Actas de las Jenui, vol. 3. 2018.
Páginas: 205-212

Hacia una metodología para el desarrollo guiado y sistemático de los Trabajos Fin de Grado

J.G. Enríquez, L. Morales-Trujillo, M. Olivero, F.J. Domínguez-Mayo, I. Ramos, M. Mejías
Departamento de Lenguajes y Sistemas Informáticos.
Escuela Técnica Superior de Ingeniería Informática. Universidad de Sevilla.
Avenida Reina Mercedes, s/n. 41010, Sevilla.
{jose.gonzalez,leticia.morales,miguel.olivero}@iwt2.org,
{fjdominguez,iramos,risoto}@us.es

Resumen

El Trabajo Fin de Grado (TFG) juega un papel muy importante en el proceso formativo de los grados en Ingeniería Informática, ya que, a través de la ejecución satisfactoria de un proyecto, se pretende que el alumno/a tenga su primera experiencia profesional tutorada como ingeniero/a en proyectos de ingeniería y tecnología del Software. En este artículo se presenta una propuesta de metodología de trabajo seguida por nuestro grupo de investigación a la hora de tutorar, dirigir y ejecutar los TFG de las titulaciones de grado relacionadas con la Ingeniería Informática. Esta metodología, está compuesta por tres fases principales: la fase de diseño del producto (basada en metodología Design Sprint), la fase de análisis de la solución (basada en la metodología NDT) y el diseño, implementación y pruebas de la solución (basada en metodologías ágiles de desarrollo). Gracias a esta forma de trabajo, el alumno es capaz de ejecutar el TFG de una forma sistemática, ordenada y teniendo una visión clara del producto y solución que debe desarrollar.

Abstract

The final project plays a very important role in the formative process of the degrees in Computer Engineering, because, the satisfactory execution of a project, intends that the student has his first professional tutored experience as real engineer in projects of software engineering and technology. In this paper we present a proposal of work methodology followed by our research group when it comes to tutoring, directing and executing the final project in the degree programs related to Computer Engineering. This methodology is composed of three main phases: the design phase of the product (based on Design Sprint methodology), the solution analysis phase (based on the NDT methodology) and the design, implementation and testing of the solution phase (based on in agile development methodologies). Thanks to this kind of working, the student is able to execute the final project in a systematic and orderly way, having a clear vision of the product and solution to be developed.

Palabras clave

TFG, Metodología de Desarrollo, Design Sprint, NDT, SCRUM.

1. Introducción

El Trabajo Fin de Grado (TFG) es una de las asignaturas más importantes de los grados de Ingeniería Informática. En ella, el alumno/a (o grupo de alumnos/as) deberá realizar un proyecto, memoria o estudio sobre un tema de trabajo que haya solicitado y se le haya sido asignado y en el que desarrollará y aplicará conocimientos, capacidades y competencias adquiridos en la titulación. A través de la realización del TFG, se pretende que el alumno/a tenga su primera experiencia profesional tutorada como ingeniero/a en proyectos informáticos. Tal es así, que algunos compañeros están llevando a cabo iniciativas en las que las empresas, son las que proponen TFGs para que sean ejecutados por los alumnos/as, intentando que esta experiencia sea lo más cercana posible a la realidad empresarial [13]. Gracias a la experiencia tutorando TFGs, los autores de esta publicación han comprobado que, a la hora de que un/a alumno/a se tiene que afrontar al desarrollo de un TFG, aunque las competencias que ha adquirido durante sus estudios son palpables, tiene serios problemas en cuanto a planificación, desarrollo y documentación del mismo. Esto, puede ser debido a que, hasta ese momento, lo más parecido que han hecho han sido trabajos finales de asignaturas con un alcance mucho más corto.

