FACULTAD DE INFORMÁTICA

UNIVERSIDAD POLITÉCNICA DE MADRID

TESIS DE MÁSTER

MÁSTER EN TECNOLOGÍAS DE LA INFORMACIÓN

ESTUDIO DE LA APLICACIÓN DE METODOLOGÍAS ÁGILES PARA LA EVOLUCIÓN DE PRODUCTOS SOFTWARE

AUTOR: Pilar Rodríguez González

TUTOR: Juan Garbajosa Sopeña

SEPTIEMBRE, 2008

A mis padres, José y Pili,
por su continuo apoyo y cariño.

Gracias.

Índice General

Índice de Figuras

Capítulo 1 Introducción

Capítulo 2 Estado del arte

Capítulo 4 Contexto del experimento

Capítulo 6 Desarrollo del experimento

Capítulo 7 Análisis de resultados

Índice de Tablas

Capítulo 2 Estado del arte

Capítulo 4 Contexto del experimento

Capítulo 5 Diseño del experimento

Capítulo 6 Desarrollo del experimento

Resumen

Las actuales características de dinamismo y variabilidad de la industria software han precisado replantear los cimientos sobre los que se sustenta el desarrollo software convencional. Un reciente estudio realizado por Boehm en [1], sobre la tendencia en ingeniería del software, indica que el mercado actual está caracterizado por el rápido desarrollo de aplicaciones y la reducción de la vida de los productos. En este entorno inestable la ventaja competitiva se encuentra en aumentar la productividad y satisfacer las variantes necesidades del cliente en el menor tiempo posible para proporcionar un mayor valor al negocio. Ante esta situación, cabe reflexionar sobre el grado de adaptación de las metodologías convencionales a estas circunstancias. La mayoría de los estudios coinciden en que el carácter normativo y la fuerte dependencia de planificaciones previas al desarrollo que definen a las metodologías convencionales, implican que resulten excesivamente *pesadas* para cubrir las necesidades de un amplio porcentaje del mercado software actual.

En los últimos años las metodologías ágiles han irrumpido con fuerza como un intento de despojar al desarrollo software del estricto corsé planteado por las metodologías convencionales, y son muchas las organizaciones punteras con creciente interés en las mismas. La novedad de estas metodologías hace que, aunque existen evidencias de los beneficios que pueden proporcionar en proyectos de pequeña envergadura, aun resulte difícil escalar a grandes proyectos. Algunos estudios recientes indican que la productividad y calidad del software aumenta aplicando los principios y valores que las rigen. No obstante, la mayoría de estos estudios se limitan a narrar observaciones cualitativas. Entre los que utilizan datos empíricos para apoyar sus conclusiones, los resultados son tan dispares como una mejora del 337% en la productividad en [2] y un decremento del 44% en [3]. Por este motivo, desde las organizaciones que promueven el desarrollo ágil de aplicaciones se solicita la realización de estudios sobre metodologías ágiles que permitan constatar o reprobar sus beneficios.

El objeto de esta investigación es estudiar la evolución de un producto software concreto utilizando las directrices marcadas por metodologías ágiles, en concreto por la metodología SCRUM. Se presentan los resultados obtenidos en aspectos tales como las características del producto a lo largo de la evolución, incluyendo estimaciones de la calidad del producto obtenido, la agilidad en el desarrollo, y evaluando el esfuerzo dedicado a adoptar la metodología. Además, dado que el factor humano es fundamental en este tipo de metodologías, se presenta un análisis cualitativo del desarrollo del proyecto.

Cabe destacar que el estudio aquí presentado se enmarca en una de las líneas de investigación del grupo SYST (*System and Software Technology Group*) de la Universidad Politécnica de Madrid, que participa en el proyecto ITEA2 Flexi [4]. En este proyecto se persigue mejorar la competitividad de la industria software europea proporcionando un entorno flexible, rápido y ágil para el desarrollo de aplicaciones que permita adaptarse a las actuales características del mercado para pasar de la idea al producto en seis meses.

Agradecimientos

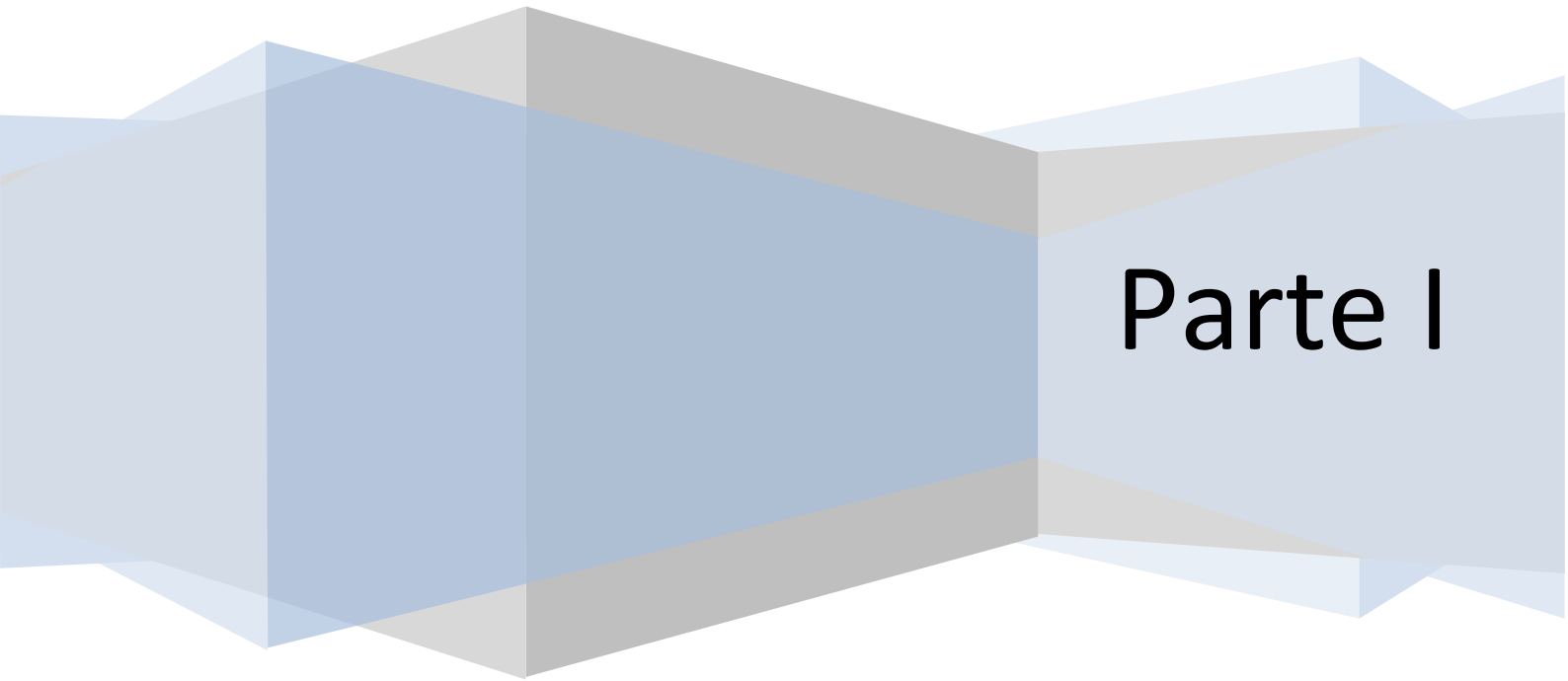
Este trabajo ha sido parcialmente financiado por los proyectos FLEXI ITEA2 (Ministerio de Industria, Turismo y Comercio, FIT-340005-2007-37 e ITEA2 6022) y OVAL/PM (Ministerio de Educación y Ciencia, TIC2006-14840).

También quisiera hacer público agradecimiento a la empresa BiogasFuelCell por el apoyo en el dominio de aplicación de la planta de biogás como proyecto dentro del experimento y Answare Tech, socio en Flexi.

Asimismo, al resto de componentes del equipo que gracias a su continuo entusiasmo y trabajo han hecho posible el desarrollo del experimento. Concretamente a Angelina Espinoza, Gema Rueda, Jennifer Pérez, Jessica Díaz, Agustín Yagüe, Arturo Gómez, Juan Garbajosa, Pedro P. Alarcón y Rodrigo Cavero.

Introducción al Estudio

Estudio de la Aplicación de Metodologías Ágiles para la Evolución de Productos Software



Capítulo 1

Introducción

Desde hace un tiempo nos venimos enfrentando a un profundo cambio en todos los sectores de la industria a nivel mundial. Un factor clave en esta situación es la emergencia de una economía global. Un porcentaje muy alto de la economía depende de las importaciones y exportaciones, lo que obliga a las compañías a gestionar y controlar su negocio en un mercado global, obviamente mucho más competitivo, y a distribuir sus funciones geográficamente para mantener una posición estratégica en el mercado. Este contexto desencadena, por tanto, que los valores que sustentan el negocio también estén sufriendo drásticas transformaciones. La ventaja competitiva se encuentra en nociones como el conocimiento, la información y la habilidad para enfrentarse a las variabilidades del mercado, por encima de otros factores clave hasta el momento como la cualificación del personal o los recursos logísticos. La aparición continua de nuevos productos y servicios, cuyo ciclo de vida se acorta, obliga a incrementar la productividad y disminuir el tiempo de reacción, adaptarse rápidamente a las variantes necesidades de los clientes y, en definitiva, aumentar la capacidad de competir en un mercado mucho más amplio.

Esta situación afecta muy especialmente a la industria software pues está llamada a ser, y de hecho ya es, el pilar que sustenta y aporta valor al resto de sectores industriales. En su libro *Software Engineering* [5] Fleeger ya identifica la transformación de la economía mundial como uno de los factores con más peso que ha cambiado el desarrollo software. Añade, además, otras claves que afectan en este sentido. La descentralización de las organizaciones y el trabajo distribuido, la continua reducción del *time-to-market*, la expansión del mercado software al público general y la constante variabilidad de sus necesidades, las interfaces de usuario, el avance tecnológico y el acaecimiento de problemas en cascada son puntos clave con enorme impacto en el desarrollo software en la actualidad.

En este entorno turbulento cabe reflexionar sobre el grado de adaptación de las metodologías utilizadas en la construcción de sistemas software hasta el momento, también conocidas, frente a otras nuevas corrientes, como metodologías convencionales. ¿Se adaptan bien las metodologías convencionales a las actuales características de dinamismo y variabilidad del mercado? ¿Necesitamos nuevas alternativas que se ajusten de un modo más fehaciente a dichas necesidades? Según Kent Beck, uno de los padres del modelo ágil, "*cada cosa en software cambia. Los requisitos cambian, el diseño cambia, el negocio cambia, la tecnología cambia, el equipo cambia y los miembros del equipo cambian. El problema no es el cambio, porque el cambio, inevitablemente, va a ocurrir. El problema, realmente, es nuestra inhabilidad para hacer frente a los cambios"*. Evidentemente, se puede concluir que parte del mercado software está variando y que estamos obligados a abordar este cambio con nuevas alternativas que nos permitan adaptarnos mejor a esta situación.

En este contexto las metodologías ágiles aparecen como una opción atractiva pues, en contraposición con las metodologías convencionales que actúan basadas en principios de estabilidad y control del contexto, las metodologías ágiles no se centran en habilidades de *predicción*, ni pretenden tener un sistema perfectamente definido como paso previo a su construcción, sino que perciben cada respuesta al cambio como una oportunidad para mejorar el sistema e incrementar la satisfacción del cliente, considerando la gestión de cambios como un aspecto inherente al propio proceso de desarrollo software y, permitiendo de este modo, una mejor adaptación en entornos turbulentos.

Por otro lado, un porcentaje muy alto del desarrollo software actual se centra en la evolución de productos ya construidos, con el objetivo de maximizar la inversión y reducir los costes, ya sea para variar parte de su funcionalidad, añadir otra nueva, o por ejemplo, adaptar el sistema a un nuevo dominio de aplicación. De hecho, algunos estudios indican que la evolución o mantenimiento del software consume un 80% del presupuesto, mientras que el nuevo desarrollo comprende el 20% restante. Este tipo de desarrollo software tiene sus propias particularidades y los nuevos enfoques que se planteen deben considerar también esta circunstancia.

En esta tesis de máster se presenta el estudio realizado sobre el comportamiento de las metodologías de desarrollo ágil en proyectos de evolución software. Concretamente, se describe el experimento realizado con la metodología ágil SCRUM en la evolución de un producto concreto, TOPENprimer, un entorno software para operar, probar y monitorizar sistemas complejos, especialmente diseñado para facilitar su adaptación a diferentes dominios de aplicación. Como quedará patente a lo largo de la tesis, el estudio lleva a concluir la adecuación de utilizar SCRUM y prácticas de desarrollo ágil en la evolución del software, dadas las actuales características del mercado. Aunque también se presenta la necesidad de introducir determinadas "correcciones" en dichas metodologías de forma que se alivien algunos problemas detectados a lo largo del estudio. A pesar del enorme acogimiento que estas metodologías están teniendo en el mercado, aún se encuentran en una etapa de crecimiento y se concluye la necesidad de realizar más investigaciones que permitan su consolidación, principalmente en el área de evolución de productos e integración continua[1].

[1] Del inglés "continuous integration"

En el resto del capítulo se describe detalladamente la motivación que ha dado pie a la realización de este estudio, así como el objetivo general del mismo. Finalmente, se presenta un recorrido global por el contenido de la tesis.

1.1. Motivación del Estudio

Las metodologías de desarrollo ágil están adquiriendo gran popularidad en los últimos años. La mayoría de las empresas punteras están comenzando a utilizarlas: Google, Yahoo, Symantec, Microsoft, y una amplia lista más de organizaciones siguen el modelo *agile*. Muchos las perciben como la panacea para todos los males de la ingeniería del software. No obstante, la relativa novedad de dichas metodologías hace que aún existan grandes lagunas en el proceso de desarrollo a utilizar siguiendo una filosofía ágil. Esta falta de madurez repercute en que, aunque existen evidencias de los beneficios que este tipo de metodologías proporcionan en proyectos de pequeña envergadura, todavía resulte difícil escalar a grandes proyectos.

En la figura 1.1 se muestra el gráfico en el que Rogers y Moore establecen el ciclo de vida por el que transcurre una innovación hasta asentarse de forma sólida en el mercado y, finalmente, en muchas ocasiones, desaparecer del mismo. Tras un tiempo de estudio de la innovación, ésta pasa a ser adoptada rápidamente por un conjunto relativamente amplio del mercado. En este punto, si logra alcanzar la madurez suficiente se establece de forma sólida en el mercado convirtiéndose en una corriente principal, en caso contrario desaparecerá sin pena ni gloria. Este es el punto de inflexión o barrera en la que se encuentran las metodologías ágiles en la actualidad, en una etapa de maduración. De ahí, el creciente interés por parte de la industria en estudiar estas metodologías que parecen permitir una mejor adaptación a las actuales características de dinamismo y variabilidad de la industria software.

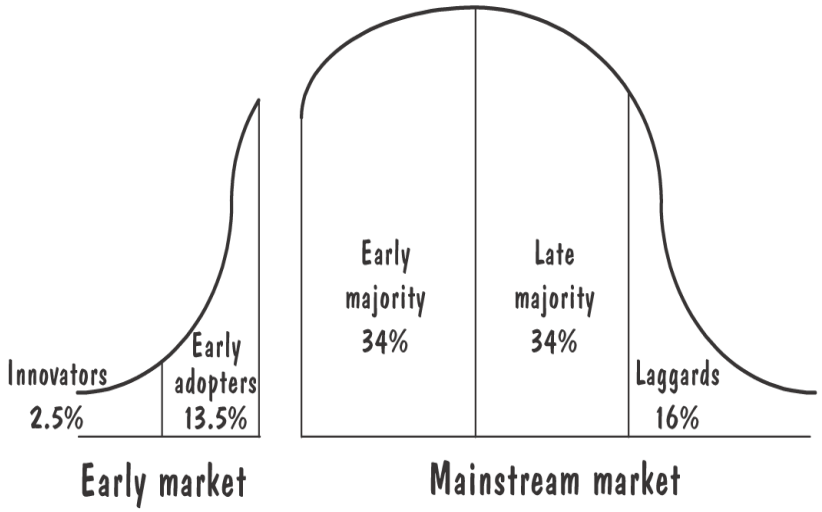


Figura 1.1 Adopción de tipos y el abismo entre las innovaciones y la corriente principal del mercado. Rogers 1995 y Moore 1991[2]

Un reciente estudio, presentado en 2008, dedicado a la inspección del estado de la investigación en metodologías de desarrollo ágil [6] advierte que la mayor parte de las investigaciones que se han realizado hasta el momento relacionadas con este campo del

[2] Figura 14.1 de [Fleeger]

desarrollo software se centran en describir experiencias exitosas que surgen al utilizar dichas metodologías, aunque en ciertas ocasiones no proporcionan información suficiente sobre el contexto en el que se establece la experiencia ni resultados cuantitativos que avalen las observaciones, y solicita la realización de estudios con base empírica, como el que se presenta en esta tesis de máster. Del mismo modo, otros estudios destacan este problema y requieren la ejecución de nuevas investigaciones, muy especialmente en el área de evolución del software e integración continua, hecho que también ha motivado la realización de este trabajo [7] [8].

Por otro lado, el estudio recogido en esta tesis se ha realizado dentro del marco del proyecto ITEA2 Flexi [4], cuya motivación es mejorar la competitividad de la industria de desarrollo software en Europa proporcionando un enfoque flexible, rápido y ágil en el desarrollo del producto. El eslogan de este proyecto define claramente los objetivos a alcanzar por los socios[3] que constituyen el consorcio: *Integración y desarrollo flexible del producto, "De la idea al producto en 6 meses"*. Flexi aplica la metodología ágil SCRUM por lo que ha sido la elegida para el desarrollo de este experimento frente a otras como Crystal, Dynamic Software Development Method o eXtreme Programming. Además, las revisiones del estado de la investigación en *agile* [6] indican que a pesar de que SCRUM es una de las metodologías ágiles más populares en el mercado es una de las menos estudiadas formalmente, circunstancia que abala la elección de SCRUM como metodología de desarrollo.

En línea con la reflexión de Boehm "*mucha disciplina sin agilidad se convierte en burocracia y estancamiento; la agilidad sin disciplina es la incertidumbre entusiasta de la puesta en marcha de una compañía previa a ganancias rápidas. Las grandes compañías tienen ambas cualidades en la medida apropiada en función de sus objetivos y entorno"*, uno de las motivaciones añadidas al experimento es estudiar las metodologías ágiles desde la perspectiva de un grupo de trabajo en el que habitualmente se han utilizado metodologías convencionales en el desarrollo, de tal forma que no se obvien los conocimientos adquiridos por el grupo hasta el momento y se aporte valor al modelo *agile*.

Obviamente para poder realizar este estudio se necesita un proyecto que de base a la investigación. La construcción del producto objeto de este proyecto completa el conjunto de motivaciones para la realización de este experimento. Se trata del desarrollo de un entorno de trabajo que permita monitorizar, operar y probar un sistema de producción de biogás. El hecho de que el cliente de este sistema se haya puesto en

[3] Claro indicador del enorme interés que existe por parte de la industria software en estudiar las metodologías ágiles es el amplio e importante consorcio que forma parte del proyecto Flexi (Belgium: Callatay&Wouters (Large industry) , Freemind (SME) and Sirris (Research) Finland: F-Secure (Large industry), Hantro Products (Large industry), Reaktor Innovations(Large industry), VTT (Research), Nokia Siemens Networks (Large industry), University of Oulu (Research), Elektrobit (Large industry) and Tampere University of Technology (Research). France: Thales (Large industry) and Tiga (SME). Irland: University of Limerick (Research) Israel: BizWise Technologies(Large industry, SME) and Opcat Systems (Large industry, SME). Netherlands: Philips Applied Technologies (Large industry), LaQuSo (University of Eindhoven, Research) and University of Utrecht (Research). Norway: Firm (SME), Objectnet (SME), Kongsberg (SME), Spacetec (SME), Geomatikk AS (SME) and SINTEF (Research) Spain: Technical University of Madrid (UPM), Telefonica (Large industry), DS2 (SME), SQS (SME), Innovalia Association (SME), Indutraux (SME), Answare (SME), IPSA (SME) and European Software Institute (Research) Sweden: ABB (Large industry) and Mardalen University (Research))

contacto con el grupo de investigación SYST (*SYstem and Software Technology group*) en el que se enmarca esta tesis de máster, y haya apostado por utilizar una metodología de desarrollo ágil en la construcción de su producto ha repercutido positivamente y ha añadido valor al experimento pues se trata de satisfacer las necesidades de un cliente real en un proyecto real.

1.2. Objetivo General de la Investigación

De lo anteriormente expuesto podemos concluir el objetivo general que debe servir de guía para la realización de este trabajo. El objetivo general de esta tesis de máster es analizar el mantenimiento o evolución de productos software, en el contexto de utilización de metodologías ágiles. Concretamente, se pretende estudiar la evolución de un producto software concreto, TOPENprimer, especialmente diseñado para ser adaptable a diferentes dominios de aplicación con un coste reducido. Se pretende evolucionar el producto del dominio al que actualmente está siendo aplicado, un sistema de máquinas de juego interconectadas, al dominio de las plantas de producción de biogás, a través de integraciones continuas de funcionalidad. Se presta especial interés al concepto de *continuous integration* y su adopción en las metodologías ágiles, por ser un área de vital importancia en el mercado actual y en el que existen grandes vacíos formales.

Desde una perspectiva industrial, como proyecto dentro del experimento que guía el estudio, el objetivo es obtener un producto software que permita operar, monitorizar y probar el funcionamiento de una planta de producción de biogás concreta, la planta de producción de biogás del cliente del proyecto que da base a este experimento. Aunque desde el punto de vista de la investigación el producto obtenido quede relegado por el estudio de la aplicación de metodologías de desarrollo ágil en proyectos de evolución software, no debemos desestimar el hecho de que se está desarrollando un producto real para un cliente real, circunstancia que proporciona un peso añadido al experimento.

Basados en este objetivo general que dirige la investigación, en la segunda parte de esta tesis de máster, en la que se describe la ejecución del experimento, concretamente en le capítulo 3, se descompondrá este objetivo principal en subobjetivos más concretos perseguidos con la realización del experimento.

1.3. Contenido de la Tesis

El estudio presentado en esta tesis se encuentra estructurado en tres partes, cada una de ella formada por capítulos, tal y como se indica a continuación:

En la primera parte se introduce al lector en el estudio objeto de esta investigación. Comprende dos capítulos.

- El primer capítulo, que corresponde al actual, conforma una breve introducción al trabajo desarrollado y contiene una descripción por un lado del planteamiento general de esta tesis de máster y, por otro, de la motivación y objetivo general perseguido con la realización de la misma.
- En el segundo capítulo se describen las áreas de investigación relacionadas con esta tesis. Así se presentan las metodologías de desarrollo ágil, poniendo especial interés en la metodología SCRUM por ser la utilizada en el desarrollo del experimento, y cuál es el estado actual de la investigación en

este campo. Seguidamente se trata el concepto de evolución en el desarrollo software y como es acogido desde la perspectiva de las metodologías ágiles.

En la segunda parte se describe la ejecución del experimento que da base a este estudio y se analizan los resultados obtenidos. Comprende los capítulos 3 a 7.

- En el tercer capítulo se definen detalladamente cada uno de los objetivos perseguidos con la realización de este estudio. Dicha definición se realiza a través de cuestiones de competencia que serán resueltas a lo largo de la tesis. Seguidamente, presenta un resumen inicial del desarrollo del experimento con el objetivo de introducir al lector en la ejecución del mismo, de tal forma que adquiera una visión global del proyecto.
- En el cuarto capítulo se presenta una detallada descripción del contexto en el que se realiza la investigación. Se trata de un capítulo clave para que otros investigadores puedan utilizar los resultados obtenidos, permitiendo la replicación del estudio, comparación o generalización de las conclusiones. Por este motivo adquiere una especial atención describiendo detalladamente aspectos como el producto de partida objeto de la evolución y el producto que se pretende obtener como proyecto dentro del experimento, el proceso metodológico concreto, adaptado a la investigación, que se utiliza en este estudio, factores sociológicos, ergonómicos, geográficos y tecnológicos.
- El quinto capítulo presenta el diseño del experimento de forma que se estudian las medidas de interés que impactan en la ejecución del mismo, así como el programa de ejecución. Se definen el plan de ejecución y el diseño del plan de mediciones utilizado. Además, se presta especial atención en describir las amenazas que pudieran poner en peligro la validación del experimento y cómo han sido controladas.
- La exposición en el capítulo sexto está dividida en ocho partes que relatan la ejecución del experimento. La primera parte recoge los hechos más destacables durante los tres primeros sprints[4] del experimento dedicados al diseño del mismo. Las siete partes restantes corresponden a la descripción del resto de sprints que constituyen el experimento en sí.
- En el séptimo y último capítulo de esta parte se presentan los resultados obtenidos en la ejecución del experimento, junto a un minucioso análisis de los mismos.

La tercera y última parte de la investigación constituye un único capítulo, el capítulo octavo, en el que se presentan las conclusiones generales de la tesis, a modo de resumen de los logros alcanzados, así como las contribuciones realizadas y las previsibles líneas futuras de estudio.

[4] En las metodologías ágiles el desarrollo de sistemas software es dividido en ciclos de corta duración, de no más de treinta días. A cada uno de estos ciclos se le denomina sprint.

Capítulo 2

Estado del Arte

Esta investigación está enmarcada en dos áreas principales: *el modelo ágil de desarrollo software* y *la evolución del producto software*. A continuación, se describe en detalle la situación actual de la investigación en cada una de estas áreas.

2.1. Modelo Ágil de Desarrollo Software

Las metodologías ágiles son sin duda uno de los temas recientes en ingeniería del software que están acaparando gran interés y controversia. A mediados de los años 90 comenzó a forjarse una definición moderna de desarrollo ágil del software como una reacción contra las metodologías utilizadas hasta el momento, consideradas excesivamente *pesadas* y rígidas por su carácter normativo y fuerte dependencia de planificaciones detalladas previas al desarrollo. En el año 2001 diecisiete miembros destacados de la comunidad software[5], incluyendo algunos de los creadores o impulsores de las metodologías en software, se reunieron en Utah (Estados Unidos) y adoptaron el nombre de "Metodologías ágiles" para denominar a esta nueva corriente de desarrollo. Poco después, algunos de estos miembros formaron la conocida como "Alianza ágil" [9], una organización sin ánimo de lucro que promueve el desarrollo ágil de aplicaciones. Desde ese momento hasta la actualidad las metodologías ágiles han ido adquiriendo gran auge dentro de la industria software y las organizaciones más punteras comienzan a apostar por este nuevo enfoque para desarrollar sus productos.

[5] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland y Dave Thomas.

El objetivo de esta sección es doble. Por un lado, describir detalladamente por qué aparece la necesidad de incorporar las metodologías ágiles al desarrollo software, en qué consiste exactamente este tipo de desarrollo y cuáles son las metodologías ágiles más destacadas en la actualidad, prestando especial atención a SCRUM por ser la metodología a seguir en esta investigación. Por otro lado, se pretende dar cuenta de cuál es el estado actual de la investigación en este campo y cuáles son los motivos que justifican la necesidad de realizar la investigación presentada en esta tesis de máster.

2.1.1. Introducción al modelo ágil. El por qué de las metodologías ágiles

Tradicionalmente se ha tendido a desarrollar software a través de metodologías que encorsetaban el proceso de desarrollo de manera un tanto rígida que, cada vez más, se demuestra errónea en las actuales características de dinamismo y variabilidad del mercado software. Como indica Boehm en la referencia [1] se tiende hacia el rápido desarrollo de aplicaciones y la vida de los productos se acorta. En este entorno inestable, que tiene como factor inherente el cambio y la evolución rápida y continua, la ventaja competitiva se encuentra en aumentar la productividad y satisfacer las variantes necesidades del cliente en el menor tiempo posible para proporcionar un mayor valor al negocio.

Sin embargo, las metodologías convencionales presentan diversos problemas a la hora de abordar un amplio rango de proyectos industriales en este turbulento entorno. Entre estos problemas podemos destacar los siguientes:

Perciben la captura de requisitos del proyecto como una fase previa al desarrollo del mismo que, una vez completada, debe proporcionar una fotografía exacta de qué desea el cliente. Se trata de evitar a toda costa que se produzcan cambios en el conjunto de requisitos inicial, puesto que a medida que avanza el proyecto resulta más costoso solucionar los errores detectados o introducir modificaciones [10], y pretenden delegar toda responsabilidad económica en el cliente en caso de que estos cambios de requisitos se produzcan. Por este motivo, se les conoce también como metodologías *predictivas*. Sin embargo, el esfuerzo, tanto en coste como en tiempo, que supone hacer una captura detallada de todos los requisitos de un proyecto al comienzo del mismo es enorme y rara vez se ve justificado con el resultado obtenido. Además, en muchas ocasiones el cliente no conoce sus propias necesidades con la profundidad suficiente como para definirlas de forma exacta a priori y, a menudo, estas necesidades y sus prioridades varían durante la vida del proyecto. Establecer mecanismos de control es una de las opciones existentes para protegerse de estos cambios, aunque frecuentemente provocan la insatisfacción de los clientes, que perciben el desarrollo del proyecto como algo inflexible que no se adapta a sus necesidades y que si lo hace repercute negativamente en costes añadidos al presupuesto del proyecto.

Por otro lado, en muchas ocasiones el proceso de desarrollo convencional está oprimido por excesiva documentación no siempre útil. Un porcentaje elevado del tiempo de desarrollo de un producto software se dedica o, desde el punto de vista de las metodologías ágiles, se malgasta en crear documentación que finalmente no se utiliza y que, por tanto, no aporta valor al negocio. Además, esta documentación innecesaria entorpece las labores de mantenimiento de la propia documentación útil lo que provoca que en muchas ocasiones el mantenimiento de la documentación se obvie agudizando,

de este modo, el coste en las tareas de documentación futuras. Evidentemente, estas circunstancias no se adaptan a las restricciones de tiempo del mercado actual.

Otra dificultad añadida al uso de metodologías convencionales es la lentitud del proceso de desarrollo. Es difícil para los desarrolladores entender un sistema complejo en su globalidad lo que provoca que las diferentes etapas del ciclo de vida convencional transcurran lentamente. Dividir el trabajo en módulos abordables ayuda a minimizar los fallos y, por tanto, el coste de desarrollo. Además, permite liberar funcionalidad progresivamente, según indiquen los estudios de las necesidades del mercado que aportan mayor beneficio a la organización. En la feroz competencia del mercado vigente, en la que los productos quedan obsoletos rápidamente, se pide básicamente rapidez, calidad y reducción de costes, pero para asumir estos retos, es necesario tener agilidad y flexibilidad.

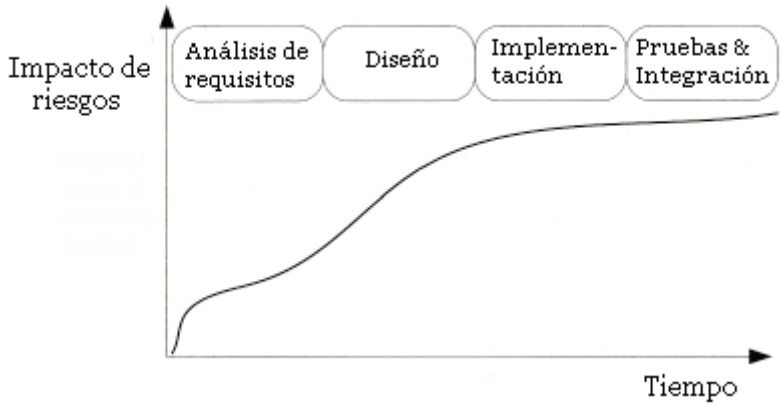


Figura 2.1 Distribución del riesgo en un desarrollo en cascada

Asimismo, las metodologías convencionales tienden a acumular los riesgos y dificultades que surgen en el desarrollo del producto al final del proyecto, como puede apreciarse en la figura 2.1, repercutiendo en retrasos en la entrega de productos o influyendo en la incorrecta ejecución de las últimas fases del ciclo de vida.

En contraposición a las metodologías convencionales, las metodologías ágiles aparecen como alternativa atractiva para adaptarse a este entorno. Tal y como se indica en [11] y [12], son apropiadas cuando los requisitos son emergentes y cambian rápidamente. De este modo, presentan diversas ventajas en el contexto actual:

- Capacidad de respuesta a cambios a lo largo del desarrollo ya que no los perciben como un lastre sino como una oportunidad para mejorar el sistema e incrementar la satisfacción del cliente, considerando la gestión de cambios como un aspecto característico del propio proceso de desarrollo software.
- Entrega continúa y en plazos breves de software funcional lo que permite al cliente verificar *in situ* el desarrollo del proyecto, ir disfrutando de la funcionalidad del producto progresivamente y comprobando si satisface sus necesidades, mejorando de esta forma su satisfacción. Además, el desarrollo en ciclos de corta duración favorece que los riesgos y dificultades se repartan a lo largo del desarrollo del producto, principalmente al comienzo del mismo y permite ir aprendiendo de estos riesgos y dificultades (ver figura 2.2).

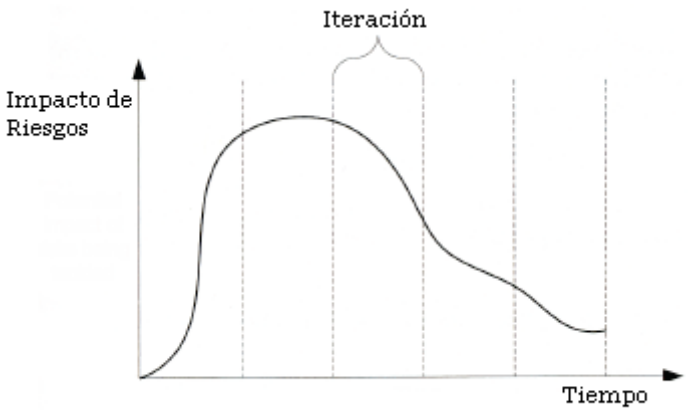


Figura 2.2 Distribución del riesgo en un desarrollo ágil

- Trabajo conjunto entre el cliente y el equipo de desarrollo con una comunicación directa que pretende mitigar malentendidos, que constituyen una de las principales fuentes de errores en productos software, y exceso de documentación improductiva.
- Importancia de la simplicidad, eliminando el trabajo innecesario que no aporta valor al negocio.
- Atención continúa a la excelencia técnica y al buen diseño para mantener una alta calidad de los productos.
- Mejora continua de los procesos y el equipo de desarrollo, entendiendo que el éxito, tal y como se indica en la figura 2.3, depende de tres factores: éxito técnico, éxito personal y éxito organizacional.

Figura 2.3 Tipos de éxito[6]

En el siguiente apartado, se profundizará en los valores y principios que sustentan las metodologías ágiles y los beneficios que estos valores aportan al desarrollo software.

2.1.2. El Manifiesto Ágil. Valores y principios

Continuamente estamos utilizando el concepto de agilidad para definir estas metodologías, pero ¿qué significa ser ágil desde una perspectiva software? Basados en el consenso de varias definiciones contemporáneas, Qumer y Henderson-Sellers ofrecieron la siguiente definición de agilidad [13]:

[6] Figura basada en la figura 1-1 de [Shore]

> *"La agilidad es un comportamiento persistente o habilidad, de entidad sensible, que presenta flexibilidad para adaptarse a cambios, esperados o inesperados, rápidamente; persigue la duración más corta en tiempo; usa instrumentos económicos, simples y de calidad en un ambiente dinámico; y utiliza los conocimientos y experiencia previos para aprender tanto del entorno interno como del externo."*

Por tanto, el desarrollo ágil no especifica unos procesos o métodos que seguir, aunque bien es cierto que han aparecido algunas prácticas asociadas a este movimiento. El desarrollo ágil es más bien una filosofía de desarrollo software. El punto de partida se establece en las ideas emanadas del Manifiesto Ágil tras la reunión de Utah [14], un documento que resume la filosofía *agile* estableciendo cuatro valores y doce principios.

Figura 2.4 Manifiesto para el desarrollo ágil del software

Según el Manifiesto se valora:

Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. La gente es el principal factor de éxito de un proceso software. Este primer valor expresa que es preferible utilizar un proceso indocumentado con buenas interacciones personales que un proceso documentado con interacciones hostiles. Se considera que no se debe pretender construir primero el entorno y esperar que el equipo se adapte automáticamente sino al revés, construir primero el equipo y que éste configure su propio entorno. El talento, la habilidad, la capacidad de comunicación y de tratar con personas son características fundamentales para los miembros de un equipo ágil.

Desarrollar software que funcione por encima de una completa documentación. Este valor es utilizado por muchos detractores de las metodologías ágiles que argumentan que éstas son la excusa perfecta para aquellos que pretenden evitar las tareas menos gratificantes del desarrollo software como las tareas de documentación. Sin embargo, el propósito de este valor es acentuar la supremacía del producto por encima de la documentación. El objetivo de todo desarrollador es obtener un producto que funcione y cumpla las necesidades del cliente y la documentación es un artefacto que utiliza para cumplir su objetivo. Por tanto, no se trata de no documentar sino de documentar aquello que sea necesario para tomar de forma inmediata una decisión importante. Los documentos deben ser cortos y centrarse en lo fundamental. Dado que el código es el valor principal que se obtiene del desarrollo se enfatiza en seguir ciertos estándares de programación para mantener el código legible y documentado.

La colaboración con el cliente por encima de la negociación contractual. Se propone una interacción continua entre el cliente y el equipo de desarrollo de tal forma que el cliente forme un tándem con el equipo de desarrollo. Se pretende no diferenciar entre las figuras cliente y equipo de desarrollo sino que se apuesta por un solo equipo persiguiendo un objetivo común.

Responder a los cambios más que seguir estrictamente un plan. Planificar el trabajo a realizar es muy útil y las metodologías ágiles consideran actividades específicas de planificación a corto plazo. No obstante, adaptarse a los cambios es vital en la industria software actual y, por tanto, también consideran mecanismos para tratar los cambios de prioridades. La regla es "planificar es útil. Seguir un plan es útil hasta que el plan se distancia de la situación actual. Pender de un plan desactualizado es caminar por un callejón sin salida".

Para cumplir estos valores se siguen doce principios que establecen algunas diferencias entre un desarrollo ágil y uno convencional:

1. La prioridad es satisfacer al cliente mediantes tempranas y continuas entregas de software que le aporten valor.
2. Dar la bienvenida a los cambios de requisitos. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
3. Liberar software que funcione frecuentemente, desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
4. Los miembros del negocio y los desarrolladores deben trabajar juntos diariamente a lo largo del proyecto.
5. Construir el proyecto en torno a individuos motivados. Darles el entorno y apoyo que necesiten y confiar a en ellos para conseguir finalizar el trabajo.
6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
7. El software que funciona es la principal medida de progreso.
8. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
10. La simplicidad es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos que se auto organizan.
12. En intervalos regulares el equipo debe reflexionar sobre cómo ser más efectivo y según estas reflexiones ajustar su comportamiento.

Estos principios marcan el ciclo de vida de un desarrollo ágil, así como las prácticas y procesos a utilizar.

2.1.3. Cómo ser ágil. Algunas prácticas

Aunque el movimiento ágil está sustentado por valores y principios para el desarrollo de productos software, la mayoría de estas metodologías tienen asociadas un conjunto de prácticas, en muchos casos comunes, que buscan la agilidad en el desarrollo. En esta sección vamos a analizar algunas de las más relevantes: planificación, programación en parejas o *pair programming*, integración continua, refactorización y desarrollo dirigido por pruebas (*Test Driven Development*).

Planificación, Historias de usuario.

Mientras que las metodologías convencionales centran la ingeniería de requisitos en habilidades de *predicción* basándose en férreas planificaciones previas al desarrollo, las metodologías ágiles perciben la gestión de cambios como un aspecto inherente al proceso de desarrollo software, considerando planificaciones continuas, más flexibles y en cortos plazos, que respondan mejor a los cambios en las necesidades del cliente.

Con el objetivo de disminuir el coste, no siempre amortizable, que supone la etapa de planificación en las metodologías convencionales, las ágiles simplifican las tareas de gestión y documentación de las necesidades del sistema. Apuestan por involucrar de una forma más intensa al cliente a lo largo de todo el proceso de desarrollo, de forma que, la comunicación directa y frecuente retroalimentación son las prácticas más importantes para la planificación y especificación de requisitos en estas metodologías [15].

En este proceso de definición de las necesidades del sistema que guiará el desarrollo, se utilizan las llamadas historias de usuario para detallar, en un lenguaje cercano al cliente, la funcionalidad que debe satisfacer el sistema. Las historias de usuario se descomponen en tareas que deberán ser realizadas para cumplir con la funcionalidad que se solicita. Nótese que la estimación de tiempo no es una restricción fija, simplemente constituye una aproximación inicial. Se considera que las historias de usuario deben cumplir seis características: independientes, negociables, valorables, estimables, pequeñas y comprobables.

Programación en pareja (*Pair Programming*).

Una de las prácticas más utilizadas en metodologías ágiles, sobre todo en sistemas de alta complejidad, es la programación en parejas, que establece que la producción de código se realice con trabajo en parejas de programadores. El utilizar *Pair Programming* en un proyecto no implica que todas las líneas de código sean escritas por una pareja, ya que mientras una persona está escribiendo el código, la otra puede estar verificando errores, pensando en alternativas mejores, identificando casos de prueba, pensando en aspectos de diseño, etc. El objetivo principal de esta técnica es disminuir la tasa de errores, mejorar el diseño y aspectos como la satisfacción de los programadores. No obstante, algunos estudios como [16] y [17] indican que se trata de una técnica intensa y estresante para los propios programadores, aunque admiten que acelera el proceso de desarrollo.

Otro detalle a tener en cuenta es las habilidades y capacidades de los miembros de la pareja. Si existe una diferencia importante entre ellos, puede llegar a ser una práctica frustrante para ambos. En este caso concreto, el Pair Programming ha de ser visto como una técnica de aprendizaje.

Integración continua (*Countinuous integration*)

La integración continua pretende mitigar la complejidad que el proceso de integración suele tener asociada en las metodologías convencionales, superando, en determinadas circunstancias, la propia complejidad de la codificación. El objetivo es integrar cada cambio introducido, de tal forma que pequeñas piezas de código sean integradas de forma continua, ya que se parte de la base de que cuanto más se espere para integrar más costoso e impredecible se vuelve el proceso. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.

Para que esta práctica sea viable es imprescindible disponer de una batería de test, preferiblemente automatizados, de tal forma, que una vez que el nuevo código está integrado con el resto del sistema se ejecute toda la batería de pruebas. Si son pasados todos los test del sistema, se procede a la siguiente tarea. En caso contrario, aunque no falle el nuevo modulo que se quiere introducir en el sistema, se ha de regresar a la versión anterior, pues ésta ya había pasado la batería de pruebas.

Refactorización.

