Universitat Autònoma de Barcelona

escola d'enginyeria

2244

Automatización de sistemas de desarrollo ágil

– Scrum: Team & Role

Memoria del Proyecto de Fin de Carrera
de Ingeniería Informática
realizado por
Héctor Mudarra Teruel
y dirigido por
Jordi Pons Aróztegui

Bellaterra, Junio de 2010

El abajo firmante, Jordi Pons Aróztegui

Profesor de la Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el trabajo que corresponde a esta memoria ha sido realizado bajo

su dirección por Héctor Mudarra Teruel

Y para que conste firma la presente.

_____

Firmado: Jordi Pons Aróztegui

Bellaterra, Junio de 2010

El abajo firmante, Susana Ramos García

de la empresa, UNIT4 domiciliada en Barberá del Vallés.

CERTIFICA:

Que el trabajo que corresponde a esta memoria ha sido realizado en la empresa bajo su supervisión, mediante el convenio para la realización de prácticas en empresas en aplicación de programas de cooperación educativa, que prevé el RD 1497/1981, del 19 de junio, modificado por el RD 1845/1994, del 9 de septiembre, firmado con la Universitat Autònoma de Barcelona.

Así mismo, la empresa tiene conocimiento y da el visto bueno al contenido que se detalla en esta memoria.

_____

Firmado: Susana Ramos García

Junio de 2010

A mi familia.

Índice

Capítulo 0. Sobre el convenio entre UNIT4 y la UAB. 1
0.1. Sobre la empresa....2
Capítulo 1. Introducción. 3
1.1. Scrum: desarrollo ágil de software....4
1.2. Objetivos....5
1.3. Situación actual....6
1.4. Contenido de la memoria....6
Capítulo 2. Metodología del proyecto: Scrum. 9
2.1. Introducción histórica....9
2.2. Ciclo de vida Scrum....10
2.3. Roles....11
2.4. Materiales de soporte usados....13
2.4.1. Product Backlog....13
2.4.2. Sprint Backlog....14
2.4.3. Gráficas Burndown....15
2.4.4. Panel de seguimiento (Task Board)....17
2.4.4.1. Variante usada....18
2.5. Reuniones....19
2.5.1. Release planning + sizing....19
2.5.2. Sprint planning....21
2.5.3. Daily Scrum....22
2.5.4. Sprint Review....23
2.5.5. Sprint Retrospective....23
Capítulo 3. Análisis. 25
3.1. Requisitos funcionales....25
3.2. Requisitos no funcionales....26
3.3. Viabilidad....26
3.3.1. Especificaciones....27
3.3.2. Viabilidad técnica....27
3.3.3. Viabilidad operativa....27
3.3.4. Viabilidad temporal....28
3.3.5. Viabilidad legal....28
3.3.6. Alternativas....28
3.3.7. Planificación....29
3.3.8. Previsión general....29
3.3.9. Diagrama de Gantt....30
3.3.10. Product Backlog....31
3.3.11. División según Sprint....32

Capítulo 4. Diseño. **33**
4.1. Herramientas karat....33
4.2. Modelo físico de la aplicación....34
4.3. Control de permisos....37
Capítulo 5. Implementación. **39**
5.1. Evolución del proyecto: Sprint Backlog....39
5.2. Implementación de los requerimientos....47
5.3. Estructura de paquetes de la aplicación....50
5.4. Resultados....51
5.4.1. Funcionalidades....51
5.4.2. Pruebas....56
Capítulo 6. Conclusiones. **57**
6.1. Resultados....57
6.2. Metodología....58
6.3. Valoración personal....59
6.4. Líneas de futuro....59
Glosario **61**
Bibliografía **63**

Índice de figuras por capítulo

Capítulo 0. Sobre el convenio entre UNIT4 y la UAB.

Figura 1.Logotipo de UNIT4....2

Capítulo 1. Introducción.

Capítulo 2. Metodología del proyecto: Scrum.

Figura 2.Ciclo de vida Scrum....11

Figura 3.Comunicación entre los roles Scrum....13

Figura 4.Product Backlog....14

Figura 5.Sprint Backlog....15

Figura 6.Burndown Product Backlog....16

Figura 7.Burndown Sprint Backlog....16

Figura 8.Pendiente teórico del Sprint Burndown....17

Figura 9.Panel de seguimiento usado....19

Figura 10.Variante del Planning Poker usada....21

Figura 11.Ciclo de vida Scrum: Reuniones....24

Capítulo 3. Análisis.

Figura 12.Tabla de previsión general....29

Figura 13.Diagrama de Gantt del proyecto....30

Figura 14.Product Backlog: inicial....31

Figura 15.Product Backlog: historias añadidas....31

Figura 16.Product Backlog: historias comunes....31

Figura 17.Historias según Sprint: previsión inicial....32

Capítulo 4. Diseño.

Figura 18.Diagrama de clases genérico de una aplicación que usa karat....34

Figura 19.Entidad-Relación: Colaborador....34

Figura 20.Entidad-Relación: Rol....35

Figura 21.Entidad-Relación: Aptitudes del Colaborador....35

Figura 22.Entidad-Relación: Equipo....35

Figura 23.Entidad-Relación: Plaza....35

Figura 24.Entidad-Relación: Miembro....35

Figura 25.Entidad-Relación: Vacante....36

Figura 26.Entidad-Relación: Reunión....36

Figura 27.Entidad-Relación: Convocante y Convocados....36

Figura 28.Entidad-Relación: Plantillas de Equipo y Reunion....37

Figura 29.Diagrama de clases de control de permisos....38

Capítulo 5. Implementación.

Figura 30.Sprint 1: Previsión....40

Figura 31.Sprint 1: Burndown....40

Figura 32.Sprint 2: Previsión....41

Figura 33.Sprint 2: Burndown....41

Figura 34.Sprint 3: Previsión....42
Figura 35.Sprint 3: Burndown....42
Figura 36.Sprint 4: Previsión....43
Figura 37.Sprint 4: Burndown....43
Figura 38.Sprint 5: Previsión....44
Figura 39.Sprint 5: Burndown....44
Figura 40.Sprint 6: Previsión....45
Figura 41.Sprint 6: Burndown....45
Figura 42.Sprint 7: Previsión....46
Figura 43.Sprint 7: Burndown....46
Figura 44.Casos de uso: Roles....47
Figura 45.Casos de uso: Niveles de aptitud....47
Figura 46.Casos de uso: Colaboradores....47
Figura 47.Casos de uso: Historial de colaboradores....48
Figura 48.Casos de uso: Impresión de colaboradores....48
Figura 49.Casos de uso: Plantillas de equipo....48
Figura 50.Casos de uso: Equipos....48
Figura 51.Casos de uso: Asignación de colaboradores....49
Figura 52.Casos de uso: Impresión de equipos....49
Figura 53.Casos de uso: Plantillas de reuniones....49
Figura 54.Casos de uso: Reuniones....50
Figura 55.Casos de uso: Consulta global....50
Figura 56.Consulta global: Product Backlog....53
Figura 57.Consulta global: Sprint Backlog....54
Figura 58.Consulta global: Team & Role....55
Figura 59.Consulta global: Incidencias....55
Capítulo 6. Conclusiones.

Capítulo 0. Sobre el convenio entre UNIT4 y la UAB.

La empresa UNIT4 mantiene desde el curso 2002/2003 un convenio con la Universidad Autónoma de Barcelona, con el objetivo de que un grupo de estudiantes se incorpore cada año como becarios a la plantilla de la empresa. Dichas incorporaciones establecen un contrato de becario con una duración de 500 a 560 horas, donde los estudiantes tienen contacto con el ambiente y las condiciones laborales de una empresa del sector del desarrollo de software.

El tema de los proyectos a realizar surge de los intereses de la empresa previamente consultados con la universidad. Estos proyectos sirven a los estudiantes como proyecto de fin de carrera, que realizarán ya sea incorporándose a un equipo de desarrollo o bien formando un equipo entre los becarios.

Al tratarse de un convenio entre la universidad y la empresa donde el trabajo será hecho en la empresa, se designa a un tutor adicional para el alumno dentro de ésta. Es por ello que el alumno tendrá dos tutores para el proyecto, pese a que el tutor universitario seguirá siendo el tutor principal como en cualquier otro proyecto y el tutor de la empresa hará de guía al alumno con temas más prácticos o bien relacionados con la empresa.

Por otro lado, al ser este un proyecto con convenio, se realizarán diferentes presentaciones de éste: las de la universidad, para defender su proyecto como PFC y demostrar su base teórica y de investigación; y las de la empresa, para defender su proyecto como un producto comercial o aplicable a los intereses de la empresa.

0.1. Sobre la empresa.

Se fundó en el año 1963 bajo el nombre de Centro de Cálculo de Sabadell. En sus inicios, y como su nombre indica, se trataba de un centro de cálculo que daba soporte y gestión financiera a distintas empresas de la comarca del Valles Occidental, realizando para éstas operaciones de cálculo sobre tarjetas perforadas. Años más tarde empezó a abrirse al mercado del software, en concreto a la gestión ERP y CRM. Allí empezó el desarrollo de las herramientas *karat*, que conforman un *framework* para la creación de software. Estas destacan por su capacidad de creación y personalización de las aplicaciones con bastante sencillez y sin necesidad de unos conocimientos informáticos muy avanzados.

Su producto bandera durante el transcurso de los años han sido las llamadas soluciones *ekon* (actualmente *UNIT4 ekon*). Con el nombre de ekon bautizan a sus diferentes *suites* hechas con las herramientas karat y le añaden un segundo nombre dependiendo del producto que se trate. Estas soluciones ekon conforman un producto base a partir del cual se amolda la aplicación al cliente. Algunas aplicaciones ekon son por ejemplo:

ekon finanzas para gestiones financieras.

ekon salus para gestión de personal de clínicas.

ekon común conjunto de aplicaciones de gestión de empresas, personal, colaboradores, países... que el resto usan como base.

En la actualidad la empresa se ha convertido en la filial española del grupo UNIT4. Este grupo, con sede en Holanda, es proveedor de soluciones ERP y CRM para empresas de rango medio y actualmente es el cuarto mayor proveedor del sector en su campo, según la analista de mercado IDC (2008). Entre sus clientes hay tanto pymes como empresas de la Administración Pública y Sanidad.

Figura 1. Logotipo de UNIT4.

Capítulo 1. Introducción.

Cuando se quiere desarrollar un proyecto software primero hay que tener claro qué es lo que se quiere obtener y luego preguntarse cómo hacerlo. El cómo acostumbra a ser siempre una decisión por parte de los analistas o bien de los programadores, aunque en raras excepciones se pide una forma concreta de hacer las cosas. Sin embargo el qué no depende del equipo sino del cliente que pide la aplicación, y para transmitir ese qué debe de hacerlo con palabras, esquemas o dibujos. Una vez se conoce el qué y se ha tomado una decisión sobre el cómo, se entra en la fase de desarrollo donde se pasa un tiempo hasta obtener un resultado funcional.

El modelo clásico de desarrollo de un proyecto software es el modelo en cascada. En este modelo, una vez han quedado debidamente definidos todos los requisitos, se realiza una división en tareas y se les asigna una prioridad. Entonces se hace una planificación temporal con estas tareas y se empieza el desarrollo siguiendo lo especificado. Si todo va según lo planeado el proyecto terminará en el tiempo previsto. Si no ha ido según lo planeado cuando llegue la fecha final habrán quedado pendientes aquellas tareas con una menor prioridad.

Por tanto, usando el modelo clásico, se habrá obtenido un resultado al cabo de un tiempo de desarrollo y se presentará lo realizado al cliente. Generalmente es en este momento donde surgen problemas ya que el cliente puede no quedar del todo satisfecho con el proyecto. Esto es debido a que resulta muy difícil transmitir exactamente lo que uno piensa y probablemente en la definición de requerimientos que se hizo tiempo atrás no logró plasmar lo que se pretendía. Por otro lado también es posible que el cliente haya decidido cambiar de idea en algunos aspectos a mitad de desarrollo y sus nuevos conceptos no concuerden con los resultados. Sea como sea, el resultado final no será del todo satisfactorio para el cliente y es muy probable que haya que hacer modificaciones que ocasionen pérdida de mucho tiempo invertido en el desarrollo debido a cambios en los requerimientos.

Analizando el modelo clásico se encuentran principalmente dos puntos que resultan conflictivos:

El primero es la dificultad de una especificación de requerimientos que sea concorde con lo que quiere el cliente. Como ya se ha comentado anteriormente aunque se haga exactamente lo que se pretendía es probable que se quieran hacer cosas nuevas o se haya cambiado de idea a mitad del desarrollo. Al no ser un conflicto que depende del equipo de desarrollo se desestima la búsqueda de solución por esta vía.

El segundo es el tiempo que transcurre entre la definición de requerimientos y la muestra de resultados. Este período de tiempo está directamente relacionado con la magnitud del proyecto a desarrollar, sin embargo aquellos proyectos con un mayor tiempo acostumbran a ser también aquellos que tienen más inversión y donde los costes de un mal resultado son mayores. En este caso sí que el equipo de desarrollo tiene control y se puede buscar una solución.

Una solución que ha nacido para combatir los largos tiempos de espera entre la definición y el resultado y la poca flexibilidad de cambio que tienen los proyectos son las metodologías de desarrollo de software ágil. En este proyecto se estudia y se hace uso de una, en concreto el método *Scrum*.

1.1. Scrum: desarrollo ágil de software.

Las metodologías de desarrollo de software ágil introducen un gran cambio respecto a las metodologías clásicas. Su principal diferencia es el uso de iteraciones. Esto significa que todo el tiempo que se dedicaba en metodología clásica a realizar el proyecto se segmenta en pequeños plazos, llamados iteraciones, donde se realizan todas las fases (planificación, análisis de requerimientos, diseño, implementación, pruebas y documentación) y se presenta software terminado y funcional en cada una de ellas.

El hecho de seccionar el desarrollo en iteraciones conlleva grandes cambios respecto al modelo clásico. El más significativo es que el cliente podrá ver resultados funcionales cada cierto tiempo y no tendrá la sensación de haber realizado una gran inversión sobre la que no obtiene resultados. Por otro lado podrá comprobar si sus requerimientos se han realizado como él se esperaba. Además las nuevas ideas que le surjan, o las modificaciones sobre requerimientos que crea convenientes, ocasionarán menor pérdida de dedicación temporal por parte del equipo ya que sólo habrá que corregir parte de una iteración o incluso se pedirán modificaciones sobre partes que aún no se han implementado, lo que no ocasionaría pérdida de tiempo.

Aunque las mejoras que ofrecen estos métodos son sustanciales y se teoriza sobre ellos desde principios-mediados de los '90, las empresas y los equipos de desarrollo siguen usando métodos clásicos para el desarrollo de sus productos. Actualmente en la sede de Barberá del Vallés de UNIT4 sólo hay un equipo que use esta metodología, el encargado del producto *Ready*, pero ha tenido tan buena aceptación por parte del equipo, como de los directivos y el cliente que se planea ampliar su uso a todos los equipos posibles.

La metodología Scrum está caracterizada por la organización de las tareas a desarrollar, el seguimiento del trabajo mediante reuniones y los roles que asumen los diferentes componentes. Aunque el método tiene

muchas características es posible amoldarlo según más guste al equipo mientras se conserve la esencia. Sobre la metodología se hará una explicación extensa en el capítulo 2, pero se ha considerado conveniente la introducción de algunos conceptos clave antes de presentar el proyecto:

ProductOwner es el encargado del producto dentro de la empresa. Será quien se encargue de hablar con el cliente y detallar el *Product Backlog*.

ScrumMaster es el encargado de velar por el correcto funcionamiento del equipo de desarrollo y de controlar que el equipo no sea objetivo de intervenciones externas.

Team son los encargados de desarrollar el software. Normalmente será un equipo pequeño, de 4 a 7 personas, donde se pretende que haya variedad de especializaciones entre sus componentes para cubrir el máximo de terreno posible.

Stakeholder son los clientes y los usuarios finales del producto. Ellos serán el conjunto interesado en la producción del software.

Product Backlog es el documento que crea el ProductOwner junto a los Stakeholder y donde se especifican los diferentes requerimientos del software a desarrollar.

Sprint Backlog es un subconjunto del Product Backlog, uno por *Sprint* (iteración), donde se especifica lo que se va a desarrollar durante el transcurso de éste.

Para el desarrollo de este proyecto se hará uso de la metodología Scrum junto a otros tres becarios de la UAB para crear una aplicación que facilite la gestión e implementación de dicha metodología. La composición del equipo estará formada por becarios y trabajadores del departamento de fábrica de UNIT4 y será la siguiente:

ProductOwner - Ramón Torres Arús

ScrumMaster - Susana Ramos García

Team

Héctor Mudarra Teruel. Tutor: Germán Álvarez Corral

Luis Alberto González Samblas. Tutor: Susana Ramos García

Pedro Antonio Siles Jiménez. Tutor: Francesc Martínez Morlanes

David Clemente Romero. Tutor: Carlos Muñoz Robles

Stakeholder - Carolina Ruiz Medina, Piero Marsilio.

1.2. Objetivos.

La metodología Scrum tiene sus ventajas como se ha indicado anteriormente, pero también tiene sus inconvenientes. El más grande de ellos es la creación y la gestión de todo el material de seguimiento de la metodología. Como las iteraciones son relativamente cortas, si se comparan con una metodología clásica, se aprovecha para incluir más detalle para los diferentes aspectos de ésta, ya sean gráficas de seguimiento, seguimiento de tareas o paneles de asignación. De esta forma se puede realizar un seguimiento de la evolución del equipo y poder ofrecer mejoras para las siguientes iteraciones.

El proyecto a realizar pretende desarrollar un conjunto de utilidades para centralizar y gestionar todos estos datos. Es por ello que se han hecho cuatro subdivisiones, una para cada proyectista:

Product Backlog para controlar el Product Backlog y los productos, proyectos, *release*.

Sprint Backlog para controlar el Sprint Backlog, así como la posibilidad de imprimir las tarjetas o visualizar los paneles.

Incidencias para controlar incidencias remotas e internas y poder hacer su traspaso a historias.

Team & Role para controlar los equipos, los colaboradores, los roles, los permisos, las reuniones y disponer de un historial detallado.

El proyecto del que se es responsable es el encargado de Team & Role. Al tratarse de una parte necesaria pero a su vez distanciada de la metodología se pretende hacer el proyecto lo más modular posible para poderlo reutilizar en otras metodologías.

1.3. Situación actual.

Actualmente existen varias aplicaciones libres o de pago que permiten la gestión y el seguimiento de una metodología Scrum (*Agile 42, Icescrum, Scrumworks*). Pese a que estas aplicaciones son útiles y cumplen con su función al gestionar el método Scrum, no acaban de convencer a los diferentes equipos de la empresa ya que no permite la integración con sus propias aplicaciones de gestión. Es por ello que en la empresa prefieren que el ScrumMaster use hojas de cálculo de *Microsoft Office* u *OpenOffice* para llevar la gestión, ya que les resultan más cómodas y no deben de instalarse software adicional para consultarlo.

Debido a ello se ha propuesto este proyecto que pretende substituir las actuales hojas de cálculo por una aplicación hecha con las herramientas de desarrollo de la propia empresa y, a su vez, pueda interactuar con sus ya existentes aplicaciones de gestión. De esta forma no será necesario instalarse software adicional por parte del equipo o el resto de colaboradores que deseen consultar lo que se está haciendo. Además, de esta forma, toda la documentación de seguimiento quedará centralizada en una misma aplicación con funcionalidades adicionales (como la visión de paneles, el control de reuniones, la gestión de incidencias) y podrá ser consultada a forma de historial cuando se desee.

1.4. Contenido de la memoria.

El resto de la memoria está compuesta por los siguientes apartados:

Metodología del proyecto Se hará una explicación de la metodología Scrum y se añadirán comentarios referentes a cómo se ha aplicado durante el desarrollo del proyecto.

Análisis Se analizan los diferentes aspectos que debe cumplir la solución y se hace un análisis de viabilidad sobre el proyecto. Posteriormente se hace una planificación temporal global del proyecto en su conjunto y un primer reparto de tareas dentro de las diferentes iteraciones.

Diseño Se realiza un diseño para la aplicación que cumpla los diferentes requerimientos analizados.

Implementación Se analizan y comentan los diferentes Sprint de forma que se vea la evolución de la aplicación. Se incluye también una explicación sobre cómo se han cumplido los requerimientos y qué pruebas se han realizado.

Conclusiones Para concluir el proyecto se hace una reflexión y se analiza el cumplimiento de los objetivos propuestos y la planificación temporal. También se exponen las líneas de continuación y posibles mejoras.

Glosario El primero de los anexos se compone de un pequeño glosario de breves descripciones que contiene los diferentes términos de la metodología Scrum, así como términos que hacen referencia al software de la empresa UNIT4.

Capítulo 2. Metodología del proyecto: Scrum.

La metodología de un proyecto de final de carrera acostumbra a ser una metodología clásica o de cascada, en este caso se ha usado otra, Scrum. Al tratarse de una metodología no común en este tipo de proyectos se hará una explicación detallada. Para ello se hará una breve introducción histórica para conocer sus orígenes. Acto seguido se explicará el ciclo de vida Scrum. Como para explicar el ciclo de vida se hará uso de vocabulario propio de la metodología, prosiguiendo a éste se definirán los roles y los materiales de soporte de una forma más detallada. Finalmente se hará inciso en las diferentes reuniones que propone la metodología.

Al tratarse de un método de organización y de trabajo hay partes que se dejan al libre albedrío para que el equipo las adapte según le resulten más convenientes. Es por ello que a medida que se expliquen las diferentes características de la metodología se añadirán comentarios sobre qué decisiones se han tomado al aplicarla en nuestro equipo de trabajo.

2.1. Introducción histórica.

Pese a ser un método que está empezando a usarse en equipos de desarrollo de software a nivel global desde hace pocos años, su nacimiento teórico se remonta muchos años atrás. Fue en 1995 cuando Ken Schwaber con la ayuda de Jeff Sutherland publicó un artículo describiendo la metodología Scrum. El nombre lo tomó de *The New New Product Development* Game [1], un estudio del 1986 realizado por Hirotaka Takeuchi e Ikujiro Nonaka y publicado por el *Harvard Business Review.* En este estudio se analizaban diferentes métodos de trabajo en equipo usados en grandes empresas haciendo analogías de las diferentes formas. Entre ellas se comparaba la metodología de superposición de procesos y de trabajo en equipo que usaban en Honda con el trabajo en equipo del rugby y su *scrum* (melé).