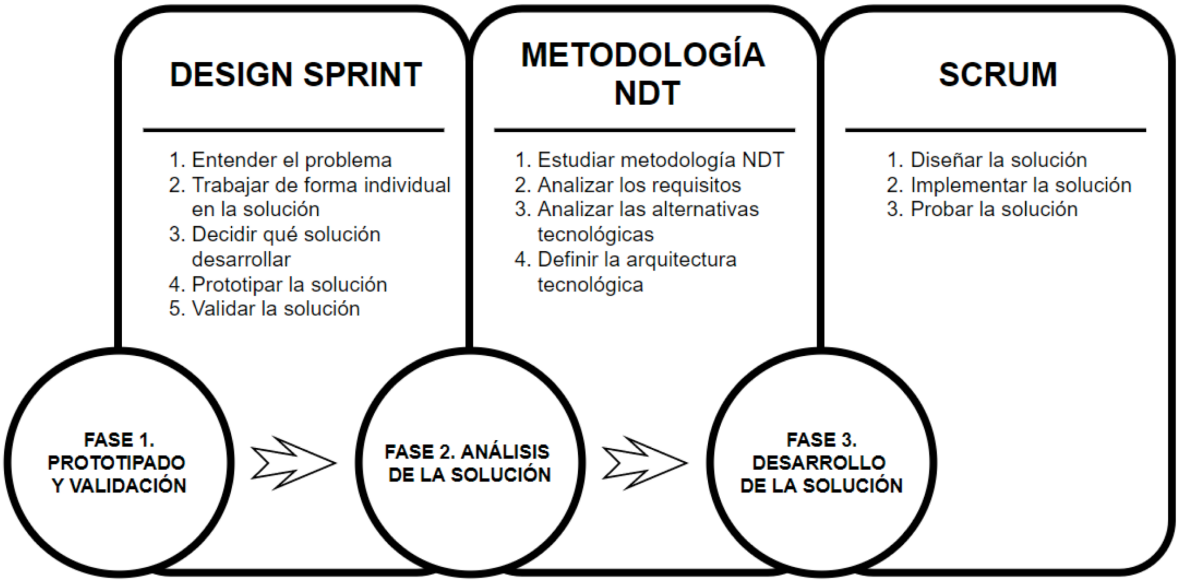

Figura 1: Metodología para desarrollo de Trabajos Fin de Grado.

En este trabajo, se presenta una propuesta metodológica para el desarrollo de TFGs, orientados a desarrollos de productos, con el fin de mejorar e intentar aprovechar al máximo esta actividad, intentando acercar lo máximo posible al alumno/a al mundo real empresarial. Para poder explicar de forma detallada todo lo mencionado anteriormente, el presente documento se estructura en las siguientes secciones: la Sección 2 presenta la propuesta metodológica en sí, en la que se explica de forma detallada cada una de las fases que la componen. La sección 3, describe todas y cada una de las herramientas de soporte que son necesarias utilizar para cumplir con las actividades propuestas para cada fase de la metodología. La sección 4, muestra una serie de resultados de aplicación de la metodología propuesta y finalmente, para concluir, la Sección 5 establece una serie de conclusiones y trabajos futuros a realizar en base a los resultados obtenidos.

2. Propuesta Metodológica

Esta propuesta metodológica (Figura 1), surge como resultado de observar cómo se desarrollan otros tipos de proyectos en las diferentes ramas de la ingeniería o arquitectura. En este sentido, podríamos dividir los proyectos en 3 fases claramente diferenciadas:

- **Prototipado y validación**: el primer paso que el alumno debe dar es entender el problema que se necesita resolver, aprender rápido, proponer un prototipo realista y, además, validarlo con el cliente. Para esta fase, se deberán utilizar técnicas de "Design Research", de forma más concreta, con Design Sprint '[20].
- **Análisis de la solución**: en esta fase, el alumno/a debe entrar mucho en el detalle de cómo debe ser la solución, realizando un análisis de requisitos, análisis de alternativas tecnológicas con las que implementar la solución y definiendo la arquitectura tecnológica de la misma. Para esta fase, se utilizará la metodología "Navigational Development Tools" (NDT) [11].
- **Desarrollo de la solución**: en esta última fase, el alumno deberá enfrentarse al desarrollo de la solución, más concretamente en el diseño, implementación y pruebas de la misma. Para esta fase y puesto que lo que se pretende es que el alumno obtenga los conocimiento más cercanos al mundo real, se ha decidido utilizar metodologías ágiles de desarrollo '[1].

Una vez introducida las tres fases, en las siguientes secciones pasaremos a explicar en más profundidad cada uno de los pasos que se deben dar para cumplir con la metodología propuesta.

