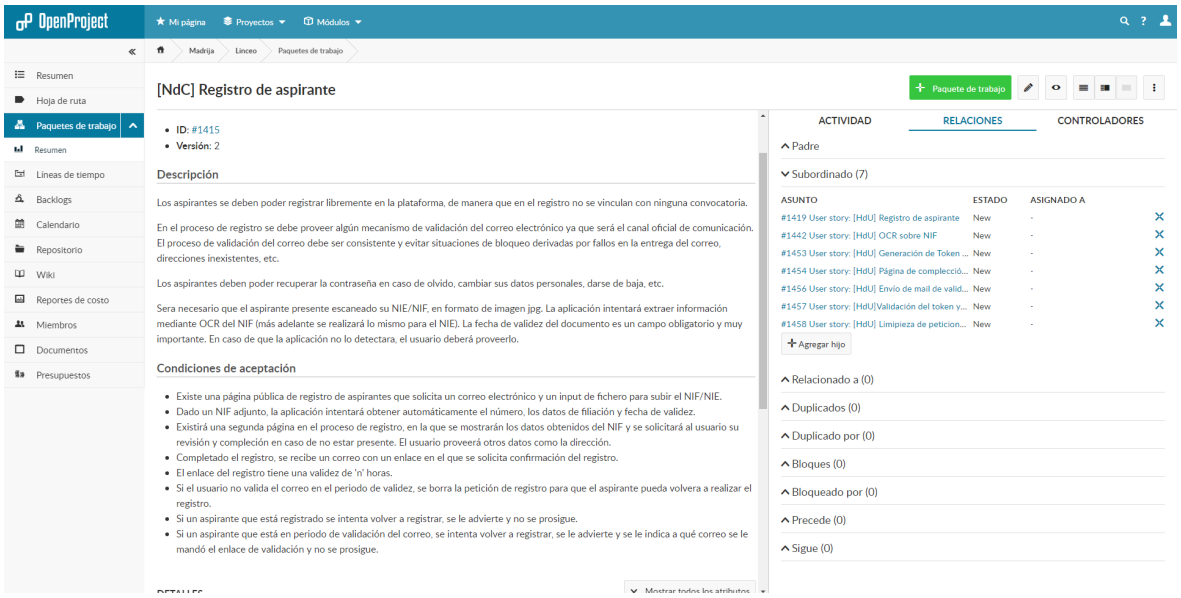


Figura 5.4: Ejemplo de Necesidad de cliente en la herramienta OpenProject

5.1.3 Plantilla de requisitos de Entidad.

Una entidad será cualquier porción de conocimiento que queramos obtener o almacenar(normalmente representada como un objeto), no es necesario que este conocimiento esté o tenga que guardarse en persistencia.

Con esta plantilla se deberán generar todos los requisitos del tipo Entidad. En OpenProject será del tipo Feature y su nombre comenzará con [Entidad]. Deberán tener un nombre lo más concreto y lo más identificativo posible y un código (generado por el propio OpenProject).

La estructura propuesta para ellas es la siguiente, mediante campos a rellenar:

ID: Será un identificador único que proporcionará la herramienta OpenProject, con la que se gestionan los requisitos. Este servirá para relacionar la HDU con el resto de tareas o requisitos.

Nombre: Un nombre identificativo y lo más conciso posible.

Descripción: En esta parte se deberá explicar el objetivo de crear y mantener esta entidad. Deberá quedar claro cuál es la información que manejará.

Atributos: Cada entidad tendrá una lista de atributos que representarán la información que usará la entidad. Cada uno de estos atributos tendrá:

- Un **Nombre**. Nombre del campo que representa al atributo, se deberá buscar un nombre distintivo.
- Un **tipo** de dato usado para representar el atributo.(entero, texto, colección, lista ...)
- Una **Longitud/tamaño**. Especificación de la longitud o configuración del tipo de dato.
- Unas posibles **Restricciones**. Especificación de restricciones aplicables al valor/uso del atributo.

Guarda en persistencia: Existirán entidades que deban tener una correspondencia con la base de datos y otras que no lo necesiten, es importante saber cuál es cada una.

Reglas de integridad: En caso de tratar entidades complejas, que requieran de algún tipo de validación a nivel global, es decir, en las que varios de los atributos deban cumplir una serie de reglas, estas reglas serán definidas aquí, de manera objetiva y representable mediante un algoritmo. Se acompañarán CDP objetivos, tanto positivos como negativos.

5.1.4 Plantilla de historias de usuario

Para la gestión de las HDU se ha desarrollado una plantilla que sirva de modelo para la elaboración de las mismas Los campos de la plantilla se explican a continuación:

ID: Será un identificador único que proporcionará la herramienta OpenProject, con la que se gestionan los requisitos. Este servirá para relacionar la HDU con el resto de elementos de Open Project como pueden ser tareas o requisitos.

Nombre: Un nombre representativo y conciso, que ofrezca una idea inicial de en qué consiste la HDU.

Versión: Independiente para cada requisito, este campo será siempre 1.0 en la creación del requisito, si el requisito es cambiado por cualquier motivo, este número cambiará y el cambio realizado se deberá registrar. De momento este registro va a ser manual, pero se planteará más adelante la posibilidad de crear un plugin para OpenProject que lo automatice.

Prioridad: Se usará la escala por defecto de OpenProject, esta escala es baja, normal, alta o inmediata. Indicará la urgencia de la HDU.

Rol: El rol deberá definir la persona que necesita esta función, no necesariamente tiene que pedirlo ella. El rol no deberá ser (por lo general) el desarrollador no deberá ser el PO. Deberá ser directo y corto pero autodefinirse bien, no será lo mismo un rol “usuario” que uno

"usuario registrado". Se deberán tener muy en cuenta determinados matices. Como aclaración, el rol no tiene porqué ser el usuario final de la aplicación, puede ser que el rol sea "jefe de Recursos Humanos (RRHH)", y que por ejemplo quiera una función en la que ver los expedientes de toda la plantilla. Esto es algo habitual al definir HDU, pero debido a que este documento será usado para instruir a los usuarios de la plantilla, se hacen este tipo de puntualizaciones. La oración se comenzará con la palabra "Como"

Ejemplo: **"Como jefe de RRHH"**

Petición: En este apartado se deberá describir la funcionalidad que se quiere lograr o lo que se quiere poder hacer con nuestro software. La petición puede englobar muchos aspectos, no tiene porqué implicar una acción directa con el sistema. Por ejemplo, "Quiero que la interfaz de la página web sea amigable y sencilla". La oración se comenzará con la palabra "Quiero"

Ejemplo: "Como jefe de RRHH"

"quiero una pestaña en mi página con los datos de mis empleados".

Propósito/beneficio: Este puede ser el campo puede ser más complicado de rellenar porque a veces es algo ambiguo, pero es importante porque a menudo da un mejor contexto, amplía la visión que tenemos de la funcionalidad o aporta matices importantes. Es posible que en ocasiones la justificación sea obvia o incluso algo redundante, pero se debe hacer un esfuerzo para completarla. La oración se comenzará con la palabra "Para".

Ejemplo: "Como jefe de RRHH"$\rightarrow$ "quiero una pestaña en mi página con los datos de mis empleados".

"para ver toda la información disponible y saber si necesitamos más personal."

Registros para auditoría: Algunas funcionalidades que, por ejemplo traten información sensible, deberán registrar ciertos datos como quien la ejecuta y que información trata. Todo esto debe reflejarse en este apartado.

Ejemplo: **"Se deberá registrar quien ha modificado los datos"**

Permisos para los datos: En esta sección se detallan qué derechos tiene cada usuario sobre los datos que se traten en el requisito si es que se trata alguno.

Ejemplo: **"Estos datos podrán verse, no podrán ni modificarse ni borrarse."**

Escenario A pesar de que los escenarios no son un campo obligatorio, en caso de que la HDU describa una funcionalidad que no sea puramente gráfica (donde haya trabajo en lógica

de dominio, por ejemplo) será obligado que exista un caso de escenario satisfactorio y otro insatisfactorio al menos.

Cada escenario representará un posible flujo de ejecución dentro de la HDU. Posteriormente cada escenario generará una prueba de aceptación al igual que los criterios de aceptación. o más bien cada escenario será verificado con un criterio de aceptación.