Sutherland había estado probando aproximaciones del método en las diferentes empresas en que había estado trabajando a principios de los '90. Fue en 1993 en *Easel Corporation* donde definitivamente nació la metodología. En 1995 introdujo a Schwaber al primer equipo de Scrum y éste decidió expandir la metodología Scrum de producción de software a nivel mundial. Se publicaron algunos escritos sobre el método, pero no fue hasta que Schwaber presentó el artículo *Scrum Development Process* [2] en la OOPSLA 1995 cuando definiría la metodología. Desde entonces tanto Sutherland como Schwaber han sido referentes para Scrum [3] y han seguido colaborando en la publicación de documentos sobre la metodología, además de mantener portales web actualizados donde cualquiera puede acceder a su información [4] [5]. Finalmente, tras la formulación del Agile Manifesto [6] en el 2001, varias metodologías, entre ellas Scrum, se agruparon bajo la categoría de *desarrollo ágil de software*.

2.2. Ciclo de vida Scrum.

Scrum es una metodología de desarrollo de software ágil. Estas metodologías surgen con la intención de mejorar los principales problemas de las metodologías clásicas. Como se ha comentado en el capítulo 1, el principal cambio sobre las metodologías clásicas es el cambio de un proceso escalonado continuo a un método iterativo, donde en cada fase se ejecutan todas las fases de desarrollo del proyecto. Es por ello que Scrum parte de la misma base que las metodologías clásicas, la búsqueda de un proyecto y la construcción del documento de requerimientos.

En primera instancia el ProductOwner se encarga de crear un proyecto, encontrar clientes y redactar una primera versión del Product Backlog donde se recojan los requerimientos iniciales. Una vez se disponga de un producto que desarrollar se busca un equipo que sea capaz de satisfacer con éxito los requerimientos estipulados en el tiempo que se haya acordado o previsto. De esta forma el equipo tiene asignados ProductOwner, Stakeholders, ScrumMaster, Team y una primera versión incompleta del Product Backlog.

Una vez se dispone de todos estos actores se inicia Scrum. Para ello se reunen ProductOwner, ScrumMaster y Team con el fin de convertir la versión incompleta del Product Backlog, donde sólo se especifican requerimientos, en una versión completa con valoración de esfuerzo, prioridad y descripciones detalladas de las diferentes *historias*.

Cuando ya se dispone de un Product Backlog completo, aunque siempre permanece abierto a nuevas propuestas, se procede a hacer una selección del trabajo a realizar durante el siguiente Sprint. Esta selección corre a cargo del Team y está supervisada por el ProductOwner, que asegurará que aquello que es más prioritario sea lo que primero se realice. El resultado de la selección es el Sprint Backlog y determina qué se hará durante el Sprint.

Durante el Sprint, el Team se encarga de transformar el Sprint Backlog en un producto funcional y listo para entregar, pasando las fases de análisis, diseño, desarrollo, test y pruebas para todas las historias que haya que implementar. Con tal de llevar un seguimiento del Sprint se hace una reunión diaria donde cada miembro del Team expone lo que ha hecho y lo que pretende hacer hasta la siguiente reunión.

Como el objetivo del Team es el de entregar un producto funcional, todo el equipo se encarga de hacer lo que esté en sus manos para terminar lo que se han propuesto. Es por ello que las diferentes tareas a realizar no tienen a una persona asociada para su desarrollo, sino que cada miembro toma aquellas tareas que considera oportunas en cada momento. Para ello se hace uso de paneles de seguimiento, donde se tiene una fácil visión gráfica del estado del Sprint en su totalidad y sus partes.

Todo aquello que se presente se considera terminado, por tanto, si alguna de las historias previstas no se termina cuando lo hace el Sprint, ésta no se presenta y se considera un fracaso a nivel de planificación. Una vez terminado el Sprint se presentan los resultados a los Stakeholder y se toma *feedback* de opiniones. Posteriormente se hace una reunión para planificar el siguiente Sprint, repitiendo todo el proceso hasta que el Product Backlog se termine o bien el proyecto se dé por terminado.

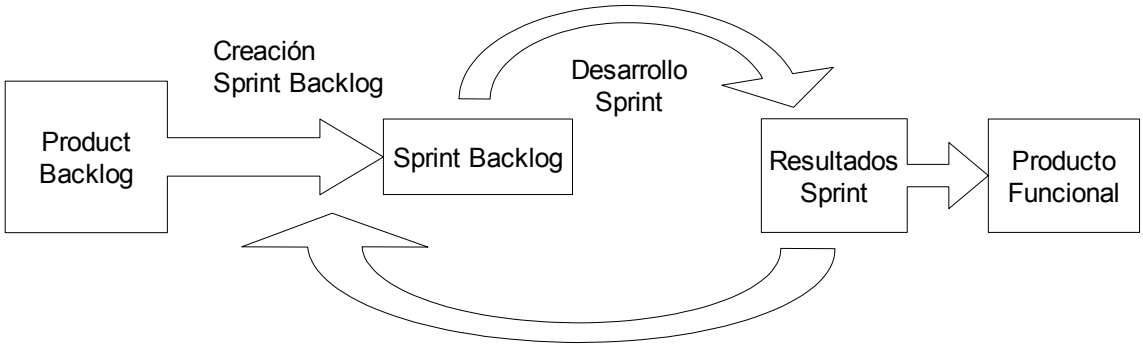

Figura 2. Ciclo de vida Scrum.

2.3. Roles.

Con tal de describir y determinar cuál es el papel de cada participante de Scrum se definen un conjunto de roles. En primera instancia se hacen dos grandes divisiones de roles, los roles “cerdo” y los roles “gallina”. Los roles “cerdo” son los implicados con un cometido específico durante el desarrollo del proyecto. Los roles “gallina”, por el contrario, no tienen un cometido específico durante el desarrollo ni están directamente implicados, pero es a ellos a quien está dirigido el software a producir y por tanto hay que tenerles en cuenta.

Hecha esta primera división se hablará en detalle sobre los diferentes roles “cerdo”, ya que son los que intervienen en el desarrollo:

ProductOwner

Es el encargado de crear y mantener el Product Backlog y de asegurarse de que, pese a la libertad de decisión y gestión del Team, se realicen con anterioridad aquellas historias con una prioridad mayor. Es también el encargado de hablar y mantener contacto con los Stakeholders (roles “gallina”) y por consiguiente será el representante de la voz del cliente dentro de la empresa.

Es el encargado del proyecto, quien determina el qué, el cuándo y la rentabilidad. Su principal objetivo será el de analizar el proyecto desde un punto de vista de negocio ya que, como responsable del Product Backlog, es el responsable del éxito del producto.

Quien realice este rol puede realizar también un rol del tipo Team, pero nunca podrá ser ScrumMaster a la vez que ProductOwner.

ScrumMaster

Al contrario de lo que su nombre pueda suscitar la función de este rol no es la de tomar decisiones y gobernar al Team, sino que se trata de un rol que vela por la integridad de la metodología. Su principal cometido es que Scrum se lleve a la práctica como debe, que el clima del equipo sea el adecuado y que no haya intervenciones externas de cara al Team.

Será entonces el encargado de adaptar la metodología a las condiciones del equipo actual y de controlar que todo funcione como es debido, ayudando siempre que sea necesario. Con tal de mantener el clima de trabajo dentro del Team, el ScrumMaster será el portavoz de éste, atendiendo todo aquello que venga de fuera y transmitiendo todo aquello que venga de dentro.

En los equipos donde el ScrumMaster no es miembro del Team se le delegan a éste las funciones de actualización del Sprint Backlog así como la creación de documentos para las reuniones Sprint Review y Sprint Retrospective. A parte, se le considera el portavoz del Team para aquellas reuniones en que éste tenga que participar.

Quien realice este rol puede realizar también un rol del equipo Team, pero nunca podrá ser ProductOwner a la vez que ScrumMaster.

Team

Son los encargados de desarrollar las funcionalidades del software. La ventaja que ofrece Scrum sobre a otras metodologías es que se le da libertad de gestión al Team, de forma que son ellos quienes crean el Sprint Backlog eligiendo qué hacer durante el siguiente Sprint. Sin embargo su creación se hace con la supervisión del ProductOwner, quien respetará la libertad de decisión pero asegurará que se respeten las prioridades principales del proyecto. Al ser los encargados de la creación del Sprint Backlog también son los responsables de que se termine con éxito y todo lo planificado se pueda presentar al final de éste.

El conjunto de roles Team está formado por 7±2 personas conformando un equipo multidisciplinario, ya que en cada iteración se pasa por todas las fases de desarrollo de un proyecto. Es posible que cada miembro del equipo tenga una especialización en concreto, pero no por ello debe de negarse a realizar otro tipo de tareas que no correspondan a las suyas. En Scrum lo importante es el resultado del equipo, no el resultado individual. La composición del Team será la misma mientras dure el Sprint y no se harán nuevas modificaciones hasta que empiece uno de nuevo.

Para lograr un equipo multidisciplinario, en UNIT4 definen la estructura del Team como un conjunto de perfiles que componen un equipo funcional. A continuación se muestra un ejemplo de composición de un equipo de 7 miembros:

Business Analist (x1) Perfil de Jefe de proyecto o Analista senior con conocimientos técnicos del proyecto en su totalidad.

Desarrolladores (x4) Perfil de programador o analista programador. Elementos productivos.

Quality Consultant (x1) Encargado de planificar y ejecutar los planes de test.

Technical Author (x1) Técnico que redacta y da formato a la documentación completa.

Esta variedad de roles es suficiente para crear un equipo de trabajo capaz de utilizar correctamente la metodología Scrum. Uno de los puntos más interesantes de los roles es el factor comunicación que se establece. El hecho de aislar al equipo hace que pueda concentrarse en la labor de producción sin la necesidad de lidiar con las perturbaciones exteriores del cliente.

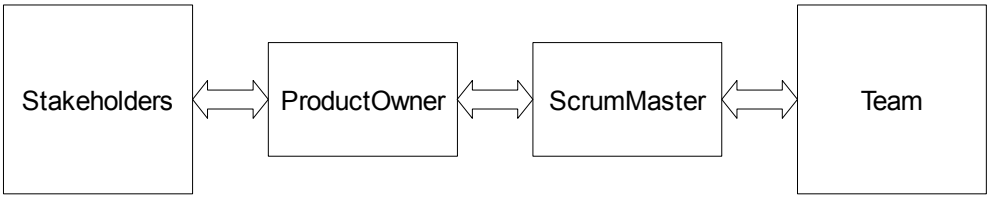

Figura 3. Comunicación entre los roles Scrum.

2.4. Materiales de soporte usados.

Para organizarse y llevar un seguimiento del producto a desarrollar durante el proyecto, Scrum nombra un conjunto de materiales de soporte que realizan dicho cometido. Como podrá observarse todos están relacionados entre ellos, a excepción de los paneles. Ese es uno de los motivos por el que los materiales usados actualmente para el almacenamiento sean hojas de cálculo de las suites *Microsoft Office* u *OpenOffice*, ya que las funcionalidades que ofrecen éstas concuerdan con las necesidades de los materiales de soporte.

2.4.1. Product Backlog.

Todos los documentos de Scrum son importantes y vitales para su correcto uso, pero si hay uno que destaque por encima del resto ese es el Product Backlog. Este documento está compuesto de una multitud de elementos llamados historias. Cada historia hace referencia a un requerimiento que se le exige al proyecto y está compuesta de varias características:

Identificador de historia Resulta más sencillo identificarlas con algún código alfanumérico antes que por su nombre.

Nombre y/o descripción Donde se detalla el qué requiere la historia, el por qué es conveniente su implementación y quién ha pedido la historia, cuál es su origen.

Valoración de prioridad No todos los requerimientos del proyecto tendrán la misma prioridad. Es necesario clasificar las historias según la prioridad para saber qué hay que desarrollar antes para, en caso de que no dé tiempo, terminar aquellas más importantes una vez finalice Scrum.

Valoración de puntos Cuando se están definiendo historias se está teorizando a un nivel de abstracción alto donde no se llega a detallar cómo desarrollarlas. Es por ello que no pueden ser cuantificadas con un tiempo estimado en horas, sino que esta valoración se hace mediante puntos. Los puntos dan una orientación sobre la cuantía de esfuerzo necesario para realizar una historia, pudiendo modificar la conversión punto/hora cuando se crea necesario y logrando mantener la escala de esfuerzo proporcional. Para fijar una referencia se asigna un valor de horas determinado a una cantidad de puntos. Por ejemplo en este proyecto, por recomendación del ScrumMaster de *Ready*, hemos usado una referencia de 1 punto por 8 horas, es decir, una jornada completa.

4				Valorados	Prioridad	I want to	So that
5	Área	Story	Título	169			
37	Incidencias	32	Clasificación	2	1	Proceso de clasificación de bugs, previsión de resolución e integración en los sprints	Para poder planificar la resolución. Clasificar también los bugs por nivel de importancia (must, should,
38	Incidencias	33	Informes abiertos	2	2	Informes estadísticos de los bugs pendientes y de los históricos	
39	Team&Role	34	Definidor roles	2	1	Definidora de perfiles	Para disponer de una especificación general.
40	Team&Role	35	Definidor colaboradores Scrum	2	1	Definidora de colaboradores para Scrum	Adaptar los colaboradores existentes a un nuevo modelo con
41	Team&Role	36	Definidora plantilla equipos	2		Definidora de plantillas equipos	Para disponer de plantillas de
42	Team&Role	37	Definidor equipos	3	1	Definidora de equipos	Equipos generales basados en
43	Team&Role	38	Vacantes	1	3	Gestor de peticiones y perfiles	Para gestionar las necesidades que

Figura 4. Product Backlog.

El Product Backlog es únicamente modificado por el ProductOwner, aunque sus contenidos son públicos para el Team y el ScrumMaster. Será creado antes de iniciarse el proceso de Scrum, cuando se estén detallando por primera vez las condiciones y las fronteras del proyecto a desarrollar. Sin embargo siempre está abierto a cambios ya sea para modificar historias, borrarlas o bien añadir de nuevas. Cualquiera de estas acciones sólo podrá ser ejecutada por el ProductOwner.

2.4.2. Sprint Backlog.

El Sprint Backlog es un documento que nace a partir del Product Backlog. Este documento contiene el conjunto de historias que se prevé que se terminarán durante el Sprint. Su principal objetivo es el de servir como diario de progreso donde se puede consultar en todo momento cuál es el estado actual del Sprint. Como Scrum potencia el trabajo en equipo y el conocimiento global de la aplicación por parte de sus miembros, las historias, que son grandes requerimientos que pueden llevar muchas horas de trabajo, son segmentadas en tareas que tienen una duración máxima de 16 horas, es decir, dos jornadas.

Así pues, el Sprint Backlog estará compuesto por las diferentes historias a realizar, que a su vez estarán segmentadas en tareas. De cada una de estas tareas se almacenará:

Identificador de la tarea Al segmentar una historia en varias tareas, cada una quedará identificada con el patrón Historia / Tarea. De esta forma podrá identificarse en qué punto de la historia se encuentra.

Descripción de la tarea Cada tarea representa una pequeña parte de los objetivos de la historia. Es por ello que se especifica en cada una qué es lo que se debe hacer.

Horas restantes Cuando se habla de tareas ya se hace un análisis más especifico de su desarrollo. Es por ello que se puede hacer una estimación del tiempo que llevará hacerla. Además este tiempo se actualiza cada día para tener constancia del tiempo restante y poder dibujar las gráficas de seguimiento.

Horas previstas Aunque la metodología Scrum no considere las horas previstas como un dato importante de las tareas, en UNIT4 se toma la costumbre de anotar el tiempo dedicado en cada tarea. De esta forma es posible analizar la evolución de cada una y ver si su valoración inicial fue suficientemente precisa.

Técnico responsable Pese a promover una filosofía de resultado global y de trabajo en equipo, a veces es necesario conocer quién ha desarrollado una tarea en caso de que surja alguna necesidad relacionada con ella.

Scrum		Release 1.0 Sprint 6						
Story		Description		06/05/2010	07/05/2010	08/05/2010	09/05/2010	10/05/2010
	Tsk	Esfuerzo pendiente	134	126	114	114	114	93
40		Consulta de proyectos						
Diseño	1	BQ/Q/BO	8	6	5	5	5	2
Diseño	2	Formularios de los distintos módulos	4	4	3	3	3	1
Desarrollo	3	Añadir hipervínculos	4	4	4	4	4	4
Desarrollo	4	Incluir permisos y funcionalidades de botones	4	4	4	4	4	4
Test	5		2	2	2	2	2	2
Documentación	6		8	8	8	8	8	8
14		Convocatoria de reuniones desde sprint						
Diseño	1	Tabla: Plantilla, lugar y duración.	2					
Diseño	2	BQ: Plantilla, lugar y duración.	4	2				
Diseño	3	Formulario : Plantilla, lugar y duración.	4	4	2	2	2	2
Desarrollo	4	Adaptar formulario sprint	12	12	12	12	12	8
Test	5		4	4	4	4	4	4
Documentación	6		4	4	4	4	4	4

Figura 5. Sprint Backlog.

El Sprint Backlog es únicamente modificado por el Team. No tendrá ningún cambio a nivel de historias asignadas durante la duración de éste, ya que el equipo de desarrollo se compromete a terminar aquello que ha planificado. Es por ello que el Team crea el Sprint Backlog al inicio de cada Sprint y es el único con derecho a modificar, añadir o quitar tareas durante la duración de éste (por ejemplo añadir tareas para corregir bugs surgidos durante el desarrollo).

2.4.3. Gráficas Burndown.

Aunque se disponga de toda la información organizada y almacenada en Backlogs, una de las mejores formas de representar el progreso del proyecto es mediante el uso de gráficas. Para ello la metodología Scrum propone el uso de dos gráficas Burndown; la del Product Backlog y la del Sprint Backlog.

El objetivo de estas gráficas es representar el progreso actual del equipo frente a un progreso teórico que les llevaría al éxito. De esta forma el equipo podrá saber en todo momento si se encuentra por encima o por debajo de lo previsto y puede tomar decisiones sobre qué desarrollar en cada momento.

La gráfica Burndown del Product Backlog proporciona una visión global del proyecto. Para ello se muestran dos funciones superpuestas. Una de ellas representa el progreso teórico que deberían seguir las historias para estar todas terminadas una vez se termine el plazo de Scrum. La otra representa el progreso real del proyecto. Así, de una forma fácil y visual, el ProductOwner sabe cual es el progreso global del proyecto y puede decidir si añadir o quitar funcionalidades para garantizar su éxito.

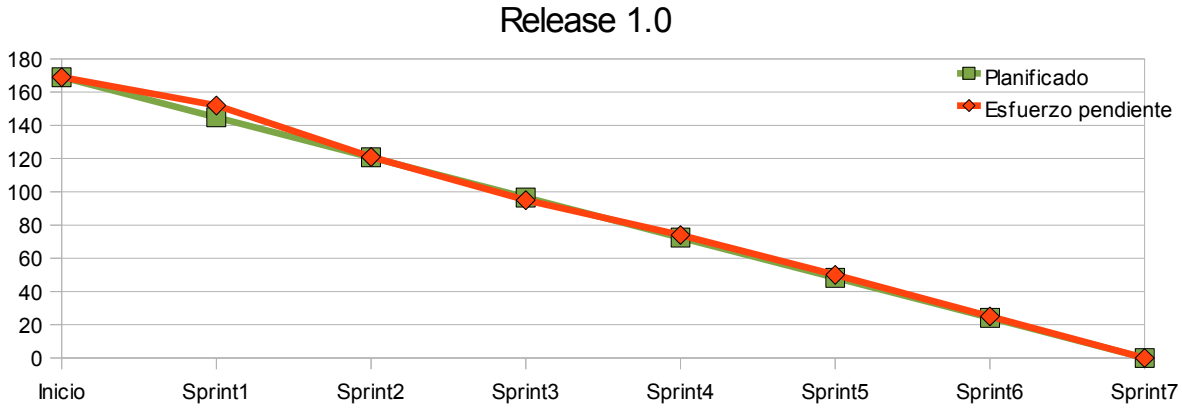

Figura 6. Burndown Product Backlog.

De forma análoga la gráfica Burndown del Sprint Backlog proporciona una visión específica del progreso del Sprint. Para ello también se muestran dos funciones superpuestas. Una de ellas representa el progreso teórico que deberían seguir las tareas para quedar finalizadas al terminar el Sprint. La otra representa el progreso real del Sprint. De esta forma el Team es capaz de ver si el Sprint actual terminará como se había planeado o bien va camino al fracaso. Además podrán añadir tareas de sugerencias o arreglo bugs, sobre historias de Sprint pasados, si el tiempo les es favorable.

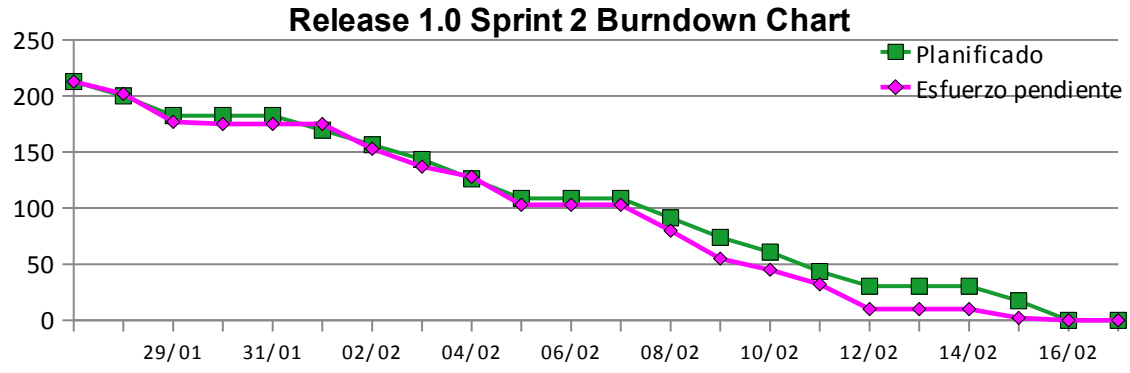

Figura 7. Burndown Sprint Backlog.