2.1. Fase 1: Prototipado y validación

Cuando un alumno empieza su proyecto la primera tarea que debe hacer es conocer qué quiere hacer, para quién, y por qué, de modo que su proyecto pueda convertirse en una oportunidad de negocio al resolver una necesidad o problema real. Para obtener este conocimiento sobre la demanda existen algunas técnicas como el Lean Canvas de Erik Reis [19], o Design Sprint

[16] que ayudan de forma guiada a conocer mejor las necesidades de quien usaría tu producto. Para el desarrollo de los Trabajo Fin de Grado a los alumnos/as les proponemos la metodología Design Sprint. Este método propone un ciclo de cinco días para dar respuesta a los aspectos críticos de un negocio mediante el diseño, prototipado de las ideas y una prueba de concepto con los usuarios finales con un alcance limitado. La base de la metodología radica en obtener datos claros a raíz de un prototipo en lugar de esperar a un producto mínimo para entender si la idea tiene buena aceptación. Hacer este prototipado rápido nos permite adelantarnos a un desarrollo completo ahorrando grandes inversiones de tiempo y dinero. Design Sprint estructura el trabajo suficiente y necesario para conocer la demanda de los clientes en cinco días, de lunes a viernes. Desde que se ha planteado el problema que se quiere resolver hasta que encontramos una forma viable y validada de resolverlo. Aunque la metodología lo divide en una semana completa e intensa de trabajo, por motivos académicos es habitual que alumno deba separar los distintos "sprints" sin que todos se lleven a cabo en la misma semana. El inicio del proyecto lo marca el primer sprint donde se establece el objetivo a largo plazo, el qué se quiere resolver. Con un pensamiento optimista se plantea cómo debería ser el producto, y de modo pesimista se definen aquellos inconvenientes que deberemos evitar ya que nos impedirían alcanzar este futuro perfecto. Durante este sprint también se diseña un diagrama de actividades de los involucrados en la propuesta de solución donde, con ayuda de expertos que puedan aportar mejoras a este diagrama o ayudar a entender mejor el problema, se describen las tareas que se desempeñan actualmente en el entorno donde la propuesta del proyecto tiene cabida. Para este diagrama se establecen una serie de preguntas del tipo "¿Cómo podríamos... ?" que cuestionen y desafíen el modo actual en el que se desarrollan las actividades. El segundo sprint de esta metodología está centrado en la solución. El objetivo de este día es reunir ideas para nuestro prototipo. En lugar de necesitar ideas muy innovadoras se parten de ideas ya existentes. El proceso empieza con un ejercicio llamado Exposiciones Relámpago. En este ejercicio se revisan las soluciones y propuestas de otras industrias o empresas, o incluso ideas que puedan surgir a un nivel más cercano. Una vez reunidas las distintas alternativas se reúnen y se presentan a un conjunto de voluntarios para que nos puedan aportar su opinión acerca de ellas. La idea de estas exposiciones es conocer la impresión de los usuarios al hablarle de cada una de ellas. Estos esbozos contienen sólo la información suficiente para expresar la idea de forma clara, para que cualquier persona las pueda entender. El tercer sprint se basa en realizar procesos estructurados para realizar rápidamente buenas decisiones. En este sprint se reciben muchos comentarios, pero a los que hay que aplicarles el pensamiento subjetivo del alumno. Hay que tener en cuenta, mientras se lleva a cabo el sprint, el objetivo a largo plazo que nos planteamos ya que ahora es cuando se deben abordar todos los riesgos posibles. A este proceso de toma de decisiones se le llama las decisiones pegadizas. En grupo se realiza una pequeña defensa para sopesar qué características deberían incorporarse en el prototipo final. El alumno es quien finalmente decide qué ideas son las que pasan definitivamente al prototipo para ser validadas y cuáles se guardan como ideas en el tintero. El cuarto sprint tiene poca interacción ya que es el sprint donde se desarrolla el prototipo. Los objetivos principales del prototipo son: que cumplan el propósito de la idea que queremos evaluar con los voluntarios, no es necesario ninguna funcionalidad extra, y que tenga suficiente calidad como para que sea creíble y haga que quien pruebe el prototipo pueda reaccionar como si fuera un producto final. No es recomendado usar las herramientas habituales del desarrollo software. En su lugar se proponen una serie de herramientas que simulan el comportamiento de sistemas software finales de forma rápida y flexible, tales como: MS Power Point (en modo presentación simulando la navegación entre pantallas), Balsamic o draw.io entre otras. Para el quinto y último sprint ya se han analizado algunas soluciones y elegido las que se consideraron mejores y se ha construido un prototipo que recoge todas las características que hemos detectado como necesarias para los potenciales usuarios finales. El objetivo de este sprint es muy sencillo, es reunir impresiones de diferentes voluntarios de modo que podamos obtener información sobre si lo que se les pone por delante les es útil o no y por qué. Las nuevas iteraciones no se tienen que entender como un fracaso. Hay que tener en cuenta que se aprende mucho sobre la opinión de los potenciales clientes acerca de los prototipos invirtiendo apenas unos días y ahorrando el desarrollo de un proyecto completo. Cada nueva iteración que se lleve a cabo estará más próxima a resolver las necesidades o problemas reales, terminando con una validación positiva de los prototipos ante los evaluadores.