Para facilitar la creación de las posteriores pruebas de aceptación usaremos para escribir los escenarios la estructura [Dado, Cuando, Entonces] o en inglés [*Given-When- Then*] que se explica a continuación:

Etiqueta	**Explicación**	**Ejemplo**
Estado inicial (Dado)	Aquí se expondrán las condiciones iniciales bajo las que se ejecutará la acción. Puede haber varias condiciones iniciales, si es así se escriben todas lo más claro posible.	Dado un usuario registrado y El usuario está validado.
Acciónes realizadas (Cuando)	En este punto se describirá la acción que se quiere poner a prueba, dependiendo de cada caso y de nuestro objetivo, los datos podrán ser más o menos concretos.	Introduce su login y contraseña correctos y pulsa enter en el teclado o el botón “acceder” de la pantalla.
Estado final (Entonces)	Aquí describiremos el resultado que esperamos de la ejecución de la acción previa al igual que en el estado inicial, puede haber varios resultados, se expone cada uno de ellos.	Se muestra la pantalla inicial de la aplicación y queda registrado en un log qué usuario a accedido y la hora a la que lo ha hecho.

Mockup: En caso de que la HDU describa el aspecto de una página se creará un esquema o mockup que se deberá almacenar en la tarea. El objetivo es trasladar la idea acordada con el cliente. Dentro de lo posible, este esquema se presentará al cliente y se corregirá siguiendo sus pautas. El *mockup* se podría generar con cualquier herramienta, incluso a mano, la empresa ha decidido usar Pencil [5], una herramienta Opensource gratuita muy simple que permite hacer bocetos de pantallas rápidamente.

Condiciones de aceptación: Deberá aparecer el nombre de la condición y un enlace a la tarea que contiene sus detalles (y que será del tipo CDP).

Estas condiciones de aceptación serán la principal herramienta para determinar que las HDU se han completado satisfactoriamente. Algunas de estas condiciones de aceptación derivarán de las condiciones de aceptación descritas en las *Necesidades de cliente*.

Teniendo la siguiente HDU.

"Como jefe de RRHH" →"quiero una pestaña en mi página con los datos de mis empleados".→"para ver toda la información posible y saber si necesitamos más personal."

- "La información de esta página será la de la Entidad #23 (ID de la entidad en Open Project) "
- "Estos datos podrá verlos solo el personal autorizado" Aunque habla sobre los permisos de datos, también lo deberemos tener en cuenta a la hora de dar por finalizada la función.

Anexos Es posible que sea necesario mencionar algún anexo de la documentación en caso de que se haga alusión a algo externo al requisito pero que tiene influencia en él. Normalmente será información respectiva al negocio. Por ejemplo, normativas internas, códigos o argot.

Datos de prueba. Este campo contendrá archivos o enlaces a los mismos, estos archivos serán los que se usarán para las prueba de funcionalidad, ya sean de forma automática o manual.

Coste h/p : En este campo se incluirá una cifra que representará las horas estimadas que costará llevar a cabo la HDU. Esta información se usará para establecer un calendario para el proyecto y un precio estimado al cliente.

Valor: En este campo se incluirá el valor para el negocio que supone llevar a cabo la HDU, este valor servirá para priorizar las tareas que se lleven a cabo. De momento, Madrija ha optado por aplazar la incorporación de este apartado debido a las consecuencias que supondría en el flujo de trabajo.

Ejemplo

En la figura 5.5 observamos un ejemplo de historia de usuario creada en la aplicación OpenProject, en la zona izquierda está la navegación por el proyecto, en el centro está la información de la NDC y a la derecha se ven las relaciones, concretamente con la NDC que le precede y con otras HdU relacionadas, además en la pestaña «Actividad» Se pueden ver los últimos cambios realizados en el paquete de trabajo. El uso de la herramienta se verá en más profundidad en el anexo B.

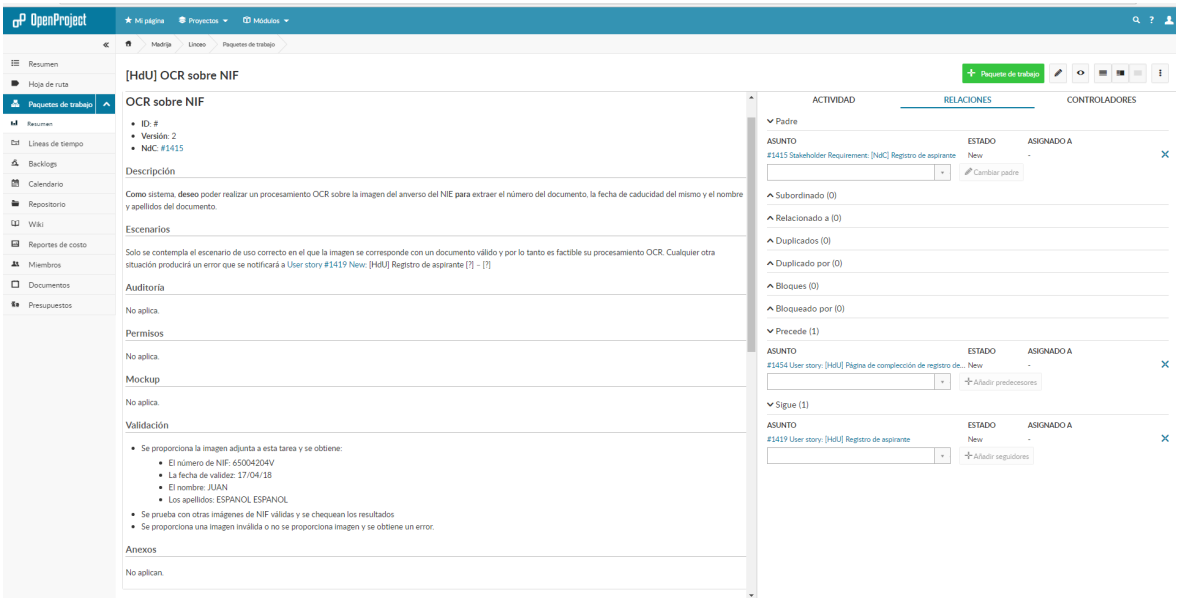


Figura 5.5: Ejemplo de Historia de usuario en la herramienta OpenProject

5.1.5 Plantilla casos de prueba

Los CDP son la herramienta que nos permitirá verificar que el sistema funciona, así como validar que hace lo que se solicita en los requisitos (**NDC e HDU**).

ID: Será un identificador único que proporcionará la herramienta OpenProject, con la que se gestionan los requisitos. Este servirá para relacionar el CDP con el resto de elementos de Open Project como pueden ser tareas o requisitos.

Nombre: Un nombre representativo y conciso, que de una idea inicial de en qué consiste la HDU. El campo nombre deberá contener la cabecera [Caso de prueba]. El nombre deberá representar el criterio de la forma más descriptiva posible.

Versión: El campo Versión registrará qué revisión del CDP estamos tratando, cuando se realice un cambio importante, bien en la descripción, bien en el guion de pruebas anexo, se usará un campo añadido en OpenProject llamado versión, este campo comenzará siendo igual a “1.0”.

Prioridad: El campo Prioridad podrá tener los valores [baja, normal, alta o inmediata], se gestionará con el campo predeterminado de OpenProject.

Descripción de la prueba: La prueba será descrita usando la un formato típico de SCRUM, el formato **Dado-Cuando-Entonces** (*Given-When-Then*).
Dado describirá la situación previa necesaria y las precondiciones (si hay).
Cuando describirá la acción o acciones realizadas.
Entonces describirá el estado final esperado.

Guion de pruebas: El campo guion de pruebas será un campo que precise de un archivo anexo con la descripción paso a paso de la Guion de pruebas/Prueba funcional que surgirá a partir del criterio de aceptación. Es posible que este anexo se encuentre en repositorio y aquí solamente haya un enlace al repositorio. Puede parecer que el campo *Descripción de la prueba* y este campo son duplicaciones o redundantes pero el destino de estos es distinto. *Descripción de la prueba* servirá para ver rápidamente que debe hacer el sistema en determinados casos. El guion de pruebas detallará los pasos que se ejecutarán para determinar que la prueba es satisfactoria y permitirá facilitar la automatización del CDP en caso de que sea necesario.

Requisito(s): En el campo Requisito(s) se relacionaran todos los requisitos que precisan del criterio de aceptación para ser validados, estos serán referenciados mediante su etiqueta Open Project, lo mínimo será que cada criterio tenga un requisito al que satisfacer. Lo más normal es que el requisito sea único.

Datos de prueba: Los datos de prueba contendrán las muestras que se usarán para llevar a cabo el CDP.