Para representar el progreso teórico de las tareas en un Sprint se hace uso de una fórmula que, para cada día, tiene en cuenta aquellos miembros presentes del Team y así adapta el esfuerzo teórico a los recursos disponibles:

$$f(x_i) = x_{i-1} - \frac{Miembros_i}{\sum_{n=inicio}^{fin} Miembros_n} \cdot x_0$$

Figura 8. Pendiente teórico del Sprint Burndown.

donde x_i es el i-ésimo día del Sprint.

2.4.4. Panel de seguimiento (Task Board).

Los paneles de seguimiento no están incluidos dentro de los materiales de Scrum que se nombran en la metodología. Sin embargo, se trata de un recurso que utilizan la gran mayoría de equipos que hacen uso del método y son de gran utilidad para el seguimiento del Sprint.

Como su nombre indica se trata de paneles para el seguimiento de las tareas. Pese a haber libertad absoluta para crear el panel y organizarlo como más convenga al equipo, la gran mayoría mantienen algunas características en común:

Tarjetas Las tarjetas representan una tarea y la información que contienen depende del equipo. Algunos elementos que siempre se muestran son el técnico encargado de la tarea y el tiempo restante de ésta. Las tarjetas deben ser elementos móviles de forma que se puedan poner, quitar y cambiar de sitio.

Estados Se separa el panel en diferentes zonas cada una de ella representando un estado de las tarjetas. De esta forma cuando una tarea está en curso, su tarjeta se sitúa en la zona de *en curso*. Los estados más habituales son: *pendiente*, *en curso* y *terminado*.

Carriles de historia A parte de la división por estados, se acostumbra a hacer una división de carriles de historia. Estos carriles tendrán una representación en cada uno de los diferentes estados, permitiendo mover las tareas pertenecientes a una historia según su estado. Así se obtiene una visión rápida del estado de las historias.

Visibilidad Una de las funciones del panel es la de poder conocer el estado del Sprint en cualquier momento y de forma visual. Su disposición será entonces un lugar cercano al de trabajo, donde el equipo pueda acceder a él sin tener que desplazarse excesivamente.

Gráfica Burndown Aprovechando que se trata de un lugar al que el equipo hace referencia constante, se sitúa la gráfica de Burndown cerca o dentro del mismo panel.

Cuando un miembro del equipo decide empezar a trabajar con una tarea, se le ocurre una tarea nueva, termina con la que estaba haciendo... sea cual sea el motivo, el primer lugar al que se acudirá para hacer una modificación sobre el Sprint será el panel de seguimiento. Por tanto se trata de un elemento que está actualizado al instante y donde el seguimiento es en tiempo real. Ello provoca un efecto motivador al equipo ya que, como las tareas son de por sí cortas de realizar, ve que hay movimiento, el proyecto fluctúa y el Sprint Backlog se va completando.

Para facilitar el seguimiento y las actualizaciones del Sprint Backlog, éste se actualiza al terminar o empezar la jornada laboral tomando el panel como referencia de cuál es el estado actual del Sprint.

2.4.4.1. Variante usada.

Como se ha mencionado anteriormente hay multitud de formas de construir los paneles de seguimiento y cada equipo toma aquella que le parece mejor. En este apartado se mostrará la variante que se ha usado en el equipo del proyecto. Es importante especificar que, como casi todo en Scrum, el panel está abierto a cambios y su formato no es cerrado.

Tarjetas En un inicio se usó un formato de tarjeta en blanco donde se indicaba de qué tipo de tarea se trataba (investigación, análisis, diseño, desarrollo, test, documentación), a qué historia pertenecía, cuál era el técnico asignado, las horas previstas y reales y una breve descripción de la tarea.

Este formato duró sólo dos Sprint, ya que a simple vista no se sabía qué era lo que se estaba haciendo en cada momento. De cara a los siguientes Sprint se optó por mantener el formato de la tarjeta pero imprimirla en papeles de color diferente dependiendo del tipo de tarea. De esta forma a simple vista se podía ver en qué fase se encontraba la historia.

Además, aprovechando la utilidad que se estaba desarrollando, la impresión de tarjetas se hacía a través del programa donde se rellenaban automáticamente la mayoría de campos de ésta y sólo hacía falta actualizar las horas.

Estados Desde un inicio se tomaron dos paneles divididos por la mitad. En el primero se situaban aquellas tareas *pendientes* y *en curso* mientras que en el segundo panel se situaban aquellas tareas *terminadas*.

Carriles de historia En un principio no se consideró el hecho de ordenar las tareas por historias, sino que todas las tareas quedaban dispuestas por el panel de seguimiento y sólo se les cambiaba el estado. En el segundo Sprint se procedió a juntar las tareas de una misma historia en un carril horizontal. De esta forma se sabía el estado de las historias y cuáles quedaban por terminar.

Visibilidad Durante la primera semana los paneles se situaron en un muro móvil que tenía unos clavos donde sujetarlos. Sin embargo en ese lugar era difícil de acceder, ya que quedaba por detrás de un lugar de trabajo y cualquier acceso al panel significaba una distracción. Posteriormente se cambiaron a un lugar más accesible, aguantados por una estantería y donde no se producían distracciones al acceder a ellos.

Gráfica Burndown Desde un inicio la cuarta división de los paneles se usó para poner la gráfica de Burndown del Sprint. Para este proyecto no se representó físicamente la gráfica Burndown del Product Backlog.

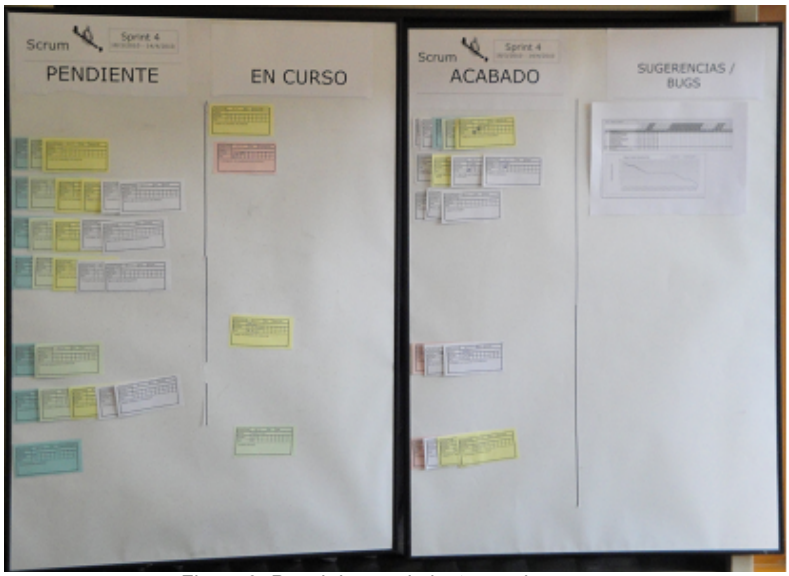

Figura 9. Panel de seguimiento usado.

2.5. Reuniones.

Una de las partes fundamentales de Scrum son las diferentes reuniones que establece. En estas reuniones es donde se mantiene informado al resto del equipo de la evolución del proyecto y donde se generan los Backlogs.

2.5.1. Release planning + *sizing*.

La metodología Scrum nombra el Release planning como aquella reunión donde se determinan los objetivos y las metas del proyecto y además se analizan de forma que su realización resulte rentable. Además se menciona que esta reunión no es obligatoria y por tanto el equipo puede empezar Scrum sin hacerla, aunque luego se notará en la falta de análisis de temas básicos.

En la práctica esta primera reunión se realiza una vez el ProductOwner dispone de un conjunto de requerimientos funcionales y no funcionales sobre el proyecto. En ese momento convoca al Team y al ScrumMaster y se empieza a analizar el proyecto en su totalidad. De esta forma se construye poco a poco el Product Backlog definiendo las diferentes historias que lo componen, donde el Team se encargará de analizar su viabilidad técnica y el ProductOwner se encargará de analizar que el resultado que se plantea sea económicamente rentable. Por tanto será el ProductOwner quien decidirá si las historias son o no incluidas en el Product Backlog y cuál será su prioridad mientras que el Team será el encargado de puntuar estas historias.

Es posible que el ProductOwner convoque la reunión disponiendo de un Product Backlog al que sólo le falta comprobar la viabilidad de algunas historias. Por tanto, como la gran mayoría de la reunión será utilizada para puntuar las diferentes historias, esta reunión es también denominada *sizing*.

Para puntuar las historias el Team lee cuáles son los objetivos y el por qué de éstas y en caso de tener cualquier duda, o necesitar una aclaración, consulta al ProductOwner. Cuando ha quedado clara y debatida la historia el Team procede a valorarla. Ya que para puntuar una historia hay que formar un consenso entre los diferentes miembros y a base de debatirlo se puede tardar mucho, se utiliza algún método para agilizar la puntuación. En este proyecto se usó el método del *Planning Poker:*

Planning Poker

Se dispone de un conjunto de 13 cartas por cada miembro del Team. Las cartas son todas iguales por detrás, mientras que por delante tienen los siguientes valores: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100, ?, taza de café.

Las cartas con un valor numérico representan ese valor en puntos, para este proyecto se tomó 8 como el valor máximo a cuantificar una historia debido a la longitud de los Sprint y la jornada. La carta ? determina que la historia no es cuantificable,o bien no se entiende o está mal definida. La carta de la taza de café indica que se necesita un descanso, es una forma de avisar ya que estas reuniones acostumbran a ser bastante largas.

La forma de puntuar es sencilla y dinámica. Una vez se ha hecho una breve exposición de la historia y se han aclarado las dudas, cada miembro del Team elige una carta con la cuantía de puntos que crea adecuada para la historia. Acto seguido se pone la carta boca abajo y se espera a que todos hayan elegido. Luego se destapan las cartas y se analiza el resultado:

Resultados muy distantes entre sí El que haya puntuado menos y el que haya puntuado más expondrán los motivos que les han llevado a elegir la puntuación. Acto seguido todo el Team vuelve a elegir carta repitiendo de nuevo la votación hasta que haya consenso.

Resultados poco distantes entre sí Se hace un promedio y la historia recibe el valor resultante.

Mismo resultado Raras veces se llegará a un consenso, en caso de que se dé ese es el valor en puntos de la historia.

Si surge la carta ? en una votación, aquel que la haya elegido expondrá sus motivos. En caso de llegar a un consenso con el resto del Team se determinará como historia mal declarada y se volverá a plantear a quien la propuso.

En cualquier momento se podrá solicitar un descanso poniendo la carta del café boca arriba. Cuando la mitad de miembros del Team coincida se tomará un breve descanso antes de proseguir con la votación.

Debido a la magnitud de nuestro proyecto, sólo se ha hecho uso de las cartas 0, ½, 1, 2, 3, 5, 8, 13, ?, café. Se restringió la puntuación máxima a 8 puntos por historia, debido a la duración de los Sprint y a que, en principio, una historia sólo era hecha por un proyectista. Por tanto, si la puntuación final era 13, se consideraba que la historia debía dividirse, simplificarse o bien descartarse.

Figura 10. Variante del Planning Poker usada.

Después de haber hecho toda la valoración y haber analizado el Product Backlog por parte de todo el equipo, la reunión se dará por concluida y el Product Backlog se encontrará en su primera versión y el equipo estará listo para empezar los Sprint.

2.5.2. Sprint planning.

Esta reunión tiene cabida al inicio de cada Sprint y está planificada para una duración máxima de 8 horas, en caso de disponer de Sprint de un mes de duración. El objetivo de esta reunión es obtener el Sprint Backlog para el Sprint que se va a realizar. Para poder crearlo hay que saber qué cantidad de puntos está dispuesto a realizar el Team. Para ello se introduce una variable llamada velocidad, donde el Team determinará cuántos puntos es capaz de hacer en un Sprint, es decir, cuál es su velocidad en puntos.

Velocidad del primer Sprint Aunque los puntos sean una unidad sin valor temporal, se le asigna una equivalencia inicial para orientarse al puntuar las historias. Tomando esta equivalencia inicial se calcula cuántos puntos se podrían realizar durante el primer Sprint, teniendo en cuenta todos los miembros de éste y la longitud. El resultado obtenido serán los puntos del primer Sprint. Dicha velocidad seguramente sea demasiado alta y provoque un Sprint fracaso, es por ello que se recomienda trabajar a una velocidad algo inferior al principio del proyecto.

Velocidad del resto de Sprint Como es muy probable que la velocidad calculada en el primer Sprint no se adapte perfectamente al tiempo disponible, será conveniente cambiar la velocidad de cara al siguiente Sprint. Si ha dado tiempo a terminar todas las historias y ha sobrado tiempo se aumentará la velocidad, en caso contrario se disminuirá. Cuantos más Sprint se realicen más datos de velocidad se obtendrán y la nueva velocidad se calculará tomando una media de las anteriores. De esta forma, pasados tres o cuatro Sprint, se habrá establecido un valor de velocidad bastante estable que determinará la velocidad del equipo.

Una vez conocida la velocidad, el Team toma el Product Backlog y elige aquellas historias que se compromete a terminar. El ProductOwner hará de supervisor durante la creación del Sprint Backlog para asegurarse de que aquellas historias elegidas sean las más prioritarias y las que garanticen un desarrollo progresivo del proyecto. Esta selección de historias se efectúa durante la primera mitad de la reunión.

Durante la segunda mitad de la reunión el Team, junto al ScrumMaster, analiza cómo se hará el desarrollo de las historias y procede a fraccionarlas en tareas, asignando una duración prevista en horas. Una vez hayan sido terminadas todas las divisiones se dispondrá de la primera versión del Sprint Backlog.

El Sprint Backlog está abierto a cambios pero nunca de historias, sólo de tareas. Es por ello que, en este proyecto, se acostumbraba a tomar una pequeña reserva de puntos dedicados a Bugs. De esta forma se disponía de un margen de trabajo y, por tanto, la aparición de bugs podía controlarse, asegurando que las historias previstas para el Sprint se finalizarían pese a imprevistos.

2.5.3. Daily Scrum.

Mediante las reuniones anteriores todos los miembros del equipo son conscientes del proyecto en su totalidad y de cómo avanza. Sin embargo, no disponen de una vía de información para realizar un seguimiento día a día del progreso, sino que es cuando ven los resultados cuando se dan cuenta de su progreso. Para ello Scrum propone una pequeña reunión diaria de seguimiento, el Daily Scrum.

El Daily Scrum es una reunión muy rápida de unos 15minutos que, pese a su corta duración, es muy útil. Debe realizarse siempre en el mismo lugar y a la misma hora. En ella sólo tiene voz el Team mientras que el ScrumMaster hace de simple moderador. El ProductOwner está invitado a la reunión pero no puede participar en ella. El objetivo principal de ésta es el de informar al resto del equipo del proceso individual del día a día. Así cada día el equipo es consciente de cuánto ha adelantado. Para esta reunión cada miembro del Team debe de responder a tres preguntas:

¿Qué se ha logrado desde la última reunión?

¿Qué se tiene previsto hacer hasta la siguiente reunión?

¿Qué obstáculos han aparecido en el trabajo?

Una vez que el miembro haya dado respuesta a las tres preguntas se le dará ayuda ha encontrado algún obstáculo que no haya podido sobreponer. Cuando todo el Team haya respondido a las tres preguntas se dará por concluida la reunión.

Esta es una forma de mantener al equipo al día, no obstante no es la única ya que la metodología Scrum favorece el trabajo en equipo. Por ello predica que, durante el desarrollo, es prioritaria la interacción con el resto de miembros y ayudar al bien común del equipo, por delante de focalizarse en el trabajo propio y fracasar como conjunto.

En nuestro equipo, el ScrumMaster no era un miembro del Team, ni tampoco estaba en el mismo lugar de trabajo que el resto. Es por ello que, durante la reunión diaria, también tomaba parte respondiendo a las tres preguntas. Así era posible hacer un seguimiento de qué nuevas peticiones había por parte del ProductOwner, si había algún cambio o actualización sobre el software de desarrollo, o cualquier otro aspecto ajeno al desarrollo pero que afectaba al equipo.

2.5.4. Sprint Review.

Una vez terminada la fase de desarrollo de un Sprint es el momento de presentar los resultados. En esta reunión participan todos los componentes de Scrum, tanto los implicados como los interesados y está prevista una duración máxima de unas 4 horas. El objetivo de esta reunión, a parte de presentar el trabajo realizado, es el de recopilar información, sugerencias e ideas para enriquecer y darle dinamismo al proyecto. De toda esta información recibida el ProductOwner decidirá qué se incluye en el Product Backlog, haciendo así que el proyecto evolucione.

La exposición de los contenidos desarrollados se hace por parte del ProductOwner, que matizará qué es lo que se ha hecho y qué es lo que no se ha hecho durante el Sprint, haciendo referencia a las historias realizadas. Acto seguido el Team presentará la evolución del Sprint y expondrá los principales aspectos de éste, así como los hechos positivos, los problemas y las soluciones tomadas. El Team terminará su intervención haciendo una demostración del producto realizado durante el Sprint.

Finalmente el ProductOwner comentará el estado actual del Product Backlog y mostrará una previsión de los siguientes Sprint para ver la evolución del proyecto. Es entonces cuando se dará por concluida la presentación del Sprint y empezará la ronda de sugerencias e ideas por parte de todos los participantes.

Para el proyecto desarrollado los Stakeholder no podían asistir a las reuniones. Por otro lado, el producto estaba enfocado al uso interno de la empresa y el ProductOwner, a su vez, realizaba el rol de ScrumMaster en el equipo *Ready.* Es por ello que las presentaciones, en lugar de ser mostradas al cliente, eran mostradas al ProductOwner que hacía en ellas el papel de Stakeholder. En este caso, el ScrumMaster exponía los contenidos desarrollados y la evolución del Sprint, y cada miembro del Team se encargaba de hacer la demostración de las partes correspondientes a su módulo.

2.5.5. Sprint Retrospective.

La última reunión que propone la metodología Scrum es el Sprint Retrospective. Esta reunión se realiza a continuación del Sprint Review y tiene prevista una duración máxima de 3 horas. Se trata de una reunión cerrada para el Team y el ScrumMaster, cuyo objetivo es analizar el funcionamiento del equipo y comentar todos aquellos aspectos que se consideren relevantes. Una vez expuestos todos los temas se intentará buscar solución a aquellos que la tengan para facilitar un mejor clima de trabajo.

El ScrumMaster, como observador y moderador de la metodología que es, hará inciso sobre aquello que crea oportuno y comentará posibles soluciones. Es importante que si hay aspectos positivos éstos sean comentados especialmente, motivando y favoreciendo un buen clima de equipo. Cuando el Team exponga comentarios lo hará de forma anónima para evitar posibles malestares. Para ello se escribe aquello que considera oportuno comentar en tarjetas o papeles. Acto seguido se barajan las opiniones y posteriormente se leen y discuten entre todo el equipo.

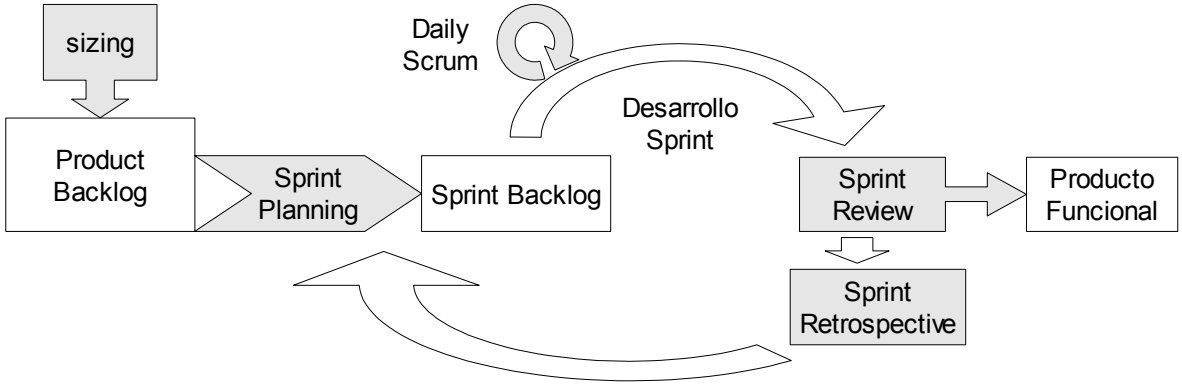

Figura 11. Ciclo de vida Scrum: Reuniones.

Capítulo 3. Análisis.

Para empezar un proyecto es necesario realizar un estudio previo. Dicho estudio permitirá definir los límites de éste una vez se haya comprobado la viabilidad de los distintos requisitos que se le atañen. En en análisis sólo se tratará el proyecto del que se es responsable, Team & Role.

Por contra al usarse un método iterativo es probable que surjan nuevos requisitos, a petición del ProductOwner, una vez el proyecto ya esté en desarrollo. Estos nuevos requisitos serán analizados para comprobar su viabilidad previamente a ser añadidos como tales, aunque dicho análisis se realizará una vez ya se haya iniciado el proyecto.

3.1. Requisitos funcionales.

A continuación se enumeran las diferentes funcionalidades que deberá tener el proyecto.

Creación y gestión de roles Especificar los distintos roles y los permisos de lectura y escritura otorgados a éstos.

Creación y gestión de colaboradores Dar de alta colaboradores para que puedan formar parte de equipos realizando un rol concreto.

Creación y gestión de aptitudes Designar diferentes grados de aptitud que un colaborador puede tener con un rol específico. Dichas aptitudes darán información adicional sobre éste cuando se asigne a un equipo.

Creación y gestión de plantillas de equipos Formadas por un conjunto de roles pero sin poder asignarles colaboradores.

Creación y gestión de equipos Formados por roles que serán realizados por diferentes colaboradores una vez sean asignados. Al crear un equipo se podrá tomar una plantilla como referencia.

Creación y gestión de perfiles vacantes Permitir a los equipos solicitar plazas vacantes para un rol específico. Además es conveniente poder identificar aquellos colaboradores que por una razón u otra deseen cambiar de equipo y gestionar su incorporación a nuevos equipos.