Actividad constante en los desarrollos ágiles cuyo objetivo es mejorar el diseño de un sistema sin influir en su funcionalidad. Se pretende reestructurar el código para eliminar duplicaciones, mejorar su legibilidad, simplificarlo y hacerlo más flexible de forma que se faciliten los posteriores cambios. Recordemos que el código que funciona es el principal aporte de valor para las metodologías ágiles, por encima de la documentación, por lo que se enfatiza en que éste sea legible y permita la comunicación entre desarrolladores.

Desarrollo dirigido por pruebas (Test Driven Development).

Al utilizar esta práctica la producción de código está dirigida por las pruebas. Concretamente, se trata de escribir las pruebas que validarán el sistema antes de comenzar a construirlo, idealmente que sea el propio cliente quien las escriba. De esta forma, ante cada modificación del sistema la batería completa de pruebas es ejecutada. Esta práctica constituye, por tanto, la primera línea para garantizar la calidad del producto, pues ésta se considera en términos de la correcta funcionalidad que se le ofrece al cliente. Entre las ventajas que aporta el desarrollo dirigido por pruebas podemos destacar que disminuye el tiempo dedicado a solucionar errores y genera un sentimiento de confianza de éxito del proyecto en el equipo. Tecnologías como *JUnit* son utilizadas para manejar y ejecutar conjuntos de pruebas automatizadas.

No obstante, el hecho de que aspectos tan importantes como la cohesión o el acoplamiento de los módulos pase a un segundo plano en el desarrollo a favor de la funcionalidad, implica la necesidad de realizar constantes etapas de refactorización que permitan mantener la calidad del código.

Como se puede apreciar, a pesar de la reciente aplicación de las metodologías ágiles la mayoría de las prácticas propuestas no so novedosas sino que de alguna manera ya habían sido propuestas en ingeniería del software. El mérito de las metodologías ágiles es proponer una aplicación conjunta, equilibrada y efectiva de forma que se complementen con ideas desde la perspectiva del negocio, los valores humanos y el trabajo [18].

2.1.4. SCRUM

SCRUM es el término que describe una forma para desarrollar productos iniciada en Japón. No se trata de un concepto nuevo, sino que ya en 1987 Ikujiro Nonaka y Hirotaka Takeuchi [19] acuñaron este término, una estrategia utilizada en rugby en la que todos los integrantes del equipo actúan juntos para avanzar la pelota y ganar el partido, para denominar un nuevo tipo de proceso de desarrollo de productos. Escogieron este nombre por las similitudes que consideraban que existían entre el juego del rugby y el tipo de proceso que proponían: adaptable, rápido, auto-organizable y con pocos descansos.

SCRUM es un proceso para la gestión y control del producto que trata de eliminar la complejidad en estas áreas para centrarse en la construcción de software que satisfaga las necesidades del negocio. Es simple y escalable, ya que no establece prácticas de ingeniería del software sino que se aplica o combina, fácilmente, con otras prácticas ingenieriles, metodologías de desarrollo o estándares ya existentes en la organización.

SCRUM se concentra, principalmente, a nivel de las personas y equipo de desarrollo que construye el producto. Su objetivo es que los miembros del equipo trabajen juntos y de forma eficiente obteniendo productos complejos y sofisticados. Podríamos entender SCRUM como un tipo de ingeniería social que pretende conseguir la satisfacción de todos los que participan en el desarrollo, fomentando la cooperación a través de la auto-organización. De esta forma se favorece la franqueza entre el equipo y la visibilidad del producto. Pretende que no haya problemas ocultos, asuntos u obstáculos que puedan poner en peligro el proyecto. Los equipos se guían por su conocimiento y experiencia más que por planes de proyecto formalmente definidos. La planificación detallada se realiza sobre cortos espacios de tiempo lo que permite una constante retroalimentación que proporciona inspecciones simples y un ciclo de vida adaptable. Así, el desarrollo de productos se produce de forma incremental y con un control empírico del proceso que permite la mejora continua [20].

La figura 2.5 muestra el ciclo de vida del desarrollo que propone SCRUM para un producto software.

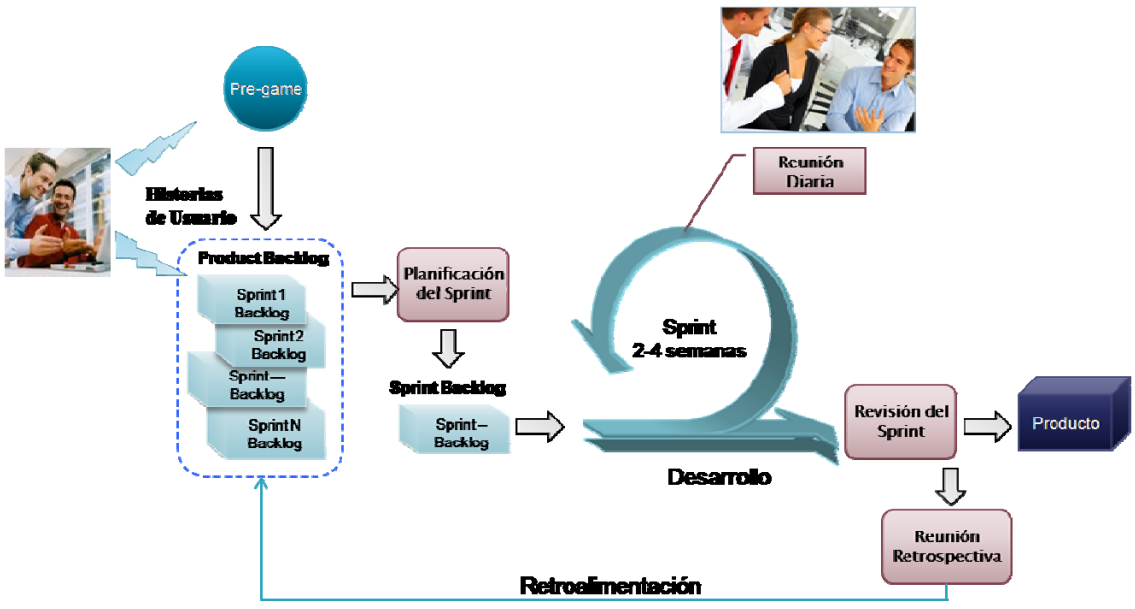


Figura 2.5 Modelo de desarrollo aplicando SCRUM

Independientemente del tipo de metodología que se utilice, cualquier desarrollo software parte siempre de un mismo problema: conocer las necesidades de los clientes. SCRUM, al igual que el resto de metodologías ágiles, pretende no centrar las tareas de desarrollo en un conjunto de requisitos formalmente definidos sino que aboga por la incorporación del cliente como un miembro más del equipo de desarrollo. De este modo, no se considera el proceso de definición de requisitos como un fin dentro del desarrollo del proyecto, sino que los requisitos aparecen implícitamente dentro del contenido de las denominadas historias de usuario.

Las historias de usuario son el elemento base que utiliza SCRUM para describir las características que el usuario espera que tenga el software que se va a desarrollar [21]. Por lo tanto, pueden incorporar tanto cuestiones relacionadas con las funciones del sistema como con cualquier otro aspecto del mismo (restricciones, rendimiento, etc.). Las historias de usuario se presentan desde la perspectiva del usuario. Así, no se describen utilizando una terminología técnica sino que se escriben utilizando un lenguaje cercano al dominio de la aplicación que se está desarrollando de forma que sea comprensible por los clientes y por los desarrolladores. Las historias de usuario se construyen bajo un mismo esqueleto que centra el foco de las características del producto.

- Primero se determina *quién* propone la historia de usuario,
- luego se describe la *característica* que se cubre con la historia de usuario y
- finalmente se especifica la *razón* por la que dicha característica es necesaria.

El proceso comienza con la fase de Pre-game, en la que se realiza de forma conjunta con el cliente una definición sencilla y clara de las características que debe tener el sistema que vaya a ser desarrollado, definiendo las historias de usuario que van a guiar el proceso de desarrollo. Posteriormente, cada historia de usuario se refina de manera que se identifican las tareas que son necesarias para llevar a cabo el desarrollo de la historia de usuario. En un principio no es necesario detallar de forma completa todas las historias de usuario sino sólo las que tienen un mayor nivel de prioridad por parte del usuario. Esto permite que el proceso de desarrollo pueda adaptarse a posteriores modificaciones de las necesidades de usuario, de forma que el proceso de desarrollo se vuelve más flexible. El resultado de esta fase de Pre-game es lo que se denomina en SCRUM "*Product Backlog*", que contiene una lista de todas las historias de usuario priorizadas así como de las tareas que se deben llevar a cabo para la realización del proyecto. Este enfoque basado en quién, qué y por qué facilita la tarea de priorización de las historias de usuario, lo que permite realizar la planificación inicial del proyecto. Como puede deducirse de lo anteriormente expuesto, la obtención del Product Backlog se realiza con la ayuda del cliente/usuario del sistema que se va a desarrollar, por lo que puede considerarse que se realiza en directo y, por lo tanto, las posibles dudas en el establecimiento de las historias de usuario pude resolverse en ese mismo instante.

Una vez identificadas y priorizadas las historias de usuario del Product Backlog, se realiza la separación de historias de usuario en etapas de corta duración (no más de 30 días) denominadas sprints. Para cada sprint se realiza una reunión de planificación de lo que se va a llevar a cabo en ese sprint y se establece la fecha de finalización del mismo. El objetivo es mover aquellas historias de usuario con mayor prioridad para el cliente al denominado "Sprint Backlog", que contiene el conjunto de tareas a desarrollar en ese sprint, incluyendo diseño, desarrollo, pruebas, integración, etc. Las historias de usuario

se congelan en el Sprint Backlog de forma que durante dicho periodo no puedan producirse cambios sobre los aspectos que se encuentran en fase de desarrollo.

De forma iterativa, todos los días que dure el sprint, se realiza una reunión operativa, informal y ágil con el equipo de desarrollo, de un máximo de quince minutos, en la que a cada integrante del equipo se le hacen tres preguntas:

- ¿Qué tareas ha hecho desde la última reunión? Es decir, tareas realizadas en un día.
- ¿Qué tareas va ha hacer hoy?
- ¿Qué ayuda necesita para poder realizar este trabajo? Es decir, identificación de obstáculos o riesgos que impiden o pueden impedir el normal avance.

Una vez finalizado el sprint se debería obtener parte del producto, un entregable o algo que se pueda mostrar y que enseñe los avances acometidos en el Sprint. En este punto se procede a una fase de revisión del proyecto para mostrar dichos avances e incorporar, si fuera necesario, nuevas historias de usuario al Product Backlog. Para mantener la calidad del producto se establece que una historia de usuario está 100% completa si supera los test unitarios, pasa las pruebas de aceptación, supera los test adicionales para mantener la calidad del producto, el código está construido e integrado satisfactoriamente, está basado en un diseño factorizado sin duplicaciones, el código es claro, estructurado y autodocumentado y, por supuesto, es aceptado por el cliente.

Además, tras la realización de cada sprint se lleva a cabo una reunión retrospectiva que permite aprender de los conocimientos y experiencias adquiridas hasta el momento y mejorar de forma continua el proceso de desarrollo. Se revisará con el equipo los objetivos marcados inicialmente en el Sprint Backlog concluido, se aplicarán los cambios y ajustes si son necesarios, y se marcarán los aspectos positivos (para repetirlos) y los aspectos negativos (para evitar que se repitan) del Sprint, que servirán de retroalimentación para el siguiente sprint. Los elementos que servirán de entrada y guiarán la planificación de un nuevo sprint serán el Product Backlog, las capacidades o habilidades del equipo, las condiciones que en ese momento se den del negocio, el avance tecnológico que se haya podido producir y el incremento que se pretenda hacer al producto que se está desarrollando. De esta forma, en contraposición con las metodologías convencionales, el proceso de desarrollo adquiere flexibilidad, agilidad y capacidad para adaptarse a su entorno.

Para orquestar este proceso SCRUM distingue distintos actores con diferentes papeles dentro del proceso. De forma general, podemos distinguir propietario del producto ó *Product Owner*, maestro de scrum ó *Scrum Master*, equipo de desarrollo ó *Scrum Team* y *cliente* o usuario.

El Product Owner es la única persona encargada de la dirección y control del Product Backlog, es decir, de las historias de usuario que debe cumplir el sistema. Se trata de una persona física (solamente una persona para eliminar las posibles confusiones o interferencias), no una organización o comité. Bien puede ser el propio cliente *in situ* en el lugar de desarrollo u otra persona que tenga el conocimiento suficiente sobre el producto o pueda estar en continuo contacto con el cliente para marcar las prioridades del proyecto. Es, por tanto, la persona oficialmente responsable del proyecto que de forma visible, vocal y objetiva debe tomar todas las decisiones de negocio para el producto. Para que el Product Owner tenga éxito, el resto del equipo de la organización tiene que respetar sus decisiones. En cuanto a su implicación debe estar en contaste interacción con el equipo de desarrollo. Debe asistir, al menos, a las

reuniones de planificación y de revisión de cada sprint y estar en continuo contacto con el equipo para proporcionar detalles sobre las historias de usuario y constante retroalimentación que dirija el desarrollo del sprint.

El Scrum Master es la persona responsable del éxito al aplicar la metodología SCRUM en el desarrollo del proyecto o producto, asegurando que los valores, prácticas y reglas son seguidos por el resto del equipo. En concreto, es la persona que asegura el seguimiento de la metodología guiando las reuniones, ayudando al equipo ante cualquier problema que pueda aparecer y controlando que el trabajo sigua el ritmo adecuado. Por tanto debe tomar decisiones inmediatas y eliminar los impedimentos que vayan surgiendo en el momento, aunque en ocasiones no cuente con toda la información necesaria. Su responsabilidad es entre otras, la de hacer de paraguas ante las presiones externas y motivar al resto del equipo. Por tanto, la labor de Scrum Master requiere una fuerte personalidad ya que debe facilitar el trabajo del equipo sin imponer autoridad. Las personas que ejercen este role deben ser capaces de hacer cualquier cosa por ayudar al equipo de desarrollo, incluso aunque estas acciones estén enfrentadas con sus propios intereses. Deben ser personas poco conformistas y con mucha iniciativa. Eliminar impedimentos requiere determinación y tenacidad. La metáfora utilizada por Kent Beck refleja perfectamente este papel: *"El Scrum Master es el perro pastor que hace cualquier cosa para proteger al rebaño, y nunca se distrae de su deber"*.

El Scrum Team lo conforman las personas responsables de implementar la funcionalidad o funcionalidades elegidas por el Product Owner. Debe ser un conjunto de personas motivadas con habilidades y capacidades complementarias que estén comprometidos por un propósito común, cumplir el objetivo del sprint. El equipo tiene plena autoridad para tomar todas las decisiones que consideren adecuadas en el desarrollo del proyecto, auto-organizándose y auto-disciplinándose. Así, por ejemplo, en las reuniones de planificación el Product Owner y el Scrum Team deben llegar a un acuerdo realista sobre las historias de usuario que se van a completar en el siguiente sprint y si en algún momento el Scrum Team considera que algunas de las prioridades del Product Owner no es razonable dispone de libertad absoluta para reseñar esta circunstancia y obligar al propietario del producto a variar sus prioridades. Teóricamente, se estima que debería estar formado por un número de entre 7 u 8 miembros como máximo y 2 como mínimo.

Finalmente, el cliente es o son los beneficiarios finales del producto, y quienes viendo los progresos, pueden aportar ideas, sugerencias o necesidades. Su participación es importantísima e imprescindible en esta metodología.

La efectividad de la metodología para la gestión de proyectos se basa en un conjunto de valores fundamentales que deben seguir todos los integrantes del equipo, principios sobre los que reposan el resto de prácticas: compromiso, esmero, franqueza, respeto y valor. Los miembros del equipo deben estar dispuestos a comprometerse con el objetivo de cada sprint y del proyecto en general. SCRUM proporciona al equipo toda la autoridad que necesiten para obtener a cambio su compromiso. El equipo se tiene que comprometer a hacer su trabajo. Cada miembro debe concentrar todos sus esfuerzos y habilidades en cumplir con el trabajo que se han comprometido a realizar sin desviarse en otros aspectos, realizando todas sus labores con esmero. Todos los aspectos del proyecto son visibles para todo el equipo por lo que un valor fundamental es también la franqueza. Además, los miembros del equipo están avalados por sus conocimientos y experiencias, el respeto es un valor fundamental con cada uno de ellos. Finalmente, cada

miembro del equipo tiene que tener el coraje suficiente para comprometerse, actuar, ser transparente en el desarrollo y respetar y hacer que le respeten.

La figura 2.6 refleja, a modo de resumen, la cultura SCRUM frente a la cultura del convencional ciclo de vida en cascada

Figura 2.6 Dos culturas, desarrollo convencional vs. SCRUM

2.1.5. Una revisión de otras metodologías ágiles

Aunque el manifiesto ágil es la piedra angular sobre la que cimientan todas las metodologías ágiles, cada una tiene unas características propias y hace hincapié en algunos aspectos más específicos. En esta sección se describen, a grandes rasgos, las particularidades fundamentales de algunas otras metodologías ágiles, que actualmente tienen gran aceptación en el mercado, como eXtreme Programming, Cristal, Dynamic Software Development Method (DSDM), Feature-driven Development y Lean Software Development.

eXtreme Programming [22, 23], también conocida como XP, es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo software. Aspectos como el trabajo en equipo, el aprendizaje de los desarrolladores y propiciar un buen clima de trabajo son pilares en esta metodología. Oficialmente fue creada en 1999 por Kent Beck, con la publicación de su libro *Extreme Programming Explained*.

XP se centra en la continua retroalimentación entre el cliente y el equipo de desarrollo, la comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define como una metodología especialmente adecuada para proyectos con requisitos muy cambiantes e imprecisos, donde existe un alto riesgo técnico. Como metodología pragmática, recoge las que considera mejores prácticas para el desarrollo software, cuya aplicación disciplinada pretende disminuir la curva exponencial del costo del cambio a lo largo del proyecto. Se trata de doce prácticas: el juego de la planificación, entregas pequeñas, metáfora, diseño simple, pruebas, refactorización, programación en parejas, propiedad colectiva del código, integración continua, 40 horas por semana, cliente in-situ y estándares de programación.

Una posterior revisión de la metodología clasificó las prácticas en primarias, aquellas que según Beck proporcionan una mejora inmediata independientemente de la metodología que se esté siguiendo, y secundarias, que implican una mayor dificultad si no se tiene experiencia en las prácticas primarias. Siguiendo esta clasificación, las prácticas primarias consideradas fueron la adecuación del lugar de trabajo, sentarse juntos, entorno de trabajo informativo, sentimiento de equipo, trabajo enérgico, programación en parejas, user stories, iteraciones cortas, integración continua, pruebas primero, diseño incremental y refactorización. El conjunto de prácticas secundarias quedó compuesto por la involucración del cliente, continuidad del equipo, equipos retractiles, código compartido y alcance del contrato negociado.

Crystal Methodologies [24, 25] es un conjunto de metodologías ágiles para equipos de diferentes tamaños y con distintas características de criticidad. Fue propulsada por uno de los padres del Manifiesto Agil, Alistair Cockburn, que consideraba que la metodología debe adaptarse a las personas que componen el equipo utilizando políticas diferentes para equipos diferentes. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores: Crystal Clear (3 a 8 miembros), Crystal Yellow (10 a 20 miembors), Crystal Orange (25 a 50 miembros), Crystal Red (50 a 100 miembros) y Crystal Blue (para más de 100 miembros). Por ejemplo, Crystal Clear, la metodología más ligera de este conjunto, esta dirigida a la comunicación de equipos pequeños que desarrollan software cuya criticidad no es elevada. Tiene asociadas siete características: liberación frecuente de funcionalidad, mejora reflexiva, comunicación osmótica, seguridad personal, atención, fácil acceso para usuario expertos y requisitos para el entorno técnico.

Dynamic Software Development Method (DSDM) [26] puede considerarse un marco para el proceso de producción de software, más que una metodología. Nació en 1994 con el objetivo de crear una metodología RAD (*Rapid Application Development*) unificada. Divide el proyecto en tres fases: pre-proyecto, ciclo de vida del proyecto y post-proyecto especificando de forma rigurosa la arquitectura y gestión del proyecto. Así, propone cinco fases en el desarrollo del proyecto: estudio de la viabilidad y estudio del negocio que constituyen la etapa de pre-proyecto; y, de forma iterativa, modelado funcional, diseño y construcción y finalmente implementación, además de una adecuada retroalimentación a todas las fases. Sus principales características son interacción con el usuario, poder del equipo de desarrollo, liberaciones frecuentes de funcionalidad, regirse siguiendo las necesidades del negocio, desarrollo iterativo e incremental, adaptación a los cambios reversibles, fijar el nivel de alcance al comienzo del proyecto, pruebas durante todo el desarrollo y eficiente y efectiva comunicación.

Feature-driven Development [27] Esta metodología, ideada por Jeff De Luca y Peter Coad, combina el desarrollo dirigido por modelos con el desarrollo ágil. Se centra en el diseño de un modelo inicial, cuyo desarrollo será dividido en función a las características que debe cumplir el software e, iterativamente, se diseñará cada una de estas características. Por tanto, cada iteración consta de dos partes, diseño e implementación de cada característica. Este tipo de metodología está dirigido al desarrollo de aplicaciones con un alto grado de criticidad.

Lean Software Development [28] es una metodología de desarrollo dirigida especialmente al desarrollo de sistemas cuyas características varían constantemente. Fue definida por Bob Charette’s a partir de su experiencia en proyecto industriales, constituyendo una adaptación para el desarrollo software de las lecciones aprendidas en la industria, en particular, en el sistema de producción automovilista japonesa de

Toyota. La metodología establece que todo cambio en el desarrollo software conlleva riesgos, pero si se manejan adecuadamente pueden convertirse en oportunidades que mejoren la productividad del cliente. Consta de siete principios dirigidos a gestionar los cambios: eliminación de todo aquello que no aporte valor al negocio, conocimiento incremental, toma de decisiones tan tarde como sea posible, deliberar funcionalidad tan pronto como sea posible, poder del equipo, construcción incremental del producto y perspectiva global del proyecto.

2.1.6. Estudios empíricos de aplicación de Metodologías ágiles

Las metodologías ágiles son sin duda uno de los temas emergentes en ingeniería del software que está acaparando gran interés investigador. De hecho, están ocupando un espacio destacado en la mayoría de conferencias y workshops celebrados en los últimos años. Aunque algunas críticas argumentan que no son más que viejo vino presentado en botellas nuevas[7] [29], otros estudios reflejan que el desarrollo de productos en entornos ágiles es muy diferente al desarrollo de productos en entornos convencionales, y que, por tanto, se necesitan estudios que indaguen en este sentido. Por ejemplo, en [30, 31] se analizan las diferencias existentes en el área de ingeniería de requisitos, y en [32] aspectos relacionados con la gestión del proyecto, concluyendo que aquellas organizaciones que utilizan metodologías ágiles en la gestión de *releases* incrementan su satisfacción.

En línea con este último estudio, la mayor parte de las investigaciones relacionadas con las metodologías ágiles se centran en describir experiencias exitosas que surgen al utilizar dichas metodologías [33, 34, 8, 7]. Por ejemplo, Mann y Maurer en [8] presentan un estudio del *overtime* y la satisfacción del cliente al utilizar SCRUM como metodología de desarrollo, concluyendo que los retrasos se reducen y que tras la realización del estudio todos los desarrolladores recomiendan la utilización de SCRUM como metodología de desarrollo. No obstante, muchos de estos estudios no proporcionan información suficiente sobre el contexto en el que se establece la experiencia, ni resultados cuantitativos junto a las observaciones que permitan la replicación del estudio para validar sus conclusiones.

Uno de los aspectos que ha sido estudiado con mayor interés en los últimos años es la productividad que permiten alcanzar las metodologías ágiles. En [7] se comparó la productividad de dos proyectos similares. Un equipo utilizó una metodología tradicional y el otro XP. Encontraron que, en general, el equipo XP obtuvo una productividad 42% mayor que la del equipo tradicional. No obstante, en la primera iteración, la productividad del equipo XP fue mucho mayor y en la última no hubo apenas diferencia. Dalcher et al. [2] realizaron un experimento en el que 15 equipos desarrollaron productos software similares. Los modelos utilizados fueron: Modelo en V, Incremental, Evolutivo, y XP. La mayor diferencia recayó entre los equipos que utilizaron XP y los equipos que utilizaron el Modelo en V. Los primeros tuvieron una productividad 337% mayor. No obstante escribieron 3.5 veces más líneas de código sin implementar más funcionalidad. Al contrario que los estudios citados anteriormente, Wellington et al. [3] encontraron un 44% menos productividad en equipos que

[7] Adaptación del título del artículo [Merisalo], "Is Exreme Programming Just Old Wine in New Bottles: A Comparison of Two Cases"

utilizaban XP. Svensson y Host [35] no observaron diferencia notable entre metodologías agiles y tradicionales. Publicaciones coetáneas con resultados tan contradictorios reflejan claramente la necesidad de realizar nuevos estudios que permitan profundizar en este aspecto.

Por otro lado, respecto al área de ingeniería de requisitos, una de las etapas más conflictivas en *agile*, aún hay muy pocos estudios sobre cómo los proyectos reales que utilizan metodologías ágiles realizan este proceso. Ciertos detractores argumentan que las metodologías ágiles podrían impactar negativamente en los principios de resolución, adecuación y veracidad de los requisitos [36]. Por otro lado, estudios recientes comienzan a identificar y dar solución a algunos de los problemas detectados en esta fase. Por ejemplo, en [37] se propone realizar una negociación de requisitos explícita con el cliente o en [38] incorporar una fase de especificación de requisitos similar a la llevada a cabo en metodologías convencionales. En [39] se propone incorporar conceptos de orientación a aspectos y en [40] establecer trazabilidad. En [41] se muestra el resultado de un experimento al aplicar el proceso de Gestión de la Interacción de Requisitos (RIM) donde se proponen modificaciones al proceso de requisitos en las metodologías ágiles, particularmente en eXtreme Programming. No obstante, muchos de estos artículos solicitan explícitamente la realización de nuevos estudios en este sentido que permitan dar nuevas y eficaces soluciones a los problemas encontrados.

Finalmente, respecto a las metodologías ágiles analizadas en las investigaciones, la tabla 2.1 muestra la distribución establecida por [6]. Este reciente estudio revisa el estado actual de la investigación respecto a las metodologías ágiles. De los estudios considerados, aquellos de mayor calidad, el 76% fueron realizados utilizando XP. Metodologías como SCRUM y Lean Software Development apenas contaron con un artículo de interés cada una.

Tabla 2.1 Distribución del tipo de metodología ágil utilizada en las investigaciones

Metodología de desarrollo ágil	Número	Porcentaje
XP	25	76
General	5	15
Scrum	1	3
Lean Software Development	1	3
Otras	1	3
Total	33	100

La principal implicación investigadora de este estudio establece la necesidad de realizar más y mejores estudios de desarrollo ágil. En particular, la prioridad se centra en realizar estudios sobre metodologías de gestión de proyectos como SCRUM, tal como el presentado en esta tesis de máster, por ser muy populares en la industria software pero sin apenas atención investigadora.

Por último, cabe destacar un nuevo estudio realizado en [42], que por ser muy reciente no fue contabilizado en [6], en el que se analiza el proceso de auto-organización al que se enfrentan los equipos de desarrollo software al introducir SCRUM como metodología para la gestión del proyecto.

2.2. Evolución del Producto Software

2.2.1. Introducción a la evolución del producto software

La ISO/EIC 14764 (1999) define mantenimiento del software[8] como *el proceso por el cual un producto sufre modificaciones en su código y documentación asociada para solucionar un problema o mejorar*. El estándar 1219 de IEEE (1998) lo define como *la modificación de un producto software tras ser liberado para corregir defectos, mejorar la ejecución de otros atributos, o adaptar el producto a un nuevo entorno* (este último objetivo es el que persigue el proyecto presentado en esta tesis de máster). Sea cual sea la definición que se considere, el concepto de evolución del software implica un cambio continuo desde un estado menor, más simple o peor a uno superior o mejor. Los objetivos que se persiguen al evolucionar un producto software son diversos: mantener operativo el sistema, incrementar la satisfacción de los usuarios, maximizar la inversión y reducir los costes, alargar la vida del software o adaptarlo a nuevos cambios o requisitos.

Las conocidas Leyes de Lehman para la evolución del software establecen:

- *1ª Ley*: Un programa grande que es utilizado se somete a un cambio persistente o se convierte en menos útil progresivamente. El proceso de cambio continúa hasta que se juzga como más rentable reemplazar el sistema por una nueva versión.
- *2ª Ley*: Como un programa grande cambia de forma continua, su complejidad la cual refleja una estructura deteriorándose, se incrementa a menos que se realice un trabajo para mantenerla o reducirla.
- *3ª Ley*: Las medidas de un proyecto global y atributos del sistema se auto regulan cíclicamente con tendencias e invariantes estadísticamente determinables.
- *4ª Ley*: El ratio de actividad global en un gran proyecto de programación es invariante.
- *5ª Ley*: En una evolución fiable y planificada, un gran programa sometido a un cambio debe estar disponible para una ejecución regular del usuario en el máximo intervalo determinado por su crecimiento neto. Es decir, el sistema desarrolla un promedio característico de crecimiento seguro, que de ser excedido, causa problemas de calidad y utilización con tiempo y coste que excede del previsto.

En el ciclo de vida del desarrollo software la primera versión es sólo una pequeña parte del trabajo ya que el mantenimiento y la evolución cubren la mayoría del ciclo de vida. De hecho, se estima que la evolución del software consume un 80% del presupuesto mientras que el desarrollo es el 20% restante. La evolución del software implica un mantenimiento del mismo durante el desarrollo pero se convierte en una garantía frente a la obsolescencia. El turbulento entorno que rodea la producción

[8] Para simplificar, en este estudio consideraremos "mantenimiento" y "evolución" como sinónimos referidos a todo el trabajo ejecutado tras la primera *release* del software.

software en la actualidad, y el cambio continuo en las necesidades del cliente implican un proceso evolutivo constante.

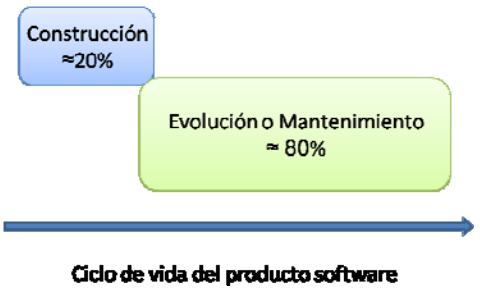


Figura 2.7 Esfuerzo del mantenimiento en el ciclo de vida de un producto software

2.2.2. El proceso de evolución del producto software

Desde una perspectiva convencional la evolución del producto software sigue un rígido proceso, cuyo inicio es la *gestión de cambios*, que permite controlar las distintas versiones y cambios de un sistema durante su ciclo de vida. En la gestión de cambios las peticiones de los cambios son registradas y se pasa a una fase de análisis del impacto en la que se determina el ámbito de los cambios requeridos como base para su implementación. Las actividades principales de esta fase son:

- Evaluar las peticiones de cambios sobre los sistemas existentes, otros sistemas, hardware, estructuras de datos, etc.
- Desarrollar una estimación preliminar de recursos.
- Documentar el ámbito del cambio y actualizar la petición realizada.

Una vez realizado el análisis de impacto se *planifica la versión del sistema* para determinar el contenido y tiempo de las versiones del sistema, constituyendo un subproceso en sí mismo con un amplio conjunto de actividades:

- Ordenar y seleccionar las peticiones de cambio para la próxima versión.
- Procesar los cambios y organizar el trabajo.
- Preparar un documento sobre el plan de la versión del sistema.
- Actualizar las peticiones de cambio aprobadas.

Constituida esta etapa, que se puede considerar como pre-evolución, se pasa a la fase de evolución en sí del producto. Se comienza *introduciendo cambios en el diseño* consistentes en revisar el diseño lógico (nivel de sistema) y físico (nivel de programa) para los cambios aprobados. Las principales actividades en esta fase son:

- Analizar los cambios aprobados y revisar la estructura del programa.
- Revisar o desarrollar los diseños lógicos y físicos.
- Diseñar los cambios de hardware si fueran necesarios.
- Actualizar los documentos del sistema y del programa.
- Restaurar los documentos bajo control del sistema de gestión de cambios.
- Actualizar las peticiones de cambios para reflejarlos en los documentos.

Posteriormente se pasa a la *fase de codificación* con el fin de reflejar los cambios aprobados y representados en diseño. Esta tarea implica:

- □ Implementar y revisar los cambios en el código.
- □ Restaurar el código fuente bajo control del sistema de gestión de la configuración.
- □ Actualizar las peticiones de cambios de los módulos modificados.

Y, finalmente, *se ejecutan las pruebas* o test tanto para asegurar la conformidad de los cambios aprobados con los requisitos originales como para asegurar la calidad del producto final. Esta fase es muy importante ya que los resultados obtenidos dependerán, en gran medida, de un correcto diseño de pruebas. Un aspecto a destacar en este sentido es que sólo necesitan revisarse los cambios y no el producto concreto.

2.2.3. Tipos de evolución software

En función del tipo de objetivo, existen cuatro tipos de mantenimiento o evolución del software:

1. *Evolución correctiva:* se define como aquel proceso orientado a la reparación de defectos existentes en un sistema software. Se utiliza, por ejemplo, cuando un programa falla o aborta, o cuando produce resultados que no son acordes con los requisitos o el diseño. Este tipo de evolución tiene asociada dos problemas principales:
    - Por un lado, reparar un defecto tiene la probabilidad de introducir otro defecto.
    - Por otro lado, incrementa notablemente las operaciones de testeo.

    La evolución correctiva se suele utilizar bien en reparaciones de emergencia, ejecutadas en cortos periodos de tiempo y, generalmente, sobre un único programa, o bien, en reparaciones planificadas que arreglan defectos que no requieren una atención inmediata

2. *Evolución adaptativa*: se define como el proceso para mejorar la funcionalidad del software, hardware y su documentación. El mantenimiento adaptativo está formado por cinco pasos:
    - Definición de requisitos, donde se revisan y entienden los requisitos de nuevos cambios.
    - Diseño del sistema, para determinar donde y cuando implementar los cambios en el sistema.
    - Diseño de datos, ya que los cambios en las estructuras de datos también deben ser diseñados.
    - Diseño del programa, para analizar los documentos de diseño del programa y determinar donde añadir, modificar y borrar las funciones que implementan los cambios propuestos.
    - Diseño del módulo, donde se analizan los documentos que determinan como integrar cambios en un módulo existente o bien crear uno nuevo.

    La evolución objeto de esta investigación se encuadra en este grupo. Los beneficios de este tipo de evolución proporciona son muy significativos. Entre otras podemos destacar que mejora la productividad automatizando actividades,

incrementa la satisfacción del usuario e incrementa las ganancias mediante la mejora de respuestas a oportunidades del negocio.

3. *Evolución perfectiva*: es un método para tratar de pulir o refinar la calidad del software y su documentación. El objetivo principal es hacerlo más fácilmente mantenible reduciendo el coste e impacto de futuros cambios. Para cumplir este objetivo se llevan a cabo actividades de reingeniería, reescritura y actualización de la documentación.
4. Finalmente, la *evolución preventiva*: es la que se ejecuta para prevenir fallos antes de que éstos ocurran. Por ejemplo, un chequeo de tipos antes de compilar y ejecutar una aplicación. Las técnicas más importantes para este tipo de evolución son:
    - Análisis de complejidad, para medir la complejidad de los módulos.
    - Análisis de funcionalidad, ya que medir la funcionalidad de un sistema ayuda a estimar su valor.
    - Ingeniería inversa, consistente en recuperar el diseño cuando el código no es acorde con la documentación.
    - Reingeniería por partes, para sistemas de vida corta y cambios evolucionarios necesarios.
    - Translación/reestructuración/modularización, para migraciones de plataforma.

2.2.4. Estudios empíricos sobre Metodologías ágiles y evolución del software

Desde hace décadas, muchos estudios han centrado sus esfuerzos en estudiar la evolución del software en metodologías convencionales [43], [44], [45], [46] o [47]. Está constatado que los cambios acumulados y el crecimiento de la complejidad es inevitable en software evolucionado. También se ha estudiado la necesidad de realizar trabajos extras con el objetivo de disminuir esta complejidad (refactorización).

Por otro lado, el amplio proceso que requiere la evolución de software en metodologías convencionales descrito en la sección 2.2.2 no se ajusta a los valores en los que se basan las metodologías ágiles. Sin embargo, considerando los principios en los que las sustentan como entregas continuas de software que funcione y aporten valor al cliente y adaptación a las variaciones en las necesidades del cliente y, por tanto, en los requisitos que debe cumplir el sistema, se puede deducir que la evolución del código es un objetivo explícito de las metodologías ágiles y la refactorización es vista como algo necesario y positivo.

La atención de los estudios sobre metodologías ágiles realizados hasta el momento se centra en la situación antes y durante el desarrollo inicial. Por ejemplo, Boehm y Turner en [48] establecen cinco dimensiones de riesgo a considerar para decidir que metodología utilizar (tamaño, criticidad, dinamismo, equipo y cultura), sin considerar el tipo de desarrollo. Evidencias empíricas en este sentido podrían ayudar a las organizaciones a decidir qué método escoger.

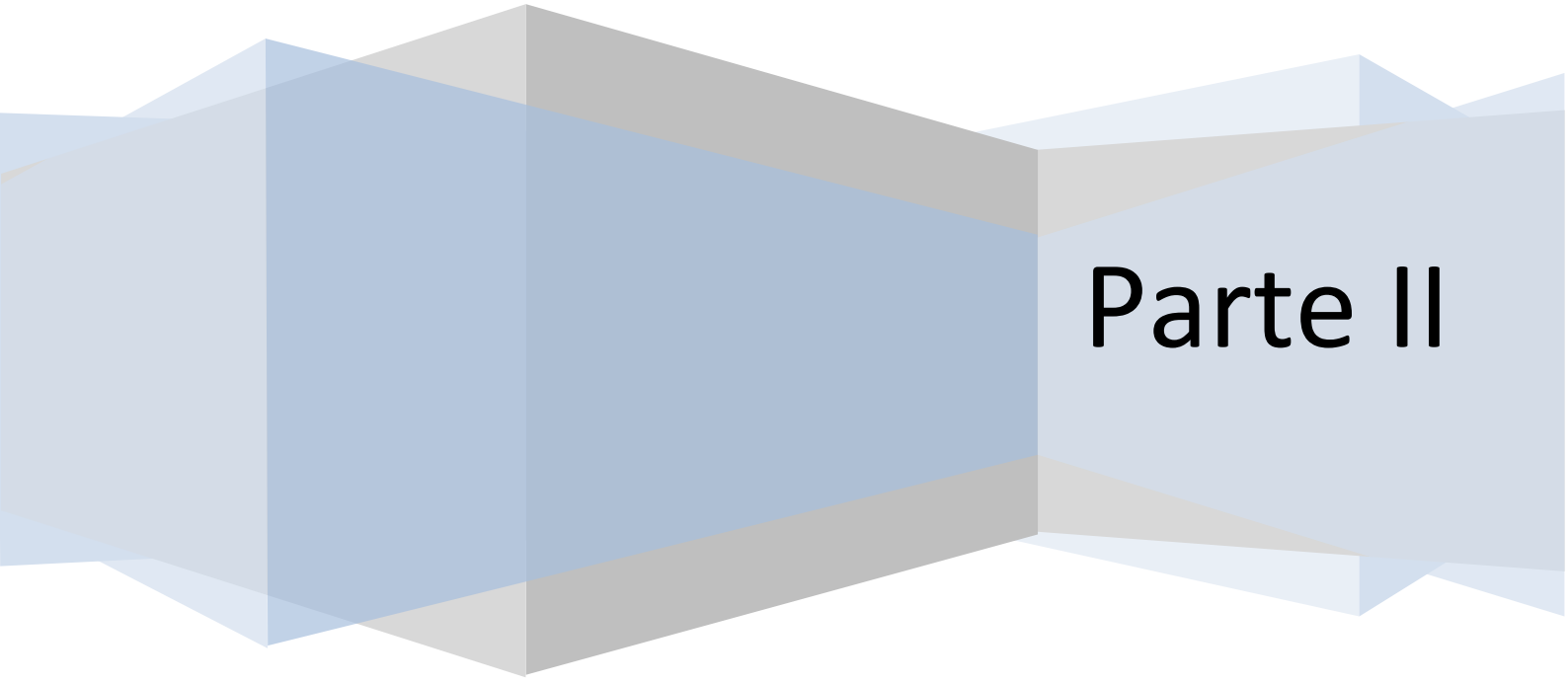
Sorprendentemente, a pesar de que la mayor parte del desarrollo software se centra en mantener y evolucionar productos, pocos estudios se han dedicado a estudiar la

evolución y mantenimiento del software ágil, ya que requiere tanto cualitativas como cuantitativas observaciones cuya recolección y estudio es dificultoso. Una de las pocas excepciones fue el estudio desarrollado por Chapin en [49] donde discute cómo encajan los procesos de agilidad en la evolución del software considerando los diferentes tipos de *stakeholders* y concluyendo que los efectos de utilizar metodologías ágiles pueden ser más positivos para unos que para otros. Wernick y Hall [50] argumentan que la técnica de pair-programming podría ser beneficiosa cuando la evolución es prolongada en el tiempo. Otro estudio destacable es [35] en el que se presentan los resultados del estudio llevado a cabo en una organización donde un método ágil fue introducido en un entorno de desarrollo de evolución y mantenimiento de producto software, aunque se trata de un amplio estudio que no se centra explícitamente en la característica de evolución. Por último, Pries-Heje y Lindwall en [51] divulgan los resultados obtenidos al realizar entrevistas en nueve organizaciones en desarrollo software sobre la evolución de productos utilizando metodologías ágiles. Intentan averiguar cuál es el motivo por el cual, a pesar de que la evolución del software y las metodologías ágiles están tan relacionadas, son conceptos disjuntos en las mayoría de las organizaciones. La conclusión a la que llegan es que en las organizaciones el proceso de mantenimiento y evolución del software sigue caminos muy tradicionales, puesto que el pilar que guía la evolución es la calidad y en ciertos sectores de la ingeniería del software se tiene la idea de que las metodologías ágiles son un proceso de codificación, sin documentación ni control alguno que no asegura la calidad de los productos. Por lo tanto, concluyen que se trata de un área en el que las metodologías ágiles aún no han sido altamente introducidas. Finalmente, solicitan la realización de estudios en este sentido que ahonden sobre los beneficios que las metodologías ágiles pueden proporcionar en la evolución y mantenimiento de productos software.

Sin embargo, las conclusiones obtenidas en estos estudios tienen carácter cualitativo. Tras una ardua exploración del estado de la investigación en este campo, desde nuestro conocimiento, solamente en [7] se presenta un estudio basado en medidas a este respecto. La experiencia de este estudio reporta que las metodologías ágiles permiten una evolución aboliendo problemas de incremento de la complejidad del código o decremento en la satisfacción de los clientes. Implícitamente, solicita el desarrollo de más sistemas utilizando metodologías ágiles para estudiar si las conclusiones obtenidas pueden ser generalizadas.