Resultado: Resultado reflejará, una vez se ha ejecutado al menos una vez la prueba, "éxito" o "fallo", se considerará éxito si al ejecutar los pasos del script el estado final es el esperado (Apartado "Entonces") y se considerará fallo en cualquier otro caso. Si el resultado es fallo, se deberá revisar tanto la prueba de aceptación como el código ejecutado, puede haber un error al reflejar el comportamiento de la funcionalidad o puede haber un fallo al representarla.

Ejemplo de transformación de criterio de aceptación a guion de pruebas.

La plantilla de los CDP seguirá la estructura [*Given-When-Then*]. Gracias a esto obtener el guion de prueba será relativamente sencillo. Ejemplo:

Criterio de aceptación

- Dado: un usuario registrado en el sistema.

- Cuando: introduzca sus datos y presione acceder.
- Entonces: Verá la página principal.

Guion de pruebas/Prueba funcional

1. Acceder a la página login.
2. Introducir nombre existente en el recuadro nombre.
3. Introducir contraseña correcta en el recuadro contraseña.
4. Pulsar botón “acceder”.
5. Comprobar que la pagina es la página principal (en las pruebas automáticas se puede hacer buscando algún trozo de texto característico).

Es importante dejar bien claro qué se hace en cada uno de los pasos para evitar equivocaciones al ejecutar la prueba manualmente o al crear el script para la prueba automática.

5.1.6 Checklist Entidades

Una vez la entidad sea redactada por el PO ésta será trasladada a una persona distinta, preferiblemente alguien que vaya a desarrollar la entidad. Esta persona, usando la checklist juzgará si la entidad está bien redactada, si no cumple algún punto, será devuelta al PO.

Pregunta	**Respuesta correcta**
¿Es del tipo feature?	Si
¿Su nombre tiene la estructura [Entidad] Nombre?	Si
¿La descripción es legible y deja claro que información estamos tratando?	Si
¿Cada uno de los atributos esta bien definido y su tipo es correcto?	Si
¿Incluye restricciones funcionales?	No

Cuadro 5.2: Checklist Entidades

5.1.7 Checklist Historias de usuario

Una vez la HDU sea redactada por el PO ésta será trasladada a una persona distinta, preferiblemente alguien que vaya a desarrollar la HDU. Esta persona, usando la checklist juzgará

si la HDU está bien redactada, si no cumple algún punto, será devuelta al PO.

Pregunta	**Respuesta correcta**
¿Es del tipo UserStory de Open Project?	Si
¿Tiene rol y se identifica bien quien es?	Si
¿Tiene petición y esta es clara y concisa?	Si
¿Tiene propósito y este deja claro el objetivo de la funcionalidad?	Si
Si no tiene registros para auditoria ¿Debería tenerlos?	No
Si no define permisos para los datos ¿Debería tenerlos?	No
¿Las condiciones de aceptación se entienden bien?	Si
¿Tiene menos de 5 escenarios?	Si
¿Si la funcionalidad no es exclusivamente de visualización, contiene al menos un escenario satisfactorio?	Si
¿Necesita un escenario insatisfactorio que no tiene?	No
¿Los escenarios tienen la estructura [Dado-Cuando-Entonces]?	Si
¿Los escenarios tienen asociado un CDP?	Si
¿Las condiciones de aceptación tienen asociado un CDP?	Si
Si tiene Entidades vinculadas ¿Pasan su checklist ?	Si
¿Los CDP vinculados pasan su checklist ?	Si

Cuadro 5.3: Checklist Historias de Usuario

5.1.8 Checklist Casos de prueba

Una vez el CDP sea redactado por el PO ésta será trasladado a una persona distinta, preferiblemente alguien que vaya a desarrollar el CDP. Esta persona, usando la checklist juzgará si el CDP está bien redactado, si no cumple algún punto, será devuelto al PO para ser revisado.

Pregunta	Respuesta correcta
¿Es del tipo task de Open Project?	Si
¿La tarea está relacionada con la HDU correspondiente en ambos sentidos?	Si
¿Su nombre tiene la estructura [Caso de prueba] Nombre?	Si
¿Refleja con precisión el comportamiento acordado con el cliente ?	Si
¿El objetivo de la condición de aceptación esta expresado con el formato [Dado-Cuando-Entonces]?	Si
¿Tiene Guión de prueba (ya sea manual o automático)?	Si
¿Dicho guión, describe de forma clara los pasos necesarios para pasar el CDP?	Si
¿Si precisa de datos de prueba los tiene?	Si
¿Si tiene Datos de prueba son los correctos?	Si

Cuadro 5.4: Checklist Casos de Prueba

5.1.9 Matriz de trazabilidad

La matriz de trazabilidad permitirá saber de una forma rápida qué tareas debemos hacer para completar el requisito.

Esta matriz de trazabilidad relacionará cada requisito con la o las pruebas necesarias para asegurar que su funcionamiento es el correcto. Dado que estas pruebas estarán relacionadas con los CDP, cada CDP deberá obligatoriamente estar relacionado (En Open Project) con la HDU que corresponda. Gracias a esta relación, Open Project nos proporcionará (no directamente) la capacidad de generar la matriz de una forma automática una vez se termine la documentación de los requisitos.

5.1.10 Ciclos de prueba.

La ejecución de las pruebas se llevará a cabo mediante ciclos de pruebas.

Cada ciclo tendrá como objetivo probar una funcionalidad (con lo que puede ser que coincida con una HDU) o varias funcionalidades relacionadas de algún modo. No todos los ciclos se podrán establecer desde el inicio de la planificación, de modo que conforme se avance en el desarrollo surgirán nuevos ciclos de pruebas.

Un ciclo podría ser, por ejemplo una prueba de regresión para una funcionalidad,esto serían la serie de pruebas que deben ejecutarse en caso de que una funcionalidad sea actualizada o cambiada, con el objetivo de que la propia función y aquellas relacionadas funcionen tal y como deben.

En cada ciclo se deberá determinar, qué pruebas se ejecutan y en que orden se harán. En nuestro sistema se ejecutarán las pruebas automáticas + las que estén relacionadas con la funcionalidad que estamos probando.

Los resultados de las ejecuciones de cada ciclo se podrán sintetizar en una Matriz como la siguiente.

5.1.11 Definition of Ready (DoR)

El DoR es el conjunto de señales que nos marcan cuando la HdU está lista para su fase de ejecución. El DoR es el trabajo que tiene que realizar el PO para que el equipo de desarrollo pueda comenzar a trabajar, es la exigencia del equipo de desarrollo al PO.

Para cada proyecto se necesitará generar un DoR específico pero hay unos indicadores básicos que compartirán los DoR de la mayoría de los proyectos. Estos indicadores son:

- Las HdU tienen asociadas las necesidades del cliente.
- Las HdU están redactados correctamente conforme a su checklist.
- Las HdU tienen los criterios de aceptación necesarios. Es decir, cada escenario tiene asociado un CdP que verifica su comportamiento y el apartado de criterios de aceptación contiene todos los necesarios y cada uno tiene un enlace al caso de prueba que lo verifica.
- Los casos de prueba están redactados correctamente conforme a su checklist.
- Las HdU tienen asociadas las entidades que usan y éstas superan su checklist.
- Las necesidades del cliente relacionadas con la HdU han sido validados por el cliente.
- Se ha estimado el coste de la HdU.

5.1.12 Definition of Done (DoD)

El DoD consiste en una serie de indicadores que nos marcan cuando hemos terminado una HdU y la podemos cerrar. El PO exige al equipo de desarrollo superar el DoD.

Para cada proyecto se necesitará generar un DoD específico pero hay unos indicadores básicos que compartirán los DoD de la mayoría de los proyectos. Estos indicadores son:

- Todas las tareas asociadas a la HdU se han completado.
- Se han documentado los datos sobre parametrización y configuración necesarios del software generado en la HdU.

- La configuración del changelog es multi-BBDD en caso de que la HDU tenga información en BBDD.
- Las pruebas unitarias, de comandos y de aceptación han sido superadas para el código generado por esa HDU.

5.1.13 Beneficios y relación con la adecuación funcional

Gracias a las HDU y las NDC, obtenidas tras el proceso de captura de requisitos, podremos saber que es lo que quiere el cliente, los escenarios de las HDU y las condiciones de aceptación junto con los casos de prueba darán más información y más detallada de cómo se deberá comportar el programa final. Esto ayudará a centrar los esfuerzos en qué debe hacer el programa, lo que unido a la matriz de trazabilidad nos dirá si todas las funciones previstas funcionan o al menos si están implementadas. (**Completitud funcional**).