Navegación general sobre un proyecto Mostrar los datos más significantes de un proyecto que use el método Scrum para tener una visión global y rápida sobre el estado de éste y poder acceder fácilmente a los diferentes formularios de la aplicación.

Durante la realización del proyecto han aparecido nuevos requerimientos que han tenido que ser analizados para comprobar si eran factibles:

Creación y gestión de plantillas de reuniones Dichas plantillas determinarán los roles asistentes a una reunión y si deberán de asistir todos los colaboradores que efectúen el rol. Al crear una reunión se podrá tomar una plantilla como referencia y ésta será aplicada al equipo correspondiente.

Creación y gestión de reuniones Tener constancia de las diferentes reuniones que se convoquen en un equipo así como sus asistentes y los temas tratados.

Convocar reuniones Enviar peticiones de asistencia a las reuniones vía e-mail a los diferentes convocados a la reunión. A ser posible que las peticiones sean interpretadas por Microsoft Outlook como convocatorias, ya que es el cliente de correo usado en la empresa y el método actual de convocar reuniones.

3.2. Requisitos no funcionales.

A continuación se enumeran las diferentes características que deberá cumplir el proyecto.

Unificación de la gestión de Scrum Centralizando toda la información referente a la gestión del método a nivel de equipo, Product Backlog, Sprint Backlog e incidencias en una sola aplicación.

Control de acceso Debido a la centralización mencionada habrá que limitar el acceso a la información dependiendo del usuario que la consulte y el rol que realice en el proyecto que consulte.

Uso de las herramientas karat Implementación del proyecto haciendo uso del software propietario de la empresa UNIT4.

Histórico de trabajo La información gestionada y almacenada por el programa debe poderse consultar a modo de histórico para tener constancia del trabajo hecho.

3.3. Viabilidad.

Para realizar el estudio se analizarán las diferentes áreas que afectan al proyecto para comprobar si los requisitos propuestos son factibles y se ajustan a las condiciones.

3.3.1. Especificaciones.

Se debe crear un gestor de personal que tome como referencia el ya existente gestor que ofrece ekon común en UNIT4 y amplíe sus funcionalidades para poder gestionar un proyecto Scrum. Habrá tres proyectos más relacionados con la gestión de Scrum que se llevarán a cabo por otros proyectistas de la UAB y que en su conjunto formarán un producto único.

Las funcionalidades que se deben añadir a este nuevo gestor de Scrum deben incluir el control y gestión de equipos y roles, así como las relaciones de los colaboradores con éstos. Además se considera la inclusión de una gestión de reuniones para agrupar, centralizar y tener histórico del máximo número de eventos relacionados con el proyecto y el equipo de desarrollo.

Para el desarrollo del proyecto se aplicará la metodología Scrum en un equipo formado por cuatro proyectistas de la UAB, que se encargarán del desarrollo, y varios trabajadores de la empresa UNIT4, que harán las funciones de cliente y tutor. Pese a ser cuatro proyectos se mantendrá un seguimiento del trabajo de los compañeros de equipo. Así se asegurará que la unificación de los proyectos conformen un producto operativo con un funcionamiento similar en sus diferentes partes, como si se tratase de un único proyecto.

3.3.2. Viabilidad técnica.

Para la realización del proyecto será necesario el uso de las herramientas karat. Dichas herramientas engloban un conjunto de funcionalidades para facilitar la implementación de programas que trabajan con una base de datos como fuente de información. Además se incluyen otras funcionalidades como el diseño de interfaz de usuario o de formularios de impresión. Para hacer uso de las herramientas será necesario obtener una licencia de programador que será proporcionada por UNIT4.

Las herramientas karat se usan mediante el lenguaje de programación Java y por tanto será necesario el entorno de desarrollo *Java Eclipse Ganimedes*, para el cual han sido desarrolladas algunas funcionalidades adicionales que amenizan su uso.

3.3.3. Viabilidad operativa.

El proyecto será desarrollado mediante las herramientas karat y por ello será necesaria una formación técnica por parte de la empresa UNIT4. La empresa ofrece dos cursos de formación:

- El primero de ellos será impartido durante el mes de octubre en el campus de Sabadell de la *Escola d'Enginyeria*.
- El segundo será impartido durante el mes de diciembre en la sede de la empresa UNIT4 situada en Barberá del Vallés.

Además el proyecto se hará conjuntamente con otros tres proyectos llevados a cabo por proyectistas de la UAB. Dichos proyectos son los siguientes:

Automatización de sistemas de desarrollo ágil – Scrum: Product Backlog.

Por: Jose Luis González Samblas.

Automatización de sistemas de desarrollo ágil – Scrum: Sprint Management.

Por: Pedro Antonio Siles Jimenez.

Automatización de un sistema de Incidencias para el desarrollo Industrial de software.

Por: David Clemente Romero.

Pese a que cada proyectista trabajará en su propio proyecto se trata de un equipo de desarrollo mediante la metodología Scrum. Es por eso que se tendrá conocimiento del estado y evolución del resto de proyectos en todo momento. Además se proporcionará ayuda y se interaccionará con los otros miembros para las partes conjuntas.

3.3.4. Viabilidad temporal.

El convenio universidad-empresa estipula un rango de 500 a 560 horas para cada uno de los proyectistas. Esto proporciona un tiempo total de unas 2000 horas para todos los proyectos como mínimo y se considera tiempo suficiente para su desarrollo.

3.3.5. Viabilidad legal.

En el proyecto se trabajará con datos pertenecientes los trabajadores de la empresa. Sin embargo sólo se tratarán y almacenarán el nombre, la dirección de correo electrónico y el rol o profesión que realice.

Los datos anteriormente mencionados son considerados fuentes accesibles al público según el artículo 3 de la LOPD 15/1999, 13 diciembre.

3.3.6. Alternativas.

En la empresa se hace uso actualmente de un conjunto de hojas de cálculo (*Microsoft Excel* u *OpenOffice Calc*) para llevar la gestión de los Backlog de Product y de Sprint y para generar las gráficas de seguimiento. En cuanto a la gestión de empleados y roles no se usa ninguna aplicación que los relacione o contenga un histórico de trabajo. Esto es debido a que no dispone de una herramienta que interaccione con la gestión de colaboradores de la empresa y permita su asignación a diferentes equipos de trabajo.

Existen varios programas dedicados a la gestión de documentos y personal para una metodología Scrum. Sin embargo éstos no ofrecen una mejora sustancial a los métodos ya utilizados actualmente. Además no proporcionan una integración con el resto de herramientas de gestión de personal que dispone la empresa.

3.3.7. Planificación.

Para la planificación del proyecto, primero se hará una previsión general donde se planificarán las fases de desarrollo, formación y realización de documentación. A continuación se construirá un Product Backlog compuesto por historias que contenga los requerimientos funcionales y no funcionales especificados.

Posteriormente se hará una primera previsión donde se asignarán las diferentes historias a cada uno de los siete Sprint. Esta previsión quedará expuesta a cambios ya que a medida que se avance el proyecto es muy probable que se reasignen las historias y aparezcan de nuevas debido a peticiones del ProductOwner.

3.3.8. Previsión general.

#	Descripción	Fecha inicio	Fecha fin	Duración
	Proyecto	15/10/09	18/06/10	595h
1	**Configuración**	**09/11/09**	**10/11/09**	**4h**
1.1	Instalación del sistema	09/11/09	10/11/09	4h
2	**Documentación previa**	**23/11/09**	**09/12/09**	**20h**
2.1	Elaboración de informes previos para la universidad y la empresa.	23/11/09	09/12/09	20h
3	**Formación**	**15/10/09**	**22/12/09**	**125h**
3.1	Curso de ekon 1	15/10/09	30/10/09	30h
3.2	Curso de ekon 2	09/12/09	22/12/09	70h
3.3	Análisis del método Scrum	10/11/09	21/11/09	20h
3.4	Análisis de los actuales métodos Scrum usados en Unit4	23/11/09	28/11/09	5h
4	**Scrum**	**04/01/10**	**18/06/10**	**446h**
4.1	Release Planning Meeting.	04/01/10	05/01/10	10h
4.2	Sprint 1	07/01/10	27/01/10	60h
4.3	Sprint 2	28/01/10	17/02/10	60h
4.4	Sprint 3	18/02/10	17/03/10	80h
4.5	Sprint 4	18/03/10	14/04/10	56h
4.6	Sprint 5	15/04/10	05/05/10	60h
4.7	Sprint 6	06/05/10	27/05/10	56h
4.8	Sprint 7	28/05/10	18/06/10	64h

Figura 12. Tabla de previsión general.

Se prevé que la redacción de la memoria se realizará durante el último Sprint del proyecto.

3.3.9. Diagrama de Gantt.

El proyecto tendrá comienzo el día 15 de octubre del 2009 cuando empiece el curso de iniciación a las herramientas karat. El final del proyecto será el día 18 de julio de 2010 cuando termine el último Sprint y por tanto terminen todas las fases de desarrollo, test, pruebas y documentación previstas. La planificación temporal del proyecto se expone en el siguiente diagrama de Gantt.

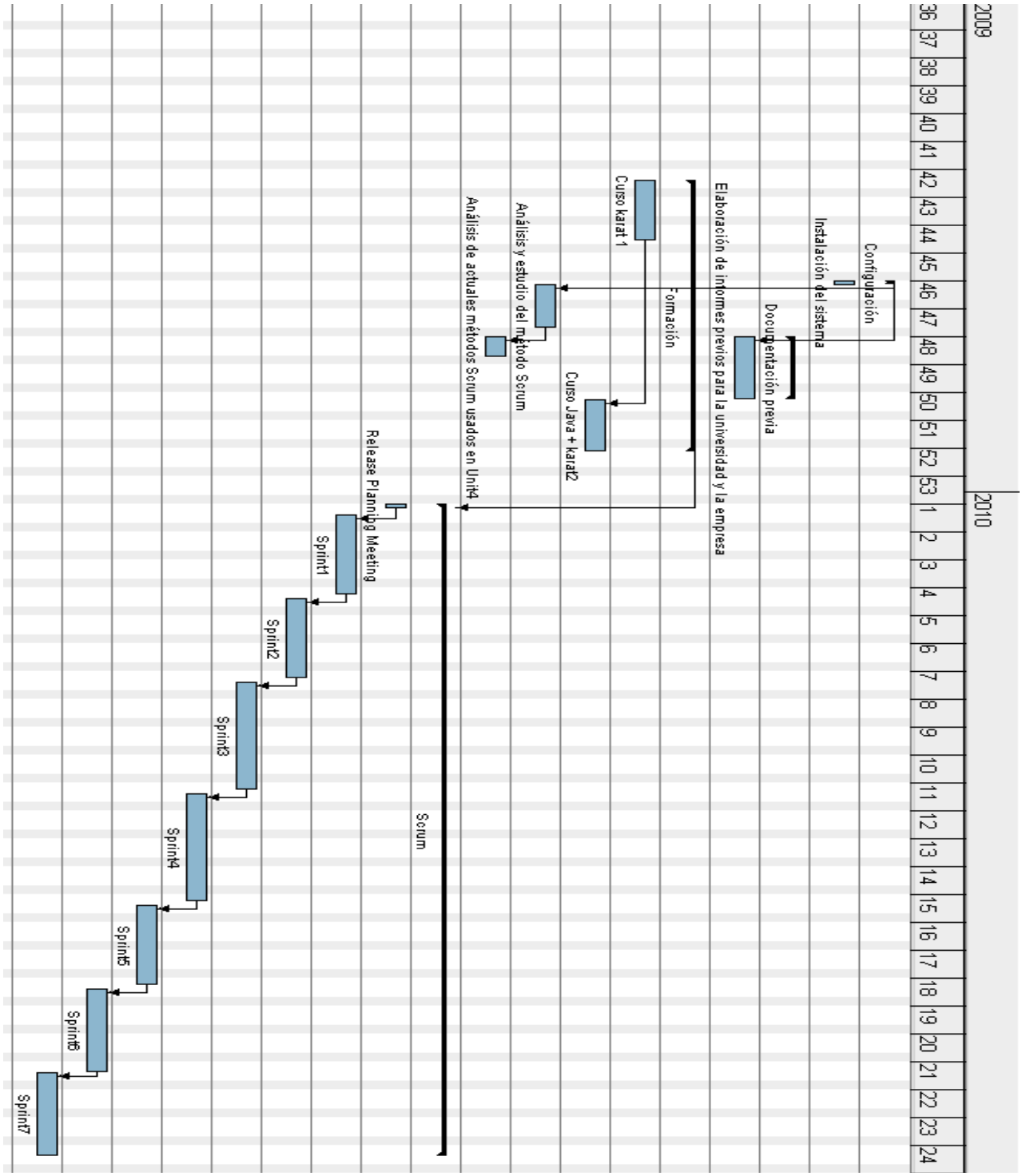

Figura 13. Diagrama de Gantt del proyecto

3.3.10. Product Backlog.

Una vez especificados los requerimientos funcionales y no funcionales se hará una reunión con los diferentes miembros del equipo, el ScrumMaster y el ProductOwner. En esta reunión se debatirán a fondo los diferentes aspectos a implementar y se hará una valoración en puntos y de prioridad de éstos. Finalmente se obtendrá como resultado el Product Backlog que determinará qué es lo que hay que hacer:

#	Nombre de la historia	Prioridad	Valoración
34	Definidora de roles	1	2
35	Definidora de colaboradores Scrum	1	2
36	Definidora de plantilla equipos	2	2
37	Definidora de equipos	1	3
38	Vacantes	3	2
39	Histórico de personal	2	2
40	Consulta de proyectos	2	5
41	Asignación de colaboradores	1	5
44	Control de permisos	1	5

Figura 14. Product Backlog: inicial.

Durante la realización del proyecto han surgido varias propuestas que han sido posteriormente convertidas a historias y añadidas al Product Backlog.

#	Nombre de la historia	Prioridad	Valoración
47	Impresión de equipos	3	1
48	Impresión de colaboradores	3	1
49	Definición tipo de reuniones	2	3
50	Definidora de reuniones	2	2
51	Convocar reunión	3	2

Figura 15. Product Backlog: historias añadidas.

Aparte existirán unas historias comunes para los cuatro proyectos cuyo valor en puntos tendrá en cuenta que se trata de cuatro personas:

#	Nombre de la historia	Prioridad	Valoración
01	Presentación de proyectos	1	5
02	Analizar otros productos	3	1/2
03	Definición del modelo de datos	1	5
69	Presentación y memoria	1	5

Figura 16. Product Backlog: historias comunes.

La historia 69 tiene un número tan alto puesto que pese a existir desde el principio, nunca ha sido considerada una historia hasta el final ya que no se sabía si daría tiempo a incluirla en el proceso.

3.3.11. División según Sprint.

Cuando el Product Backlog haya sido creado se hará un primer reparto de las diferentes historias entre los Sprint de los que se disponga. Este reparto no será para nada estricto y se verá modificado a medida que avance el proyecto y surjan nuevas historias o bien se alarguen o acorten las ya existentes según lo que estaba previsto. Se ha omitido la conversión de puntos a horas en esta planificación ya que los puntos pueden variar su valor dependiendo del progreso del equipo.

Sprint	**Historias planificadas**
Sprint 1	Definición del modelo de datos. Presentación de proyectos. Analizar otros productos. Definidora de roles.
Sprint 2	Definidora de colaboradores Scrum. Definidora de plantillas de equipos. Definidora de equipos.
Sprint 3	Histórico de personal. Asignación de colaboradores. Vacantes.
Sprint 4	Control de permisos. Sugerencias.
Sprint 5	Consulta de proyectos. Sugerencias.
Sprint 6	Sugerencias.
Sprint 7	Presentación y memoria.

Figura 17. Historias según Sprint: previsión inicial.

La planificación anterior es la que se realizó al principio del proyecto y no tiene en cuenta las propuestas que aparecieron una vez ya empezado el Scrum. A partir del Sprint 4 se puede comprobar como aparece la historia Sugerencias. Esto es debido a que por convenio no se aceptan modificaciones o sugerencias hasta pasados unos Sprint, de esta forma se puede construir la base de la aplicación antes de añadirle funcionalidades adicionales.

Capítulo 4. Diseño.

Este apartado recogerá el diseño del proyecto Team & Role. Se creará un control de colaboradores, equipos, roles y reuniones que servirá al resto de proyectos como fuente de datos para sus distintos requerimientos. Pese a tratarse de un módulo de control para la metodología Scrum se hará un diseño independiente del resto de módulos para que sea escalable y sea posible su reutilización para otras metodologías de equipo.

4.1. Herramientas karat.

Un requerimiento no funcional del proyecto es el uso de las herramientas karat para el desarrollo de la aplicación. Dichas herramientas componen un framework orientado al desarrollo de aplicaciones ERP y CRM. Se hará una muy breve introducción a éstas y a la forma de usarlas para poder así comprender el diseño y la implementación del proyecto.

Las herramientas karat se basan en una entidad denominada objeto de negocio. Estas entidades toman o bien proporcionan información para una o varias consultas a base de datos y ofrecen una variedad de funcionalidades sobre los diferentes campos de éstas. A partir de los objetos de negocio es posible crear formularios para interaccionar con el usuario ya sea tomando o mostrando datos. También es posible crear listados de impresión que, dado un formato, proporcionarán una impresión de los datos del objeto en diferentes formatos soportados (pdf, word, excel, imagen, impresión).

Sin embargo los objetos de negocio y formularios no son suficientes para el desarrollo de una aplicación de estas características y por ello se da la opción de ampliar sus funcionalidades sobrecargando los ya existentes procesos de karat mediante el lenguaje Java. Para ello es necesario derivar la clase de control

del objeto de negocio o formulario y sobrecargar los métodos correspondientes a los eventos cuya funcionalidad se quiere ampliar.

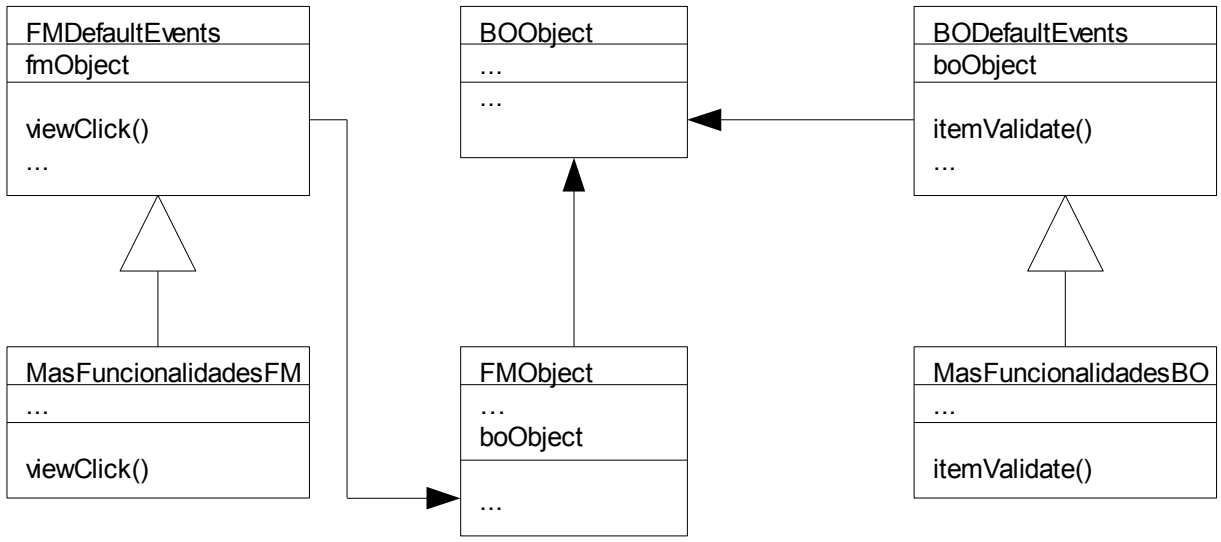

Figura 18. Diagrama de clases genérico de una aplicación que usa karat.

4.2. Modelo físico de la aplicación.

Debido al uso de las herramientas karat, el diseño de la aplicación estará basado en la estructura de la base de datos sobre la que ésta se desarrolle:

Colaboradores Se denomina colaborador a una persona que es contratada y por tanto forma parte de una empresa. Los colaboradores que usa la aplicación representan un subconjunto de los colaboradores existentes en las bases de datos de ekon común. Cuando se hable de colaboradores realmente se hará referencia al subconjunto colaboradores Scrum.

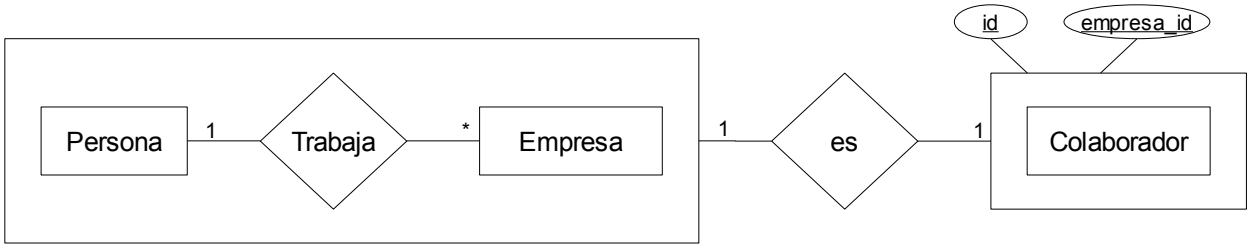

Figura 19. Entidad-Relación: Colaborador.

Roles Un rol engloba un conjunto de características que conforman un perfil de trabajo. Además determinará qué privilegios se otorgan a quien realice el rol. Para escalar los permisos de los roles, y no limitarlos sólo a los de la aplicación, se usa un campo numérico donde se guarda una codificación de los permisos otorgados. Así se podrán añadir nuevos permisos sin la necesidad de modificar la tabla.

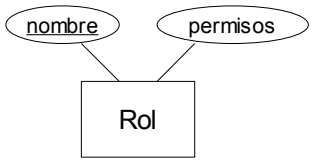

Figura 20. Entidad-Relación: Rol.