2.2. Fase 2. Análisis de la solución

NDT, es una propuesta englobada en el paradigma de la ingeniería guiada por modelos que se mueve dentro del entorno de la Ingeniería Web [8,9,11]. La Ingeniería Web es una rama de la Ingeniería del Software que define procesos, técnicas y modelos específicos para el entorno de la web. Sin embargo, actualmente la metodología NDT puede ser aplicada para especificación de cualquier tipo de sistema software. En sus inicios, NDT contemplaba únicamente un conjunto de metamodelos para el tratamiento formal de las fases

de ingeniería de requisitos y análisis de un proyecto de desarrollo de software. Asimismo, también definía un conjunto de reglas de derivación, especificadas con el lenguaje formal de transformaciones QVT (Query View Transformations, con las que es posible generar de manera sistemática todos los modelos de la fase de análisis a partir de los modelos de requisitos. En los últimos años, la metodología ha evolucionado y actualmente, proporciona soporte completo a todas las fases del ciclo de vida software: estudio de viabilidad, requisitos, análisis, diseño, implementación, mantenimiento y pruebas. Además, establece nuevas reglas de transformación entre cada una de estas fases. Toda esta definición teórica de la metodología se contemplaba como infranqueable en contextos empresariales debido al uso de una terminología demasiado abstracta (metamodelos, transformaciones, conceptos, etc.). Por ello, se hizo necesario desarrollar herramientas software que ocultas en esta terminología teórica, mejoraran la aplicabilidad de NDT en estos entornos y proyectos reales [7,10]. El grado de automatización de la metodología NDT es una de sus cualidades más relevantes gracias a que dispone de un amplio conjunto de herramientas distribuidas bajo el nombre de NDT-Suite [12]. Además, NDT ha evolucionado recientemente para proporcionar un marco de trabajo que motive el uso de nuevos enfoques, normas y paradigmas para el desarrollo de software de calidad. Todo ello desde una perspectiva basada en la definición de procesos software. Todos estos procesos están definidos de manera formal y completa en la herramienta NDTQ-Framework.

2.3. Fase 3. Desarrollo de la solución

Las metodologías de desarrollo ágil buscan elaborar software totalmente funcional en el tiempo o plazo establecido para el desarrollo del proyecto. Desde un primer momento, se estableció en esta propuesta metodológica que la tercera fase de la metodología propuesta, consistente en el desarrollo de la solución, se realizara con metodologías ágiles de desarrollo puesto que es una de las más extendidas en el mundo empresarial a día de hoy. En este sentido, fueron estudiadas diferentes propuestas antes de elegir cuál sería la que se debería aplicar. A continuación, se describen las metodologías que fueron valoradas. Dynamic Systems Development Method (DSDM) [15], define el marco para desarrollar un proceso de producción de software. Es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases. Feature-Driven Development (FDD) ‘[3]. Define un proceso iterativo que consta de 5 pasos. Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software. Adaptive Software Development (ASD) ‘[14], presenta las características: proceso iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de vida que propone define tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda se desarrollan las características y finalmente en la tercera se revisa la calidad del producto y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo. SCRUM [21], define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas “sprints”, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. Crystal Methodologies [4]. Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo (de ellas depende el éxito del proyecto) y la reducción al máximo del número de artefactos producidos. La Metodología XP programación extrema [2], se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, la comunicación fluida entre todos los participantes, la simplicidad en las soluciones implementadas y el coraje para enfrentar los cambios. XP tiene como objetivo proporcionar pequeños lanzamientos iterativos y frecuentes a lo largo del proyecto, lo que permite a los miembros del equipo y clientes examinar y revisar el progreso del proyecto a lo largo de todo el ciclo de vida de desarrollo de un sistema. En Lean Development (LD) [18], los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades que mejoren la productividad del cliente. Su principal característica es introducir un mecanismo para implementar dichos cambios. Metodología KANBAN ‘[17], es una metodología de desarrollo software centrada en gestionar el trabajo, con énfasis en la entrega justo a tiempo, mientras no se sobrecarguen los miembros del equipo. Se basa en el desarrollo incremental, dividiendo el trabajo en partes. La metodología KANBAN surgió en Toyota para organizar mejor su producción de vehículos dividiendo el proceso en fases bien delimitadas que se tenían que cubrir correctamente para pasar a