Los CDP, que hacen las veces de pruebas funcionales, nos proporcionarán información detallada de qué debe hacer el programa en situaciones específicas, además nos informará de si en estas situaciones el programa funciona correctamente. (**Corrección funcional**)

La especificación de requisitos (HDU, NDC y CDP) (**Pertinencia funcional**) es la característica de la adecuación funcional más sutil, para evitar hacer esfuerzos innecesarios o consumir recursos inútiles, se determinará mediante las reuniones con el cliente, u otro contacto pertinente, que es necesario y que no en el programa.

El **DOR** reducirá al mínimo las posibilidades de que se necesite cambiar la especificación de la HDU una vez comenzado el desarrollo. Esto evitará perdidas de tiempo y permitirán a los desarrolladores centrarse en tareas que están preparadas para realizarse.

Superar el **DOD** implicará que el equipo de desarrollo tiene la certeza de que su trabajo está terminado y que es el correcto y el solicitado por el cliente pues todas las pruebas de la HDU estarán superadas.

5.1.14 Resultados del modelo de madurez

A continuación se detalla en una tabla los resultados esperados para adquirir el nivel 2 de madurez siguiendo el modelo expuesto en [23], estos resultados se exponen junto a la evidencia equivalente dentro del ciclo de vida de Madrija definidos, algunos por parte de Madrija y otros por parte del alumno, también se relaciona esta evidencia con el responsable de llevarla a cabo.

Definición de requisitos de Stakeholder		
Descripción	**Evidencia**	**Responsable**
RP1: Se especifican las características requeridas y el contexto de uso de los servicios.	Primero con las *actas de reuniones* y *documentos* aportados por el cliente y posteriormente con *necesidades de cliente* y con las HDU.	Alumno
RP2: Se definen las restricciones del sistema.	Primero con las *actas de reuniones* y *documentos* aportados por el cliente y posteriormente con *necesidades de cliente* y con las HDU.	Alumno
RP3: Existe trazabilidad de los requisitos, entre stakeholders y entre sus necesidades.	La trazabilidad se establece entre las *Necesidades del cliente* y las HDU.	Alumno
RP4: Se describe la base para la definición de los requisitos del sistema	Se redactan las *Necesidades del cliente.*	Alumno
RP5: Se describe la base para validar la conformidad de los servicios	*Criterios de Aceptación.*	Alumno
RP6: Se proporciona la base para negociar y acordar la entrega de un producto o servicio.	Las *Necesidades del cliente* se usarán para tal fin, posteriormente serán las HDU.	Madrija
Análisis de requisitos del sistema		
Descripción	**Evidencia**	**Responsable**
RP1: Se establece y define un conjunto de requisitos funcionales y no funcionales que describen el problema a resolver.	Primero con las *actas de reuniones* y *documentos* aportados por el cliente y posteriormente con *necesidades de cliente* y con las HDU.	Alumno
RP2: Se desarrollan las técnicas apropiadas para optimizar la solución seleccionada para el proyecto.	Proceso de consultas iterativas con el cliente	Madrija y Alumno
RP3: Se analizan los requisitos del sistema para ser corregidos y probados.	Uso de *checklists.*	Alumno
RP4: Se comprende el impacto de los requisitos del sistema en el entorno de explotación.	Estimaciones del Sprint Planning.	Madrija
RP5: Se priorizan, aprueban y actualizan los requisitos.	Sprint planning estimación y cambios de requisitos.	Madrija

Análisis de requisitos del sistema		
Descripción	**Evidencia**	**Responsable**
RP6: se establece la consistencia y la trazabilidad entre los requisitos del sistema y la línea de base de requisitos del cliente.	Relaciones de Open Project entre HDU y *necesidades del cliente.*	Alumno
RP7 se evalúan los cambios en la línea de base frente al coste, calendario e impacto técnico.	Sprint planning y estimaciones.	Madrija
RP8: Los requisitos del sistema se comunican a toadas las partes afectadas y se colocan en la linea de base.	Los requisitos estarán visibles para todos los desarrolladores en Open Project y se explicarán en los sprint planning, al cliente se le comunica mediante reuniones o cualquier otro medio pertinente.	Madrija
Aseguramiento de la calidad del software		
Descripción	**Evidencia**	**Responsable**
RP1: Se desarrolla una estrategia para llevar a cabo el aseguramiento de la calidad.	Conjunto de pruebas(unitarias, de integración, de aceptación).	Madrija y el alumno
RP2: Se producen y mantienen pruebas de aseguramiento de la calidad del software.	Proceso de pruebas de aceptación.	Alumno
RP3: Se identifican y registran problemas y no conformidades con los requisitos.	Reuniones con clientes, checklists, y versiones de las HDU.	Madrija y el alumno
RP4: Se verifica el cumplimiento por parte de los productos, procesos y actividades de los estándares, procedimientos y requisitos.	Checklist para saber si se registran requisitos, si se pasan pruebas, si se cumplen los estándares para que pruebas,etc. .	Madrija

Cuadro 5.5: Resultados para el modelo de madurez

5.2 Ejecución de las iteraciones

En este apartado se explicará de forma sintetizada y dando prioridad a lo más importante cómo se han desarrollado las diferentes iteraciones planificadas.

5.2.1 Iteración 1

Esta iteración es el arranque del trabajo y se desarrolla según lo planeado, la tarea propuesta es la obtención de un sistema de requisitos y el resultado es la creación de una serie de plantillas para recoger los documentos que consideramos serán nuestros requisitos las *Historia(s) de Usuario (*HDU*)*, entidades, requisitos de seguridad y requisitos tecnológicos. Se crean además unas *Checklists* para que los requisitos antes mencionados se redacten conforme a lo que se desea y necesita.

Se destaca la fase de identificación de errores, donde se encuentran fallos en el sistema actual de Madrija para la gestión de los requisitos. Y la definición del problema donde se analizan y resumen estos fallos. Estas fases serán la piedra angular de esta iteración y las cuales guiarán las plantillas posteriormente desarrolladas.

Identificación de errores.

Analizando la gestión de requisitos realizada hasta ahora en la empresa se encuentran una serie de errores. Los requisitos analizados proceden de uno de sus últimos proyectos. Los números corresponden a la identificación de OpenProject, la herramienta de gestión de la empresa:

Los errores mostrados a continuación pertenecen a un proyecto que tiene el objetivo de crear un sistema de bolsas de trabajo. Aunque los errores descritos cumplen una función esencial para avanzar en el trabajo, es probable que algunos no tengan sentido para alguien ajeno al proyecto.

- En la HDU 513 no se aclara cuantos dígitos debe tener el login como máx. ni tampoco qué hacer si se introducen más. (bug 531).
- En la HDU 513, no se hace referencia a que el campo DNI pueda ser otro tipo de documento (Bug 592).
- No hay ninguna referencia en las HDU a la posibilidad de subir la vida laboral. Debería aparecer, probablemente en la HDU 519. Sin embargo, esto aparece en la Task 536. Cambio el tipo Task a HDU.
- De la misma manera la tarea 540 ha sido cambiada a HDU.
- Bug 582. La HDU de recuperación de contraseña (517) debería describir el comportamiento completo pero no especifíca si al regresar a la página se debe dar la opción de hacer login o no.

- Bug 572. No hay ninguna referencia en los requisitos que nos indique que los méritos son modificables o que siquiera se pueden borrar. Esto se encuentra en la task 535.
- Bug 564. Los bug no deberían tener solo nombre.
- Bug 532. En pantallas de este tipo se debe tener en cuenta que los usuarios tienen diferentes costumbres a la hora de introducir datos, se debería aclarar en una HDU global para todo el producto el comportamiento del teclado si se cree pertinente. Otra opción es aclarar, en cada HDU en la que se precise, el comportamiento del teclado.
- Bug 545. En La HDU 513 quizá hubiera que establecer el formato de los apellidos, y de esta manera dar al usuario la información necesaria para introducir los datos correctamente.
- Bug 566. No hay un requisito/HDU que explique como debe funcionar la función del correo.
- En la HDU 520 hay un cambio de requisitos que no se registra (Extensión de archivos subidos), es conveniente actualizar las HDU.
- En la HDU 513 se anuncia una HDU que explicará el contenido del correo, HDU que no encuentro por ningún sitio.
- 514 no tiene estructura de HDU.
- 522 no es una HDU.
- No se describe en ningún lugar el comportamiento de la pestaña formación, la cual da un fallo en 562.