Relación con colaborador Las aptitudes de un colaborador hacen referencia a su relación con los diferentes roles existentes. Ese conjunto determinará sus diferentes capacidades y dará información adicional sobre el colaborador.

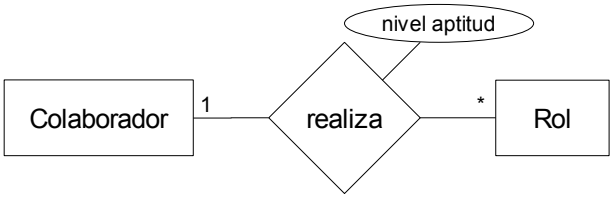

Figura 21. Entidad-Relación: Aptitudes del Colaborador.

Equipos Las metodologías de desarrollo ágil de software basan su trabajo en un equipo. Éste es asignado a un proyecto y está formado por un conjunto de perfiles que a su vez son llevados a cabo por diferentes colaboradores.

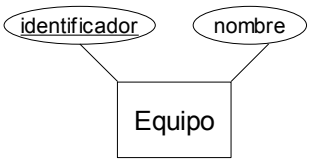

Figura 22. Entidad-Relación: Equipo.

Relación con roles La estructura de un equipo es determinada por los diferentes roles que requiere y ello generará las plazas de las que se dispone para completarlo.

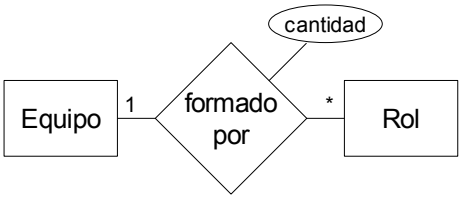

Figura 23. Entidad-Relación: Plaza.

Relación con colaboradores Una vez se ha especificado la composición de un equipo ya se conocen las diferentes plazas disponibles para los colaboradores. Por tanto los colaboradores podrán ocupar estas plazas y su asignación será durante un periodo de tiempo determinado. Además será posible que un colaborador que forme parte de un equipo quiera cambiar de equipo.

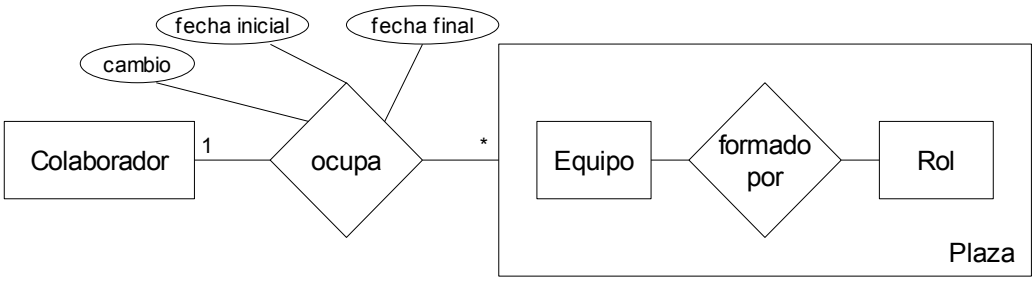

Figura 24. Entidad-Relación: Miembro.

Vacantes Una vez se ha creado y se ha definido la composición de un equipo existe la posibilidad de que no se rellenen todas las plazas. Estas plazas vacías puede que se quieran ocupar o no. Con ese fin se crean las vacantes donde el equipo podrá especificar qué plazas están disponibles y requieren a alguien para el puesto.

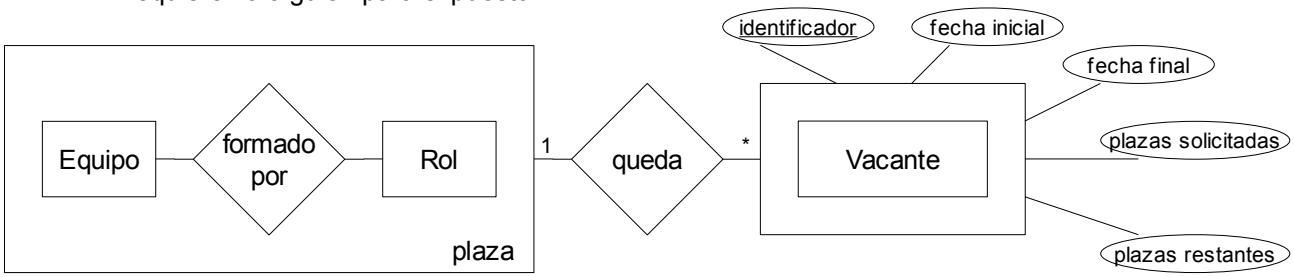

Figura 25. Entidad-Relación: Vacante.

Reuniones Las reuniones son eventos de un equipo que sólo pueden existir si pertenecen a uno. El propósito de convocar una reunión es el de exponer un conjunto de objetivos y tomar decisiones en equipo sobre los objetivos expuestos.

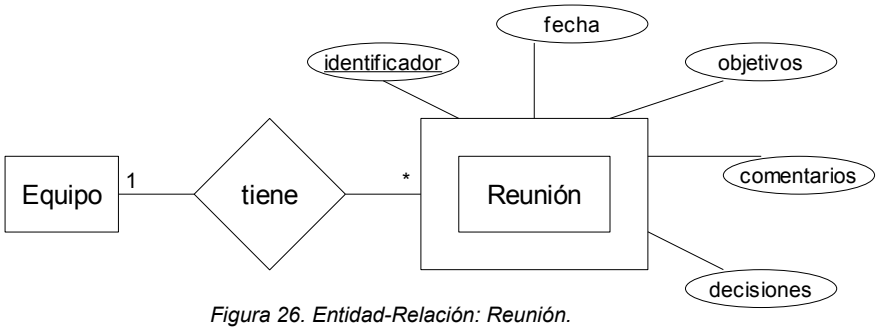

Figura 26. Entidad-Relación: Reunión.

Relación con miembros de un equipo Aunque una reunión pertenezca a un equipo siempre es un miembro del equipo quien la convoca y a su vez hay un conjunto de miembros del equipo que son convocados a ésta para tratar los diferentes objetivos.

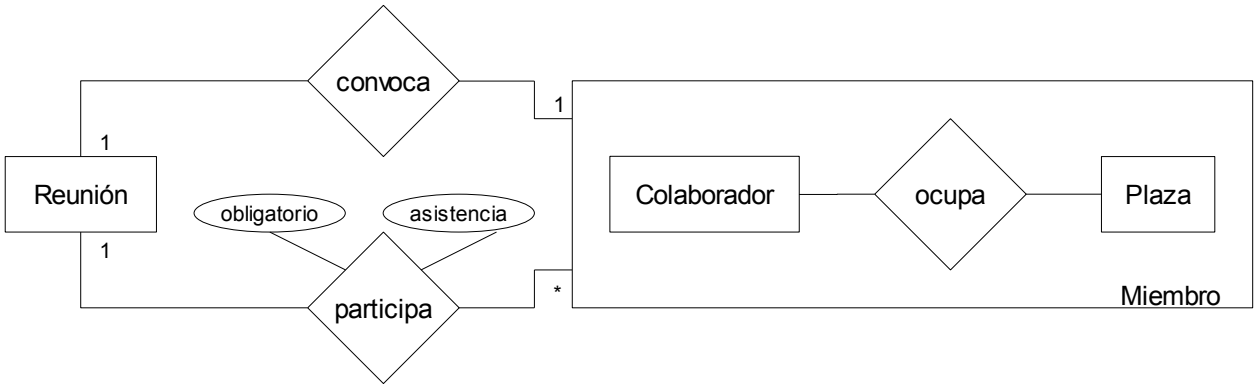

Figura 27. Entidad-Relación: Convocante y Convocados.

Plantillas Con la finalidad de agilizar el proceso de creación de equipos y reuniones se ha considerado la creación de plantillas. Así será posible partir de una base y simplificar los procesos de creación. Estas plantillas se tratarán de la forma más general posible y es por ello que estarán compuestas por roles y no por miembros.

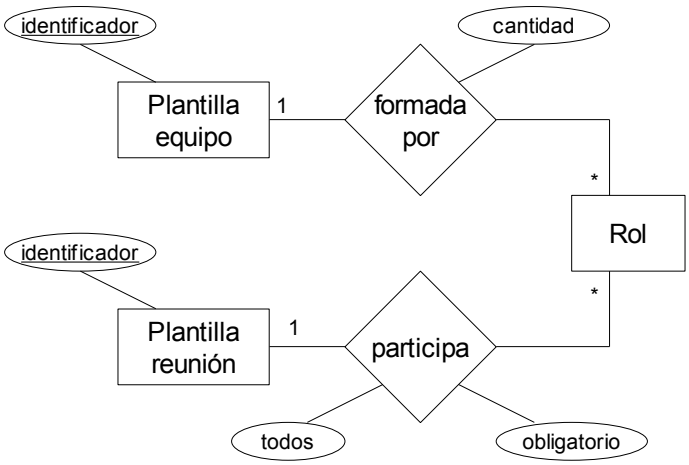

Figura 28. Entidad-Relación: Plantillas de Equipo y Reunion.

4.3. Control de permisos.

Para el diseño del control de permisos se han analizado por un lado las posibilidades que nos ofrece karat y por otro se ha planificado un nuevo modelo de gestión de permisos basado en la aplicación.

Permisos karat

El conjunto de herramientas karat proporcionan una gestión de permisos interna con la cual es posible definir grupos de roles y agregar a los diferentes usuarios de la aplicación a estos grupos. El control de permiso de acceso a los grupos de roles, ya sean de lectura o de escritura, se especifica para cada uno de los diferentes formularios a los que la aplicación permite acceder. Por tanto todo usuario que pertenezca a un rol tendrá los permisos especificados por éste en el formulario en que se encuentre, independientemente de a qué equipo pertenezca.

Permisos basados en la aplicación

Ya que en la aplicación se gestionan las diferentes asignaciones de personal a cada equipo, es posible consultar a partir de ésta cual es el rol de cada colaborador en cada uno de los diferentes equipos. Para ello será necesario que los roles determinen un conjunto de permisos de lectura y escritura sobre los diferentes módulos de la aplicación. Por tanto al realizar una consulta sobre los roles de un colaborador en un equipo se obtendrán los diferentes permisos de éste y se podrá actuar en consecuencia.

Decisión final

Los permisos basados en la aplicación son mas específicos y hacen distinción según el equipo. Para hacer posible la identificación del usuario de la aplicación cada colaborador estará relacionado con su cuenta. De esta forma será posible comprobar qué permisos hay activos según quién y qué se esté consultando en ese momento. Sin embargo habrá momentos en que no podrá comprobarse qué roles tiene el usuario, ya sea

porque lo que se está consultando no depende de ningún equipo o bien porque se está creando el equipo y aún no hay nadie asignado.

Por tanto pese a que el control de permisos se haga basado en la aplicación se utilizará un método mixto utilizando uno de los roles que ofrece karat. Se hará uso del rol administrador que será otorgado a un conjunto de usuarios. Como los puntos de la aplicación donde no se puede hacer uso del tratamiento interno de permisos corresponden a formularios de configuración o de creación de nuevos elementos, son acciones que concuerdan con este rol y su uso queda justificado.

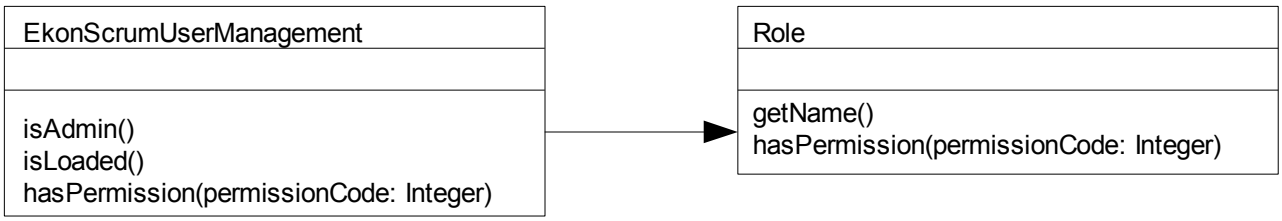

Figura 29. Diagrama de clases de control de permisos.

Capítulo 5. Implementación.

En este apartado se recogerá la evolución del proyecto conjunto. Para ello se mostrará el Sprint Backlog de cada uno de los Sprint, así como su gráfica Burndown para poder ver cómo ha ido avanzando. También se expondrá, mediante diagramas de casos de uso, cómo se han implementado los diferentes requerimientos del proyecto Team & Role.

Finalmente se hará mención de cómo se ha estructurado la aplicación para poder comprender los cuatro proyectos en uno de solo. Terminada la explicación sobre la implementación se presentarán los resultados y el test de pruebas.

5.1. Evolución del proyecto: Sprint Backlog.

Para seguir la evolución del proyecto se hará uso de los materiales que proporciona Scrum. Para cada Sprint se mostrará el Sprint Backlog, sólo sus historias, y la gráfica Burndown. De las historias de los Sprint se remarcarán aquellas pertenecientes al proyecto desarrollado, Team & Role, y aquellas en las que se haya participado activamente. Así podrá verse el progreso que ha ido tomando tanto el proyecto realizado como el conjunto de proyectos.

Además se hará un seguimiento de la velocidad especificada en cada Sprint. Al hacer referencia a la velocidad se puede hacer por el total de puntos que se prevén en un Sprint, o bien por la cantidad de puntos realizados por colaborador/día, expresados en tanto por ciento del punto.

El valor de puntos que se tomó como referencia fue de 8 horas por 1 punto. Como el contrato con la empresa especifica que el trabajo por parte de los becarios es a media jornada, una velocidad del 50% sería equiparable a una velocidad de 100% en un equipo a jornada completa.

Sprint 1

Al tratarse del primer Sprint donde los miembros del Team aún se tenían que habituar tanto al proyecto como al sistema de trabajo y a las herramientas, se tomó una velocidad del 33%, lo que equivale a 20 puntos.

Nombre de Historia	Puntos de Historia	Prioridad	Horas previstas
Presentación de proyectos	**5**	**1**	**32**
Analizar otros productos	**1/2**	**3**	**4**
Modelo de datos	**5**	**1**	**35**
Definidora de Producto	1	1	8
Diseñar imagen de la tarjeta.	1	1	4
Impresión de listado de las tarjetas.	2	1	16
Definidora de roles	**2**	**1**	**10**
Reunión sizing y planning.	**5**	**1**	**72**

Figura 30. Sprint 1: Previsión.

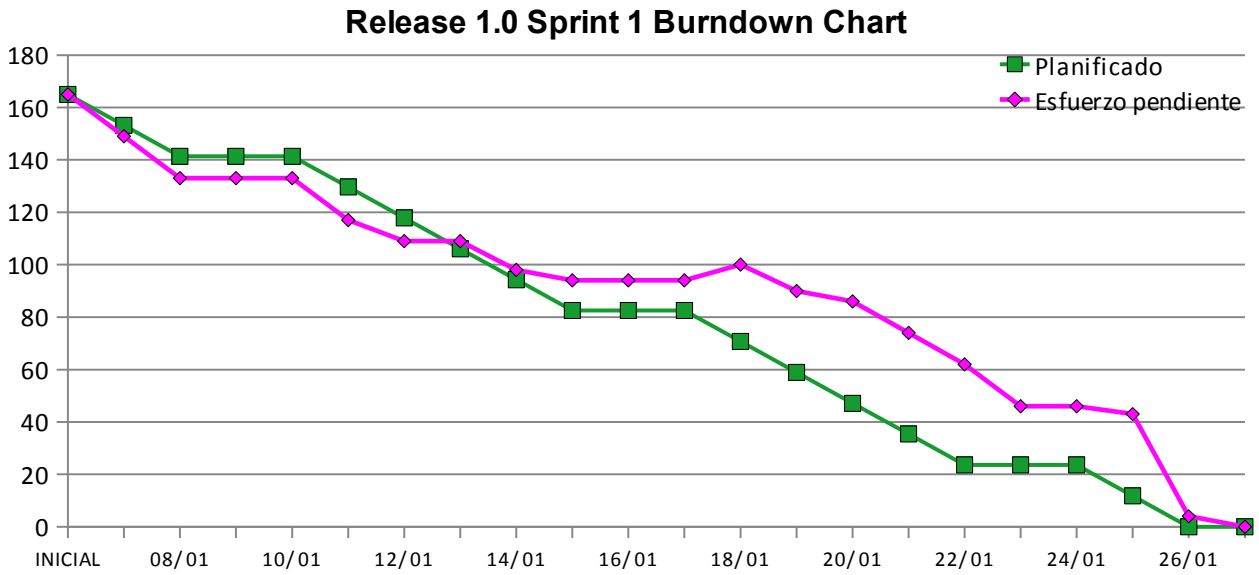

Figura 31. Sprint 1: Burndown.

La historia *Analizar otros productos* tiene un valor bajo debido a que ya se habían analizado previamente al inicio de Scrum. Esta historia sólo pretendía que se analizase cómo se habían implementado los *backlogs* en los diferentes productos existentes.

Como muestra la gráfica, el Sprint terminó con carga de trabajo, por tanto fue un fracaso. Las historias *Definidora de Roles* e *Impresión listado de las tarjetas* no fueron terminadas a tiempo. Es por ello que estas historias se volvieron a puntuar, teniendo en cuenta el trabajo ya realizado, y se volvieron a incluir en el Product Backlog como nuevas historias.

Parte del trabajo no hecho fue debido al retraso de la sesión de sizing y planning, ya que el ProductOwner se encontraba ausente cuando el proyecto se inició. Sin embargo en el Sprint Retrospective se hizo énfasis en los aspectos positivos del Sprint y se procedió a empezar el siguiente con espíritu de superación.

Sprint 2

Tras el fracaso del Sprint 1, que a una velocidad del 33% no había terminado con éxito, se tomaron las historias previstas para el Sprint 2 que se había hecho al inicio del proyecto. El total de puntos era de 28 y eso significaba incrementar la velocidad del 33% al 50%.

Tras analizar de nuevo las historias, se consideró que algunas de ellas estaban sobrevaloradas, y el Team se comprometió a terminar con lo previsto en un inicio. Por tanto la velocidad del Sprint quedó en 28 puntos, el 50%.

Nombre de Historia	Puntos de Historia	Prioridad	Horas previstas
Impresión de tarjetas (Re-evaluada)	2	1	14
Definidora de roles (Re-evaluada)	**1**	**1**	**7**
Definidora de proyectos	2	1	24
Definidora de historias	3	1	24
Definidora de Sprints	8	1	48
Registro en línea de incidencias	8	1	64
Definidor de colaboradores Scrum	**2**	**1**	**16**
Definidor de plantillas de equipo	**2**	**2**	**16**

Figura 32. Sprint 2: Previsión.

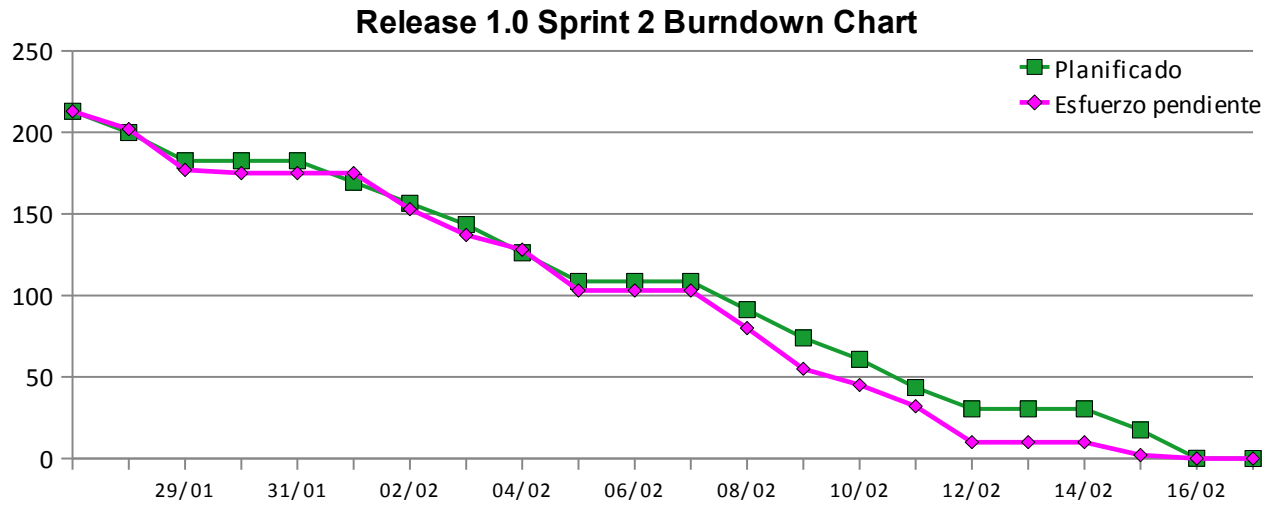

Figura 33. Sprint 2: Burndown.

El Sprint se terminó como estaba previsto. Sin embargo, esto fue debido a que las historias habían estado mal puntuadas en el *sizing* y llevaron menos tiempo del previsto en realizarse.

En el Retrospective se decidió hacer un promedio de las velocidades, para ser estrictos con la metodología, ya que las historias restantes no parecían estar mal puntuadas.

Sprint 3

Para este Sprint se hizo el cálculo adecuado de la velocidad quedando ésta en un 31%, por tanto 26 puntos. El número elevado de puntos pese a la poca velocidad es debido a que este Sprint era más largo que los anteriores.

Nombre de Historia	Puntos de Historia	Prioridad	Horas previstas
Definidora de equipos	**3**	**1**	**18**
Asignación de colaboradores	**5**	**1**	**32**
Definidora release	1	1	8
Definidora propuestas	2	3	16
Control delayeds	2	3	16
Control dependencias	2	1	16
Tipos de tareas	1	1	6
Registro de tareas	5	1	32
Gestor de velocidad	2	1	16
Validación registro remoto	3	2	24
Histórico de personal	***2***	***2***	***13***
Definidora de seguimiento daily	*3*	*1*	*13*

Figura 34. Sprint 3: Previsión.