la siguiente fase, garantizando así un producto de calidad. Una de las principales aportaciones es que utiliza técnicas visuales para ver la situación de cada tarea. Teniendo en cuenta esta pequeña revisión, y otros estudios como [1,5,6] donde se hace una revisión más extensa, incluso algún análisis de las propias metodologías, podemos concluir que no existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software sino que toda metodología debe ser adaptada al contexto del proyecto. En base a lo aprendido en los estudios mencionados anteriormente y en base a la experiencia en gestión de proyectos que atesora el grupo de investigación que propone esta metodología, se ha optado por el uso de SCRUM, una de las más utilizadas en el mundo empresarial a día de hoy.

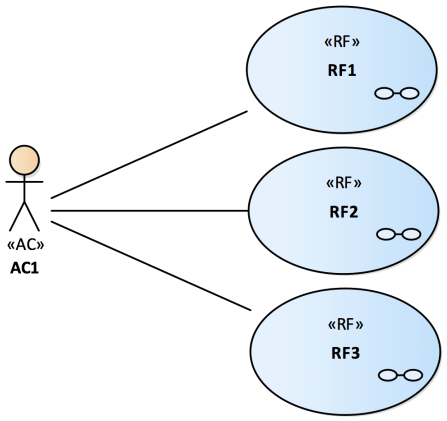

Figura 2: Definición de Caso de Uso con NDT-Suite y EA.

3. Herramientas de Soporte a la Metodología

Como hemos mencionado anteriormente, esta propuesta se basa en tres fases principales: la fase de diseño del producto (basada en metodología Design Sprint), la fase de análisis de la solución (basada en la metodología NDT) y el diseño, implementación y pruebas de la solución (basada en metodologías ágiles de desarrollo). En este sentido, y para dar apoyo a cada una de las fases, será necesario el uso de herramientas de soporte que permitan llevar a cabo el cometido de cada una de estas fases. En cuanto a la primera fase basada en la metodología Design Sprint, tal y como la propia metodología indica, necesitaremos un espacio amplio, en el que haremos uso de material de oficina como libretas, notas adhesivas y rotuladores, con el fin de plasmar y seleccionar aquellas ideas que definirán al proyecto en las siguientes fases. La fase de análisis de la solución estará cubierta por la metodología NDT. El conjunto de herramientas que da soporte a la metodología NDT es NDT-Suite. Para desarrollar NDT-Suite lo primero que se hizo fue hacer una extensión de la propia metodología. Tomando las ideas de NDT y siguiendo las premisas marcadas por la metodología Métrica V3 se hizo una extensión para abordar todo el ciclo de vida. NDT-Suite trabaja bajo el paraguas de la herramienta Enterprise Architect y está compuesto por 8 herramientas que serán descritas a continuación, sin embargo, tan solo las resaltadas en negrita son las que van a ser utilizadas en la metodología propuesta.