Ejecución del Estudio

Estudio de la Aplicación de Metodologías Ágiles para la Evolución de Productos Software



Capítulo 3

Introducción a la Investigación

En esta segunda parte de la tesis de máster se presenta el experimento que da base a este estudio. Dado que el objetivo general del estudio es muy amplio, en este capítulo se acota dicho objetivo general, definiendo claramente las áreas de interés para el mismo a través de cuestiones de competencia que serán resueltas a lo largo del estudio.

Además, con la pretensión de que el lector tenga una visión global del experimento que le permita una mejor comprensión del mismo, en este capítulo se presenta una síntesis general del experimento, que será minuciosamente detallada en el resto de capítulos.

3.1. Definición de los Objetivos. Cuestiones de Competencia

Atendiendo al objetivo general, presentado en el capítulo 1, que dirige la investigación de esta tesis de máster:

> *"El objetivo general de esta tesis de máster es analizar el mantenimiento o evolución de productos software, en el contexto de utilización de metodologías ágiles. Concretamente, se pretende estudiar la evolución de un producto software concreto, TOPEN, especialmente diseñado para ser adaptable a diferentes dominios de aplicación con un coste reducido."*

En esta sección se presentan, a través de cuestiones de competencia, los subobjetivos a satisfacer para cubrir este objetivo general.

Los objetivos perseguidos en este estudio siguen principalmente seis líneas de interés: características del producto evolucionado durante el desarrollo, aspectos de gestión en el equipo de desarrollo al utilizar la metodología SCRUM, calidad del producto obtenido, esfuerzo de adaptación de la metodología en grupos que carecen de experiencia en este tipo de desarrollo, satisfacción del cliente y de los desarrolladores durante y tras la ejecución del proyecto, y finalmente, análisis de la metodología tras la experiencia adquirida en la investigación.

- Características del producto durante la evolución

  El principal interés de la investigación es analizar la evolución del producto a lo largo del desarrollo y la integración continua que se ha experimentado al utilizar SCRUM como metodología de desarrollo, principalmente por la importancia que este aspecto tiene en el desarrollo software y la ausencia de estudios en este sentido:

  - ¿Cómo han evolucionado las líneas de código a lo largo del proyecto?

  Dado que el lenguaje de programación utilizado es Java

  - ¿Cómo han evolucionado las clases del producto a lo largo de los sprints?
  - ¿Cuántas nuevas han aparecido? ¿Cuántas han cambiado? ¿Cuántas han sido eliminadas?
  - ¿Hay clases que han sufrido un mayor impacto?

- Aspectos de agilidad en el equipo de desarrollo

  Una de las principales motivaciones del movimiento ágil es aumentar la productividad del equipo para adaptarse a la feroz competencia de la industria software actual, reduciendo drásticamente los tiempos de desarrollo y adaptándose a las variabilidades del mercado rápidamente para obtener una ventaja competitiva. Por tanto, es de interés estudiar aspectos relacionados con la gestión del equipo durante el experimento. Concretamente se pretende dar respuesta a las siguientes cuestiones:

  - ¿Hemos sido ágiles?
  - ¿Hemos cumplido los objetivos previstos?
  - ¿Es adecuada la relación entre el esfuerzo dedicado y el producto obtenido?
  - ¿Hemos sido capaces de adaptarnos a los cambios e imprevistos que han surgido a lo largo del desarrollo?

- Calidad del producto obtenido

  Por otro lado, las metodologías ágiles ponen gran empeño en obtener productos de calidad a través de la continua interacción con el cliente. De hecho uno de sus principios es que la atención continua a la calidad técnica y al buen diseño mejora la agilidad. En este sentido estudiaremos:

  - ¿Hemos obtenido un producto de calidad?
  - ¿Cuál es la distribución de defectos identificados a lo largo del desarrollo?
  - ¿Cuál es el nivel de defectos en productos ya entregados al cliente?
  - ¿Cuánto de ágiles hemos sido para dar solución a los defectos detectados?

- Esfuerzo de adaptación de la metodología

  Es también de interés estudiar el esfuerzo que ha supuesto evolucionar desde el estado anterior de desarrollo, considerado más convencional, a la metodología SCRUM, incluyendo las variaciones incluidas en la misma, y la formación de aquellos miembros que no estaban familiarizados con la metodología ágil.

  - ¿Cuánto de grande es el esfuerzo de familiarización con la metodología SCRUM?
  - ¿Cuál es la curva de aprendizaje del equipo en función de aspectos tales como el cumplimiento de objetivos, su satisfacción personal ó hábito en las prácticas agile?

- Satisfacción del cliente y de los desarrolladores

  Dado que el movimiento ágil y más concretamente SCRUM no establece prácticas ingenieriles, sino que más bien se puede entender como una nueva cultura de desarrollo software, es de interés para el estudio conocer las percepciones que tanto el cliente como el equipo de desarrollo tienen sobre la metodología.

  - ¿La metodología SCRUM satisface al cliente?
  - ¿La metodología SCRUM satisface al equipo de desarrollo?

- Análisis de la metodología SCRUM

  Una vez realizado el experimento, apoyándonos en la experiencia adquirida en el mismo, se analizarán las metodologías ágiles y las prácticas que proponen desde una perspectiva más ingenieril.

  - ¿Cuáles son las ventajas o puntos fuertes de estas metodologías en la evolución de productos software?
  - ¿Qué problemas presenta la metodología?

Tras el estudio y con los resultados obtenidos será posible estimar con más conocimiento de causa la evolución de un producto abordado con metodologías ágiles tanto desde el punto de vista del proceso como desde la perspectiva del producto. La pregunta que guía la investigación se puede formular como *"¿Cómo valorar la evolución de un producto si aplicamos una metodología ágil?"*

3.2. Visión Global del Estudio

Como ya hemos comentado reiteradamente el experimento se centra en evolucionar el dominio de aplicación de un producto concreto, TOPENprimer, utilizando en dicha evolución la metodología ágil SCRUM. De este modo se pretende obtener un entorno que permita la operación, monitorización y ejecución de pruebas sobre plantas de producción de biogás, entorno al que denominaremos TOPENbiogas, a partir de un entorno que actualmente permite la operación, monitorización y ejecución de pruebas sobre máquinas de juego interconectadas, TOPENprimer. Un aspecto importante a considerar en esta evolución es que en la arquitectura TOPEN, sobre la que se sustentan ambos productos, se han identificado las partes constantes y variables con el objetivo de construir una línea de producto software lo que permite hacerse una idea de qué partes de TOPEN van a ser susceptibles de modificación y qué partes permanecerán constantes y, lo que es más importante aún, qué partes de TOPEN deberán ser probadas en la

evolución del producto. Aunque podemos considerar el entorno TOPEN, por su naturaleza, complejo, esta circunstancia nos permite apartar cierta complejidad en el desarrollo del producto para centrar los esfuerzos en el seguimiento de la metodología en la evolución del producto.

El estudio tiene una duración aproximada de ocho meses, en el que se realizan nueve sprints de dos semanas de duración cada uno, exceptuando algunos sprints que por cuestiones propias del desarrollo o por incluir días festivos en su ejecución vieron reducida o ampliada esta duración más un análisis de los resultados obtenidos. Como puede observarse el desarrollo del experimento se aparta levemente de la planificación establecida para el mismo (ver sección 5.2 *Plan de Ejecución* del capítulo 5 *Diseño del Experimento*). La principal razón de esta circunstancia se debe a que cuando se realizó dicha planificación, aunque se limitó que el desarrollo del proyecto que da base al experimento debía tener una duración de 14 semanas (ver figura 5.2 de la sección indicada anteriormente) y no debía salirse de estos márgenes, aspectos como la etapa de *pregame* fueron obviados así como las posibles incidencias que prolongase la duración de los sprints. De este modo, se planificó que el experimento constate de diez sprints de dos semanas de duración cada uno y, finalmente, por circunstancias del desarrollo, descritas en secciones posteriores, sólo se pudieron llevar a cabo nueve.

Durante este periodo se contó con un equipo formado por una media de nueve miembros. No obstante, el hecho de que el contexto en el que se enmarca esta investigación es un grupo de investigación universitario implica que parte del equipo no dedica el 100% de su tiempo al desarrollo, ya que tienen que desempeñar otras tareas docentes y de investigación.

Se distingue una clara división en el desarrollo del experimento comprendida entre el tercer y el cuarto sprint. Los tres primeros sprints del mismo (t_0+6 semanas[9]) se dedicaron de forma exclusiva a la puesta a punto de la investigación. Por un lado, recopilación de información relativa a las metodologías de desarrollo ágil y al estado de la investigación en este campo para fijar los objetivos del estudio. Por otro lado, diseño del experimento. Se dedicó gran esfuerzo en este sentido para controlar las posibles amenazas, derivadas principalmente del contexto en el que se realiza la investigación, que pudieran poner en peligro la autenticidad del estudio. Además, se puso especial interés en adecuar el entorno en el que se iba a desarrollar el experimento así como el material y herramientas a utilizar en el mismo. Aunque en esta primera parte del experimento no se trataba de desarrollar ningún producto software, se optó por poner en práctica la metodología SCRUM al tratarse de una metodología de gestión de proyectos más que de desarrollo puro. La principal motivación de esta decisión fue adquirir cierta experiencia en la puesta en práctica de la metodología.

Por tanto, el desarrollo del proyecto que da base al experimento se llevó a cabo en los seis sprints restantes (t_0+7 – t_0+22 semanas). A grandes rasgos el transcurso del mismo se caracterizó por los siguientes acontecimientos:

- Sprint 4

  La mayor parte del equipo de trabajo carecía de experiencia previa en metodologías ágiles de desarrollo y, por tanto, este sprint suponía el primer contacto con una de ellas, SCRUM. Además, la inclusión de desviaciones respecto a la metodología entendida ortodoxamente, por ejemplo al utilizar el

[9] En el estudio los tiempo se medirán en semanas representando t_0 el comienzo del proyecto.

perfil U2OP [52] para el modelado de operaciones, supuso una adaptación. A grandes rasgos este sprint se caracterizó por una pésima planificación, pues no se tenía un conocimiento lo suficientemente profundo del punto de partida ni del dominio de aplicación.

- Sprint 5

  En el sprint anterior se constató que la planificación del contenido de cada sprint era una fase muy delicada que influía radicalmente en el resto de fases. Por este motivo se puso especial énfasis en la fase de planificación. Las necesidades del cliente a cumplir en este sprint e identificadas en el Product Backlog, fueron descompuestas en un nivel mayor de detalle, lo que permitió realizar una planificación más realista. Además, durante el desarrollo se limaron algunos errores metodológicos. Se sucedieron una serie de incidencias, principalmente en la parte de pruebas y al añadir algunas funcionalidades que no estaban previstas, que provocaron leves retrasos. No obstante, el incremento en el conocimiento del dominio a evolucionar resultó muy positivo para la evolución del producto en este sprint.

- Sprint 6

  En este sprint el equipo comenzó a auto-organizarse basándose en su propia experiencia. Un hecho significativo en este sprint sucedió al constatar que había ciertas necesidades que se apartaban del foco de la funcionalidad pero de alta importancia en el producto final y que, sin embargo, estaban pasando inadvertidas porque el cliente centraba todos sus esfuerzos en aspectos funcionales. El hecho de que el producto desarrollado fuese en realidad la evolución de otro existente permitió advertir esta circunstancia y darle solución en este sprint. Aunque los objetivos no fueron cubiertos en su totalidad el balance continuaba siendo positivo y la evolución del producto trascurría de forma más ágil.

- Sprint 7

  Este sprint supuso un punto de inflexión en el proyecto pues aparecieron contratiempos a superar. Una serie de sucesos externos, propios del contexto universitario en el que se desarrolla la investigación pero imprevistos, influyeron en la imposibilidad por parte de algunos integrantes de mantener su compromiso. Un enorme sobreesfuerzo y una respuesta ágil por parte del equipo para redistribuir tareas, permitió cumplir los objetivos previstos. Cabe destacar que en este sprint se comenzó a fraguar la solución a un problema detectado durante el desarrollo, la representación y gestión de aquellas necesidades del cliente transversales al proyecto. La solución se basada en la introducción de un nuevo artefacto, las *historias de sistema*.

- Sprint 8

  En este sprint la motivación del equipo se estabilizó y el trabajo transcurría de acuerdo a los principios ágiles. Se utilizó por primera vez en el proyecto la técnica de refactorización sobre el simulador con resultados muy positivos. Otro aspecto importante a considerar en este sprint fue el hecho de que comenzasen a divulgarse resultados del estudio. Concretamente se realizó la ponencia "*Requisitos convencionales frente a Historias de Usuario en Metodologías ágiles*", presentada en las Jornadas Solo Requisitos 2008. Este sprint resultó, por tanto, muy positivo pues además de lograr superar una situación bastante

delicada presentada en el sprint anterior, se comenzaban a explotar resultados del estudio a parte de la consecución de los productos del proyecto.

- Sprint 9

  El sprint 9 constituía el último sprint del proyecto. El conocimiento adquirido tanto sobre los productos que estaban siendo desarrollados como sobre el dominio de aplicación, las plantas de producción de biogás, favoreció que en este sprint el trabajo pudiese ser calificado como muy ágil y se lograsen cumplir prácticamente todos los objetivos previstos en el experimento, además con una alta calidad. La satisfacción final del cliente y del equipo de desarrollo tanto desde una perspectiva industrial como desde una perspectiva metodológica fue muy alta. Cabe destacar que el esfuerzo necesario para evolucionar los productos en los últimos sprints fue considerablemente menor al esfuerzo en los primeros sprints.

En el resto del capítulo se describirá de una forma más profunda el desarrollo del experimento analizando detalladamente aspectos como el contexto en el que se desarrolla, el diseño y ejecución del mismo, y el análisis de los resultados obtenidos.

Capítulo 4

Contexto del Experimento

Uno de los *hándicaps* con los que nos encontramos al evaluar la validez de un experimento, especialmente en experimentos referentes a metodologías de desarrollo ágil, es la ausencia de una descripción detallada del contexto en el que se desarrolla la investigación**.** Definir el contexto es extremadamente importante puesto que permite tanto asegurar que los objetivos de la investigación están apropiadamente definidos como garantizar que la descripción es lo suficientemente detallada para que otros investigadores puedan utilizar los resultados obtenidos, permitiendo la replicación del estudio, comparación o generalización de las conclusiones [53].

En este apartado se describen detalladamente los factores contextuales que impactan en la realización del experimento. En alineación con el marco de validación XP (*XP-Evaluation Framework*) [54], que proporciona un formato para describir el contexto en experimentos con la metodología ágil XP y extensible a cualquier otra metodología ágil, y los estudios realizados en [53] y [55], se detallan los siguientes aspectos:

- caracterización del producto software,
- caracterización del proceso software,
- factores sociológicos,
- factores ergonómicos,
- factores geográficos
- y factores tecnológicos,

4.1. Caracterización del Producto Software

Como proyecto, dentro del experimento, se ha desarrollado un entorno de trabajo que permita operar/monitorizar y probar el comportamiento de una planta de producción de biogás (en adelante, planta de biogás) utilizando la metodología de desarrollo ágil SCRUM. En realidad, se trata de la evolución de un producto concreto para su adaptación al dominio de la producción de biogás. A continuación, se describe detalladamente el contexto del desarrollo y del producto sobre el que se realiza la investigación.

4.1.1. Contexto del desarrollo: de TOPENprimer a TOPENbiogas

El cliente ha elegido al grupo de investigación SYST para desarrollar el producto que le permita probar/operar y monitorizar la planta de biogás debido a la experiencia que el grupo ha adquirido durante años de trabajo en este área. Además, el grupo ya dispone de tecnología software para dar soporte a las pruebas y operación/monitorización de sistemas complejos, lo cual repercute positivamente en el tiempo de desarrollo, ya que el producto no se hace desde cero y parte del desarrollo consistirá en adaptar dicha tecnología software al dominio de plantas de biogás. En concreto, dentro del experimento se evolucionará la herramienta software ya existente de nombre *TOPENPrimer*.

TOPENPrimer es una adaptación del entorno TOPEN (*Test and OPeration ENvironment*) para máquinas recreativas. Tal como se describe en [56], TOPEN es un entorno para la validación, monitorización y operación de sistemas intensivos en software. El objetivo principal es proporcionar a los ingenieros de pruebas un marco para definir, compilar y ejecutar procedimientos de prueba sobre un sistema, y almacenar la información de dichos procedimientos y los resultados de sus ejecuciones para análisis posteriores. El sistema a validar, monitorizar u operar se denomina de forma genérica SUT (*System Under Test*). La figura 4.1 muestra la arquitectura operacional de TOPENprimer que comprende cuatro componentes distribuidos:

- *Topen Engine*, es el núcleo del entorno TOPEN que se encarga de la compilación y traducción de procedimientos de prueba, y soporta el control de ejecución de los mismos.
- MIB (*Mission Information Base*), contiene la base de datos y reglas de negocio asociadas.
- El componente *Gateway*, se encarga de la comunicación entre SUT y el componente Topen Engine.
- Y, finalmente, GUI (*Graphical User Interface*) constituye la interfaz gráfica de usuario desde la que el ingeniero de pruebas visualiza el estado del SUT en cada momento, y soporta la definición/ejecución de procedimientos de prueba y la monitorización del sistema.

El producto del que se parte TOPENprimer está, por tanto, desarrollado para validar, monitorizar y operar sistemas complejos de máquinas de juego interconectadas; más detalles de la adaptación se pueden encontrar en [56].

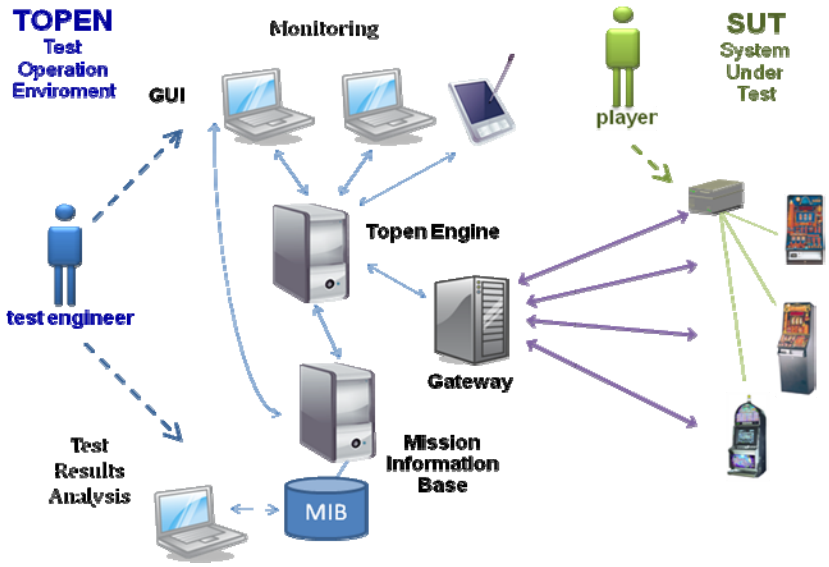


Figura 4.1 Escenario operacional del producto de partida TOPENprimer.

El nuevo dominio de aplicación al que habrá que adaptar TOPENprimer es el correspondiente a una planta de producción de biogás. El objetivo es obtener un producto, *TOPENbiogas*, que permitirá operar, monitorizar y probar el funcionamiento de dicha planta. Dada la complejidad de este tipo de sistemas, la implementación, de momento, se limitará a algunos componentes de la planta explicados en la sección 4.1.2.

Cabe destacar en este sentido que TOPEN está especialmente diseñado para ser adaptable a diferentes dominios de aplicación con un coste reducido [56]. De esta forma, se distingue una parte específica del dominio al que se pretende adaptar la herramienta y una parte independiente del mismo que le da soporte. Así se puede conocer de antemano qué partes de TOPEN son susceptibles de sufrir modificación facilitando la evolución del producto de TOPENprimer a TOPENbiogas.

Asimismo, como paso previo a la utilización de TOPENbiogas en la planta de biogás, se implementará un simulador de la misma que permita validar el entorno antes de su puesta en marcha en la planta real. Inicialmente, por motivos de seguridad, el cliente no quiere implantar el sistema en la planta real ya que podría interferir en su correcto funcionamiento. Por tanto, desea que se desarrolle un simulador que se comporte de igual forma que la planta de biogás y que, durante un tiempo, el entorno de pruebas/operación y monitorización se ejecute contra este simulador. Finalmente, tras un tiempo de validación, el entorno será implantado y el simulador se aparcará hasta que se demanden nuevas funcionalidades en el sistema y requieran su validación antes de la implantación. El simulador, al igual que los componentes de la planta, dispondrá de una interfaz orientada al usuario final del sistema, que le permita actuar sobre el mismo, por ejemplo, encendiendo o apagando los componentes, aumentando o disminuyendo la velocidad de un triturador, abriendo o cerrando una compuerta de entrada o salida, etc. También proveerá una interfaz software que posibilite la interacción con el entorno de operación y pruebas TOPENbiogas.

El desarrollo de este simulador aporta un valor añadido al experimento puesto que se desarrollará conjuntamente con TOPENbiogas aplicando la metodología ágil SCRUM, permitiendo, por tanto, analizar la metodología tanto en la evolución de un producto concreto como en la creación de un producto software desde cero en las mismas condiciones contextuales. Como muestra la figura 4.2, el entorno de trabajo

consta, por tanto, de dos partes fundamentales: 1) Entorno de pruebas/operación y monitorización TOPENbiogas y 2) Simulador.

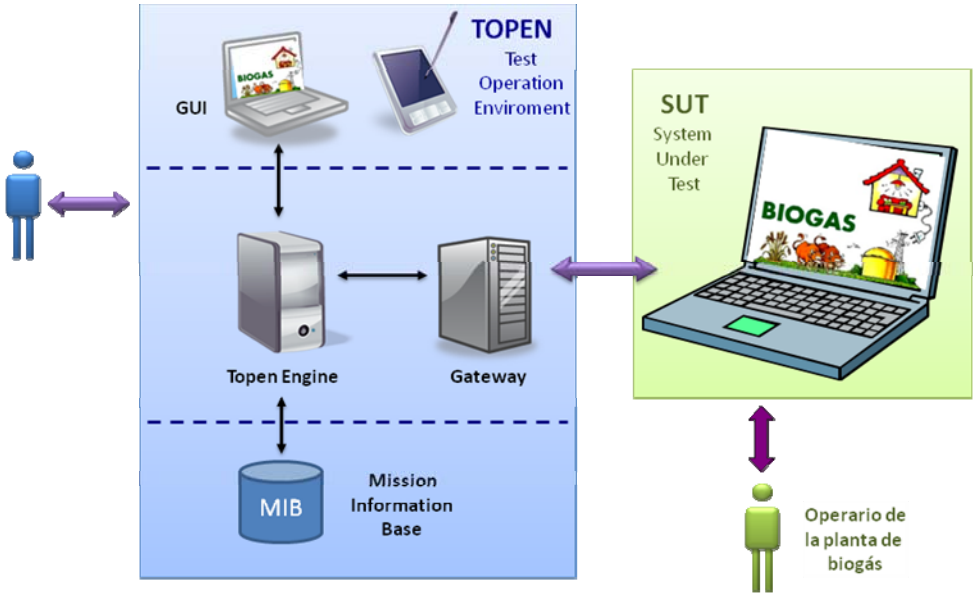


Figura 4.2 Proyecto TOPENbiogas

Con el objetivo de que se pueda valorar el proyecto desarrollado, la tabla 4.1 muestra algunas de las características generales del producto de partida TOPENprimer.

Tabla 4.1 Características del producto de partida TOPENprimer.

Factor Contextual	**Característica**	**PRODUTO DE PARTIDA TOPENprimer**	
Estructura	Arquitectura	Producto formado por 4 componentes distribuidos: MIB, TopenEngine, Gateway, TOE	
Tamaño	Número de líneas de código del sistema	30667	
	Número de clases del sistema	216	
	Número de líneas de código por componente	MIB	7779
		TopenEngine	8372
		Gateway	907
		TOE	13609
	Número de clases por componente	MIB	48
		TopenEngine	55
		Gateway	10
		TOE	103
Características Técnicas	Lenguaje de programación	Java	
	Comunicaciones	RMI (Java Remote Method Invocation) y Sockets	

Finalmente, tras analizar el producto a desarrollar cabe destacar que, según la clasificación establecida en XP-Evaluation Framework para los tipos de software, tanto

el producto principal TOPENprimer como el simulador a desarrollar pueden catalogarse como software *outsourced*, es decir, software construido bajo los requisitos de un cliente u organización concreta.

4.1.2. Contexto del producto

El cliente de este proyecto dispone de una planta de biogás que produce biogás y bio-fertilizantes a partir de residuos y subproductos orgánicos. Posteriormente, dicho biogás será convertido en energía eléctrica. Considerando que, para que el proceso tenga lugar con la mayor eficiencia posible, se deben controlar una serie de factores como la temperatura, la velocidad a la que trabajan los elementos del sistema, los nutrientes o el tiempo de resistencia, se tienen automatizadas ciertas tareas y controlados algunos de estos parámetros en la planta. La solución planteada por el cliente, objeto del producto resultante en esta investigación, es la de disponer de una herramienta software que le permita tanto realizar pruebas sobre el funcionamiento de la planta, como operar y monitorizar la misma, de forma remota. Si bien, en un principio otorga prioridad a la realización de pruebas. La herramienta software le permitirá almacenar en una base de datos aspectos del comportamiento de la planta al objeto de monitorizar y mejorar el rendimiento de la misma.

Para la realización del producto se tomará como normativa aplicable a las plantas de producción de biogás, la dispuesta por la Comisión Europea, reflejada en los siguientes reglamentos:

- *RCE 1774/2002*. Reglamento (CE) Nº 1774/2002 del Parlamento Europeo y del Consejo de 3 de octubre de 2002 por el que se establecen las normas sanitarias aplicables a los subproductos animales no destinados al consumo humano.
- *RCE 208/2006*. Reglamento (CE) Nº 208/2006 de la Comisión de 7 de febrero de 2006 por el que se modifican los anexos VI y VIII del Reglamento (CE) nº 1774/2002 del Parlamento Europeo y del Consejo, en lo que se refiere a las normas de transformación para las plantas de biogás y compostaje y las condiciones aplicables al estiércol.

Concretamente, en el contexto de este proyecto se entenderá por planta de biogás la definición recogida en el Anexo 1 de "Definiciones Específicas" del RCE 1774/2002, que dice así:

> *« Planta de biogás: planta en la que se proceda a la degradación biológica de productos de origen animal en condiciones anaerobias para la producción y recogida de biogás. »*

En la figura 4.3 se muestra de forma esquemática el proceso, que se basa en esta normativa, utilizado por el cliente en su planta para la síntesis de biogás y bio-fertilizantes.

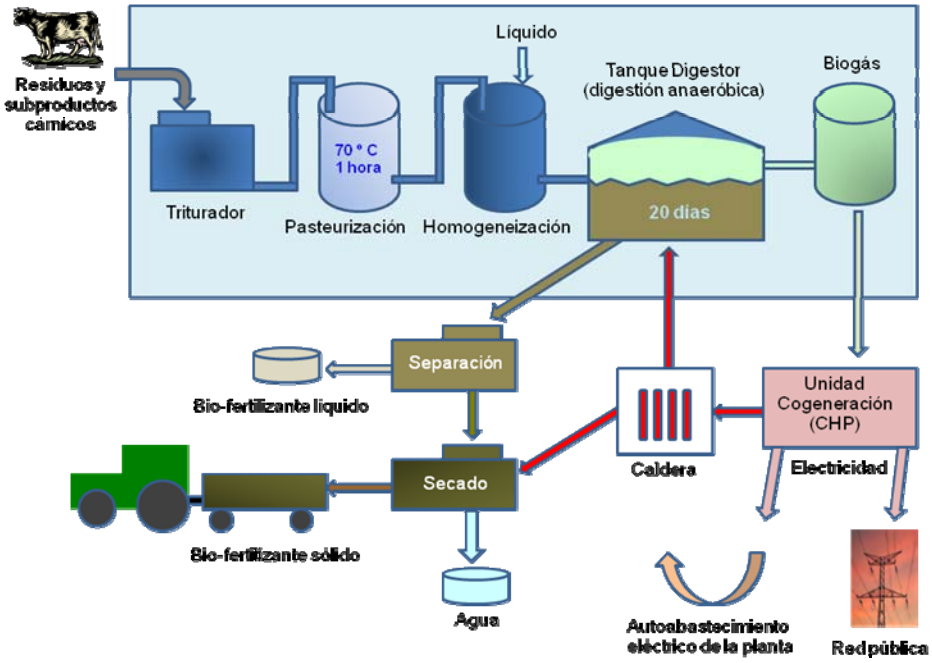


Figura 4.3 Esquema de funcionamiento de la planta de biogás del cliente

Debido a la complejidad del proceso, en una primera fase, resulta de interés para el cliente, el subproceso enmarcado en la parte superior de la figura por medio del recuadro, a través del cual se obtiene el biogás. Este subproceso incluye las siguientes etapas:

- **Etapa de trituración**

  En esta etapa se lleva a cabo la trituración de restos orgánicos sólidos procedentes de

  – Subproductos animales
  – Contenidos gastrointestinales como intestinos porcinos, restos de vacuno, etc.
  – Fangos de depuradoras procedentes de industrias agroalimentarias
  – Residuos lácteos

  Estos restos se introducen en una trituradora (manual o automática) y por medio de un tornillo sin fin se conducen al interior del tanque de pasteurización que constituye la siguiente etapa.

- **Etapa de Pasteurización**

  Los restos procedentes de la etapa anterior se tratan térmicamente a unos 70º C con el fin de esterilizar la materia, durante un tiempo no inferior a 1 hora. Además, se introduce agua para hacer una mezcla más homogénea. Realizada la pasteurización la masa se introduce en el tanque de hidrólisis.

- **Etapa de Hidrólisis**

  En el tanque de hidrólisis, también denominado de homogeneización, se tritura nuevamente la biomasa y se mezcla con agua para hacer la mezcla más homogénea aún. La materia ha de permanecer en el tanque de hidrólisis al menos durante 1 hora. Después se introduce la biomasa en el tanque digestor.

- **Etapa de Digestión**

  Esta etapa es la más importante del proceso. Dentro del tanque digestor la biomasa se trata de forma anaeróbica durante un periodo de 20 días, obteniéndose biogás y lodos orgánicos de los que se obtiene bio-fertilizante.

El cliente, por tanto, se pone en contacto con el grupo de investigación SYST en el que se encuadra esta investigación, para que le proporcionen una herramienta que le permita probar y monitorizar este proceso.

4.2. Caracterización del Proceso Software

Desde el punto de vista del proceso de desarrollo software seguido en el experimento podemos identificar dos partes claramente diferenciadas en los nueve sprint que constituyen el experimento. El punto de inflexión se encuentra en el sprint cuatro.

Los tres primeros sprint del experimento se han dedicado única y exclusivamente al diseño del mismo por lo que solamente ha sido necesario el esfuerzo de dos personas, responsable del experimento y director de la investigación, para realizar esta tarea y el trabajo ha sido más personal. Estas circunstancias han marcado la metodología seguida hasta este punto, ya que en estas etapas preliminares se ha seguido la metodología SCRUM con determinados matices. El hecho de que el responsable del experimento hiciese un trabajo más personal y el director de la investigación guiase sus pasos y supervisase el trabajo realizado, ha repercutido en que algunos aspectos de la metodología, como las reuniones diarias, careciesen de sentido. En este caso, por ejemplo, el trabajo a realizar por el responsable del experimento para completar una tarea requería más de un día por lo que las reuniones diarias en realidad se realizaban cada tres o cuatros días. No obstante, la principal motivación para seguir SCRUM en estas fases preliminares ha sido adquirir un conocimiento y experiencia previos al proyecto que mitigasen el proceso de aprendizaje de parte del equipo.

El proyecto que da base a este experimento, por tanto, comienza en el sprint cuatro. El proceso de desarrollo software seguido durante los seis sprint que constituyen esta parte ha sido lo más rigurosamente posible cercano a lo que teóricamente se conoce por SCRUM. No obstante, se han producido algunas adaptaciones al contexto en el que se establece la investigación.

- Se han realizado sprints de dos semanas de duración.

Además, han aparecido algunas limitaciones derivadas del contexto de la investigación que se apartan levemente de lo teóricamente establecido por SCRUM.

- Por un lado, algunos miembros del equipo tienen una dedicación parcial al proyecto. No obstante, este hecho no ha impedido que todos los

componentes del equipo hayan participado en todos los aspectos que establece la metodología. Es decir, todos los componentes del equipo han asistido y participado en todas las reuniones que implanta la metodología para la gestión del proyecto, aunque su dedicación a tareas de desarrollo fuese parcial.

- Por otro lado, se ha adaptado la metodología debido a las limitaciones del cliente, considerando una distancia geográfica notable entre el equipo de desarrollo y el cliente. De este modo, el cliente ha aportado la documentación base para la definición del Product Backlog y de las historias de usuario, y ha participado en el proceso de desarrollo, en ocasiones de forma distribuida.
- Finalmente, que con el objetivo de medir la evolución del equipo a lo largo del desarrollo se ha diseñado un método de puntuación de sprints, de tal forma que en función de los aspectos tratados al finalizar las reuniones retrospectivas el equipo obtuviese una calificación en el sprint. Las normas de puntuación de sprints aparecen en la sección de Anexos.

La figura 4.4 muestra gráficamente la metodología de desarrollo adaptada al proyecto concreto dentro del experimento.

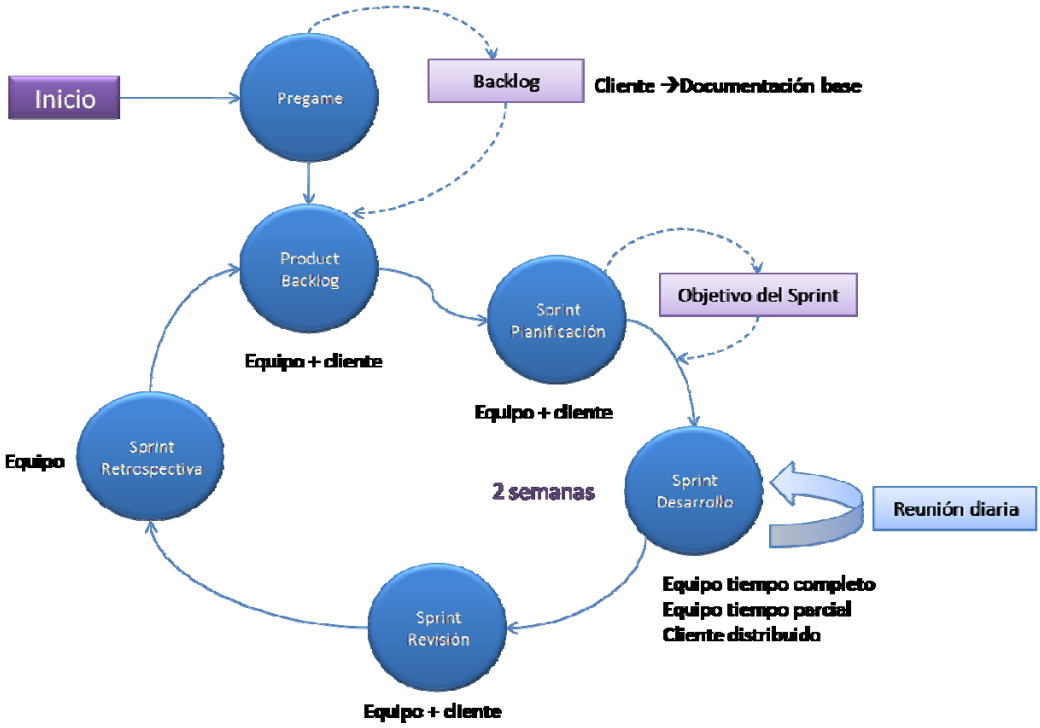


Figura 4.4 Metodología SCRUM adaptada al proyecto que da base al experimento

Cabe destacar también que en el desarrollo del experimento se ha utilizado el perfil de operaciones U2OP (UML 2 Operations Profile) definido por Pedro P. Alarcón [52]. El perfil U2OP, basado en los conceptos de modelo de operaciones y metamodelo de operaciones definidos por el mismo autor [52][57], aporta una extensión de UML para representar las operaciones de un sistema. De acuerdo al modelo, se entiende por operaciones de un sistema al conjunto de interacciones que se producen entre un operador y el sistema en términos de entradas y salidas del sistema, estando compuesto éste por una serie de elementos interconectados que tienen capacidad de interacción por

sí mismos con el operador. Estos elementos del sistema responden ante acciones de entrada concretas provenientes de algún tipo de aplicación software, a las que llama genéricamente *frontend de operaciones del sistema*. Como consecuencia de ello, el sistema produce ciertas salidas hacia el frontend de operaciones, en forma de respuestas a las entradas recibidas y de notificaciones de determinados eventos producidos en el sistema que son de interés para el contexto de operación definido. El sistema a su vez, monitoriza variables de su entorno por medio de *sensores*, a las que llama variables monitorizadas. Y controla variables del entorno por medio de *actuadores*, a las que llama variables controladas.

La figura 4.5 muestra el perfil U2OP. La aplicación de dicho perfil al modelo de un sistema, identifica explícitamente aquellos aspectos del sistema perceptibles por el operador al interaccionar con el sistema.

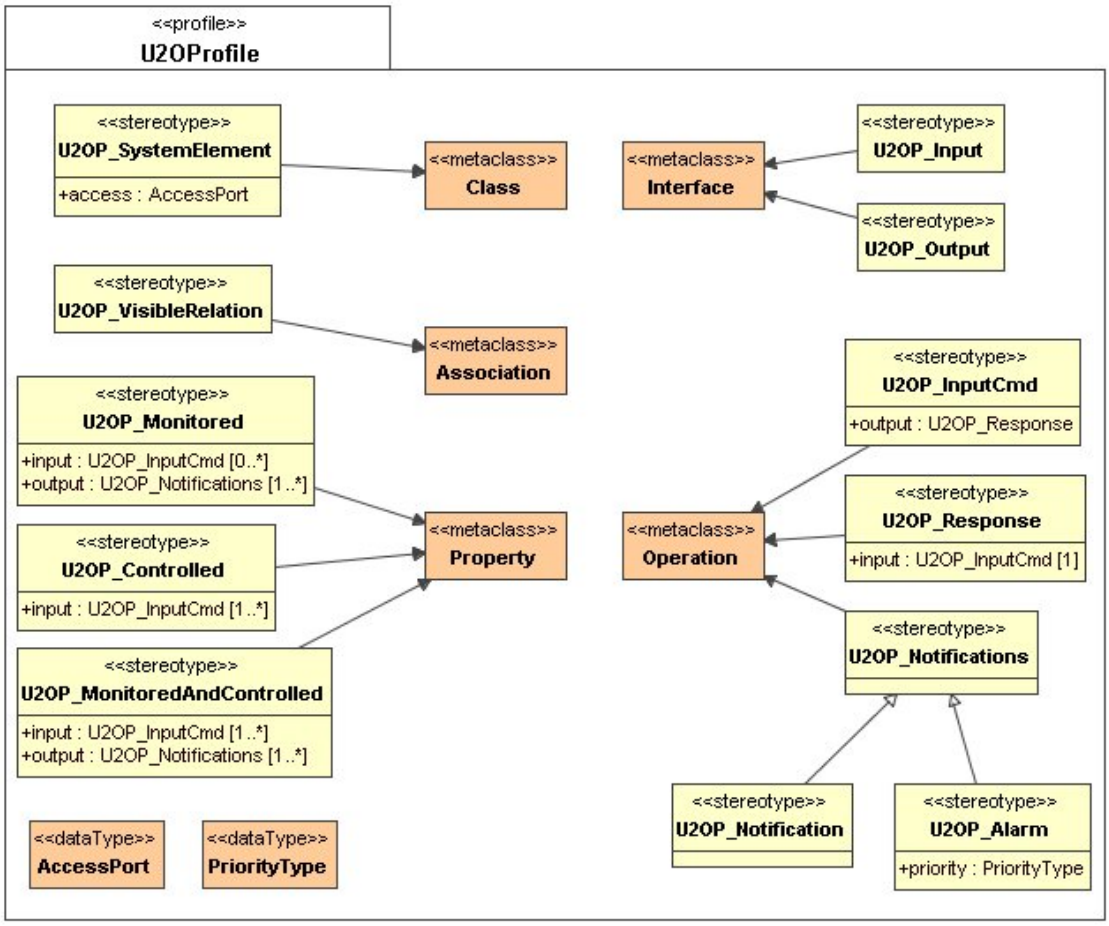


Figura 4.5 Perfil U2OP

La utilización del perfil U2OP en el desarrollo ha permitido definir inicialmente el lenguaje de operaciones del sistema de la planta de biogás, aplicándolo a los diferentes elementos, tanques, que componen la planta. La definición del modelo de operaciones de cada tanque aplicando el perfil U2OP, describe el lenguaje de operaciones que define la interacción entre cada tanque y el operador. Se identifica un tanque como elemento del sistema, y los componentes que lo integran. Se identifican las propiedades del tanque perceptibles por el operador, monitorizadas y/o controladas. Además se definen las entradas aceptadas por el sistema así como las salidas que genera por medio de diferentes interfaces y operaciones.

4.3. Factores Sociológicos

Los factores sociológicos se refieren a las características del personal que participa en el proyecto, tales como el tamaño del equipo, su experiencia o su disponibilidad para la ejecución del proyecto. Tienen un especial interés puesto que constituyen uno de los factores de riesgo más importantes en el desarrollo software [58]. Concretamente, en las metodologías ágiles, algunos estudios apuntan que los tres factores más importantes para su éxito son la cultura, la gente y la comunicación [11]. Los factores sociológicos generales del equipo que desarrolla este experimento se encuentran sintetizados en la tabla 4.2.

Tabla 4.2 Factores sociológicos generales.

Factor Sociológico	**Valor**	
Contexto	**Universitario**	
Tamaño del equipo	Tiempo completo	3
	Tiempo parcial	4
	Soporte	2
	Total	**9**
Titulación académica	Doctor en informática	2
	Estudiante de doctorado en informática	3
	Ingeniero técnico en informática (además, estudiante de máster)	2
	Estudiante de ingeniería técnica informática	2
Turnover [10]	0.2%	
Factores morales	Ninguno	

Cabe destacar que el proyecto se desarrolla en un contexto universitario. Concretamente, se enmarca dentro de una de las líneas de investigación que se están desarrollando en el grupo de investigación SYST de la Universidad Politécnica de Madrid y el proyecto de investigación ITEA2 FLEXI [4]. Por tanto, se ha de tener en cuenta que el ser un grupo de investigación universitario implica que algunos de sus miembros no dedican el 100% de su tiempo al proyecto, ya que tienen que desempeñar otras tareas docentes y de investigación.