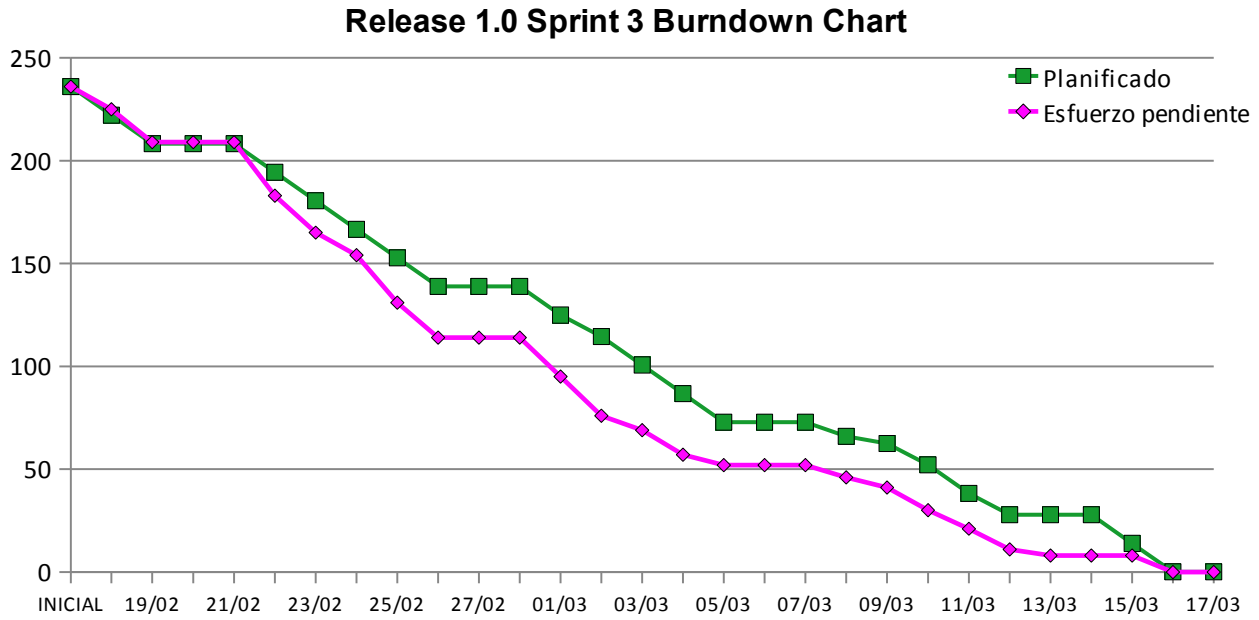

Figura 35. Sprint 3: Burndown.

En este caso la previsión de velocidad resultó ser muy inferior a la capacidad de trabajo del Team. Por ello se añadieron sugerencias y dos nuevas historias, inicialmente planeadas para el Sprint siguiente.

Pese a que la metodología indica que el Sprint Backlog no puede ver su cantidad de historias modificadas, se consideró que hace referencia a no eliminar historias. En este caso las historias eran incluidas y el cambio no significaba hacer menos carga de la prevista, sino más.

Sprint 4

La velocidad real del Sprint 3 resultó ser bastante superior a la prevista, ya que en total se realizaron 31 puntos y varias sugerencias. Así pues, para la previsión del Sprint 4 se tomó un valor del punto del 38% que, teniendo en cuenta semana santa, resultó un total de 21 puntos.

Nombre de Historia	Puntos de Historia	Prioridad	Horas previstas
Confirmación de propuestas	1	3	8
Imprimir fichas en historias	2	3	16
Impresión Product Backlog	1	1	8
Visualización Burndown	3	1	26
Mantenimiento Retrospective	2	3	16
Widgets Incidencias	3	2	22
Vacantes	**2**	**3**	**14**
Control de permisos	**5**	**1**	**31**
Resolver incidencias	2	2	13

Figura 36. Sprint 4: Previsión.

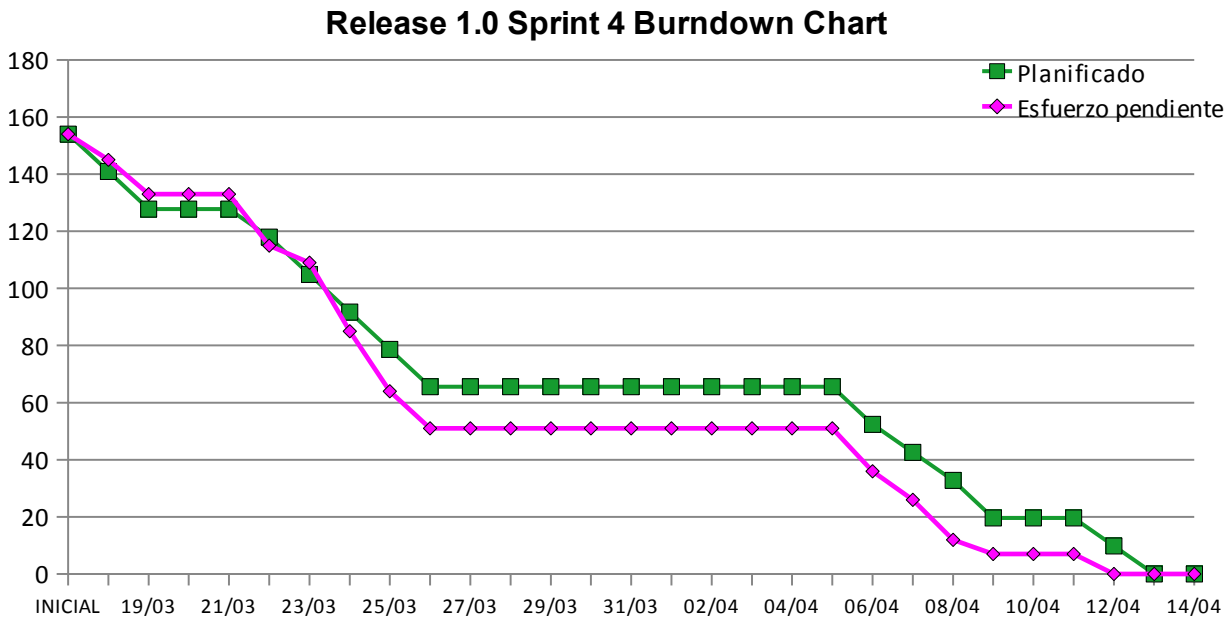

Figura 37. Sprint 4: Burndown.

El Sprint 4 fue como estaba planificado e incluso se aprovechó para resolver varias sugerencias.

La velocidad prevista empezaba a ajustarse a la velocidad real del Team, lo que equivaldría a un 40% del valor real del punto, que teniendo en cuenta la media jornada laboral equivaldría a un 80%. Por tanto, la asignación de puntos a las historias del *sizing* se había aproximado bien.

Sprint 5

En este Sprint, la gran mayoría del proyecto había sido desarrollada y se empezaban a escoger historias de prioridades inferiores. Se estimó un valor del 42% del punto y por tanto la velocidad del Sprint resultó ser 24 puntos.

Nombre de Historia	Puntos de Historia	Prioridad	Horas previstas
Visualizar paneles	5	1	48
Clasificación bugs	2	1	13
Informes abiertos	2	2	16
Definición de tipo de reuniones	**3**	**2**	**24**
Reuniones	**2**	**2**	**16**
Convocar reuniones	**2**	**3**	**11**
Aplicar permisos al Product Backlog	1	1	8
Impresión de propuestas	2	2	15
Bugs y sugerencias	**5**	**3**	**-**

Figura 38. Sprint 5: Previsión.

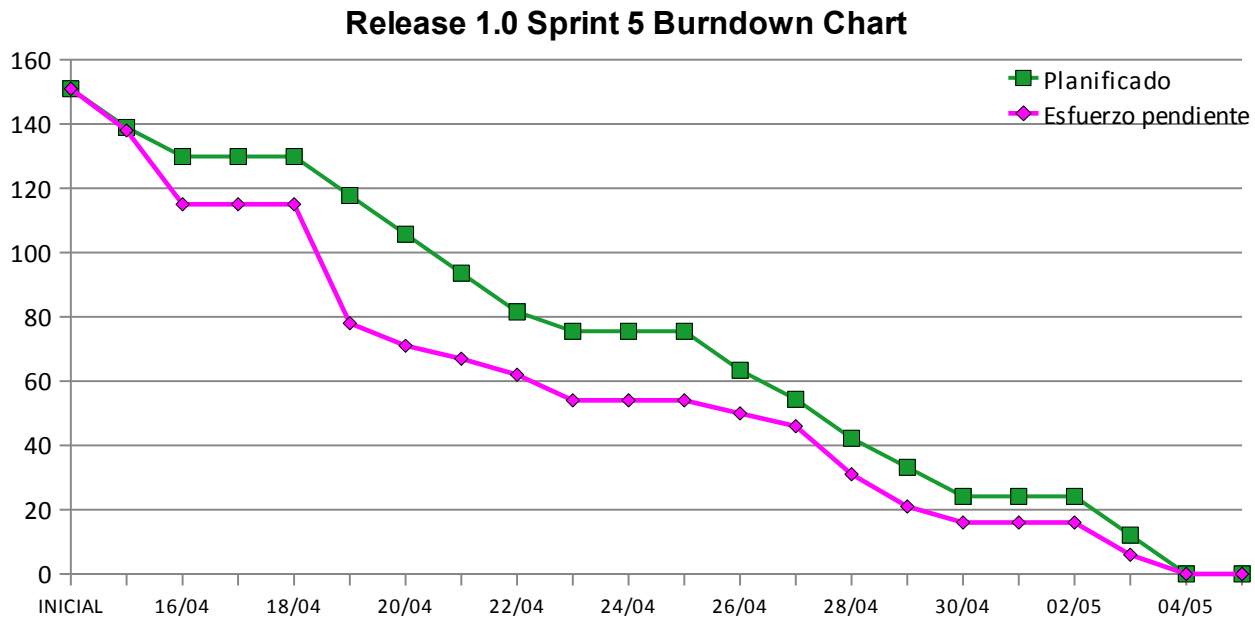

Figura 39. Sprint 5: Burndown.

Para este Sprint se reservó un total de 5 puntos de dedicación para sugerencias y bugs. En la gráfica de Burndown no se incluyó el seguimiento de estas tareas adicionales, sin embargo si que se se hizo su seguimiento en el panel de seguimiento.

Sprint 6

A estas alturas el proyecto se encontraba en un estado prácticamente terminado. Por ello se volvieron a dedicar puntos a corrección de bugs e implementación de sugerencias, sobretodo esta segunda. La velocidad se mantuvo respecto del anterior Sprint y por tanto el total de puntos asignados fue de 25.

Nombre de Historia	Puntos de Historia	Prioridad	Horas previstas
Widgets Product Backlog	5	2	36
Reuniones del Sprint	**5**	**1**	**30**
Consulta de proyectos	**5**	**2**	**30**
Responsable del proyecto	1	3	10
Listados de componentes	2	3	14
Listados de bugs	2	2	20
Bugs y sugerencias	**5**	**3**	**-**

Figura 40. Sprint 6: Previsión.

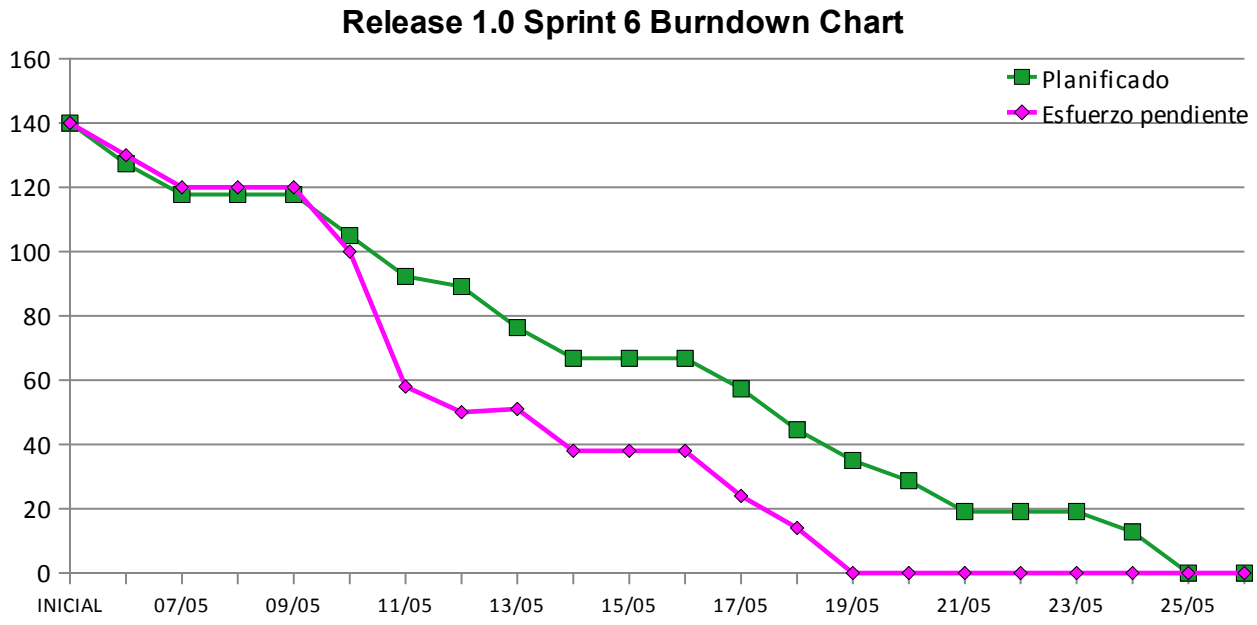

Figura 41. Sprint 6: Burndown.

Como puede verse, en este caso, el total de historias previstas se terminó cinco días antes de la entrega. Esto es debido a que la mayoría de historias del Sprint estaban mal valoradas.

En la gráfica Burndown no se muestra el seguimiento de las tareas de sugerencias y bugs.

La historia de *reuniones del Sprint* se propuso durante el Review del Sprint 5. En ella, se pretendía relacionar los Sprint con las Reuniones, para tener constancia de las reuniones que se definen en la metodología. Es por ello que, pese a pertenecer al proyecto Sprint Backlog, se colaboró en su desarrollo.

Sprint 7

Tratándose del último Sprint no podían quedar muchas historias. La mayoría de los esfuerzos se dedicaron a corregir bugs, implementar sugerencias y, sobretodo, asegurar que la aplicación estuviese lista para su presentación.

Aprovechando que la carga de trabajo era menor, se dedicó tiempo de Sprint a preparar las presentaciones de la empresa. Manteniendo la velocidad de los últimos Sprint se disponía de un total de 25 puntos a repartir. Del total de 25 puntos, sólo se repartieron 19 y el resto se dejó para corregir errores.

Nombre de Historia	Puntos de Historia	Prioridad	Horas previstas
Widgets Sprint Backlog	3	1	24
Sprint Retrospective	2	3	12
Formato de interpretación de errores	?	3	1
Impresión de equipos	**1**	**3**	**12**
Impresión de colaboradores	**1**	**3**	**6**
Permisos para Sprint Backlog	**1**	**1**	**8**
Impresión de componentes	1	2	8
Widgets Product Backlog (sugerencia)	5	3	36
Presentación y memoria	**5**	**1**	**40**
Sin asignar / Preparar aplicación	**6**	**1**	**-**

Figura 42. Sprint 7: Previsión.

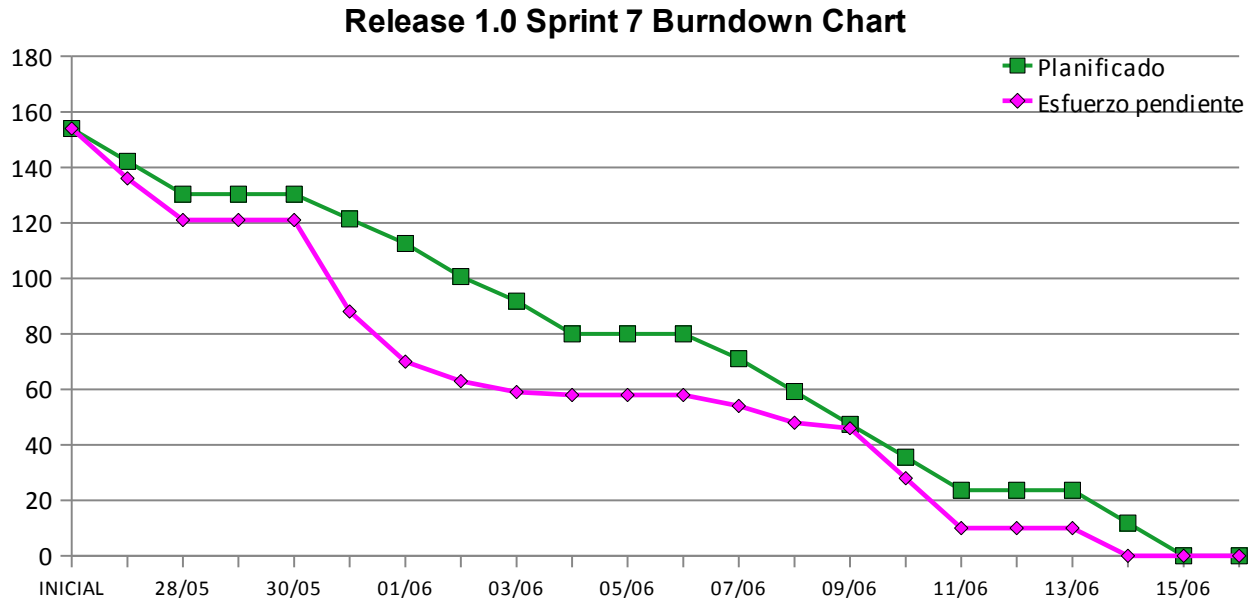

Figura 43. Sprint 7: Burndown.

Después del gran avance de la segunda semana, y aprovechando que la aplicación estaba siendo probada por el equipo *Ready,* se dedicaron recursos a corrección de bugs que surgieron a última hora y a implementar algunas sugerencias de poca carga. Estas correcciones deberían de haberse dejado para el final del Sprint, una vez todo estuviese terminado, pero como la presentación de proyectos dentro de la empresa estaba prevista para una semana después del final de éste, se priorizó el acabado de la aplicación.

5.2. Implementación de los requerimientos.

Los requerimientos han sido implementados en la aplicación a través de diferentes formularios. Al entrar a estos formularios el usuario será capaz de realizar varias acciones dependiendo del rol que posea en la aplicación o en el equipo/proyecto que esté consultando.

La aplicación está estructurada por carpetas. Navegando por ellas se puede acceder a los diferentes formularios. Para exponer las funcionalidades de estos se mostrarán diagramas de casos de uso, acompañados de una breve explicación.

Cuando se especifica que un miembro del equipo tiene permiso de lectura o escritura, se hace referencia a los permisos sobre gestión de equipos y roles:

Scrum Configuración/Roles/Roles Los usuarios administradores podrán crear y eliminar roles, así como añadir y quitar permisos.

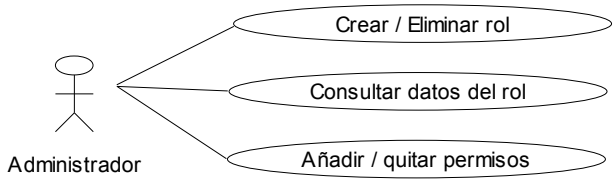

Figura 44. Casos de uso: Roles.

Scrum Configuración/Aptitudes/Niveles de aptitud Los usuarios administradores podrán crear y eliminar diferentes niveles de aptitud, que posteriormente podrán asignar a los colaboradores.

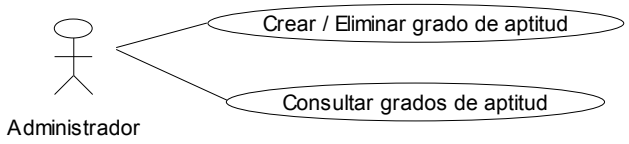

Figura 45. Casos de uso: Niveles de aptitud.

Scrum/Colaboradores/Colaboradores Los usuarios administradores podrán buscar colaboradores almacenados en las bases de datos de ekon común e importarlos a Scrum. También podrán gestionar las aptitudes de éstos.

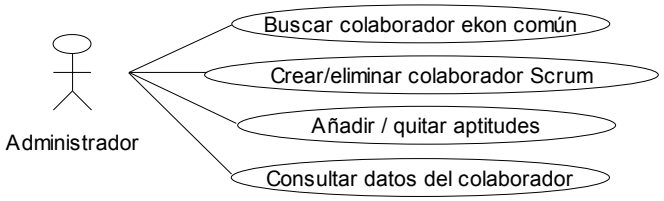

Figura 46. Casos de uso: Colaboradores.

Scrum/Colaboradores/Historial de colaboradores Cualquier usuario podrá consultar el historial de cualquier colaborador, ya que son datos públicos.

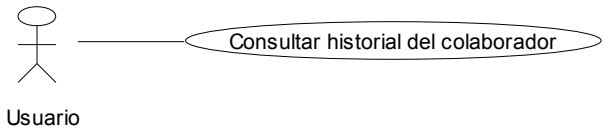

Figura 47. Casos de uso: Historial de colaboradores.

Scrum/Colaboradores/Impresión de colaboradores De la misma forma que cualquier usuario puede consultar el historial de cualquier colaborador, también le será posible imprimirlo.

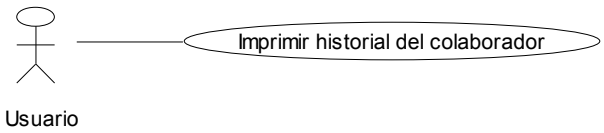

Figura 48. Casos de uso: Impresión de colaboradores.

Scrum Configuración/Equipo/Plantillas de equipo Los administradores podrán crear y eliminar plantillas de equipos. Las plantillas estarán compuestas por un conjunto de roles y las plazas disponibles para cada uno de ellos.

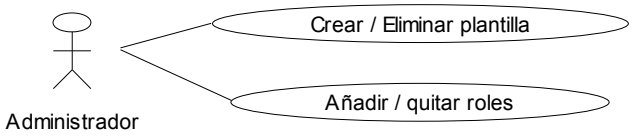

Figura 49. Casos de uso: Plantillas de equipo.