En definitiva, se observan una serie de errores generales:

- Se detecta que una gran cantidad de bugs registrados, los más graves al parecer, se podrían evitar con una mejor gestión de los requisitos.
- Se observan errores de forma y de concepto a la hora de documentar las HDU.
    - Extensión excesiva de forma innecesaria.
    - Datos que no deben estar en una HDU.
    - Cada HDU sigue una estructura distinta sin ninguna norma aparente.
- Las HDU no están organizadas de ninguna manera, perjudicando la trazabilidad con los CDP.

Definición y explicación de errores.

El sistema de requisitos actual de Madrija en este proyecto concreto (Linceo), consiste en reflejar mediante "HDU" el comportamiento esperado del software, el jefe de proyecto actúa

de “proxy” del PO, práctica común en el mundo real al aplicar metodologías ágiles. Después, se usan HDU, optando por un sistema ágil para la gestión de los requisitos.

El problema aparece en el momento en que se aplican mal las HDU, los principales fallos que he encontrado son:

- Errores en el concepto de HDU, historias que no describen funcionalidad o lo hacen centrándose en cosas que no deberían.
- HDU que no estaban clasificadas como tal (no tenían la etiqueta de Historia de Usuario), lo que puede ocasionar errores a la hora de garantizar que los requisitos se cumplen, así como problemas para gestionar las HDU.
- Funcionalidades que surgen de la nada, en las que no hay ninguna HDU que indique que debe hacer. Si se implementa una función debe haber algún documento que refleje el comportamiento que debe seguir, de nuevo, buscando que esas funcionalidades estén unidas a los requisitos correspondientes.
- HDU desactualizadas, si el único documento de requisitos van a ser las HDU, deben estar al día para posteriores revisiones. No hay que tener miedo a cambiarlas, son un documento vivo y cambiante.
- Falta de detalle. Existen entre las HDU algunas en las que la falta de detalle provoca posteriormente algún que otro bug, este es el error más complicado de solucionar, pues la única manera es que el encargado (PO) de redactar estas HDU, tenga la experiencia o sagacidad suficiente para que la precisión de sus descripciones sea la correcta para cada tarea (tampoco hay que excederse en el detalle).
- Aunque hay intentos de trazabilidad entre algunos bugs y susHDU, algunos bugs no están relacionados con ninguna HDU cuando si deberían estarlo.
- Los CDP están nombrados, pero no hay ninguna descripción.

5.2.2 Iteración 2

El objetivo de esta segunda iteración es conseguir un sistema de pruebas que verifiquen y validen los requisitos.

Para conseguirlo se ha optado por la creación de una plantilla de Casos de prueba(CDP) que verificarán los criterios de aceptación y los escenarios de las HDU. Además se usará una matriz de trazabilidad que relacione los requisitos con las pruebas que los validan.

En esta segunda iteración el trabajo a realizar se centrará en la segunda parte considerada en este trabajo como fundamental para la adecuación funcional: **las pruebas**.

Para los CDP no hay aún un sistema establecido aún en Madrija pero sí que se han hecho esfuerzos informales para comprender su funcionamiento y aplicarlos en la medida de lo posible.

El error más grave cometido en las pruebas funcionales/de aceptación que se realizan actualmente es que no se sigue ninguna secuencia, las pruebas se hacen sobre la marcha, simulando a un usuario cualquiera. Además no hay registros de ningún tipo sobre qué se ha probado, cómo se ha probado o si funciona correctamente o no. Con la excepción de algunos bugs publicados en la plataforma OpenProject, pero ni siquiera se publican todos los bugs.

Esta forma de hacer las pruebas implica que las pruebas se centrarán en la parte del sistema que la persona que lo testea crea conveniente y que se dependerá simplemente de su visto bueno para validar el sistema.

El objetivo del nuevo método de pruebas es que se compruebe que todas las funcionalidades acordadas con el cliente funcionan, consiguiendo de esta manera verificar los requisitos. Para conseguir este objetivo se plantearán unas buenas prácticas para las pruebas funcionales que comenzarán con la toma de requisitos.

Durante la creación de las HDU, se asignará a cada HDU una serie de criterios de aceptación, a raíz de estos criterios de aceptación se configurarán una serie de documentos llamados pruebas de aceptación o CDP y que contendrán unos pasos que hay que ejecutar para que la prueba sea superada. Además, las HDU contemplan en la mayoría de casos la inclusión de escenarios, estos escenarios definirán a fondo el comportamiento del sistema desarrollado, los escenarios tendrán, cada uno, asociada otro CDP que al igual que los criterios de aceptación verifiquen su adecuado comportamiento. Si todos los CDP funcionales de cada requisito superan los pasos significará que todos los requisitos han sido saldados.

5.2.3 Iteración 3

Dado que la empresa quiere conseguir certificarse con respecto al modelo de madurez de la norma ISO 12207, se aprovechará el trabajo realizado buscando la adecuación funcional para satisfacer también el modelo de madurez.

De los procesos solicitados por el segundo nivel de madurez de Spice (ISO 15504) este TFG se centrará en los tres procesos que se han considerado tienen más relación con el trabajo desarrollado. Estos procesos son:

Proceso de definición de requisitos de *stakeholder*. El objetivo de este proceso es definir los requisitos necesarios para que el sistema pueda proporcionar los servicios necesarios a usuarios y otros interesados en un entorno definido. Para ello se identifican las necesidades y deseos de las partes interesadas o las clases de *stakeholders* participantes en el sistema a lo largo del ciclo de vida. Estas necesidades y deseos se analizan y transforman en un conjunto común de requisitos de *stakeholders* que describe la interacción deseada que el sistema tendrá con su entorno operativo. Los requisitos de este conjunto son la referencia contra la que se valida cada servicio operacional resultante con el fin de confirmar que el

sistema satisface necesidades.

Proceso de análisis de los requisitos del sistema El objetivo de este proceso es transformar los requisitos de *stakeholders* en un conjunto de requisitos técnicos del sistema deseado que guiarán el diseño del sistema.

Proceso de aseguramiento de la calidad del software. El objetivo de este proceso es asegurar que los productos de trabajo y los procesos cumplen con las disposiciones y los planes predefinidos.

De cada uno de los procesos se ha analizado qué se pide hacer para cumplir el nivel 2 de madurez. Los procesos constan de una serie de actividades que generarán un conjunto de resultados y de productos de trabajo. Se ha considerado que lo más importante a tener en cuenta son los resultados y tras el análisis se determinan una serie de evidencias para cada uno de los resultados de cada uno de los tres procesos.

A modo de resumen, se genera una tabla, dividida en: Descripción de los resultados del proceso, Evidencia desarrollada o por desarrollar y Responsable, el campo responsable mostrará si la evidencia ha sido desarrollada/será desarrollada por el alumno o si, por el contrario ha sido la empresa Madrija quien lo ha desarrollado.

5.2.4 Iteración 4

Esta iteración estaba planificada para generar un nuevo flujo de trabajo incorporando las novedades para los procesos ya existentes y los procesos que antes no se llevaban a cabo en modo alguno.

Debido a errores encontrados durante la ejecución de la tercera iteración, así como una serie de necesidades no previstas durante la planificación inicial, en adición al flujo de trabajo, se llevarán a cabo tareas de corrección sobre las diferentes plantillas de requisitos y se creará una nueva plantilla para las *Necesidades de cliente*.

Las necesidades de cliente servirán para dar una primera versión de los requisitos y también se usarán a modo de acuerdo con el cliente. A partir de estas *Necesidades de cliente* se conseguirán el resto de requisitos (HDU, entidades y CDP). En esta iteración se descartan también las plantillas de requisitos de seguridad y tecnológicos, optando por usar la misma plantilla de HDU para todos los requisitos funcionales y no funcionales, a excepción de las entidades.

Capítulo 6

Conclusiones y Trabajo Futuro

TRAS finalizar este TFG aparte de los resultados ya expuestos, se obtienen una serie de conclusiones que se expondrán a continuación. Además se presentarán ideas para trabajo futuro así como los objetivos que han quedado incompletos debido a la falta de tiempo y a la priorización de otros objetivos.

6.1 Conclusiones

Elicitación de requisitos. Las plantillas para requisitos junto con las técnicas de elicitación propuestas y el método de trabajo diseñado facilitan la tarea de elicitación de requisitos.