El equipo está formado por nueve miembros, siete de ellos de forma continua a lo largo de todo el experimento y dos incorporados durante la última parte, de ahí el valor

[10] Métrica cuyo objetivo es examinar la variabilidad del personal. Las metodologías ágiles simplifican las tareas de documentación por lo que la mayor parte del conocimiento del proyecto puede considerarse tácito más que documentado. Esto hace que los proyectos que utilizan metodologías ágiles tengan especial dependencia del equipo de desarrollo.

Turnover= (nº miembros añadidos + nº miembros que abandonan el proyecto)/ nº total de miembros

0.2% de la métrica *turnover*. De ellos, dos son doctores en informática y el resto del equipo lo forman profesores, estudiantes de doctorado, máster y último año de carrera en ingeniería técnica en informática. Por tanto, se cuenta con un equipo con un alto grado de cualificación.

Las labores desempeñadas por cada uno de ellos varían levemente a lo largo del estudio. En los tres primeros sprints se llevaron a cabo labores de diseño y puesta a punto del experimento por lo que sólo fue necesario el trabajo de una persona durante este tiempo. A partir del cuarto sprint tres miembros se dedicaron única y exclusivamente al desarrollo de este proyecto (scrum máster y dos personas del equipo de desarrollo), empleando como mínimo 30 horas semanales. El resto del equipo lo compaginó con otro tipo de tareas universitarias.

Aunque, al tratarse de un equipo reducido y utilizar una metodología *agile*, el trabajo fue muy colaborativo y todos los miembros ayudaron en todas las fases de desarrollo del producto, podemos diferenciar distintos roles y miembros liderando diferentes tareas: tareas de gestión del proyecto, tareas de especificación de historias de usuario, tareas de diseño y modelado, tareas de desarrollo, tareas de pruebas y soporte técnico. Varios miembros del equipo desempeñaron más de una función dentro del contexto del proyecto. De esta forma, las personas que duplicaron el liderazgo de tareas fomentaron que se creara un lazo de unión más fuerte entre el equipo favoreciendo la filosofía *agile*. Finalmente, cabe destacar que, uno de los componentes dedicó sus esfuerzos dentro del proyecto, única y exclusivamente, a las tareas de *tester*, si bien una segunda persona fue incorporada durante los dos últimos sprints para apoyar en esta tarea. La figura 4.6 muestra gráficamente la distribución del equipo a lo largo del desarrollo del proyecto.

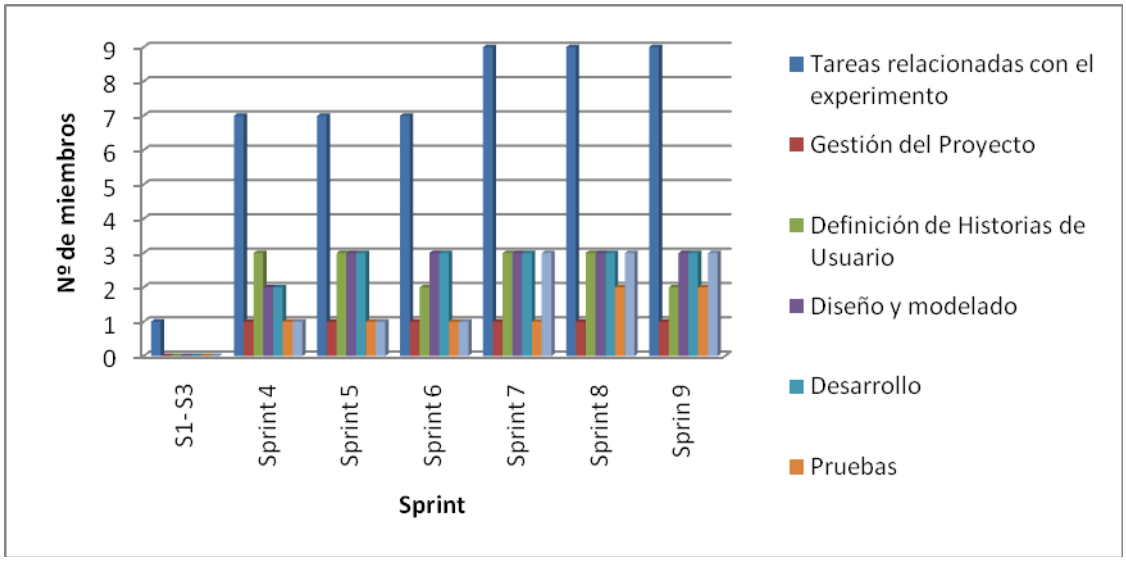


Figura 4.6 Distribución del equipo a lo largo del tiempo.

Otro factor sociológico crítico en las metodologías ágiles, en términos de calidad y productividad, es la experiencia del equipo en industria software y, concretamente, en desarrollo ágil, así como su experiencia en el dominio de desarrollo. Los valores de la experiencia están medidos en número de miembros del equipo que tienen una serie de años de experiencia en las diferentes áreas medidas y aparecen en las tablas 4.3 y 4.4, y en la figura 4.7, respectivamente.

A continuación se indica la experiencia de los componentes del equipo en aquellas áreas apropiadas en función del role ejercido dentro del proyecto y en desarrollo siguiendo una metodología ágil. Asimismo, se incluye la experiencia del cliente en utilizar este tipo de metodologías como forma de trabajo.

Tabla 4.3 Experiencia del equipo en industria software en función del role ejercido

Role	**Factor**	**Valor**				
		0 años	De 1 a 5	De 6 a 10	De 11 a 15	Más de 16
Product Owner	*Desarrollo software*					☑
	Gestión de proyectos					☑
	Dirección de grupos de trabajo					☑
Scrum Master	*Desarrollo software*			☑		
	Gestión de proyectos	☑				
	Dirección de grupos de trabajo	☑				
Equipo de desarrollo						
Diseño y modelado	*Desarrollo software*					☑
	Diseño y modelado					☑
Implementación	*Desarrollo software*		☑			
	Desarrollo con Java		☑			
Pruebas	*Desarrollo software*					☑
	Testing			☑		

Tabla 4.4 Experiencia en desarrollo siguiendo una metodología ágil

Factor	**Valor**		
	Bajo	Medio	Alto
Conocimiento teórico en metodologías de desarrollo ágil	3	5	1
Experiencia práctica en desarrollo con metodologías ágiles	9		
Experiencia en pair-programming	7	2	
Experiencia del cliente en metodologías ágiles	☑		

Previamente a la introducción de SCRUM como metodología de desarrollo, el proceso de desarrollo software seguido por el grupo podría considerarse convencional. Se seguía algo cercano al estándar ECSS-E-40 [59], por lo que el equipo tiene un alto grado de experiencia en desarrollo software siguiendo metodologías convencionales. Sin embargo, aunque debido al desarrollo de proyectos como FLEXI se tiene cierto conocimiento teórico sobre metodologías ágiles, nunca se ha experimentado con ninguna de ellas. No obstante, algunos componentes del equipo están acostumbrados a trabajar en pareja por lo que la técnica de *pair-programming* ya ha sido utilizada con

anterioridad. Del mismo modo, el cliente del proyecto tampoco posee experiencia en provisión de productos siguiendo metodologías ágiles.

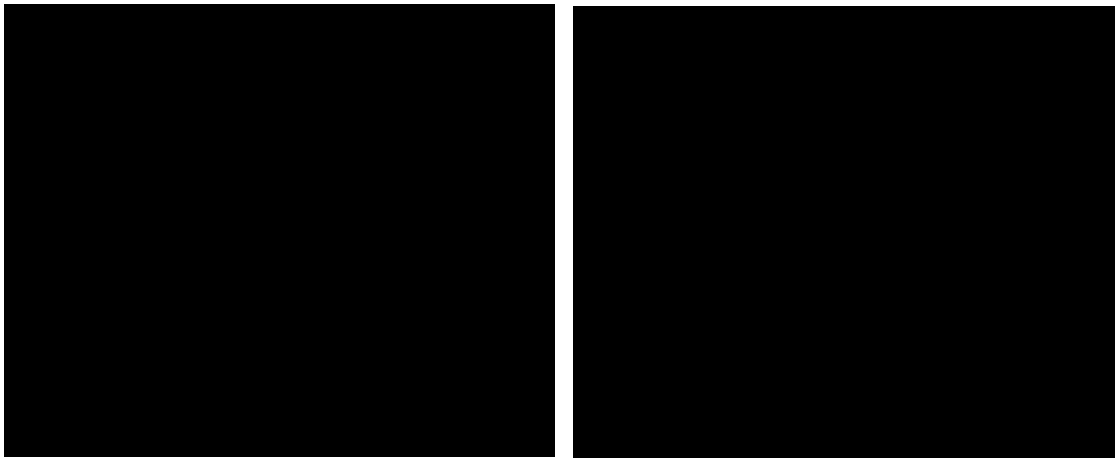


Figura 4.7 Conocimiento del dominio de desarrollo

Uno de los hechos que más positivamente repercute en el desarrollo del experimento es que parte del equipo, el 33,3%, posee un alto conocimiento en los entornos de operación, monitorización y pruebas de sistemas software complejos tras años de trabajo dentro del grupo en esta línea investigadora. En el estudio realizado en [11] se concluye que la experiencia que se tenga en el sistema que se ha de construir es más importante incluso que la experiencia en metodologías ágiles. De hecho, se estima que entre el 25 y 33% del personal involucrado en el proyecto debe ser experimentado en el sistema a construir para que el proyecto tenga éxito utilizando una metodología ágil. En línea con este trabajo, intencionadamente se ha establecido esta conformación del grupo de trabajo.

4.4. Factores Ergonómicos

Las características del entorno físico de trabajo en el que se realiza el desarrollo son fundamentales en las metodologías ágiles, ya que influyen en aspectos tales como la habilidad del equipo para adoptar técnicas propias del desarrollo ágil, como *pair-programming*, o en la fluida comunicación entre sus componentes. La figura 4.8 muestra la distribución del laboratorio en el que se ha realizado el experimento.

Se trata de un espacio amplio y luminoso en el que se pueden distinguir tres zonas:

- El espacio de la zona superior derecha, zona 1, está reservado para aquellos miembros del equipo que dedican su tiempo completo al desarrollo del experimento. Concretamente, el Scrum Master y dos miembros del equipo de desarrollo, que parte del tiempo adoptan la técnica de *pair-programming*.
- En la zona 2 se distribuye el resto del equipo. Como puede observarse en la figura, no se contemplan espacios suficientes para todos los componentes. Esto se debe a que al tratarse de un contexto universitario y, por tanto, instalado en un campus universitario, parte del equipo, con dedicación parcial al experimento, dispone de sus propios despachos, cercanos al laboratorio, donde realizan algunas tareas que no requieren una comunicación directa con el resto del equipo. No obstante, en las

situaciones en las que todos los miembros del equipo trabajan simultáneamente en el laboratorio se utiliza la zona 3.

- La zona 3 está reservada para la celebración de reuniones, tanto las propias de la metodología como aquellas reuniones colaterales que conlleva todo desarrollo software. Se trata de una zona aislada del resto del espacio, con el objetivo de que dichas reuniones, principalmente las colaterales, molesten lo menos posible al resto del equipo que no participa en ellas.

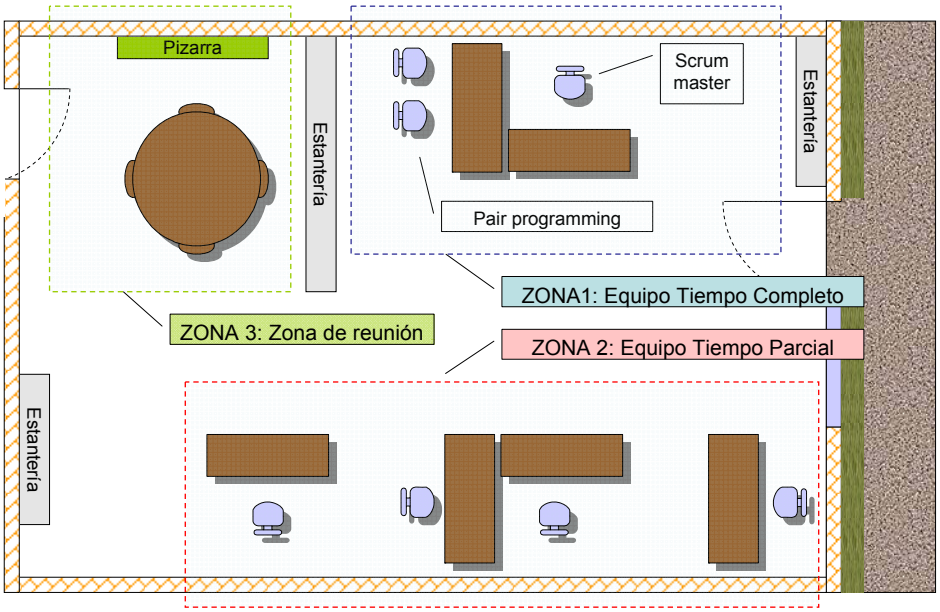


Figura 4.8 Distribución del laboratorio de desarrollo del experimento

Esta distribución favorece la fluida comunicación entre aquellos componentes que se dedican única y exclusivamente al desarrollo del experimento. Además, atenúa las posibles distracciones, provenientes de la interacción verbal entre el equipo. Se pretende conseguir un equilibrio entre la distracción que puede suponer un entorno tan colaborativo como el que proponen las metodología ágiles, que podría reducir la concentración y tener efectos negativos en la productividad del equipo, y el extremo de aislamiento total de los miembros del equipo que puede tener implicaciones morales negativas en un desarrollo *agile*.

4.5. Factores Geográficos

Todo el equipo de desarrollo pertenece al grupo de investigación SYST y se encuentra localizado en el mismo edificio, la Escuela Técnica de Informática de la Universidad Politécnica de Madrid. De los nueve componentes que forman el equipo seis se encuentran de forma permanente en el laboratorio y el resto trabaja tanto en el laboratorio como en sus propios despachos, cercanos al laboratorio, en función del grado de interacción que requiera la tarea a realizar. Cabe destacar que uno de los miembros del equipo, con dedicación parcial, trabaja de forma distribuida dos días en semana.

El trabajo con el cliente también se realiza de forma distribuida. Esta forma de trabajar es una adaptación de la metodología ágil que tiene en cuenta alguna de las limitaciones que realmente existen al desarrollar un proyecto software, más con una

distancia geográfica notable entre el equipo de desarrollo y los clientes, pues estos tienen establecido su negocio fuera de Madrid.

4.6. Factores Tecnológicos

Los métodos de desarrollo ágil ponen especial énfasis en la comunicación "cara a cara" y en los resultados, más que en la generación de documentación y, por eso, muchos de sus procesos pueden llevarse a cabo prácticamente sin herramientas. Esta búsqueda de sencillez no ha impedido que aparezcan en el mercado aplicaciones específicas para trabajar con metodologías ágiles que prometen facilitar la realización de algunas tareas. En esta sección se presentan las diferentes herramientas que se han utilizado durante el desarrollo del experimento. Se podrían clasificar en seis grupos:

- herramientas para la gestión del proyecto,
- herramientas de gestión de la configuración,
- herramientas de desarrollo,
- herramientas de soporte a la diseminación de información,
- herramientas de interacción y
- finalmente, herramientas de soporte para la toma de datos para mediciones asociadas al experimento.

A continuación, se describen detalladamente cada una de ellas. Cabe destacar que muchas de las herramientas utilizadas en el desarrollo resultan desconocidas para los componentes del equipo por lo que el proyecto se ha utilizado para pasar de un entorno de herramientas a otro, no tan conocido.

Herramientas para la gestión del proyecto

Rally [60]**.** Debido a que la interacción con el cliente y, en ciertas ocasiones, entre el propio equipo de desarrollo, es distribuida, se consideró la opción de utilizar una herramienta para la gestión del proyecto que permitiese una eficaz y fluida comunicación de la información entre todas las partes. Entre el amplio abanico de herramientas para la gestión de proyectos que ofrece actualmente el mercado, se ha optado por utilizar herramientas específicas para trabajar con metodologías ágiles, de tal forma que el carácter *agile* del proyecto no se vea mermado por la inclusión de herramientas.

La herramienta seleccionada se denomina Rally. Proporciona un sistema centralizado para la gestión de productos con metodologías ágiles. Su aplicación cubre todo el ciclo del proyecto, y permite crear un flujo entre la labor de los gestores, desarrolladores y responsables de testing. Se centra en las siguientes funciones: organización del proyecto, reporte y control, planificación y seguimiento, gestión de calidad, colaboración de equipos, integración, y administración de permisos.

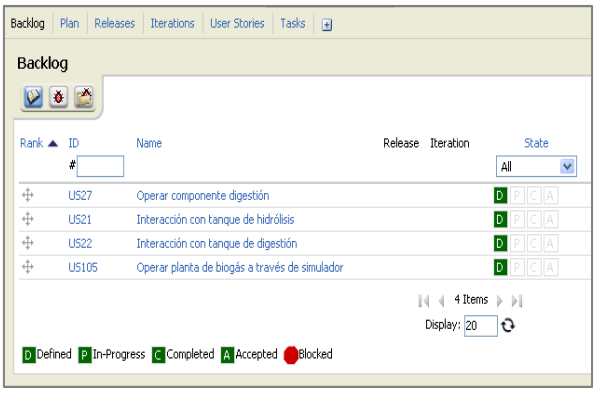


Figura 4.9 Gestión del Product Backlog en Rally

A través de Rally, el cliente puede estar en continuo contacto con el proyecto, aún trabajando de forma distribuida, y conocer en todo momento su estado. Del mismo modo, todos los componentes del equipo pueden conocer en cada momento la evolución del producto y el estado en el que se encuentran las historias de usuario que lo forman. Además, la herramienta genera automáticamente gran parte de la documentación, útil para el posterior mantenimiento del producto. Como aspecto negativo, se puede criticar que no permite administrar eficientemente todos los documentos que surgen al desarrollar un producto software con una metodología ágil por lo que no se trata de una solución completa para la gestión del proyecto.

Herramientas de gestión de la configuración

SVN subversion [61]. El trabajo colaborativo, la integración continua y la gestión de *releases* son conceptos clave en todo desarrollo ágil, más aún al tratarse de un proyecto de evolución del producto. De ahí, la importancia de contar con un software que permita una eficiente gestión de aquellos documentos que están siendo modificados a la vez por diferentes componentes del equipo y de los que, por lo tanto, se obtienen diversas versiones a lo largo del proyecto. SVN subversión es un software de sistema de control de versiones que facilita la administración de las distintas versiones del producto y permite analizar cómo éste evoluciona a lo largo del proyecto. Las distintas versiones del código del producto han sido almacenadas en el servidor de tal forma que se ha podido analizar a posteriori la evolución del producto y el impacto que las historias de usuario incorporadas han tenido en el código.

Herramientas de desarrollo

Eclipse [62]. El lenguaje de programación en el que se desarrollan las dos aplicaciones base de esta investigación es Java. Eclipse ha sido el entorno de desarrollo elegido para implementar los productos por ser de código abierto, fácil e intuitivo de usar, y uno de los más utilizados en la actualidad.

IntelliJ. Además, dado que el entorno de programación utilizado hasta el momento por parte del equipo es IntelliJ, se ha seguido utilizando también este entorno para aprovechar la familiarización que esta parte del equipo tiene con el mismo.

Herramientas para diseminación de información

eGroupWare [63]. El hecho de que las metodologías ágiles simplifiquen las tareas de gestión y documentación en los proyectos, no impide que aparezca documentación que deba ser administrada. Dicha documentación no tiene por qué ser electrónica, sino que pueden ser documentos escritos o dibujados a mano que, posteriormente, sean escaneados para una gestión más eficiente. Independientemente del tipo de documento, la cohesión del equipo de desarrollo se ve favorecida si dicha información está disponible para todos, pues ayuda a que cada componente perciba el proyecto como algo propio. Además, dado que en el caso concreto de este experimento, todos los miembros del equipo ayudan en todas

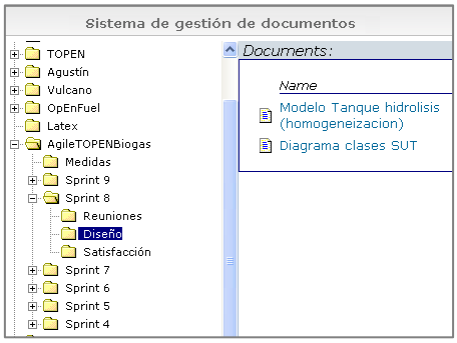


Figura 4.10 Gestión de documentos con eGroupWare

las tareas de desarrollo del mismo, es imprescindible que dicha documentación esté accesible.

eGroupWare es una herramienta vía web orientada al trabajo en grupo, que permite la gestión de calendarios, documentos, etc. En el proyecto se ha utilizado como herramienta de soporte para administrar la documentación que emergía durante del desarrollo.

eGroupWare, Sistema de gestión de documentos. Esta opción de la herramienta permite administrar documentos dentro de un sistema de archivos personal (ver figura 4.10). De tal forma que aquella documentación que, por su naturaleza, no tiene cabida en la herramienta de gestión de proyectos ágiles Rally, se ha clasificado por sprints en eGroupWare.

Herramientas para interacción entre los componentes del equipo

Dado que gran parte del trabajo con el cliente se realiza de manera distribuida y que algunos componentes del equipo no se encuentran el 100% de su tiempo de dedicación al proyecto físicamente en el mismo laboratorio, aunque sí en las mismas instalaciones universitarias, ha surgido la necesidad de utilizar herramientas que permitan una eficiente comunicación "*seudo* cara a cara". Estas son:

- Teléfono
- Correo electrónico
- Chat
- Skype. Software que permite hacer llamadas sobre internet de tal forma que se puede escuchar la voz y ver la imagen del interlocutor.
- Wiki. Se ha utilizado una wiki para notificar ciertas informaciones y hacer público el avance del proyecto. La wiki se ha creado a través de la opción para wikis que ofrece la herramienta eGroupWare.

Herramientas para la toma de datos para mediciones

Se persigue un doble objetivo al utilizar herramientas en el proceso de toma de métricas. Por un lado, disminuir el impacto sobre el desarrollo software que supone para el equipo la toma de métricas. Las medidas son imprescindibles para el posterior análisis del experimento pero conllevan un sobreesfuerzo asociado, más aún, considerando las características de un entorno *agile,* que se intenta minimizar, pues puede poner en peligro la veracidad del experimento. Por otro lado, se pretende que el proceso de toma de métricas sea lo más riguroso posible de tal forma que los datos sean fidedignos y útiles para su posterior análisis.

Microsoft Excel. Esencialmente, los datos se han recogido a través de hojas de cálculo Microsoft Excel diseñadas para el experimento. Excel permite un manejo fluido y sencillo de la información. Además, se trata de una aplicación conocida por todo el equipo de desarrollo disminuyendo, de este modo, la resistencia inicial que suele llevar asociado un plan de métricas, y anulando el coste de aprendizaje que supone el introducir una herramienta desconocida.

eGroupWare, Hoja de Presencia. Por otro lado, para registrar el tiempo dedicado al experimento se ha utilizado la opción *Hoja de Presencia* de la herramienta eGroupWare que permite a los desarrolladores introducir está información en una base de datos. La herramienta permite contabilizar el tiempo dedicado a las diferentes tareas

que se llevan a cabo en el desarrollo de un producto software, asociando esta información al sprint correspondiente.

Practiline Source Code Line Counter. Finalmente, para medir aspectos como el tamaño del código se ha utilizado esta herramienta. Permite distinguir entre el número de líneas de código fuente, el número de líneas de comentarios, el número de líneas en blanco y el número de líneas totales. En el estudio, para realizar estas métricas se han seleccionado el número de líneas de código fuente (eliminando por tanto líneas en blanco y de comentarios).

Capítulo 5

Diseño del Experimento

Una vez establecidos los objetivos del experimento y analizado el contexto en el que se llevará a cabo, es necesario planificar cómo se ejecutará el experimento, es decir, realizar un diseño del mismo. En este capítulo se presenta el diseño que se ha realizado del experimento. Atendiendo a las recomendaciones establecidas en [64], [65] y [53], se han considerado los siguientes aspectos:

- *Mediciones.* Diseño de un plan de mediciones que se utilizará para recoger los datos empíricos que permitan analizar el experimento. De este plan de mediciones se deducen las variables dependientes e independientes.
- *Plan de Ejecución*. Planificación inicial que se utilizará para ejecutar el proyecto y observar, a través de datos empíricos, los acontecimientos relativos a la evolución del producto.
- *Amenazas a la validación del experimento*. Factores que pueden limitar la habilidad para interpretar u obtener conclusiones del estudio de los datos poniendo en peligro el resultado del experimento.

5.1. Definición de Mediciones

En este apartado se presenta el plan de mediciones desarrollado para evaluar el experimento, concretamente, veremos los datos que serán recogidos como parte del estudio y la metodología seguida para su recolección. El objetivo principal es disponer de datos objetivos y consistentes que nos permitan obtener conclusiones basadas en hechos rigurosos y no en simples intuiciones u observaciones subjetivas. Dada la naturaleza de la investigación, uno de los criterios fundamentales que se han considerado en el diseño del plan de mediciones es que el proceso de recogida de las

mismas sea lo más ligero posible, de tal forma que no obstruya, o impacte en la menor medida posible, las actividades diarias del equipo ágil.

Las entidades de interés para dar respuesta a las cuestiones planteadas como objetivos del experimento y que, por tanto, van a ser medidas en él se pueden clasificar en tres grupos:

- Las *entradas o recursos* con los que se dota el desarrollo del experimento que, evidentemente, influirán en los resultados obtenidos en el mismo.
- Los *productos obtenidos*, concretamente la evolución de los productos en los diferentes sprints de los que consta el experimento, a medida que se consolida la aplicación de la metodología, y la calidad de estos productos obtenidos.
- Las actividades que forman el *proceso* de desarrollo del experimento y la productividad del mismo.

Para el análisis de dichas entidades, los datos serán recogidos de tres fuentes de información: registros con información automatizada, cuestionarios y observaciones en el lugar de trabajo.

A continuación, la tabla 5.1 muestra, a modo de resumen, el conjunto de mediciones a recoger junto con los miembros del equipo implicados en su recogida. En este diseño se han considerado las típicas medidas para este tipo de estudios como el tamaño [10], los defectos encontrados en los productos o la satisfacción de los implicados en el proceso [55], la experiencia del equipo [54] o medidas especialmente indicadas para experimentos sobre metodologías ágiles [66]. Posteriormente, se describe en detalle cada una de estas mediciones. Este conjunto de medidas permitirá dar respuesta a las cuestiones planteadas como objetivo de la tesis de máster.

Tabla 5.1 Mediciones del experimento e involucrados en su recogida

Entidad de interés	Medición	Involucrado en la recogida
Entradas o recursos	Experiencia del equipo	Todo el equipo (recogida de datos en función del role desempañado)
	Esfuerzo dedicado al proyecto	Todo el equipo
Productos obtenidos	Defectos en cada sprint	Scrum Team (persona dedicada a pruebas)
	Defectos en productos entregados	Cliente, Product Owner
Productos obtenidos	Nº de líneas de código del producto por sprint	Scrum Team (desarrolladores)
	Clases impactadas por cada historia de usuario	Scrum Team (desarrolladores)

Entidad de interés	Medición	Involucrado en la recogida
Proceso	% de realización del Product Backlog	Product Owner junto con Scrum Master
	% de realización del objetivo del sprint	Product Owner junto con Scrum Master
	% de realización de cada historia de usuario por sprint	Product Owner junto con Scrum Master
	Nº de sprints hasta completar una historia de usuario	Product Owner
	Historias de usuario repriorizadas	Product Owner
	Satisfacción en cada sprint	Todo el equipo incluido el cliente
	Satisfacción global con la metodología	Todo el equipo incluido el cliente

A continuación se describe detalladamente cada una de ellas y el método a través del cual van a ser recogidas.

Experiencia del equipo. Al comienzo del experimento se medirá el conocimiento y experiencia de cada uno de los miembros del equipo en los siguientes términos: grado académico, experiencia en desarrollo software, experiencia en gestión de proyectos, experiencia en el lenguaje de programación Java, por ser el utilizado en el desarrollo, y experiencia en desarrollo agile y *pair-programming*. Además, dado que estamos evolucionando un producto ya existente será de interés medir el grado de conocimiento del producto (TOPENprimer) y el grado de conocimiento del dominio al que se pretende adaptar dicho producto (plantas de biogás). Dichas mediciones se tomarán también, si se diese el caso, en la incorporación de nuevos miembros al equipo. La unidad de medida a utilizar en los datos de experiencia será el año o, en su defecto, el mes. Los grados de conocimiento podrán tomar los valores alto, medio o bajo. Cabe destacar que aunque esta información será solicitada a todos los miembros del equipo, pues al tratarse de un desarrollo ágil y ser muy alto el grado de interacción entre los componentes, el conocimiento se transmite fácilmente de unos miembros a otros y es, por tanto, interesante conocer estos aspectos en todos los componentes. No obstante, las distintas medidas de experiencia serán tratadas según el role ejercido de cada componente pues, por ejemplo, es más importante la experiencia que un desarrollador tenga en el lenguaje de programación Java que en gestión de proyectos. Esta información será registrada a través de cuestionarios electrónicos.

Esfuerzo dedicado al proyecto. Es interesante registrar las horas, o en su defecto minutos, que se dedican a realizar cualquier actividad relacionada con la adaptación de TOPENprimer al dominio del biogás. Entre otras utilidades, estas mediciones servirán para analizar los objetivos obtenidos a lo largo del proceso respecto al esfuerzo dedicado al mismo. Además permitirá recapacitar sobre el proceso de aprendizaje de la metodología y la posible influencia del esfuerzo dedicado a cada aspecto de la misma. Esta medición está dirigida a todos los miembros del equipo, incluido el cliente. Para facilitar la recogida de la misma se utilizará la opción *Hoja de*

Presencia de la herramienta eGroupWare al finalizar cada tarea. En ella se deberá indicar la siguiente información:

- Seleccionar el *Identificador del proyecto*, por ejemplo: P-2008-0002:Sprint 4
- Indicar el *Tipo de la tarea*, ya que para facilitar la posterior gestión, las mediciones de tiempo se dividirán en tareas. Este campo debe comenzar por alguno de los acrónimos que aparecen a continuación, seguido de dos puntos y el nombre de la tarea.

  Ejemplo: **D: Análisis del comportamiento del sistema**

  Acrónimos:
  - **EC** = *Environment Configuration* (tareas relacionadas con el acondicionamiento del entorno para utilizar la metodología agile: gestión de herramientas, gestión del equipo, etc.)
  - **PB** = *Gestión del Product Backlog*
  - **US** = *Especificación de historias de usuario*
  - **D** = *Diseño*
  - *Continuous integration*
    - **CIC** = *codificación*
    - **CIP** = *pruebas*
  - **RM** = *Release Management*
  - **TS** = *Technical Support*
  - **DOC** = *Documentación*
  - **AT** = *Agile Tasks* (otras tareas relacionadas con agile. Por ejemplo, reuniones)
  - **O =** *Otras tareas relacionadas con el experimento*
- Indicar la *Fecha* de recogida de la métrica,
- *Hora de comienzo de la tarea*
- *Duración* (medido en minutos u horas si es un valor exacto), y
- *Notas* (descripción de la tarea, si fuese necesario)

Defectos en cada sprint. Al finalizar cada sprint, con el objetivo de evaluar la calidad del producto obtenido, se medirán el número de defectos del mismo, así como la importancia y el origen de estos defectos. Esta métrica será recogida por la persona del Scrum Team encargada de realizar las pruebas a través de una hoja Excel diseñada con el fin de recoger estas mediciones. En esta hoja deberá indicar la fecha de recogida de la medida, el identificador del sprint así como el identificador del defecto encontrado, una descripción del defecto y una estimación de su importancia (alta, media o baja) y la causa dónde tuvo origen.

Defectos en productos entregados. Asimismo, es de interés conocer los defectos encontrados por el cliente cuando el producto ya ha sido entregado. La continua interacción con el cliente facilitará la toma de esta medición. La información a recoger será similar a la de la métrica anterior, utilizando de nuevo la hoja Excel. Sin embargo, en esta ocasión el encargado de toma la medida será el cliente en colaboración con el Product Owner.

Número de líneas de código del producto por sprint. Para evaluar la evolución del producto, al finalizar cada sprint se anotarán el número de líneas de código del producto construido hasta ese momento, tanto para TOPENbiogas como para

el simulador de la planta. Estas mediciones quedarán tomadas automáticamente cuando los desarrolladores suban las distintas versiones del producto a la herramienta subversión diariamente. De esta forma, se podría analizar incluso la evolución del producto cada día.

Clases impactadas por cada historia de usuario. Con propósito similar al de la métrica anterior, se van a medir las clases que sufren algún tipo de impacto al evolucionar el producto TOPENprimer al dominio de las plantas de biogás. Esta medición se tomará en función de la historia de usuario que este siendo desarrollada. El desarrollador, apoyándose en el software DOORS y subversion, indicará el identificador del sprint, de la historia de usuario y de la clase que está siendo impactada, especificando, asimismo, el tipo de impacto sufrido (clase nueva, clase modificada o clase eliminada) y la fecha de recogida de la medida.

Porcentaje de realización del Product Backlog. Para medir la productividad al utilizar una metodología de desarrollo ágil como SCRUM, se tomarán diferentes mediciones del porcentaje de realización del producto. El porcentaje de realización del Product Backlog será una de ellas. Esta medida será tomada por el Product Owner en colaboración con el Scrum Master al finalizar cada sprint y al finalizar el proyecto. Este porcentaje será calculado como se muestra a continuación:

- *Elemento del Product Backlog (BE)*
- *Número de elementos del Product Backlog realizados (ABE)*
- *Número de elementos totales del Product Backlog (TBE)*

$$\%\ realización\ backlog\ = \frac{(\sum_{i=0}^{ABE} BEi * pi\ ) * 100}{(\sum_{i=0}^{TBE} BEi * pi\ )}$$

pi: peso asignado al elemento del Product backlog ya que puede constituir un objetivo primario o secundario

Los datos serán recogidos en las hojas Excel diseñadas para tomar las medidas.

Porcentaje de realización del Objetivo del Sprint. También se recogerá el porcentaje de realización del objetivo marcado para cada sprint. La medición se tomará de forma similar a la anterior, siendo la fórmula de cálculo la que aparece a continuación:

- *Historia de usuario del sprint (US)*
- *Número de historias de usuario totales (TUS)*
- *Número de historias de usuario completadas (CUS)*

$$\%\ realización\ del\ sprint\ = \frac{(\sum_{i=0}^{CUS} USi * pi\ ) * 100}{(\sum_{i=0}^{TUS} USi * pi\ )}$$

pi: peso asignado a la historia de usuario dentro del sprint ya que puede ser un primaria o secundaria

Porcentaje de realización de cada Historia de Usuario por Sprint. Para que el análisis sea más completo, profundizando en un nivel mayor de detalle en la productividad de cada sprint, se medirá el porcentaje de realización de cada historia de

usuario en el sprint en el que se define, así como, si no está completa en ese sprint, en sprints posteriores.

- *Número total de tareas que componen la historia de usuario (TT)*
- *Número de tareas completas (CT)*

$$\%\ realización\ de\ una\ historia\ de\ usuario\ = \frac{CT*100}{TT}$$

Número de sprints hasta completar una historia de usuario. Dado que puede suceder que las historias de usuario no se completen en el sprint planificado sino que aparezcan retrasos, en alineación con la medición anterior, se medirán estos retrasos contabilizando el número de sprints que se dedican a cada historia de usuario ya que, teóricamente, cada historia de usuario debería ser definida de tal forma que fuese capaz de desarrollarse en un único sprint. Esta medida será recogida al final del proyecto por el Product Owner en las plantillas Excel habilitadas para las mediciones, indicando el identificador del sprint en el que se incorpora la historia de usuario y el identificador del sprint en el que finalmente es aceptada por el cliente.

Historias de usuario repriorizadas. Uno de los aspectos más interesantes de las metodologías ágiles es su capacidad de adaptación a los cambios que se produzcan en el entorno. Por este motivo será interesante medir los cambios acontecidos a través de las historias de usuario que sean repriorizadas en el proyecto. Esta medida la tomará el Product Owner a través de la plantilla Excel.

Satisfacción en cada sprint. Finalmente, puesto que las metodologías ágiles están enfocadas a una nueva filosofía de desarrollo, resulta muy interesante conocer la opinión, percepción y observaciones tanto de los componentes del equipo como del cliente con esta nueva forma de trabajo. Así al finalizar cada sprint deberán rellenar un cuestionario acerca de estos aspectos en los que deberán realizar una breve descripción con reflexiones personales sobre la metodología seguida y valorar su grado de satisfacción (valor comprendido entre 0, satisfacción mínima, y 10, satisfacción máxima). Al finalizar el proyecto, del mismo modo, se rellenará un cuestionario similar con la **satisfacción global de la metodología.**

5.2. Plan de Ejecución

Dado que las metodologías ágiles no siguen planificaciones detalladas a medio y largo plazo sino que apuestan por adaptarse a las condiciones del mercado mediante planificaciones de ciclos de desarrollo muy cortos, en el proyecto de adaptación de TOPENprimer a TOPENbiogas, presentado en este estudio, tampoco se realizarán este tipo de planificaciones. No obstante, el estudio que engloba este proyecto si ha sido detalladamente planificado, con objetivo de cumplir unos tiempos y objetivos para la realización de la tesis de máster.

Se trata de una planificación aproximada que puede sufrir pequeñas variaciones a lo largo del desarrollo. Se ha considerado dividir en tres partes el estudio:

1) En la parte inicial del experimento (t_0 - t_0+7) se realizará la puesta a punto del mismo, consistente en analizar bibliografía relativa a las áreas principales en las que se enmarca el experimento y estudios empíricos

que ya se hayan realizado en esta línea, y configurar el entorno de desarrollo del experimento.

2) Posteriormente, se ejecutará el experimento en sí, utilizando la metodología SCRUM en el desarrollo. Inicialmente, se ha planificado que esta parte conste de siete sprints de una duración aproximada de dos semanas cada uno, aunque se contemplan posibles variaciones en este sentido en función de los acontecimientos propios que tengan lugar en el desarrollo. No obstante, a pesar de cualquiera de estos acontecimientos esta parte tiene una duración limitada acotada entre el t_0+6 y t_0+22.
3) La tercera y última parte considerada se dedicará a la evaluación y obtención de resultados. Dada la envergadura del experimento a realizar y las limitaciones temporales de esta tesis de máster, se pretende presentar en la misma las primeras conclusiones obtenidas tras un mes aproximado de análisis de resultados, aunque se pretende profundizar en el análisis del experimento posteriormente.

A continuación, en las tablas 5.2, 5.3 y 5.4, se visualizan gráficamente las planificaciones de las distintas partes que forman el desarrollo temporal del experimento mediante diagramas de Gantt de la parte primera, segunda y tercera respectivamente.

Tabla 5.2 Diagrama Gantt primera parte del experimento: análisis preliminar e identificación del problema.

Planificación del experimento (PARTE I Identificación del problema)								
Nombre	Mes1				Mes 2			
	1ª Sem	2ª Sem	3ª Sem	4ª Sem	1ª	2ª	3ª	4ª
1. Análisis preliminar e identificación del problema								
Estudio preliminar de SCRUM								
Sprint1: Estudio agile, especialmente SCRUM								
Sprint2: Análisis estudios empíricos agile								
Sprint3: Configuración entorno de ejecución								

Tabla 5.3 Diagrama Gantt segunda parte del experimento: desarrollo del proyecto de evolución.

Planificación del experimento (PARTE II Desarrollo del proyecto de evolución)																				
Nombre	Mes2				Mes3				Mes4				Mes5				Mes6			
	1ª	2ª	3ª	4º	1ª	2ª	3ª	4ª	1ª	2ª	3ª	4ª	1ª	2ª	3ª	4ª	1ª	2ª	3ª	4ª
2. Desarrollo del proyecto																				
Sprint 4																				
Sprint 5																				
Sprint 6																				
Sprint 7																				
Sprint 8																				
Sprint 9																				
Sprint 10																				

Tabla 5.4 Diagrama Gantt tercera parte del experimento: evaluación y obtención de resultados.

Planificación del experimento (PARTE III Análisis de resultados)												
Nombre	Mes6				Mes7				Mes8			
	1ª	2ª	3ª	4º	1ª	2ª	3ª	4ª	1ª	2ª	3ª	4ª
3. Evaluación y obtención de resultados												
Recuperación trabajo pendiente												
Evaluación de la realización del proyecto												
Obtención resultados finales												
Elaboración libro con resultados de tesis												

5.3. Posibles Amenazas para la Validación del Experimento

Un problema potencial en cualquier estudio que se base en un experimento son las denominadas amenazas al experimento. En esta sección se describen las amenazas identificadas que pueden poner en peligro la validación de este experimento y como han sido minimizadas o anuladas. Se han considerado cinco amenazas: 1) proyecto desarrollado, 2) proceso de desarrollo, 3) sujetos implicados en el desarrollo, 4) disponibilidad de los sujetos implicados en el desarrollo y 5) toma de métricas.

La mayoría tienen origen en el contexto en el que se realiza el estudio, pues se aparta, en algunos sentidos, de los cánones que establecen las metodologías ágiles, en muchas ocasiones casi utópicos, y se acerca a la realidad del mercado. Por tanto, podríamos considerarlas más bien como amenazas que acechan a las propias metodologías ágiles.

1) **Proyecto desarrollado**

   El experimento se ha realizado sobre un único proyecto, evolucionar TOPENprimer para obtener TOPENbiogas y puede que, por tanto, durante el desarrollo no hayan surgido todas las situaciones que se pueden dar en todos los proyectos. Para minimizar el impacto que esta amenaza pueda tener en la habilidad para generalizar los resultados del estudio en la industria, en el contexto de la investigación, se ha proporcionado una completa descripción tanto del producto de partida como del producto obtenido que permita a otros investigadores replicar el estudio.

2) **Proceso de desarrollo**

   Uno de los principales inconvenientes que los investigadores se encuentran a la hora de abordar una investigación sobre metodologías ágiles es el propio proceso de desarrollo utilizado. El hecho que las metodologías ágiles aboguen por la auto-organización de los equipos supone una amenaza para las investigaciones pues es muy difícil generalizar el proceso seguido. En el caso concreto de esta investigación, se han seguido de una forma muy estricta los aspectos teóricos y las directrices marcadas por la metodología SCRUM para minimizar esta amenaza, describiendo detalladamente como ha transcurrido el desarrollo y las ventajas e inconvenientes que ha proporcionado al proyecto. Todos los aspectos relacionados con la auto-organización del equipo, que no

están, por tanto, recogidos en los libros pues son propios del proyecto concreto, también han sido cuidadosamente descritos en el contexto de la investigación.

3) Sujetos implicados en el desarrollo

Dado el contexto universitario en el que se desarrolla la investigación puede que se llegue a considerar que los sujetos implicados en el desarrollo no son representativos de los profesionales que trabajan en la industria software. No obstante, la mayoría de los miembros del equipo tienen un alto grado de experiencia en la industria, tal y como se recoge en el contexto del experimento, adquirido durante distintas fases de su carrera profesional.