Scrum/Equipos/Equipos Los administradores podrán crear y eliminar equipos. Además de ellos, los miembros del equipo con permiso de escritura podrán añadir y quitar roles y usar plantillas para definir la estructura del equipo. Los usuarios con permiso de lectura podrán consultar la estructura del equipo.

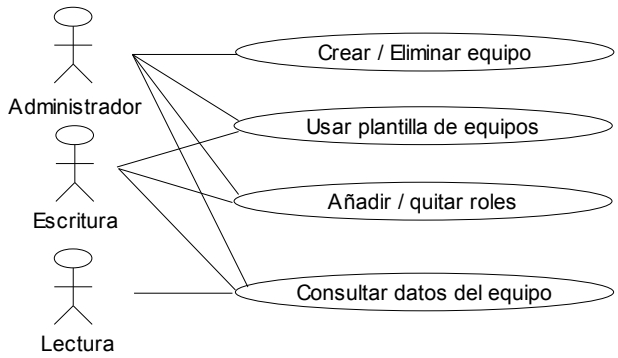

Figura 50. Casos de uso: Equipos.

Scrum/Equipos/Asignación de colaboradores Los administradores podrán añadir y quitar colaboradores a un equipo determinando su rol asignado. También podrán gestionar las vacantes y realizar una búsqueda de colaboradores según las aptitudes de éstos, para que les sirva de ayuda al asignar nuevos miembros. Se tendrá también la posibilidad de determinar qué colaboradores quieren cambiar de equipo, lo que les incluirá en la lista de colaboradores con petición a cambio, que también puede ser consultada para asignar nuevos miembros.

Cuando un administrador haya asignado a algún miembro el permiso de escritura, éste podrá realizar las mismas tareas que el administrador, permitiendo que los propios miembros del equipo

hagan la gestión.

Los miembros con permiso de lectura podrán ver las asignaciones, las vacantes y qué miembros del equipo tienen activa la petición de cambio, pero no podrán consultar la lista de miembros de otros equipos que esperan un cambio.

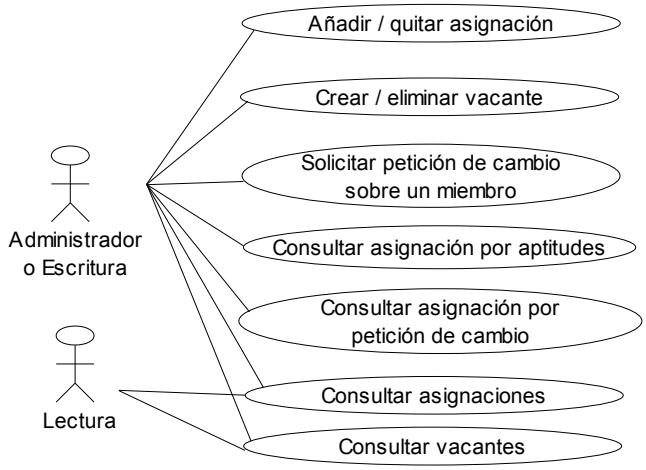

Figura 51. Casos de uso: Asignación de colaboradores.

Scrum/Equipos/Impresión de equipos Los administradores y cualquier miembro del equipo podrá imprimir el historial de éste. En el historial se muestra un listado de los miembros que ha tenido asignados, así como el rol que han realizado y también las vacantes que ha habido. El hecho de que cualquier miembro del equipo pueda imprimir el historial, es porque los datos mostrados están debidamente filtrados y son accesibles a todos. Por otro lado, los datos que se muestran en la asignación de colaboradores tienen un carácter administrativo y de gestión y no son públicos.

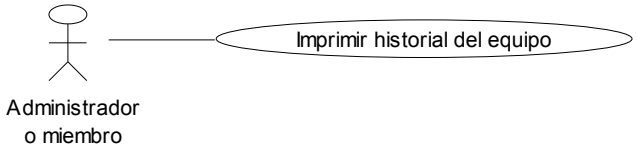

Figura 52. Casos de uso: Impresión de equipos.

Scrum Configuración/Reuniones/Plantillas de reuniones Los administradores podrán crear y eliminar plantillas de reuniones, así como determinar qué roles serán los que asistan a ellas. Sobre los roles asistentes se podrá especificar si todos los miembros del equipo, con ese rol, serán automáticamente invitados o bien si sólo alguno de sus miembros.

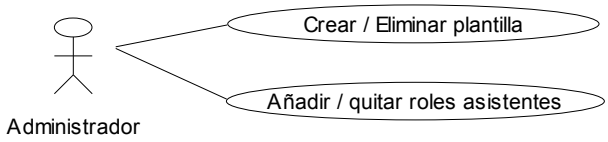

Figura 53. Casos de uso: Plantillas de reuniones.

Scrum/Reuniones/Reuniones Los administradores y los miembros con permiso de escritura podrán crear y eliminar reuniones, especificar sus contenidos, añadir y quitar asistentes, usar plantillas de convocatoria y enviar e-mails a los asistentes.

Por otro lado, los miembros con permiso de lectura sólo podrán consultar los datos referentes a la reunión, asistentes y contenidos, pero no podrán enviar convocatorias.

Finalmente, los miembros del equipo sin permisos podrán consultar los datos generales de la reunión y sus asistentes, pero no tendrán acceso a los contenidos de ésta.

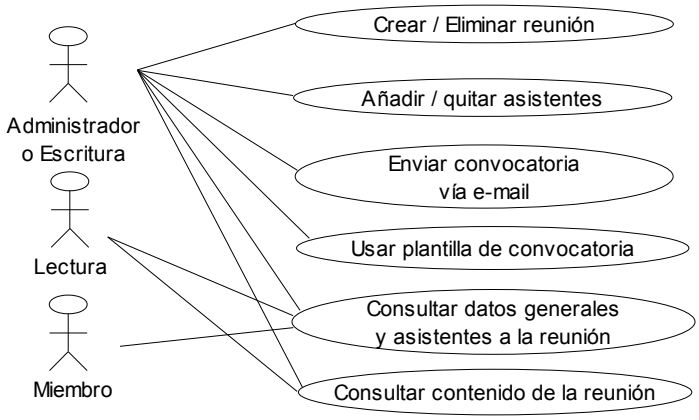

Figura 54. Casos de uso: Reuniones.

Scrum/Consulta global Todo usuario de la aplicación podrá hacer una consulta sobre un proyecto. En la consulta global se muestran datos pertenecientes a los diferentes módulos de la aplicación: Product Backlog, Sprint Backlog, Team & Role, Incidencias.

Es por ello que, dependiendo de los permisos que tenga el usuario en el equipo asignado al proyecto a consultar, podrá acceder a más o menos datos sobre el proyecto. Los administradores podrán consultar cualquier proyecto en su totalidad sin importar si pertenecen al equipo o si tienen permisos.

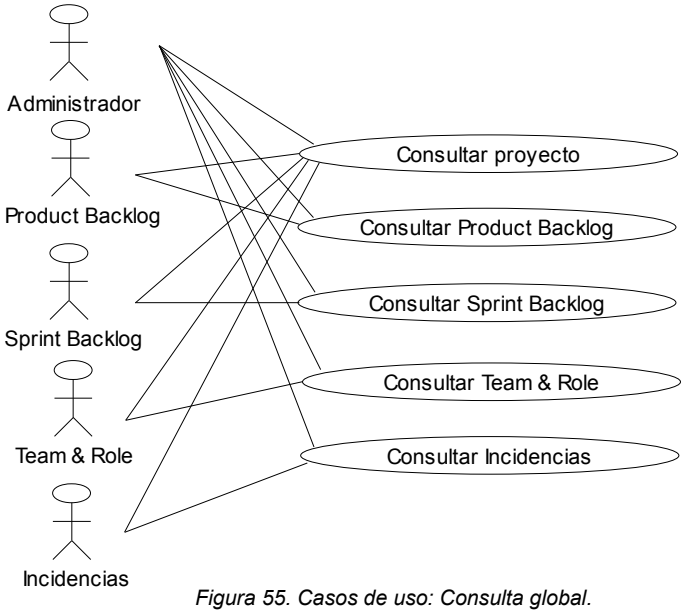

Figura 55. Casos de uso: Consulta global.

5.3. Estructura de paquetes de la aplicación.

Tal y como se especificó en los requerimientos no funcionales, se ha hecho uso de las herramientas karat para el desarrollo del proyecto. Al tratarse de varios proyectos donde el resultado debe ser un proyecto único, se ha aprovechado que las herramientas karat funcionan sobre Java para hacer una estructura de paquetes que integre los diferentes proyectos en uno:

com.unit4.gen.scrum Éste es el paquete principal donde se almacenará el proyecto en su totalidad.

.base En este paquete se encontrarán aquellas clases auxiliares para la implementación de las diferentes partes. Cada una se encontrará en su paquete correspondiente, quedando la estructura:

.backlog Clases adicionales que se hayan implementado para Product Backlog.

.sprint Clases adicionales que se hayan implementado para Sprint Backlog.

.incident Clases adicionales que se hayan implementado para Incidencias.

.teamrole Clases adicionales que se hayan implementado para este proyecto, Team & Role. Aquí se encontrarán, entre otras, las clases de control de permisos, que son usadas por el resto de proyectos.

.teamrole En este paquete se encontrarán todas las clases que hagan uso del framework karat. A su vez estarán separadas en dos paquetes.

.bo Donde se encontrarán las clases que añaden funcionalidades a los objetos de negocio.

.fm Donde se encontrarán las clases que añaden funcionalidades a los formularios.

Análogamente, el resto de proyectos tendrán la misma estructura y se situarán al mismo nivel.

.backlog Para el proyecto Product Backlog.

.sprint Para el proyecto Sprint Backlog.

.incident Para el proyecto Incidencias.

5.4. Resultados.

Una vez finalizado el proyecto se ha obtenido una aplicación funcional capaz de hacer la gestión de un proyecto que use la metodología Scrum.

5.4.1. Funcionalidades.

Los requisitos funcionales del proyecto se han llevado a cabo de la siguiente manera:

Creación y gestión de roles Los administradores serán capaces de crear roles y determinar los permisos de éstos.

Creación y gestión de colaboradores Los administradores podrán dar de alta colaboradores que ya existan en las bases de datos de ekon común.

Creación y gestión de aptitudes Los usuarios administradores podrán crear diferentes niveles de aptitud que podrán ser asignados a los colaboradores y determinarán su relación con los roles.

Creación y gestión de plantillas de equipos Los administradores podrán crear plantillas de equipo donde se especifique una estructura básica formada por un conjunto de roles y sus plazas.

Creación y gestión de equipos Los administradores podrán crear equipos y posteriormente añadir colaboradores a la plantilla de éstos. Una vez hayan sido añadidos colaboradores, aquellos con permisos de escritura o lectura sobre la gestión de personal podrán acceder a la gestión de equipos.

La creación de los equipos podrá tomar como base una plantilla. La composición especificada por la plantilla podrá ser modificada al gusto de quien cree el equipo.

Durante la asignación de colaboradores a un equipo, podrá hacerse una consulta que muestre las aptitudes de los colaboradores para el rol que se pretende asignar.

Creación y gestión de perfiles vacantes Los administradores y aquellos con permisos de escritura sobre la gestión de personal podrán crear y gestionar perfiles vacantes para los equipos. Estos perfiles servirán para indicar que el equipo busca colaboradores del rol especificado.

Además se podrá solicitar una petición de cambio a un miembro del equipo. De esta forma el colaborador aparecerá en una lista a la cual se puede acceder durante el proceso de añadir miembros a un equipo.

Creación y gestión de plantillas de reuniones Los administradores podrán crear plantillas de reuniones a partir de las cuales será posible especificar qué roles deben de asistir a éstas. Además podrá indicarse si deben asistir todos los miembros del equipo que realicen el rol o no.

Creación y gestión de reuniones Los administradores y aquellos con permiso de escritura sobre la gestión de personal podrán crear y gestionar reuniones. El resto de miembros del equipo sólo podrán consultar las reuniones ya existentes.

Las reuniones están lo suficientemente detalladas como para servir de historial, así se podrá consultar los temas tratados y los asistentes a ésta.

Por otro lado, cuando se convoque una reunión se podrá utilizar una plantilla. De esta forma se hará una convocatoria automática de todos aquellos miembros del equipo actual que cumplan los requisitos especificados en la plantilla, facilitando y agilizando la creación de reuniones.

Convocar reuniones Los administradores y aquellos con permiso de escritura sobre la gestión de personal, podrán indicar a la aplicación que envíe un e-mail a uno o a todos los miembros convocados a ésta. El e-mail enviado contendrá una tabla donde se indicará el día, lugar, hora y objetivos de la reunión, además de quién la ha convocado.

Navegación general sobre un proyecto Los administradores y todos aquellos miembros del equipo podrán realizar una consulta general sobre el proyecto. Dependiendo de los permisos de quien haga la consulta se mostrarán más o menos datos.

La consulta proporcionará una visión global del proyecto donde se podrán ver aquellos datos más relevantes de éste. Además será posible acceder a formularios más específicos de cada uno de los elementos mediante el uso de hipervínculos.

Control de permisos Dependiendo de si se es administrador y de qué permisos le otorga su rol en el equipo, el usuario tendrá acceso controlado a los diferentes módulos de la aplicación.

Para ello se ha implementado una clase que comprueba los permisos del usuario en el equipo actual. Además, se puede utilizar en cualquier módulo de la aplicación, logrando su propósito de controlar los permisos de toda la aplicación.

Para dar una muestra del software desarrollado, se incluyen unas capturas de pantalla de la funcionalidad de *navegación general*. En ellas se puede ver la estructura general de un formulario donde, en la parte superior, se encuentran aquellos valores que identifican lo que se está consultando y, en la parte inferior separada por pestañas, la información más relevante de cada módulo.

Gracias al formulario de consulta global, es posible tener una visión sobre un proyecto entero y su estado actual. Éste es especialmente útil para realizar las actualizaciones y consultas diarias, sin la necesidad de navegar por diferentes formularios. Si el usuario desea más información, puede hacer uso de los hipervínculos que identifican los diferentes elementos, pudiendo acceder a sus respectivos formularios:

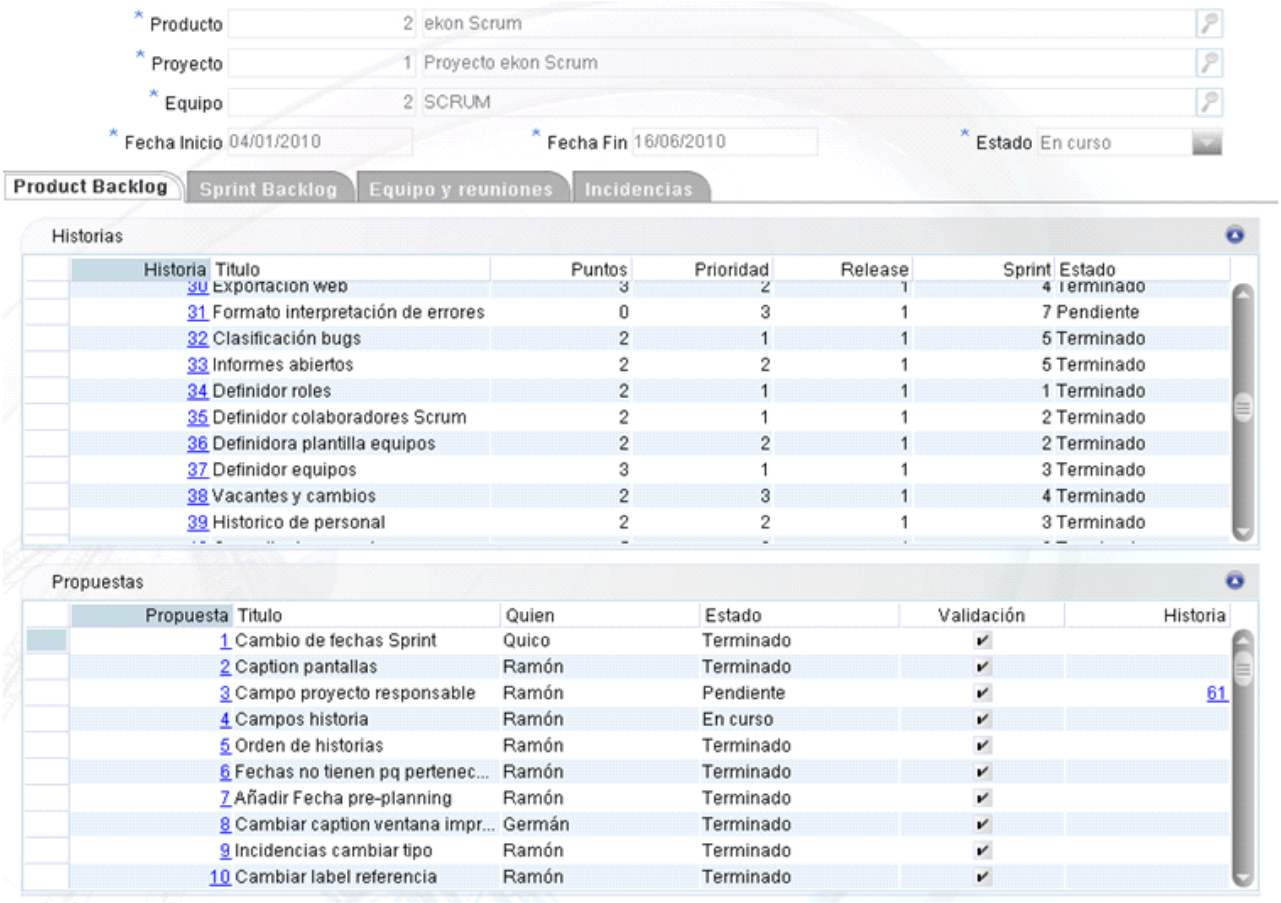

Figura 56. Consulta global: Product Backlog.

Mediante la pestaña de Product Backlog (figura 56), se pueden consultar las diferentes historias que lo conforman, así como las sugerencias que han surgido y su historia relacionada, si es que existe. Haciendo uso de hipervínculos, se podrá acceder a formularios donde se detallan las historias o propuestas, viendo quién las solicito y cuales son sus objetivos.

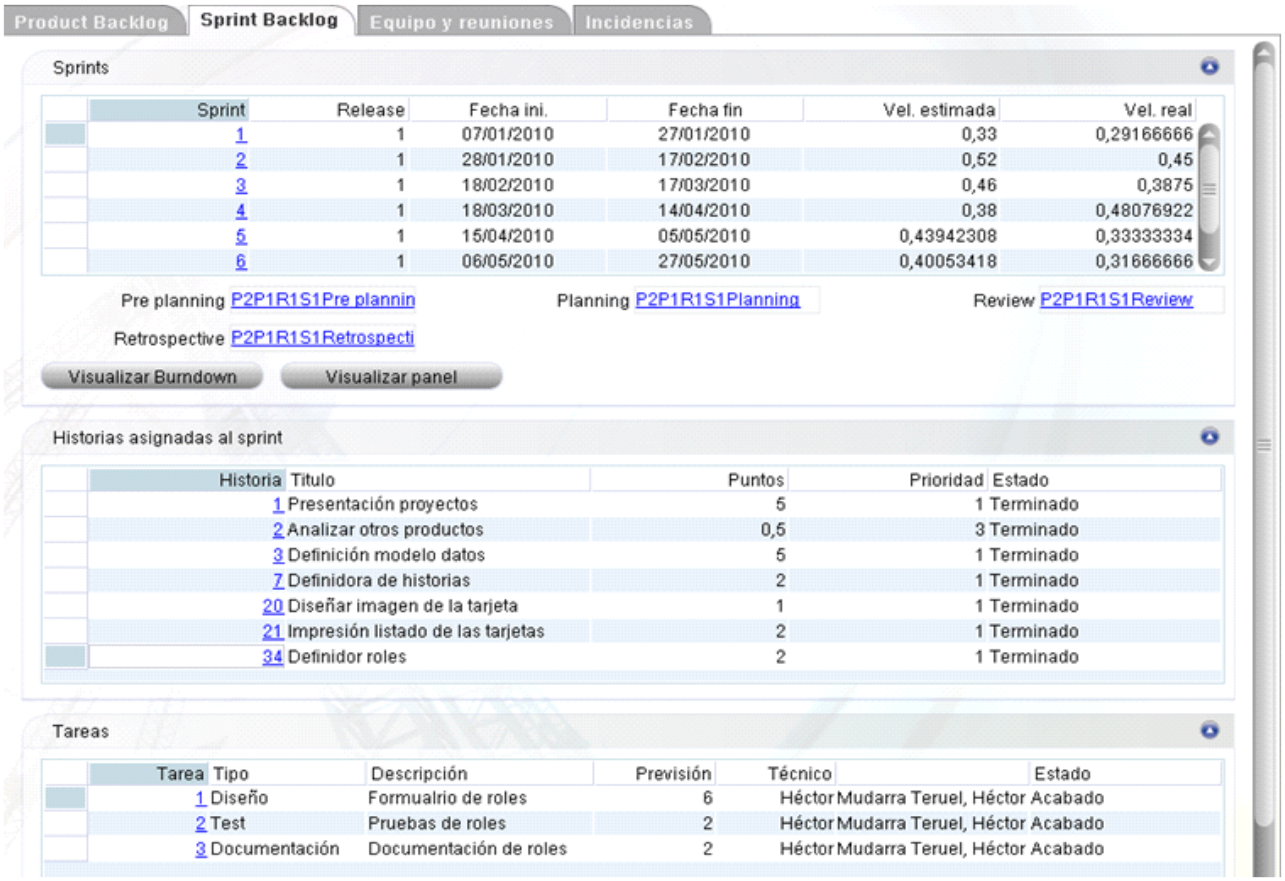

Figura 57. Consulta global: Sprint Backlog.

Mediante la pestaña de Sprint Backlog (figura 57), se pueden consultar los diferentes Sprint. En la primera lista se muestran los diferentes Sprint del proyecto, así como sus reuniones asociadas. Cuando uno de ellos esté seleccionado, se mostrarán en la segunda lista aquellas historias asignadas al Sprint. De la misma forma, cuando se seleccione una de las historias, en la tercera lista se mostrarán las diferentes tareas correspondientes a la historia seleccionada.