- **NDT-Profile**: implementa un perfil definido sobre la herramienta Enterprise Architect (ver ejemplo en Figura 2). En dicho perfil, se define cada uno de los metamodelos de NDT junto con sus artefactos asociados para trabajar con NDT.
- **NDT-Quality**: es una herramienta que toma como entrada un un proyecto realizado en base a NDT-Profile y comprueba que la trazabilidad y las normas de NDT se cumplen en el proyecto indicado.
- **NDT-Driver**: implementa un conjunto de procedimientos automáticos que permite llevar a cabo transformaciones entre los distintos modelos NDT definidos en un proyecto desarrollado con NDT-Profile.
- NDT-Prototypes: es una herramienta software de utilidad para dar soporte a la tarea de construcción sistemas de información web que han sido especificados utilizando la metodología NDT.
- NDT-Glossary: es una herramienta software de utilidad para dar soporte a la tarea de Elicitación de Requisitos, englobada dentro de la fase de Ingeniería de Requisitos del ciclo de vida de desarrollo de un producto software.
- NDT-Checker: es la única herramienta de NDT-Suite que no está basada en el paradigma MDE. Esta herramienta incluye un conjunto de plantillas, diferente por cada elemento de NDT.
- NDT-Counter: es una aplicación de escritorio, integrada en la suite de la metodología NDT. Esta aplicación nos servirá para calcular una estimación de esfuerzo para proyectos software, usando para ello la técnica de puntos de casos de uso.
- **NDT-Report**: está constituido por un amplio conjunto de plantillas integradas dentro de la herramienta NDT-Profile. Con NDT-Report puede generar fácilmente un documento para cada fase del ciclo de vida de NDT.

A través del uso de este conjunto de herramientas, los alumnos, serán capaces de describir y detallar el ciclo de vida completo del desarrollo de un producto software. Tal y como se ha definido anteriormente, la fase de diseño, implementación y pruebas de la solución se desarrollará con el soporte de metodologías

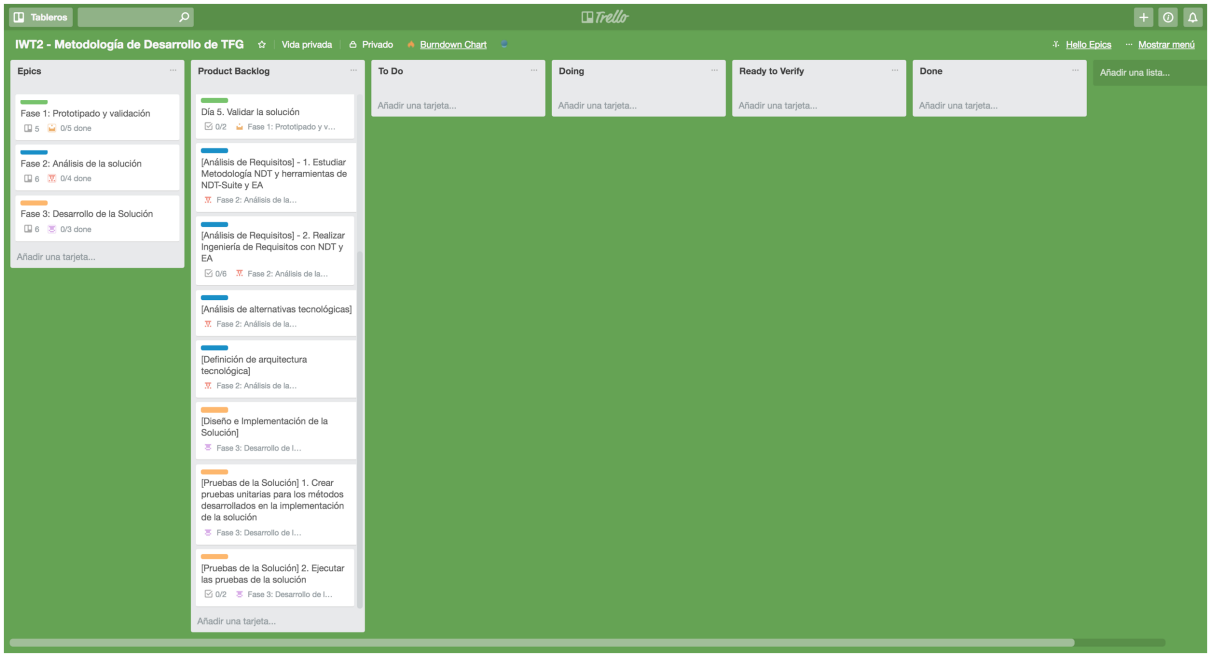

Figura 3: Tablero Trello predefinido.