4) Disponibilidad del equipo de desarrollo y del cliente

Las metodologías ágiles establecen la completa disponibilidad tanto del equipo de desarrollo como del cliente en el proyecto. Es decir, de acuerdo a la filosofía ágil llevada, tal vez, al extremo, el equipo de desarrollo únicamente debería dedicarse a la realización del proyecto y el cliente debería encontrarse el 100% del tiempo de desarrollo físicamente en el lugar de trabajo. Evidentemente, no es el caso del contexto de esta investigación ni para el equipo de desarrollo ni para el cliente, pues la mayoría de los desarrolladores tienen otras responsabilidades docentes y de investigación, y al cliente no le es posible permanecer físicamente en el lugar de desarrollo de forma permanente, debido principalmente a la distancia geográfica existente entre ambos, pues su negocio no se encuentra en Madrid. No obstante, estas características hacen que el proyecto desarrollado sea más representativo de la realidad pues en el mercado actual un alto porcentaje de organizaciones desarrolla proyectos pequeños y sus empleados deben compaginar más de un proyecto a la vez. Tampoco es rentable para una organización disponer de una persona muy válida que se centre en un único proyecto y no realice otro tipo de tareas en la organización como, por ejemplo, labores de apoyo en otros proyectos. Por otro lado, la idea de total disponibilidad del cliente en la actualidad es casi una utopía. Bien es cierto, que la mayoría de las metodologías ágiles no son tan estrictas y, aunque aconsejan seguir estos principios, contemplan la posibilidad de introducir variaciones.

5) Toma de mediciones

Finalmente, una de las mayores amenazas a la que se ha enfrentado el experimento es la recogida de métricas para el posterior análisis de resultados. Las metodologías ágiles se centran en producir solamente aquello que aporte un valor directo al negocio minimizando aspectos tales como la documentación. Desde una perspectiva puramente industrial la toma de métricas no aporta ningún valor al proyecto, sin embargo, para el experimento resultan fundamentales. Para minimizar el impacto de este factor se ha diseñado un plan de métricas especialmente pensado para que su interferencia en el proceso ágil sea lo más leve posible.

Capítulo 6

Desarrollo del Experimento

En este capítulo se describe el desarrollo del experimento que da base al estudio. Finalmente, consta de nueve sprints, por lo que se desvía levemente de la planificación inicial, en los que, por un lado, se diseñará el experimento y, por otro, se evolucionará TOPENprimer para construir un entorno para la monitorización, operación y validación del proceso de producción de biogás de la planta del cliente para obtener energía eléctrica y bio-fertilizantes, y un simulador de dicha planta.

A lo largo de este capítulo se describirán los hechos más destacables de cada uno de los sprints desarrollados respecto a la evolución del producto, así como la influencia y resultado de las prácticas ágiles utilizadas y de cada uno de los elementos para la gestión del proyecto propuestos por SCRUM. La figura 6.1 refleja el momento de comienzo y finalización para cada sprint completado en el proyecto. Como se puede apreciar el experimento en el que se basa el estudio tiene una duración total aproximada de cinco meses y medio (22 semanas).

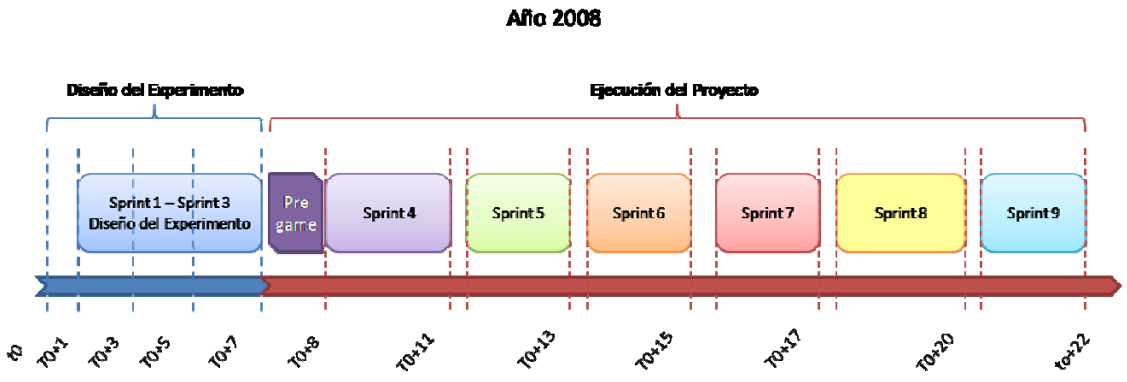


Figura 6.1 Línea en el tiempo del experimento

6.1. Sprint 1 - Sprint 3. Diseño del Experimento

El análisis de literatura especializada sobre estudios empíricos en ingeniería del software revela que a menudo estos trabajos no poseen una alta calidad pues están perjudicialmente caracterizados por un pobre diseño estadístico, una imposibilidad de escalar a otros sistemas debido a la ausencia de una información contextual adecuada y una breve ejecución desde el punto de vista temporal que invalida las conclusiones de los autores [53]. Un aspecto fundamental en este sentido es que muchos estudios empíricos carecen de hipótesis simples, no plantean cuestiones y no tienen un final claramente definido. En esta situación, los investigadores solamente pueden presentar observaciones sobre los datos.

Con el objetivo de que el estudio del contexto y diseño del experimento presentado en esta tesis fuese efectivo y quedasen claras las pretensiones del estudio se destinó un alto porcentaje del tiempo (aproximadamente el 20%) a un análisis preliminar e identificación del problema. Concretamente los tres primeros sprints, aunque de periodos de tiempo más breves que los restantes. Se pretendía realizar un detallado análisis previo que permitiese obtener interpretaciones rigurosas y proporcionase la información suficiente para que otros investigadores pudiesen reproducir el experimento.

El esfuerzo destinado a esta fase de diseño fue el de dos personas del equipo, director y autora de esta tesis de máster, con una media de trabajo diario de 6 horas por parte de la autora y una constante orientación por parte del director de la tesis. Con el propósito de obtener el mayor partido posible de esta etapa inicial del experimento, y dado que SCRUM es una metodología enfocada a la gestión de proyectos software pero aplicable en la gestión de cualquier tipo de proyectos, se creyó que sería un aporte positivo el poner en práctica la metodología en cuanto se tuviesen los conocimientos teóricos imprescindibles para que, de esta manera, parte del equipo tuviese un primer contacto con ella a modo de entrenamiento. De esta forma, durante esta fase de diseño del experimento se utilizó la metodología SCRUM adaptada al contexto concreto de aplicación.

Tal y como establece el ciclo de vida propuesto por esta metodología, el primer paso fue la etapa de *pre-grame*, en la que se definió el Product Backlog. Básicamente las necesidades a cubrir en esta primera fase del experimento eran:

- Estudio del concepto de desarrollo ágil
- Estudio teórico en profundidad de la metodología SCRUM
- Análisis de los estudios empíricos de aplicación de metodologías ágiles focalizándose en la evolución de productos
- Diseño y configuración del entorno de ejecución del experimento

En el sprint 1 se llevaron a cabo los dos primeros ítems del Product Backlog y parte del tercero. Durante este periodo se analizó la literatura existente sobre el movimiento ágil y, a alto nivel, sobre las distintas metodologías de desarrollo ágil que se utilizan actualmente en el mercado (Cristal, Dynamic Software Development Method (DSDM), Feature-driven Development, Lean Software Development y Extreme Programming). Posteriormente se realizó un estudio más detallado de la metodología SCRUM por ser la metodología seleccionada para el desarrollo del experimento, considerando cómo podría ser adaptada al contexto de investigación en el que se iba a desarrollar el experimento. Además se comenzó a avanzar en el tercer objetivo de esta

fase, abordando el proceso de búsqueda y análisis de estudios empíricos que tuviesen como protagonista a las metodologías ágiles.

El segundo sprint se prosiguió el cuidadoso análisis de los estudios empíricos de aplicación de metodologías de desarrollo ágil que se habían realizado hasta el momento. Aunque a lo largo del estudio se ha permanecido en constante contacto con el estado de la investigación en este campo, examinando los estudios más recientes que iban apareciendo, en un primer momento se analizaron principalmente las 36 publicaciones identificadas en [6] como aquellas de mayor calidad en desarrollo ágil en la actualidad. Llamó especialmente la atención resultados tan contradictorios como los obtenidos, por ejemplo, en los artículos [2] y [3] que, aunque se establecen en contextos diferentes y ambos aplicando eXtreme Programming en lugar de SCRUM, vierten resultados tan dispares referentes a la productividad alcanzada en el desarrollo como de +337% y -44% respectivamente. Estos datos evidenciaban claramente la necesidad de realizar nuevos estudios en este sentido.

Además, en este segundo sprint se comenzó a realizar el diseño del experimento, estableciendo, por un lado, el proceso de desarrollo de la investigación como se indica a continuación:

1. Marcar los objetivos del estudio a través de cuestiones
2. Observar el transcurso del experimento
3. Abstraer las observaciones en datos
4. Analizar los datos
5. Obtener conclusiones con respecto a los objetivos perseguidos a través de respuestas a las cuestiones planteadas

Y fijando, por otro lado, los objetivos del estudio en función de los trabajos existentes hasta el momento, las áreas de interés del grupo de investigación SYST y del proyecto Flexi, y las características del contexto en el que se desarrolla la investigación.

En el tercer y último sprint de esta parte se finalizó el diseño y configuración del entorno de ejecución del experimento. Se efectuó una concienzuda búsqueda de herramientas que diesen soporte al desarrollo ágil. Del mismo modo, se prestó especial interés en analizar las amenazas que pudiesen poner en peligro la validación del experimento. Se diseñó un plan de métricas que permitiese alcanzar los objetivos planteados y se acondicionó el entorno de trabajo tratando de equilibrar el trabajo en grupo con el trabajo individual. Además, se adaptó la metodología SCRUM al contexto concreto de desarrollo del experimento y se asignaron los diferentes roles considerados por la metodología a los miembros del equipo en función de la experiencia, disponibilidad y habilidades de éstos.

En esta parte del experimento la metodología sufrió algunas desviaciones pues al tratarse en esta etapa de un equipo tan reducido, únicamente el esfuerzo de dos personas, y en un dominio de aplicación un tanto diferente para el que fue concebido SCRUM, influyó en que muchas de las tareas no siguieran al 100% las directrices marcadas por la metodología. Así las reuniones diarias se realizaron, en realidad, cada dos días pues las tareas a desempeñar requerían más de un día de trabajo. Por otro lado, las reuniones de planificación, revisión y retrospectiva tuvieron una duración sensiblemente menor a la que teóricamente implican estas reuniones pues las dos personas involucradas en el trabajo, que a su vez eran sus propios clientes, conocían perfectamente el alcance en cada momento. No obstante, esta experiencia resultó muy útil para experimentar con esta filosofía de desarrollo y para comprender el significado

de conceptos como *pre-game*, *product backlog*, *sprint backlog* o las diversas reuniones de gestión del proyecto. Más aún considerando que la autora de la tesis ejercería posteriormente el papel de Scrum Master.

6.2. Pre-game

La etapa de *pre-game* supone el comienzo del proyecto dentro del experimento. Con el objetivo de descubrir las necesidades que debía satisfacer el sistema que iba a ser desarrollado se destinaron unos días a esta primera fase. En esta etapa de pre-game, utilizando la ayuda y documentación base aportada por el cliente, se realizó una primera definición sencilla y clara de las características que debía cumplir el sistema a evolucionar para operar, monitorizar y probar la planta de producción de biogás. Asimismo, se detallaron las propiedades que debía cumplir el simulador de dicha planta. En este estudio inicial no se pretendía obtener todos los detalles de los productos a construir, sino una primera aproximación que dirigiese el desarrollo, pues la continua interacción con el cliente serviría para dar respuesta a los interrogantes que pudieran aparecer. Del mismo modo, la propia metodología nos ayudaría a abordar los cambios que se pudiesen presentar de este análisis preliminar.

El resultado de esta etapa fue el *Product Backlog*, artefacto que, por tanto, contiene las historias de usuario a alto nivel que guiarán el proceso de desarrollo y que posteriormente, a medida que avancen los sprints, serán descompuestas en un nivel mayor de detalle. El hecho de que el producto a desarrollar fuese en realidad la evolución de otro ya existente, del cual, a su vez, se tienen identificadas las partes susceptibles de variación en la evolución, benefició en este sentido pues se conocían de antemano las historias de usuario a cumplir por la parte invariable del producto, acotando de este modo el campo de estudio.

La gestión de las historias de usuario se realizó a través de la herramienta Rally que permite su estructuración jerárquica, anidándolas en relaciones padre-hijo, facilitando de este modo dichas tareas de gestión. Además, se siguió el esquema quién, qué y por qué (ver sección 2.1.4) para definirlas. El Product Backlog resultado de esta fase, en síntesis[11], responde al contenido de la tabla 6.1.

Como puede apreciarse en dicha tabla para cada uno de los productos a construir, herramienta de operación, monitorización y pruebas TOPENbiogas y simulador de la planta de biogás, se tienen que considerar cuatro componentes:

- Tanque de trituración
- Tanque de pasteurización
- Tanque de hidrólisis
- Tanque de digestión

que corresponden a las cuatro etapas del proceso de obtención de biogás (Para más información ver figura 4.1 con el esquema de funcionamiento de la planta de biogás).

[11] Debido a la magnitud del proyecto y al gran volumen de información que se manejó en el desarrollo del mismo, no se incluye en esta tesis de máster el contenido completo de toda esta información asociada, sino que se presentan pequeños fragmentos relevantes que permiten la ilustración del experimento. Si desea más información sobre alguno de estos documentos póngase en contacto con el grupo de investigación SYST (https://syst.eui.upm.es) en el que se enmarca este estudio.

Tabla 6.1 Síntesis del Product Backlog resultado de la etapa de Pre-game

Producto	PRODUCT BACKLOG			
	Id Historia de Usuario	Usuario	Definición	Motivación
TOPEN biogás	US13	Ingeniero de monitorización y pruebas	Evolucionar TOPEN a la planta de biogás	Operar, monitorizar y probar planta de biogás
	US45	Desarrollador TOPENbiogas	Analizar estado actual de TOPEN	Construir TOPEN biogás
	US17	Ingeniero de monitorización y pruebas	Interacción con tanque de trituración	Operar, monitorizar y probar tanque de trituración de la planta
	US20	Ingeniero de monitorización y pruebas	Interacción con tanque de pasteurización	Operar, monitorizar y probar tanque de pasteurización de la planta
	US21	Ingeniero de monitorización y pruebas	Interacción con tanque de hidrólisis	Operar, monitorizar y probar tanque de hidrólisis de la planta
	US22	Ingeniero de monitorización y pruebas	Interacción con tanque de digestión	Operar, monitorizar y probar tanque de digestión de la planta
Simulador Planta de Biogás	US15	Operador planta de biogás	Operar planta de biogás a través de un simulador	Operar planta de biogás
	US23	Operador planta de biogás	Operar con el tanque de trituración	Operar tanque de trituración
	US25	Operador planta de biogás	Operar componente de pasteurización	Operar tanque de pasteurización
	US26	Operador planta de biogás	Operar componente de hidrólisis	Operar tanque de hidrólisis
	US27	Operador planta de biogás	Operar componente de digestión	Operar tanque de digestión

La visión del producto completo abarcaría la implementación en el sistema de los cuatro tanques, así como la implementación de la unidad de cogeneración a partir de la cual se obtiene electricidad. Con esta implementación el producto abarcaría todo el proceso de la planta, siendo una solución completa. Sin embargo, la versión que se desea alcanzar en la realización de este proyecto, contempla la implementación de los cuatro primeros tanques, que representaría entre un 55% y 60% del producto completo. Esta versión nos permite una consistente imagen del producto y plantea con sólidos argumentos lo que conlleva el manejo de una planta de Biogás (ver figura 6.2).

Con el objetivo de que el cliente pueda ir disfrutando de los avances que se producen en el desarrollo se acordó liberar una *release* del producto por cada tanque que fuese implementado, tanto en el producto TOPENbiogas como en el simulador. El objetivo inicial sería evolucionar TOPENprimer para implementar un tanque por sprint de desarrollo.

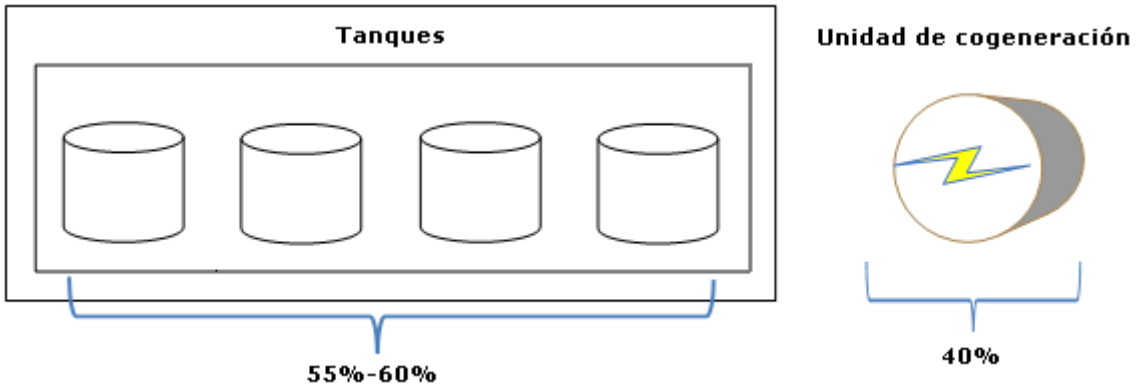


Figura 6.2 Visión del producto

6.3. Sprint 4

Tal y como establece SCRUM, el proyecto comenzó con la reunión de planificación del primer sprint, el sprint 4. Un factor importante a considerar es que en esta reunión, al igual que en la mayor parte del proyecto, la comunicación con el cliente se realizó de forma distribuida debido a la imposibilidad del cliente para encontrarse físicamente en el lugar de desarrollo. No obstante, uno de los miembros del equipo, presente *in situ* durante todo el desarrollo, el Product Owner, tenía todo el conocimiento necesario para tomar las decisiones del cliente. Esta forma de trabajar es una adaptación de la metodología que tiene en cuenta la limitación de la disponibilidad completa del cliente que habitualmente existe al desarrollar un proyecto real.

El objetivo de esta reunión fue seleccionar las historias de usuario del Product Backlog a cumplir en este sprint para comenzar a evolucionar el producto TOPENprimer. La prioridad del cliente era tener disponible una primera versión del entorno de operación, monitorización y pruebas que actuase sobre los parámetros del tanque de trituración, y un simulador de dicho tanque que permitiese comprobar el funcionamiento del entorno.

Para realizar la planificación de este sprint se descompusieron las historias de usuario implicadas, US17 y US23 respectivamente, en tareas que habría que realizar para cumplimentar dichas historias de usuario. La falta de conocimiento en el dominio a evolucionar complicó la representación de la funcionalidad a través de historias de usuario y la estimación del tiempo necesario para efectuar cada tarea. Finalmente, el equipo optó por llegar a un acuerdo basándose en intuiciones y dando prioridad a las opiniones de aquellos miembros con un mayor conocimiento en este tipo de entornos y en el producto de partida. El *Sprint Backlog* del sprint 4 quedó, por tanto, constituido por estas dos historias de usuario más la historia de usuario US45 relativa a la familiarización con el entorno (tarea vital para iniciar la evolución de un producto).

El desarrollo siguió el orden lógico de las tareas establecidas en la planificación, comenzando por una familiarización con el entorno TOPENprimer a evolucionar y con el dominio de aplicación. A medida que avanzaba el sprint comenzaron a aparecer complicaciones en la parte del diseño para evolucionar TOPENprimer, causadas principalmente por la complejidad del entorno y falta de conocimiento de los desarrolladores en el producto. La labor del Product Owner y el continuo contacto telefónico con el cliente ayudó en gran medida a mitigar estas complicaciones. No obstante, el hecho de que estas dificultades en diseño no fuesen consideradas en la

planificación del sprint propició que comenzasen a aparecer retrasos. Otro aspecto importante a considerar es que en el diseño realizado en este sprint, así como en diseños de tanques posteriores, se utilizó el perfil UM2OP para especificar de forma explícita las interacciones entre TOPENbiogas y la planta de biogás [57]. Así, se contemplaron comandos de operación sobre los atributos del sistema (comandos *get* y *set* para obtener y modificar el valor de los atributos) y notificaciones emitidas por el sistema y capturadas por el entorno TOPENbiogas.

Dos de los componentes del equipo, que practicaron la técnica de pair-programming, se dedicaron única y exclusivamente a la parte del simulador. En este caso, al tratarse de un producto que partía desde cero el avance fue más destacable, pues se evitaba la parte de familiarización con el producto a evolucionar, aunque del mismo modo se presentaron algunos retrasos surgidos también en la parte de diseño (puesto que el simulador se utilizará para validar el funcionamiento del entorno TOPENbiogas, el diseño de ambos está íntimamente ligado y caracterizado por el dominio de aplicación).

Por otro lado, el diseño de pruebas se hizo de forma simultánea al diseño e implementación, por una parte del equipo dedicado única y exclusivamente a esta labor. De esta forma las pruebas no se dejaron para el final del sprint sino que se realizaron en el instante en el que alguna de las funcionalidades estaba integrada. A medida que se agregaba funcionalidad se pasaba de nuevo toda la batería de pruebas. Cabe destacar que en este sprint los avances respecto a funcionalidad implementada se centraron únicamente en el simulador y fue, por tanto, la parte que acaparó el trabajo en pruebas.

Además, siguiendo la metodología, diariamente se realizaron reuniones en las que cada miembro del equipo debía describir el estado del trabajo que tenía designado. Se constató el beneficio que estas reuniones aportaban para detectar y solucionar posibles problemas de forma rápida en la evolución del producto. No obstante, la falta de hábito en realizar este tipo de reuniones propició que la mayoría se prolongaran más de lo previsto, convirtiéndose en reuniones de más de media hora en lugar de los 10 minutos que teóricamente debían durar.

La reunión de revisión evidenció los retrasos ocurridos. De hecho, aunque se mostraron al cliente avances en algunas tareas de diseño e implementación, no se logró concluir ninguna de las historias de usuario previstas, con lo cual los resultados respecto a la planificación establecida fueron catastróficos.

Tras la reunión de revisión se llevó a cabo la reunión retrospectiva. En ella se trataron las numerosas dificultades a las que nos habíamos enfrentado en este primer sprint, principalmente motivadas por la falta de conocimiento en el dominio y producto a evolucionar. Las más significativas fueron:

- El gran lastre de este sprint fue la dificultad con la que nos encontramos para definir las historias de usuario y realizar las estimaciones de tiempo para cada una de ellas. Los problemas detectados siguieron básicamente tres vertientes. Por un lado, la dificultad que supuso establecer el nivel de detalle en la definición de las historias de usuario. Concretamente, la definición se realizó a un alto nivel de detalle lo que provocó una estimación imprecisa en tiempo. Por otro lado, la ausencia de datos históricos influyó decisivamente en que las estimaciones fuesen imprecisas. Desconocíamos cuánto trabajo éramos capaces de llevar a cabo en un sprint y sobrestimamos nuestras posibilidades, especialmente en el coste de familiarización con el entorno de aplicación y el producto concreto a evolucionar. Finalmente, también supuso un grave problema

la aparición de necesidades y tareas transversales que afectaban a múltiples historias de usuario a la vez y para las que se desconocía su gestión.

- Por otro lado, cambiar de hábitos es siempre una tarea complicada. Aspectos como adaptarse a las directrices marcadas por la nueva metodología, por ejemplo el realizar todos los días una breve reunión diaria, o habituarse a utilizar nuevas herramientas, consumieron un alto porcentaje de tiempo descontado de la dedicación para cumplir los objetivos en sí del sprint.

En contrapartida, la metodología también aportó aspectos muy positivos al desarrollo:

- El aumento de la interacción entre el grupo de trabajo y el contacto continuo con el cliente favoreció la comunicación y solución de problemas. El ideal de *"ninguno de nosotros es tan inteligente como todos nosotros"* se vio constatado. Así, se favoreció la toma de decisiones durante el desarrollo pues la responsabilidad se repartió entre todo el grupo. Aunque no se lograron cumplir los tiempos, las tareas avanzaron de forma más rápida y segura.
- Del mismo modo, se constató un aumento de la motivación en el trabajo por prácticamente todos los componentes del grupo.
- Por otro lado, la práctica de pair-programming resultó positiva pues los mayores avances se lograron en la parte del simulador donde estaba siendo aplicada.

No obstante, el resultado de está retrospectiva fue negativo, concretamente la puntuación alcanzada, siguiendo las normas establecidas en el diseño del experimento, fue -10. A continuación, aparece una tabla resumen del desarrollo de este sprint.

Tabla 6.2 Tabla resumen sprint 4.

Factor de Desarrollo	**Valor**
Duración	Número total de días hábiles: 13
Objetivo marcado	US17 Evolución de TOPEN que permita operar, monitorizar y probar el tanque de trituración de la planta de biogás
	US23 Adaptación del Simulador para la operación, monitorización y pruebas del tanque de trituración.
	US45 Analizar estado actual de TOPENprimer
Desarrollo	*Número de Clases 216 Líneas de código 30667*
Objetivos obtenidos	Aunque se han producido avances en conocimiento del dominio y de los productos y en diseño e implementación, no se ha logrado completar las historias de usuario principales planificadas. Los avances más significativos se han producido en la historia de usuario US45.
Puntuación retrospectiva	-10

6.4. Sprint 5

Con el objetivo de mejorar los resultados obtenidos en el sprint 4 dio comienzo el sprint 5. En la reunión de planificación se prestó especial atención en definir las historias de

usuario en un nivel mayor de detalle para realizar una planificación realista de acuerdo a los recursos disponibles. El hecho de que en el sprint anterior se profundizase ampliamente en el dominio de aplicación repercutió muy positivamente en este sentido. El objetivo propuesto para este sprint fue completar las historias de usuario US17 y US23, pendientes del sprint pasado. Dichas historias de usuario fueron descompuestas en un nivel mayor de detalle considerando los elementos de interés del tanque de trituración. De este modo, apareció un amplio conjunto de historias de usuario nuevas. Se constató, en ese momento, la imposibilidad de cumplir todas estas historias de usuario referentes al tanque de trituración en un solo sprint con los recursos disponibles y el cliente tuvo que restablecer sus prioridades. Finalmente, el Sprint Backlog quedó constituido por un subconjunto de historias de usuario del tanque de trituración (US19, US28, US30, US31, US41 y US42). Aunque reforzados por la experiencia adquirida en el sprint anterior, la reunión de planificación de este sprint se realizó de una manera más apropiada, no obstante, continuaron presentes algunos de los interrogantes que ya nos habían acechado en el sprint pasado, como la representación y gestión eficiente de aquellas necesidades que afectaban transversalmente a más de una historia de usuario, y para los cuales aún no encontrábamos solución. Parte del contenido de esta reunión de planificación, en la que también se trataron otros aspectos como la distribución del trabajo y la fecha de entrega del resultado de este sprint, puede encontrarse en el acta de la misma, incluida en la sección Anexos de esta tesis de máster[12].

El trabajo en este sprint avanzó a buen ritmo ajustándose de un modo más aproximado a la planificación planteada, hecho que acrecentó la motivación del equipo. El trabajo en diseño, implementación y pruebas se sucedía de forma ágil, en colaboración con el cliente, para resolver cualquier tipo de cuestión que apareciese en el desarrollo. El crecimiento del código y el número de clases impactadas en este sprint fue muy significativo en ambos productos. Sin embargo, surgieron algunas limitaciones técnicas en la parte de pruebas, especialmente en la aplicación TOPENbiogas. En este producto el lenguaje de pruebas depende de la gramática de comandos por lo que las pruebas no se pueden comenzar a diseñar hasta que esta gramática esté definida. En las pruebas funcionales de operación del simulador de la planta de biogás no existe esta limitación, pues se utiliza un lenguaje de pruebas específico que no depende de la gramática y, por tanto, las pruebas pueden diseñarse desde el comienzo, aunque el problema persiste en el conjunto de pruebas sobre la interacción del simulador con TOPENbiogas. Al mismo tiempo, se evidenció una fuerte dependencia temporal entre las pruebas de TOPENbiogas y del simulador, ya que en algunos casos los fallos encontrados en pruebas de TOPENbiogas pueden deberse, en realidad, a fallos en la aplicación del simulador. No obstante, cabe destacar que le hecho de que en el producto TOPENprimer se conociesen las partes variables en la evolución ha repercutido en que las pruebas de validación del mismo, al introducir cambios controlados, no hayan planteado excesivos problemas y hayan sido relativamente sencillas de realizar. De hecho, los fallos se han ido absorbiendo prácticamente en la parte de desarrollo una vez

[12] Tras las reuniones celebradas en el proyecto se redactaba un acta que resumía los principales aspectos acordados en la misma. Un aspecto importante a considerar es que este documento no se considera como parte de la documentación del proyecto, de hecho podría considerarse contraproducente al aplicar una metodología ágil como SCRUM y, en principio, no aportar un valor directo al proyecto. Sin embargo, su redacción estaba motivada para facilitar el posterior análisis del experimento en la realización del estudio.

que la gramática de comandos estaba definida. En realidad, los defectos encontrados se pueden considerar más bien modificaciones del cliente que errores técnicos.

Por otro lado, algunas desviaciones respecto a la metodología comenzaron a limarse. Es el caso, por ejemplo, del desarrollo de las reuniones diarias. Se detectó que no se estaba haciendo un correcto uso de las mismas porque se trataban en ellas aspectos que se apartaban de su propósito. La causa de la prolongación de las reuniones diarias se encontraba, principalmente, en el intento de solucionar los problemas que se presentaban en el desarrollo. Se optó por que los problemas que aparecían en el proyecto fuesen mencionados en las reuniones diarias pero si la solución a dicho problema, por su envergadura, requería un tiempo prolongado fuese en reuniones colaterales, con las personas implicadas en el problema y la posible solución, en las que se plantease dicha solución evitando, de este modo, que el resto del equipo dedicase un tiempo excesivo a algo que no era de su total competencia. Una vez encontrada la solución se exponía, en la siguiente reunión diaria, al resto del equipo para que todos tuviesen un conocimiento global del avance del proyecto.

En la reunión de revisión, a la que asistieron tanto el cliente como todos los miembros del equipo, se constataron los avances y el buen funcionamiento en el desarrollo del proyecto. Tal y como establecen las metodologías ágiles, la mejor prueba de este avance es mostrar código que funciona y en esta reunión se mostró operativo gran parte del trabajo previsto, constituyendo un 77% de las tareas planificadas (para más información consultar el acta de revisión de este sprint en la sección de Anexos). No obstante, por tratarse de tareas pertenecientes a distintas historias de usuario, solamente el 50% de éstas quedaron totalmente completadas.

La satisfacción tanto del cliente como del equipo aumentó notablemente durante este quinto sprint y así quedó reflejado en la reunión retrospectiva del mismo. En esta reunión la mayor parte de aspectos tratados fueron positivos, añadidos a los ya detectados en la retrospectiva del sprint 4, los más significativos fueron:

- Se valoró muy positivamente los aciertos en la fase de planificación del sprint, produciéndose un gran avance en la definición de las historias de usuario, principalmente por el incremento del conocimiento en el producto a evolucionar. No obstante, este avance fue cogido con cautela pues aún quedaban algunas lagunas pendientes que debían ser salvadas. Además, la celebración de reuniones colaterales fue acogida con gran entusiasmo por el equipo.
- En este sprint la motivación y comunicación entre el grupo de trabajo se vio igualmente muy reforzada.

No obstante, también aparecieron algunos aspectos negativos como el incumplimiento 100% de la planificación o el hecho de no congelar del todo el Sprint Backlog, tal y como establece la metodología, pues a lo largo del sprint se añadieron algunas necesidades o variaciones del cliente, de pequeña envergadura, que no estaban contempladas en el Sprint Backlog, pero que estaban íntimamente relacionadas en el trabajo del mismo. No obstante, el equipo obtuvo una conclusión positiva de este hecho pues comprobó como la metodología se adaptaba a los cambios imprevistos del cliente para evolucionar el producto.

Tras este análisis y partiendo de una puntuación negativa en el sprint anterior la puntuación obtenida fue de +25. La tabla 6.3 muestra un resumen del desarrollo en este sprint.

Tabla 6.3 Tabla resumen sprint 5.

Factor de Desarrollo	**Valor**	
Duración	Número total de días hábiles: 11	
Objetivo marcado	Adaptación de TOPEN que permita operar, monitorizar y probar el tanque de trituración, centrándose en los comandos básicos de encendido, apagado y control de la velocidad del mismo.	
	Simulador: operar tanque de trituración a través del simulador.	
Desarrollo	*Número de Clases 232*	*Líneas de código 32491*
% Objetivos obtenidos	TOPENbiogas 50% (70% de las tareas)[13]	Simulador (SUT) 50% (77% de las tareas)
Puntuación retrospectiva	+25	

6.5. Sprint 6

En este punto, el tanque de trituración continuaba sin estar terminado al 100%. Los retrasos en la implementación constituían las principales causas de esta situación. Sin embargo, las labores en diseño ya se daban por concluidas. Esta fue la principal razón que propició que la planificación de este sprint se descompusiese en tres líneas. Por un lado, sin la posibilidad de aumentar los recursos disponibles en implementación, completar el resto del tanque de trituración, tanto para la evolución de la herramienta TOPENbiogas como para el simulador, constituía el principal objetivo del sprint y, por tanto, acaparó la mayor parte de recursos disponibles. Se debía finalizar el trabajo pendiente del sprint pasado y agregar nueva funcionalidad, no considerada hasta el momento, relativa a los comandos que operan, monitorizan y prueban algunos parámetros del tanque de trituración. Concretamente, parte de los comandos *set* y *get*[14], especificados en las historias de usuario US30, US31, US32, US33, US34, US35, US36, US37, US38, US39, US52, US57, US58, US59, US60 y completar la historia de usuario US42 del simulador[15]. Por otro lado, se consideró adecuado que un subconjunto del equipo de trabajo, concretamente dos personas, comenzasen a trabajar en el diseño del siguiente tanque, el tanque de pasteurización, con el propósito de adelantar trabajo para la evolución del producto en el siguiente sprint (US63).

Además, en este sprint se reflejó uno de los problemas que, hoy por hoy, siguen acechando a las metodologías ágiles. Aunque la estrecha relación con el cliente puede considerarse muy satisfactoria, en este sprint, especialmente, se ha comprobado que, en muchas ocasiones, el cliente se centra en tratar aspectos sobre lo que necesita que haga

[13] El porcentaje de tareas completadas respecto a número de tareas totales, por lo que pueden pertenecer a distintas historias de usuario que no estén 100% finalizadas y, por tanto, no computen en el porcentaje de objetivos conseguidos.

[14] Comandos que se utilizan para asignar y recoger el valor de un atributo del sistema a operar, monitorizar o probar.

[15] Si desea más información sobre estas historias de usuario póngase en contacto con el grupo de investigación SYST en el que se enmarca esta investigación (https://syst.eui.upm.es)

el sistema pero no tiene la visión suficiente sobre otros aspectos críticos, principalmente encasillados como requisitos no funcionales en las metodologías convencionales, relativos por ejemplo al uso de recursos, mantenimiento, portabilidad, seguridad, o diseño. Un ejemplo, advertido en este sprint y que ni siquiera aparece en las historias de usuario pues es "transparente" al usuario, es el hecho de que un componente completo de TOPENbiogas, el Gateway, tuviese que ser completamente rediseñado e implementado para su adaptación al nuevo protocolo de comunicación de la planta de biogás, ya que hasta el momento, en el producto de partida TOPENprimer, sólo estaba preparado para comunicarse vía *sockets*. Por tanto, en la tercera línea de planificación de este sprint se consideró el esfuerzo para evolucionar e integrar el Gateway, de forma que permitiese esta comunicación (US56 y US62). El hecho de que una de las líneas de investigación del grupo estuviese enfocada a la construcción de un Gateway Genérico que permitiese la comunicación entre TOPEN y cualquier protocolo de comunicaciones benefició y se vio beneficiada en esta tarea [67].

El desarrollo del trabajo fue similar al del sprint pasado, considerando prácticamente las mismas características. Se constató un decremento en el número de líneas y de clases del producto en evolución TOPENbiogas debido, principalmente, al aumento de conocimiento en el sistema que propició que se comprobara que muchas de las clases evolucionadas en el sprint pasado fuesen erróneas, es decir, no se necesitaba evolucionar tanto código para conseguir la funcionalidad. Además, a medida que aumentaban las dimensiones del proyecto y se profundizaba en el conocimiento de las metodologías ágiles y en nuestras propias necesidades de trabajo, se comenzaron a detectar vacíos en las herramientas utilizadas en el desarrollo lo que provocó un sentimiento general de rechazo hacia la herramienta, pues más que potenciar, minaba la agilidad en el desarrollo. Por otro lado, comenzaron a aparecer algunos pequeños indicios de desánimo en el equipo manifestado, principalmente, con un descenso en el compromiso con algunos hábitos que parecían ya adquiridos como las reuniones diarias y, tal vez, influenciado por la falta de agilidad que se percibía en algunas tareas.

En la reunión de revisión del sprint se mostraron los avances del mismo. Se había completado gran parte del trabajo previsto, señal del perfeccionamiento en la planificación del sprint. En el simulador se habían alcanzado todos los objetivos, integrándolo, además, con el Gateway Genérico. En el producto TOPENprimer se había avanzado adecuadamente, aunque aún quedaban pendientes algunas historias de usuario vinculadas a comandos *set* que no estaban totalmente finalizadas y completar la integración con el Gateway Genérico. Además, se presentaron algunos retrasos en el caso particular de las pruebas causados por los obstáculos encontrados en el uso de la herramienta Rally.

En la reunión retrospectiva se constató que gracias al buen análisis que se realizó del punto de partida, la planificación fue más realista, no obstante, se concluyó que podríamos hilar más fino si no realizábamos planificaciones tan ajustadas. Por otro lado, el espíritu de grupo, la motivación y el compromiso del equipo habían decaído y se catalogaron, unánimemente, como puntos a mejorar. Entre los aspectos negativos más destacables podemos encontrar la funcionalidad ofrecida por las herramientas y la incorrecta distribución del trabajo. Cabe destacar, finalmente, que las reuniones en este sprint comenzaban a acortarse en duración, aunque no en contenido y resultados conseguidos en las mismas, muestra inequívoca de que se estaba consiguiendo una adecuada adaptación de la metodología. La puntuación en este sprint descendió levemente respecto al sprint anterior, obteniendo una valoración de 7 puntos. La tabla 6.4 muestra un resumen del desarrollo del sprint 6.

Tabla 6.4 Tabla resumen sprint 6

Factor de Desarrollo	**Valor**	
Duración	Número total de días hábiles: 10	
Objetivo marcado	Continuar la evolución de TOPEN que permita operar, monitorizar y probar el tanque de trituración.	
	Finalizar la adaptación del Simulador para la operación, monitorización y pruebas del tanque de trituración	
	Análisis del tanque de pasteurización	
	Integración Simulador y TOPEN con Gateway Genérico	
Desarrollo	*Número de Clases 209* *Líneas de código 28960*	
% Objetivos obtenidos	TOPENbiogas 56,25%	Simulador (SUT): 100%
Puntuación retrospectiva	7	

6.6. Sprint 7

Considerando el desbordamiento de trabajo que sufrían algunos miembros del equipo y la imposibilidad de aumentar el compromiso de otros por la ineludible responsabilidad en diferentes tipos de tareas docentes, universitarias y de investigación, en este sprint se optó por incorporar dos nuevos componentes al equipo. Uno de ellos apoyaría en labores de recogida de métricas para el posterior análisis del experimento, con el objetivo de mitigar el esfuerzo que suponía para los integrantes del equipo llevar a cabo esta tarea. El otro componente ayudaría en tareas de gestión pues hasta el momento estas tareas eran responsabilidad del Scrum Master que venía desbordado su trabajo.

En la reunión de planificación se establecieron objetivos primarios, claramente alcanzables, y objetivos secundarios, deseables pero no obligatorios. Como objetivo primario en TOPENbiogas se comprendió el completar los comandos *set* del tanque de trituración que quedaban pendientes del sprint pasado, la integración con el componente Gateway Genérico y el tratamiento de las notificaciones que producía el tanque de trituración, aspecto no identificado hasta el momento y que fue añadido a través de la historia de usuario US43. Como objetivo secundario para este sprint respecto al producto TOPENbiogas, se consideró el comenzar la implementación del siguiente tanque, tanque de pasteurización, del cual ya existía un diseño bastante aproximado. Descomponiendo en un nivel mayor de detalle la historia de usuario US17, ya identificada en el Product Backlog y que describía la interacción que TOPENbiogas debía tener con el tanque de pasteurización, aparecieron las historias de usuarios US63, US69-US90. De este conjunto de historias de usuario, como objetivo secundario en el Sprint Backlog, solo se contemplarían las historias que hacían referencia al encendido y apagado del tanque, US72 y US73 respectivamente. Respecto al simulador los objetivos, todos primarios, fueron emitir las notificaciones que en la planta real emite el

tanque de trituración (completar US42 con las notificaciones) y evolucionarlo para que contemplase asimismo el tanque de pasteurización (US25).

En este sprint, debido a los problemas que estaban causando la gestión de aquellas necesidades transversales al sistema, se consideró la posibilidad de tratarlas por separado bajo el concepto que comenzaba a vislumbrase de *historia de sistema* [Artículo JISBD]. Un ejemplo que ilustra claramente esta situación y que llevaba presente durante todo el proyecto es el que aparece en la tabla 6.5.

Tabla 6.5. Requisitos transversales

REQUISITO FORMAL		**HISTORIA DE SISTEMA (System Story)**			
Id Req	Definición	Id SS	Usuario	Definición	Motivación
R16	El entorno de operación y monitorización TOPEN podrá ser accedido tanto en forma local como en forma remota.	SS6	Ingeniero de monitorización y pruebas	Acceder al entorno de operación y monitorización de forma local o remota	Operar y monitorizar la planta de biogás
R32	La presión interna de un tanque no podrá superar la presión límite de resistencia del mismo	SS20	Ingeniero de monitorización y pruebas	Mantener la presión interna de un tanque	Operar y monitorizar la planta de biogás

Por ejemplo, que el sistema pueda ser accedido tanto de forma local como de forma remota repercute en que todos los comandos que TOPENbiogas pueda tratar deberán tener en cuenta esta consideración y, a su vez, las tareas de planificación, estimación del esfuerzo y validación de las historias de usuario que se refieran a comandos también se verán afectadas. Evidentemente, dado que el concepto es nuevo, la herramienta Rally, utilizada en la gestión del proyecto, no daba soporte a estas historias de sistema y fueron consideradas en tarjetas separadas.