Estos elementos son necesarios para facilitar la tarea de elicitación pero no son suficientes, a la hora de trabajar los requisitos, hay un eslabón fundamental, el product owner. El product owner será el encargado de que la elicitación sea un éxito y para ello se disponen las herramientas ya mencionadas (plantillas y técnicas de elicitación) pero son cruciales dos factores independientes: Habilidades sociales y sagacidad.

Habilidades sociales. Durante las reuniones para obtener los requisitos es muy importante la capacidad del Product Owner para guiar las conversaciones, reconduciendo la reunión hacia temas que considere incompletos o hacia temas que crea importantes y que aún no hayan surgido. Estas habilidades sociales también permiten comprender mejor las peticiones de los *stakeholders*.

Sagacidad. Será muy importante la velocidad para determinar qué es realmente lo que está pidiendo el *stakeholder*, bien durante las reuniones, bien a la hora de redactar las NDC, HDU, entidades y CDP.

Estas dos facultades pueden tener dos orígenes distintos, hay gente que tiene estas facultades de forma **innata**, pero como cualquier habilidad se pueden perfeccionar gracias a la **experiencia**.

6.2 Trabajo futuro

Plugin para OpenProject. Se propone la creación de un plugin que añada campos más específicos a OpenProject con funciones especiales, un campo versión que cambie con cambios

importantes y registre dichos cambios.

Generar documentación desde open project. Se plantea la creación de un pequeño programa en Java que genere la documentación registrada en Open Project y la transfiera a un documento PDF o docx. Este documento se podrá usar como parte del contrato con el cliente.

Automatización de las pruebas. Se plantea la creación de un sistema de automatización de pruebas de aceptación. Esta automatización se realizará seguramente mediante *Selenium*. La automatización permitirá ahorrar costes y tiempo de trabajo en, por ejemplo, pruebas de regresión.

Mejora de los procesos para el nivel 3. Siempre hay espacio para la mejora, actualmente se está trabajando en la certificación del nivel 2 del modelo de madurez, lo natural es que, tras superar la evaluación del nivel 2 y obtener la certificación del mismo, el paso siguiente sea conseguir la certificación del nivel 3.

Posibles cambios. Lo más probable es que tanto las plantillas como el flujo de trabajo se vayan adaptando a nuevas tecnologías o paradigmas que se crea conveniente implantar en Madrija.

Evaluación del sistema. Todo el trabajo realizado no tendría sentido si finalmente no se realizase una evaluación de la empresa para la adecuación funcional según el estándar ISO/IEC 25000. Esta evaluación será llevada a cabo por Madrija en un futuro no lejano. Se realizará un proceso de evaluación real de la adecuación funcional, utilizando como entradas para el laboratorio los entregables generados siguiendo las mejoras a los procesos de requisitos y de pruebas realizados en este TFG, para validar que con esta nueva forma de trabajar se cumplen con las necesidades para que posteriormente el producto pueda asegurar su adecuación funcional.

6.3 Cumplimiento de objetivos

A continuación se muestra una tabla resumen con los objetivos previstos para este TFG, si se han logrado y un comentario de justificación.

Objetivos del TFG		
Descripción	**Logrado**	**Aclaración**
Objetivo 1. Analizar, diseñar e implantar un buen sistema de adquisición y gestión de requisitos.	Sí	Se han diseñado unas plantillas con las que se registrarán los requisitos en forma de NDC, HDU y Entidades
Objetivo 2. Analizar, diseñar e implementar un sistema de pruebas de aceptación.	Sí	Las pruebas se registrarán como CDP y se deberán superar para dar por finalizado un desarrollo
Objetivo 3. Se adaptará el flujo de trabajo del equipo de desarrollo al nuevo sistema de requisitos y de pruebas.	Sí	Se han diseñado unas líneas base para la gestión del ciclo de vida de los proyectos. Esta guía usa OpenProject y las plantillas desarrolladas.
Objetivo 4. Automatizar las pruebas, en la medida de lo posible se buscará e implementará la manera de que la mayor cantidad de pruebas de aceptación sean automáticas.	No	A pesar de que se ha hecho un esfuerzo de investigación en diferentes herramientas, se ha optado por dar prioridad a otros objetivos que eran más importantes para la empresa.
Objetivo 5. Se sincronizará el trabajo sobre requisitos y pruebas con el modelo de madurez SPICE. Comprobando qué procesos de la norma procesos descritos en la ISO/IEC 12207 se pueden cumplir gracias al sistema nuevo y cuales se pueden cumplir sin cambios radica-les.	Si	Se ha llevado a cabo una evaluacion liviana de los procesos solicitados por el modelo de madurez de AENOR que se considera que están más relacionados con los requisitos y las pruebas.
Objetivo 6. Como complemento se buscarán herramientas de gestión de pruebas para seleccionar la que mejor se ajuste a la forma de trabajar de la empresa.	No	Al igual que la automatización de las pruebas. Se deja de lado este objetivo para dar prioridad a otros que Madrija considera más importantes.

Cuadro 6.1: Tabla resumen de objetivos del TFG

ANEXOS

Anexo A

Listado de siglas y acrónimos

TFG	Trabajo Fin de Grado
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
CMM	Capability Maturity Model
JAD	Joint Aplication Development
CMMI	Capability Maturity Model Integration
TDD	Desarrollo guiado por tests/pruebas
BDD	Busines driven developement
PO	Product Owner
ATDD	Desarrollo Dirigido por Test de Aceptación
STDD	Story Test-Driven Development
NDC	Necesidad/es de cliente
HDU	Historia(s) de Usuario
CDP	Caso(s) de prueba
DOR	Definition of Ready
DOD	Definition of Done
ENAC	Entidad nacional de acreditación
AENOR	Asociación Española de Normalización y Certificación
RRHH	Recursos Humanos

Anexo B

Uso de plantillas en OpenProject

BUSCANDO ilustrar el uso de las plantillas desarrolladas en el ecosistema de la empresa, y usando la herramienta OpenProject de gestión de proyectos, se plasmarán a continuación una serie de capturas de la aplicación junto con una breve explicación. Estas capturas pertenecen a un proyecto real llevado a cabo por Madrija usando el sistema de requisitos creado en el presente TFG.

OpenProject permite administrar cada proyecto de forma independiente, creando diferentes paquetes de trabajo para ellos estos paquetes de trabajo son, por ejemplo tareas, historias de usuario o bugs entre otros. OpenProject permite mostrar en una página todos los paquetes de trabajo de un proyecto.

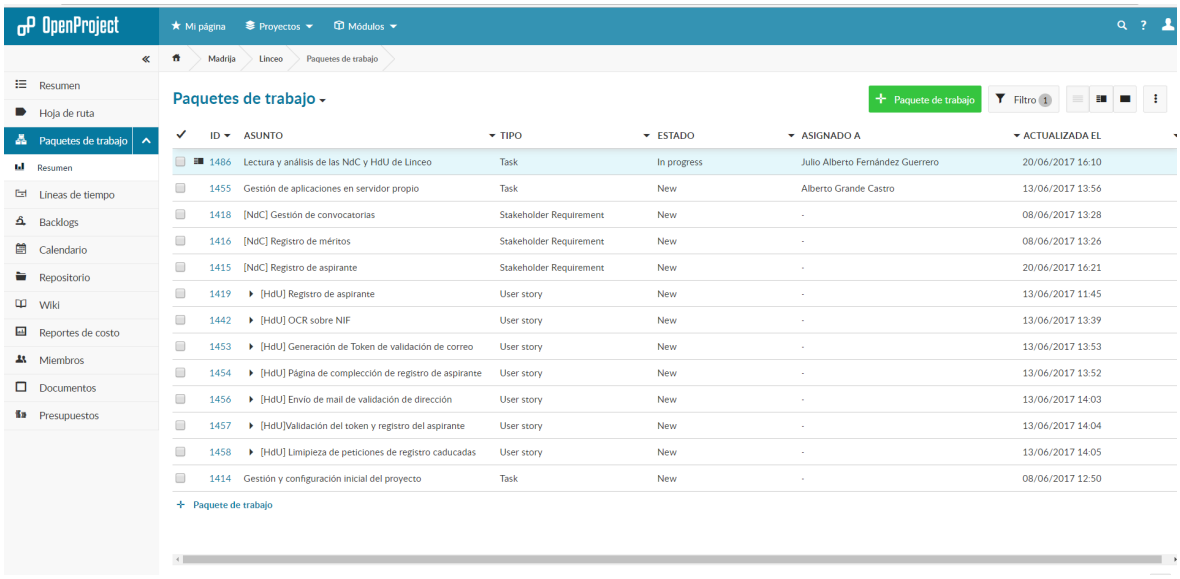


Figura B.1: Pantalla de Paquetes de trabajo de OpenProject

En la figura B.1 se ve el listado de paquetes del proyecto *Linceo*. Éstos paquetes se pueden ordenar según fecha de inicio, fecha de fin, nombre, responsable, estado, etc.