ágiles de desarrollo, concretamente, usando SCRUM. Para dar soporte a esta metodología, utilizaremos Trello. Empleando el sistema kanban para el registro de actividades con tarjetas virtuales organiza tareas, permite agregar listas, adjuntar archivos, etiquetar eventos, agregar comentarios y compartir tableros. Trello es un tablón virtual en el que se pueden colgar ideas, tareas, imágenes o enlaces. Es versátil y fácil de aprender pudiendo usarse para cualquier tipo de tarea que requiera organizar información. Finalmente, ofrecemos a los alumnos/as un tablero ya predefinido (Figura 3), con todas las tareas a desarrollar para la realización de un TFG. Éste contiene una lista denominada “Epics”, que define cada una de las fases propuestas por la metodología. Por otra parte, también cuenta con una lista denominada “Product Backlog”, con todas las actividades que son necesarias desarrollar por cada una de las épicas. Finalmente, cuenta con las listas: “To Do”, que define tareas pendientes de realizar, “Doing”, que define tareas que están en curso, “Ready to Verify”, que define tareas que están listas para ser revisadas y “Done”, que define tareas ya terminadas.

4. Resultados de aplicación

Los alumnos que han realizado el TFG basándose en esta metodología, suelen tener una experiencia bastante satisfactoria. Algunos de los aspectos que nos hacen llegar a esta conclusión son los siguientes:

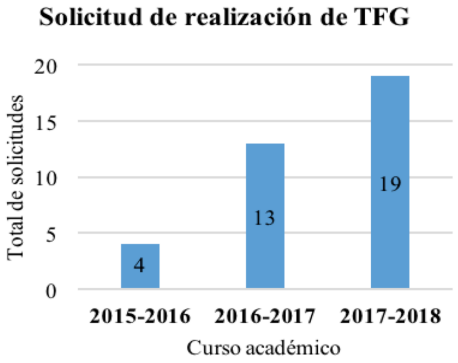

Figura 4: Evolución de solicitud de realización de TFG.

- **Solicitud de realización de TFG**: tal y como se muestra en la Figura 3, hemos notado un alto crecimiento en el número de alumnos que quieren desarrollar el TFG con nuestro grupo de investigación. Este método de trabajo les ayuda a enfrentarse a un desarrollo lo más cercano posible a un desarrollo en el mundo laboral. Esto, se debe al “boca a boca” de otros alumnos cuya experiencia fue positiva ya que la mayoría de los alumnos comentan que son éstos los que les han recomendado realizar el TFG usando esta metodología de trabajo.

- **Calificación de los TFGs realizados**: las calificaciones obtenidas por los alumnos son bastante

Curso Académico	Código	Calificación
2015-2016	TFG_1	9,00
	TFG_2	9,50
	TFG_3	9,00
	TFG_4	9,00
2016-2017	TFG_1	9,00
	TFG_2	9,00
	TFG_3	8,50
	TFG_4	9,50
	TFG_5	10,00
	TFG_6	9,00
	TFG_7	9,00
	TFG_8	9,00
	TFG_9	9,50
	TFG_10	8,50
	TFG_11	9,50
	TFG_12	10,00
	TFG_13	10,00

Cuadro 1: Histórico de calificaciones.

altas (Tabla 1), siendo un 8,50 la más baja obtenida en las últimas convocatorias. Por otra parte, los alumnos nos comentan que el tribunal suele darles la enhorabuena, y que, además, ponen en valor la forma en la que han desarrollado el TFG siendo muy cercano al desarrollo de un proyecto real.