El desarrollo del sprint varió en gran medida de los tres anteriores. El acaecimiento de una serie de compromisos ineludibles externos al experimento repercutió en la imposibilidad de mantener el compromiso de algunos componentes. Además, no se tomaron medidas contundentes para atajar los principios de desmotivación presentes en parte del grupo. Estos hechos provocaron que no todos los componentes cumpliesen los compromisos adquiridos en la etapa de planificación. Básicamente, el equipo quedó reducido a un grupo de cuatro componentes, centrados principalmente en labores de implementación, más los dos nuevos miembros incorporados. No obstante, y dado que las labores de diseño estaban bastante avanzadas y se tenía un mayor conocimiento del sistema a evolucionar, la implementación avanzó de forma ágil. De forma simultánea se daba solución a problemas detectados en el resultado del sprint anterior. Sin embargo, hubo tareas que no se realizaron adecuadamente, principalmente en la parte de pruebas.

A pesar de esta situación, los objetivos primarios alcanzados para este sprint fueron excelentes. Al finalizar el sprint, el tanque de trituración estaba totalmente finalizado en ambos productos. Además, en el simulador el tanque de pasteurización estaba completo en un 70% de las tareas que tenía planificadas y, al formar éstas parte de historias de usuario diferentes, en un 50% de su funcionalidad.

Sin aplicar SCRUM, este sprint hubiese fracasado ya que hubo una porción importante de recursos que tuvieron el, denominado dentro del experimento, *efecto vaivén* (efecto inherente al propio contexto en el que se desarrolla el experimento debido a las obligaciones de parte del equipo docentes e investigadoras en otros

proyectos). El contradictorio valor de objetivos alcanzados respecto a recursos disponibles puede explicarse por la ágil respuesta del grupo, incrementando los esfuerzos de los componentes comprometidos para suplir los de aquellos que colaboraban de una forma menos activa. Sin esta ágil respuesta en la distribución de tareas el resultado de este sprint hubiese sido catastrófico.

En la reunión retrospectiva se identificados los problemas acontecidos. Todos eran de calado filosófico y de seguimiento de la metodología más que técnico. El cumplimiento de objetivos y la sensación de progreso fueron los principales, y casi únicos, aspectos positivos detectados. La ausencia de un espíritu de grupo, de motivación y de compromiso personal, la actitud pasiva en el desarrollo, la ausencia en las reuniones establecidas, con los consiguientes problemas de comunicación, y la falta de recogida de mediciones fueron los principales lastres arrastrados en el sprint. De este modo, la puntuación obtenida, partiendo de 7 puntos, se precipitó a -61. Los objetivos se cumplían pero a costa del sobreesfuerzo de un subconjunto del equipo.

Tabla 6.6 Tabla resumen sprint 7

Factor de Desarrollo	**Valor**	
Duración	Número total de días hábiles: 10	
Objetivo marcado	Finalización de TOPEN que permita operar, monitorizar y probar el tanque de trituración.	
	Finalización del Simulador para la operación, monitorización y pruebas del tanque de trituración, completando la parte de notificaciones. Evolución del Simulador para la contemplar el tanque de pasteurización.	
Desarrollo	*Número de Clases 209 Líneas de código 29065*	
Objetivos obtenidos	TOPENbiogas 100%	Simulador (SUT) 50% (70% de las tareas)
Puntuación retrospectiva	-61	

6.7. Sprint 8

Paradójicamente, el hecho de que en el sprint anterior se hubiese tocado fondo respecto a los principios y valores que guían SCRUM resultó muy positivo para este octavo sprint. Los componentes del equipo estaban concienciados del sobreesfuerzo que supuso para algunos miembros lograr cumplir objetivos en el sprint anterior, y dispuestos a mejorar y cambiar aquellos aspectos que les habían llevado a esa situación.

Los objetivos marcados para este sprint en el producto TOPENbiogas, de acuerdo a las prioridades del cliente, consistieron en evolucionar el producto para que permitiese la operación, monitorización y pruebas sobre el tanque de pasteurización, segunda etapa del proceso de obtención de biogás. Debido al amplio conocimiento que proporcionaba el informe obtenido en sprints anteriores, que especificaba los cambios efectuados para adaptar el producto TOPENprimer al producto TOPENbiogas para el tanque de

trituración, y dado que las tareas en el diseño de este tanque estaban prácticamente completadas, se concluyó viable el completar todos los comandos del tanque de pasteurización para este sprint, dejando como objetivo secundario la implementación de la captura y tratamiento de las notificaciones. En el producto simulador, en el que se seguía utilizando la técnica de pair-programming, se avanzaba más rápido. De hecho el tanque de pasteurización ya estaba prácticamente implementado y se propuso evolucionarlo para que contemplase también el siguiente tanque, el tanque de hidrólisis. El desarrollo del tanque de hidrólisis debía ser completo, incluyendo diseño del mismo. Se valoró este factor por lo que se estableció como objetivo principal todas las necesidades relacionadas con comandos y se dejaron como objetivos secundarios las necesidades referentes a notificaciones.

El desarrollo, con un talante por parte del grupo similar al del sprint 5, transcurrió con normalidad. Los aciertos en las planificaciones comenzaban a dar sus frutos. Una vez completado el tanque de pasteurización en el simulador, se llevó a cabo una refactorización del mismo para mejorar, limpiar y hacer más legible el código. Se produjo una reducción de un 5.4% de líneas de código tras la refactorización. Después de la refactorización la funcionalidad del sistema se mantuvo inalterada gracias al repositorio de pruebas automatizadas del que se disponía. Un aspecto importante a considerar es que la inclusión del nuevo tanque para cuya implementación se habían estimado 25 horas, se realizó en 3 horas, con tan solo 152 líneas de código nuevas. Los recursos que quedaron disponibles en el simulador fueron rápidamente redistribuidos. En TOPENbiogas las tareas también se realizaban de acuerdo a lo planificado. El hecho de disponer de un informe en el que se indicaba claramente el código a manipular para evolucionar el producto fue de gran utilidad en este sentido.

Los resultados obtenidos en el sprint fueron muy buenos. La planificación propuesta para el simulador fue cumplida sobradamente, realizando además una etapa de refactorización que no había sido considerada en un principio. En TOPENbiogas el objetivo primario era implementar todos los comandos de interacción con los elementos del tanque de pasteurización. Aunque el trabajo resultó mucho más sencillo, no se logró cumplir la planificación en su totalidad y quedó un pequeño porcentaje de este trabajo pendiente para el siguiente sprint, además del objetivo secundario asociado a las notificaciones del tanque. Este retraso se debió a la redistribución de recursos que se tuvo que realizar por un acontecimiento no considerado en la planificación, pero sumamente importante para el experimento: la divulgación de resultados experimentales. Un porcentaje del esfuerzo de este sprint se dedicó a elaborar una presentación sobre la experiencia al representar las necesidades del cliente mediante historias de usuario. Concretamente, la presentación, titulada "*Requisitos convencionales frente a Historias de Usuario en Metodologías ágiles*", fue expuesta el 20 de junio en las Jornadas de Experiencias y Soluciones Solo Requisitos 2008, en las que participaron empresas como Borland o Gesein y universidades como la Universidad Politécnica de Cataluña (UPC) o la de Sevilla.

En la reunión retrospectiva de este sprint la mayor parte de los aspectos tratados volvieron a ser positivos. El hecho de que se lograse superar el fracaso del sprint anterior fue acogido con gran entusiasmo, aunque con cautela para no volver a repetir los motivos que nos habían llevado a aquella situación de fracaso. Se valoró muy positivamente la consecución de los productos y la sensación de progreso que volvía a invadir a los miembros del equipo. El hecho de que los objetivos marcados para los productos estuviesen prácticamente cubiertos aumento notablemente la satisfacción del cliente. Las prácticas ágiles como el pair-programming y la refactorización también

habían resultado muy positivas. Finalmente, la explotación de resultados, desde la perspectiva del experimento completó el conjunto de aspectos positivos del sprint. Como aspectos a mejorar se podría destacar de nuevo la motivación, el compromiso de los componentes del equipo y la interacción entre el grupo que, aunque en este sprint se vio claramente reforzada aún eran susceptibles de mejora.

La desastrosa puntuación con la que partíamos en este sprint, -61 puntos, hacía difícil la remontada, no obstante se mejoró el resultado obteniendo -26 puntos. La tabla 6.7 refleja los acontecimientos que caracterizaron el transcurso del octavo sprint.

Tabla 6.7 Tabla resumen sprint 8

Factor de Desarrollo	**Valor**	
Duración	Número total de días hábiles: 13	
Objetivo marcado	Adaptación de TOPEN que permita operar, monitorizar y probar el tanque de pasteurización (exceptuando las notificaciones)	
	Finalización del tanque de pasteurización del simulador (notificaciones). Evolución del simulador que contemple el tanque de hidrólisis	
Desarrollo	*Número de Clases 213 Líneas de código 30111*	
Objetivos obtenidos	TOPENbiogas 81%	Simulador (SUT) 100%
Puntuación retrospectiva	-26	

6.8. Sprint 9

El sprint 9 constituyó el último sprint del experimento dando. Recapitulando, según los objetivos marcados al comenzar el proyecto, el trabajo pendiente era:

- En el producto TOPENbiogas, finalización de la evolución del producto para contemplar el tanque de pasteurización, concretamente un pequeño porcentaje de comandos *set* y tratamiento de las notificaciones, y reiteradas evoluciones del producto para contemplar los tanques de hidrólisis y digestión.
- En el producto simulador, evolucionar el simulador para la integración total con el tanque de digestión.

En la reunión de planificación, analizando el coste que suponía la realización de todo este trabajo y basándonos en la experiencia previa, se concluyó la imposibilidad de completar todo el producto TOPENbiogas considerando los recursos disponibles. De este modo, se planificó completar la funcionalidad relativa al tanque de pasteurización e implementar el tanque de hidrólisis. Puesto que el modelo del tanque de hidrólisis ya estaba diseñado en el sprint anterior para el simulador, todos los recursos podrían ser destinados a tareas de implementación. Al igual que en sprints anteriores, la historia de usuario US21, referente a la interacción con el tanque de hidrólisis, fue descompuesta en un nivel mayor de detalle obteniendo las historias de usuario US93-US104. Por

tanto, la historia de usuario US22, relativa a la evolución de TOPEN para contemplar el tanque de digestión, quedaba pendiente. Por otro lado, teniendo en cuenta el esfuerzo requerido para evolucionar el simulador en el sprint anterior, tras la refactorización, completar la integración total con el tanque de digestión se veía más que viable.

El equipo tenía pleno conocimiento tanto del dominio de aplicación como de los productos que estaban siendo evolucionados por lo que el trabajo en este sprint fue sumamente ágil. A mitad del sprint las labores en el simulador ya se daban por concluidas y el resto del sprint, respecto a este producto, se destinó a dos puntos: por un lado, incrementar el número de pruebas del repositorio de pruebas para el simulador y, por otro lado, perfeccionar algunos aspectos del mismo relacionados, principalmente, con el interfaz gráfico de usuario. En el producto TOPENprimer el trabajo fue más prolongado y consumió todo el sprint. Los trabajos respecto al tanque de pasteurización consumieron más de la mitad del sprint y, pesimistamente, se especuló con la imposibilidad de terminar todas las tareas pendientes a tiempo. Sin embargo, la experiencia adquirida hasta el momento, el hecho de que el producto siguiese algo parecido a una línea de producto y que, una vez analizado, se comprobase que el tanque de hidrólisis constituía, en realidad, una mezcla de los dos tanque implementados anteriormente (tanque de trituración y tanque de pasteurización), favoreció enormemente los trabajos de evolución y, finalmente, el trabajo fue completado en su totalidad.

En la reunión de revisión se mostró al cliente el producto final construido tras el proyecto. La satisfacción de éste fue elevada puesto que, exceptuando el tanque de digestión para el componente TOPENbiogas, todos los objetivos marcados se habían cumplido, obteniendo un producto de alta calidad.

En la reunión retrospectiva de este sprint, además de analizar aspectos sobre la situación actual del grupo de trabajo, cada miembro del equipo explicó sus impresiones tras la realización del experimento de evolución software. La opinión general fue muy favorable a la metodología. Los aspectos más destacados fueron:

- El espíritu de grupo creado al utilizar SCRUM resultó muy positivo.
- En todas las ocasiones el equipo logró adaptarse a las variabilidades de las necesidades del cliente debido a la división del trabajo en ciclos de corta duración y a la estrecha interacción con el mismo.
- El rendimiento del equipo de trabajo en la evolución de TOPENprimer fue muy destacado. Junto con la ágil redistribución del trabajo en situaciones críticas, permitió que los resultados obtenidos desde la perspectiva industrial fuesen muy satisfactorios tanto para el cliente como para el propio equipo.

Entre los aspectos negativos destacaron cuatro:

- Por un lado, la utilización de herramientas en la gestión del proyecto.
- Para algunos miembros del equipo la continua asistencia a distintos tipos de reuniones se hacía poco efectiva y tenían cierta sensación de pérdida de tiempo.
- Por otro lado, la dificultad de disponer del cliente *in situ* durante el 100% del proceso de desarrollo del proyecto. No obstante, se considera que el hecho de que la interacción con el cliente fuese distribuida y existiese un miembro del equipo con el conocimiento necesario para tomar las principales decisiones no afectó negativamente en el desarrollo del experimento aplicando SCRUM.
- Finalmente, se analizó que la ausencia de documentación detallada perjudicaría el posterior mantenimiento del producto.

La puntuación obtenida en este sprint fue 9 puntos. La tabla 6.8 muestra, a modo de resumen, los principales sucesos del sprint 9.

Tabla 6.8 Tabla resumen sprint 9

Factor de Desarrollo	**Valor**	
Duración	Número total de días hábiles: 10	
Objetivo marcado	Completar la adaptación de TOPEN que permita operar, monitorizar y probar el tanque de pasteurización. Evolucionar TOPENbiogas para que contemple el tanque de hidrólisis	
	Completar el simulador con el tanque de digestión	
Desarrollo	*Número de Clases 213 Líneas de código 30148*	
Objetivos obtenidos	TOPENbiogas 94%	Simulador (SUT) 100%
Puntuación retrospectiva	9	

La tabla 6.9 muestra a modo de resumen las características del producto final obtenido tras el desarrollo TOPENbiogas

Tabla 6.9 Características del producto final TOPENbiogas.

Factor Contextual	**Característica**	**PRODUTO DE PARTIDA TOPENprimer**	
Estructura	Arquitectura	Producto formado por 4 componentes distribuidos: MIB, TopenEngine, Gateway (genérico), TOE	
Tamaño	Número de líneas de código del sistema	30148	
	Número de clases del sistema	213	
	Número de líneas de código por componente	MIB	7789
		TopenEngine	7893
		Gateway	857
		TOE	13609
	Número de clases por componente	MIB	48
		TopenEngine	52
		Gateway	10
		TOE	103
Características Técnicas	Lenguaje de programación	Java	
	Comunicaciones	RMI (Java Remote Method Invocation) Servicios	

Capítulo 7

Análisis de Resultados

En este capítulo se realiza el análisis del experimento que da base al estudio. El objetivo es dar respuesta a las cuestiones de competencia que guían la investigación y, finalmente, valorar la evolución del producto aplicando una metodología ágil como SCRUM.

7.1. Características del Producto durante la Evolución

En esta sección se analiza la evolución que los productos, TOPENbiogas y Simulador, han sufrido durante el desarrollo hasta constituir el producto final. Se ha tratado de un proceso de integración continua a pequeña escala, a través del cual, y mediante historias de usuario, se han ido sucediendo incorporaciones de funcionalidad en el entorno de producción. La figura 7.1 muestra gráficamente la evolución del Sprint Backlog que guiaba el desarrollo, agrupando las historias de usuario tanto para el entorno de operación, monitorización y pruebas como para el simulador[16].

[16] Dado que entre el sprint1 y el sprint3 la metodología utilizada fue SCRUM con algunas desviaciones respecto a la metodología entendida ortodoxamente, con el objetivo de adquirir cierta experiencia práctica en SCRUM, la satisfacción será analizada a partir del sprint 4 pues se considera que a partir de este momento la metodología ya puede ser generalizada.

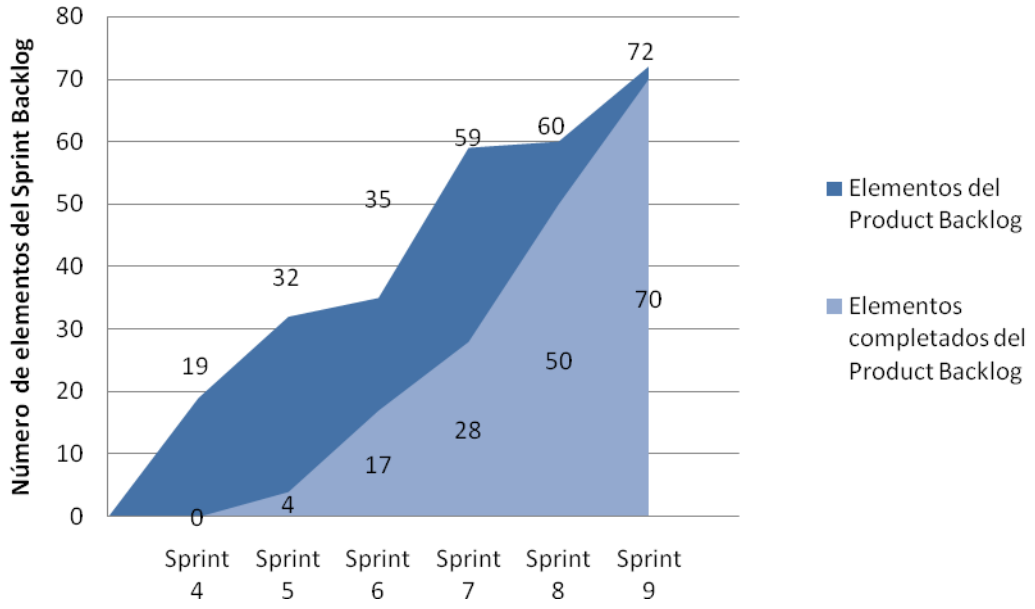


Figura 7.1 Evolución del Product Backlog a lo largo del desarrollo

El área oscura refleja el incremento de funcionalidad que sufre el producto en su evolución, entendiendo por producto el conjunto de TOPENbiogas y simulador, en los sucesivos sprints. Durante el desarrollo se pasa de un Product Backlog constituido por apenas 19 historias de usuario en el sprint 4 a alcanzar una dimensión de 72 historias de usuario en el último sprint del proyecto. El hecho de que a medida que avanzaba el proyecto el nivel de detalle con el que se veían los componentes fuese mayor influyó decisivamente en este sentido pues las 19 historias de usuario iniciales, referentes a cada uno de los tanques, fueron descompuestas a lo largo del desarrollo en función de los atributos que componían dichos tanques. De esta forma, y en continua interacción con el cliente, se fue refinando el producto.

El área azul clara muestra la integración de funcionalidad que ha ido sufriendo el producto a lo largo del desarrollo, conforme se daban por concluidas las historias de usuario. Puede deducirse que a medida que avanzaban los sprints, y con ellos el conocimiento tanto en el dominio de aplicación como en el producto a evolucionar, aumentaba, casi exponencialmente, la funcionalidad completada. En la figura 7.2, que muestra el porcentaje de necesidades del cliente cubiertas en cada sprint respecto al conjunto total de necesidades, puede observarse con mayor detalle esta circunstancia.

Aunque el producto no fue desarrollado completamente, dado que en TOPENbiogas quedó pendiente el tanque de digestión, el porcentaje final de realización del Product Backlog es de un 97%. La explicación a este valor la encontramos en que la historia de usuario correspondiente al tanque de digestión aún no ha sido descompuesta en un nivel mayor de detalle y estas historias de usuario descendientes no forman parte del Product Backlog final.

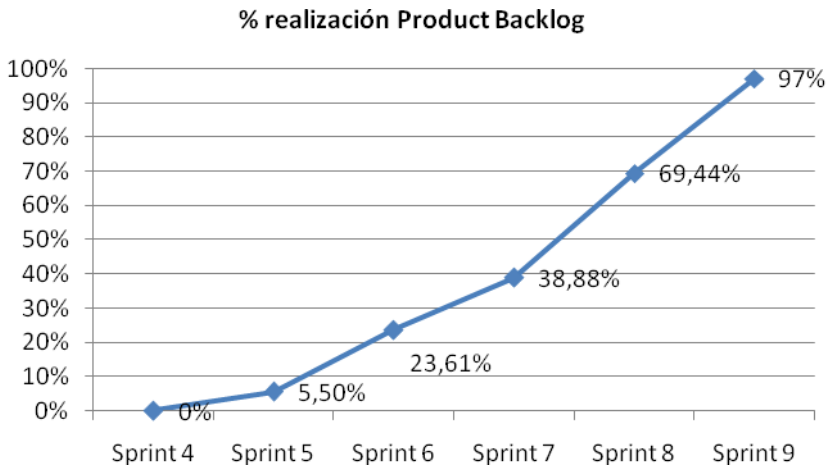


Figura 7.2 Evolución del porcentaje de realización del Product Backlog

Cabe destacar que el proceso de integración a gran escala que se realiza en las metodologías convencionales puede resultar incluso más complejo que la propia codificación. En el caso concreto del experimento, al tratarse de integraciones mucho más pequeñas (varias veces, o como mínimo una, al día), esta práctica no ha supuesto grandes problemas ya que la evolución del producto se ha realizado progresivamente, considerando la continua interacción con el cliente muy positiva en este sentido. La clave para que la integración continua haya sido viable fue disponer de una batería de pruebas, de tal forma que una vez que el nuevo código estaba integrado con el resto, se ejecutaba toda la batería de pruebas para validar la integración.

La evolución en líneas de código, tanto para el producto TOPENbiogas como para el simulador, puede verse en las figuras 7.3 y 7.4, respectivamente. Del mismo modo las figuras 7.5 y 7.6 muestran la evolución en clases impactadas durante el desarrollo para ambos productos. Cabe destacar la notable diferencia existente entre los dos productos. La evolución, en número de líneas de código del producto TOPENbiogas, apenas es palpable pues al tratarse de un producto de evolución las líneas nuevas se ven compensadas con las líneas eliminadas. No obstante, el producto final TOPENbiogas es de un tamaño sensiblemente menor que el producto de partida TOPENprimer. Sin embargo, el impacto en términos de clases es elevado. Puede observarse como las clases impactadas aumentan al principio del desarrollo. La principal explicación que se deduce de esta circunstancia es el hecho de que al comienzo del desarrollo, la falta de conocimiento en el producto a evolucionar, influye en que se manipulen más clases de las que, en realidad, es necesario manipular pues no se tiene el conocimiento suficiente para hacer esta tarea. A medida que este conocimiento aumenta se conocen las clases implicadas en la evolución del producto por lo que el impacto es menor. En el simulador, al tratarse de un producto nuevo, el incremento de líneas de código es evidente a lo largo del desarrollo, al igual que las clases impactadas.

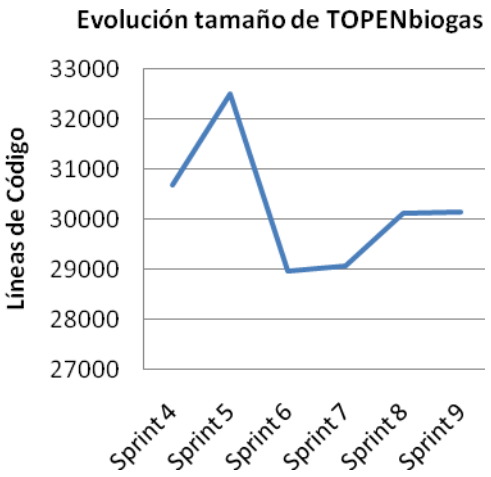


Figura 7.3 Evolución del código del producto TOPENbiogas

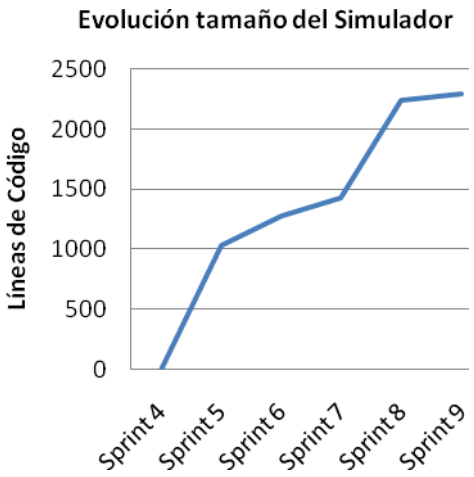


Figura 7.4 Evolución del código del Simulador

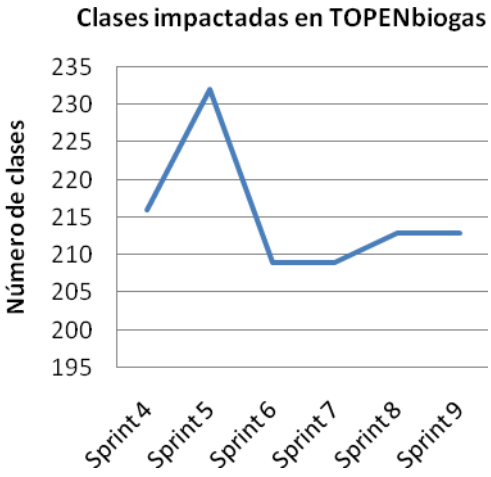


Figura 7.5 Evolución del impacto en las clases de TOPENbiogas

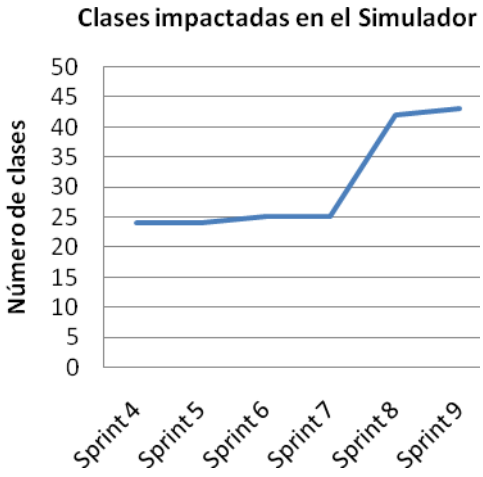


Figura 7.6 Evolución del impacto en las clases del simulador

Otro aspecto destacable en este sentido es el esfuerzo dedicado al proyecto para evolucionar el producto a lo largo de los sprints. En la figura 7.10, que aparece en la sección siguiente, puede observarse como el esfuerzo dedicado al proyecto, unión del esfuerzo dedicado a la evolución de TOPENprimer para obtener TOPENbiogas y el esfuerzo dedicado a la construcción del simulador, disminuye levemente a medida que avanzan los sprints, aunque no en detrimento de objetivos conseguidos. Este hecho se puede explicar porque se ha reducido el riesgo en el espacio temporal, es decir, los sprints anteriores nos han marcado el camino para la evolución del producto por lo que el conocimiento adquirido hace que seamos capaces de cumplir más objetivos dedicado un esfuerzo menor.

Podemos concluir, que en el desarrollo de este experimento la experiencia adquirida por el equipo durante el desarrollo, tanto en la metodología como en el dominio de aplicación del producto, ha sido crucial en el volumen de funcionalidad integrada. De este modo, la funcionalidad integrada desde el séptimo sprint hasta el final de proyecto es muy superior a la integrada en ciclos anteriores cuando aún no se

tenía un conocimiento profundo en el producto a evolucionar y no se estaba habituado a las directrices establecidas por SCRUM.

7.2. Aspectos de Agilidad en el Desarrollo

La productividad del desarrollo ha captado la atención de gran parte de los artículos empíricos que, hasta el momento, se han realizado sobre las metodologías de desarrollo ágil. En línea con estos estudios, en esta sección, se analizará la evolución de la productividad en los distintos sprints que se han desarrollado hasta conseguir el producto final, con el objetivo de dar respuesta a una de las principales cuestiones planteadas en esta tesis de máster, referente a cómo evoluciona la agilidad en el desarrollo al utilizar una metodología ágil como SCRUM.

Para el estudio de la productividad, ésta se ha medido en términos de objetivos alcanzados del Product Backlog final, en cada sprint, respecto a esfuerzo invertido en el mismo. En primer lugar, la figura 7.7 visualiza gráficamente el esfuerzo destinado al proyecto en cada sprint, medido en horas.

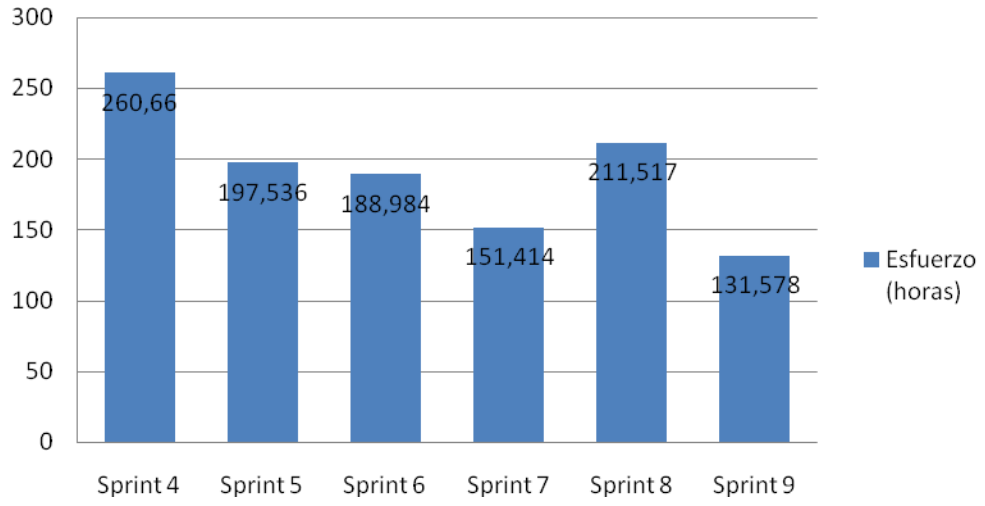


Figura 7.7 Esfuerzo dedicado al proyecto en cada sprint

El descenso de motivación a mediados del proyecto junto con la aparición de un conjunto de acontecimientos propios del contexto universitario en el que se desarrolla el experimento pero ajenos al mismo, propició la imposibilidad para algunos miembros del equipo de cumplir su compromiso y, consecuentemente, el esfuerzo dedicado en esta parte del desarrollo es sensiblemente menor. Por otro lado, la agilidad alcanzada en el último sprint explica que se cumpliesen los objetivos del mismo con una destinación de recursos menor.

Una vez conocidos los valores de esfuerzo dedicado al proyecto, y considerando el porcentaje de realización del Product Backlog en cada sprint que aparece en la figura 7.2, se ha calculado, en términos absolutos, el valor de la productividad en cada sprint del modo que se indica a continuación:

$$\text{Productividad } (S_x) = \left(\frac{\%\ Objetivos\ alcanzados\ del\ Product\ Backlog\ (sprint\ x)}{Esfuerzo\ dedicado\ (sprint\ X)}\right) * 100$$

La figura 7.8 muestra los resultados obtenidos.

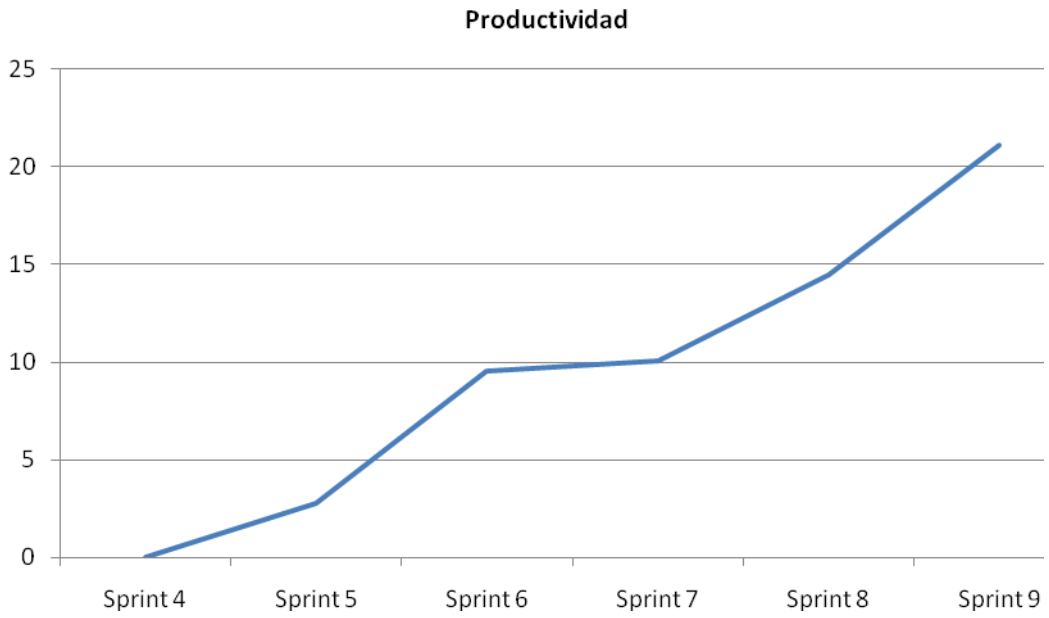


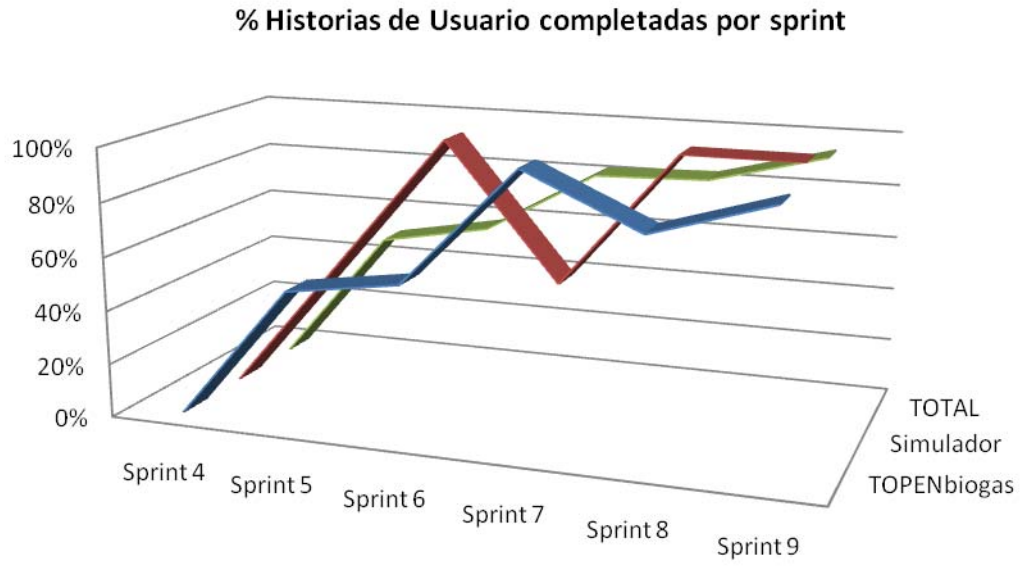
Figura 7.8 Evolución de la productividad a lo largo del proyecto

Los valores de la productividad están íntimamente ligados con la evolución del producto descrita en el apartado anterior. Como muestra la gráfica a medida que transcurren los sprints aumenta la productividad en el proceso, es decir, a medida que transcurren los sprints se cumplen más objetivos y se es más ágil. Existen dos factores principales que explican esta situación:

- Por un lado, la adaptación a la metodología en los primeros sprints repercute negativamente en el tiempo dedicado a cumplir los objetivos en sí del sprint, puesto que gran parte del esfuerzo se destina a conocer la metodología, habituarse a ella y aprender a utilizar las herramientas consideradas en este entorno. Además, se debe considerar que este proceso de adaptación repercute negativamente en que las tareas establecidas por la metodología se realicen de forma incorrecta minando la productividad del desarrollo. Así, los errores por ejemplo en la etapa de planificación influyen decisivamente en que el porcentaje de objetivos cubiertos en esta parte sea muy pequeño.
- Por otro lado, la ausencia de un conocimiento profundo del dominio de aplicación ha influido drásticamente en que las historias de usuario aparezcan a un nivel muy alto de detalle y no se puedan completar. Además, la falta de conocimiento en el producto a evolucionar hace que en los primeros sprints sea difícil evolucionar notablemente el producto porque los desarrolladores deben destinar un alto porcentaje del tiempo a conocer, en cierta medida, la arquitectura y el código del producto antes de proceder a su evolución.

La figura 7.9 muestra gráficamente la evolución en el porcentaje de historias de usuario completadas por sprint, considerando en este caso el Sprint Backlog. Se puede observar que a medida que avanza el proyecto las planificaciones de los sprints son más certeras, gracias al buen análisis que se realiza del punto de partida, aumentando

claramente el porcentaje de cubrimiento del Sprint Backlog. Por tanto, partir de una situación real y planificar de forma no optimista y reflexionando todos los miembros del equipo ha sido fundamental para realizar buenas planificaciones.



	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8	Sprint 9
TOPENbiogas	0%	50%	56%	100%	81%	94%
Simulador	0%	50%	100%	50%	100%	100%
TOTAL	0%	50%	57,80%	81,80%	83%	94%

Figura 7.9 Evolución de las historias de usuario completadas por sprint

En líneas generales el cumplimiento de objetivos en los sprints ha sido más elevado en el producto Simulador, producto iniciado en este proyecto, que en el producto TOPENbiogas, producto procedente de la evolución de TOPENprimer. Evidentemente, la evolución de un producto software incorpora un conjunto de dificultades no consideradas en productos de nueva construcción. Además, durante el desarrollo de este producto se ha utilizado la técnica de Pair-Programming con resultados, evidentemente, muy positivos tanto en cumplimiento de objetivo como en satisfacción de los miembros implicados. Por otro lado, en el sprint 8 ser realizó una fase de refactorización en dicho producto con un tiempo empleado aproximado de 5 horas. El resultado fue una reducción de un 5.4% de líneas de código tras la refactorización. A priori no implica un gran porcentaje del código escrito. Las mayores ventajas se derivan de haber eliminado código (funcionalidad) duplicada en distintos puntos del diseño, lo que implicará una mayor facilidad en el mantenimiento. Otro aspecto importante a considerar es que la inclusión de un nuevo tanque en el sprint 9, para el que se habían estimado 25 horas, ha sido realizado en 3 horas, con tan solo 152 líneas de código nuevas.

Por otro lado, es interesante analizar el esfuerzo que se ha destinado a las distintas tareas para lograr estos valores de productividad y explicar porque ha evolucionado de este modo. La figura 7.10 muestra la distribución del esfuerzo destinado al proyecto

para cada una de las tareas consideradas (EC: Configuración de entorno; RM: Gestión de versiones; AT: Prácticas relacionadas con la metodología; US: Definición y gestión de historias de usuario; D:diseño; CIC: Codificación; CIP: diseño y ejecución de pruebas; TS: soporte técnico; DOC: documentación; y O:otras tareas relacionadas con el experimento).

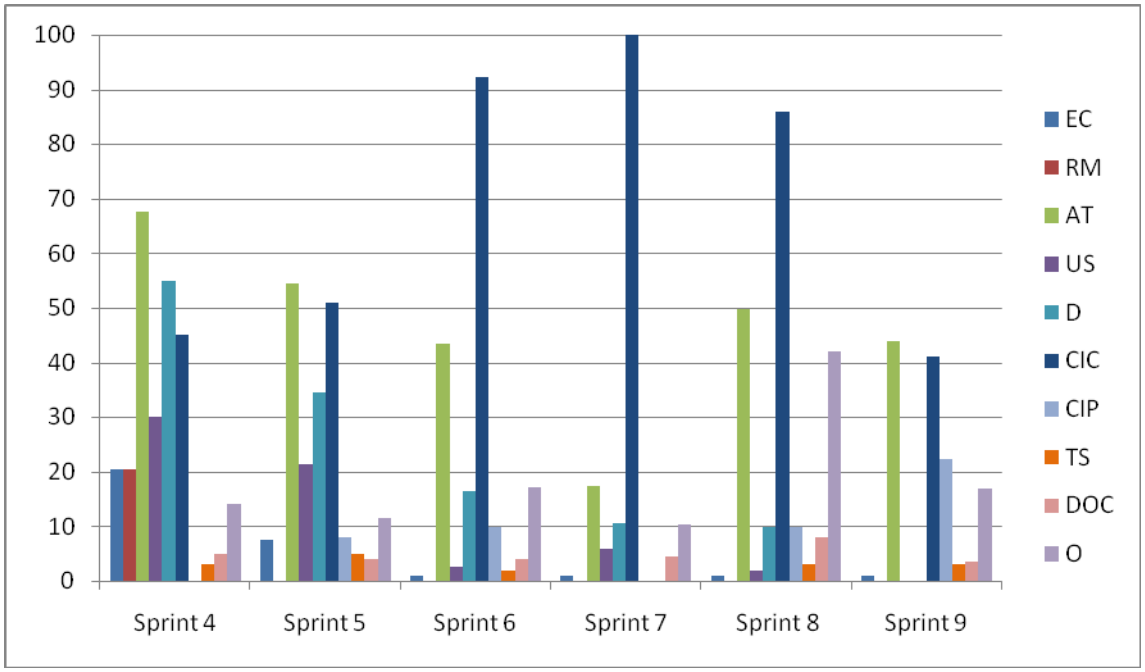


Figura 7.10 Distribución de esfuerzos a lo largo del desarrollo

Como puede observarse, los esfuerzos en los primeros sprints se encontraban muy distribuidos entre las distintas tareas. El tiempo dedicado a tareas ágiles relacionadas con la metodología, a la definición y gestión de las historias de usuario y al diseño es muy elevado, en detrimento del esfuerzo dedicado a tareas de integración continua (codificación y pruebas) lo que redunda altamente en el incumplimiento de objetivos en sí de cada sprint. A medida que se fue adquiriendo conocimiento en la metodología de desarrollo y en el dominio de aplicación los esfuerzos en codificación pasaron a copar el más alto porcentaje, y con el ellos el cumplimiento de objetivos. Una reflexión que cabe hacerse en este sentido es si el hecho de realizar una etapa tan ligera en la definición de las necesidades del sistema influya en que en los primeros sprints no se tenga el conocimiento suficiente para cumplir los objetivos previstos.

El sprint 7, que supuso un punto de inflexión en el desarrollo, los sobreesfuerzos de parte del equipo para cumplir la planificación fueron muy elevados en tareas de codificación y pruebas. En contraposición, el tiempo destinado a tareas metodologías fue muy inferior a la media. En los últimos sprints los esfuerzos se concentraron en tareas de implementación.

La figura 7.11 muestra gráficamente la distribución global de esfuerzos que se ha realizado en el proyecto.