Haciendo uso de hipervínculos, se podrá acceder a la asignación de historias del Sprint, la información de las reuniones de éste, la asignación de tareas correspondiente a sus historias, y al seguimiento de sus tareas. Además, se incluye la posibilidad de acceder al panel virtual de seguimiento del Sprint y a su gráfica de Burndown.

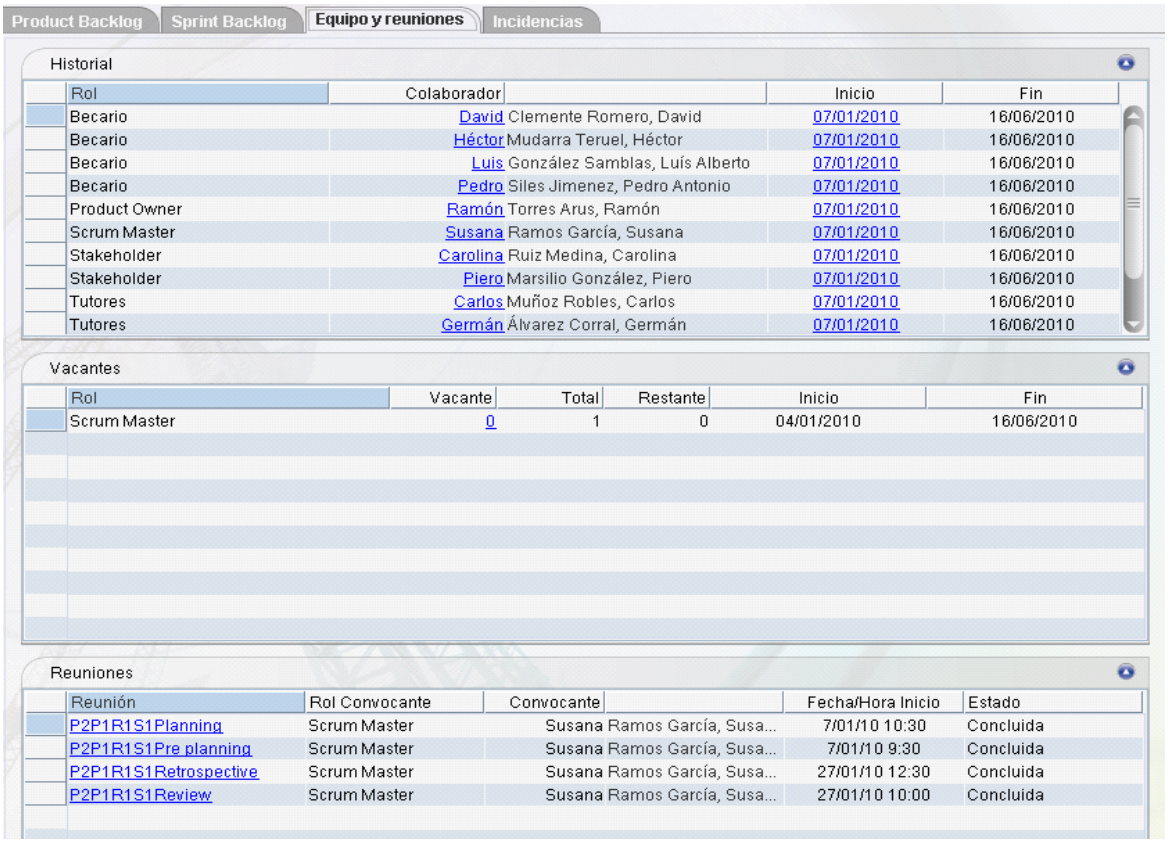

Figura 58. Consulta global: Team & Role.

Mediante la pestaña de equipos y reuniones (figura 58), se podrá consultar información del equipo y de las reuniones convocadas por éste. Sobre la información del equipo, se muestra el historial de colaboradores que ha tenido, así como el de vacantes. Además, se listarán todas las reuniones, tanto las que formen parte de los Sprint, como las que no. Al igual que en las anteriores pestañas, también será posible acceder a formularios más detallados para cada uno de los elementos mostrados.

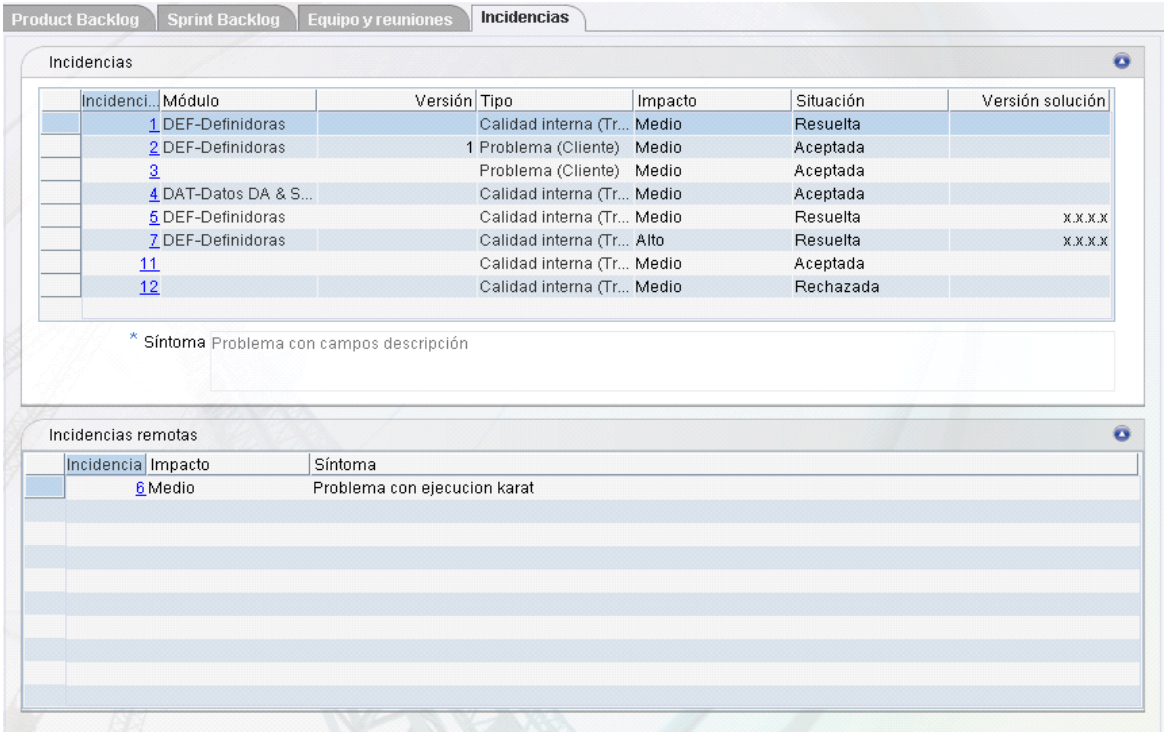

Figura 59. Consulta global: Incidencias.

Por último, mediante la pestaña de incidencias (figura 59), podrá consultarse el historial de incidencias surgidas durante el proyecto. La primera lista mostrará todas aquellas incidencias internas que hayan sido registradas por colaboradores de la empresa. La segunda lista mostrará todas las incidencias remotas que hayan sido registradas por clientes externos. Mediante los hipervínculos de esta pestaña, se podrá acceder al seguimiento de las incidencias internas, así como al control de las incidencias externas.

Sobre la consulta global, el usuario sólo verá el contenido de aquellas pestañas a las que su rol le permita acceder; protegiendo los datos del proyecto y permitiendo que cualquier miembro del equipo pueda acceder a la consulta.

5.4.2. Pruebas.

Para comprobar el correcto funcionamiento de la aplicación se ha sometido cada una de las historias implementadas a un conjunto de pruebas. El conjunto de pruebas se ha separado en dos categorías:

Cumplimiento de los requisitos En este caso se ha comprobado que la aplicación funcionase correctamente y como estaba previsto. Para ello se han tomado las diferentes funcionalidades que implementaba cada formulario y se ha probado su correcto funcionamiento.

Ya que el proyecto se ha realizado mediante la metodología Scrum, se ha usado al propio proyecto como información a introducir y con la que probar las funcionalidades.

Pruebas destructivas En este caso se ha sometido a la aplicación a prueba, examinando todos los casos extremos y obligando al código implementado a pasar por todas sus líneas, comprobando su correcto funcionamiento. Muchas de las pruebas de casos extremos ya están controladas por karat, es por eso que sólo se han probado sobre aquellas funcionalidades añadidas y que son controladas por el código realizado.

Para ello se han introducido valores con caracteres especiales en todos los campos donde era posible, cifras negativas donde sólo se considera una de positiva, se han eliminado datos antes y después de guardarlos en la base de datos para comprobar el correcto funcionamiento en todos los casos, se han eliminado registros de la base de datos que contenían claves externas, …

Además, durante el último Sprint, se ha distribuido la aplicación al equipo *Ready* de UNIT4, el cual también utiliza la metodología Scrum. Algunos de los miembros del equipo *Ready* han usado la aplicación y han probado todo aquello que han creído oportuno; tanto realizando pruebas destructivas como comprobando el cumplimiento de las funcionalidades especificadas.

Esto ha supuesto una gran ventaja ya que se ha podido comprobar cuál será el uso que se le dará a la aplicación por parte de los usuarios finales y, además, se han recibido sugerencias que han sido implementadas durante el último Sprint.

Capítulo 6. Conclusiones.

Para finalizar se hará una valoración del trabajo realizado durante el proyecto Team & Role y el proyecto global, empezando por los resultados obtenidos. A continuación se comentarán las ventajas de la metodología usada, así como del trabajo en equipo. Posteriormente se hará una valoración personal sobre la metodología y la experiencia adquirida. Para finalizar, se analizará la posible continuidad del proyecto.

6.1. Resultados.

Debido a la metodología Scrum, los requerimientos han ido variando durante el ciclo de vida del proyecto, siendo algunos de ellos modificados, otros eliminados y otros añadidos. Además, según la primera planificación del Sprint Backlog, el Product Backlog inicial quedaba completado al terminar el Sprint 5. Es por ello que no sólo han tenido cabida nuevos requerimientos, sino que también ha sido posible implementar múltiples sugerencias.

Referente a Team & Role, el único objetivo que no se ha podido llevar a cabo como estaba previsto ha sido el de *convocatoria de reuniones*. En él se requería que se pudiesen hacer convocatorias en el formato de *Microsoft Outlook*, pero para ello era necesario un servidor específico que mandase dichas peticiones. Por políticas de la empresa, sólo uno de los servidores puede enviar mensajes de correo, y ha sido por falta de acceso a éste que no se ha podido realizar. Es por ello que se habló con quien había propuesto la historia, el ProductOwner, y se especificó un nuevo formato. En éste se envía un e-mail con contenido HTML que incluye la información más relevante sobre la reunión.

En el capítulo 3 se han detallado los diferentes requerimientos de la aplicación, y en el capítulo 5 se ha expuesto la solución de éstos. Al comparar los dos capítulos, puede observarse que la planificación inicial

de los Sprint no se ha respetado en su totalidad. Sin embargo, eso no ha supuesto ningún inconveniente para el desarrollo del proyecto. Gracias a la auto-gestión de las iteraciones y a las gráficas de Burndown, ha sido posible re-planificar y añadir nuevos requerimientos, ya que siempre se ha valorado más la prioridad de éstos que el orden establecido en un inicio. Es por ello que se han logrado todos los requerimientos especificados, tanto los del inicio, como los incorporados a medida que avanzaba el proyecto.

Por otro lado, durante el diseño y desarrollo de los diferentes requerimientos, se ha mantenido una estructura independiente del resto de módulos y se ha hecho un diseño lo más general posible. Así, la gestión de equipos, roles y reuniones es abierta y sin restricciones. Además, gracias a la relación con ekon común, permite ampliar las funcionalidades de gestión de personal disponibles. Por último, el proyecto Team & Role es independiente del resto, permitiendo que pueda ser usado en otras metodologías.

El resultado final del proyecto conjunto ha sido una aplicación funcional, capaz de crear y gestionar todos aquellos elementos relacionados con la metodología Scrum. La nueva aplicación posee suficientes funcionalidades como para sustituir a los actuales métodos de gestión. Sin embargo, el uso de paneles no debería sustituirse por el implementado, ya que perdería su propiedad principal de servir de diario de trabajo.

6.2. Metodología.

En un principio, podía parecer incoherente el uso de una metodología de trabajo en equipo para la realización de cuatro proyectos separados, donde cada uno era asignado a un proyectista. Sin embargo, los cuatro proyectos estaban relacionados entre si y, aunque cada proyectista realizase el suyo, el trabajo en equipo y el conocimiento de los otros módulos era imprescindible para el desarrollo de las diferentes partes.

Por otro lado, una vez terminadas las funcionalidades principales de los proyectos, el resto de funcionalidades guardaban relación entre módulos, y éstas han sido desarrolladas conjuntamente por los responsables de los diferentes proyectos. Además, una vez se acercaba el proyecto a su fin, se ha contribuido con la carga de trabajo de aquellos proyectistas que aún tenían que desarrollar funcionalidades. Todo ello es debido al seguimiento de la filosofía de la metodología, donde el resultado final es el del equipo y no el individual.

En cuanto al progreso del desarrollo, el uso de iteraciones y reuniones ha proporcionado un conjunto de ventajas considerables respecto al modelo clásico. Terminada cada iteración era posible observar como la aplicación era completada y, a su vez, se conocía la opinión de los interesados y se recibían críticas y sugerencias. De esta forma, el proyecto ha gozado de dinamismo y ha ido evolucionando. Además, la auto-gestión del Sprint Backlog por parte del Team ha permitido adaptar la carga de trabajo a las condiciones del momento, priorizando aquellos requerimientos principales y desestimando aquellas sugerencias que se salían de los límites del proyecto.

6.3. Valoración personal

El hecho de haber realizado el proyecto en una empresa y que éste necesitase de un equipo de trabajo, ha servido para ver una situación real de implementación de proyectos en entornos profesionales. Se ha podido comprobar la dificultad que supone una correcta especificación de los requerimientos, las complejidades que conlleva la sincronización y el trabajo en equipo, la presión personal y de grupo que supone el acercarse a las fechas de entrega, … en definitiva, ha resultado muy provechoso como experiencia personal y de grupo para acercar al estudiante al entorno profesional.

Es por ello que, desde la experiencia, se recomienda el uso de la metodología para desarrollar proyectos siempre y cuando ésta sirva de ayuda. En caso de proyectos de poca densidad o duración, ésta no sería la metodología más adecuada ya que se invertiría demasiado tiempo en gestión de recursos y reuniones. Por otro lado, es posible que se quiera usar la metodología, pero el proyecto necesite una plantilla de personal que supere con creces el número de miembros recomendado. En ese caso, para aplicarla, convendría crear varios equipos relacionados entre ellos que, conjuntamente, realizasen el proyecto.

Un claro ejemplo de uso sería cuando el proyecto a realizar sea de gran densidad o se prevea que pueda tener variaciones. En estos casos, las iteraciones y las reuniones permitirán tanto que el proyecto evolucione, como que los requerimientos sean implementados sin causar un estrés excesivo en el equipo. Será entonces cuando una metodología de estas características haga que los resultados finales difieran completamente del método clásico

6.4. Líneas de futuro.

La aplicación se está empezando a usar por el equipo *Ready*, que está valorando las ventajas que le ofrece. También se ha presentado a los equipos de la sede de Granada, donde se encontraba uno de los Stakeholders, con el fin de extender su uso en las diferentes sedes de la empresa.

Pese a que la aplicación resultante es completamente funcional, se encuentra en el *release* 1.0 y, por tanto, sus funcionalidades pueden seguir siendo ampliadas y mejoradas a medida que se reciba *feedback* de los usuarios. A continuación se nombran algunas líneas de continuidad, tanto para la aplicación en general como para el módulo implementado:

Gestor de configuración La puesta a punto de la aplicación requiere de un proceso de configuración extenso, donde hay que crear un producto, proyecto, *release*, equipo, roles, tipos de tareas, tipos de incidencias... Todo ello podría ser mejorado mediante un proceso automatizado de configuración que agilice el procedimiento, tanto para la puesta a punto, como para la creación de nuevos proyectos.

Modo simplificado A medida que el proyecto ha ido avanzando, se han añadido más funcionalidades que han cargado visualmente la aplicación y han ampliado su complejidad de uso. Actualmente el único acceso simplificado que se ofrece es la consulta global, pero a través de ésta no se puede

acceder a la aplicación al completo. Convendría crear nuevas interfícies simples para facilitar el uso de la aplicación.

Nuevas metodologías Aprovechando la independencia de los módulos de Team & Role e Incidencias, es posible dar soporte a nuevas metodologías, reutilizando parte de las funcionalidades implementadas. De esta forma, la aplicación no sólo se limitaría a la gestión de Scrum, sino que gestionaría más metodologías, convirtiéndose en una aplicación de gestión de proyectos global.

Convocatoria de reuniones Pese a ser una funcionalidad no escalable, *Microsoft Outlook* es la herramienta usada en la empresa para gestionar convocatorias a reuniones. Es por ello que su inclusión en la aplicación agilizaría el proceso. Sin embargo, la funcionalidad implementada actualmente cumple con su propósito de anunciar la reunión, aunque no ofrece las ventajas de *Outlook*.

Glosario

Burndown Gráfica de seguimiento de la metodología Scrum. Puede representar el seguimiento global del proyecto o el seguimiento del Sprint.

Definidora (Nomenclatura de la empresa) Dicho de un proceso que permite crear y gestionar.

ekon Nombre con el que se bautizan las *suites* de gestión CRM y ERP desarrolladas con karat.

ekon común *Suite* de gestión y control de personal, empresas, colaboradores, países, etc. Es usada como base por el resto de *suites* ekon.

Historia elemento del Product Backlog que representa un requerimiento funcional o no funcional del producto.

karat herramientas de desarrollo que conforman un *framework* para la creación de software. Propiedad de UNIT4 ibérica.

Product Backlog Lista de requerimientos del producto, llamados historias. Ordenados por prioridad y con un valor cuantificado en puntos.

ProductOwner Rol de la metodología Scrum. Su papel es el de interactuar con el cliente. Es el encargado de crear y mantener el Product Backlog.

Punto Unidad utilizada en el Product Backlog para cuantificar el esfuerzo que supone el desarrollo de una historia.

***Ready*, equipo** Equipo de la empresa encargado del proyecto *Ready.* Fueron los primeros en usar Scrum en la sede de Barberá del Vallés.

Scrum Metodología de desarrollo de software ágil. Se compone de un seguido de iteraciones (Sprint) en las cuales se va desarrollando el producto. Su composición está definida por un conjunto de roles. Para llevar a cabo la metodología es necesario el uso de un conjunto de materiales específicos.

ScrumMaster Rol de la metodología Scrum. Su papel es el de velar por el correcto uso y ejecución del método. Además, se encarga de que el Team no tenga interrupciones externas y, a su vez, transmite la voz de éste al exterior.

Stakeholder Persona relacionada con un proyecto pero sin estar implicada en su desarrollo.

Sprint Iteración del método Scrum. Su duración puede variar pero es siempre fija durante una implantación del método. En ella se desarrolla el producto.

Sprint Backlog Lista de historias a realizar durante el Sprint. Divididas en tareas con un valor cuantificado en horas.

Tarea Subdivisión de una historia. El Sprint Backlog está compuesto de tareas y cada una de ellas tiene asignado un tiempo estimado de desarrollo. Indican y describen el trabajo a realizar.

Team Rol de la metodología Scrum. Encargados del desarrollo del producto. Está compuesto por un total de 7 ± 2 miembros y se trata de un equipo multidisciplinario y con capacidad de auto-gestión.

Bibliografía

[1] Takeuchi, Hirotaka; Nonaka, Ikujiro (enero-febrero 1986).
"The New New Product Development Game". Hardvard Business Review. Reprint 86116.

[2] Schwaber, Ken. (1995)
"Scrum Development Process". OOPSLA '95.

[3] Jeff Sutherland – LinkedIn, último acceso: junio 2010.
<http://www.linkedin.com/in/jeffsutherland>

[4] Scrum Alliance, último acceso: mayo 2010.
<http://www.scrumalliance.org>

[5] The home of Scrum, último acceso: noviembre 2009.
<http://www.scrum.org>

[6] Agile manifesto, último acceso: noviembre 2009.
<http://agilemanifesto.org>

[7] Wikipedia Scrum (development), último acceso: noviembre 2009
<http://en.wikipedia.org/wiki/Scrum_(development)>

Firmado: Héctor Mudarra Teruel

Bellaterra, Junio de 2010

Resum

Aquest projecte ha estat realitzat a una empresa a través d'un conveni entre la UAB i la seu UNIT4 de Barberà del Vallès. En ell s'estudia l'evolució de la implementació d'un projecte desenvolupat segons la metodologia Scrum. Per tal de realitzar el seguiment, l'autor ha participat a un equip de desenvolupament format per un conjunt de projectistes de la UAB. El projecte desenvolupat consta de cuatre mòduls que, en el seu conjunt, constitueixen una aplicació capaç de gestionar els recursos de la metodologia. Cada projectista és responsable d'un mòdul. En aquest document es recullen tant les parts relacionades amb el mòdul desenvolupat, com amb l'evolució del projecte.

Resumen

Este proyecto ha sido realizado en una empresa mediante un convenio entre la UAB y la sede UNIT4 de Barberá del Vallés. En él se estudia la evolución de la implementación de un proyecto desarrollado mediante la metodología Scrum. Para realizar el seguimiento, el autor ha participado en un equipo de desarrollo formado por un conjunto de proyectistas de la UAB. El proyecto desarrollado consta de cuatro módulos que, en su conjunto, constituyen una aplicación capaz de gestionar los recursos de la metodología. Cada proyectista es responsable de un módulo. En este documento se recogen tanto las partes relacionadas con el módulo desarrollado, como con la evolución del proyecto.

Abstract

This project has been done in a company through an agreement between the UAB and UNIT4's headquarters at Barbera del Valles. Within it there's a research about a project done using the Scrum development method. To meet this objective, the author has joined a team whose members are other UAB's final year students. The developed project has a modular design composed of four modules which, altogether, are meant to manage Scrum's resources. Each team's member supervises one module. In this report both module's implementation and project's development are discussed.