Una vez el PO tenga los documentos y la información necesaria, creará las NdC junto con el grupo de desarrolladores, las NdC se redactarán en la herramienta OpenProject siguiendo la plantilla.

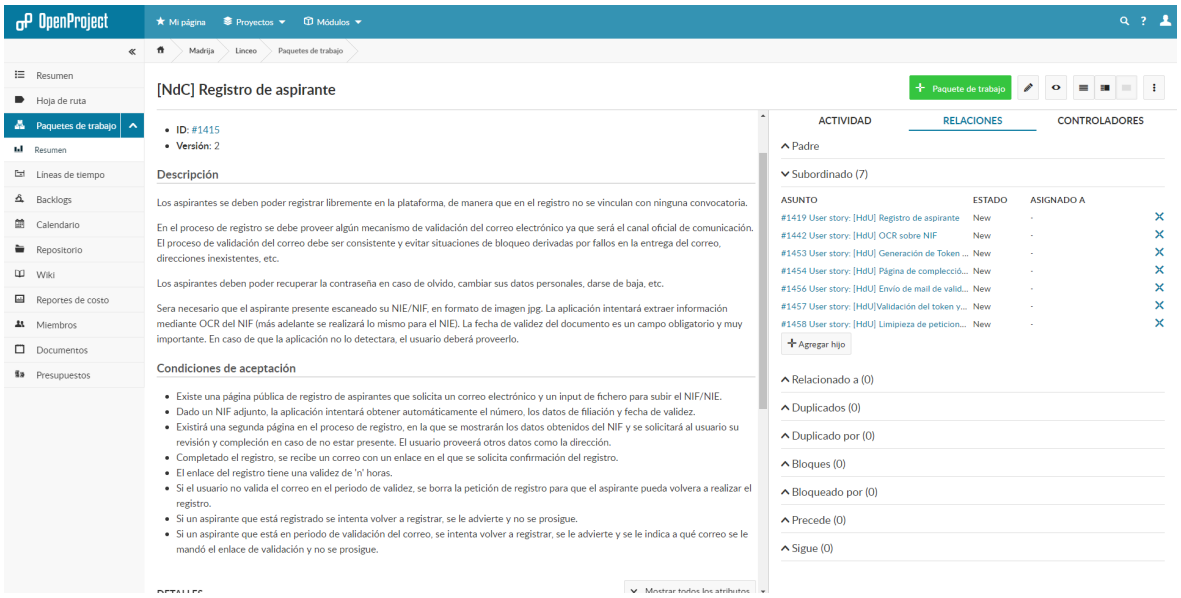


Figura B.2: Ejemplo de Necesidad de Cliente en OpenProject

En la figura B.2 se observa a la derecha la lista de HdU relacionadas con la tarea, y en el centro se aplica la plantilla en el campo de texto libre proporcionado por OpenProject. Un campo «Descripción» donde se expone la petición del cliente o *StakeHolder* y las «Condiciones de aceptación» las cuales permiten al equipo y al cliente ver de forma más detallada el alcance del requisito. A partir de estas condiciones de aceptación surgirán las HdU y los CdP.

Una vez creadas las NdC se procederá a crear las HdU.

En la figura B.4 se percibe una historia de usuario muy sencilla en la cual simplemente se necesita la descripción y un pequeño criterio de validación. Pero a medida que la historia de usuario sea más compleja se le añadirán campos, por ejemplo «escenarios» como se puede ver en la figura B.4 donde aparecen 1 escenarios alternativos y un escenario habitual. Pueden necesitarse otros campos como el de Mockup, el cual muestra un boceto de como será una pantalla del programa, en la figura B.5 se puede ver un ejemplo de este campo. Existen otros campos como «permisos», que muestran detalles sobre los permisos de acceso sobre los datos que se manejen en la HdU, o «Auditoría» que indica si hay datos que se necesiten registrar para posteriores auditorías o revisiones, este campo está relacionado con asuntos legales.

A partir de las HdU, las NdC y los criterios de aceptación surgirán los CdP (que se usarán como pruebas funcionales y de aceptación) y también se crearán las «Entidades», que serán los elementos que almacenen la información (en persistencia o de forma temporal). Tanto los casos de prueba como las Entidades no se pueden mostrar por motivos de privacidad acordados con Madrija.