5. Conclusión y Trabajos Futuros

Gracias a la experiencia tutelando trabajos de fin de grado, los autores de este trabajo, hemos podido concluir que los alumnos/as tienen grandes dificultades a la hora de ordenar todas las competencias adquiridas y ponerlas en práctica en el desarrollo del TFG. En este sentido, este trabajo de investigación presenta una propuesta metodológica que sirve de guía para el desarrollo de este trabajo de una forma clara, concisa y sistematizada. La propuesta metodológica desarrollada, se basa en tres fases fundamentales: (i) prototipado y validación, donde el alumno/a deberá entender y diseñar la solución al problema que se está planteando utilizando técnicas de Design Sprint [20], (ii) análisis de la solución, donde el alumno/a deberá entrar en el detalle de cómo debe ser la solución realizando las fases de requisitos y análisis a través del uso de la metodología NDT [11] y (iii), el desarrollo de la solución donde el alumno/a deberá diseñar, implementar y probar la misma, a través del uso de metodologías ágiles de desarrollo [1]. En cuanto a lo que trabajo futuro se refiere, los autores están trabajando en comprobar de una forma más empírica que los resultados del uso de la metodología propuesta son positivos, ya que, hasta el momento, tan solo han podido validarlo a través de la experiencia de los alumnos/as. Además, se está trabajando en comprobar la validez de este método para TFGs no orientados al desarrollo de un producto, como pueden ser los TFGs de investigación. Finalmente, los autores también están trabajando en realizar una comparativa entre las calificaciones de los TFGs que han seguido esta metodología y aquellos que no la siguieron.

6. Agradecimientos

Este trabajo también ha sido apoyado por el Proyecto Pololas (TIN2016-76956-C3-2-R) del Ministerio de Economía y Competitividad y el VPPI de la Universidad de Sevilla.

Referencias

[1] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J. (2017). Agile software development methods: Review and analysis. arXiv preprint arXiv:1709.08439.

[2] Ambler, S. (2002). Agile modeling: effective practices for extreme programming and the unified process. John Wiley Sons.

[3] Coad, P., Luca, J. D., Lefebvre, E. (1999). Java modeling color with UML: Enterprise components and process with Cdrom. Prentice Hall PTR.

[4] Cockburn, A. (2002). Agile software development (Vol. 177). Boston: Addison-Wesley.

[5] Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development.

[6] Dybå, T., Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. Information and software technology, 50(9-10), 833-859.

[7] Escalona, M. J., Gutierrez, J. J., Villadiego, D., León, A., Torres, J. (2007). Practical experiences in web engineering. In Advances in Information Systems Development (pp. 421-433). Springer, Boston, MA.

[8] Escalona Cuaresma, M. J., Mejías Risoto, M., Torres Valderrama, J. (2004). Developing systems with NDT and NDT-tool.

[9] Escalona, M. J., Mejías, M., Torres, J., Reina, A. (2003, July). The NDT development process. In International Conference on Web Engineering (pp. 463-467). Springer, Berlin, Heidelberg.

[10] Escalona, M. J., Torres, J., Mejías, M., Reina, A. (2003, July). NDT-Tool: A case tool to deal with requirements in web information systems. In In-

ternational Conference on Web Engineering (pp. 212-213). Springer, Berlin, Heidelberg.

[11] Escalona, M. J., Aragón, G. (2008). NDT. A model-driven approach for web requirements. IEEE Transactions on software engineering, 34(3), 377-390.

[12] García-García, J. A., Ortega, M. A., García-Borgoñon, L., Escalona, M. J. (2012, July). NDT-Suite: a model-based suite for the application of NDT. In International Conference on Web Engineering (pp. 469-472). Springer, Berlin, Heidelberg.

[13] Gutiérrez, J.J., I. Ramos, I. Barba, M. Mejías, J. Aroba, Actividades y resultados del Plan de Innovación Docente para la Participación de Empresas en la Docencia, in: Actas de Las XXIII JENUI, 2017: pp. 147–154. doi:978-84-697-4077-4.

[14] Highsmith, J. (2013). Adaptive software development: a collaborative approach to managing complex systems. Addison-Wesley.

[15] Jennifer Stapleton, (1997). Dynamic systems development method-The method in practice.

[16] Knapp, J., Zeratsky, J., Kowitz, B. (2016). Sprint: How to solve big problems and test new ideas in just five days. Simon and Schuster.

[17] Kniberg, H., Skarin, M. (2010). Kanban and Scrum-making the most of both.

[18] Poppendieck, M., Poppendieck, T. (2003). Lean software development: an agile toolkit. Addison-Wesley.

[19] Ries, E., The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Publishers Weekly. 258 (2011) 43. doi:10.1111/j.1540-5885.2012.00920.x.

[20] Rink, S., Google Design Sprint Overview, in: Google Design Sprint Overview, 2016: p. 103.

[21] Schwaber, K., M. Beedle, Agile Software Development with Scrum, 2001. doi:10.1109/2.947100.