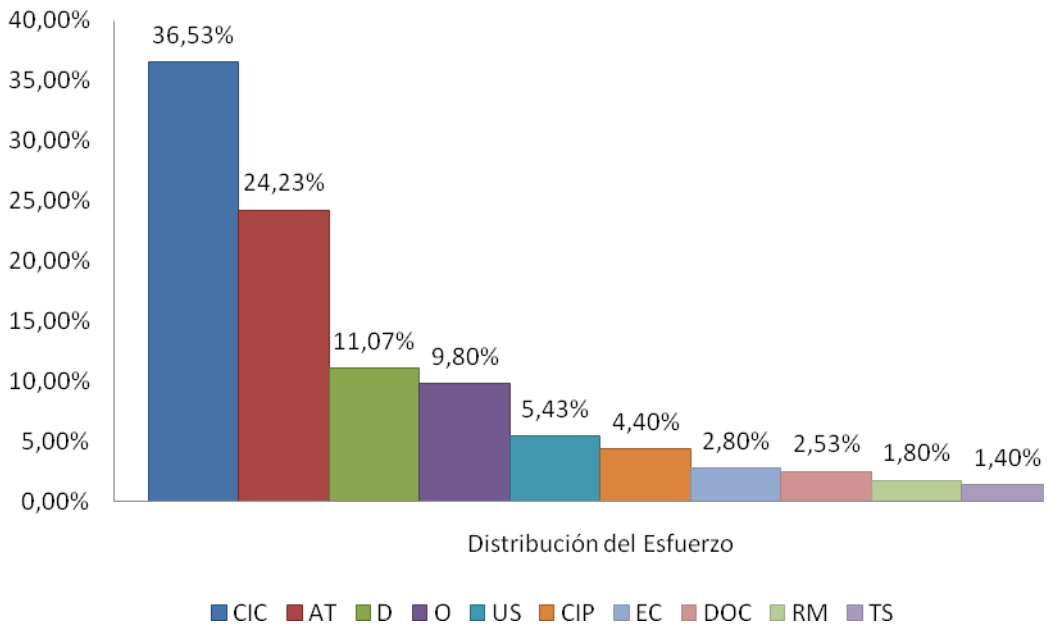


Figura 7.11 Distribución global de esfuerzos

Analizando en detalle esta distribución, cabe destacar el elevado porcentaje de tiempo destinado a otras tareas relacionadas con el experimento, un 9.8% del tiempo, y que suele estar reflejado en incidencias imprevistas. Esta situación repercute negativamente en el esfuerzo destinado al resto de tareas que rigen el cumplimiento de objetivos en sí del sprint, por lo que se recomienda sea considerada en la planificación.

Además es significativo, el pequeño porcentaje de tiempo destinado a pruebas, tan solo un 4,4%. La causa de este escaso valor tiene su origen en el enfoque como línea de producto que sigue TOPENbiogas y que facilita enormemente las tares relacionadas con la validación del producto. Finalmente, destacar los pequeños esfuerzos dedicados a la documentación del producto, pues la documentación se encuentra implícita en otro tipo de tarea como la definición de historias de usuario y la documentación del código en la fase de codificación. Esto demuestra que no se necesita excesiva documentación para la correcta evolución de un producto.

7.3. Calidad del Producto Obtenido

Uno de los aspectos que más satisfacción ha aportando tanto cliente como al equipo de desarrollo es la calidad del producto obtenido.

El hecho de que el producto TOPENbiogas este especialmente diseñado para ser adaptable a diferentes dominios de aplicación ha repercutido muy positivamente en este sentido, pues se conocía de antemano las partes variantes del producto que, por tanto, debían ser probadas tras el incremento de funcionalidad. En un principio una parte del equipo se dedicó a realizar la validación de este producto. No obstante, al constatar que no se encontraban fallos en el producto pues los aspectos a probar resultaban tan obvios que ya habían sido probados por los propios desarrolladores, se decidió que fuesen los propios desarrolladores quienes ejecutasen este trabajo. Aunque parezca una contradicción, pues se entiende que los desarrolladores no deben probar sus propios productos, no se encontraron defectos en las sucesivas entregas de este producto al cliente. Esta es una de las principales ventajas de la evolución de productos software, ya

que las tareas de pruebas se simplifican enormemente al saber de antemano que partes del producto son susceptibles de ser probadas.

En el producto simulador la tasa de defectos encontrados en la fase de pruebas sigue la distribución que aparece en la figura 7.12

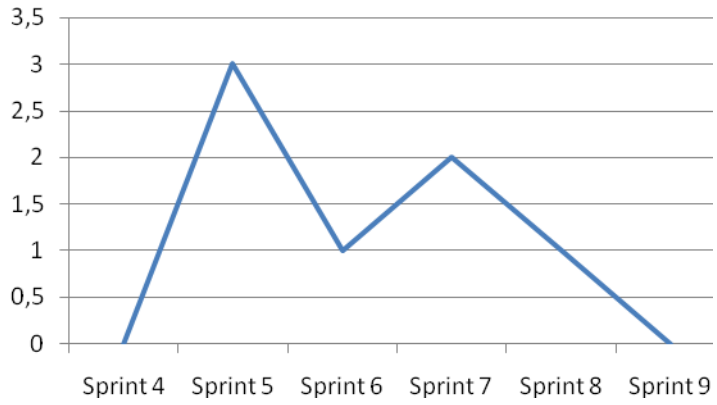


Figura 7.12 Evolución de la tasa de errores en el producto simulador

El número de defectos encontrados es muy bajo ya que el hecho de que dos personas se encarguen de construir este producto simultáneamente favorece la obtención de productos de calidad.

7.4. Esfuerzo de Adaptación

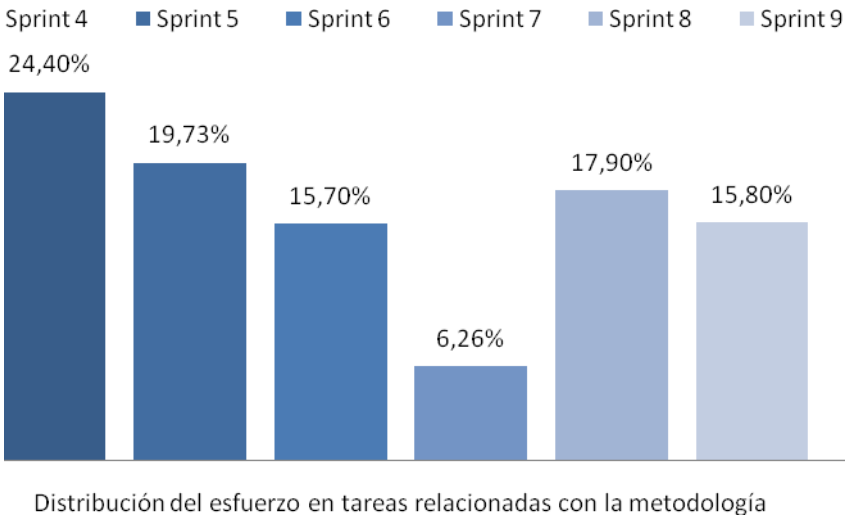


Figura 7.13 Evolución del esfuerzo dedicado a tareas relacionadas con la metodología SCRUM a lo largo del desarrollo

Adaptase a los nuevos hábitos establecidos por SCRUM, de talante más social que técnico, no es trivial pues la complejidad radica en el propio factor humano. Una métrica que nos puede dar una idea de esta situación es la variación del esfuerzo empleado en tareas relacionadas con aspectos metodológicos a lo largo del experimento,

tales como preparación y realización de reuniones. La figura 7.13 muestra gráficamente esta evolución. En los primeros sprints del desarrollo, la motivación del equipo propicio que todas las tareas relacionadas con la metodología fueran llevadas a cabo, aunque la falta de adaptación a la metodología hizo que consumieran un tiempo considerablemente más elevado al teóricamente establecido. Tiempo descontado de la realización de otro tipo de tareas que influyó decisivamente en el incumplimiento de objetivos. De hecho el número de horas de diferencia entre la media de dedicación a este tipo de tareas (46 horas por sprint) y su valor en el sprint 4 (sprint con mayor dedicación a tareas metodológicas con un valor aproximado de 68 horas) es de 22 horas, un porcentaje muy importante del esfuerzo dentro del sprint. A medida que avanzaba el desarrollo, el tiempo dedicado a este tipo de tareas se fue estabilizando. No obstante, apareció un nuevo enemigo, la desmotivación, que causó que en el sprint 7 el esfuerzo dedicado a estos aspectos fuese inferior al necesario. Superada esta circunstancia, y con conocimiento suficiente en la metodología el esfuerzo se estabilizó alrededor de 16% del total en cada sprint.

Dado que uno de los aspectos metodológicos que más dificultades ha causado en el desarrollo ha sido la definición de las necesidades del cliente, es interesante analizar del mismo modo como ha evolucionado el esfuerzo dedicado a definir y gestionar las historias de usuario. La figura 7.14 muestra esta distribución.

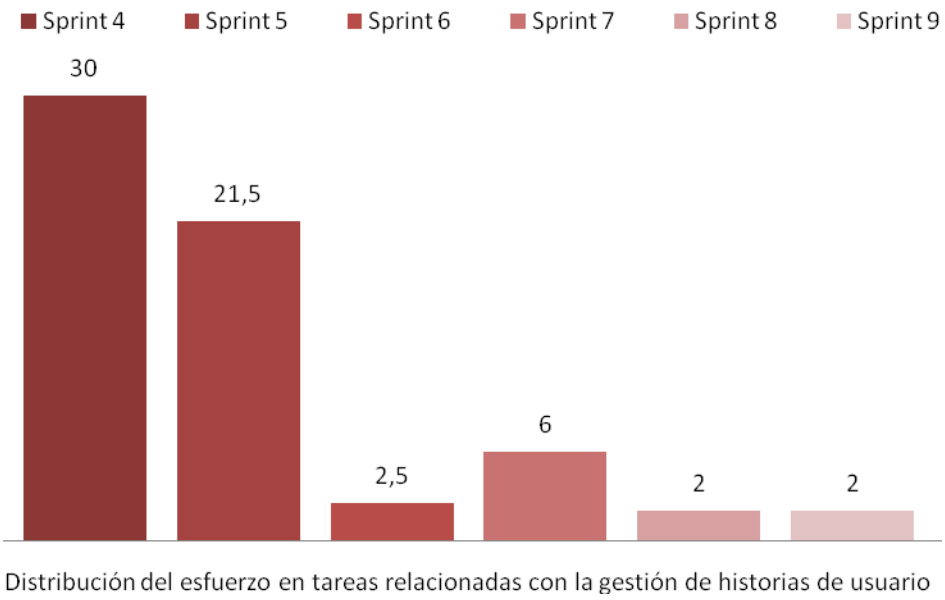


Figura 7.14 Evolución del esfuerzo dedicado a tareas relacionadas con la gestión de historias de usuario

Como se puede apreciar los primeros sprints fue uno de los grandes lastres arrastrados y, consecuentemente, una de las tareas a las que más esfuerzo se dedicó. Una vez superada la laguna metodología en este campo el esfuerzo destinado a esta fase se redujo radicalmente.

Por otro lado, la puntuación con la que fue calificado cada sprint nos proporciona también información sobre el grado de adaptación del equipo de trabajo a la metodología, pues las reuniones retrospectivas se centraban principalmente en aspectos metodológicos.

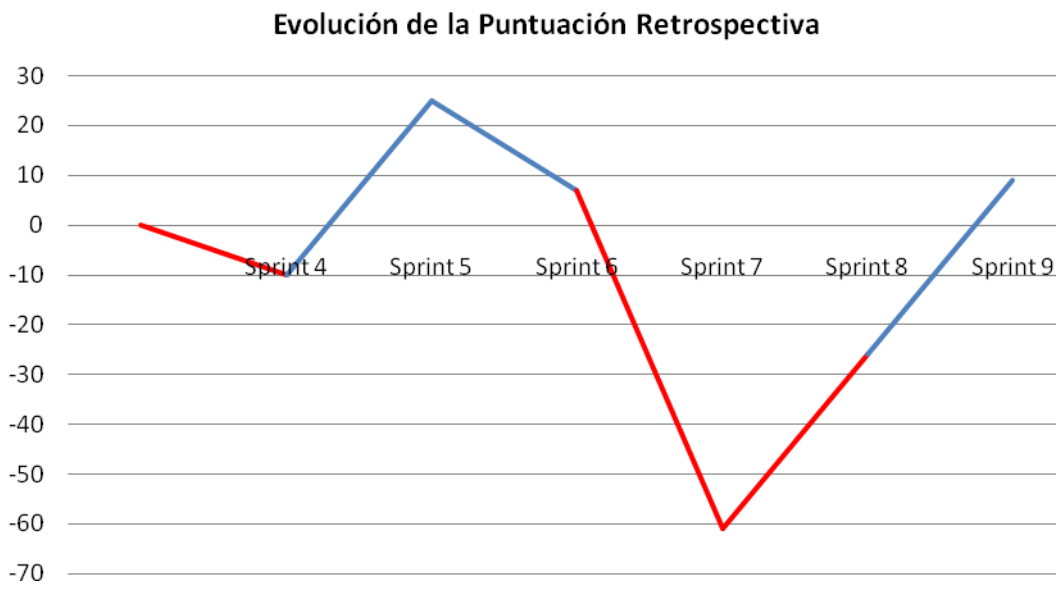


Figura 7.15 Evolución en la calificación de los sprints

En la figura 7.15 se muestra como tras un periodo breve de adaptación a la metodología el equipo comienza a adquirir progresivamente hábitos ágiles. No obstante, cuando parecía que la adaptación estaba prácticamente completada, consiguiendo altas calificaciones, hubo un periodo de desmotivación que provocó que muchos de los conceptos y hábitos adquiridos hasta el momento fuesen prácticamente olvidados. Asumido este periodo, el equipo comenzó a restablecer prácticas de desarrollo ágil y a adaptar los aspectos teóricos aprendidos a sus propias necesidades de desarrollo.

Finalmente, tras el análisis del esfuerzo de implantación de la metodología se puede concluir que como las metodologías ágiles se basan en principios y valores de ámbito sociológico más que técnico, donde el factor humano es fundamental en su desarrollo. Las características y complejidad inherente al comportamiento de las personas repercuten en que la adaptación a la metodología suela sufrir altibajos pues la adquisición de nuevos hábitos es una tarea que requiere un periodo prolongado en el tiempo.

7.5. Satisfacción General del Equipo

La satisfacción final, en líneas generales, fue positiva para todos los miembros del equipo, incluido el cliente. Las medidas de satisfacción han sido recogidas a través de cuestionarios, realizados al finalizar cada sprint, en los que los todos los componentes del proyecto debían valorar el grado con el que el sprint se había ajustado a sus necesidades, considerando tanto el resultado desde la perspectiva del producto como sus sensaciones respecto a la metodología, otorgando a cada sprint un valor comprendido entre 0 y 10.

La figura 7.16 muestra gráficamente una media de la satisfacción del equipo a lo largo del desarrollo junto a la satisfacción del cliente.

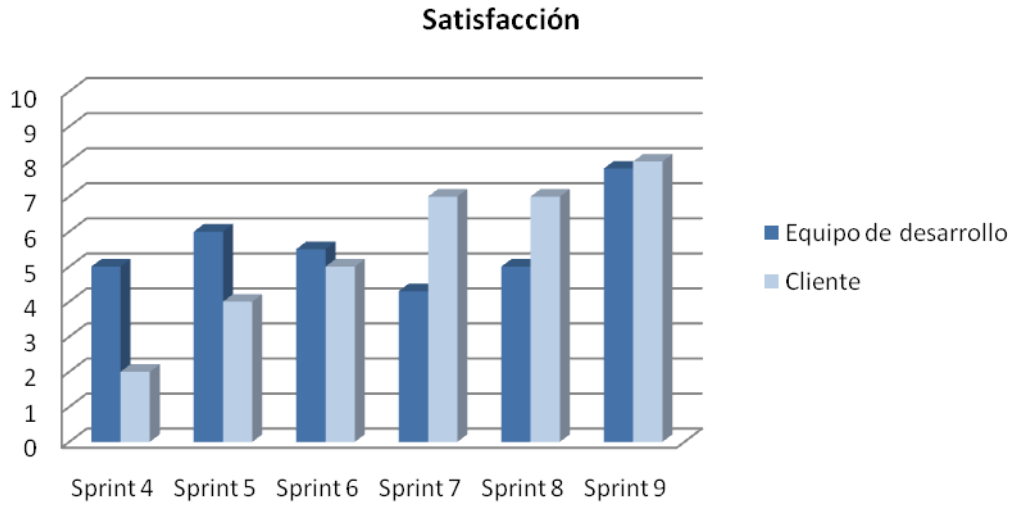


Figura 7.16 Evolución de la satisfacción del equipo y del cliente a lo largo del desarrollo

Como refleja la figura, la satisfacción del cliente ha ido incrementando a medida que avanzaba el desarrollo, debido principalmente a la consecución de objetivos. No obstante, aunque dicha satisfacción refleja claramente el agrado del cliente con la calidad del producto obtenido, el no concluir el proyecto cubriendo el 100% de los objetivos marcados en un inicio influye, evidentemente, en que su satisfacción no llegué a alcanzar el valor máximo.

Por otro lado, la evolución de la satisfacción del equipo, en la que se considera de forma más equilibrada la perspectiva industrial y metodológica, ha pasado por diferentes etapas. Dada la condición de equipo convencional y el posicionamiento un tanto escéptico frente al funcionamiento de las mismas, influyó en que el comienzo fuese complicado. La definición de las necesidades del cliente acaparó la mayoría de reflexiones en este periodo en la evolución del producto. Un componente del equipo reflexionaba: *"el sprint tiende al fracaso ya que la planificación se realiza desde un punto de vista erróneo, por lo tanto, la estimación de dicha planificación es irreal. Este punto de partida erróneo es debido a que se desconoce el estado inicial de partida, es decir, cuál es el estado actual de TOPENprimer, para modificarlo y para llevar a cabo el desarrollo de TOPENbiogas"*. Indudablemente, la satisfacción de la mayor parte de los componentes en este punto se vio minada por la fase de planificación. No obstante, se valoraron positivamente aspectos como la interacción entre el equipo. *"Hemos notado un considerable aumento de la motivación en el trabajo. La estrecha relación entre los componentes del equipo es muy satisfactoria ya que hace que la motivación se transmita de unos miembros a otros. Además, se ve favorecida la comunicación y la solución de problemas, repartiendo la responsabilidad entre todo el grupo de trabajo. Aunque no hayamos logrado cumplir los tiempos marcados, muchas de las tareas han avanzado de forma más rápida y segura"*.

A medida que aumentó el conocimiento y se fue adquiriendo experiencia en la metodología y en el dominio de aplicación del producto a evolucionar se fueron solventando las incertidumbres iniciales y, análogamente, la satisfacción comenzó a aumentar. *"Existe un mejor conocimiento de la metodología y una mayor conciencia de la necesidad de aplicarla adecuadamente. Esto hace que la metodología en este sprint* (sprint 5) *se esté aplicando de manera más efectiva, aunque todavía debamos mejorar algunos aspectos. Debido a esto el grupo está más cohesionado y motivado"*. No

obstante, este aumentó de conocimiento también provocó muchas reflexiones negativas, principalmente relacionadas con las herramientas o, mejor dicho, con la inadaptación a las necesidades del equipo de las herramientas utilizadas en el experimento (principalmente la herramienta de gestión de proyectos Rally). *"El uso de herramientas tiene bastantes obstáculos en el grupo. Aunque se ha mejorado existe un sentimiento general de rechazo"*, *"Creo que SCRUM como metodología de gestión y desarrollo es adecuada, aunque hay que limitar el uso de herramientas de gestión del mismo. Las herramientas de gestión de proyectos basadas en SCRUM son más una losa que un apoyo debido a su poca agilidad"*.

Los hechos acontecidos en el sprint 7 supusieron un punto de inflexión que hizo descender radicalmente la satisfacción del equipo. Cabe destacar que no sucedió lo mismo con el cliente pues los resultados obtenidos fueron satisfactorios, y la rápida reorganización del grupo ayudó a que algunos de los inconvenientes presentados fuesen prácticamente transparentes para el cliente. Superados los problemas del sprint 7, la satisfacción del equipo creció paulatinamente de ahí al final del proyecto. En línea general, las conclusiones finales a las que se llegó fueron favorables a la metodología. *"Tras la realización de este experimento considero que la metodología de desarrollo ágil realmente funciona y da resultados en la evolución de productos software, también deja ver claro que la experiencia es necesaria, lo que apunta estas metodologías como un proceso iterativo de mejora y adaptación continua"*, *"Gracias al desarrollo de TOPENbiogas el grupo está más cohesionado y motivado, se han fomentado las relaciones personales y nos conocemos todos mucho mejor, tanto en lo personal como en lo profesional. Esto es debido a que el trabajo en equipo durante un tiempo prolongado permite, tarde o temprano, que las personas se muestren tal cual son en sus distintas facetas"*, *"Estoy convencido de que esta metodología funciona. Creo que los argumentos son de sobra conocidos por todos los participantes en el proyecto. Se ha comprobado que el equipo está cohesionado, aunque existen roces personales, lógicos en cualquier equipo. El equipo de trabajo sabe redistribuir el esfuerzo y se genera un ambiente de confianza en la finalización con éxito del proyecto"*. Además, el sentimiento generalizado del equipo fue de gran satisfacción por el esfuerzo dedicado al proyecto y los objetivos conseguidos. *"Es importante analizar el rendimiento del equipo de trabajo. Creo que ha sido sobresaliente, fundamentalmente porque ha estado motivado"*, *"Sin aplicar SCRUM este experimento habría fracasado en poco tiempo ya que ha habido una porción importante de recursos que han tenido efecto de vaivén en el proyecto. Sin la ágil respuesta del grupo habría líneas de trabajo que habrían fracasado"*.

No obstante, muchos miembros apuntaban la necesidad de cambiar algunos de los aspectos metodológicos. *"[…] tendríamos que haber sido más ágiles únicamente realizando las reuniones necesarias (y no las impuestas) y con los miembros adecuados. Esto disminuiría la sensación de pérdida de tiempo en excesivas reuniones"*. También se reflexionó sobre algunos inconvenientes que nos habían provocado la falta de documentación electrónica al tratar de agilizar el desarrollo en este sentido. *"Observo dificultad para aplicar la metodología SCRUM si todo el equipo no trabaja físicamente en la misma sala. Esto es debido a que la agilidad en el desarrollo a veces causa no disponer de documentos importantes que puedan ser consultados, actualizados y gestionados por varios miembros del equipo mediantes herramientas. Por ejemplo, el modelo de clases del simulador lo tenemos en papel en la sala de trabajo. Y aunque lo tuviésemos escaneado y fuera accesible, yo no podría hacer cambios si estoy físicamente en otro lugar distinto al que esté el resto del equipo*".

Finalmente, desde una perspectiva de cultivo personal el experimento también resultó muy satisfactorio para los miembros del equipo, *"Pienso que el experimento ha sido de utilidad para todos los que hemos participado. En lo personal me ha permitido ahondar en un tipo de metodología que considero importante, una filosofía que ha de ser tenida en cuenta en los años que están por venir, una inyección de pragmatismo al mundo de la ingeniería del software"*.

7.6. Definición de las Necesidades del Cliente

Una de las líneas de investigación con más peso dentro del experimento se ha dedicado al estudio de la representación de la funcionalidad y de las necesidades del cliente, por tratarse de uno de los aspectos del proyecto donde más dificultades se han encontrado a lo largo del desarrollo. Estas dificultades han tenido un gran impacto en la realización de planificaciones poco precisas, principalmente en los primeros sprints del proyecto.

Los problemas identificados pueden clasificarse en cinco grupos: identificación de requisitos, granularidad, requisitos derivados, transversalidad y documentación de las necesidades del cliente a través de historias de usuario.

1) Identificación de requisitos

Un grave problema que hemos detectado a la hora de utilizar un desarrollo ágil es la importancia que adquieren todas aquellas necesidades relacionadas con la funcionalidad del sistema. Realmente el hecho de que la funcionalidad guíe el desarrollo no constituye un problema en sí mismo, hasta que este foco de interés se produce en detrimento de aquellos requisitos que también tienen gran importancia pero se apartan del foco de la funcionalidad. Los conocidos en metodologías convencionales como requisitos no funcionales. La figura 7.17 muestra las diferentes vistas que se producen del sistema.

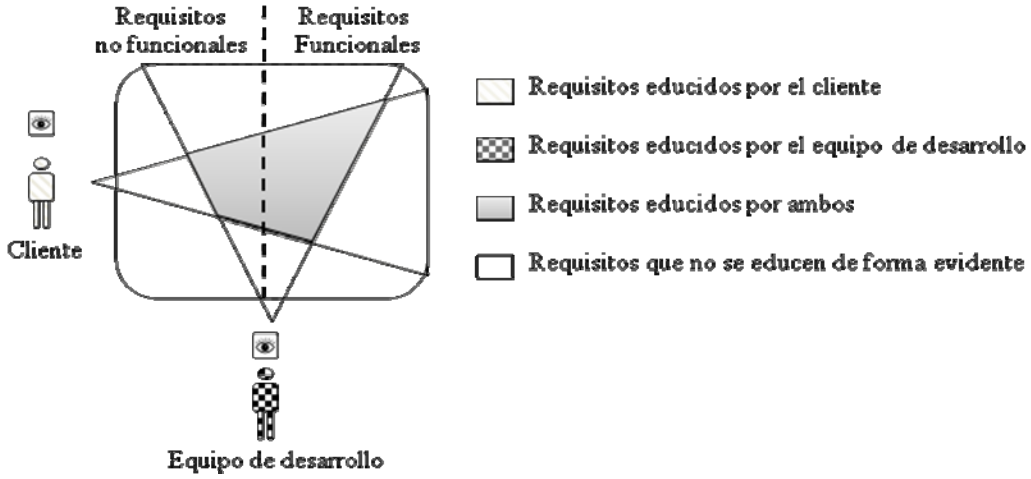


Figura 7.17 Vista del sistema desde la perspectiva del cliente y desde la perspectiva del equipo de desarrollo [68]

El cliente tiene una visión del producto representada por la zona rallada, dejando aspectos sin descubrir. Por otro lado, el equipo de desarrollo tiene otra perspectiva, representada por la zona con trama cuadriculada. Esta vista abarca aspectos que quedan fuera de la percepción del cliente, por lo que complementa su visión. Como puede apreciarse, hay zonas sobre las que no hay visibilidad. Esto se debe a que inicialmente no se dispone de la información suficiente, que va emergiendo según avanza el proyecto. En un enfoque tradicional, la perspectiva que se tiene del producto, como

máximo, es la del equipo de desarrollo. En las metodologías ágiles es la unión de ambas, lo que disminuye las áreas sin visibilidad.

No obstante, el conocimiento que se tenía del sistema y el material de partida existente, principalmente la especificación de requisitos formal del producto de partida TOPENprimer, desarrollado siguiendo una metodología convencional, ha permitido darse cuenta de que durante la evolución del producto con SCRUM se obviaban muchos detalles de gran importancia para el desarrollo del proyecto, principalmente aquellos referidos a aspectos diferentes a la funcionalidad del sistema. La solución que se propone en este estudio, consiste en, aprovechar la etapa de reevaluación y repriorización de requisitos al final de cada sprint para evaluar las historias de usuario, que representan requisitos funcionales, considerando los requisitos no funcionales que, por su naturaleza, se educen de forma menos evidente. En consecuencia, puede ocurrir que algunas historias de usuario no puedan validarse por causa de requisitos no funcionales y, por tanto, queden pendientes para su desarrollo en sprints posteriores. De esta forma, los requisitos no funcionales pueden emerger a lo largo del desarrollo. La figura 7.18 muestra este proceso.

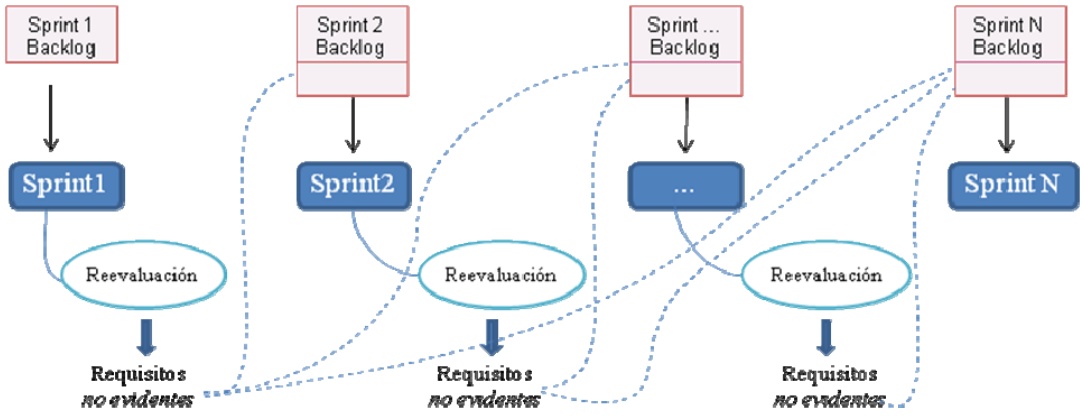


Figura 7.18 Ciclo de vida propuesto para un desarrollo ágil [68]

Es el caso, por ejemplo, de la historias de usuario US52 y US62 referentes a la construcción de un nuevo componente Gateway del producto TOPENbiogas para permitir la comunicación con la planta de biogás, aparecidas en el sprint 6 (ver sección 6.5 Sprint 6: buscando nuestro propio equilibrio).

2) Granularidad

Uno de los problemas que se ha detectado es que determinadas necesidades del usuario se ven con un nivel de detalle que no siempre es el que realmente necesita. Este problema se ha denominado nivel de granularidad. En las primeras iteraciones los componentes se ven con un bajo nivel de detalle y según avanza el desarrollo de los sprints aumenta este nivel de detalle.

La figura 7.19 refleja gráficamente como aumenta el número de elementos del Product Backlog a medida que avanza el desarrollo, pues aparecen muchas historias de usuario nuevas al disminuir el nivel de detalle con el que se ve cada componente.

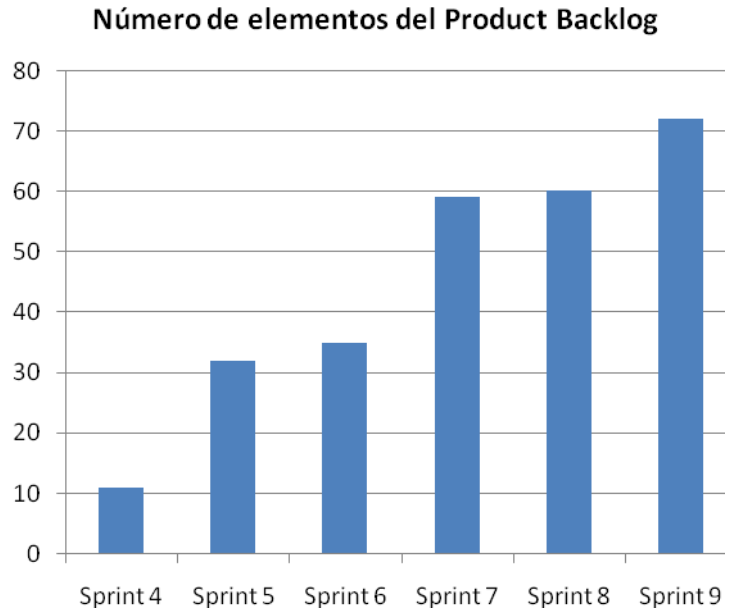


Figura 7.19 Nivel de granularidad para los elementos del Product Backlog

Al comienzo del proyecto, los tanques (trituración, pasteurización, hidrólisis y digestión) se veían como las variables a monitorizar por TOPENbiogas y los elementos a simular por el simulador. A medida que avanzó el desarrollo, y, a su vez, el conocimiento que se tenía del sistema, aparecían variables de más baja granularidad para cada uno de estos tanques, que tenían la entidad suficiente como para convertirse por si solas en historias de usuario para el componente TOPENbiogas. Así, en el sprint 5 aparecieron 18 historias de usuario nuevas relacionadas con el tanque de trituración; en el sprint 7, 23 relacionadas con el tanque de pasteurización; y, finalmente, en el sprint 9, 12 historias de usuario nuevas relacionadas con la operación, monitorización y validación del tanque de hidrólisis. El hecho de que en el producto simulador la implementación de un tanque fuese abordable en un único sprint, dada la menor complejidad respecto al producto TOPENprimer, hizo que no apareciesen historias de usuario a más bajo nivel para cada uno de los tanques a simular, sino tareas a completar dentro de la historia de usuario.

Las características de estas variables afectan al formato de los comandos de operación, que han sido definidos de una forma simplista. Esto lleva consigo rehacer componentes enteros de TOPENbiogas en el momento en el que alguno de estos comandos no se adapte al formato definido en la etapa anterior.

3) Requisitos derivados

Existen ocasiones en las que una necesidad que parece evidente y sencilla de obtener desde la perspectiva del cliente puede tener un gran impacto en el desarrollo del proyecto y descomponerse a su vez en múltiples necesidades. Este aspecto debe ser considerado en las planificaciones de los sprints.

4) Documentación de las necesidades del cliente

Existen muchas necesidades del cliente que se apartan del foco de la funcionalidad y resulta complicado definirlas a través de historias de usuario, ya que como indica Kassab en [69], la gestión de este tipo de necesidades es diferente al resto de requisitos y, por otro lado, muchos requisitos no funcionales suelen afectar transversalmente a

varias historias de usuario, cuestión que sería difícil de considerar bajo la naturaleza de las historias de usuario. Así, apareció la necesidad de crear un nuevo concepto para alojar esta información, las *historias de sistema*. Las historias de sistema que se han utilizado en este caso de estudio, se han definido como un elemento incorporado a las metodologías ágiles que se utiliza para recoger cualquier característica que los clientes/promotores quieran que el equipo produzca relacionada con aspectos diferentes a la funcionalidad del sistema.

5) Transversalidad

Uno de los problemas que más repercusión ha tenido en las tareas de gestión del proyecto hace referencia a todas aquellas necesidades del cliente que por su naturaleza afectan de forma transversal al proyecto y, por tanto, son difíciles de descomponer en historias de usuario, aspecto ya identificado, aunque no elaborado, en [70]. De hecho seguridad (safety) es un buen ejemplo de transversalidad. Las metodologías ágiles presentan importantes lagunas en el tratamiento de este tipo de requisitos, sobre todo en tareas de planificación, estimación del esfuerzo o validación de las mismas. En el experimento presentado se ha optado por tratarlas por separado bajo el concepto de historia de sistema, facilitando de este modo su gestión.

Los resultados obtenidos de este estudio se divulgarán en las Jornadas de Ingeniería del Software y Base de Datos 2008 (JISBD'08), a través del artículo "Metodologías ágiles desde la perspectiva de la especificación de requisitos funcionales y no funcionales" [68].

7.7. Distribución de la Funcionalidad

No debemos perder la perspectiva de que el mundo del software es un negocio que mueve millones de euros anualmente y el principal objetivo de un cliente al solicitar un producto software es aumentar el valor de su negocio y adquirir una ventaja competitiva, que a su vez, incremente sus ingresos.

Desde una perspectiva económica, uno de los aspectos metodológicos que más positivamente ha repercutido en la satisfacción del cliente ha sido el progresivo aporte de valor que el desarrollo del proyecto le ha proporcionado desde las primeras iteraciones. En contraposición a las metodologías convencionales, el cliente ha podido ir disfrutando incrementalmente de la funcionalidad y, por tanto, obteniendo ingresos desde el momento en que una nueva funcionalidad estaba lista para ser puesta en producción. Además, el desarrollo software incremental le ha permitido variar o perfeccionar aquellas características del producto que no se ajustaban a sus expectativas, bien porque el equipo de desarrollo no ha logrado captar todos los matices que implicaban sus necesidades, bien porque el propio cliente no conocía a priori, y en profundidad, cuáles eran sus necesidades (problema del tipo *"Sabré lo que quiero cuando lo vea"*). Al mismo tiempo, le ha facilitado la toma de decisiones de negocio, verificando tras la realización de cada sprint la dirección en la que progresaba el proyecto y su adecuación a las necesidades actuales de su negocio.

La figura 7.20 muestra cómo ha evolucionado la distribución de la funcionalidad operativa a lo largo del proyecto hasta completar el producto.

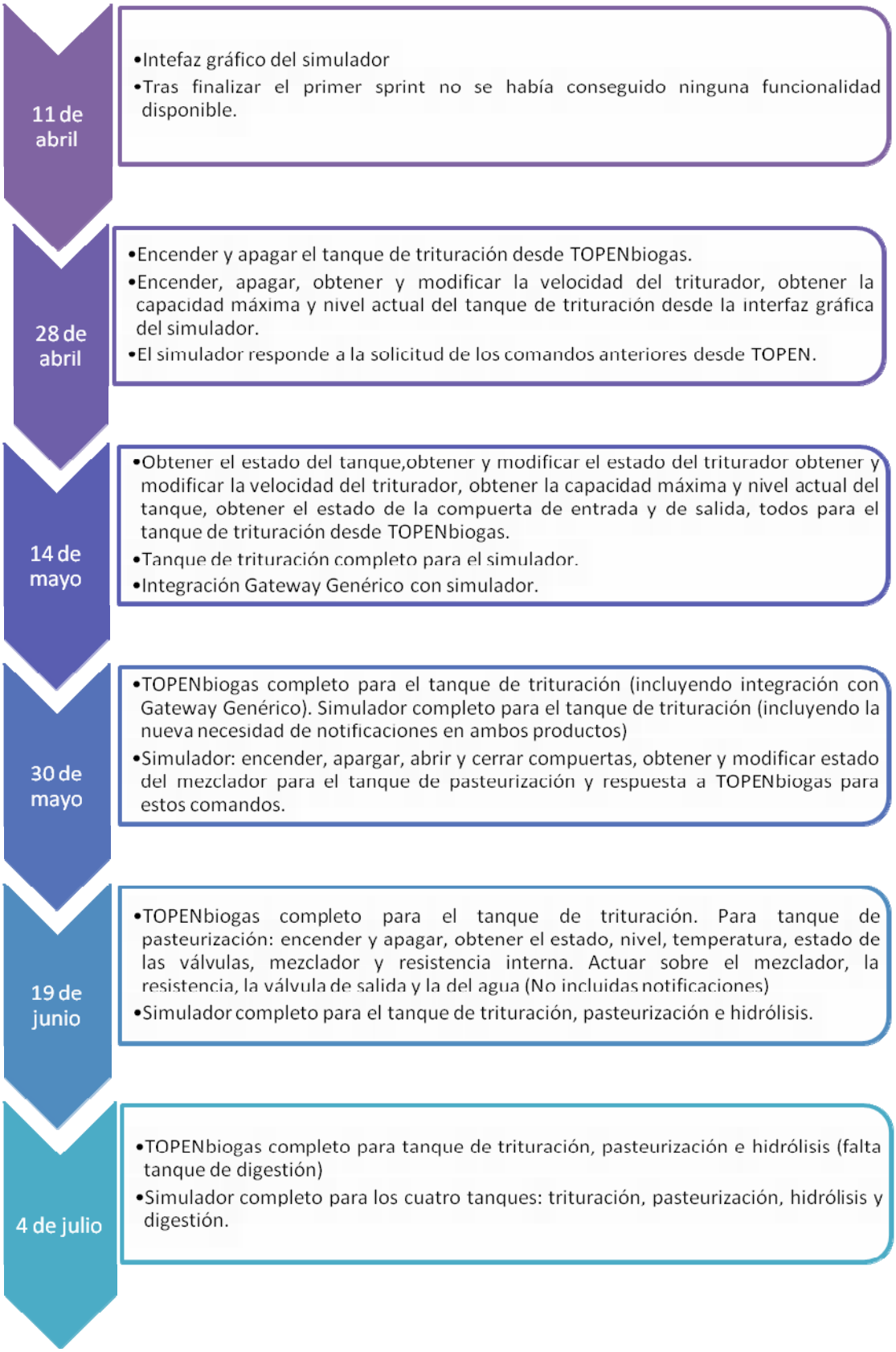
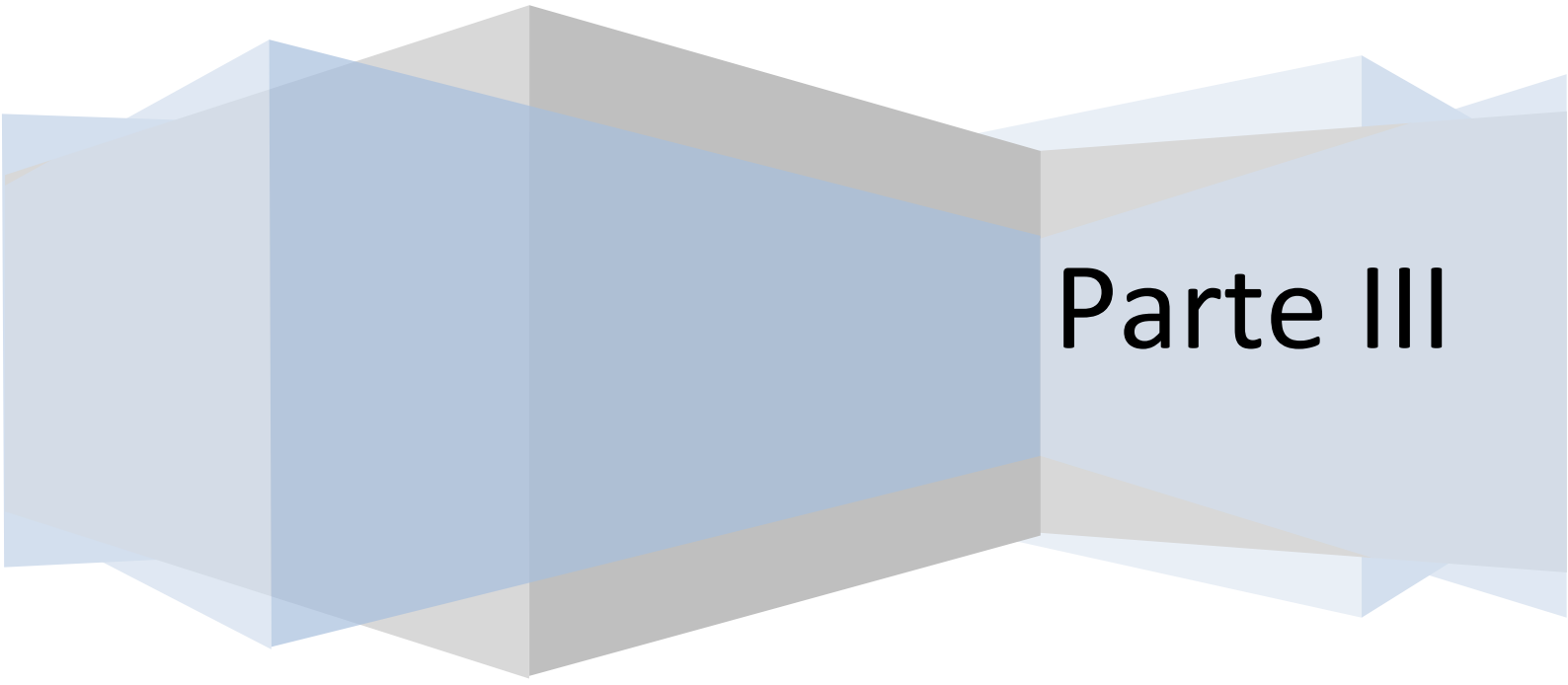


Figura 7.20 Evolución de la funcionalidad del sistema en el tiempo

Resumen, Conclusiones y Líneas Futuras de Estudio

Estudio de la Aplicación de Metodologías Ágiles para la Evolución de Productos Software



Parte III

Capítulo 8

Resumen, Conclusiones y Líneas Futuras de Estudio

Como cierre a este estudio se presentan las conclusiones a las que se ha llegado tras la realización del mismo. Este capítulo se encuentra estructurado en cuatro secciones. Por un lado, se presentan las aportaciones generales del trabajo realizado en este estudio. A continuación se indican las conclusiones a las que se ha llegado tras la realización del mismo respecto a los objetivos que han guiado la investigación. En una tercera sección se muestra la divulgación de los resultados obtenidos, que se ha realizado hasta el momento. Finalmente, se indican las futuras líneas de estudio de la investigación.