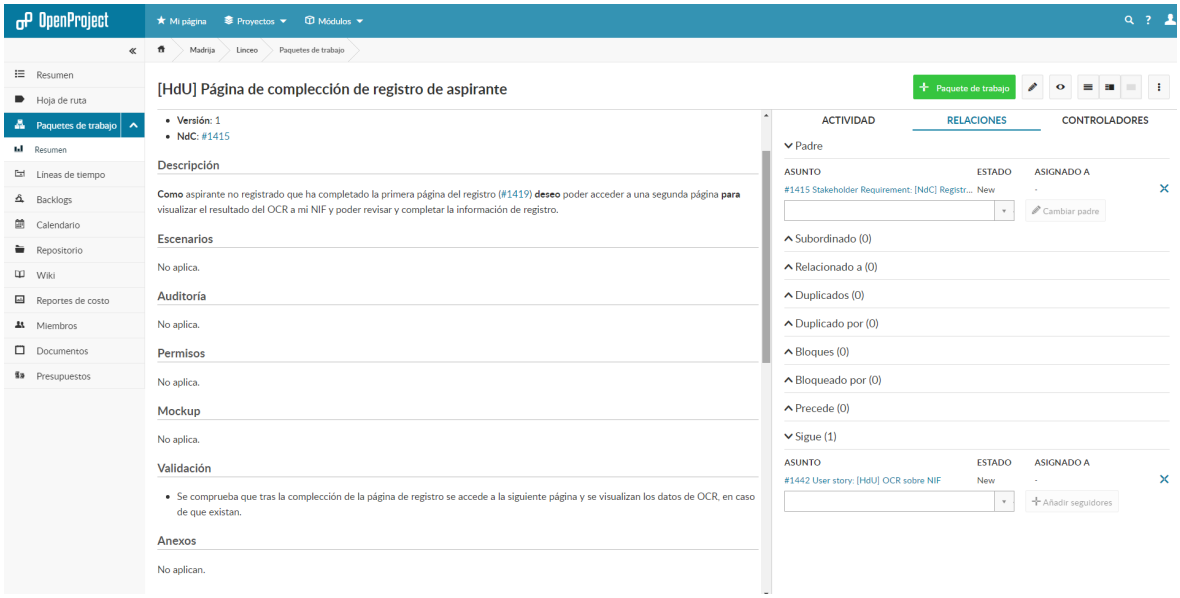


Figura B.3: Ejemplo de historia de Usuario en OpenProject

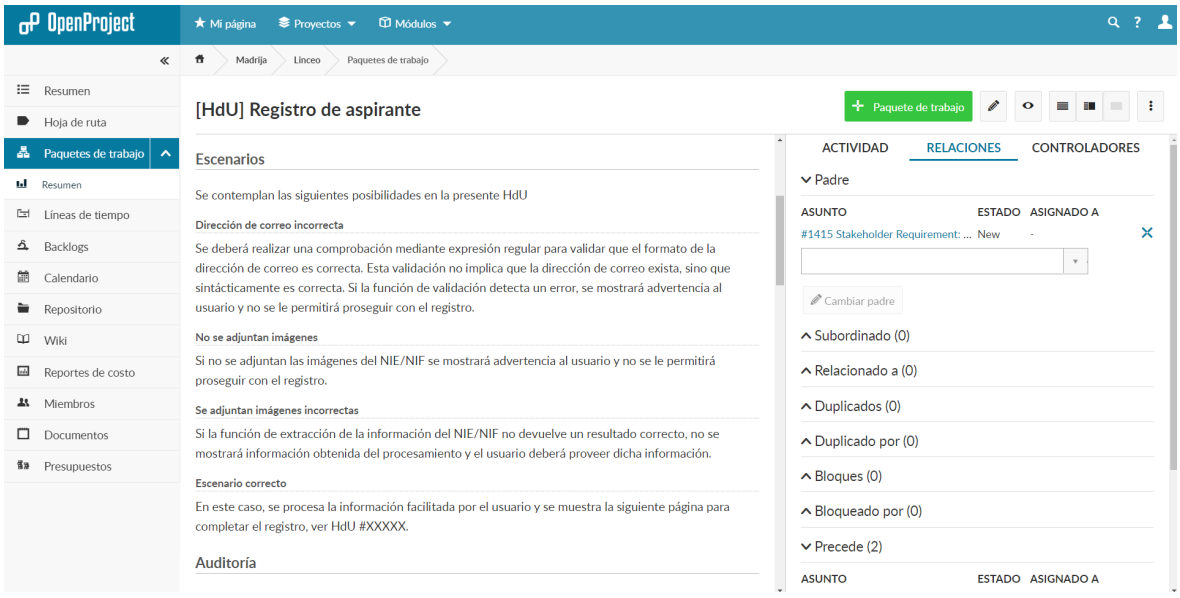


Figura B.4: Ejemplo de escenarios historia de Usuario en OpenProject

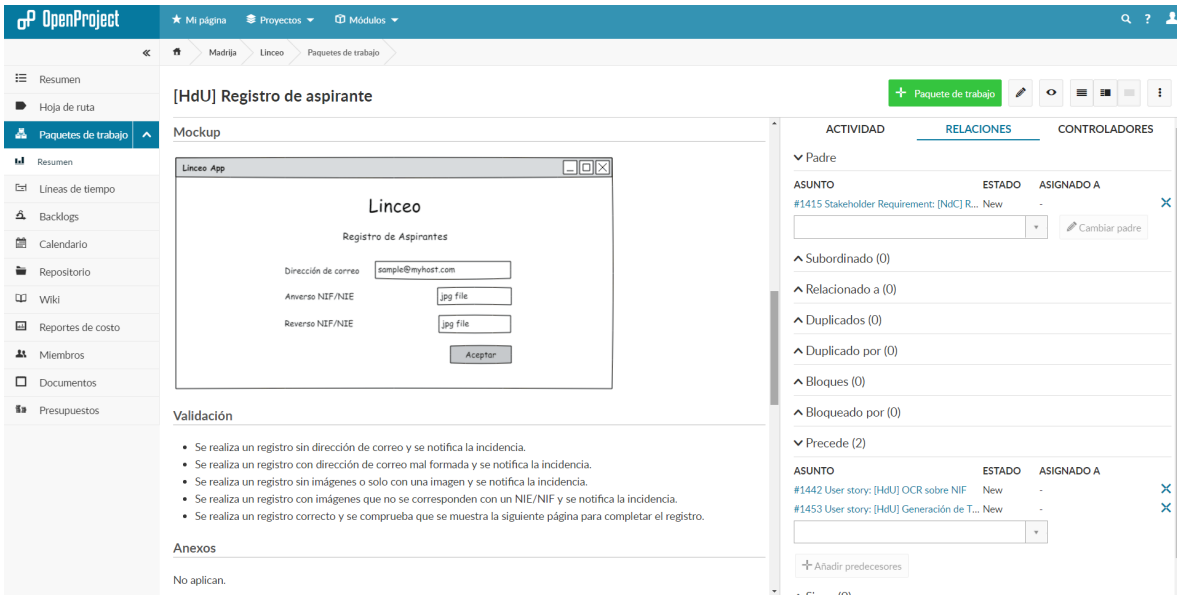


Figura B.5: Ejemplo de Mockup Historia de Usuario en OpenProject

Referencias

[1] BRANSFORD, J. D., AND STEIN, B. S. *The Ideal Problem Solver*. New York : W.H. Freeman, 1993.

[2] CMMI PRODUCT TEAM. Cmmi® for development, version 1.2. Tech. rep., Carnegie Mellon University, aug 2006.

[3] COHN, M. *User stories applied: for agile software development*. Addison-Wesley, 2004.

[4] DURÁN TORO, A., AND BERNÁRDEZ JIMÉNEZ, B. Metodología para la elicitación de requisitos de sistemas software. Tech. rep., Universidad de Sevilla. Departamento de Lenguajes y Sistemas Informáticos, Escuela Técnica Superior de Ingeniería Informática, oct 2001.

[5] EVOLUS. Pencil project. `http://pencil.evolus.vn/`, 2017. Accedido por última vez 08/06/2017.

[6] FERNÁNDEZ, C. M., RODRÍGUEZ, M., AND PIATTINI, M. Iso/iec 25000 calidad del producto software. *AENOR REVISTA DE LA NORMALIZACIÓN Y LA CERTIFICACIÓN*, 288 (dec 2013), 30–35.

[7] HOYER, R., AND HOYER, B. B. What is quality? *QUALITY PROGRESS 24*, 7 (2001).

[8] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. Web iso/iec 25000. `http://iso25000.com/`. Accedido por última vez 07/06/2017.

[9] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION & INTERNATIONAL ELECTROTECHNICAL COMMISSION. *ISO/IEC 14598-6:2001 Preview Software engineering – Product evaluation*. Ginebra, Suiza, 1999-2001.

[10] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION & INTERNATIONAL ELECTROTECHNICAL COMMISSION. *SO/IEC 9126-1:2001 Software engineering – Product quality*. Ginebra, Suiza, 2001-2004.

[11] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION & INTERNATIONAL ELECTROTECHNICAL COMMISSION. *ISO/IEC 12207:2008 Preview Systems and software engineering – Software life cycle processes*. Ginebra, Suiza, 2008.

[12] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION & INTERNATIONAL ELECTROTECHNICAL COMMISSION. *ISO/IEC 25000, SQuaRE (Software Product Quality Requirements and Evaluation)*. Ginebra, Suiza, 2011.

[13] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION & INTERNATIONAL ELECTROTECHNICAL COMMISSION. *ISO/IEC 15504, (SPICE) Software Process Improvement Capability Determination*. Ginebra, Suiza, 2013.

[14] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS & INTERNATIONAL ELECTROTECHNICAL COMMISSION. *ISO/IEC/IEEE 29119-2, Software and systems engineering – Software testing – Part 2: Test processes*. Ginebra, Suiza, 2013.

[15] JACOBSON, I. *Object-oriented software engineering: a use case driven approach*. ACM Press, 1992.

[16] JHONSON, J. *CHAOS Collection 2016*. Standish group, 2016.

[17] KOTONYA, G., AND SOMMERVILLE, I. *Requirements Engineering: Processes and Techniques*. Wiley Publishing, 1998.

[18] KYBELE-CONSULTING, AND AENOR. Certificaciones iso/iec 15504. `http://www.iso15504.es/`. Accedido por última vez 07/06/2017.

[19] LEVESON, N., AND TURNER, C. S. An investigation of the therac-25 accidents. *IEEE Computer 26*, 7 (1993), 18–41.

[20] NORTH, D. The evolution of behavior-driven development. *Better Software magazine 2006-03* (2006).

[21] OPENBOXER. `http://www.openboxer.260mb.com/asignaturas/is.php`. Accedido por última vez 20/06/2017.

[22] PIATTINI VELTHUIS, M. G., AND GARZÁS PARRA, J. *Fábricas de software : Experiencias,tecnologías y organización.*, segunda ed. RA-MA, 2010, ch. 7. LAS PRUEBAS DEL SOFTWARE.

[23] PINO CORREA, F. J., MARIO, P. V., AND FERNÁNDEZ SÁNCHEZ, C. M. *Modelo de madurez de ingeniería del software*. AENOR ediciones, 2014.

[24] RAGHAVAN, S., ZELESNIK, G., AND FORD, G. *Lecture Notes on Requirements Elicitation*. Software Engineering Institute, 1994.

[25] RODRÍGUEZ, M., OVIEDO, J. R., AND PIATTINI, M. Evaluation of software product functional suitability: A case study. *SOFTWARE QUALITY PROFESSIONAL 18*, 3 (jun 2016), 18–29.

[26] RODRÍGUEZ, M., AND PIATTINI, M. RevisiÓn sistemÁtica sobre la certificaciÓn del producto software. *Computer Science and Engineering* (jul 2012), 16–24.

[27] SOGETI, CAPGEMINI, AND ENTERPRISE, H. World quality report. Tech. rep., 2016-17.

[28] THE GANG OF FOUR: ERICH GAMMA, RICHARD HELM, RALPH JOHNSON AND JOHN VLISSIDES. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

Este documento fue editado y tipografiado con LaTeX empleando
la clase **esi-tfg** (versión 0.20170627) que se puede encontrar en:
`https://bitbucket.org/arco_group/esi-tfg`