8.1. Aportaciones Generales

En esta tesis de máster se ha estudiado el proceso de evolución de un producto software concreto utilizando la metodología de desarrollo ágil SCRUM. El objetivo inicial era, basándonos en la experiencia adquirida tras la realización de un experimento diseñado para dar respuesta a interrogantes en esta línea de investigación, valorar la evolución de un producto si aplicamos una metodología ágil, analizando, entre otros aspectos, características del producto durante el desarrollo, aspectos de agilidad del equipo, la calidad del producto obtenido y el esfuerzo de adaptación a la metodología.

Para ello se han realizado, entre otros, los siguientes trabajos:

- Se ha llevado a cabo un análisis exhaustivo del estado de la investigación en el campo de las metodologías de desarrollo ágil para identificar objetivos con interés investigador. Se ha trabajado desde dos perspectivas: los estudios empíricos de aplicación de metodologías de desarrollo ágil en

general y, más específicamente, los estudios empíricos de aplicación de metodologías ágiles en la evolución de productos software.

Respecto a los estudios empíricos de aplicación de metodologías ágiles generales se han descubierto importantes incongruencias entre los resultados que arrojan. De este modo, existen estudios que concluyen un aumento del 337% de la productividad al utilizar este tipo de metodologías ante otros que indican un decremento del 44%, [2] y [3] respectivamente. Asimismo, se ha constatado la clara necesidad de realizar nuevos estudios empíricos que permitan aclarar las lagunas existentes en las metodologías ágiles [6, 7, 8]. Respecto a la evolución de productos, se ha detectado un enorme vacío en el estado de la investigación en este tipo de desarrollo utilizando metodologías ágiles. De hecho, solamente se tiene constancia de un estudio empírico en este sentido [7], que concluye que las metodologías ágiles permiten una suave evolución del producto eliminando complejidad en el proceso de evolución y aumentando la satisfacción del cliente.

En función de los trabajos existentes hasta el momento en este campo, las áreas de interés del grupo de investigación SYST y del proyecto Flexi, en el que se encuadra esta investigación, y las características del contexto en el que se desarrolla la investigación, se ha establecido como objetivo general que guíe el estudio valorar la evolución de un producto software si utilizamos una metodología ágil, como SCRUM, en el proceso de desarrollo. Este objetivo genérico ha sido descompuesto mediante cuestiones de competencia relativas a la evolución del producto, agilidad del desarrollo, calidad del producto obtenido y esfuerzo de adaptación a la metodología.

- En línea con los objetivos establecidos, se ha diseñado el experimento que da base al estudio. Se ha diseñando un plan de mediciones que permitiese obtener conclusiones basadas en datos empíricos pero que, a su vez, fuese ligero para que interfiriera lo menos posible en el desarrollo. Entre las medidas consideradas podemos destacar el tamaño del código fuente, medido en líneas de código, y las clases impactadas por cada historia de usuario para estudiar la evolución del producto obtenido a lo largo del desarrollo. El porcentaje de realización de las necesidades del cliente y el esfuerzo dedicado a cumplir estas necesidades en los distintos sprints del desarrollo con el objetivo de analizar la productividad del equipo a lo largo del mismo. Los defectos detectados en los productos para estudiar la calidad del producto obtenido. Y la satisfacción de los implicados en el proyecto con la metodología seguida así como el esfuerzo dedicado a cada una de las tareas del desarrollo, para analizar el esfuerzo de adaptación a la metodología. Asimismo, en esta fase de diseño, se ha elaborado una planificación general del estudio con una duración total de unos ocho meses y se han identificando las posibles amenazas para la validación del estudio tales como la dificultad de generalizar un desarrollo ágil al tener como principio la auto-organización o la influencia de la toma de métricas en el desarrollo.
- En la ejecución del experimento se ha desarrollado el proyecto de evolución del software TOPENprimer para obtener el software TOPENbiogas. Se trata de la evolución de un entorno para la operación, monitorización y pruebas de sistemas complejos de máquinas recreativas

interconectadas para adaptarlo al dominio de las plantas de producción de biogás, utilizando la metodología ágil SCRUM en el desarrollo. Como resultado de esta fase se ha obtenido el software TOPENbiogas.

- Durante el experimento se han realizado tareas de observación del mismo, abstrayendo las observaciones en datos a través del plan de mediciones diseñado con anterioridad. Por tanto, se ha prestado especial atención en observar la evolución del producto a lo largo del desarrollo, entendida como la evolución de la funcionalidad, del tamaño del código y de las clases impactadas; la calidad de las versiones del producto obtenidas al finalizar cada sprint, contemplando los defectos que presentaban durante y después del desarrollo; la agilidad en el desarrollo en función de los objetivos alcanzados en cada sprint respecto a el esfuerzo dedicado al mismo; y la adaptación del equipo a la metodología observando tanto la satisfacción de los implicados como otros indicadores en este sentido. Además, durante el experimento se han realizado también observaciones cualitativas como por ejemplo los problemas detectados para definir las necesidades del cliente.
- Durante y tras el experimento se ha realizado un proceso continuo de análisis del mismo. Se han analizado cada una de las medidas tomadas así como las percepciones y observaciones cualitativas del equipo de desarrollo.
- Finalmente, se ha realizado un proceso para recapitular la información obtenida en el proceso de análisis y obtener conclusiones, presentadas en este capítulo, con respecto a los objetivos perseguidos a través de respuestas a las cuestiones planteadas.

Los trabajos realizados verifican el objetivo general que guía el estudio, tanto desde una perspectiva investigadora como desde una perspectiva industrial.

8.2. Conclusiones del Estudio

El análisis de los resultados obtenidos parece indicar que las metodologías de desarrollo ágil limitan el coste de la evolución de productos software. No obstante, esta afirmación debería ser confirmada con la realización de nuevos estudios en esta línea de investigación. Analizando los resultados obtenidos se puede concluir:

1. *Evolución del producto durante el desarrollo*

   En línea con los resultados obtenidos en [7], se ha detectado que la evolución del producto software TOPENprimer utilizando una metodología ágil como SCRUM se ha desarrollado progresivamente (el tamaño del código y el número de clases impactadas crece gradualmente), aboliendo dificultades de complejidad, al realizar integraciones pequeñas de funcionalidad, e incrementando la satisfacción del cliente durante el desarrollo, pues la metodología le permitía ir disfrutando del producto incrementalmente, añadiendo mayor valor a su negocio.

   Se ha detectado que un aspecto fundamental en la evolución del producto siguiendo una metodología ágil es el conocimiento que se tenga del producto de partida y, en el caso de modificar el dominio de aplicación, el conocimiento que se tenga también de dicho dominio. La ausencia de un conocimiento profundo

por parte de los desarrolladores en el producto a evolucionar, TOPENprimer, provocó que el número de clases impactadas fuese muy elevado al comienzo del desarrollo, debido a la manipulación errónea de algunas de ellas, y fue estabilizándose a medida que éste avanza. La metodología ágil permite al grupo de trabajo ir adquiriendo este conocimiento progresivamente sin impedir la obtención de resultados desde las primeras fases de evolución.

También se ha constatado que la evolución continua permite adquirir experiencia en el proceso, reduciendo el riesgo en el espacio temporal y, por tanto, disminuyendo el esfuerzo de evolución progresivamente. De hecho, uno de los resultados obtenidos más reveladores hace referencia a que en el último sprint de desarrollo, dedicando un esfuerzo 60 horas inferior a la media se logró cumplir los objetivos planificados, de envergadura similar al resto de sprints.

2. *Agilidad en el desarrollo*

   En términos generales podemos concluir que el equipo ha sido ágil en la evolución del producto pues un 97% de las necesidades del cliente han sido cubiertas tras el mismo, en un espacio muy reducido de tiempo. Considerando que tanto el producto de partida TOPENprimer como el producto final TOPENbiogas son productos de alta complejidad, al tratarse de entornos de testeo para sistemas software complejos, y que el domino de aplicación al que se ha evolucionado el producto, las plantas de producción de biogás también es un dominio complejo, se puede concluir que los resultados obtenidos respecto a la productividad del equipo han sido muy satisfactorios. Además, la adaptación a la metodología y el progresivo incremento de conocimiento en el producto a desarrollar, han propiciado que la productividad del equipo se haya incrementado casi linealmente a lo largo del desarrollo.

   Por otro lado, el resultado de los cuestionarios que todos los implicados en el proceso rellenaban de forma periódica, indica que de manera unánime los componentes del equipo consideran que el uso de la metodología SCRUM en la evolución del producto ha propiciado que el desarrollo se caracterizase por una mayor rapidez en la resolución de problemas y un excelente rendimiento del equipo de trabajo. Factores como la cohesión e interacción entre el grupo de trabajo, la motivación en el desarrollo y la continua interacción con el cliente se han considerado muy positivos en este sentido.

3. *Calidad del producto obtenido*

   El hecho de que en una etapa previa a la ejecución del experimento se identificasen las partes variable y aquellas que permanecen constantes en el producto a evolucionar, en la línea de investigación que pretende establecer TOPENprimer con un enfoque línea de producto, ha sido beneficioso pues el conocimiento que se tenía de la parte susceptible de modificación ha favorecido el proceso de pruebas obteniendo un producto sin errores.

   En el producto simulador, la calidad del producto obtenido ha sido también alta, favorecido, principalmente, por las prácticas de pair-programming y de refactorización utilizadas en este desarrollo.

4. *Esfuerzo de adaptación a la metodología*

   El esfuerzo inicial de adaptación a la metodología ha sido considerable, principalmente en la adquisición de nuevos hábitos. Durante el desarrollo este proceso ha pasado por diferentes etapas, alternando puntos de mayor y menor

esfuerzo. No obstante, al final del mismo podemos considerar un alto grado de adaptación en la mayor parte del equipo.

5. *Satisfacción del cliente y el equipo de desarrollo*

   La satisfacción del cliente ha ido incrementando a lo largo del desarrollo por el aumento incremental en la consecución de objetivos en la evolución del producto. Se ha detectado que uno de los aspectos que más positivamente repercute en la satisfacción del cliente al evolucionar un producto utilizando una metodología ágil es la continua aportación de valor a su negocio, al permitir de forma progresiva ir disfrutando de la funcionalidad del producto.

   La satisfacción del equipo de desarrollo ha sufrido variaciones a lo largo del proceso causadas, principalmente, por aspectos dependientes del contexto universitario en el que se ha desarrollado la investigación, ajenos al experimento. Se puede concluir en este sentido, que utilizar la metodología SCRUM en proyectos paralelos resulta complicado pues merma la motivación y satisfacción de los involucrados en varios proyectos a la vez. No obstante, la reflexión final, recogida a través de cuestionarios, de la mayoría de los componentes del equipo es favorable a la metodología.

La falta de madurez de las metodologías ágiles implica que aún existan importantes lagunas en su aplicación y se hagan necesarias ciertas “correcciones”. Este trabajo ha descrito diversos problemas encontrados en la evolución del producto con metodologías ágiles. La mayor parte de los problemas detectados se centran en la definición de las necesidades del cliente utilizando metodologías de desarrollo ágil. Este análisis ha sido posible gracias a que el producto objeto de la investigación es en realidad la evolución de otro existente, desarrollado utilizando metodologías convencionales y del que se disponía de una especificación de requisitos tradicional. El proceso de definición de las necesidades del cliente ha tenido algunas dificultades: dificultad de identificación de algunas necesidades, principalmente aquellas que se apartan del foco de la funcionalidad, necesidades del cliente que por su naturaleza afectan de forma transversal al proyecto (problema identificado como transversalidad), requisitos derivados, granularidad y documentación de algunas necesidades del usuario. Además de identificar y tipificar los problemas que pueden surgir, se aportan soluciones para que, siguiendo un desarrollo ágil, conseguir llegar a la meta a tiempo y en presupuesto. Respecto a los requisitos no funcionales, mientras la identificación de algunos se podría asociar a historias de usuario, no es el caso de otros cuyo estudio se debería abordar de forma explícita e independiente de las historias de usuario bajo el concepto propuesto de *historia de sistema*. Del mismo modo se propone incluir una adecuada etapa de reevaluación al finalizar cada sprint para identificar aquellos requisitos que no son obvios desde la perspectiva del cliente.

8.3. Divulgación de Resultados

El estudio presenta una experiencia real en la que un grupo de desarrollo ha realizado el esfuerzo de ir analizando empíricamente el proceso de evolución de un producto software utilizando la metodología ágil SCRUM, a medida que se iban acometiendo los sucesivos sprints.

Además de la realización de la presente tesis de máster, los resultados obtenidos en este estudio han sido divulgados en dos medios, uno de carácter científico y otro con una orientación empresarial:

- ***"Metodologías ágiles desde la perspectiva de la especificación de requisitos funcionales y no funcionales"*** Pilar Rodríguez, Agustín Yagüe, Pedro P. Alarcón, Juan Garbajosa. XIII Jornadas de Ingeniería del Software y Bases de Datos, JISBD'08. Octubre 2008.

  Por un lado, algunas de las conclusiones obtenidas en este estudio se han plasmado en una publicación dedicada a la definición y gestión de las necesidades del cliente en las metodologías de desarrollo ágil. El artículo analiza los problemas que aparecen al aplicar SCRUM relacionados con la especificación de requisitos funcionales y no funcionales. Para paliar algunos de los problemas identificados, se propone, por un lado, un nuevo artefacto, las historias del sistema y, por otro, una tarea de reevaluación que se realizaría al finalizar cada sprint de SCRUM. El artículo ha sido aceptado en el congreso JISBD'08 y será publicado en el mes de octubre.

- ***"Requisitos convencionales frente a Historias de usuario en Metodologías ágiles"***. Jornadas Solo Requisitos 2008.

  Además, con el objetivo de valorar el alcance que las metodologías de desarrollo ágil tienen en la industria software, principalmente en el área de evolución software, y transferir los resultados investigadores obtenidos en este estudio al sector empresarial, se participó en las *Jornadas Solo Requisitos 2008*. Concretamente se realizó una ponencia en la que, basados en la experiencia adquirida en el experimento, se trato la representación de las necesidades del cliente a través de historias de usuario frente a los requisitos formales de las metodologías convencionales analizando las ventajas e inconvenientes en cada caso.

8.4. Líneas Futuras de Estudio

Como futuras líneas de estudio se pretende profundizar en algunos aspectos de la investigación como las características del producto obtenido en la evolución y la productividad del proceso para confirmar las conclusiones obtenidas en este estudio. El análisis de resultados presentado nos lleva a pensar que las metodologías ágiles pueden ser una alternativa atractiva para enfocar la evolución y mantenimiento de productos software. Por tanto, se considera adecuado seguir trabajando para confirmar los prometedores resultados obtenidos. Se pretenden continuar el estudio explorando otros caminos, combinando observaciones cuantitativas y cualitativas, que permitan ampliar la valoración de la evolución de productos software utilizando metodologías de desarrollo ágil. Dada la escasez de estudios dedicados a esta área de investigación, se necesitarían, además, futuros estudios que profundizasen en este aspecto para generalizar los resultados obtenidos.

Aunque los resultados llevan a pensar que el desarrollo ha transcurrido con gran agilidad no se poseen datos empíricos que afirmen esta idea pues no se dispone de datos históricos sobre el desarrollo utilizando metodologías pesadas. Como futura línea de

investigación se pretende repetir el estudio utilizando en el desarrollo una metodología convencional para comparar las productividades conseguidas en cada caso.

Como futuras líneas de investigación respecto al área de ingeniería de requisitos para representar las necesidades del cliente mediantes historias de usuario en metodologías de desarrollo ágil, sería necesario, profundizar en la identificación de requisitos no funcionales. Consecuentemente, se necesitan estudios que permitan conocer mejor la relación entre la aproximación ágil y este tipo de requisitos. Las etapas de reevaluación y priorización parecen llamadas a tener un papel destacado. Asimismo, se pretende profundizar en el concepto de historia de sistema para albergar aquella información que se considera no tiene cabida bajo las historias de usuario. Finalmente, sería muy útil estudiar el impacto que la aparición de estos requisitos provoca sobre el código.

Referencias

[1] Boehm, B.: A View of 20th and 21st Century Software Engineering. In: 28th international conference on Software engineering, pp. 12--29. Shanghai, China (2006)

[2] Dalcher, D., Benediktsson, O., y Thorbergsson, H., "Development Life Cycle Management: A Multiproject Experiment", Proceedings of the 12th International Conference and Workshops on the Engineering of Computer Based Systems (ECBS'05), 2005

[3] Wellington, A., Briggs, T., y Girard, C.D., "Comparison of Student Experiences with Plan-Driven and Agile Methodologies", Proceedings of the 35th ASEE/IEEE Frontiers in Education Conference, 2005.

[4] ITEA 2 Flexi. In: http://www.flexi-itea2.org/index.php

[5] Fleeger, P.: "Software Engineering. 3ed." Prentice Hall, 1999.

[6] Dyba, T., Dingsoyr, T.: Empirical Studies of Agile Software Development: A Systematic Review, Information and Software Technology doi: 10.1016/j.infsof.2008.01.006 (2008)

[7] Capiluppi, A., Fernandez-Ramil, J., Higman, J., Sharp, H.C., Smith, N.: An empirical study of the evolution of an agile-developed software system. In: 29th International Conference on Software Engineering (ICSE), pp. 511{518. (2007)

[8] Mann, C., Maurer, F.: A case Study on the Impact of Scrum on Overtime and Customer Satisfaction. In: Proceedings of the Agile Development Conference (ADC'05). IEEE Computer Society (2005)

[9] www.agilealliance.org

[10] Boehm, B.W.: Software Engineering Economics. In: Prentice Hall (1981). ISBN: 0138221227

[11] Lindvall, et al..: Empirical Findings in Agile Methods. In: Lecture Notes In Computer Science; Vol. 2418. Proc. of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - XP/Agile Universe. Berlin, Germany (2002)

[12] Nerur, S., Mahapatra, R., Mangalaraj, G.: Challenges of migrating to agile methodologies. In: Commun. ACM, vol. 48, no. 5, pp. 72-78 (2005)

[13] Qumer, A., Henderson-Sellers, B.: An evaluation of the degree of agility in six agile methods and its applicability for method engineering. In: Information and Software Technology. (2007)

[14] Beck, K., et al.: The Agile Manifesto. Manifesto for Agile Software Development. www.agilemanifesto.org

[15] Cao, L., Ramesh, B.: Agile Requirements Engineering Practices: An Empirical Study. In: IEEE Computer Society Press Los Alamitos, CA, vol 25, pp-60-67 (2008)

[16] Robinson, H. Sharp, L., "The social side of technical practices in eXtreme Programming and Agile Processes in Software Engineering", Lecture Notes in Computer Science. Berlin: Springer Verlag, 2005, pags 139-147.

[17] Ilieva, S., Ivanov, P., Stefanova, E., “Analyses of an agile methodology implementations”, Proceedings 30th Euromicro Conference, 2004, IEEE Computer Society Press, pags 326 - 333.

[18] Canós, J.H., Letelier, P., Penadés, M.C., “Metodología Ágiles en el Desarrollo Software”. VIII Jornadas de Ingeniería de Software y Bases de Datos, JISBD 2003.

[19] Takeuchi H. and Nonaka I., The New New Product Development Game. Harvard Business Review (January 1986), pp 137-146, 1986.

[20] Schwaber, K., Beedle, M.: Agile Software Development with SCRUM. In: Conchango (2006) ISBN: 0130676349

[21] Cohen, M. User Stories Applied for Agile Software Development. The Addison-Wesley Signature Series. 2004. ISBN: 0321205685

[22] Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change. Second Edition. In: Addison Wesley Professional, Boston (November, 2004)

[23] http://www.extremeprogramming.org/

[24] A. Cockburn, Crystal Clear: A Human-Powered Methodology for Small Teams: Addison-Wesley, 2004, ISBN 0-201-69947-8.

[25] http://www.cystalmethodologies.org

[26] J. Stapleton, DSDM: Business Focused Development, Second ed: Pearson Education, 2003, ISBN 978-0321112248.

[27] S.R. Palmer and J.M. Felsing, A Practical Guide to Feature-Driven Development. Upper Saddle River, NJ: Prentice Hall, 2002, ISBN 0-13-067615-2.

[28] M. Poppendieck and T. Poppendieck, Lean Software Development - An Agile Toolkit for Software Development Managers. Boston: Addison-Wesley, 2003, ISBN 0-321-15078-3.

[29] Merisalo-Rantanen, H., Tuunanen T., Rossi, M.: Is Exreme Programming Just Old Wine in New Bottles: A Comparison of Two Cases. In: J. Database Management, vol 16, nº4, pp-41-61 (2005)

[30] Paetsch, F.E., Eberlein, A., and Maurer, F.: Requirements Engineering and Agile Software Development. In: Proc. 12th IEEE Int’l Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises (Wetice), pp. 308 -313. Linz, Austria (2003)

[31] Sillitti, A., Ceschi, M., Russo, B., Succi, G.: Managing Uncertainty in Requirements: A Survey in Documentation-Driven and Agile Companies. In: 11th IEEE International Software Metrics Symposium (Metrics 05), pp17. IEEE Press (2005)

[32] Ceschi, M., Sillitti, A., Succi, G., De Panfilis, S. : Project Management in Plan-Based and Agile Companies. In : IEEE Software, vol. 22, no. 3, pp. 21-27 (2005)

[33] Sutherland, J.: Inventing and Reinventing Scrum in five Companies. In: http://www.agilealliance.org/system/article/file/888/file.pdf, (Accesed May 2008)

[34] Schwaber, K.: Agile Project Management with Scrum. In: Microsoft Press, Redmond, WA (2004)

[35] Svensson, H., y Host, M., “Views from an organization on how agile development affects its collaboration with a software development team”, Lecture Notes in Computer Science, vol. 3547. Berlin: Springer Verlag, 2005, pags 487-501

[36] Pinheiro, F.A.C.: Requirements Honesty. In: Proc. 2002 International Workshop Time-Constrained Requirements Eng. (2002).

[37] Grünbacher, P., Hofer, C.: Complementing XP with Requirements Negotiation. In: Proceedings 3rd Int. Conf. Extreme Programming and Agile Processes in Software Engineering., pp.105--108. Springer (2002)

[38] Nawrocki, J., Jasinski, M., Walter, B., Wojciechowski, A.: Extreme Programming Modified: Embrance Requirements Engineering Practices. In: Proc. IEEE Joint International Conference Requirements Engineering. (RE 02). IEEE CS Press, pp. 303-310 (2002)

[39] Araujo, J., Ribeiro, J.C.: Towards an Aspect-Oriented Agile Requirements Approach. In: Proceedings 8th International Workshop Principles of Software Evolucion, pp 140-143. IEEE Press (2005)

[40] Lee, M.: Just-in-Time Requirements Analysis-the Engine That Drives the Planning Game. In: Proceedings 3rd International Conference Extreme Programming and Agile Processes in Software Engineering., pp.138--141. Springer (2002)

[41] Denise M. Woit: Requirements interaction management in an eXtreme programming environment: a case study. In: 27th International Conference on Software Engineering, pp489-494 ICSE 2005 (2005)

[42] Brede Moe, N., T. Dybå, T. Dingsøyr, "Understanding self-organizing teams in agile software development". pp.76-895 19 th Australian Conference on Software Engineerging (Marzo, 2008)

[43] Halpern, K., I. "Machine independence: its technology and economics" Communications of the ACM, Vol. 8, p. 782-785 (1965)

[44] Wegner, P. "Research directions in software technology". Proceedings from 3rd international conference on Software Engineering, p243-259 (1978)

[45] Lehman, M.M., Belady, L.A. "Program Evolution: Processes of Software Change". London: Academic Press (1985)

[46] Bennett, K.h., Xu, J. "Software Services and Software Maintenance" Proceedings of the Seventh European Conference of Software Maintenance and Reengineering, p3-12 (2003)

[47] Madhavji, N., Fernandez-Ramil, J., Perry, D.E., "Software evolution and feedback – theory and practice". Wiley (2006)

[48] Boehm, B., Turner, R., "Balancing Agility and Discipline. A Guide for the Perplexed" ISBN 0321186125 Addison-Wesley (2004)

[49] Chapin, N., "Agile Method's Contributions in Software Evolution". Proceedings of the 20th IEEE International Conference on Software Maintenance, p.522-522 (2004)

[50] Wernick P. and Hall T., 'The Impact of Using Pair Programming on System Evolution: a Simulation-Based Study', Proc. ICSM 2004, 11-14 Sept: 422 – 426. (2004)

[51] Pries-Heje, J., Lindwall, D. "Evolution or Maintenance of Quality Software: An exploratory interview study in nine Swedish organisation". Proceedings of IRIS (2006)

[52] Alarcón, Pedro P. "Especificación de un modelo de operaciones aplicable a procesos de desarrollo y operación de sistemas con software". PhD thesis, Facultad de Informática. Universidad Politécnica de Madrid. 2008.

[53] Kitchenham, B.A.; S. L. Pfleeger; L. M. Pickard; P. W. Jones; D. C. Hoaglin; K. El Emam; and J. Rosenberg, "Preliminary Guidelines for Empirical Research in Software Engineering," IEEE Transactions on Software Engineering, vol. 28, No. 8, pp. 721-733, 2002.

[54] Williams, Laurie A.; Layman, Lucas; Krebs, William, "Extreme Programming Evaluation Framework for Object-Oriented Languages Version 1.4", NCSU Computer Science TR-2004-18, June 2004.

[55] Endres, A; Rombach D, "A Handbook of Software and Systems Engineering, Empirical Observations, Laws and Theories", Institut Experimentelles Software Engineering, Fraunhofer. Ed. Pearson Addison Wesley, 2003.

[56] B. Magro, J Garbajosa, J. Perez-Benedi: A Software Product Line Definition for Validation Environments. In: Software Product Lines Conference SPLC 2008.

[57] Alarcón Pedro P. and Garbajosa J., "Identifying application key knowledge through system operations modeling" in Proceedings of the 6th IEEE International Conference on Cognitive Informatics (ICCI'07), (Lake Tahoe, California), pp. 246–254, IEEE CS Press, August 2007.

[58] B. Boehm, "Software Risk Management: Principles and Practices" IEEE Software, vol. 8. Pp. 32-41, 1991.

[59] ECSS-E-40 European Cooperation for Space Standardization. ISSN 1028-396X. Marzo, 2005

[60] http://www.rallydev.com/products/editions/

[61] http://subversion.tigris.org/

[62] http://www.eclipse.org/

[63] http://www.egroupware.org/

[64] Diaz, Y., Garbajosa, J., "Guidelines for results collecting-practices". FLEXI project deliverable. Project info: http://www.flexi-itea2.org. ITEA2 proposal no. 06022

[65] Juristo, N., Moreno, A.M., "Basic of Software Engineering Experimentation". Universidad Politécnica de Madrid.

[66] Hartmann, D., Dymond, R., "Appropiate Agile Measurement: Using Metrics and Diagnostics to Deliver Business Value". Proceedings of AGILE 2006 Conference (AGILE'06)

[67] J. Díaz, Agustín Yagüe, Pedro P. Alarcón and Juan Garbajosa. "A Generic Gateway for Testing Heterogeneous Components in Acceptance Testing Tools". Aceptado en 7th IEEE International Conference on Composition-Based Software Systems (ICCBSS 2008).Madrid. Spain. February 2008

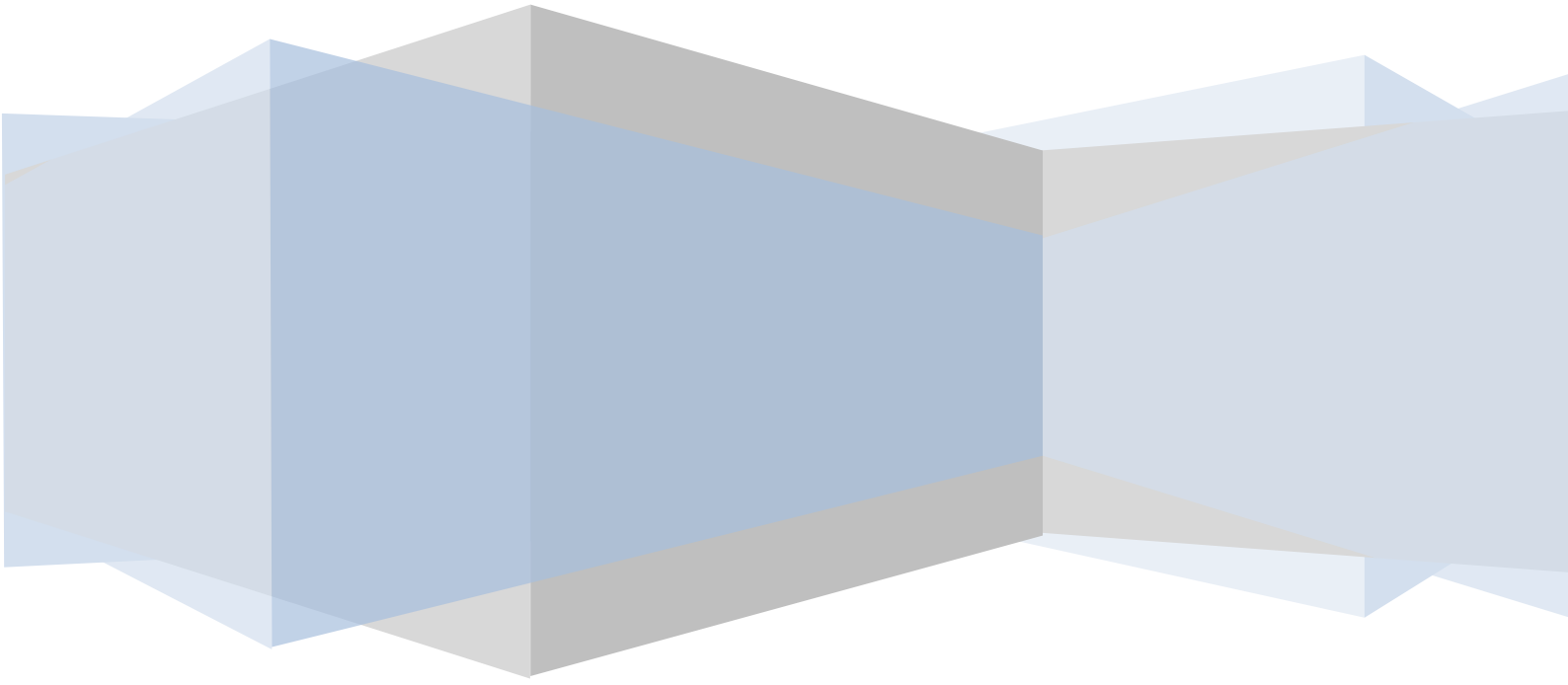
[68] Rodriguez, P., Yagüe, A., Alarcón, P.P., Garbajosa, J., "Metodologías ágiles desde la perspectiva de la especificación de requisitos funcionales y no funcionales". 13th Conference on Software Engineering and Databases (JISBD'08)

[69] Kassab, M., Daneva, M., Ormandjieva, O.: Scope Management of Non-Functional Requirements. In: EUROMICRO '07: Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, pp.409—417. Washington (2007)

[70] Rodriguez-Dapena, P., "Software safety certification: a multidomain problem," *Software, IEEE* , vol.16, no.4, pp.31-38, Jul/Aug 1999

Anexos

Estudio de la implantación de Scrum en equipos con experiencia en metodologías convencionales



A. *Ejemplo acta reunión de planificación*

Proyecto: **Flexi - Adaptación de TOPEN**
Documento: **Reunión de planificación del Sprint 5.**
Versión: **0.1**
Autor: **Pilar Rodríguez González** (SYST-UPM)
Fecha: **--**

En este documento se presenta tanto la agenda como los resultados obtenidos en la reunión celebrada para planificar el Sprint 5 del desarrollo del proyecto.

Reunión de planificación del sprint 5:

Fecha: --
Hora: 13'00 – 14'00

- **Agenda de la reunión**:
  - Planificación del Sprint 5.
    Temas a tratar:
    - Introducción a la reunión llevada a cabo por Scrum Master. 3 minutos
    - Product owner definirá los objetivos de este sprint, resumirá el product backlog e indicará la fecha de finalización de este sprint. (10 minutos)
    - Se seleccionarán las historias de usuario que se han de satisfacer en este sprint y, en base a ellas, se identificarán las tareas que se deben realizar y se estimará el tiempo necesario para realizarlas. (40 minutos)
    - Se asignarán tareas a cada uno de los miembros del equipo, definiendo de este modo su compromiso (5 minutos)
    - Se definirá cómo se validará el sprint y los productos que se deben obtener cómo resultado. (5 minutos)
    - Fecha de entrega del sprint.
    - Lugar y hora de la reunión diaria

- **Resultados obtenidos**:
  - Planificación del Sprint 5 (13'00-14'30)
    - **OBJETIVO DEL SPRINT**

      Adaptación de TOPEN que permita operar, monitorizar y probar el tanque de trituración que forma parte de las plantas de producción de energía a partir de Biogás, centrándose en los comandos básicos (encendido, apagado y control de la velocidad del triturador)
    - **PRODUCTOS RESULTADO**
      - Aplicación del simulador
      - Aplicación de Topen
      - Informe de pruebas de la aplicación del simulador y la aplicación de Topen
      - Informe de cambios (documento con los cambios realizados para adaptar Topen según el objetivo de este sprint)

- **SPRINT BACKLOG**

TESTING ENGINEERGIN (OPERADOR TOPEN)

US43 Recibir notificaciones del tanque de trituración

US19 Encender tanque de trituración

US28 Apagar tanque de trituración

US30 Obtener velocidad del triturador del tanque de trituración

US31 Modificar velocidad del triturador del tanque de trituración

OPERADOR SUT

US42 Operar tanque de trituración a través del simulador

- **TAREAS PARA CUMPIR LAS HISTORIAS DE USUARIO DEL SPRINT BACKLOG**

COMPONENTE	**ID**	**TAREA**	**TIEMPO ESTIMADO PARA SU REALIZACIÓN**
TOPEN	T1	Revisión modelo	1 día
	T2	Gramática	½ día
	T3	Generar TopenEngine	3 días
	T4	Acondicionar BD	½ día
	T5	Interfaz gráfico	3 días
	T6	Gateway	2 días
SUT	T7	Recepción comandos	1 día
	T8	Tratamiento comando	5 días
	T9	Envío respuesta	1 día
	T10	Diseño pruebas	3/2 días
	T11	Ejecución de pruebas	3/2 días

- **ASIGNACIÓN DE TRABAJO**

	T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**
Componente 1	**x**	**x**		**x**							
Componente 2	**x**	**x**	**x**	**x**	**x**	**x**					**x**
Componente 3	**x**	**x**	**x**							**x**	**x**
Componente 4							**x**	**x**	**x**		
Componente 5							**x**	**x**	**x**		
Componente 6	**x**	**x**									
Componente 7						**x**	**x**	**x**	**x**		**x**
Componente 8	**Soporte técnico**										
Componente 9											

- **ENTREGA DEL SPRINT: --**
- **REUNIÓN DIARIA:**
  Lugar: Laboratorio L5
  Hora: De 14’00 a 14’15

B. *Ejemplo acta reunión de revisión*

Proyecto:	**Flexi - Adaptación de TOPEN**
Documento:	**Sprint 5 Review.**
Versión:	**0.1**
Autor:	**Pilar Rodríguez González** (SYST-UPM)
Fecha:	**--**

En este documento se presentan los resultados obtenidos en la reunión de revisión del Sprint 5 del proyecto de adaptación del TOPEN al dominio de biogás. En primer lugar se describen los objetivos propuestos para este sprint (especificados en la reunión de planificación del mismo) y se comparan con los resultados obtenidos a la finalización del Sprint. El documento finaliza con una estimación del porcentaje de realización del sprint.

Sprint 5 Review:

Fecha: --
Hora: 12’00 – 13’00

- **Agenda de la reunión**:
  - Breve descripción del objetivo del sprint, así como de las historias de usuario que se propusieron cumplir en el mismo.
  - Revisión del estado de las historias de usuario al finalizar el sprint: in progress, completed and accepted
  - Revisión del estado de las tareas involucradas en aquellas historias de usuario que no han sido aceptadas.
  - % de realización del sprint.

- **Resultados obtenidos**:
  - **Descripción del objetivo y las historias de usuario**

    Objetivo del Sprint 5: Adaptación de TOPEN que permita operar, monitorizar y probar el tanque de trituración que forma parte de las plantas de producción de energía a partir de Biogás, centrándose en los comandos básicos de encendido y apagado del mismo.

    Historias de usuario a cumplir:

TESTING ENGINEERGIN (OPERADOR TOPEN)

US43 Recibir notificaciones del tanque de trituración
US19 Encender tanque de trituración
US28 Apagar tanque de trituración
US30 Obtener velocidad del triturador del tanque de trituración
US31 Modificar velocidad del triturador del tanque de trituración

OPERADOR SUT

US42 Operar tanque de trituración a través del simulador

- **Revisión estado historias de usuario**

TESTING ENGINEERGIN (OPERADOR TOPEN)

US19 Encender tanque de trituración	Completed
US28 Apagar tanque de trituración	Completed
US30 Obtener velocidad del triturador del tanque de Trituración	In progress
US31 Modificar velocidad del triturador del tanque de Trituración	In progress
US43 Recibir notificaciones del tanque de trituración	In progress

OPERADOR SUT

US42 Operar tanque de trituración a través del simulador	In progress

- **Revisión estado tareas**

TESTING ENGINEERGIN (OPERADOR TOPEN)

Sprint 5: Tanque trituración	
US19 Encender tanque de trituración	Completed
ID TAREA	**Estado**
Revisión Modelo	**Completed**
Gramática	**Completed**
Generar TopenEngine	**Completed**
Revisión y acondicionamiento BD	**Completed**
Interfaz Gráfica	**Completed**
Gateway	**Completed**
Diseño Pruebas	**Completed**
Ejecución Pruebas	**Completed**

US28 Apagar tanque de trituración	Completed
ID TAREA	**Estado**
Revisión Modelo	**Completed**
Gramática	**Completed**
Generar TopenEngine	**Completed**
Revisión y acondicionamiento BD	**Completed**
Interfaz Gráfica	**Completed**
Gateway	**Completed**
Diseño Pruebas	**Completed**
Ejecución Pruebas	**Completed**
US30 Obtener velocidad del triturador del tanque de trituración	In progress
ID TAREA	**Estado**
Revisión Modelo	**Completed**
Gramática	**Completed**
Generar TopenEngine	**In progress**
Revisión y acondicionamiento BD	**Completed**
Interfaz Gráfica	**Completed**
Gateway	**In progress**
Diseño Pruebas	**Completed**
Ejecución Pruebas	**In progress**
US31 Modificar velocidad del triturador del tanque de trituración	In progress
ID TAREA	**Estado**
Revisión Modelo	**Completed**
Gramática	**Completed**
Generar TopenEngine	**In progress**
Revisión y acondicionamiento BD	**Completed**
Interfaz Gráfica	**Completed**
Gateway	**In progress**
Diseño Pruebas	**Completed**
Ejecución Pruebas	**In progress**
US43 Recibir notificaciones del tanque de Trituración	In progress
ID TAREA	**Estado**
Revisión Modelo	**Completed**

Gramática	**Completed**
Generar TopenEngine	**In progress**
Revisión y acondicionamiento BD	**In progress**
Interfaz Gráfica	**Completed**
Gateway	**In progress**
Diseño Pruebas	**Completed**
Ejecución Pruebas	**In progress**

OPERADOR SUT

Sprint 5: Tanque trituración	
US42 Operar tanque de trituración a través del simulador In progress	
ID TAREA	**Estado**
TA67 Encender el tanque de trituración	**Completed**
TA68 Realizar operación de obtener velocidad del agitador	**Completed**
TA69 Realizar operación de modificar velocidad del agitador	**Completed**
TA70 Realizar operación obtener capacidad máxima	**Completed**
TA74 Realizar operación obtener capacidad actual	**Completed**
TA75 Realizar operación de abrir compuerta	**In progress**
TA76 Realizar operación de cerrar compuerta	**In progress**
TA96 Apagar tanque de trituración	**Completed**
TA97 Petición de encendido el tanque de trituración	**Completed**
TA98 Petición de apagado del tanque	**Completed**
TA99 Petición de obtener la velocidad del agitador	**Completed**
TA100 Respuesta a la petición de obtener la velocidad del agitador	**Completed**
TA101 Petición de modificación de la velocidad del agitador	**Completed**
TA102 Petición de obtener la capacidad máxima	**Completed**
TA103 Respuesta a la petición de obtener la capacidad máxima	**Completed**
TA104 Petición de obtener la capacidad actual	**Completed**
TA105 Respuesta a la petición de obtener la capacidad actual	**Completed**
TA106 Petición de abrir compuerta	**In progress**
TA107 Solicitud de cerrar compuerta	**In progress**
TA116 Diseño de pruebas	**Completed**
TA117 Ejecución de pruebas	**In progress**

- **% realización sprint**

  TOPEN: (28 tareas completadas/40 tareas a realizar) = 70%

  SUT: (15 tareas completadas/21 tareas a realizar) = 77%

C. *Normas de puntuación de sprints*

- Los aspectos que se han realizado correctamente en cada sprint sumarán 5 puntos.
- Un aspecto realizado correctamente en un sprint que se realiza incorrectamente en un sprint posterior restará 10 puntos.
- Un aspecto realizado correctamente en un sprint anterior y que en el sprint actual se deba mejorar restará 7 puntos.
- Los aspectos positivos que se mantienen de un sprint al siguiente sumarán 3 puntos.
- Los aspectos que se pueden mejorar no suman ni restan ningún punto en el momento en el que se detectan.
- Un aspecto que se pueda mejorar sumará 7 puntos si se mejora en el siguiente sprint.
- Un aspecto que se pueda mejorar restará 5 puntos si no se mejora en el sprint siguiente.
- Un aspecto que se pueda mejorar y se convierta en un error en el sprint siguiente restará 7 puntos.
- Cada error cometido restará 5 puntos.
- Cada error corregido sumará 10 puntos.
- Cada error no corregido restará 5 puntos.
- Cada error corregido pero que se deba mejorar sumará 7 puntos